ON ITERATIVE DECODING OF HIGH-DENSITY PARITY-CHECK CODES USING EDGE-LOCAL COMPLEMENTATION

ON ITERATIVE DECODING OF HIGH-DENSITY PARITY-CHECK CODES USING EDGE-LOCAL COMPLEMENTATION

Joakim Grahl Knudsen

DISSERTATION FOR
THE DEGREE OF PHILOSOPHIAE DOCTOR

THE SELMER CENTER
DEPARTMENT OF INFORMATICS
UNIVERSITY OF BERGEN
NORWAY

AUGUST 2010

ABSTRACT

The overall topic of this work is a graph operation known as *edge-local complementation* (ELC) and its applications to iterative decoding of *classical* codes. Although these legacy codes are arguably not well-suited for graph-based decoding, they have other desirable properties resulting in much current research on the general problem of forging this alloy. From this perspective, these codes are typically referred to as *high-density parity-check codes.* Our approach is to gain diversity by means of ELC. Based on the known link between ELC and the information sets of a code, $\mathcal{C}$, we identify a one-to-one relationship between ELC operations and the automorphism group of a code, $\mathrm{Aut}(\mathcal{C})$. With respect to a specific parity-check matrix, H, we classify these code-preserving permutations into *trivial* and nontrivial permutations, based on whether the matrix is preserved (under ELC) up to row permutations, or not. The corresponding *iso-ELC* operations preserve the structure of the graph, and simulation data are presented on the performance benefit of using iso-ELC operations as a source of diversity. Generalizing this to random (noniso) ELC, we explore the benefits of a simplified, entirely graph-local (i.e., distributive) implementation of ELC-based decoding. Special codes are chosen, which are structurally well-suited for this type of random ELC decoding. At an extreme, certain codes are *ELC-preserved,* in that any ELC operation is an iso-ELC operation. Although less useful from a coding perspective, the corresponding graphs are interesting to determine and classify. However, in the general case, we observe negative effects of random ELC, which causes the weight of the graph to increase. Based on this, we explore the specific structural properties which determine which ELC operations (i.e., which edges of a graph) are desirable for ELC, from a decoding perspective. This is dominated by the weight increase (in terms of number of new edges) resulting from ELC, which also causes detrimental short cycles, so we identify the graphical conditions (i.e., subgraphs) for which ELC is *weight-bounding ELC* (WB-ELC). These operations are used to reduce the weight of a systematic H (given a code), and also used in a proposed decoder based on WB-ELC. At a slightly different approach, we also apply ELC operations in an *adaptive* decoding scheme. As ELC is a local operation, we can, to a certain extent, target the (inferred) least reliable positions specifically. This is a well-known technique to improve decoding, normally implemented via Gaussian elimination, which we improve by using beneficial properties of the ELC operation.

ACKNOWLEDGEMENTS

This research is an extension of the work in my Masters thesis, all under the supervision of Matthew G. Parker. His creative ideas, and often unconventional perspective, has been a steady source of research topics for me to pursue. When I'm now completing this thesis on ELC, some *pivotal* moments come to mind. During my first year, I was fortunate enough to work adjacent to Mikael Gidlund (currently at ABB, Västerås) who prompted me for my progress on a weekly basis, and persuaded me to present what I was working on (see Paper I) at Nera Networks, Bergen. This was to be my first pivot, which connected me to Constanza Riera and Lars Eirik Danielsen – both previous students of Matthew's – who were working on related topics and were interested in my applications of ELC to iterative decoding. The presentation at Nera revealed important issues with ELC in this context, which I could now discuss with Eirik Rosnes and Øyvind Ytrehus (my co-supervisors). The second pivot is Eirik, who would always take the time to help me when I was struggling with some theoretical or technical problems, and, most importantly perhaps, he has been endlessly patient. Special thanks to these people for your creative input, and for your enjoyable company!

Thanks also to fellow PhD students Mohammad Ravanbakhsh, Sondre Rønjom, Michal Hojsik, Inge Skjælaaen and Espen Vaular, for moral support and encouragement towards this common goal. Kudos to Kjetil Ottesen (Sonofotis) for drawing the excellent ELC animation. Those in charge of funds should be thanked for letting me travel to conferences in exotic places to present our work: Medina del Campo, Spain (2ICMCTA 2008); Seoul, Korea (ISIT 2009); Tel Aviv, Israel (HDPCC 2010); Zürich, Switzerland (IZS 2010), and as a participant on excellent conferences and research schools (ICT) hosted by the Selmer Center. I would also like to thank the administative staff at the Dept. of Informatics for assisting me with more practical things; mainly Ida Holen, Tor Bastiansen, Marta Lopez, and Torleif Kløve (former head of department). Also, Haakon Nilsen, Svein Harald Soleim, Halvor Utby, and the engineers at BCCS for saving my skin on numerous occasions (everything from C++/Linux problems, to undoing accidents via backup systems). I also want to thank the helpful people occupying IRC channels `#math` and `#c++` for sharing their time and advice (and dishing out the pepper when needed).

On a personal note, I am fortunate to be surrounded by loving family and friends. Love to Marthe, who will be my wife by the time this is published! Thankfully, we met when I was already a student, so by the start of my PhD studies she had learned to interpret and respond correctly to my frustrated ramblings at recessions in my studies. With the utmost sincerity: You make life fun and easy. At the same time I must thank my main man Tom F. Danielsen, with whom I have spent a decade of studies, for inspiration and always good advice. All our countless time spent with music, movies, or just hanging around is outside the scope of this work. Big hugs also to my mother, father, brother and grandmother for being there for me.

- Joakim Grahl Knudsen

Bergen, August 31 2010

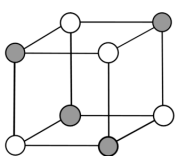
CONTENTS

Introduction

1 Information Theory

Information theory is the field of research centered on quantification of information. A mathematical theory of such *digital* representations was to a large extent defined in the groundbreaking work of Shannon in 1948 [1]. At the core lies a description of an optimal representation of a finite set of messages, which constitute some information measured in *bits.* This measure of information content is in terms of *entropy,* which can be thought of as the *surprise* involved in learning (observing or receiving) each of these messages. If a particular message is less likely than the others, then the entropy of this message is relatively high. This concept forms the basis for many fields of research, from efficient (and lossless) compression to reliable communications over a noisy *channel.* Such a channel can be any medium over which digital or analog information is conveyed. There are many examples, from the extremes of deep-space probes sending high-resolution images back to Earth, or communications satellites within our own atmosphere relaying information to the ground. At the other extreme, our everyday life is also highly dependant on reliable communications through noisy channels. For instance, music and movies may be read off an imperfect, perhaps even scratched compact disc, such that its content may be enjoyed through expensive speakers or high-definition television screens. This would not be possible without the groundbreaking work by Shannon, which gave birth to an entire science.

The insight of Shannon was that the disturbances due to the natural presence of noise could, at least in theory, be overcome by the use of *coding.* The compression mentioned above is one example, and is referred to as *source coding,* which optimizes the amount of data to be transmitted. To improve the noisy situation, however, a different type of coding is needed, known as *channel coding.* The idea is to introduce *redundancy* into the transmission [1]. In its simplest form, one may consider a simple repetition code, in which the information to be conveyed – e.g., a single number – is repeated several times. Say the receiver is a person at the other end of a busy street, where the noise is due to traffic. By keeping a list of what he thinks he has heard, the receiver will eventually – after a sufficient number of repetitions – have some number which has occurred more frequently than others, and which he may conclude to be the correct number. Although the error rate is still not zero, it can be made arbitrarily low by increasing the number of repetitions.



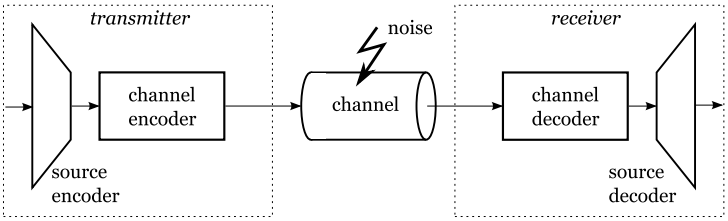


Fig. 1: *One-way communications scenario from a transmitter to a source, illustrating the problem of forward error correction.*

The disctinction between information and redundancy is apparent; only the single number of useful content is delivered, at the cost of the number of repetitions (the redundancy). This thesis is deals with the *forward* error correction problem, in which the transmitter can communicate to the receiver and there is no *feedback* link in the opposite direction. This situation is shown schematically in Fig. 1. The notion of redundancy has a double meaning in information theory. Consider the situation at the source; in order to attain efficient communications, the information to be transmitted is compressed so as to remove (ideally) all redundancy. However, as mentioned, specific, carefully constructed redundancy is then *added* to the compressed information before venturing a transmission. The specific design of such redundancy is an important research field in its own right.

1.1 Shannon Capacity

The *capacity* of a channel gives the maximum *rate* at which error-free communications can be achieved. In other words, how many bits of (source) information can be conveyed per channel use (bits actually transmitted). In the binary case, this rate, R, is a number less than 1.

Shannon described the source coding (compression) in terms of a minimum rate *needed* to describe the information content to be transmitted. Furthermore, he also proved the concept of a *maximum* possible rate for reliable transmission, in terms of the minimum amount of redundancy required to protect this information. This is dealt with in his source coding and channel coding theorems, respectively. This maximum rate is also known as the *capacity*, C, (also called the Shannon limit) of a channel, and is the maximum rate at which error-free communications can be guaranteed [1]. By error-free, it is understood that the probability

of error can be made arbitrarily low as long as $R < C$. Communication at a rate higher than C will, accordingly, incur an error probability bounded away from 0, regardless of blocklength. This is achieved by adding redundancy to the code, so in order to maintain the desired transmission rate, the blocklength must be increased accordingly. Shannon's channel theorem is asymptotic, and it is nonconstructive in the sense that Shannon did not conceive of any concrete code construction for which this could actually be performed – one of the most important research problem within this field of science.

1.2 The Basic Notation

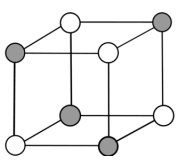
A (block) code, $\mathcal{C}$, is defined as the set of length-n strings (codewords) output of some encoding function. Our focus in this work is restricted to the binary case, where each codeword symbol is a *bit*, taking a value in the binary Galois field, denoted by $\mathrm{GF}(2) = \{0, 1\}$. Binary addition (modulo 2) is denoted by $\oplus$, and is sometimes referred to as an "XOR" operation. The encoding is a mapping from length-k to length-n vectors ($n > k$), where k bits are information and $n - k$ bits are redundant. As discussed above, the transmission rate (number of information bits conveyed per transmitted symbol) may now be defined as $R = k/n$. We will use uppercase italics for matrices, script notation and curly brackets for sets, and boldface notation for vectors.

A *linear* code is written $[n, k]$, and has the useful property that the binary sum of any two codewords is also a codeword. Furthermore, a linear code is conveniently represented using a basis for the space of codewords. Any set of k linearly independent codewords which form a basis for $\mathcal{C}$ is referred to as a *generator matrix*, G, of the code. In other words, a linear code is understood as the vector space generated by the rows of a $k \times n$ matrix, G

$$\mathcal{C} = \langle G \rangle \subset \mathrm{GF}(2)^n \tag{1}$$

where $\mathrm{GF}(2)^n$ is the space of all binary vectors of length n. In this work, we will focus exclusively on linear codes. The dimension of the code is $\dim(\mathcal{C}) = \dim(G) = k$, such that $|\mathcal{C}| = 2^k$. The corresponding null space of $\mathcal{C}$ is the vector space in which all vectors are orthogonal to $\mathcal{C}$. Thus, this is also a linear code, commonly referred to as the dual code of $\mathcal{C}$

$$\mathcal{C}^\perp = \{\mathbf{x}' \in \mathrm{GF}(2)^n \mid \mathbf{x} \cdot \mathbf{x}' = \mathbf{0} \; \forall \; \mathbf{x} \in \mathcal{C}\}. \tag{2}$$



This dual code has dimension $n-k$ (the co-dimension of $\mathcal{C}$), so it may be compactly represented by an $(n-k) \times n$ matrix, H. This also gives that $GH^T = \mathbf{0}$, so H is typically referred to as a *parity-check matrix* of $\mathcal{C}$. These matrices are said to be in *systematic* form if an identity matrix can be found among the columns. In particular, G is in *standard* form if the k rightmost columns comprise the identity matrix, I_k. The corresponding standard form parity-check matrix follows directly

$$G = [P \mid I_k] \;\Leftrightarrow\; H = [I_{n-k} \mid P^T] \tag{3}$$

where P^T is the transpose of P. The sets of *coordinates* (column positions) corresponding to the I and P parts of G are an *information set*, $\mathcal{I}$, and a *parity set*, $\mathcal{P}$, respectively. The matrices H and G are not unique for a code. For instance, by means of Gaussian elimination, an identity matrix (associated with an information set) can be placed in different subsets of columns of G. Each resulting matrix has a distinct P-part, and gives an associated systematic parity-check matrix. The number of distinct information sets is a property of the code. The weight of any parity-check matrix, H, for the code is lower-bounded by

$$|H|_{\min} \geq (n-k)\, d_{\mathbf{min}}(\mathcal{C}^{\perp}).$$

In systematic form we have

$$|H|^{\text{IP}}_{\min} \geq k(d_{\mathbf{min}} - 1) + n - k \tag{4}$$

and (3) gives

$$|G|^{\text{IP}}_{\min} = |H|^{\text{IP}}_{\min} + 2k - n.$$

In the following, even if H is not in systematic form, we will always assume that it has full rank, meaning that it contains $n-k$ linearily independent rows. If $\mathcal{C}^{\perp} = \mathcal{C}$, we say that the code is *self-dual*, and the above observations on G and H give that such a code is also always half-rate, $R = k/n = 1/2$.

The group of permutations, acting on the n columns of H, which preserve the code, is known as the automorphism group of the code, $\text{Aut}(\mathcal{C}) = \{\sigma : \sigma(\mathcal{C}) = \mathcal{C}\}$. The automorphism group of the code is the same as that for the dual code [2].

1.3 Channel Models

Any scenario in which content is to be transmitted from one place to another should be thought of as a channel. Moreover, channels exist over

which information is transferred not in space, but (conceptually) in time, e.g., where content is written to a storage medium, and later read off using some imperfect mechanism for reading and writing. As we have mentioned, in any channel there is a natural presence of disturbance or noise. Several models exist which attempt to describe the behaviour of such real-life channels. These models are necessarily simplified mathematical constructions, which focus on certain parameters of the actual channel noise.

In this work, our focus is restricted to the case of binary transmission, where the information to be transmitted is completely described by the binary alphabet, denoted by GF(2). All channels considered also share the property of being *symmetric* on the output which means that, from the perspective of the receiver, the two possible origins of a received bit, y_i, are equally likely. Then, the channel transition probability, p, is

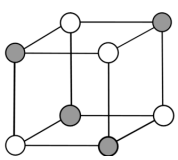
$$p := \Pr(y_i \mid x_i = 0) \;=\; \Pr(y_i \mid x_i = 1). \tag{5}$$

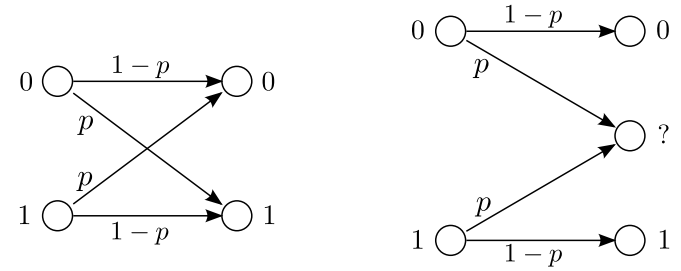
Another important general assumption on binary-input output-symmetric channels is that the noise affects each bit independently. These are referred to as *memoryless* channels, for which the (blockwise) transition probability may be written

$$\Pr(\mathbf{y} \mid \mathbf{x}) = \prod_{i=0}^{n-1} \Pr(y_i \mid x_i).$$

The simplest channel model is the binary symmetric channel (BSC), in which a bit is *flipped* with some fixed transition probability – see Fig. 2(a). A general assumption is that $p < 1/2$. If, otherwise, a majority of the bits are received in error, it would be more useful to simply flip *all* received bits and simply consider the transition probability to be $1 - p$. For the BSC, the capacity is known to be $C_{\text{BSC}}(p) = 1 - H(p)$, where $H(p) = \text{-}p \log_2 p - (1-p)\log_2(1-p)$ is the binary entropy function.

Of conceptual importance, we also describe the binary erasure channel (BEC) [3] illustrated in Fig. 2(b). This channel differs significantly from the other channels described – and from most channels in general – in that the channel makes no mistakes; a received value of 0 or 1 is known to be correct. However, a different type of disturbance characterizes this channel, in that a value is *erased* with probability p, upon which all information on that particular value is lost to the receiver. As a consequence, no mistakes can be amplified or introduced by a





(a) Binary Symmetric Channel (b) Binary Erasure Channel

Fig. 2: *Binary-input discrete output-symmetric channel models.*

decoding algorithm; if a symbol is corrected (i.e., the value of an erasure is determined) then this must be the correct value, and is never subsequently changed. The channel parameter is the erasure probability, $p = \Pr(y = ?)$ (assuming symmetry).

This channel was devised by Elias in 1955 as a simplified, theoretical channel model [3]. And, as we will see, the BEC has indeed led to significant observations which, to a certain extent, are transferrable to more general channel models. Also, with the arrival of the Internet, the BEC has come to model a real-life channel, where information is lost in *packets*, which is a detectable event.[1] It is known that certain code-specific properties affecting the performance of a code on the BEC also influence the performance on more general channels. The capacity of the BEC is $C_{\text{BEC}}(p) = 1 - p$.

The most important channel model for this work is the additive white Gaussian noise (AWGN) channel. This channel differs from the BSC and the BEC mainly in that the output is real-valued, and that the noise is *additive*. The additive noise has a (zero mean) Gaussian distribution, written $\mathcal{N}(0, \eta^2)$, where the channel parameter describing the noise level is the variance, $\eta^2 = N_0/2$. Here, $N_0 = 2\eta^2$ is the double-sided power spectral density [4]

The AWGN channel uses a *modulation* scheme to map the input to peaks of a waveform in such a way as to maximally separate these peaks, making the signal more robust against channel noise. Binary phase shift

[1] Although, in pactice and especially on a wired network, packets are not strictly independent.

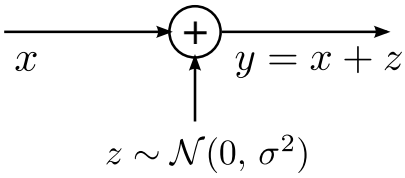


Fig. 3: *Binary-input additive white Gaussian noise channel.*

keying (BPSK) maps 0 and 1 to $\sqrt{E_s}$ and -$\sqrt{E_s}$, respectively, where E_s is the energy (amplitude) used in transmission. It is standard to use unit energy, $E_s = 1$. This gives mean $\mu = \pm\sqrt{E_s}$ for the corresponding Gaussian probability density function (pdf)

$$f(y;\, \eta^2, \mu) \triangleq \frac{1}{\sqrt{2\pi\eta^2}}\, \mathrm{e}^{-(y-\mu)^2/2\eta^2}. \tag{6}$$

In a coded transmission, in which only a fraction $R = k/n$ of the bits convey information (as opposed to redundancy), the energy per information bit is denoted by $E_b = E_s/R$. We will refer to the *signal-to-noise ratio* (SNR) as E_b/N_0, where all reported values (e.g., on error-rate plots) are in decibels (dB), $10\log_{10}(E_b/N_0)$.

The received signal must be *demodulated* using the distribution (6) of the noise [5], giving likelihood densities (as shown in Fig. 4)

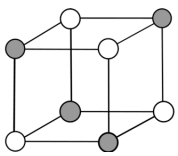
$$\Pr(y_i \mid x_i = 1) = f(y_i;\, N_0/2,\, -\sqrt{E_s}) = \frac{1}{\sqrt{\pi N_0}}\, \mathrm{e}^{-(y_i+\sqrt{E_s})^2/N_0},$$

and, symmetrically,

$$\Pr(y_i \mid x_i = 0) = f(y_i;\, N_0/2,\, \sqrt{E_s}) = \frac{1}{\sqrt{\pi N_0}}\, \mathrm{e}^{-(y_i-\sqrt{E_s})^2/N_0}.$$

Alternatively, this may be converted to a binary output by *quantizing* the real output to the nearest constellation point (i.e., $\pm\sqrt{E_s}$ for BPSK), denoted by $[y_i < 0]$. This expression evaluates to 1 *iff* the argument (inside the brackets) is true [6].

As illustrated in Fig. 4, $\Pr(y_i \mid x_i)$ is typically interpreted as the channel likelihood, although the probability of any specific y_i is by definition 0 (since f is a pdf). In the case of antipodal modulation (as with BPSK, with unit energy), the pdf's for (source symbols) '0' and '1' are completely symmetrical and equally spaced from energy 0. By



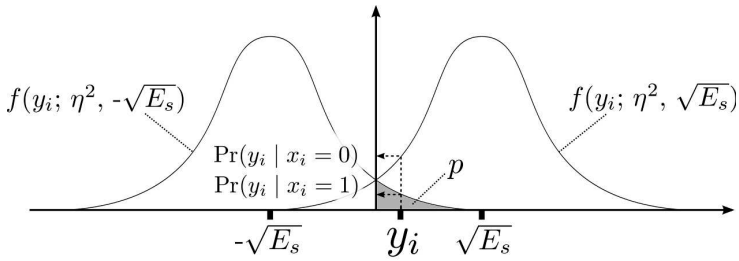


Fig. 4: *Demodulating channel output, y_i, to get channel likelihood densities.*

"reading off" the pdf's, we see that the density $\Pr(y_i \mid x_i = 0)$ will be greater than the alternative $\Pr(y_i \mid x_i = 1)$ so long as $y_i > 0$. The opposite holds for $y_i < 0$. Thus, a hard decision (quantization) can be made by simply comparing the received channel value against 0. The total probability of receiving $y \in [a,\, b]$ corresponds to the area under the pdf in that same interval, $\int_a^b f(y;\, \eta^2,\, \mu)\, \mathrm{d}y$. This gives a transition probability, which relates the AWGN channel to the BSC when using BPSK [7]

$$\begin{aligned} p &= \Pr(y < 0 \mid x = 0) = \Pr(y > 0 \mid x = 1) \\ &= \int_0^\infty f(y;\, N_0/2,\, -\sqrt{E_s})\, \mathrm{d}y = Q(\sqrt{2E_s/N_0}) \end{aligned}$$

as illustrated in Fig. 4. The complementary error function, or Q-function, is

$$Q(y) \triangleq \frac{1}{\sqrt{2\pi}} \int_y^\infty \mathrm{e}^{-u^2/2}\, \mathrm{d}u. \tag{7}$$

More robust communications (in terms of noise tolerance) is achieved by either increasing μ or decreasing η^2, either of which reduce the intersecting area by moving the pdf's further apart and/or by increasing the amplitude (making the curves taller and more narrow).

The capacity of the binary-input, continuous-output AWGN channel can be computed numerically as

$$C_{\text{AWGN}} = 1 - \int_{-\infty}^{\infty} f(y;\, \eta^2,\, \mu)\ \log_2(1 + \mathrm{e}^{-y})\, \mathrm{d}y.$$

The AWGN channel is among the most prevalent channel model used in simulating a code and/or decoder, since additive noise is found to be a constituent part of most real-life channels. Especially, deep-space

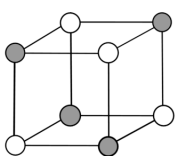
communications is described quite accurately by an additive noise model, whereas everyday wireless communications, on the other hand, is affected by a much more complex disturbance pattern, also including fading and interference with other communications.

2 Classical Coding Theory

The term *classical coding theory* is a reference to the early work of coding theory, which formed the foundations of the field as it is today. The term is in a sense a misnomer, in suggesting that the results and techniques are outdated – not *modern* – when, in truth, the classical concepts are still active research fields. Perhaps a better interpretation of the term would be a *classical view* on coding theory, as this view has undergone a significant change recently. The classical view on coding theory can be summarized as taking a *code perspective.* Intuitively, the early work in the field would be inspired by Shannon, trying to find specific, practical code constructions promised by his famous theorems.

2.1 Hamming Distance

Among the first codes invented were the results of Hamming's efforts to cope with errors and faulty circuitry in early computers in the 1950s [8]. By examining current error control measures, such as error detection via a parity bit or correction via simple repetition codes, Hamming's codes were designed with several overlapping parity bits, in order to increase – and even maximize – the difference (in terms of bits) between codewords. The number of bits which differ between two codewords became known as the Hamming distance, where the minimum Hamming distance between any pairs of distinct words in a code is a concrete measure on the capabilities of the code. By increasing the minimum distance, stronger codes result in which more noise is tolerated before a codeword is, eventually, confused with another codeword. A code of minimum distance $d_{\mathbf{min}}$ can *detect* any pattern of up to $t_d = d_{\mathbf{min}} - 1$ errors (the corresponding noisy vectors are distinguishable from valid codewords), and *correct* (i.e., detect *and* locate) up to $t = \lfloor (d_{\mathbf{min}} - 1)/2 \rfloor$ errors. (The same code can correct $d_{\mathbf{min}} - 1$ erasures on the BEC.) We sometimes extend the notation for a linear code to $[n, k, d_{\mathbf{min}}]$. The



Hamming distance between two binary vectors equals the weight of the binary sum (XOR) of the vectors,

$$\mathrm{d}_H(\mathbf{x}, \mathbf{x}') = |\mathbf{x} \oplus \mathbf{x}'|. \tag{8}$$

In order to facilitate error correction, Hamming used the concept of the *syndrome* of a vector. The parity-check matrix of a linear code spans the null space of the code (or the dual code), in that, for any $\mathbf{x} \in \mathcal{C}$, $\mathbf{x}H^T = 0$. H generates the dual code, $\mathcal{C}^\perp$, consisting of codewords orthogonal to $\mathcal{C}$. The result of such a code membership test for any length-n binary vector, $\mathbf{y}$

$$\mathbf{s}(\mathbf{y}) := \mathbf{y}H^T \tag{9}$$

is referred to as the *syndrome* of $\mathbf{y}$, as a nonzero syndrome indicates that something is wrong with this vector (in terms of being a codeword in $\mathcal{C}$).

By constructing H from all 2^m binary tuples of length $m = n - k$ (except the all-zero vector), Hamming ensured that any pair of columns in H are pairwise linearly independent. As such, no sum of two columns could give the zero syndrome, giving a code of minimum distance 3. This gives $t = 1$, and since all non-zero syndromes are present (as columns) in H, any single error can be corrected by flipping the bit corresponding to the column in H matching the syndrome, alternatively declaring a decoder failure (more than a single error must have occurred). By convention, column i, $0 \leq i < n$, of H is the binary representation of the number $i + 1$. The specific construction of Hamming codes means that these are all $[2^m - 1, 2^m - 1 - m, 3]$ codes, where $m \geq 3$. In this sense, the Hamming codes are *fixed-minimum distance* codes.

The syndrome (9) of a received noisy vector, $\mathbf{y} = \mathbf{x}_s + \mathbf{e}$, (assuming additive noise) can be written

$$\mathbf{s}(\mathbf{y}) = H\mathbf{y}^T = H(\mathbf{x}_s + \mathbf{e})^T = H\mathbf{x}_s^T + H\mathbf{e}^T. \tag{10}$$

Since $H\mathbf{x}_s^T = \mathbf{0}$, the syndrome is $\mathbf{s}(\mathbf{y}) = H\mathbf{e}^T$. Thus, a lookup table can be made based on the syndrome of all possible correctable error patterns, $H\mathbf{e}^T$, and the corresponding error pattern, $\mathbf{e}$. The Hamming codes (in the column ordering described above) comprise a special case, where H is itself the lookup table, identifying all error patterns of weight $t = 1$. Although computing the syndrome lookup table may be

complex to compute (it is nontrivial to find the proper coset leaders), this is mainly a preprocessing cost.

2.2 Maximum Likelihood Decoding and Union Bound

An *optimum* decoder outputs the codeword which minimizes the *a posteriori* probability (APP) (i.e., the probability of word error) given the received vector, $\Pr(\mathbf{x} \neq \mathbf{x}_s \mid \mathbf{y})$. Equivalently, this can be implemented by *maximizing* $\Pr(\mathbf{x} = \mathbf{x}_s \mid \mathbf{y})$. Using Bayes' rule

$$\overbrace{\Pr(\mathbf{x} = \mathbf{x}_s \mid \mathbf{y})}^{\text{posterior (APP)}} = \frac{\overbrace{\Pr(\mathbf{y} \mid \mathbf{x} = \mathbf{x}_s)}^{\text{likelihood}} \overbrace{\Pr(\mathbf{x} = \mathbf{x}_s)}^{\text{prior}}}{\Pr(\mathbf{y})} \propto \Pr(\mathbf{y} \mid \mathbf{x} = \mathbf{x}_s). \quad (11)$$

The proportionality between likelihood and posterior follows from the simplifying assumption that all codewords, $\mathbf{x}_s \in \mathcal{C}$, have equal prior probability, $\Pr(\mathbf{x} = \mathbf{x}_s) = 1/|\mathcal{C}|$, and that the normalizing constant, $\Pr(\mathbf{y})$, is independent of $\mathbf{x}$. From a vector perspective, one typically considers the (prior) probability of the source outputing any specific codeword to be the same for all codewords. This is referred to as the assumption of *uniform priors*. The optimum *maximum a posteriori probability* (MAP) decoder outputs the codeword for which the associated APP is maximized

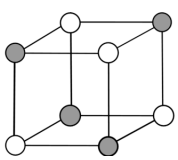
$$\mathbf{x}^{\text{MAP}} := \arg\max_{\mathbf{x} \in \mathcal{C}} \Pr(\mathbf{x} = \mathbf{x}_s \mid \mathbf{y}) = \arg\max_{\mathbf{x} \in \mathcal{C}} \Pr(\mathbf{y} \mid \mathbf{x} = \mathbf{x}_s) =: \mathbf{x}^{\text{ML}} \quad (12)$$

which coincides with the output of from the optimum *maximum likelihood decoder* (MLD). This equality between $\mathbf{x}^{\text{MAP}}$ and $\mathbf{x}^{\text{ML}}$ follows from the assumption of uniform priors [4].

Consider the BSC(p) channel. Since we assume that $p < 1/2$ (that a bit is more likely to be received correctly, than in error), (12) can be maximized in terms of Hamming distance (8)

$$\Pr(\mathbf{y} \mid \mathbf{x} = \mathbf{x}_s) = \mathrm{d}_H(\mathbf{x}, \mathbf{y})p + (n - \mathrm{d}_H(\mathbf{x}, \mathbf{y}))(1 - p). \quad (13)$$

Still, this does not guarantee that $\mathbf{x}^{\text{ML}} = \mathbf{x}_s$. When no more than $t = \lfloor (d_{\mathbf{min}} - 1)/2 \rfloor$ bits are in error on a vector, this approach is guaranteed to succeed (correctly determine $\mathbf{x}_s$). This can be viewed as *spheres* of radius t centered on each codeword, which then contain no other codewords. Any received vector which arrives within such a sphere,



is then decoded (correctly) to the corresponding codeword. Any algorithm which achieves this is known as a *bounded distance decoder* (BDD). In this definition, we assume the BDD is configured to decode up to its maximum capability of t errors.

In comparison, MLD will correctly identify some (but not all) vectors at a distance greater than t. The performance of the (perhaps impractical) MLD can be estimated by considering not only the minimum distance of the code, but the weight of all nonzero codewords in $\mathcal{C}$. The probability of the optimum decoder making an error (outputing a different codeword than what was sent) can be bounded using a *union bound.* Consider the specific event $\{\mathbf{x}^{\mathrm{ML}} \mid \mathbf{x}_s\}$, of sending $\mathbf{x}_s$ and receiving $\mathbf{y}$, which is decoded to $\mathbf{x}^{\mathrm{ML}} \neq \mathbf{x}_s$. This event represents a simplified code containing only the two codewords $\mathbf{x}_s$ and $\mathbf{x}^{\mathrm{ML}}$. The likelihood of this event may be expressed in terms of the distance between $\mathbf{x}^{\mathrm{ML}}$ and $\mathbf{x}_s$. In a linear code, we may assume $\mathbf{x}_s = \mathbf{0}$. The overall error probability (of the entire code) can then be expressed as the probability of the union of the events, corresponding to the possible decoder errors

$$\Pr(\mathbf{x}^{\mathrm{ML}} \neq \mathbf{0} \mid \mathbf{y}) = \Pr\Big(\bigcup_{\mathbf{x}\in\mathcal{C}, \mathbf{x}\neq\mathbf{0}} \{\mathbf{x} \mid \mathbf{0}\}\Big) \leq \sum_{\mathbf{x}\in\mathcal{C}, \mathbf{x}\neq\mathbf{0}} \Pr(\{\mathbf{x} \mid \mathbf{0}\}) \quad (14)$$

which is upper bounded by the corresponding *sum.* In the general case, the constituent events are not disjoint.

The *weight enumerator, A*, of a code counts the codewords grouped by weight, such that A_w contains the number of codewords of weight w. On the BSC(p) channel, (14) can be expressed as

$$\Pr\Big(\bigcup_{\mathbf{x}\neq\mathbf{0}\in\mathcal{C}} \{\mathbf{x} \mid \mathbf{0}\}\Big) \leq \sum_{w\geq d_{\mathbf{min}}} A_w \sum_{i=\lceil w/2\rceil}^{w} \binom{w}{i} p^i (1-p)^{w-i}.$$

On a channel with unquantized output, a soft distance measure can be in terms of squared Euclidean distance. For the AWGN channel, the union bound is expressed in terms of the Q-function (7),

$$\Pr\Big(\bigcup_{\mathbf{x}\neq\mathbf{0}\in\mathcal{C}} \{\mathbf{x} \mid \mathbf{0}\}\Big) \leq \sum_{w\geq d_{\mathbf{min}}} A_w \, Q\left(\sqrt{2wR\,E_b/N_0}\right)$$

where R is the rate of the code, and $R\,E_b/N_0$ is the rate-normalized SNR [7].

2.3 Some Classical Codes

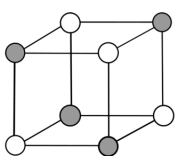
Typically, decoding algorithms for classical codes would have to be tailored for the specific design of the code, and general low-complexity

algorithms are indeed rare in the literature. The popularity and success of a classical code construction typically lies in the combination of attractive code parameters (most commonly, minimum distance), and the description of a practical decoding algorithm. In this section, we will briefly describe some of the most popular classical code constructions in use today.

Binary quadratic residue (QR) codes are cyclic codes of prime block-length, n, such that 2 is a quadratic residue modulo n [9]. The rate and minimum distance of these $[n, (n \pm 1)/2, d_{\mathbf{min}} \geq \lceil \sqrt{n} \rceil]$ codes are bounded away from 0, as n increases. In this work, we consider only the QR codes of dimension $(n+1)/2$. By adding an extra parity bit, we obtain an *extended* QR (EQR) code of parameters $[n+1, (n+1)/2, d_{\mathbf{min}} + 1]$, giving a code which is half-rate and self-dual (or equivalent to its dual). Furthermore, for certain parameters, the structure of $\mathrm{Aut}(\mathcal{C})$ of EQR codes is known, such that three generators may be calculated [9].

Well known classical codes, such as the [7, 4, 3] Hamming code and the [23, 12, 7] Golay code (and their extensions), are examples of QR codes. These particular codes (and, in fact, all Hamming codes) have very special properties which make them well-suited to test various concepts within graph and coding theory. For instance, they are *perfect* codes in that the codewords are spaced by a radius of exactly t to neighboring codewords, meeting the *sphere-packing bound,* also called the Hamming bound, with equality. In other words, there are no gaps between the spheres spanned out around each codeword, so – regardless of noise level – there is always exactly one codeword (though not necessarily the transmitted one) within distance t of the received vector. These codes are also *optimal,* in the sense that they have the maximum minimum distance possible, for a fixed rate (k and n). Furthermore, we will see later that these codes also have strong *graph-structural* properties.

Binary Bose-Chaudhuri-Hocquenghem (BCH) codes is a class of codes which is a t-error correcting generalization of the Hamming codes, of parameters $[2^m - 1, k, d_{\mathbf{min}}]$, for $m \geq 3$ [7]. The resulting dimension, k, depends on the construction, $n - k \leq mr$, where $r < 2^{m-1}$ and $d_{\mathbf{min}} \geq 2r + 1$. The construction of these codes is based on extended Galois fields, to using primitive (irreducible) polynomials. Codewords are viewed as polynomials over this field, and encoding performed via polynomial division (the codeword is the remainder). Polynomials may



be represented by binary length-n sequences, so the codes are binary codes. These codes are *cyclic* codes, in that any shift (with wrap at the boundary) of a codeword is also a codeword.

BCH codes are designed for random noise, and decoded using syndrome decoding where the procedure described for Hamming codes is implemented in terms of polynomials, $y(z) = y_o + y_1 z + \cdots + y_{n-1} z_{n-1} = x(z) + e(z)$. The syndrome polynomial, $s(z)$, then identifies the error location(s) allowing correction (assuming no more than t errors have occurred) via an *error-location polynomial.* BCH codes can be decoded efficiently using the Berlekamp-Massey (BM) algorithm [7].

A special type of (nonbinary) BCH code designed to handle bursts of noise is the Reed-Solomon (RS) code [7]. These codes are generalized to the nonbinary case, where each *symbol* of the code is over GF(2^q). Mapped to a binary code, each symbol is represented by q bits. As such, these codes are very suitable for correcting *burst errors,* where many consecutive bits are in error, since a symbol-level decoder will correct t symbols (i.e., qt bit errors). Such errors are frequent on compact discs, where a scratch or a fingerprint affects many bits in one physical area of the disc which is otherwise error-free.

The parameters of RS codes are $[n, k, n-k+1]$. This is the maximum possible minimum distance (Singleton bound), which means that the codes are *optimal.* As such, they can correct up to $t = \lfloor (n-k)/2 \rfloor$ symbol errors, i.e. a maximum of $\lfloor q(n-k)/2 \rfloor$ bit errors. A *binary image* is a $[qn, qk, d_{\mathbf{min}}]$ code, where the resulting minimum distance depends on the particular representation of the field symbols as q-dimensional binary vectors (i.e., the mapping). Being BCH codes, these codes are also well-suited for BM decoding.

Furthermore, in this thesis, we consider some general extremal, self-dual codes described by Harada and Gulliver [10, 11], which have trivial or small ($|\mathrm{Aut}(\mathcal{C})| \approx n$) automorphism group.

2.4 Permutation Decoding

We will now expand the discussion on syndrome decoding introduced with the Hamming codes. Certain properties of the syndrome can be used to devise a more powerful algebraic decoding algorithm, known as permutation decoding (PD) [9]. This algorithm may be applied to any code for which the *structure* supports a nontrivial $\mathrm{Aut}(\mathcal{C})$. The

main idea is to avoid the lookup table, and rather focus entirely on the syndrome. Rather than locating the error positions, the concept is to produce an error-free information set, such that the output codeword is computed by simply *re-encoding* the parity bits.

The premise for this technique is that H is in systematic form, where the I-part (submatrix) of H corresponds to the columns indexed by the parity set $\mathcal{P}$. Let $\mathbf{y}_I$ and $\mathbf{y}_P$ be the subvectors of $\mathbf{y}$ indexed by $\mathcal{P}$ and $\mathcal{I}$, respectively. Then, the syndrome (10) can be split into two parts

$$\mathbf{s}(\mathbf{y}) = H\mathbf{y}^T = I\mathbf{y}_I^T + P\mathbf{y}_P^T = \mathbf{y}_I^T + P\mathbf{y}_P^T \tag{15}$$

where the identity submatrix simplifies the expression. If there is no noise ($\mathbf{e} = \mathbf{0}$), we receive $\mathbf{y} = \mathbf{x}$ which is detected by the zero syndrome. Thus, for $\mathbf{y} \in \mathcal{C}$, (15) reduces to

$$\mathbf{y}_I^T = P\mathbf{y}_P^T \tag{16}$$

since $\mathbf{s}(\mathbf{y}) = \mathbf{0}$. An error in position i of $\mathbf{y}$ ($e_i \neq 0$) adds column $\mathbf{h}_i$ to the syndrome

$$\mathbf{s}(\mathbf{y}) = \sum_{i\,:\,e_i \neq 0} \mathbf{h}_i \mod 2. \tag{17}$$

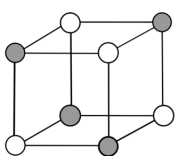
Let us assume $|\mathbf{e}| \leq t$ errors have occurred, such that $\mathbf{e}$ is correctable. Then, BDD would always be able to determine $\mathbf{x} = \mathbf{y} - \mathbf{e}$, the ML decision.

Consider the special cases where all errors occur in the I-part of H. This is a parity set of the code, so BDD on $\mathbf{y}$ is achieved by simply *re-encoding* $\mathbf{y}_P$ (i.e., the corresponding information set) which we assume to be error-free. The decoder output is

$$\mathbf{x} = \mathbf{y}_P G. \tag{18}$$

In the case where $|\mathbf{e}| \leq t$, this gives $\mathbf{x} = \mathbf{x}_s$. In fact, in many cases we are only concerned with obtaining the information bits, so the decoder can stop by simply outputing $\mathbf{y}_P$. This brings us to the idea behind PD: It is possible to determine whether all errors have indeed occurred within the I-part of H (a parity set).

Theorem 1 ([2, Ch. 17])**.** *Given a systematic parity-check matrix, H, for an $[n, k, d_{\mathbf{min}}]$ code, and a received channel vector $\mathbf{y} = \mathbf{x}_s + \mathbf{e}$. Assuming $|\mathbf{e}| \leq t = \lfloor (d_{\mathbf{min}} - 1)/2 \rfloor$ errors have occurred, then these are all confined to the $n - k$ positions corresponding to the I-part of H if and only if the weight of the syndrome is $|\mathbf{s}| \leq t$.*



Definition 1 (Permutation Decoding [9, 12])**.** *Given a noisy codeword,* $\mathbf{y} = \mathbf{x}_s + \mathbf{e}$*, where* $|\mathbf{e}| \leq t$*, try to find a* $\sigma \in \mathrm{Aut}(\mathcal{C})$ *such that* $|\mathbf{s}(\sigma(\mathbf{y}))| \leq t$*. If all errors have been moved into a parity set, then Theorem 1 shows that the decoding may stop, using* (18) *to produce* $\mathbf{x} = \sigma^{-1}(\mathbf{y}_P G) = \mathbf{x}_s$ *as the decoder output.*

The decoding ends when $|\mathbf{s}| \leq t$ (success), or when $\mathrm{Aut}(\mathcal{C})$ has been exhausted (failure). The effectiveness of PD depends on $\mathrm{Aut}(\mathcal{C})$. Even if $|\mathbf{e}| \leq t$, the required permutation to detect (and correct) this particular noise pattern may not exist in $\mathrm{Aut}(\mathcal{C})$. Furthermore, the preprocessing stage of determining $\mathrm{Aut}(\mathcal{C})$ may be prohibitally complex. Since $\mathrm{Aut}(\mathcal{C})$ is a *code* property, it is valid irrespective of the information set (for any choice of parity-check matrix). The re-encoding may be done directly on H, by adding columns of P, alleviating the need to keep a generator matrix in memory.

2.5 Soft Decision Decoding

Although classical coding is dominated by hard decision (quantized) decoding, an entire field of research is devoted to devising algebraic decoding algorithms which can take advantage also of the unquantized, "raw" channel output, typically referred to as *soft information.* It can be shown that maximum likelihood decoding (optimum decoding) has a gain of up to 2-3 dB when soft decisions are used rather than hard decisions [7]. The main efforts in this field concentrate on either convolutional codes or on list-based algorithms for block codes [7]. An important algorithm for computing exact posterior information on either a convolutional or a block code is the Viterbi algorithm. A significant element in this algorithm is that the soft decoding takes place on a *trellis,* a special graph representation of the code. Therefore, the Viterbi algorithm is one of the first examples of the decoding of *codes on graphs* via *message passing.* However, the complexity of the Viterbi algorithm depends on the complexity of the underlying trellis, and it generally holds that powerful codes have high complexity. As such, this approach quickly becomes impractical for some applications.

3 Modern Coding Theory

Modern coding theory embodies the "paradigm shift" from classical to modern coding schemes. In broad strokes, the shift in paradigm that

took place in the 1990's, was sparked by the advent of turbo codes in 1993 [13], and the discovery that these simple parallel concatenated, randomly interleaved convolutional codes closely approached the capacity on any memoryless channel. The re-discovery of Gallager's low-density parity-check (LDPC) codes [14] at around the same time [15, 16] confirmed that random codes perform surprisingly well, precisely in line with Shannon's predictions half a century before [1].

As opposed to classical coding theory, the defining properties of modern coding theory are not only minimum distance, but also randomness, sparsity, and locality; i.e., codes on graphs. One important discovery is that optimized pseudorandom codes can achieve what classical codes (under optimal BDD) can not. By increasing the blocklength of the code, optimized graph codes (i.e., carefully designed Turbo codes and irregular LDPC codes) may closely approach the capacity of the channel [17]. Furthermore – and of equal importance – this capacity-approaching performance is achieved using an efficient (though suboptimal) decoding algorithm able to handle long codes, if a *sparse* graph representation (of the parity-check matrix) exists. A graph, $\mathcal{G}$, is defined as sparse if $|\mathcal{G}| = \mathcal{O}(n)$ [18]. By taking a *localized* (i.e., graph-based) view on the decoding problem, both the overall decoding complexity is reduced, and the decoder will generally correct more than t errors in a block. Recent developments in the field are collected in [4].

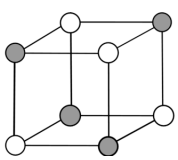
3.1 Codes on Graphs

A linear code, represented by an $(n-k) \times n$ parity-check matrix, H, can equivalently be represented by a graph. To form an adjacency matrix, H may be combined with its transpose, giving a $(2n-k) \times (2n-k)$ matrix

$$\mathbf{TG}(H) = \begin{bmatrix} 0 & H \\ H^T & 0 \end{bmatrix}$$

representing the *Tanner graph* of H [15]. This graph is bipartite, where one partition, denoted by $\mathcal{V}$, corresponds to the n columns of H, and the other, $\mathcal{U}$, to the $n-k$ rows of H. Just as H is a nonunique basis for $\mathcal{C}^{\perp}$, there are many structurally distinct Tanner graphs for the same code.

The emphasis on locality in modern coding theory is related to the partitioning of a complex problem into a system of simpler subproblems. The decoding problem of a linear code has an inherently simple partitioning. The parity-check matrix defines a set of simultaneous



constraints which must all be satisfied for a vector to be a codeword. If a code has a sparse parity-check matrix, then the rows of this matrix define a system of truly local (weakly-linked) constraints sharing the same variables (codeword positions).

The Tanner graph is a special case of the more general class of *factor graphs* [19], which are used to encode – or *realize* – a multivariate function, F, on a set of variables, $\mathcal{V}$, in a graphical manner, based on some nontrivial partitioning into a system of subfunctions

$$F(\mathcal{V}) = f_1(\mathcal{V}_1)f_2(\mathcal{V}_2)\dots f_{K-1}(\mathcal{V}_{K-1})$$

where $\mathcal{V}_i \subset \mathcal{V}$. This is a general description, reflecting the range of applications of factor graphs across different scientific fields [19]. The most famous application of factor graphs is Pearl's *belief propagation* (BP) algorithm, which, as we will see, forms the basis also for the application of factor graphs to the problem of statistical inference and error-correction [20].

In general terms, the nodes of the bipartite factor graph comprise one set of function nodes and one set of variable nodes, with an edge (f, v) *iff* $v \in \mathcal{V}$ is in the domain of f. In this sense, $f_i(\mathcal{V}_i)$ is a subfunction of $F(\mathcal{V})$, with "communications links" to all the other nodes (subfunctions) concerned with one or more of the same variables.A function node operates by executing this subfunction, producing an output for each adjacent edge. An important principle of BP is that all functions operate according to an *extrinsic* principle; the output of a function, f, along a specific edge (f, v) is defined as the *marginal* of the function on all its variables *except* v, i.e. the sum

$$\Pr(v) = \sum_{v' \in \mathcal{V}_i \setminus \{v\}} f_i(\mathcal{V}_i). \tag{19}$$

For each fixed value of v (in the binary case, 0 and 1), the marginal $\Pr(v)$ equals the total probability of f_i over all assignments to the *other* variables, $\mathcal{V}_i \setminus \{v\}$. In the following, we will adopt the standard shorthand notation for a marginal on v, rewriting (19) as, $\Pr(v) = \sum_{\sim v} f_i(\mathcal{V}_i)$ [19]. The marginal on v represents the posterior probability (APP) of the corresponding variable, conditioned on the other variables, under the restrictions of the local subfunction. This way, all the information needed for a node to operate is locally available. The (local) *neighborhood* of a node, v, (i.e., the set of nodes incident on v) is denoted by $\mathcal{N}_v$, and we also use the notation $\mathcal{N}_v^u = \mathcal{N}_v \setminus \{u\}$.

Since variables are shared between factor nodes, the variable nodes must serve the special purpose of linking the entire system together. By enforcing the simple *equality function,* these nodes attempt to ensure a coherence between their adjacent edges, which is vital as these represent a specific variable. As we will see, a variable node can also be seen as containing a local subfunction which facilitates the description of a single update rule. If the function F has a factorization in which no subfunctions share more than one variable, then the resulting factor graph is a *tree.* In fact, to form a connected tree, all subfunctions must share at most one variable. In this case, all subfunctions are independent such that the solutions to the subproblems may be combined to an exact global solution.

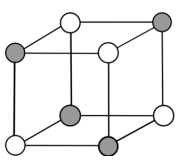
We now return to the application of factor graphs that is relevant for this thesis. In the context of optimum soft-decision decoding, the global problem to be solved is to determine the codeword *nearest* to the received noisy channel vector, $\mathbf{y}$, according to some metric. Recall the MAP decision problem. Computing the APPs for each codeword is not practical, as the size of the task grows exponentially in $k = \dim(\mathcal{C})$.[2] The purpose of a factor graph is to partition the problem of determining the maximum *bitwise* APPs, $\Pr(x_i \mid \mathbf{y})$, for each codeword position $0 \leq i < n$. For a linear code, we may assume that the probability of a bit being 0 or 1 at the source, is equal (uniform priors) [7]. The vector formed by quantizing the MAP decision at each position is not necessarily a codeword, so the bitwise MAP decoder is optimal in terms of bit error rate [21].

A factor graph provides a very efficient framework for computing this MAP decision, at a complexity linear in the blocklength. Each row of H is a constraint (or *check* equation), sharing the set $\mathcal{V}$ of n variable nodes. Any codeword of $\mathcal{C}$ must satisfy all the constraints of H, so that $\mathcal{C}$ can be viewed as the intersection of the $n-k$ *supercodes* (of $\mathcal{C}$) "checked" by each individual row, h_j, of H, such that

$$\mathcal{C} = \langle H \rangle = \langle h_0 \rangle \cap \langle h_1 \rangle \cap \cdots \cap \langle h_{n-k-1} \rangle.$$

Each of these supercodes have dimension $n-1$ (only one constraint). From the graph perspective, where a check node has $\rho = \deg(f)$ adjacent edges, the supercodes can be viewed as $[\rho, \rho-1, 2]$ single parity-

[2] Alternatively, exponential in the codimension, $\dim(\mathcal{C}^{\perp})$, if based on the parity-check matrix.



check (SPC) code. This supercode can detect all single ($t_d = 1$) *parity errors.*

Example 1 (Partitioning of the decoding problem)**.** *Consider the* $[7, 4, 3]$ *Hamming code represented by the standard form (i.e., systematic) parity-check matrix,*

$$H = \begin{array}{c} \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \end{array}. \tag{20}$$

The (global) code membership function is then

$$F(\mathcal{V}) = f_0(v_0 + v_3 + v_4 + v_6) f_1(v_1 + v_3 + v_5 + v_6) f_2(v_2 + v_4 + v_5 + v_6). \tag{21}$$

A single constraint, say the first row (f_0), gives the supercode parity check matrix, $H^{(0)} = [1 \mid 1\ 1\ 1]$ *(checking positions 0, 3, 4, and 6), so this is a* $[4, 3]$ *SPC code with generator matrix*

$$G^{(0)} = [P \mid I_{\rho-1}] = \begin{array}{c} \begin{matrix} 0 & 3 & 4 & 6 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \end{array}.$$

where we have omitted the zero columns. The fact that the $\rho = 4$ *positions of this code correspond to positions 0, 3, 4, and 6 of the global code is encoded in the factor graph realization. This way, a local supercode can decode, contributing information to the global decoding problem.*

Similarily, choosing column $h_3 = (1\ 1\ 0)^T$ *of* (20) *(variable* v_3*), we see that the input likelihood,* $\Pr(y_3 \mid v_3)$*, is repeated to* $\gamma = 2$ *check nodes,* f_0 *and* f_1*. As such, this equality constraint (EQC) can be implemented via a length-3 repetition code. The two possible codewords are* $(0\ 0\ 0)$ *and* $(1\ 1\ 1)$*, so the corresponding generator matrix for this* $[3, 1]$ *repetition code is,* $G_{(3)} = [1\ 1 \mid 1]$*, which gives*

$$H_{(3)} = [I_\gamma \mid P^T] = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

Consider the $\gamma + 1$ *inputs to variable node* v_3*. The equality constraint of this repetition code enforces that edge 0 equals edge 2, and that edge 1 equals edge 2.* ◊

For simplicity in the following arguments, let us assume that the code is such that an acyclic Tanner graph representation exists. In this case, all messages passed along edges are independent of each other (satisfying the extrinsic principle), and the APPs computed by BP are exact (as on a trellis) [21]. With this simplifying assumption, we will describe the concepts of graph-based decoding.

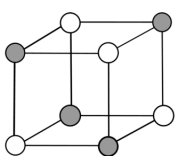
3.2 Message Passing

Each supercode covers a subset of the positions of the 'global' code, $\mathcal{C}$. Since these subsets overlap, the problem of decoding a received vector with respect to $\mathcal{C}$ is solved by the simultaneous decoding of the supercodes. In this sense, one often says that a check node *checks* its adjacent variables, and when the quantized sum of these variables is zero a codeword of this supercode is obtained. When the global syndrome check (for the main code) is satisfied, *all* $n - k$ supercodes have a codeword, and a codeword for the global code is found at the variable nodes.

In the acyclic case, the BP rule is that a node may update an edge as soon as messages are *pending* on all other adjacent edges (recall the extrinsic principle discussed above). This process initiates at the leaf nodes, which serve as *input nodes* in the sense that the input vector (in this case, the channel likelihood vector) is placed onto these nodes, which is then injected into the factor graph as these nodes update their single edge. Eventually, all edges are updated in both directions. Beyond this time, any subsequent updates will simply recalculate existing information, so the BP algorithm stops [4].

The process of feeding the output of one decoder into the input of another, following the extrinsic principle, is known as *message passing*. This process invokes the equality constraint in all the variable nodes, and the SPC decoder in the check nodes. In order to solve the global decoding problem, the codewords of the constituent supercodes must be combined in an attempt to form a global decision (i.e. a codeword) for the full $[n, k]$ code which, hopefully, agrees with the source vector, $\mathbf{x}_s$. The received channel vector, $\mathbf{y}$, consists of soft information, the real-valued input values, corresponding to the source vector plus some amount of noise resulting from transmission across the channel (assuming additive noise). From a local perspective, each supercode is only aware of its immediate surroundings, namely its adjacent edges and nodes in the factor graph.

The computation of a function can be expressed by a "truth table," which relates the configuration space of the variables to an indicator function [22]. This indicator function is a restriction of the function on the configurations, in the sense that it is 1 only when the corresponding configuration on the variables gives a valid function output. Let $\mathcal{V}$ denote the (sub) set of c q-ary variables, which take a configuration of the configuration space, $\mathbf{s} \in \mathcal{S}_q^c$. Any local function, F, can be implemented



by its appropriate binary indicator vector, $\mathbb{1}_F(\mathbf{s})$, of length q^c, which is 1 *iff* the corresponding configuration, $\mathbf{s}$, of the input variables is a valid state for this function; otherwise, it is zero. Here, c is the number of variables to the function, where one of these is considered an output while the rest are inputs. When notationally convenient, we use the equivalent function notation, $f_F(\mathbf{s}) = \mathbb{1}_F(\mathbf{s})$, or simply $f(\mathbf{s})$ for short.

Before describing the details on computing local functions, we should establish a notion of what the input messages are. First of all, the initial messages are the likelihoods coming from the channel. After being demodulated from channel symbols, we have a vector of normalized bitwise likelihoods, $\Pr(y_i \mid v_i)$. These are attached to the factor graph on special-purpose *input nodes,* or Forney-style *half edges* [23], see Fig. 6. These should be viewed as leaf nodes, one connected to each variable node. The purpose of message passing on the factor graph is to solve the decoding problem, which we encode as a distributed form of a bitwise MAP decoding. The decoding is based on BP, so the messages deal with probabilities, or likelihoods, of binary variables, as represented by real-valued vectors of length 2.

Consider the case of a parity check function, $f = \mathbb{1}_{\text{XOR}}$, on three variables, $\mathcal{N}_f = \{b, p, q\}$, i.e., a [3, 2] SPC code. Say we want to marginalize on b. The input messages are then $\mu_{p\to f} = (p_0, p_1)$ and $\mu_{q\to f} = (q_0, q_1)$. However, as these are *output* messages of variable nodes p and q (we have not yet discussed variable node update) we initially take the simplifying assumption that the "input" nodes (p and q; b is currently the output) are leaf nodes. Then, the input messages to f are the likelihoods from the channel on these nodes; $(p_0, p_1) = \Pr(y \mid p)$ and $(q_0, q_1) = \Pr(y \mid q)$. The corresponding truth table for $f = \mathbb{1}_{\text{XOR}}$ is then

p	q	b	Ψ^b	$\mathbb{1}_{\text{XOR}}$	$\Psi^b \cdot \mathbb{1}_{\text{XOR}}$	$\sum_{\sim b} \Psi^b \cdot \mathbb{1}_{\text{XOR}}$
0	0	0	p_0q_0	1	p_0q_0	$p_0q_0 + p_1q_1$
0	1	0	p_0q_1	0	0	
1	0	0	p_1q_0	0	0	
1	1	0	p_1q_1	1	p_1q_1	
0	0	1	p_0q_0	0	0	$p_0q_1 + p_1q_0$
0	1	1	p_0q_1	1	p_0q_1	
1	0	1	p_1q_0	1	p_1q_0	
1	1	1	p_1q_1	0	0	

where Ψ^b denotes the product of incoming messages from all adjacent variables except b, according to the configuration of the corresponding variables in the truth table. The notation $\sum_{\sim b}$ denotes the component-wise sum, according to the possible assignments of the marginalized variable, b, giving the vector

$$\mu_{f\to b} \;=\; \sum_{\sim b} \Psi^b f(b,\, p,\, q) \;=\; \sum_{\sim b} \Psi^b \cdot \mathbb{1}_{\mathrm{XOR}}. \tag{22}$$

This message from f to b conveys information on the value of b, from the perspective of f (given the subtree rooted in the node f)

$$\Pr(b = \alpha \;\mid\; \bigoplus_{v\in\mathcal{N}_f^b} v = \alpha) \tag{23}$$

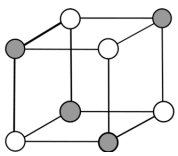
where $\alpha \in \{0, 1\}$. In the top half of the configuration space, variable b is fixed to 0, and the two valid (sub) configurations of p and q (i.e., 00 and 11) are summed to produce the 0-field of $\mu_{f\to b}$. Similarily, if $b = 1$, then p and q must be configured to 01 or 10, which gives the 1-field. Thus, for two inputs, the (normalized) marginal on b coming from f is

$$\mu_{f\to b} = z^{-1}(p_0 q_0 + p_1 q_1,\; p_0 q_1 + p_1 q_0) \tag{24}$$

where $z^{-1} = 1/(\sum \Psi^b \cdot \mathbb{1}_{\mathrm{XOR}})$ is a normalization constant such that the fields (marginal likelihoods) sum to 1. This exact same process is then repeated at each variable, to produce the three marginals which are then passed out onto the corresponding adjacent edges. As is apparent from the small example, a marginal is computed as a *sum of products.* Using vector notation, we denote this *soft-XOR* operation [24] by $\tilde{\oplus}$, such that (24) becomes

$$\mu_{f\to b} = z^{-1} \bigoplus_{v\in\mathcal{N}_f^b}^{\sim} \mu_{f\leftarrow v} \quad . \tag{25}$$

Extending to four variables (three inputs and one output), the configuration space has 2^4 configurations. However, the structure of the XOR function is such that it is easily factorized. Extending the three-variable case by a fourth variable, d, is a matter of adding a column to the right of b in the truth table, which is 0 in the initial 8 configurations. Then, a copy of $\mathcal{S}_2^3$ (the three-variable truth table considered above) is added to the end of the table, where the new column d now takes value 1. Since we are simply copying $\mathcal{S}_2^3$, the computed function values for $\Psi^p \mathbb{1}_{\mathrm{XOR}}$,



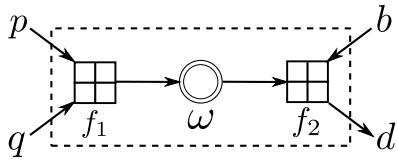


Fig. 5: *Factoring a four-variable XOR function into a cascade of atomic (three-variable) XOR functions, linked by a state variable, ω. This figure shows the calculation of $\sum_{\sim d} \Psi^d f(d,\, b,\, p,\, q)$ as $\sum_{\sim d} \Psi^d f_2(d,\, b,\, \omega)$, by reusing $\sum_{\sim \omega} \Psi^\omega f_1(\omega,\, p,\, q)$.*

$\Psi^q \mathbb{1}_{\mathrm{XOR}}$, and $\Psi^b \mathbb{1}_{\mathrm{XOR}}$ can also be re-used when computing $\Psi^d \mathbb{1}_{\mathrm{XOR}}$. Similar for the next 8 configurations, for which d is configured to 1. To evaluate the 4-variable case graphically, we factor the problem into two instances of 3-variable cases, combined by a new state variable, ω, as shown in Fig. 5. Denote the $[4,\, 3]$ SPC supercode by $f = f_1 \circ f_2$. To evaluate $\mu_{f \to d}$ we first evaluate $\mu_{f_1 \to \omega} = (\omega_0^1,\, \omega_1^1)$ using (24). We use the superscript '1' to denote that this is only the "left-hand part" contribution to the value of ω. This is the purpose of the state node. Then, the desired output (for d) may be evaluated the same way, as $\mu_{f \to d} = \mu_{f_2 \to d} = (d_0,\, d_1)$. To see that this gives the same result as directly evaluating the 2^4 state XOR function, we expand

$$\begin{aligned}(d_0,\, d_1) &= (\omega_0^1,\, \omega_1^1) \tilde{\oplus} (b_0,\, b_1) = (s_0^1 b_0 + s_1^1 b_1,\, \omega_0^1 b_1 + \omega_1^1 b_0) \\ &= [b_0(p_0 q_0 + p_1 q_1) + b_1(p_0 q_1 + p_1 q_0) \\ &\quad\; b_1(p_0 q_0 + p_1 q_1) + b_0(p_0 q_1 + p_1 q_0)] \\ &= (p_0,\, p_1) \tilde{\oplus} (q_0,\, q_1) \tilde{\oplus} (b_0,\, b_1).\end{aligned}$$

Thus, each outgoing message can be calculated in a recursive manner. After computing d, we continue by computing for b, such that we may reuse $(\omega_0^1,\, \omega_1^1)$ when we calculate $(b_0,\, b_1) = (\omega_0^1,\, \omega_1^1) \tilde{\oplus} (d_0,\, d_1)$. Next, the state variable is used to calculate p and q, where the "right-hand part" of ω is $(\omega_0^2,\, \omega_1^2) = (b_0,\, b_1) \tilde{\oplus} (d_0,\, d_1)$.

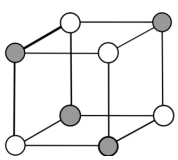
3.3 Sum-Product Algorithm

The Sum-Product algorithm (SPA) [19] is a well-known instance of the BP algorithm, which follows naturally from the evaluation of functions according to a truth table. It is common in the literature to describe the SPA in terms of two separate update rules; one for a function node, and one slightly simpler for a variable node. However, also the variable node

rule can be expressed using a truth table. Representing the workings of a variable node in terms of a "function" is somewhat forcing the issue, but consider the purpose of a variable node, v. The incoming messages from the adjacent check nodes are marginals on v. In a tree, they represent the "view" on the value of v, from the perspective of each adjacent check node. Consider one, f_i. The message (vector) $\mu_{f_i \to v}$ expresses the probability that v is 0, given the state of the *other* variables adjacent to f_i. In other words, f_i is telling v, "the probability that you are 0, is the sum of the probabilities that my other variables *also* sum to 0." And similarily for the probability that v is 1. From another adjacent check node, say f_j, the node v receives "beliefs" on its state from the perspective of a different branch of the tree. In total, v receives a set of information (messages) on its state, which it must attempt to combine in order to decide its state; i.e., the APP $\Pr(v \mid \mathbf{y})$. In terms of a "local function," this can be expressed as an equality constraint, with two valid states; when all inputs are 0 or all inputs are 1. The output of v to f_i, then, expresses the joint probability that v is 0 given the marginals received from all the check nodes except f_i – again, in terms of the extrinsic principle. In this sense, node v acts as a mediator representing and negotiating the value of v.

Example 2. *Consider again variable node v_3 of the previous example. Say the $\gamma = 2$ adjacent edges (check nodes), f_0 and f_1, contain marginal likelihoods (pending messages) $(0.1, 0.9)$ and $(0.6, 0.4)$, respectively. The third edge contains the channel likelihood, $y_3 = (0.3, 0.7)$, such that the hard decision vector is $(1\ 0\ 1)$. The truth table representation of this function on four variables is*

u_3	y_3	f_0	f_1	$\mathbb{1}_{\text{EQC}}$	$\Psi^{u_3}\mathbb{1}_{\text{EQC}}$	$\Psi^{f_0}\mathbb{1}_{\text{EQC}}$	$\Psi^{f_1}\mathbb{1}_{\text{EQC}}$
0	0	0	0	1	$0.3 \cdot 0.1 \cdot 0.6$	$0.5 \cdot 0.3 \cdot 0.6$	$0.5 \cdot 0.3 \cdot 0.1$
0	0	0	1	0	0	0	0
		...		...		...	
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
		...		...		...	
1	0	1	1	0	0	0	0
1	1	1	1	1	$0.7 \cdot 0.9 \cdot 0.4$	$0.5 \cdot 0.7 \cdot 0.4$	$0.5 \cdot 0.7 \cdot 0.9$



where $\mathbb{1}_{\mathrm{EQC}}$ is the indicator vector for the equality constraint (EQC). Let us first calculate the bitwise APP for v_3, i.e. the (normalized) marginal posterior on the state node, u_3. According to (26) *we have*

$$\mu_{u_3}^{(0)} = z^{-1}\,0.018 = 0.067 = \Pr(v_3 = 0 \mid \mathbf{y}),$$
$$\mu_{u_3}^{(1)} = z^{-1}\,0.252 = 0.933 = \Pr(v_3 = 1 \mid \mathbf{y}),$$

where $z^{-1} = (0.018 + 0.252)^{-1} = 3.704$.

The two output messages, $\mu_{v\to f_0}$ and $\mu_{v\to f_1}$, are computed the same way as the APP, by marginalizing $\mathbb{1}_{\mathrm{EQC}}$ on f_0 and f_1, respectively. The "input message" from the degree-1 state node remains neutral, $u_3 = (0.5,\, 0.5)$, and we have

$$\mu_{v\to f_0} = z^{-1}(0.18,\, 0.28) = (0.39,\, 0.61)$$

where $z^{-1} = (0.18 + 0.28)^{-1} = 2.174$. The calculation for f_1 yields

$$\mu_{v\to f_1} = z^{-1}(0.03,\, 0.63) = (0.067,\, 0.932).$$

The indicator function, $\mathbb{1}_{\mathrm{ECQ}}$, for a variable node is simple. The truth table representation in Example 2 leads to the following update rule: ♢

$$\begin{aligned}
\mu_{b\to f}^{(\alpha)} &= \sum_{\sim f} \Psi^f \cdot \mathbb{1}_{\mathrm{EQC}} \\
&= \Pr\Big(b = \alpha \;\Big|\; \overbrace{\bigoplus_{v\in\mathcal{N}_{f_0}^b} v = \alpha,\ \bigoplus_{v\in\mathcal{N}_{f_1}^b} v = \alpha,\ \ldots,\ \bigoplus_{v\in\mathcal{N}_{f_{\gamma-1}}^b} v = \alpha}^{\text{For all } f\in\mathcal{N}_b^f}\Big) \\
&= \Pr\Big(b = \alpha \;\Big|\; \bigwedge_{f'\in\mathcal{N}_b^f} \Big[\bigoplus_{v\in\mathcal{N}_{f'}^b} v = \alpha\Big]\Big)
\end{aligned} \tag{26}$$

as illustrated in Fig. 6. Since $\mathbb{1}_{\mathrm{ECQ}}$ has only one contribution to each marginal, the sum-part (as well as the two-state indicator function) is usually omitted such that (26) may be written in vector notation as

$$\mu_{v\to f} = y_v \prod_{f'\in\mathcal{N}_v\setminus\{f\}} \mu_{v\leftarrow f'} \tag{27}$$

where the product is vector point product.

The marginal likelihood on v, $\Pr(y \mid v)$, is then the joint likelihood of all adjacent messages to v. Using Bayes' theorem (11), and the fact that

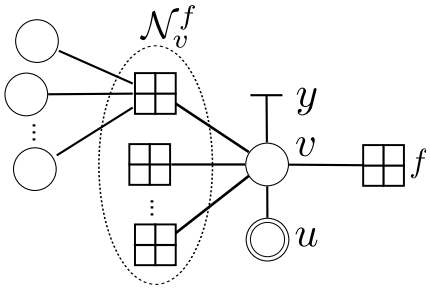


Fig. 6: *Update of a variable node, v. Notice the input half-edge, y, and the output state node, u, containing the channel likelihood and the posterior probability (APP), respectively.*

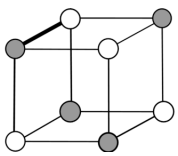
$\Pr(y)$ is independent of v and the assumption of uniform priors, we have that the bitwise APP is porportional to the likelihood. As such, the normalization stage produces the APP, $\Pr(v \mid \mathbf{y})$. As seen in Example 2, this APP can be calculated by attaching a state node, u, to v – see Fig. 6. Note that such degree-1 nodes will not produce any output message (along their single edge), due to the extrinsic principle.[3]

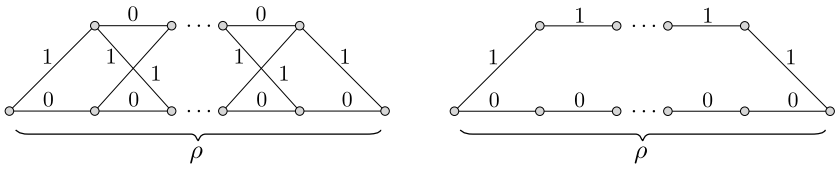
From an implementational point of view, the XOR rule on $|\mathcal{N}_f^v|$ variables can be implemented in a cascade of degree-3 nodes, interconnected by state nodes

$$\mu_{f\to v} = \tilde{\bigoplus_{v'\in\mathcal{N}_v\setminus\{v\}}} \mu_{f\leftarrow v'} \quad . \tag{28}$$

We have thus seen two important simplifications of message passing, in which the particular structure of the indicator function is exploited in order to give an efficient implementation of both the function and the variable node rules. In fact, as shown in Fig. 5, this is an extension of the factorization approach which makes factor graphs an efficient tool for computing exact marginals. Further simplifications are used in the literature, to facilitate implementation. Rather than working in the

[3] Also note that this notion of degree-1 nodes differs from that discussed in Paper IV, in which the "systematic nodes" also contain an input node which makes them degree-2 nodes in this current context.





(a) SPC (b) REP

Fig. 7: *The trellis of the dual (repetition) code has simpler trellis than that of the SPC code, so the check node is commonly computed on the dual code.*

probability domain, it is common to compress the likelihoods into a *likelihood ratio*,

$$\nu_i = \frac{\mu_i^0}{\mu_i^1} = \frac{\Pr(y_i \mid v_i = 0)}{\Pr(y_i \mid v_i = 1)}. \tag{29}$$

Appropriate clipping (truncating of real numbers in a finite-precision computer) is performed during the demodulation process to avoid dividing by zero and buffer overflow. For increased numerical stability, one typically takes the logarithm of the ratio, working in the log-likelihood ratio (LLR) domain, denoted by

$$L_i = \ln(\nu_i). \tag{30}$$

The quantization rule becomes the *sign* operation, $\text{sign}(L_i)$.

Working in the LLR domain has several benefits in implementing the SPA. For instance, the variable rule (27) reduces to addition (which is, generally, a less costly operation than real-valued multiplication)

$$\mu_{v\to f} = y_v + \sum_{f'\in\mathcal{N}_v^f} \mu_{v\leftarrow f'}. \tag{31}$$

The check rule (28) can be re-expressed [4] as

$$\mu_{f\to v} = \left(\prod_{v'\in\mathcal{N}_f^v} \text{sign}(\mu_{f\leftarrow v'})\right) 2\tanh^{-1}\left(\prod_{v'\in\mathcal{N}_f^v} \tanh\left(\frac{|\mu_{f\leftarrow v'}|}{2}\right)\right). \tag{32}$$

This can be computed efficiently via the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm [25], which is reminiscent of the Viterbi algorithm

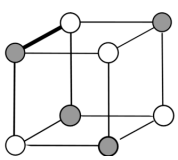
but which also produces the bitwise APP values for all variables (outgoing messages for all adjacent edges). This algorithm traverses the trellis twice (rather than once), and with a more complex local operation at each edge (i.e., the hyperbolic tangent function, and its inverse, in (32)). Although more complex than the Viterbi algorithm, the soft output (extrinsic values) generated facilitate message passing. This is implemented based on a trellis representation of the $[\rho, \rho - 1]$ (SPC) supercode, $\langle h_j \rangle$, generated by one row h_j of H – see Fig. 7(a). The maximum *state space* of the trellis is known to be bounded by $\rho_{\max}\langle h_j \rangle \leq \min(\dim\langle h_j \rangle,\ \dim\langle h_j \rangle^{\perp})$ [7], which gives $\rho_{\max}\langle h_j \rangle \leq \min(\rho, 1) = 1$. Thus, for any SPC code (regardless of ρ), the minimal trellis has only 2 states. However, the trellis *branch complexity*, in terms of number of "edges" (transitions between trellis sections), determines the number of computations required in a trellis-based decoding algorithm to decode a received sequence [7]. The dual of the SPC code is the repetition (REP) code. Since the two codewords of the dual code are uniform (all-zero and all-one), there is no need for any interconnecting (i.e., $0 \leftrightarrow 1$) branches, and obviously the branch complexity is simpler – see Fig. 7(b).

An "efficient" implementation of the BCJR algorithm will thus be on the dual code, using a Fourier transform to convert the "SPC LLRs" into dual "REP LLRs," and back. Then, for this specific two-state, no-intersection trellis, the forward and backward (past and future APP) pass of the BCJR algorithm may be performed using only two independent arrays, with a subsequent combination stage to produce the outputs.

3.4 SPA ON CODES WITH CYCLES

The SPA update rules produce exact APP values when the corresponding code realization is acyclic. However, to achieve this the code must either be very weak,[4] or the partitioning of the code must be coarse, such that cycles are confined within component codes, at the cost of increased complexity (which is exponential in component code dimension) [26]. Hence, we need cycles to give codes which are simultaneously useful and useable.

[4] A tree graph is a code of minimum distance 2. This follows from the presence of multiple leaf variable nodes in the graph. When $R > 1/2$, the I-part of H is smaller than the P-part, such that a weight-2 codeword is formed by a vector which is nonzero in these two positions (syndrome sums to $\mathbf{0}$) [4].



The practical approach to BP message passing decoding of cyclic graphs is to implement the SPA rules, and compensate for the lack of starting and ending states by allowing the decoder to *iterate,* with some stopping criterion. The cost is apparent, in terms of latency (more than one pass through the graph), and the need for a global stopping criterion which violates the locality argument of the SPA; either the syndrome is zero or a maximum number of iterations is exhausted (or some other stopping criterion). The stopping criterion requires polling the state of all check nodes in the graph, as well as issuing a halting signal to all variable nodes. Finishing the example used in the previous sections, the entire Tanner graph of the $[7, 4, 3]$ Hamming code is drawn in Fig. 8, where the cycles are obvious – especially the shortest of length 4. Attached to each check node is a state node, s_j, which contains the *soft syndrome* value of the corresponding supercode. This value can be viewed as the state of the check node, reflecting the probability that the adjacent variable nodes hold a codeword of the supercode. Only when all checks hold such a supercodeword does $\mathcal{V}$ correspond to a codeword of $\mathcal{C}$, and decoding may stop.

The crucial insight by Gallager was that BP in the context of decoding does not need to be exact. The APP value for each variable node is only required to converge *towards* its correct value. In BPSK signalling, the LLR should only tip slightly to the correct side of zero for the quantizing (hard decision) stage to produce the correct bit – at which point the APP is discarded anyway. Furthermore, the feedback of cycles can be reconsidered in terms of information attenuation. As information passes around a cycle, it is gradually moderated by other nodes such that the self-reinforcing effect on the originating node is reduced. As such, the adverse effect of a cycle is depends on the length of the cycle, where especially length-4 cycles are detrimental to performance. The minimum cycle length in the graph is referred to as the *girth,* g, of the graph. Within the first $g/2$ iterations, information has not yet passed through the shortest cycles, and calculations are still exact – a finite-length formulation of the *local tree assumption.*

It is well known that certain properties of the parity-check matrix (or the corresponding graph) will also affect the error-correction performance, and must be taken into account along with the overall properties related to the code. With the shift into *modern* (i.e., graph-based, iterative) coding theory, one says that the minimum distance is not everything [21]. For instance, one is no longer satisfied to decode t

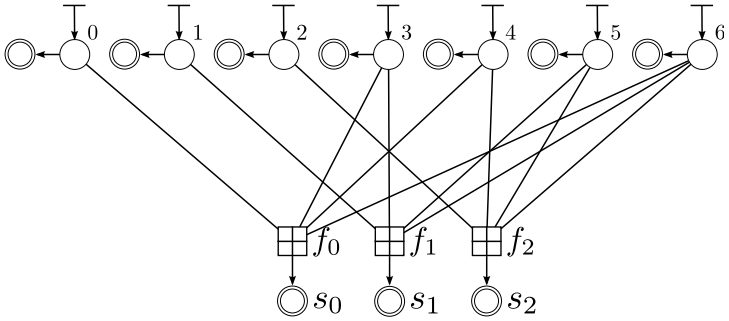


Fig. 8: *Tanner graph representation of the* $[7, 4, 3]$ *Hamming code. The syndrome bits* s_j *correspond to the "state" or output of each check node. The hard decision on each is combined to form the final output of the global function* (21)*,* $F(v_0, \ldots, v_6) = \mathbb{1}_{\mathcal{C}}(v_0, \ldots, v_6)$.

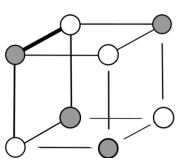
errors; to compete with current records, one needs to decode beyond the minimum distance.

The specific instance of SPA message passing on a Tanner graph is the standard system for decoding of codes on graphs, with LDPC codes as the most famous example. The constituent supercodes and the necessary communication between these, as described above, lead to the SPA. This algorithm is exact so long as the problem instance (i.e., the Tanner graph) is a tree. However, an iterative version for decoding of general codes, which are commonly either extremely large (LDPC codes) or very dense (HDPC codes), becomes very complex to analyze.

3.5 LDPC Codes

Gallager's LDPC codes, in the 1960's, were the first codes to approach the capacity as described by Shannon on a general channel – they are "asymptotically good" [18].[5] However, again there were challenges; Gallager's codes were described simply as random constructions, with only very general parameters such as size (matrix dimensions), row and column degree (regular codes), and a property that no pair of columns share more than one common nonzero entry (no cycles of length 4). As such, LDPC codes exist only as *ensembles,* or families, of codes which

[5]Gallager proved that the minimum distance of most (with a high probability, any random selection) LDPC codes in a regular ensemble with column weight $\gamma \geq 3$ is, asymptotically, lower bounded by a positive constant times the blocklength [27].



share the same parameters, but are otherwise completely random and different. Subsequently, it was found that irregular LDPC codes could perform better than regular ones [28]. Such an ensemble is defined by the variable and check node degree distributions, referring to the fraction of nodes of specific degree (number of adjacent edges).

When an ensemble can be proven to approach capacity on the AWGN channel, less is known for the performance of some specific sample (i.e., code) from the ensemble. Given an ensemble of codes of some rate, $R < C$, (codes from an ensemble have the same design rate, such that almost all have the same rate) the error probability function characterizing the decoding algorithm can be used to calculate a *threshold* on the error-rate performance of the ensemble with the channel/algorithm pair [29] (where we focus on the AWGN channel). This is the minimum SNR required for the decoder to converge (on a cycle-free Tanner graph) averaged over all codes in the ensemble, as the blocklength goes to infinity. Any sample (LDPC code) from the ensemble can then be simulated, to compare the slope of the FER-vs-SNR curve against this threshold.

3.6 Belief Propagation Threshold

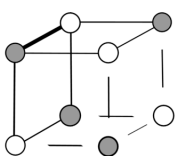
As BP is known to compute the exact MAP values when the factor graph is a tree (acyclic), it is possible to estimate the limit of the performance of an ensemble of codes under message passing decoding. The *concentration* theorem proves that, asymptotically in n, the performance of a randomly chosen code converges to the expected performance of its ensemble [29]. Independently of decoding algorithm (i.e., the variant of BP used), as long as the bit-error probability of the decoder, $P_b^{(\tau)}$, can be shown to be monotonically *increasing* with channel parameter, p (where $q > p$ is a *worse channel*), while simultaneously *decreasing* in the number of BP iterations τ, p has a finite limit known as the *threshold*, p^{BP}.

Using a technique called *density evolution* (DE) practical experiments can be implemented to allow the determination of a threshold channel parameter, beyond which (i.e., for less noise) error-free communications is possible. That means, the error probability can be made arbitrarily low, by choosing a sufficiently long code from the ensemble in question [29, 30]. DE is an asymptotic analysis based on the degree distributions of the ensemble. By considering an infinitely long code, the independece of messages under any finite number of iterations of BP is assured, and

DE may compute the threshold on the performance of the ensemble by tracking the "evolution of densities" – i.e., the convergence of the pdfs – as a function of iterations. Under these ideal, yet artifical, circumstances, one may determine the best possible performance of the ensemble.

A simplification of DE is to use Gaussian approximations on the pdfs, which are reduced to their corresponding means. These are easier to compute, and tracking these gives a very useful estimate on the threshold. This is commonly implemented in terms of *extrinsic information transfer* (EXIT) charts [31], which visualize the information exchange between concatenated decoders as *decoding trajectories.* An estimate on the threshold is then the largest channel parameter (i.e., the highest level of noise) for which the constituent decoders remain able to exchange new information. For LDPC codes, the corresponding supercodes are the variable and check nodes, representing each different degree if the ensemble is irregular. As we have seen, the corresponding iterative decoders exchange information in terms of message passing such that the output of one decoder is the input to another. Graphically, for each decoder, the extrinsic mutual information can be plotted as a function of the *a priori* mutual information (in the interval 0 to 1). The information exchange can then be visualized by plotting the inverse of the other decoder on the same chart. The information exchange, as a function of decoder iteration number, is then visualized as a zig-zag line between the two plots. Unless the plots intersect, the mutual information converges to 1 (upper-right corner of the chart), corresponding to virtually error-free decoding [31]. The threshold, in this context, is then the maximum noise level for which the plots do not intersect, giving what is often called an *open tunnel.*

Using these techniques, the asymptotic performance of different ensembles can be compared, which has led to the discovery of optimized irregular LDPC codes with record-breaking gaps to capacity (the Shannon limit) on various channels. Implemented as an optimization problem, it is possible to determine the degree distributions which optimize the threshold. Assuming long blocklengths, codes from such ensembles perform similar to their ensemble averages. How this relates to finite-length codes from the same ensembles, however, is not easily predicted, as structural dependencies on the code *representation* can not be ruled out. This is typically estimated numerically for the specific code and graph in question.



In practice, it is possible to estimate this threshold by approximated analysis. To motivate the meaning of the threshold, we will outline a naive (perhaps inefficient) procedure where the threshold is estimated by means of standard error-rate simulations. A sufficiently large code from the ensemble is chosen, for which the average $P_b^{(\tau)}$ (for some large number of iterations, τ) for increasing p (i.e., simulating the bit-error probability) is determined by simulations. This is repeated, to produce an average BER plot over many random codes from the ensemble. By repeating this procedure for increasing values of n (simulating the limiting bit-error probability as n goes to infinity), eventually a slope resembling a vertical limit will appear. When the curve reaches some predefined *target FER* (low enough to emulate "error-free" communications, but not so low that it is impractical to simulate accurately), the crossing point (p-value) is the estimated p^{BP}. This verifies that P_b can indeed be made as small as desired (the target FER) for any sufficiently large, but otherwise randomly chosen, code from the ensemble, provided that the noise level is below the threshold, $p < p^{\text{BP}}$.

This threshold value can be thought of as a capacity of a code/decoder pair on a given channel, indicating the best possible (asymptotic) performance of the code *ensemble.* The capacity of the channel is a maximum *rate,* computed according to a function of the channel parameter, $C = f(p)$. Thus, the capacity interpreted as a maximum channel parameter, p^{Sha}, corresponds to evaluating the inverse of this function for the design rate, R, of the ensemble, $p^{\text{Sha}} = f^{-1}(R)$. The gap from threshold to capacity is then reported as the limiting performance of the ensemble, on the given channel.

3.7 Finite-Length Challenges

We have seen how finite-length topological problems, such as weight and girth, hamper the performance of soft-decision decoding of practical-sized LDPC codes. Allowing the SPA to be iterative does to a certain extent improve the performance, but one is still failing to approach the near-capacity asymptotical performance of large codes. The frame error-rate (FER) performance of finite-length codes under iterative decoding is characterized by two classes of errors. The FER plot obtained by simulating such a code over an SNR range (as described above) then takes the form of the *sum* of the corresponding two curves, and it is this curve which differs significantly from the limiting curve giving the threshold. At low SNR (high noise), the first class of errors is due to the

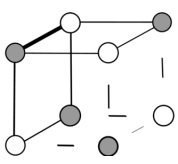
decoding algorithm, which struggles, and often fails, to converge to a codeword. We will go into detail on some of the causes of this problem in the following, but for now only refer to the fact that the *slope* of the FER curve at low SNR is much less steep than that of a code well-suited for iterative decoding (e.g., a long LDPC code).

As the SNR improves (in the simulations context), the decoder has an easier task of decoding, causing the curve to drop at some rate. This is referred to as the *waterfall region* of the curve. However, at some point, the number of errors is low enough for the decoder to begin making errors due to minimum-weight codewords of the code. Although difficult to calculate for long codes, it is known that the minimum distance of a regular LDPC code is in the order of the blocklength; i.e. quite large. Accordingly, for finite-length codes, the minimum distance is weaker, preventing the error-rate from maintaining its slope. In terms of a threshold, this effect means that this code will not reach the threshold it might have appeared to approach, but rather flatten out towards a significantly higher SNR. The discontinuity resulting from the now even further decreased slope is referred to as the *error-floor region.*

This identifies two obvious areas of improvement, either by increasing the slope in the waterfall region, or "lowering the floor," or both. For HDPC codes, large minimum distance (compared to blocklength) is usually among the defining properties, so the FER response is dominated by the waterfall region with less dramatic (or nono) floor-effect.

Experience from the conceptually simple binary erasure channel (BEC), which is easier to analyze than the BSC and AWGN channels, has identified some of the factors in this complex problem. Interestingly, these observations also have an effect on the performance under more general channels, thus providing valuable insight. Since positions not erased on the BEC are known to be correct, optimal decoding may be done by a hard-decision algorithm which iteratively finds a *solvable* check equation, corrects the erasure, and removes all the edges involved from the graph. If all erasures are correctable, the ML codeword is recovered. In this sense, a solvable equation corresponds to a check node to which only *one* of the adjacent variable nodes correspond to an erasure. This single error is always correctable, as the check-sum (XOR) of the other bits.

Among the most important recent discoveries on the dynamics of iterative decoding are *stopping sets* [32] which cause a deadlock in the iterative process. On the BEC, the iterative process stops when no



solvable equation can be found. A stopping set is a subset of codeword positions which, if all are erased, will cause the decoder to stop. To test whether some subset of variable nodes, $\mathcal{V}' \subseteq \mathcal{V}$, is a stopping set, it is easiest to consider the subgraph of $\mathbf{TG}(H)$ induced by these variable nodes; i.e. the nodes $\mathcal{V}'$, the adjacent check nodes, $\mathcal{U}' = \bigcup_{v \in \mathcal{V}'} \mathcal{N}_v \subseteq \mathcal{U}$, as well as the edges connecting $\mathcal{V}'$ to $\mathcal{U}'$. We discard all the other variable nodes, which correspond to codeword positions which are assumed *not* to be erased. If $\mathcal{V}'$ is a stopping set, such that decoding fails if all these positions are erased, then $\mathcal{U}'$ must contain no check nodes of degree 1.

According to this definition, the *support sets* (the nonzero positions) of all codewords are also stopping sets – which makes sense, in terms of our interest in a decoding algorithm which stops on reaching a codeword. To see that a codeword $\mathbf{x}$ corresponds to a stopping set, consider the subgraph induced by the *support* of $\mathbf{x}$. From the discussions on the SPA, we know that all the check equations must be satisfied (i.e., sum to 0) for $\mathbf{x}$ to be a codeword. Focusing on the check nodes in $\mathcal{U}'$, we must then have that these all connect back to $\mathcal{V}'$ an even number of times (recall that all positions outside the support of $\mathbf{x}$ are zero, by definition), which completes the argument since 2 is the smallest nonzero even number. However, the converse is generally not true; a stopping set does not need to be the support set of a codeword. This is obvious from thinking of an induced graph (on a stopping set) for which one or more of the check nodes have odd degree greater than 2. As such, the stopping sets are not a code property, but depend on the structure of $\mathbf{TG}(H)$. Stopping sets completely characterize the performance of the SPA (or BP) decoder on the BEC, since knowledge of the stopping sets enables us to improve the decoder by taking the proper action when the decoder stops. To allow decoding to resume, one would modify the structure of $\mathbf{TG}(H)$ such that one or more of the erasures are no longer in a stopping set. Such action may be to apply a permutation (on the columns of H) from $\mathrm{Aut}(\mathcal{C})$ [33, 34], or, simply, to add rows of H such that the sum (modulo 2) becomes a new linear combination (codeword of the dual code) which actually has the desired single edge into the erased positions. Another approach is to simply keep a set of distinct Tanner graphs, periodically changing graphs during decoding [35].

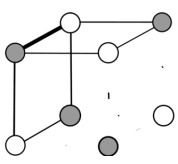
As an indication of the complexity of analyzing the SPA decoder on more general channels (such as the AWGN channel), even with the concrete tools derived from the BEC channel, it has been shown that

mirages exist in the decoding space, in which the decoder attempts to converge. These attract the iterative process towards what appears to be a solution, only to find once it gets nearer that there really is nothing there. These false solutions are described as *pseudocodewords* [36–38] and *near-codewords* are known to arise due to the local action of the SPA. A code is partitioned into a system of supercodes, in which a node is only concerned with the structure of its immediate surroundings. As a local subgraph can be the same in several distinct Tanner graphs for the same code, there is no guarantee that check nodes converge "their" subset of the variable bits (codeword positions) towards super-codewords which actually intersect with a global solution (codeword in the main code). In fact, it has been shown that pseudocodewords coincide with failed (i.e., non-integer) solutions of the *linear programming* decoder [39]. As a general class of problematic topologies in the Tanner graph, Richardson defined (a, b) *trapping sets* as any set of a variable nodes adjacent to b check nodes via an odd number of edges which, simply put, fail to converge after some high number of SPA iterations [40]. Despite this loose-fitting definition, the same trapping sets may be observed by repeated experiments, yet the link to the underlying Tanner graph is generally not well understood.

3.8 High-Density Parity-Check Codes

Although classical codes were found to have graphical representations ill fitted for iterative decoding, the question of whether iterative decoding might be modified such as to accomodate the use of strong classical codes, has recently received renewed interest. The aim of this research is to modify the dynamics of message passing so as to reduce the negative impact of various finite-length problems. Some of these problems are understood analytically – such as weight or density, cycles, stopping sets, and pseudocodewords, to name perhaps the most important.

In the context of applying soft-decison, iterative decoding to small blocklength, high-rate codes, the classical codes used are grouped under the category *high-density parity-check* (HDPC) codes. This term signifies the legacy from LDPC codes; that these codes are to be used for iterative SPA, or SPA-like, decoding, while at the same time pointing out the main distinction from LDPC codes. The various code constructions which qualify as HDPC codes is diverse, yet the trait of high density is the most important common qualifier. As we have seen, low-density is understood as a fraction of nonzero entries which is constant in



blocklength. Accordingly, *high-density* refers to a fraction which grows linearly in blocklength.

The concept of HDPC codes is quite recent, and begs the question why, in the context of iterative decoding, one would want *high-density* codes. The motivation for this area of research – to which this thesis is devoted – is to enjoy the benefits of both iterative decoding *and* small code size. Furthermore, classical code constructions are based on maximizing Hamming distance, and usually defined according to an algebraic or recursive structure which gives practical implementation in hardware (as opposed to storing a large, random matrix). Benefits include efficient encoding – one of the bottlenecks for LDPC schemes – as well as structural properties which can be used to enhance decoding. As we have discussed, stopping sets represent one of the pitfalls for iterative decoding. The algebraic, nonrandom design of HDPC codes generally ensures a nontrivial $\mathrm{Aut}(\mathcal{C})$. Applying a permutation from $\mathrm{Aut}(\mathcal{C})$ on the columns of H might then move one or more erasures away from a stopping set – without really changing the structure of $\mathbf{TG}(H)$ (see graph isomorphism in the following). Such dynamic decoding can be implemented efficiently on HDPC codes, by simply permuting the soft input vector rather than H. This decoding algorithm for HDPC codes is referred to as *random redundant decoding* (RRD) [33], and stems from the permutation decoder (PD) described earlier in this thesis. Rather than checking the weight of the syndrome after each permutation, and correcting errors by re-encoding, RRD attempts to correct errors via standard SPA iterations between permutations. The rationale of this decoder is not explicitly in terms of avoiding stopping sets, but rather that the *increased diversity* from permutations may have beneficial effects on SPA decoding in general. For instance, to name another example, the structure of small cycles in $\mathbf{TG}(H)$ is arguably amortized over codeword positions. In the following papers, we refer to this algorithm simply as SPA-PD.

The concept of employing code-preserving permutations to gain diversity forms one of two main categories of iterative SISO HDPC decoding. As mentioned, the *structure* of $\mathbf{TG}(H)$ is really preserved under permutations: This is especially easy to see if the permutations are on the soft input vector rather than H. This naturally identifies the other main category as containing algorithms based on changing the structure of $\mathbf{TG}(H)$. Again, obviously, the code must be preserved, so the permissible operations are, roughly, row-additions (online) and keeping several preprocessed matrices (or Tanner graphs) in memory

(offline). This is described as *multiple-bases belief propagation* (MBBP) [41], with numerous variations in the literature. A more in-depth survey is given in Paper III of this thesis.

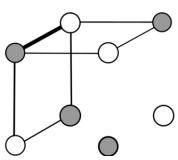
4 Edge-Local Complementation

The main topics of this thesis, as indicated by the title, are HDPC codes and a graph operation known as edge-local complementation (ELC). In short, the aim of this research is to investigate the specifications and benefits of using ELC to gain diversity during SISO decoding of HDPC codes. As with much other work on HDPC codes, our approach is a unification of the two categories outlined above; in one sense, as we will see, ELC amounts to row-additions on H, thus changing the basis during decoding, yet we also outline how ELC may, in fact, preserve the structure of H, amounting to permutations on H.

4.1 Some Graph Theory

Before going into the details on ELC, some basic concepts of graph theory are summarized. A *graph*, $\mathcal{G} = (\mathcal{W}, \mathcal{E})$, is defined as a set of nodes, $v \in \mathcal{W}$, connected via a set of edges, $\mathcal{E} \subset \mathcal{W} \times \mathcal{W}$. This graph is represented by an $n \times n$ *adjacency* matrix, $\mathcal{G}$. At some abuse of notation, we denote both the graph and its adjacency matrix by $\mathcal{G}$, and typically write $(v_i, v_j) \in \mathcal{G}$ (rather than $\mathcal{E}$). A *simple* graph is defined as an undirected (i.e., $(v_i, v_j) = (v_j, v_i)$) graph with no repeated edges or "self-loops" (i.e., $(v_i, v_i) \notin \mathcal{G}$). A graph is *bipartite* if $\mathcal{E}$ induces a split of $\mathcal{W}$ into two disjoint sets, $\mathcal{W} = \mathcal{U} \cup \mathcal{V}$, where $\mathcal{U} \cap \mathcal{V} = \emptyset$ and all edges are between $\mathcal{U}$ and $\mathcal{V}$. The weight of a graph is the number of edges, $|\mathcal{G}| = |\mathcal{E}|$. An adjacency matrix for an undirected graph is always symmetrical about the main diagonal. Thus, the weight of $\mathcal{G}$ is also equal to the number of nonzero entries in the upper (or lower) triangular submatrix. For convenience, we repeat some previous definitons here. The local neighborhood of a node v is the set of nodes adjacent to v, and is denoted by $\mathcal{N}_v$, while $\mathcal{N}_v^u$ is shorthand notation for $\mathcal{N}_v \setminus \{u\}$. Let $\mathcal{E}_{A,B}$ denote the subgraph induced by the nodes in $A \cup B$ – i.e., it is a set of $|\mathcal{E}_{A,B}|$ edges. Furthermore, $\mathcal{E}_{u,v}$ is shorthand notation for $\mathcal{E}_{\mathcal{N}_u^v, \mathcal{N}_v^u}$, the local neighborhood of the edge (u, v).

In this work, we consider *labelled* graphs, where each node is assigned a unique identificator; usually a number. The nodes in $\mathcal{U}$ and $\mathcal{V}$ are numbered separately, running from 0 to $|\mathcal{V}| - 1$, and from 0 to $|\mathcal{U}| - 1$.



The *canonical form* of a graph, denoted by $N(\mathcal{G})$, is defined as the graph resulting from sorting the rows and columns of $\mathcal{G}$ in some deterministic order. As such, any graph has a deterministic canonical form, but which may be common for several distinct graphs. In this case, we say that these graphs are structurally equivalent – or *isomorphic*. More specifically, two graphs $\mathcal{G}$ and $\mathcal{G}'$ are isomorphic *iff* there exists a permutation π on $\mathcal{W}$, such that $(u,v) \in \mathcal{G}$ *iff* $(\pi(u),\pi(v)) \in \mathcal{G}'$. We may thus determine isomorphism by comparing canonical form graphs, $N(\mathcal{G}) = N(\mathcal{G}')$. The specific sorting rules which facilitates to resolve such a problem are naturally very complex indeed and we only refer to the graph automorphism software package, NAUTY, which handles this in an efficient manner [42]. As a simpler concept, consider a parity-check matrix, H. In being a basis for $\mathcal{C}^{\perp}$, it is invariant under reordering of rows; this is an elementary row operation which preserves the code. The adjacency matrix for a Tanner graph is $\mathbf{TG}(H) = \left[\begin{smallmatrix} 0 & H \\ H^T & 0 \end{smallmatrix}\right]$, so we may describe two Tanner graphs, $\mathbf{TG}(H)$ and $\mathbf{TG}(H')$, as isomorphic *iff* the parity-check matrices, H and H', are *equal* after sorting the rows (in lexicographical order). The canonical form of a parity-check matrix, H, is written $R(H)$.

4.2 EDGE-LOCAL COMPLEMENTATION

ELC is defined on an edge of a simple (not necessarily bipartite) graph, $(u,v) \in \mathcal{G}$ [43]. The subgraph induced by (u,v) is the set of nodes adjacent to either u or v, or both. The remaining nodes of the graph (adjacent to neither u nor v) are ignored – we say they are *nonlocal* to (u,v). Let these three first sets be $A = \mathcal{N}_u^v \setminus \mathcal{N}_v^u$, $B = \mathcal{N}_v^u \setminus \mathcal{N}_u^v$, and $C = \mathcal{N}_u^v \cap \mathcal{N}_v^u$. The arbitrary edges between two sets of nodes is denoted by $A \sim B$. ELC on an edge (u,v) will *complement* any edges (edges replaced by nonedges, and vice versa) connecting all these sets, $A \nsim B$, $A \nsim C$, $B \nsim C$, followed by the swap of nodes u and v. This is illustrated in Fig. 9. In this sense, we say that ELC is a local operation as it only affects edges within a distance of one from the ELC edge, (u,v). The resulting graph, after ELC, is denoted by $\mathcal{G}_{(u,v)}$. ELC is a self-invertible operation as two ELC operations on the same edge is the identity. Since the edges connecting u to $\{A,C,v\}$, and those connecting v to $\{B,C,u\}$, are are not affected by ELC on (u,v), it follows that ELC preserves the connectedness of the graph.

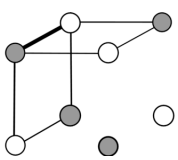
ELC was initially described in [43], as "complementation along an edge." Other authors have since referred to the operation as *pivot* on

(an edge of) a graph, to connotate the resulting transformation which is centered on an edge. Thus, the edge becomes a pivot (a fixed-point), around which the graph is transformed. The term "pivot" is used interchangeably with "ELC."

ELC has been used in various contexts, mainly graph-theoretical work. For instance, ELC has been used to classify the *orbits* of graphs (under ELC). The orbit of a graph is defined as the set of structurally distinct (nonisomorphic) graphs found via any sequence of ELC operations. Nonbipartite graphs are of interest in various contexts within *quantum* information theory. In the context of "conventional" coding theory, as we consider, we have seen how graphs are restricted exclusively to the bipartite case. This slightly simplifies the description of ELC, in that the common set, C, is always empty. Also, since the nodes in A and B must belong to opposite partitions, the complementation must preserve the *bipartiteness* of the graph – but, generally, the specific bipartition is changed: u and v change partitions. For bipartite graphs, ELC orbits have been used to classify all binary linear codes [44], and is known to generate all information sets (i.e., parity-check matrices, up to row-equivalence) of the code. The animation at the bottom of odd-numbered pages shows an example of ELC on the simple 8-node graph corresponding to the $[8, 4, 4]$ extended Hamming code. This code has the very special property of having only a single graph in its orbit, such that any ELC will be an *iso-ELC*. In this example, note that the beginning and ending graphs both take the shape of a "cube," yet with a different labelling of nodes (note the nodes swapped at the end). In this sense, this code is *ELC-preserved*.[6]

The effect of (bipartite) ELC on the underlying (parity-check) matrix is to reduce the corresponding column to a systematic (weight-1) column. This is precisely as in the definition of Gaussian elimination (GE), where the single remaining nonzero entry is referred to as the pivot of the column, or a pivot position. Furthermore, the fact that ELC can be implemented as row additions on a matrix (as in GE), proves that ELC preserves the code. As we have seen, there is a one-to-one relationship between a Tanner graph $\mathbf{TG}(H)$ and a simple bipartite graph, $\mathcal{G}$. This is an important observation, which gives the foundations for applying graph operations (i.e., ELC) to improve decoding. To our knowledge, ours is the only use of ELC as an aid to improve graph-based decoding.

[6]The fact that ELC is (self) invertible is apparent by flipping the animation in the opposite direction. □



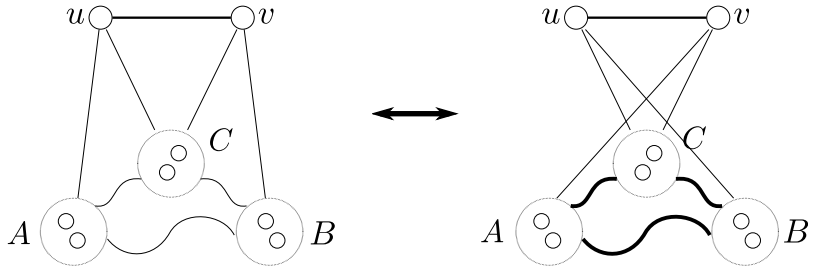


Fig. 9: *ELC on edge (u, v) of a simple graph. Bold links mean that the edges connecting the two sets have been complemented. This graph may be a subgraph of a larger graph, in which case the rest of the graph remains unchanged.*

The ELC operation is related to another simple graph operation, *local complementation* (LC). This operation acts on a node, v, rather than an edge, complementing the neighborhood, i.e., $v \nsim \mathcal{N}_v$. Originally defined by de Fraysseix [45], this operation was also studied by Bouchet [46]. In fact, the ELC operation may be implemented by a sequence of LC operations: $\mathrm{ELC}(u, v) = \mathrm{LC}(u)\,\mathrm{LC}(v)\,\mathrm{LC}(u)$. *LC orbits* of graphs have been used to classify self-dual *additive codes* over $\mathrm{GF}(4)$ (i.e., quantum codes) [47]. Furthermore, *interlace polynomials,* defined with respect to ELC and LC [48, 49], have been used to study problems in DNA sequencing [50]. The LC operation also preserves connectedness, but not bipartiteness.

5 Contributions of this Thesis

The novel work presented in this thesis consists of five papers. I will first give a short overview of each paper. All the work is presented as joint work, and I have made an attempt to point out my main contributions to each paper. This is an inherently difficult task, as the cooperation in this group has been very tight. As a general comment, I should emphasize that all papers where I am listed as the first author were written by me, while the coauthors have (mainly) proof-read and made general adjustments.

Paper I

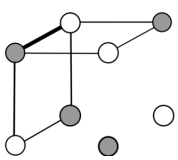
This paper introduces our work on using ELC to improve an iterative, SISO decoder. This work considers the general graphical requirements for ELC to be an isomorphic (iso-ELC) operation. The $[8, 4, 4]$ extended Hamming code is used as a toy code and a proof of concept for the SPA-ELC decoding algorithm. This code is ELC-preserved, which means that the resulting graphs under ELC are equal up to node permutations. This gives the benefit of using randomly placed ELC operations and still achieve the desired effect of using iso-ELC operations to give diversity during decoding [51].

For this work, we consider both a serial and a parallel, list version of the decoder, and show an improvement over standard *flooding* SPA decoding. We call these the random adaptive decoder (RAD), and the list adaptive decoder (LAD). Alternative approaches to implementing iso-ELC operations during decoding have significant drawbacks, involving precomputing and storing a possibly large list of ELC sequences which preserve graph structure. Furthermore, as these depend on the Tanner graph, it would be required to compensate for changes in labelling resulting from ELC operations, or, alternatively, undoing the previous iso-ELC operation to return to the initial graph such that the next operation may be applied. The results and experiences from the extended Hamming code and the RAD led us to investigate further the use of ELC in decoding of HDPC codes.

My Contributions to Paper I

I have implemented a simulations setup to test the performance of the isomorphic SPA-ELC decoder. This setup consists mainly of an AWGN noise simulator; a Tanner graph implementation suited for ELC; an SPA implementation working in the LLR domain; and a database system for keeping track of simulations data (when run in parallel on a Linux cluster). The Tanner graph view of ELC is based on my work, and the knowledge of the extended Hamming code being ELC-preserved led us to look into how and when ELC is a graph isomorphism (and, correspondingly, a code automorphism). Using my software, I designed and implemented decoding algorithms (LAD and RAD) to test the concept of isomorphic SPA-ELC decoding.

Knudsen, J. G., Riera, C., Parker, M. G., Rosnes, E.: Adaptive soft-decision decoding using edge local complementation. In *Proc. Second Int. Castle Meeting on Coding Theory and Applications, LNCS 5228*, pp. 82–94. Castillo de la Mota, Medina del Campo, Spain, Sep. 2008.



PAPER II

The effect of the SPA-ELC decoder is extended to larger, more general codes. The codes considered are still small and strongly structured, from the perspective of iterative decoding which is typically concerned with large and sparse codes; i.e., we consider HDPC codes rather than LDPC codes. The desired effect is, here, to test the effect of the SPA-ELC decoder when random ELC does generally not preserve graph structure (nonisomorphic). As a natural step from the extended $[8, 4, 4]$ Hamming code, we consider the $[24, 12, 8]$ extended Golay code, as well as two EQR codes, $[48, 24, 12]$ and $[104, 52, 20]$. The extended Golay code is *nearly* ELC-preserved: ELC gives only two structurally distinct graphs, which, similarily to the case in Paper I, gives a benefit for SPA decoding and a performance comparable to SPA-PD [33]. The length-48 and 104 EQR codes, on the other hand, have vast such orbits of graphs, containing graphs of varying weight [44]. Increased weight is well known to have an adverse effect on decoding, affecting both error-rate performance (increase in number of small cycles), and complexity (larger input space for each node to process). As such, the performance of SPA-ELC cannot keep up with that of SPA-PD as we go on to bigger codes (at least for the specific EQR codes considered in this work). However, the gain to SPA decoding remains large, and increasing for larger codes (at FER 10^{-4}, we find a gain of over 1 dB for the EQR48 code, and over 2dB for EQR104). This indicates both that the SPA-ELC algorithm scales with blocklength, but also that SPA decoding does not.

An important aspect of many SISO HDPC decoding algorithms is that the extrinsic contribution (as opposed to the channel input) to the LLRs in variable nodes is scaled down by a *damping coefficient*, $0 < \alpha(t) < 1$, which is an nondecreasing function of the iteration number, $0 \leq t \leq \tau$ [34]. As such, $\alpha(t)$ reflects our confidence in the soft information produced by the decoder at iteration t. This technique derives from the iterative decoding problem interpreted as a *gradient descent* algorithm, where the solution takes the form of a minimum, and $\alpha(t)$ controls the step width in the convergence (descent). The damping used in [33] is a global operation, affecting the entire Tanner graph, so we propose an edge-local variant which damps only those edges affected by ELC. The performance of SPA-ELC with (and without, as a simplified algorithm) such local damping is compared to the performance of SPA-PD and SPA, both in terms of FER performance

and decoder complexity (in average number of SPA messages computed per frame).

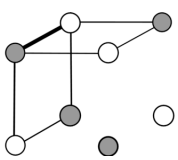
My Contributions to Paper II

Having determined suitable types of codes (i.e., graphs) for ELC decoding in Paper I, I was able to produce positive results also for random (non-isomorphic) ELC. For this work, I implemented the SPA-PD algorithm [33], which we have seen corresponds to iso-ELC, to serve as a benchmark for decoder performance. After calibrating the setup, I used the same framework (loops I_1, I_2, and I_3) to implement our SPA-ELC decoder – with and without edge-local damping. (Many variants were tested and found inefficient, including the use of global damping with ELC.) Also, as preprocessing, the parity-check matrices used for decoding were optimized (using several approaches) on both weight and number of 4-cycles. Extensive FER simulations were then maintained on a Linux cluster.

Knudsen, J. G., Riera, C., Danielsen, L. E., Parker, M. G., Rosnes, E.: Iterative decoding on multiple Tanner graphs using random edge local complementation. In *Proc. IEEE Int. Symp. Inform. Theory*, pp. 899–903. Seoul, Korea, Jun./Jul. 2009.

Paper III

Inspired by the success of SPA-ELC on the special Hamming and Golay codes, the third paper explores the structural properties of the Tanner graph which affect the performance of ELC and SPA-ELC decoding. In an abbreviated version of this paper [52] (omitted), we determine the specific subgraphs on which ELC will not increase the weight of the graph beyond a given threshold value – which we define as *weight-bounding ELC* (WB-ELC). Based on the SPA-ELC and SPA-PD algorithms, a generalized SISO HDPC decoder framework is outlined, which is used to implement a novel SPA-WBELC decoder. In addition to the theoretical observations on WB-ELC, the main results of the abbreviated version is to show a gain in error-rate performance of SPA-WBELC over SPA-PD. The strength of the SPA-PD algorithm is strongly affected by the size of $\mathrm{Aut}(\mathcal{C})$. For this work, we choose HDPC codes with strong structural properties (most importantly, *extremal* in terms of minimum distance) [10, 11], but with relatively small $\mathrm{Aut}(\mathcal{C})$. Especially, a gain is shown for codes where $\mathrm{Aut}(\mathcal{C})$ is trivial, where SPA-PD reduces to SPA.



This work is extended to a fulltext version, which is included as Paper III. The WB-ELC observations are related to our previous work, and presented as a generalization of iso-ELC, as discussed in Paper I. To formalize the theoretical background for WB-ELC, this extended work discusses the concept of iso-ELC and its relation to $\mathrm{Aut}(\mathcal{C})$ in greater mathematical detail. In this context, the orbit of the code is generalized to a set of equivalent *isomorphic orbits,* each of size $|\mathrm{Aut}(\mathcal{C})|/|\mathcal{D}|$, centered on each structurally distinct graph in the ELC orbit of the code. A notion of Tanner graph *row-equivalence* is defined, which gives subgroups $\mathcal{D}_H \preceq \mathrm{Aut}(\mathcal{C})$ (varying with H), consisting of *trivial* permutations which preserve the Tanner graph and have no effect on iterative decoding. This gives rise to a canonical Tanner graph form, based on sorting the rows of H. The one-to-one relationship between permutations in $\mathrm{Aut}(\mathcal{C})$ and iso-ELC operations (i.e., sequences of 0 to $\max(k,\ n-k)$ ELC operations) is proven via an algorithm to find the minimum-length sequence of ELC operations connecting two Tanner graphs for the same code (including also nonisomorphic graphs from the same orbit, for which the ELC sequence does *not* correspond to a permutation from $\mathrm{Aut}(\mathcal{C})$). The classification of $\mathrm{Aut}(\mathcal{C})$ into trivial and nontrivial permutations, denoted $\mathcal{D}_H$ and $\mathcal{K}_H$, respectively, where $\mathrm{Aut}(\mathcal{C}) = \mathcal{D}_H \circ \mathcal{K}_H$, depends on the parity-check matrix, H. In systematic form, this depends on whether the action of a permutation is confined to within $\mathcal{I}$ and $\mathcal{P}$, or not. We explore the structure of the subset of nontrivial permutations, with an emphasis on whether or not this can be written as a group (for any parity-check matrix for the code).

The paper ends with a discussion on several applications of WB-ELC operations, with an emphasis on the SPA-WBELC decoding algorithm. Other applications, which are used in a preprocessing stage of the SPA-WBELC decoder, are an heuristic to optimize (reduce) the weight of a parity-check matrix in systematic form, and an algorithm to traverse the *bounded-weight sub-orbit* of the code (i.e., the orbit of graphs under WB-ELC, with some threshold). All applications are centered around a base algorithm to enumerate all WB-ELC sequences, given a graph and a threshold. The search complexity is bounded, and verified by simulations. Several instances of this base algorithm are then proposed to improve the overhead of determining a random WB-ELC sequence (typically for real-time use). The complexity of these is compared by simulations on various self-dual HDPC codes.

My Contributions to Paper III

The specific *cases* for WB-ELC, at depth 1 and depth 2, were implemented using my graph software, so that these could be checked and adjusted to the form presented in the paper. This work led to an understanding of the relationship between the cases, and what would be a practical (nested) implementation of the enumeration algorithm. This was then simulated extensively, to verify and adjust the complexity bound deviced by Danielsen. Simulations on the performance of the SPA-WBELC decoder were run, after choosing parameters based on empirical data. Furthermore, in being a generalization of WB-ELC, various observations on iso-ELC, which we have worked on after Paper I, fit naturally in the context of this extended paper. An algorithm by Rosnes to convert a permutation from $\mathrm{Aut}(\mathcal{C})$ to an iso-ELC sequence, was generalized to handle also the case of nonisomorphic graphs. The Tanner graph canonical form stems from B. D. McKay's Nauty graph format (encoding rows of a binary adjacency matrix, $\mathcal{G}$, as integers), which I modified to a specific Tanner graph format (encoding only H – the upper-right quadrant of $\mathcal{G}$). This way, Tanner graphs can be represented by a unique string of Ascii characters, for ease of comparison and storage on disk.

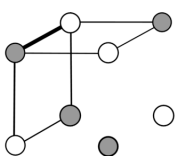
Knudsen, J. G., Riera, C., Danielsen, L. E., Parker, M. G., Rosnes, E.: Random edge-local complementation with applications to iterative decoding of HDPC codes. Tech. Report no. 395, Department of Informatics, University of Bergen, Norway, Aug. 2010.

Submitted to IEEE Trans. Inform. Theory, 2010.

The material in this paper was presented in part at the International Zürich Seminar on Communications (IZS), Zürich, Switzerland, Mar. 2010, and at the Workshop on Linear Programming and Message-Passing Approaches to High-Density Parity-Check Codes and High-Density Graphical Models, Tel Aviv, Israel, Mar. 2010.

Paper IV

This paper differs from the previous three in that the ELC-stage of the proposed decoder is *adaptive*, rather than random. This work is based on the adaptive belief propagation (ABP) decoder [53], in which Gaussian elimination (GE) is used in an attempt to reduce the columns of H corresponding to the *least reliable* parity set to an identity matrix. The reliability measure is simply the LLR magnitude. From the perspective of iterative decoding, variable nodes of degree one (not counting the input "half edge") are practically isolated in terms of message-passing,



in the sense that they receive information from their single check node, without affecting the rest of the graph (sending only the input message in return). Furthermore, these nodes are not part of any cycles. The contribution of our proposed ABP-ELC decoder is to implement the adaptive stage using $p \ll n - k$ ELC operations, for a complexity improvement over the GE-stage of ABP. This is an important gain, as a GE-stage is considered a heavy overhead. Our experience with the SPA-ELC decoder suggests a significant reduction in the number of ELC operations is obtainable (p is typically configured between 1 and 3). Also, we extend the heuristic used to determine good ELC locations. This is based on the fact that each ELC operation swaps a pair of positions between $\mathcal{I}$ and $\mathcal{P}$ (in H). The best choice, in a greedy sense, should ensure not only that the weakest position is swapped into $\mathcal{P}$, but also that a reliable position is swapped into $\mathcal{I}$.

The technique of edge-local damping is extended to a somewhat more involved form of local damping, which we refer to as *local neighborhood damping*, scaling all edges adjacent to all variable nodes affected by ELC. We also observe a gain in FER by SPA-ABP over ABP. This gain is verified by simulation data, for both the length-48 EQR code, as well as a binary image of the $[31, 25, 7]$ RS code over $\mathrm{GF}(2^5)$. Efficient soft-decision decoding of RS codes is considered an important problem in coding theory [54]. The observed gain is accredited to the generalized damping scheme, as well as the benefit of preserving soft information by avoiding global operations. Specifically, we avoid global damping which involves discarding soft information on all edges. Interestingly, this more involved damping has a beneficial effect on the high-rate RS code, but not for the self-dual EQR code.

My Contributions to Paper IV

The improved heuristic was my idea, and the proposed adaptive ELC-stage (to replace the GE-stage) was designed and tested by me. Both implemented using the SISO HDPC framework, the ABP-ELC decoder is simulated and compared against the ABP decoder. First, a calibration was made to ensure that our implementation would reproduce the data reported in [53]. The reported gain was reached through fine-tuning of ABP-ELC parameters; mainly, determining an "optimum" choice of p through repeated simulations.

Knudsen, J. G., Riera, C., Danielsen, L. E., Parker, M. G., Rosnes, E.: Improved adaptive belief propagation decoding using edge-local complementation. In *Proc. IEEE Int. Symp. Inform. Theory*, pp. 774–778. Austin, Texas, Jul. 2010.

Paper V

This paper stems from the work on enumerating codes and graphs in terms of their ELC orbit, focused on the extreme case of orbit size 1. Any ELC operation will – necessarily – give the same graph up to isomorphism, so the graph and code are ELC-preserved. This work explores the graph theoretical structural properties of such graphs, identifying all instances of blocklength up to 16. From this exhaustive search, the resulting codes include the Hamming codes (and their extensions), as well as some trivial star graphs. The remaining codes found are all categorized as *expansions* of either Hamming codes or star graphs, to form larger graphs. From a coding theory perspective, we conclude that all corresponding codes from the expansions do not have favorable minimum distance.

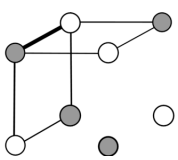
We also consider non-bipartite ELC-preserved graphs, identify all such graphs of up to 12 nodes, and describe expansion constructions. Although non-bipartite graphs do not have immediate applications in error-correction, these graphs are interesting from a graph theoretical point of view.

My Contributions to Paper V

The focus on ELC-preserved graphs was, in part, motivated by my request for a larger ELC-preserved graph than the $[8, 4, 4]$ extended Hamming code, on which SPA-PD can be easily implemented via random ELC operations (one does not even need to know $\mathrm{Aut}(\mathcal{C})$). Collaboration on iso-ELC for Paper I and Paper III helped clarify the situation and requirements for the ELC-preserved case. Furthermore, simple testing and simulations revealed the challenges in applying "Hamming expansion" codes for iterative decoding, namely $d_{\mathbf{min}} = 4$ (where also the numvber of minimum-weight codewords is $\mathcal{O}(n)$), and many 4-cycles.

Danielsen, L. E., Parker, M. G., Riera, C., Knudsen, J. G.: On graphs and codes preserved by edge local complementation, 2010. arXiv:1006.5802.

This paper was presented at the 10th Nordic Combinatorial Conference (NORCOM 2010), Reykjavik, Iceland, May 2010.



REFERENCES

[1] SHANNON, C. E.: A mathematical theory of communication. *Bell System Tech. J.* 27, 379–423, 623–656, Jul. and Oct. 1948.

[2] HUFFMAN, W. C.: Handbook of coding theory. In PLESS, V. S., HUFFMAN, W. C. (eds.), *Handbook of Coding Theory*. Elsevier, North-Holland, Amsterdam, 1998.

[3] ELIAS, P.: Coding for two noisy channels. In *Proc. 3rd London Symp. Inf. Theory*, pp. 61–76. London, England, 1955.

[4] RICHARDSON, T., URBANKE, R. L.: *Modern Coding Theory*. Cambridge University Press, 2008.

[5] PROAKIS, J. G.: *Digital Communications*. McGraw-Hill, 2000.

[6] KNUTH, D. E.: Two notes on notation. *Am. Math. Monthly* 99(5), 403–422, 1992.

[7] LIN, S., COSTELLO, D. J., JR.: *Error Control Coding*. Pearson, Prentice Hall, 2004.

[8] HAMMING, R. W.: Error detecting and error correcting codes. *Bell System Tech. J.* 26, 147–160, 1950.

[9] MACWILLIAMS, F. J., SLOANE, N. J. A.: *The Theory of Error-Correcting Codes*, chap. 16. North Holland, 1977.

[10] HARADA, M.: New extremal self-dual codes of lengths 36 and 38. *IEEE Trans. Inform. Theory* 45(7), 2541–2543, Nov. 1999.

[11] GULLIVER, T. A., HARADA, M.: Classification of extremal double circulant self-dual codes of lengths 64 to 72. *Des. Codes Cryptogr.* 13(3), 257–269, Mar. 1998.

[12] MACWILLIAMS, F. J.: Permutation decoding of systematic codes. *Bell System Tech. J.* 43, 485–505, 1964.

[13] BERROUX, C., GLAVIEUX, A., THITIMAJSHIMA, P.: Near Shannon limit error-correcting coding and decoding: Turbo codes. In *Proc. IEEE Int. Conf. Commun.*, pp. 1064–1070. Geneva, Switzerland, May 1993.

[14] GALLAGER, R. G.: Low-density parity-check codes. *IRE Trans. Inform. Theory* 8(1), 21–28, Jan. 1962.

[15] Tanner, R. M.: A recursive approach to low complexity codes. *IEEE Trans. Inform. Theory* 27(5), 533–547, Sep. 1981.

[16] MacKay, D. J. C., Neal, R. M.: Good codes based on very sparse matrices. In *Proc. 5th IMA Conf. on Cryptography and Coding LNCS 1025*, pp. 100–111. Royal Agricultural College, Cirencester, UK, Dec. 1995.

[17] Chung, S., Forney, G. D., Jr., Richardson, T. J., Urbanke, R.: On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit. *IEEE Commun. Lett.* 2, 58–60, Feb. 2001.

[18] MacKay, D. J. C.: Good error-correcting codes based on very sparse matrices. *IEEE Trans. Inform. Theory* 45(2), 399–431, Mar. 1999.

[19] Kschischang, F. R., Frey, B. J., Loeliger, H. A.: Factor graphs and the sum-product algorithm. *IEEE Trans. Inform. Theory* 47(2), 498–519, Feb. 2001.

[20] McEliece, R., MacKay, D., Cheng, J.-F.: Turbo decoding as an instance of Pearl's belief propagation algorithm. *IEEE J. Sel. Areas Commun.* 16(2), 140–152, feb. 1998.

[21] MacKay, D. J. C.: *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

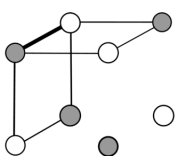
[22] Knudsen, J. G.: *Randomised Construction and Dynamic Decoding of LDPC Codes*. Master's thesis, University of Bergen, Bergen, Norway, 2006.

[23] Forney, G. D., Jr.: Codes on graphs: Normal realizations. *IEEE Trans. Inform. Theory* 47(2), 520–548, Feb. 2001.

[24] Loeliger, H.-A., Lustenberger, F., Helfenstein, M., Tarköy, F.: Probability propagation and decoding in analog VLSI. *IEEE Trans. Inform. Theory* 47(47), 837–843, 2000.

[25] Bahl, L., Cocke, J., Jelinek, F., Raviv, J.: Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Trans. Inform. Theory* 20(2), 284–287, January 2003.

[26] Rosnes, E.: Iterative soft decoding of binary linear codes using a generalized Tanner graph, 2008. Preprint.



[27] Tanner, R. M., Sridhara, D., Sridharan, A., Fuja, T. E., Costello, D. J., Jr.: LDPC block and convolutional codes based on circulant matrices. *IEEE Trans. Inform. Theory* 50(12), 2966–2984, Dec. 2004.

[28] Luby, M. G., Mitzenmacher, M., Shokrollahi, M. A., Spielman, D. A.: Improved low-density parity-check codes using irregular graphs. *IEEE Trans. Inform. Theory* 47, 585–598, 2001.

[29] Richardson, T. J., Urbanke, R. L.: The capacity of low-density parity-check codes under message-passing decoding. *IEEE Trans. Inform. Theory* 47(2), 599–618, Feb. 2001.

[30] Chung, S.-Y., Richardson, T. J., Urbanke, R. L.: Analysis of sum-product decoding of low-density parity-check codes using a gaussian approximation. *IEEE Trans. Inform. Theory* 47(2), 657–670, 2001.

[31] ten Brink, S.: Convergence behavior of iteratively decoded parallel concatenated codes. *IEEE Trans. Commun.* 49(10), 1727–1737, oct. 2001.

[32] Di, C., Proietti, D., Telatar, I. E., Richardson, T. J., Urbanke, R. L.: Finite-length analysis of low-density parity-check codes on the binary erasure channel. *IEEE Trans. Inform. Theory* 48(6), 1570–1579, Jun. 2002.

[33] Halford, T. R., Chugg, K. M.: Random redundant iterative soft-in soft-out decoding. *IEEE Trans. Commun.* 56(4), 513–517, Apr. 2008.

[34] Jiang, J., Narayanan, K. R.: Iterative soft decoding of Reed-Solomon codes. *IEEE Commun. Lett.* 8(4), 244–246, Apr. 2004.

[35] Hehn, T., Huber, J. B., Laendner, S., Milenkovic, O.: Multiple-bases belief-propagation for decoding of short block codes. In *Proc. IEEE Int. Symp. Inform. Theory*, pp. 311–315. Nice, France, Jun. 2007.

[36] Forney, G. D., Jr., Koetter, R., Kschischang, F. R., Reznik, A.: On the effective weights of pseudocodewords for codes defined on graphs with cycles. In Marcus, B., Rosenthal, J. (eds.), *Codes, Systems, and Graphical Models*, vol. 123 of *IMA Vol. Math. Appl.*, pp. 101–112. Springer Verlag, 2001.

[37] Koetter, R., Vontobel, P. O.: Graph-covers and iterative decoding of finite length codes. In *Proc. 3rd Int. Symp. on Turbo Codes & Related Topics*, pp. 75–82. Brest, France, Sep. 2003.

[38] Vontobel, P. O., Koetter, R.: Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes, 2005. arXiv:cs/0512078.

[39] Feldman, J., Wainwright, M., Karger, D.: Using linear programming to decode binary linear codes. *IEEE Trans. Inform. Theory* 51(3), 954–972, mar. 2005.

[40] Richardson, T.: Error floors of LDPC codes. In *Proc. 41st Annual Allerton Conf. on Commun., Control, and Computing*, pp. 1426–1435. Monticello, IL, Oct. 2003.

[41] Hehn, T., Huber, J., Milenkovic, O., Laendner, S.: Multiple-bases belief-propagation decoding of high-density cyclic codes. *IEEE Trans. Commun.* 58(1), 1–8, Jan. 2010.

[42] McKay, B. D.: Nauty; software for computing automorphism groups of graphs and digraphs. Web page, 2007. `http://cs.anu.edu.au/people/bdm/nauty/`.

[43] Bouchet, A.: Isotropic systems. *European J. Comb.* 8, 231–244, Jul. 1987.

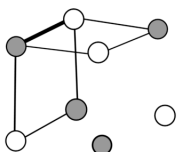
[44] Danielsen, L. E., Parker, M. G.: Edge local complementation and equivalence of binary linear codes. *Des. Codes Cryptogr.* 49(1-3), 161–170, Dec. 2008.

[45] de Fraysseix, H.: Local complementation and interlacement graphs. *Discrete Mathematics* 33(1), 29–35, 1981.

[46] Bouchet, A.: Graphic presentations of isotropic systems. *J. Comb. Theory, Series B* 45(1), 58–76, 1988.

[47] Danielsen, L. E., Parker, M. G.: On the classification of all self-dual additive codes over GF(4) of length up to 12. *J. Comb. Theory, Series A* 113(7), 1351–1367, Oct. 2006.

[48] Arratia, R., Bollobás, B., Sorkin, G. B.: The interlace polynomial of a graph. *J. Comb. Theory, Series B* 92(2), 199–233, Nov. 2004.



[49] Aigner, M., Van der Holst, H.: Interlace polynomials. *Linear Algebra Appl.* 337, 11–30, 2004.

[50] Arratia, R., Bollobás, B., Coppersmith, D., Sorkin, G. B.: Euler circuits and DNA sequencing by hybridization. *Discrete Appl. Math.* 104(1-3), 63–96, 2000.

[51] Knudsen, J. G., Riera, C., Parker, M. G., Rosnes, E.: Adaptive soft-decision decoding using edge local complementation. In *Proc. Second Int. Castle Meeting on Coding Theory and Applications, LNCS 5228*, pp. 82–94. Castillo de la Mota, Medina del Campo, Spain, Sep. 2008.

[52] Knudsen, J. G., Riera, C., Danielsen, L. E., Parker, M. G., Rosnes, E.: On iterative decoding of HDPC codes using weight-bounding graph operations. In *Proc. Int. Zürich Seminar on Commun.*, pp. 98–101. Zürich, Switzerland, Mar. 2010.

[53] Jiang, J., Narayanan, K. R.: Iterative soft-input soft-output decoding of Reed-Solomon codes by adapting the parity-check matrix. *IEEE Trans. Inform. Theory* 52(8), 3746–3756, Aug. 2006.

[54] Koetter, R., Vardy, A.: Algebraic soft-decision decoding of Reed-Solomon codes. *IEEE Trans. Inform. Theory* 49(11), 2809–2825, Nov. 2003.

PAPER I

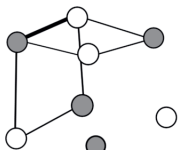
ADAPTIVE SOFT-DECISION ITERATIVE DECODING USING EDGE LOCAL COMPLEMENTATION *

Joakim Grahl Knudsen Constanza Riera

Matthew G. Parker Eirik Rosnes

*Knudsen, J. G., Riera, C., Parker, M. G., Rosnes, E.: Adaptive soft-decision decoding using edge local complementation. In *Proc. Second Int. Castle Meeting on Coding Theory and Applications, LNCS 5228*, pp. 82–94. Castillo de la Mota, Medina del Campo, Spain, Sep. 2008.

The notation in this chapter has been edited to make it consistent with the thesis.

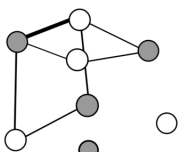


ADAPTIVE SOFT-DECISION ITERATIVE DECODING USING EDGE LOCAL COMPLEMENTATION

Joakim Grahl Knudsen Constanza Riera

Matthew G. Parker Eirik Rosnes

We describe an operation to dynamically adapt the structure of the Tanner graph used during iterative decoding. Codes on graphs–most importantly, low-density parity-check codes–exploit randomness in the structure of the code. Our approach is to introduce a similar degree of controlled randomness into the operation of the message-passing decoder, to improve the performance of iterative decoding of classical structured (i.e., non-random) codes for which strong code properties are known. We use ideas similar to Halford and Chugg (IEEE Trans. on Commun., April 2008), where permutations on the columns of the parity-check matrix are drawn from the automorphism group of the code, $\mathrm{Aut}(\mathcal{C})$. The main contributions of our work are: 1) We maintain a graph-local perspective, which not only gives a low-complexity, distributed implementation, but also suggests novel applications of our work, and 2) we present an operation to draw from $\mathrm{Aut}(\mathcal{C})$ such that graph isomorphism is preserved, which maintains desirable properties while the graph is being updated. We present simulation results for the additive white Gaussian noise channel, which show an improvement over standard sum-product algorithm decoding.



1 INTRODUCTION

Inspired by the success of iterative decoding of low-density parity-check codes (LDPC) codes, originally introduced by Gallager [1] and later rediscovered in the mid 1990's by MacKay and Neal [2], on a wide variety of communication channels, the idea of iterative, soft-decision decoding has recently been applied to classical algebraically constructed codes in order to achieve low-complexity Belief Propagation decoding [3–5]. Both Reed-Solomon and Bose-Chaudhuri-Hocquenghem (BCH) codes have been considered in the context of iterative decoding. Certain algebraically constructed bipartite graphs are known to exhibit good code properties, such as large minimum distance and a non-trivial automorphism group. However, these typical *classical properties* do not necessarily lend themselves well to modern graph-based coding theory. Factors which influence the performance of iterative, soft-decision decoders are pseudo-codewords [6], stopping and trapping sets [7, 8], sparsity, girth, and degree distributions [9]. Structural weaknesses of graphical codes are inherent to the particular parity-check matrix, H, which can be said to implement the code in the decoder. This matrix is a non-unique $(n-k)$-dimensional basis for the null space of the code, $\mathcal{C}$, which, in turn, is a k-dimensional subspace of $\{0,1\}^n$. Although any basis (for the dual code, $\mathcal{C}^\perp$) is a parity-check matrix for $\mathcal{C}$, their performance in decoders is not uniform. In this work, we assume that H is of full rank and in *standard* $[I \mid P]$-form, where I is the identity matrix.

We propose a class of adaptive decoders which facilitate message-passing on classical linear codes, by taking advantage of (non-trivial) graph structure. It is well known that H can be mapped into a bipartite (Tanner) graph, $\mathbf{TG}(H)$, which is described by its adjacency matrix, $\left[\begin{smallmatrix} 0 & H \\ H^T & 0 \end{smallmatrix}\right]$. With H being in standard form, a specific information set (on the codeword positions) is implied. We will refer to bit nodes (i.e., columns of H) corresponding to I and P as *parity* and *information* nodes, respectively,[1] and rows of H correspond to *constraint* (check) nodes. Using a localized, low-complexity graph edge-operation, we update the parity-check matrix, but still stay within the automorphism group of the code, $\mathrm{Aut}(\mathcal{C})$. Thus, the graph update rule can be viewed as a particular relabelling (isomorphism) of the bit nodes. Furthermore, by selectively or randomly shifting sensitive substructures (e.g., short

[1] Note that these terms refer to the generator matrix of the code, $G \triangleq [P^T \mid I_k]$.

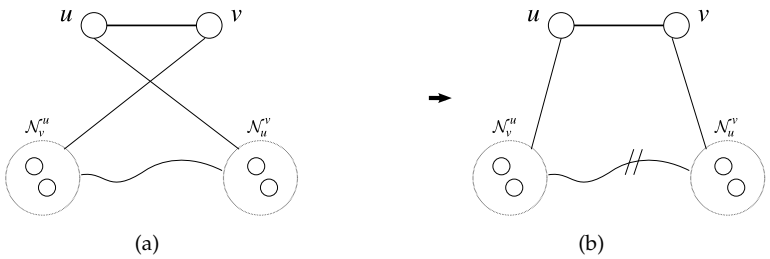


Fig. 1: *ELC on edge (u, v) of a bipartite graph. Doubly slashed links mean the edges connecting two sets have been complemented.*

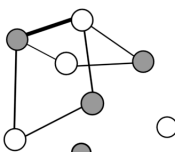
cycles, or weight-1 nodes) within the graph, we aim to influence the flow of extrinsic information through $\mathbf{TG}(H)$ in a way helpful to the decoding process.

In a recent paper by Halford and Chugg, "random redundant iterative decoding" is achieved by applying permutations drawn at random from $\mathrm{Aut}(\mathcal{C})$ [5]. Rather than applying these permutations to H, the same effect is achieved by permuting the soft input vector. While their strategy is perceived to be a series of global updates, our approach achieves a similar effect by using only local updates on $\mathbf{TG}(H)$. In our characterization of locality, we assume that an edge can not 'see' beyond a radius of a constant number of edges. Similarly to [5], permutations can be drawn from a precomputed list input to the decoder. However, our distributed approach also allows us to dispense with precomputation, to realize a completely distributed and local graph update rule, which, nevertheless, keeps the series of graphs generated within $\mathrm{Aut}(\mathcal{C})$.

2 Edge Local Complementation

The operation of edge local complementation (ELC) [10–12], also known as *pivot*, is a local operation on a simple graph. Fig. 1(a) shows $\mathcal{G}_{\mathcal{N}_u \cup \mathcal{N}_v}$, the local subgraph of a bipartite graph induced by nodes u, v, and their disjoint neighborhoods which we denote $\mathcal{N}_u^v \triangleq \mathcal{N}_u \setminus \{v\}$ and $\mathcal{N}_v^u \triangleq \mathcal{N}_v \setminus \{u\}$, respectively.

ELC on a bipartite graph is described as the complementation of edges between these two sets; $\forall\ v' \in \mathcal{N}_u^v,\ u' \in \mathcal{N}_v^u$, check whether edge $(u', v') \in \mathcal{G}$, in which case it is deleted (otherwise, it is created). Finally,

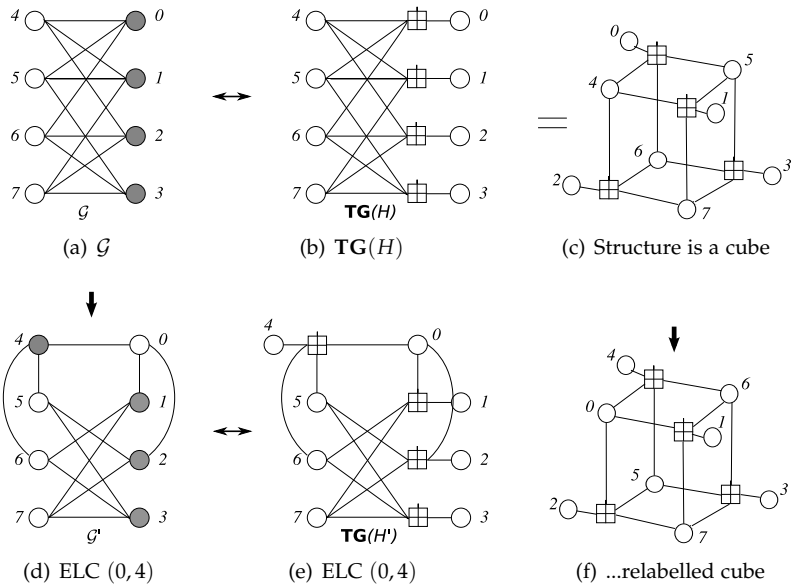


the edges adjacent to u and v are swapped. As such, ELC updates the set of constraints (rows of H) by changing the edges of $\mathbf{TG}(H)$, whereas nodes are invariant. The complexity of the graph-based algorithm is $\mathcal{O}(\deg(u)\deg(v))$. The fact that ELC amounts to row additions assures that the code is preserved. In the following, we use the notation $\mathcal{G}^i$ to denote a graph $\mathcal{G}$ that has been subject to i ELC operations (similarly for H^i).

Consider the simple, n-node bipartite graph described by $\mathcal{G} = \begin{bmatrix} 0 & P \\ P^T & 0 \end{bmatrix}$. This graph is related to $\mathbf{TG}(H)$ by the abstraction of degree-1 parity nodes, as shown in Figs. 2(a) - 2(b). Keeping track of the bipartition of $\mathcal{G}$ (which changes due to the swap), means we can obtain an associated parity-check matrix, H^1, for $\mathcal{C}$ by mapping grey nodes onto rows (constraint nodes), and white nodes to columns (bit nodes), with non-zero entries according to edges. While the mapping of bit nodes must follow the prescription of the labelling of $\mathcal{G}$ (i.e., the code), the ordering of rows is arbitrary.

The local application of ELC has the global effect of row additions on the associated H, thus preserving the bipartiteness and vector space (i.e., $\mathcal{C}$) [12]. Consider again Fig. 1, where we choose u to be a constraint node, and v a bit node. With this setup, ELC on edge (u, v) is equivalent to adding 'row u' to rows $u' \in \mathcal{N}_v^u$ (as dictated by the non-zero entries of 'column v'). Since H is in standard form, an immediate effect of ELC on some edge (c, p) is that the edges adjacent to information node, p, are swapped with that of the degree-1 parity node adjacent to the constraint node, c, as seen in Fig. 2 (b,e). As opposed to [5], we are permuting H, whereas the soft input vector remains invariantly connected to (the bit nodes of) $\mathbf{TG}(H)$. The indices of Fig. 2 show how the order of the soft input vector is preserved. Extrinsic information is lost on edges deleted in the local complementation. However, sum-product algorithm (SPA) update rules are such that these messages remain stored in adjacent bit nodes as *a posteriori probabilities* (APPs) [13].

As can be readily verified, although ELC preserves the code, it can have a negative impact on parameters of its implementation, H, as a decoder. Edges complemented are at distance 2 from (u, v), so for a typical sparse, girth-6 graph, many 4-cycles result, and density increases [14]. ELC does not generally preserve graph isomorphism (structure), so the operation will often given us a different structure in the (ELC) *orbit* of $\mathcal{G}$ [15]. The matrices H^i in this orbit are the set of structurally different parity-check matrices for the same code, as discussed in the



(a) $\mathcal{G}$ (b) $\mathbf{TG}(H)$ (c) Structure is a cube

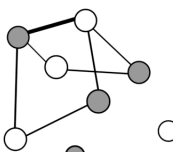
(d) ELC $(0,4)$ (e) ELC $(0,4)$ (f) ...relabelled cube

Fig. 2: *(a) through (c) are three equivalent representations of the* $(8,4)$ *extended Hamming code. (d) through (e) show the corresponding representation after ELC is applied to edge* $(0,4)$*. Fig. 5 shows the parity-check matrices of (b) and (e), respectively.*

Introduction. We briefly mention that all information sets of $\mathcal{C}$ may be enumerated by traversing this orbit of $\mathcal{G}$ [11].

2.1 Iso-ELC

In this section we describe an application of ELC to preserve key features of the graph, to remedy the drawbacks enumerated in the previous section. We define iso-ELC as a sequence of ELC operations over which (global) graph isomorphism is preserved. Such an operation will be in $\mathrm{Aut}(\mathcal{C})$, in that its action has the appearance of a relabelling on the nodes of a graph, or–equivalently–a permutation on the columns of a matrix (H). If there exist sequences of ELC operations which preserve the structure of $\mathcal{G}$, then $\mathrm{Aut}(\mathcal{C})$ must be non-trivial. Isomorphism is a certificate on the properties of the resulting graphs (matrices) used during decoding; that these remain the same as for the initial $\mathcal{G}$ (H), which can be assumed to have been carefully selected. The relabelling,



however, alters the flow of messages in $\mathbf{TG}(H)$, i.e., which nodes are exchanging information. Note in particular how, after the iso-ELC in Fig. 2(f), node 4 is no longer part of a 4-cycle (whereas node 0 now is).

In the following, we derive three requirements for ELC being an isomorphism.

A. Most generally, to have an isomorphism, the number of edges in $\mathcal{G}$ must remain invariant under ELC. ELC is a local operation, so we only have to consider the subgraph $\mathcal{G}_{\mathcal{N}_u \cup \mathcal{N}_v}$. Edge complementation on edge (u, v) can then be achieved by complementing the corresponding $\deg(u) \times \deg(v)$ submatrix, $H_{(u,v)}$. The resulting matrix after complementation is denoted by $H^{\mathsf{C}}_{(u,v)}$. Define $|H|$ as the weight (number of non-zero entries) of H. In order for $|H_{(u,v)}| = |H^{\mathsf{C}}_{(u,v)}|$, at least one of the dimensions must be an even number, and $|H_{(u,v)}|$ must equal $(\deg(u) \cdot \deg(v))/2$. If these conditions are met, we define the ELC operation as *edge-count preserving*.

B. More specifically, we define a *local isomorphism* as an operation which preserves the structure of subgraph $\mathcal{G}_{\mathcal{N}_u \cup \mathcal{N}_v}$, without making any assumptions on the overall (global) structure of $\mathcal{G}$. We then define the ELC operation to be *local iso-ELC iff* $H_{(u,v)}$ can be recovered from $H^{\mathsf{C}}_{(u,v)}$ by row/column permutations only. Fig. 3 shows a small example.

C. Finally, most specifically, we say that ELC is a (global) *iso-ELC iff* H can be restored from H^1, using only row/column permutations, considering the entire matrix.

These requirements lead to the following observation,

$$C \Rightarrow B \Rightarrow A.$$

In the following, we will consider global isomorphisms only, and we will refer to such sequences as as simply being iso-ELC operations, or sequences.

2.2 Iso-Orbit

The definitions of iso-ELC are naturally extended to the case where a single ELC can not by itself be an isomorphism. Consider, for instance,

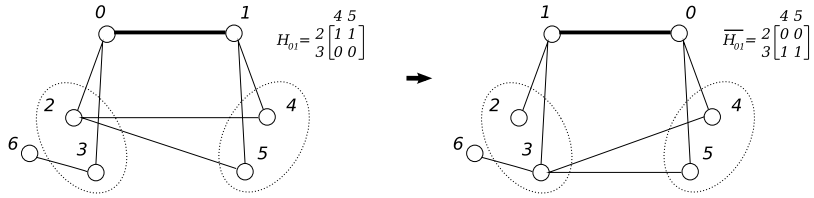


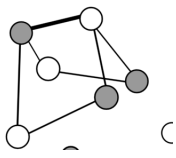
Fig. 3: *ELC on* $(0,1)$ *is (A) edge-count preserving and (B) a local isomorphism, but not (C) a global isomorphism due to node 6. This node is not local to the ELC edge.*

a girth-6 graph. Here, the local neighborhood (of any edge) must be empty, and, after a single ELC, this neighborhood becomes a complete (bipartite) (sub)graph at distance 2 from the ELC edge (all 4-cycles). This violates requirement A, and the resultant graph can not be isomorphic to the initial one–neither locally, nor globally.[2]

In the general case, iso-ELC is described as an ordered set of l edges on which ELC must be applied to achieve an isomorphism. This is referred to as a l-iso sequence (or, a length-l iso sequence). The set of all isomorphisms of $\mathcal{G}$ (reachable via iso-ELC, for $d \geq 1$) is called the *iso-orbit* of $\mathcal{G}$, which corresponds to a subset of $\mathrm{Aut}(\mathcal{C})$. ELC can be used, in a preprocessing stage, to recursively search for iso-ELC sequences. For each such relabelling of $\mathcal{G}$, we keep the corresponding iso sequence leading to it. Since ELC is reversible, identical isomorphisms may be found via sequences of different length, and involving different edges, where certain operations cancel each other out. As such, for each unique labelling, we keep only the minimum length sequence in the iso-orbit.

From a decoding perspective, row permutations of H give the same $\mathbf{TG}(H)$. By canonising the rows of H (in our case, sorting according to decimal value of the binary rows), we ensure that the iso-orbit contains only non-trivial isomorphisms of $\mathcal{G}$. The complexity of such a brute force search may be high, so for strongly structured (and large) graphs it may be necessary to bound the recursion with a maximum depth, $l_{\max}$. This possibly partial iso-orbit is then simply referred to as the l-iso-orbit of $\mathcal{G}$.

[2]This is also evident simply from the change in girth.



2.3 Local Iso Criterions

Although the message-passing decoder can be provided with $\mathbf{TG}(H)$ and a list of iso-sequences, to facilitate adaptive decoding, our stated graph local approach lends itself to *ad hoc* determination of iso-sequences during decoding. In this subsection, we describe some 1-iso conditions which ensure that ELC on the single edge (u, v) of $\mathcal{G}$ gives an isomorphism of $\mathcal{G}$.

From a local perspective, an edge (u, v) can sometimes determine whether or not (global) structure will be preserved if it applies an ELC. This edge may only examine its local subgraph, $\mathcal{G}_{\mathcal{N}_u \cup \mathcal{N}_v}$. In this manner, we alleviate both the potentially expensive preprocessing stage, as well as the overhead of storing and permuting a list of sequences. Where a local criterion is satisfied, (u, v) may remain unaware of the implicit (iso) permutations that occur, except from the fact that (u, v) remains invariant (hence the alternative term, *pivot*).

We define $\ominus$ as the symmetric difference, i.e., for sets A and B, $A \ominus B \triangleq (A \setminus B) \cup (B \setminus A)$.

Lemma 1. *ELC on the edge (u, v) of a simple bipartite graph, $\mathcal{G}$, preserves $\mathcal{G}$ up to local graph isomorphism if at least one of the sets $\mathcal{N}_u^v$ and $\mathcal{N}_v^u$ satisfy one of the following conditions, with $\{\alpha, \alpha'\} = \{u, v\}$,*

- *$\exists\ \alpha,\ \alpha'$ such that $\mathcal{N}_\alpha^{\alpha'} = \emptyset$, or*
- *$\exists\ \alpha, \alpha'$ such that $\mathcal{N}_\alpha^{\alpha'}$ can be partitioned in pairs $\{w_i, w_i'\}$, where $\mathcal{N}_{w_i} \ominus \mathcal{N}_{w_i'} = \mathcal{N}_{\alpha'}^{\alpha}\ \forall i, \{w_i, w_i'\} \cap \{w_j, w_j'\} = \emptyset, i \neq j$.*

Global isomorphism can be ensured by the condition that the subgraphs induced by $\mathcal{N}_\alpha^{\alpha'}$ and their neighbors, and $\mathcal{N}_{\alpha'}^{\alpha}$ and their neighbors, are both bipartite complete graphs. Less restrictive conditions will also ensure global isomorphism, depending on the permutation of the vertices of the graph.

Proof. We will prove the two parts separately.

- Either $\mathcal{N}_u^v = \emptyset$, or $\mathcal{N}_v^u = \emptyset$. Let $\mathcal{N}_u^v = \emptyset$. Then ELC on (u, v) has the effect of disconnecting v from $\mathcal{N}_v^u$, while connecting u to $\mathcal{N}_v^u$. The permutation that gives us the isomorphism is $\sigma = (u\ v)$. The same permutation applies when $\mathcal{N}_v^u = \emptyset$.
- For every $w_i \in \mathcal{N}_\alpha^{\alpha'}, \exists\ w_i' \in \mathcal{N}_\alpha^{\alpha'}$ such that $\mathcal{N}_{w_i} \ominus \mathcal{N}_{w_i'} = \mathcal{N}_{\alpha'}^{\alpha}$: The permutation that gives us the isomorphism is $\sigma = (u\ v) \prod (w_i\ w_i')$.

□

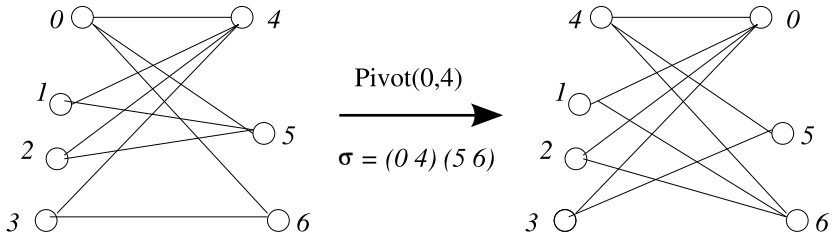


Fig. 4: *Example of Lemma 1, where* $\alpha = 0$, $\alpha' = 4$, $w_0 = 5$, *and* $w'_0 = 6$. *These graphs are isomorphic.*

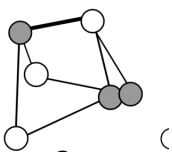
An example of Lemma 1 is found in Fig. 4.

As any individual ELC operation complements edges local to u and v (i.e., 4-cycles), we say that 1-iso 'sequences' can only exist for graphs of girth 4, or locally acyclic (tree) graphs for which the first part of Lemma 1 applies. Similar criteria have been identified for $l = 2$, but these were not applied in this initial work.

3 Structure of the $[8, 4]$ Extended Hamming Code

The $(8, 4)$ extended Hamming code is a well-suited test case for adaptive decoding; it has strong classical properties (large automorphism group and minimum distance), yet for any implementation H it is ill-suited for message-passing (dense, and many 4-cycles). We acknowledge that this is a toy code, which presents obvious difficulties in arguing any sense of 'locality' of such a small graph. However, the positive nature of our results show that this code does suffice as motivation for the proposed class of adaptive decoders, and we direct the reader to the Future Work section of this article.

The associated graph has one structure in its orbit; the cube of Fig. 2(c), meaning that its structure is so strong that any edge of any $\mathcal{G}^i$ satisfies Lemma 1 (is an iso-ELC). Starting from $\mathcal{G}$ as in Fig. 2(b), with parity-check matrix in Fig. 5(a), we find the iso-orbit of the graph. Grouped by length, $l = 1$ to 4, this orbit consists of 12, 30, 12, and 1 isomorphisms, respectively, all resembling a cube. Including the initial labelling, this sums up to 56 structurally distinct, non-trivial parity-check



$$H = \begin{array}{c} \overset{I}{\overbrace{}}\ \overset{P}{\overbrace{}} \\ \left[\begin{array}{cccccccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}\right] \end{array} \qquad H^{l} = \left[\begin{array}{cccccccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}\right]$$

(a) Standard form (b) After ELC on $(0, 4)$

Fig. 5: *The* $[8, 4]$ *extended Hamming code, implemented by its standard form parity-check matrix (a), and an isomorphism (b).*

matrices for the code.[3] Necessarily, the 12 1-iso sequences correspond to the 12 edges of $\mathcal{G}$ (P-part of H).

4 Simulation Results

The adaptive decoder has been tested in two instances, and compared against a SPA decoder using standard flooding scheduling on output $\mathbf{y}$ from the additive white Gaussian noise (AWGN) channel.[4] During implementation we made sure that all decoders were allocated an equal maximum number of iterations ($\tau = 100$). In the following description, we assume an initial syndrome check has failed, so we have a vector to input to the decoder.[5]

Due to the symmetry of the $[8, 4]$ code in standard form, we know any ELC will preserve isomorphism. Thus, when considering the adaptive decoders presented and analyzed in the following, the reader is encouraged to think of these as truly localized (i.e., independent of preprocessing and input lists), as if these were determined *ad hoc*. In comparison, Halford and Chugg [5] are applying (non-local) permutations drawn at random from the full automorphism group of the code. They also restrict to a cyclic subgroup of $\mathrm{Aut}(\mathcal{C})$–we do not do this. As discussed, our use of iso-ELC naturally gives a subset of $\mathrm{Aut}(\mathcal{C})$.

[3] Note that the full automorphism group of this code may be found by row permutations on these generators; $56 \cdot 4! = 1344 = |\mathrm{Aut}(\mathcal{C})|$.

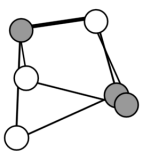
[4] One flooding iteration consists of the SPA update of all bit (information and parity) nodes, followed by the update of all constraint nodes.

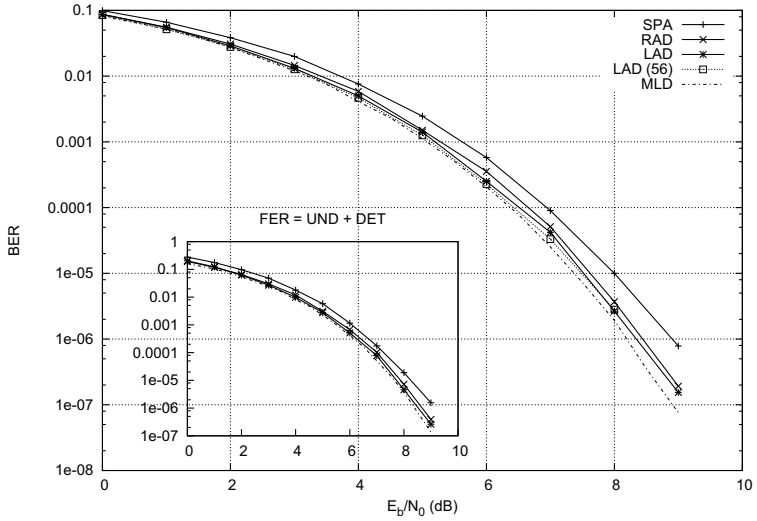
[5] For locality, we emphasize that constraint nodes of $\mathbf{TG}(H)$ can be viewed as $[n, n-1, 2]$ component parity-check codes, which can be computed (checked) concurrently and distributively. However, a stopping criterion for the whole code is inherently a global decision.

The proposed *random adaptive decoder* (RAD) is a (flooding) SPA decoder, but which is designed to adapt (via random iso-ELC) to another $\mathcal{G}^i$, with regular iteration interval, t. The decoder stops as soon as the syndrome check is satisfied (valid codeword, though not necessarily the one sent), or when T iterations are exhausted (detected frame error). In a localized manner, this decoder performs a random walk (with repetitions) in the iso-orbit of $\mathcal{G}$, taking advantage of the discussed symmetry. As such, the *range*–i.e., the number of matrices available to this decoder–includes all 56 non-trivial isomorphisms.

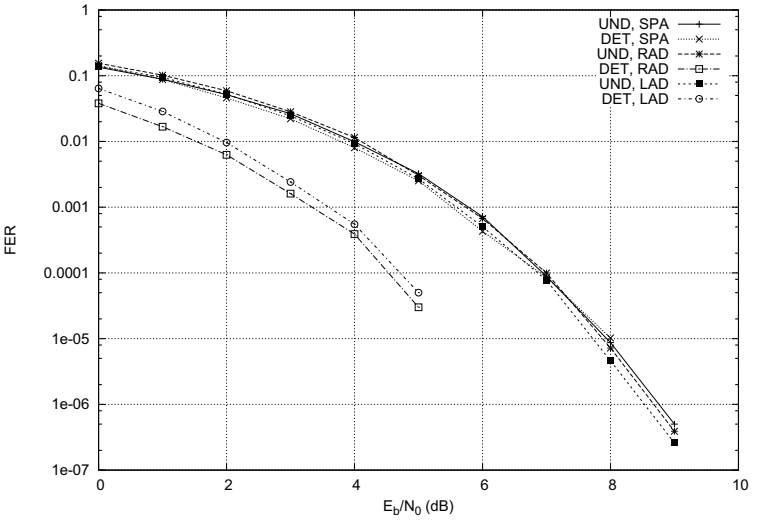
The *list adaptive decoder* (LAD) is an extension of this idea, but where we apply iso-ELC operations from a precomputed list, $L \subseteq \text{iso-orbit}(\mathcal{G})$. In addition to the initial labelling, the range of this decoder is $D = |L| + 1$. A pool of τ flooding iterations is allocated. Graph $\mathcal{G}^i$, $0 \leq i < D$, is allocated $h_i = \lceil (\tau - I)/(D - i) \rceil$ iterations to come to a decoder decision, where I is the total number of iterations used by previous decoders $\mathcal{G}^j$, $j < i$. Depending on τ and L, h_i may go to 0, so an overall minimum, $h_{\min}$, should be set. This means that, although the list L may not be employed in its entirety, we ensure that the graphs used are doing useful work (more than 1 iteration). This minium should reflect parameters of the graph and code. Before applying the next iso-ELC from L, $\mathcal{G}^i$ compares its local decision to a running optimum kept in the decoder, and overwrites if a better decoder output is found (in squared Euclidean distance from $\mathbf{y}$). This comparison is devised to favor valid decoder states, in that distance measures of detected failures are only considered as long as no valid state has been found. The LAD does not stop on reaching a valid decoder state, but continues until *timeout* (τ iterations). The final graph, $\mathcal{G}^\delta$, where $\lfloor \tau/h_{\min} \rfloor \leq \delta \leq D - 1$, outputs the optimum decision as the decoder result. In case no graph reached a valid syndrome, the error state nearest to $\mathbf{y}$ (of the δ timeout states) is output. This is in an effort to reduce the bit-error contribution.

Fig. 6 benchmarks the performance of RAD and LAD against standard SPA, and the optimal maximum likelihood decoder (MLD), in terms of bit-error rate (BER) and frame-error rate (FER), where an improvement is seen. The LAD plot is slightly nearer to the optimal MLD plot than the RAD, but the gain is not significant compared to the complexity tradeoff (Fig. 7). A detailed look at the (detected and undetected) frame errors, Fig. 6(b), reveals that adaptive decoders outperform SPA in terms of detected errors (timeout), where RAD shows the best gain. The RAD performs a random walk around the 56 sequences in the iso-orbit of $\mathcal{G}$, while, for the LAD, we chose the subset





(a) Our class of decoders (both RAD and LAD) outperform SPA, both in BER and FER. Only a small improvement was seen when using the entire iso-orbit, LAD(56).



(b) DETected (timeouts) and UNDetected frame errors compared separately. A significant gain is found in the class of detected word errors.

Fig. 6: *Simulations results on an AWGN channel. Maximum* $\tau = 100$ *iterations used.* $t = 10$ *for RAD, and* $|L| = 12$ *for LAD. At least* 100 *detected and* 100 *undetected frame errors were sampled for each* E_b/N_0 *point.*

of 12 1-iso sequences (defined by the 12 edges of the initial $\mathcal{G}$) such that $h_{\min} = \lceil 100/13 \rceil = 7$.

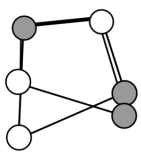
Only a small additional gain was achieved by using the full iso-orbit. In this case, we used the same minimum as for the LAD; $h_{\min} = 7$ iterations. This means that not all 56 sequences were guaranteed to be used, so we permuted the order of sequences in L before every decoding instance. As such, in the cases where the graphs did non-negligible work (i.e., there were errors), on average each graph ran all its $h_{\min}$ iterations. Hence, we may say that 13 random iso-ELC operations (sequences) were applied at random from the iso-orbit of $\mathcal{G}$. The simulation 'LAD(56)' in Fig. 6(a) demonstrates the benefit of using the entire iso-orbit, albeit slim, for this small code.

Fig. 7 shows the complexity (average number of flooding iterations used) of the decoders, where we observe another improvement of RAD over SPA and LAD decoding. At high E_b/N_0, complexity averages go to 0, which is due to the majority of received frames satisfying the initial syndrome check (which we do not count as an iteration). While LAD expectedly uses a higher average number of iterations, since it does not stop at the first valid syndrome, an interesting observation is the complexity gain of RAD, which is linked to the reduction in number of timeouts (detected frame errors–see Fig. 6(b)).

5 Conclusion and Future Work

We have described and tested a class of adaptive iterative decoders, which dynamically update the edge-space of the code implementation, $\mathbf{TG}(H)$, using local decisions and operations. Concrete 'iso-criterions' are described and mathematically proven, and simulations on the AWGN channel show a gain when using our ideas. Two related instances of our class of adaptive decoders are described, where we conclude that, although LAD is slightly better than RAD in terms of BER, that gain comes at a cost of increased complexity (average number of iterations used) and loss of locality. Furthermore, RAD outperforms LAD in terms of FER, which gives an interesting latency reduction.

As iso-ELC rotates sensitive substructures in $\mathbf{TG}(H)$, we expect a gain in selectively applying iso-ELC based on local convergence assessments (e.g., using entropy or reliability measures). We mention shifting short cycles away from unreliable bit nodes–as seen in the cube of Fig. 2. An ELC adjacent to unreliable positions also causes these to become temporarily 'isolated' in terms of message-passing (weight-1



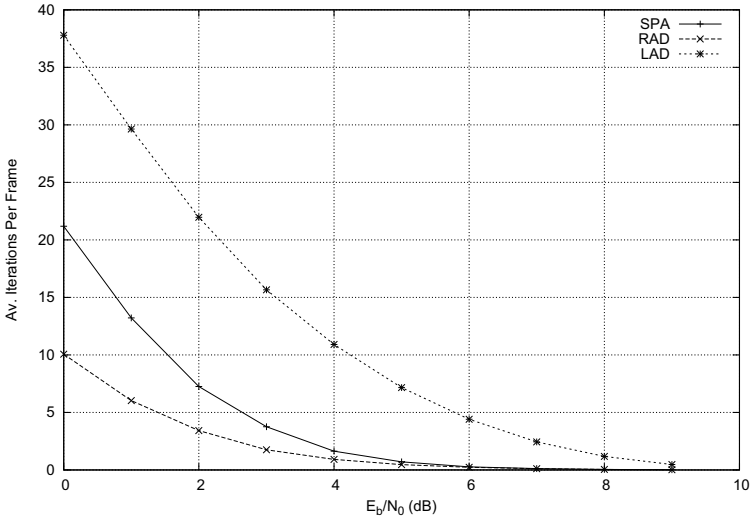


Fig. 7: *The total number of iterations used (where timeout states contribute τ iterations, and error-free frames contribute 0) averaged over total number of simulated frames.*

node), such that these are set in a 'listening state,' rather than confusing the adjacent nodes with its (presumed) unreliable APP [3, 16, 17]. In our scheme, we achieve this effect without the overhead of Gaussian elimination.

Local iso-criterions for $l = 2$ have been identified, and we are also working on further generalizations. This is interesting, as, due to the link between ELC and 4-cycles, girth-6 graph isomorphisms can not be preserved with less than 2 ELC operations. Our results on global isomorphisms indicate that it is not trivial to find graphs which exhibit a non-empty iso-orbit, which simultaneously are good codes (i.e., sparse and girth greater than 4). A reasonable next step is a more methodical search through all codes up to some length, yet we are also looking towards the use of local isomorphisms during decoding. For instance, when girth is not preserved (see Section 2.1), cycle-splitting or cycle-reduction–as in the way two 4-cycles can sometimes be split into one 6-cycle–may improve decoding.

We are working on a generalized ELC operation, which does not depend on the matrix (graph) being in systematic form. With this

tool, we expect to be able to compare our results with the realistically sized BCH code of [5]. We anticipate more significant results where larger Tanner graphs allow more true localization. Euclidean geometry LDPC codes [18] are also potential, sufficiently structured candidates for adaptive decoding.

Enforcing a strictly local perspective does present some practical difficulties, most notably, decoder stopping criterion and optimum decoder state comparison used in LAD. However, in the context of decoding–as in this work–this does not present a problem, yet rather suggests potential implementations of the iterated decoder where the graph nodes are distributed in space and/or time.

ACKNOWLEDGMENTS

Thanks to Øyvind Ytrehus and Lars Eirik Danielsen, at the University of Bergen, for helpful discussions.

REFERENCES

[1] GALLAGER, R. G.: Low-density parity-check codes. *IRE Trans. Inform. Theory* 8(1), 21–28, Jan. 1962.

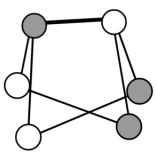
[2] MACKAY, D. J. C., NEAL, R. M.: Good codes based on very sparse matrices. In *Proc. 5th IMA Conf. on Cryptography and Coding LNCS 1025*, pp. 100–111. Royal Agricultural College, Cirencester, UK, Dec. 1995.

[3] JIANG, J., NARAYANAN, K. R.: Iterative soft-input soft-output decoding of Reed-Solomon codes by adapting the parity-check matrix. *IEEE Trans. Inform. Theory* 52(8), 3746–3756, Aug. 2006.

[4] JIANG, J., NARAYANAN, K. R.: Iterative soft decoding of Reed-Solomon codes. *IEEE Commun. Lett.* 8(4), 244–246, Apr. 2004.

[5] HALFORD, T. R., CHUGG, K. M.: Random redundant iterative soft-in soft-out decoding. *IEEE Trans. Commun.* 56(4), 513–517, Apr. 2008.

[6] VONTOBEL, P. O., KOETTER, R.: Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes, 2005. arXiv:cs/0512078.



[7] Di, C., Proietti, D., Telatar, I. E., Richardson, T. J., Urbanke, R. L.: Finite-length analysis of low-density parity-check codes on the binary erasure channel. *IEEE Trans. Inform. Theory* 48(6), 1570–1579, Jun. 2002.

[8] Richardson, T.: Error floors of LDPC codes. In *Proc. 41st Annual Allerton Conf. on Commun., Control, and Computing*, pp. 1426–1435. Monticello, IL, Oct. 2003.

[9] Richardson, T. J., Urbanke, R. L.: The capacity of low-density parity-check codes under message-passing decoding. *IEEE Trans. Inform. Theory* 47(2), 599–618, Feb. 2001.

[10] Bouchet, A.: Isotropic systems. *European J. Comb.* 8, 231–244, Jul. 1987.

[11] Danielsen, L. E., Parker, M. G.: Edge local complementation and equivalence of binary linear codes. *Des. Codes Cryptogr.* 49(1-3), 161–170, Dec. 2008.

[12] Riera, C., Parker, M. G.: On Pivot orbits of Boolean functions, optimal codes and related topics. In *Fourth Int. Workshop on Optimal Codes and Related Topics*, pp. 248–253. Sofia, Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Jun. 2005.

[13] Kschischang, F. R., Frey, B. J., Loeliger, H. A.: Factor graphs and the sum-product algorithm. *IEEE Trans. Inform. Theory* 47(2), 498–519, Feb. 2001.

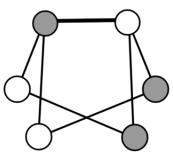
[14] Knudsen, J. G.: *Randomised Construction and Dynamic Decoding of LDPC Codes*. Master's thesis, University of Bergen, Bergen, Norway, 2006.

[15] Danielsen, L. E., Parker, M. G.: On the classification of all self-dual additive codes over GF(4) of length up to 12. *J. Comb. Theory, Series A* 113(7), 1351–1367, Oct. 2006.

[16] Catherine, C.: *Enhancing the error-correction performance of low-density parity-check codes*. Ph.D. thesis, Univ. of Mauritius, 2008.

[17] Kothiyal, A., Takeshita, O. Y.: A comparison of adaptive belief propagation and the best graph algorithm for the decoding of linear block codes. In *Proc. IEEE Int. Symp. Inform. Theory*, pp. 724–728. Adelaide, Australia, Sep. 2005.

[18] Kou, Y., Lin, S., Fossorier, M. P. C.: Low-density parity-check codes based on finite geometries: A rediscovery and new results. *IEEE Trans. Inform. Theory* 47(7), 2711–2736, Nov. 2001.



PAPER II

ITERATIVE DECODING ON MULTIPLE TANNER GRAPHS USING RANDOM EDGE LOCAL COMPLEMENTATION *

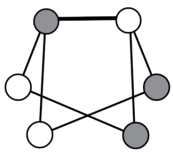
Joakim Grahl Knudsen Constanza Riera

Lars Eirik Danielsen Matthew G. Parker

Eirik Rosnes

*Knudsen, J. G., Riera, C., Danielsen, L. E., Parker, M. G., Rosnes, E.: Iterative decoding on multiple Tanner graphs using random edge local complementation. In *Proc. IEEE Int. Symp. Inform. Theory*, pp. 899–903. Seoul, Korea, Jun./Jul. 2009.

The notation in this chapter has been edited to make it consistent with the thesis.



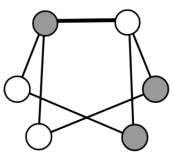
ITERATIVE DECODING ON MULTIPLE TANNER GRAPHS USING RANDOM EDGE LOCAL COMPLEMENTATION

Joakim Grahl Knudsen Constanza Riera

Lars Eirik Danielsen Matthew G. Parker

Eirik Rosnes

In this paper, we propose to enhance the performance of the sum-product algorithm (SPA) by interleaving SPA iterations with a random local graph update rule. This rule is known as edge local complementation (ELC), and has the effect of modifying the Tanner graph while preserving the code. We have previously shown how the ELC operation can be used to implement an iterative permutation group decoder (SPA-PD)–one of the most successful iterative soft-decision decoding strategies at small blocklengths. In this work, we exploit the fact that ELC can also give structurally distinct parity-check matrices for the same code. Our aim is to describe a simple iterative decoder, running SPA-PD on distinct structures, based entirely on random usage of the ELC operation. This is called SPA-ELC, and we focus on small blocklength codes with strong algebraic structure. In particular, we look at the extended Golay code and two extended quadratic residue codes. Both error rate performance and average decoding complexity, measured by the



average total number of messages required in the decoding, significantly outperform those of the standard SPA, and compares well with SPA-PD. However, in contrast to SPA-PD, which requires a global action on the Tanner graph, we obtain a performance improvement via local action alone. Such localized algorithms are of mathematical interest in their own right, but are also suited to parallel/distributed realizations.

1 Introduction

Inspired by the success of iterative decoding of low-density parity-check (LDPC) codes, originally introduced by Gallager [1] and later rediscovered in the mid 1990's by MacKay and Neal [2], on a wide variety of communication channels, the idea of iterative, soft-decision decoding has recently been applied to classical algebraically constructed codes in order to achieve low-complexity belief propagation decoding [3–8]. Also, the classical idea of using the automorphism group of the code, $\mathrm{Aut}(\mathcal{C})$, to permute the code, $\mathcal{C}$, during decoding (known as *permutation decoding* (PD) [9]) has been successfully modified to enhance the sum-product algorithm (SPA) in [5]. We will denote this algorithm by SPA-PD. Furthermore, good results have been achieved by running such algorithms on several structurally distinct representations of $\mathcal{C}$ [3, 6]. Both Reed-Solomon and Bose-Chaudhuri-Hocquenghem (BCH) codes have been considered in this context. Certain algebraically constructed codes are known to exhibit large minimum distance and a non-trivial $\mathrm{Aut}(\mathcal{C})$. However, additional properties come into play in modern, graph-based coding theory, for instance, sparsity, girth, and trapping sets [10, 11]. Structural weaknesses of graphical codes are inherent to the particular parity-check matrix, H, used to implement $\mathcal{C}$ in the decoder. This matrix is a non-unique $(n-k)$-dimensional basis for the null space of $\mathcal{C}$, which, in turn, is a k-dimensional subspace of $\{0,1\}^n$. Although any basis (for the dual code, $\mathcal{C}^\perp$) is a parity-check matrix for $\mathcal{C}$, their performance in decoders is not uniform. H is said to be in standard form if the matrix has $n-k$ weight-1 columns. The weight of H is the number of non-zero entries, and the minimum weight is lower-bounded by $(n-k)d_{\mathbf{min}}(\mathcal{C}^\perp)$, where $d_{\mathbf{min}}(\mathcal{C}^\perp)$ denotes the minimum distance of $\mathcal{C}^\perp$. It is well-known that H can be mapped into a bipartite (Tanner) graph, $\mathbf{TG}(H)$, which has an edge connecting nodes v_i and u_j *iff* $H_{ji} \neq 0$. Here, $v_i, 0 \leq i < n$, refers to the bit nodes

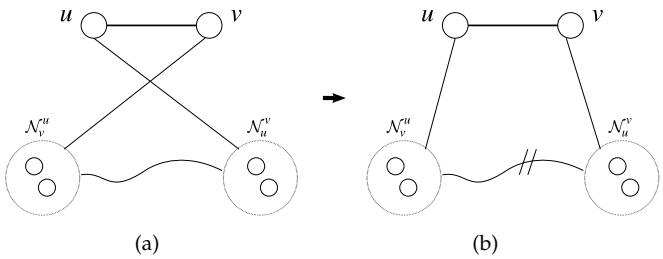


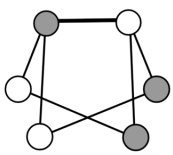
Fig. 1: *ELC on edge (u, v) of a bipartite graph. Doubly slashed links mean that the edges connecting the two sets have been complemented.*

(columns of H), and $u_j, 0 \leq j < n - k$, refers to the check nodes (rows of H). The local neighborhood of a node, v, is the set of nodes adjacent to v, and is denoted by $\mathcal{N}_v$. The terms standard form and weight extend trivially to $\mathbf{TG}(H)$. In the following, we use bold face notation for vectors, and the transpose of H is written H^T.

This paper is a continuation of our previous work on edge local complementation (ELC) and iterative decoding, in which selective use of ELC (with preprocessing and memory overhead) equals SPA-PD [8]. In this work, we use ELC in a truly random, online fashion, thus simplifying both the description and application of the proposed decoder. The key difference from our previous work is that we do not take measures to preserve graph isomorphism, and explore the benefits of going outside the automorphism group of the code. This means that we alleviate the preprocessing of suitable ELC locations (edges), as well as the memory overhead of storing and sampling from such a set during decoding. Our proposed decoding algorithm can be thought of as a combination of SPA-PD [5] and multiple bases belief propagation [6]. We also discuss the modification of the powerful technique of damping to a graph-local perspective.

2 The ELC Operation

The operation of ELC [12–14], also known as *pivot*, is a local operation on a simple graph (undirected with no loops), $\mathcal{G}$, which has been shown to be useful both for code equivalence and classification [13], and for decoding purposes [8]. It has recently been identified as a useful local unitary primitive to be applied to *graph states* [14]–a proposed paradigm



for quantum computation [15]. Fig. 1(a) shows $\mathcal{G}_{\mathcal{N}_u \cup \mathcal{N}_v}$, the local subgraph of a bipartite graph induced by nodes u, v, and their disjoint neighborhoods which we denote by $\mathcal{N}_u^v \triangleq \mathcal{N}_u \setminus \{v\}$ and $\mathcal{N}_v^u \triangleq \mathcal{N}_v \setminus \{u\}$, respectively. ELC on a bipartite graph is described as the complementation of edges between these two sets; $\forall\, v' \in \mathcal{N}_u^v$ and $\forall\, u' \in \mathcal{N}_v^u$, check whether edge $(u', v') \in \mathcal{G}$, in which case it is deleted, otherwise it is created. Finally, the edges adjacent to u and v are swapped – see Fig. 1(b). ELC on $\mathcal{G}$ extends easily to ELC on $\mathbf{TG}(H)$ when H is in standard form [8]. Given a bipartite graph with bipartition $(\mathcal{V}, \mathcal{U})$, we then have a one-to-one mapping to a Tanner graph, with check nodes from the set $\mathcal{U}$ and bit nodes from $\mathcal{V} \cup \mathcal{U}$. Fig. 2 shows an example, where the bipartition is fixed according to the sets $\mathcal{V}$ and $\mathcal{U}$. In Fig. 2(a), the left and right nodes correspond to $\mathcal{V}$ and $\mathcal{U}$, respectively, for the simple graph $\mathcal{G}$. $\mathbf{TG}(H)$ may be obtained by replacing grey nodes by a check node singly connected to a bit node, as illustrated in Fig. 2(b). Figs. 2(a) and 2(c) show an example of ELC on the edge $(0, 5)$. Although the bipartition changes (edges adjacent to 0 and 5 are swapped), Figs. 2(b) and 2(d) show how the map to Tanner graphs, in fact, preserves the code.

By complementing the edges of a local neighborhood of $\mathbf{TG}(H)$, ELC has the effect of row additions on H. The complexity of ELC on (u, v) is $\mathcal{O}(|\mathcal{N}_u||\mathcal{N}_v|)$. The set of vertex-labeled graphs generated by ELC on $\mathbf{TG}(H)$ (or, equivalently, $\mathcal{G}$) is here called the *ELC-orbit* of $\mathcal{C}$. Each information set for $\mathcal{C}$ corresponds to a unique graph in the ELC-orbit [13]. Note that this is a code property, which, as such, is independent of the initial parity-check matrix, H. The set of structurally distinct (unlabeled) graphs generated by ELC is here called the *orbit* of $\mathcal{C}$, and is a subset of the ELC-orbit. Graphs are structurally distinct (i.e., non-isomorphic) if the corresponding parity-check matrices are not row or column permutations of each other. Each structure in the orbit has a set of $|\mathrm{Aut}(\mathcal{C})|$ isomorphic graphs, comprising an *iso-orbit* [8]. In the following, we will refer to ELC directly on $\mathbf{TG}(H)$, keeping Fig. 2 in mind.

3 Decoding Algorithms

3.1 SPA

The SPA is an inherently local algorithm on $\mathbf{TG}(H)$, where the global problem of decoding is partitioned into a system of simpler subprob-

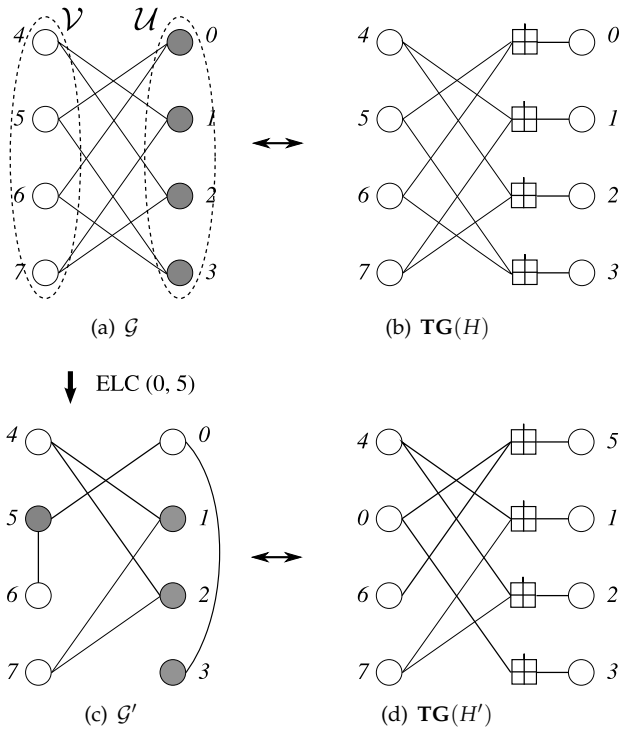
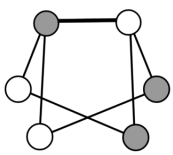


Fig. 2: *(a) and (c) show ELC on the edge* $(0,5)$ *of a small simple graph* $\mathcal{G}$*. The corresponding Tanner graphs, in (b) and (d), are distinct structures (e.g., the weight of* $\mathcal{G}$ *and* $\mathcal{G}'$ *is not the same) for the same toy* $[8, 4, 2]$ *code. This code has a total of three structures in its orbit.*

lems [16]. Each node and its adjacent edges can be considered as a small constituent code, and essentially performs maximum-likelihood decoding (MLD) based on local information. The key to a successful decoder lies in this partitioning–how these constituent codes are interconnected. The summed information contained in a bit node, v_i, is the *a posteriori* probability (APP), $\hat{x}_i$, at codeword position i. The vector $\hat{\mathbf{x}}$ constitutes a tentative decoding of the received channel vector, $\mathbf{y}$. The decoder input is the log-likelihood ratio (LLR) vector $\mathbf{L} = (2/\sigma^2)\mathbf{y}$, where σ is the channel noise standard deviation on an additive white Gaussian noise (AWGN) channel. Subtracting the input from the APP leaves the extrinsic information, $\hat{x}_i - L_i$, which is produced by the decoder. The message on the edge from node v to u, in the direction of u, $\mu_{v\to u}$, is



computed according to the SPA rule on node v. The SPA computation of all check nodes, followed by all bit nodes, is referred to as one *flooding* iteration.

Classical codes, for which strong code properties are known, are typically not very suitable for iterative decoding mainly due to the high weight of their parity-check matrices, which gives many short cycles in the corresponding Tanner graphs.

3.2 Diversity Decoding

A few recent proposals in the literature have attempted to enhance iterative decoding by dynamically modifying $\mathbf{TG}(H)$ during decoding, so as to achieve diversity and avoid fixed points (local optima) in the SPA convergence process. Efforts to improve decoding may, roughly, be divided into two categories. The first approach is to employ several structurally distinct matrices, and use these in a parallel, or sequential, fashion [3, 6]. These matrices may be either preprocessed, or found dynamically by changing the graph during decoding. However, this incurs an overhead either in terms of memory (keeping a list of matrices, as well as state data), or complexity (adapting the matrix, e.g., by Gaussian elimination [3]). The other approach is to choose a code with a non-trivial $\mathrm{Aut}(\mathcal{C})$, such that diversity may be achieved by permuting the code coordinates [4, 5, 7, 8]. An example is SPA-PD, listed in Alg. 1, where $\mathrm{Aut}(\mathcal{C})$ is represented by a small set of generators, and uniformly sampled using an algorithm due to Celler *et al.* [17]. These permutations tend to involve all, or most, of the code coordinates, making it a global operation. Note that line 7 in Alg. 1 is to compensate for the fact that permutations are applied to $\mathbf{L}$ in line 12, rather than to the columns of H, after which the messages on the edges no longer 'point to' their intended recipients. This is yet another global stage. The extrinsic information is damped by a coefficient α, $0 < \alpha < 1$, in line 10 before being used to re-initialize the decoder. Each time α is incremented, the decoder re-starts from the channel vector, $\mathbf{y}$.

3.3 SPA-ELC

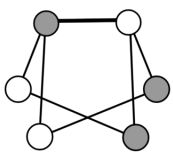
Our proposed local algorithm is a two-stage iterative decoder, interleaving the SPA with random ELC operations. We call this SPA-ELC, and say that it realizes a local diversity decoding of the received codeword. Our algorithm is listed in Alg. 2. Both SPA-PD and SPA-ELC perform

Algorithm 1 SPA-PD$(I_1, I_2, I_3, \alpha_0)$ [5]

1: // Input: $(\mathbf{y}, H, \alpha_0, I_1, I_2, I_3)$.
2: // Output: $\Theta^{-1}(\hat{\mathbf{x}})$.
3: $\alpha \leftarrow \alpha_0$.
4: **for** I_3 times **do**
5: $\quad \mathbf{L} \leftarrow (2/\sigma^2)\mathbf{y}$ and $\Theta \leftarrow \pi_0$ // identity permutation.
6: $\quad$ **for** I_2 times **do**
7: $\quad\quad \mu_{v\to u} \leftarrow L_v$, $\forall\ (u, v) \in \mathbf{TG}(H)$.
8: $\quad\quad$ Do I_1 flooding iterations, $\hat{\mathbf{x}} \leftarrow \text{SPA}(\mathbf{TG}(H))$.
9: $\quad\quad$ Take the hard decision of $\hat{\mathbf{x}}$ into $\mathbf{c}$, **stop** if $\mathbf{c}H^T = \mathbf{0}$.
10: $\quad\quad L_i \leftarrow (\hat{x}_i - L_i)\alpha + L_i$, $\ 0 \le i < n$.
11: $\quad\quad$ Draw random permutation $\pi \in \text{Aut}(\mathcal{C})$ [17].
12: $\quad\quad \mathbf{L} \leftarrow \pi(\mathbf{L})$ and $\Theta \leftarrow \pi(\Theta)$.
13: $\quad$ **end for**
14: $\quad \alpha \leftarrow \alpha_0 + (1 - \alpha_0)\frac{I_3}{I_3 - 1}$.
15: **end for**

a maximum of $\tau \triangleq I_1 I_2 I_3$ iterations. SPA update rules ensure that extrinsic information remains summed in bit nodes, such that an edge may be removed from $\mathbf{TG}(H)$ without loss of information. New edges, (u', v'), should be initialized according to line 13 in Alg. 2. Although neutral (i.e., LLR 0) messages will always be consistent with the convergence process, our experiments clearly indicate that this has the effect of 'diluting' the information, resulting in an increased decoding time and worse error rate performance.

The simple SPA-ELC decoder requires no preprocessing or any complex heuristic or rule to decide when or where to apply ELC. As ELC generates the orbit of $\mathcal{C}$, as well as the iso-orbit of each structure, diversity of structure can be achieved even for random codes, for which $|\text{Aut}(\mathcal{C})|$ is likely to be 1 while the size of the orbit is generally very large. However, going outside the iso-orbit means that we change the properties of H, most importantly in terms of density and number of short cycles. Ideally, the SPA-ELC decoder operates on a set of structurally distinct parity-check matrices, which are all of minimum weight. With the exception of codes with very strong structure, such as the extended Hamming code, the ELC-orbit of a code will contain structures of weight greater than the minimum. SPA-ELC should take measures against the negative impact of increased weight. In this paper, we adapt



Algorithm 2 SPA-ELC$(p, I_1, I_2, I_3, \alpha_0)$

1: // Input: $(\mathbf{y}, H, \alpha_0, I_1, I_2, I_3, p)$.
2: // Output: $\hat{\mathbf{x}}$.
3: $\alpha \leftarrow \alpha_0$.
4: **for** I_3 times **do**
5: $\quad \mathbf{L} \leftarrow (2/\sigma^2)\mathbf{y}$.
6: $\quad \mu_{v \to u} \leftarrow L_v,\ \forall (u, v) \in \mathbf{TG}(H)$.
7: $\quad$ **for** I_2 times **do**
8: $\quad\quad$ Do I_1 flooding iterations, $\hat{\mathbf{x}} \leftarrow \mathrm{SPA}(\mathbf{TG}(H))$.
9: $\quad\quad$ Take the hard decision of $\hat{\mathbf{x}}$ into $\mathbf{c}$, **stop** if $\mathbf{c}H^T = \mathbf{0}$.
10: $\quad\quad$ **for** p times **do**
11: $\quad\quad\quad$ Select random edge $e = (u, v) \in \mathbf{TG}(H)$.
12: $\quad\quad\quad$ $\mathbf{TG}(H) \leftarrow \mathrm{ELC}(\mathbf{TG}(H), e)$.
13: $\quad\quad\quad$ $\mu_{v' \to u'} \leftarrow (\hat{x}_{v'} - L_{v'})\alpha + L_{v'}$, $\forall (u', v') \in \mathbf{TG}(H),\ u' \in \mathcal{N}_v^u,\ v' \in \mathcal{N}_u^v$.
14: $\quad\quad$ **end for**
15: $\quad$ **end for**
16: $\quad \alpha \leftarrow \alpha_0 + (1 - \alpha_0)\frac{I_3}{I_3 - 1}$.
17: **end for**

the technique of damping to our graph-local perspective. Damping with the standard SPA, where $\mathbf{TG}(H)$ is fixed, does not work, so we only want to damp the parts of the graph which change. As opposed to SPA-PD, only a subgraph of $\mathbf{TG}(H)$ is affected by ELC, so we restrict damping to new edges in line 13. Note that SPA-ELC simplifies to a version without damping, denoted by SPA-ELC(p, I_1, τ), when $\alpha_0 = 1$, $I_2 = \tau / I_1 I_3$, and $I_3 = 1$. This is, simply, flooding iterations interspersed with random ELC operations, where new edges are initialized with the adjacent APP (line 13).

Currently, the SPA stopping criterion (i.e., the parameters used to flag when decoding should stop) is still implemented globally. However, a reasonable local solution would be to remove the syndrome check ($\mathbf{c}H^T = \mathbf{0}$) from the stopping criterion, and simply stop after $\hat{\tau}$ SPA-ELC iterations, where $\hat{\tau}$ can be empirically determined. However, this has obvious implications for complexity and latency. In some scenarios a stopping criterion can be dispensed with anyway–for instance when using the decoder as some form of distributed process controller, or for a pipelined implementation in which the iterations are rolled out.

Table 1: *Optimization of codes used in simulations*

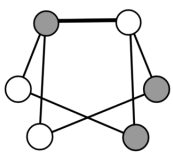
	Initial		Reduced		Reduced IP	
	W	C	W	C	W	C
[24, 12, 8]	* 96	366	* 96	147	* 96	366
[48, 24, 12]	320	4936	* 288	897	* 288	2672
[104, 52, 20]	1344	89138	1112	16946	1172	49839

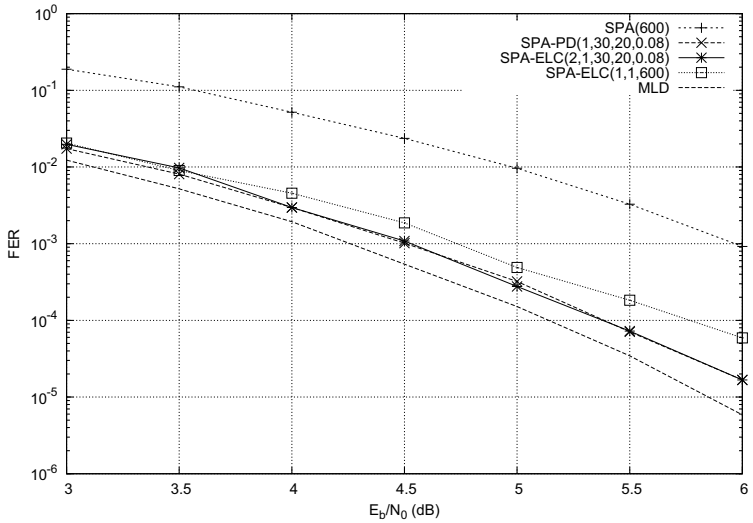
4 Results

We have compared SPA-ELC against standard SPA, and SPA-PD. Extended quadratic residue (EQR) codes were chosen for the comparison, mainly due to the fact that for some of these codes, $\mathrm{Aut}(\mathcal{C})$ can be generated by 3 generators [18]. In fact, our experiments have shown that EQR codes have Tanner graphs well-suited to SPA-ELC, at least for short blocklengths. The codes considered have parameters $[24, 12, 8]$ (the extended Golay code), $[48, 24, 12]$ (EQR48), and $[104, 52, 20]$ (EQR104). Parity-check matrices for the codes were preprocessed by heuristics to minimize the weight and the number of 4-cycles. The results are listed in Table 1, where columns marked 'W' and 'C' show the weight and the number of 4-cycles, respectively. Columns marked 'Initial' show the weight and the number of 4-cycles of the initial Tanner graph constructions. 'Reduced' and 'Reduced IP' refer to optimized Tanner graphs, where the latter is restricted to Tanner graphs in standard form. Entries marked by an asterisk correspond to minimum weight parity-check matrices.

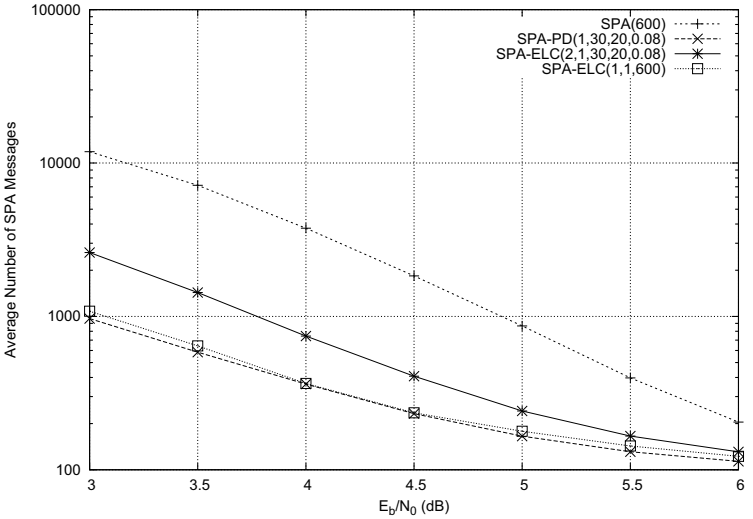
In Figs. 3-5, we show the frame error rate (FER) performance and the average number of SPA messages of SPA, SPA-PD, and SPA-ELC for the extended Golay code, the EQR48 code, and the EQR104 code, respectively, on the AWGN channel versus the signal-to-noise ratio, E_b/N_0.

The specific parameters used are indicated in the figure legends. For the extended Golay code and the EQR48 code, we set a maximum at $\tau = 600$ iterations, which we increased to $\tau = 2000$ to accommodate the larger EQR104 code. For SPA-ELC we have also included results without damping. Since SPA-ELC changes the weight of $\mathbf{TG}(H)$, we can not compare complexity by simply counting iterations. Since the



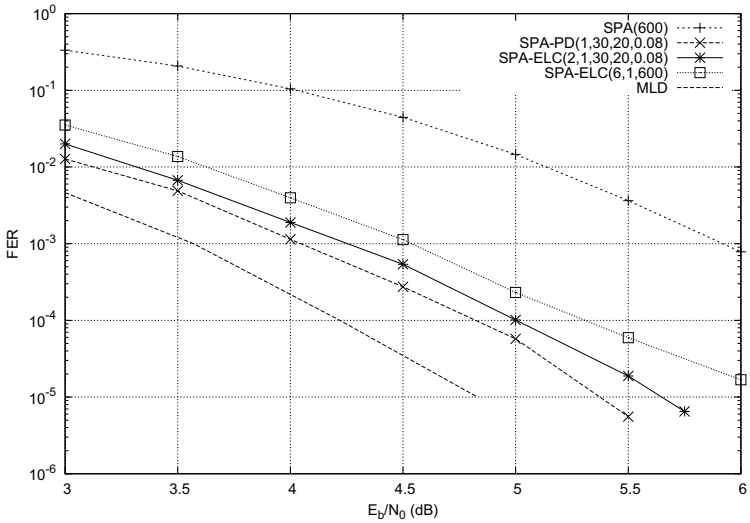


(a) FER performance

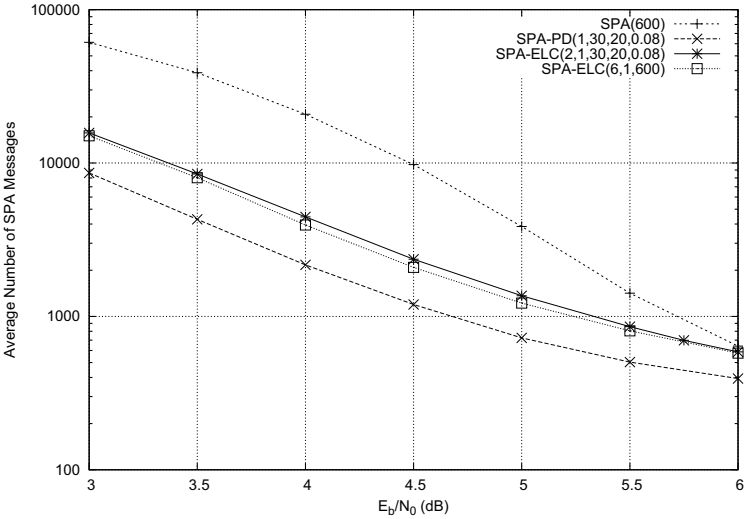


(b) Average number of SPA messages

Fig. 3: [24, 12, 8] *extended Golay code*

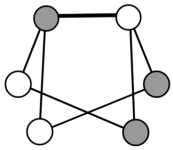


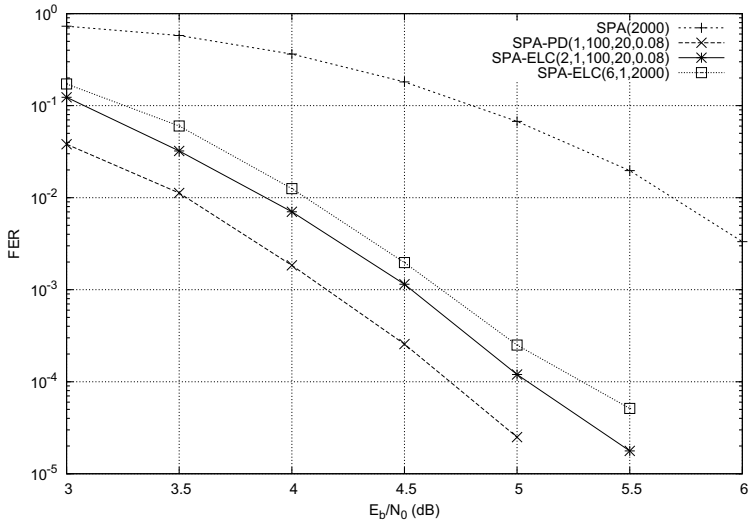
(a) FER performance



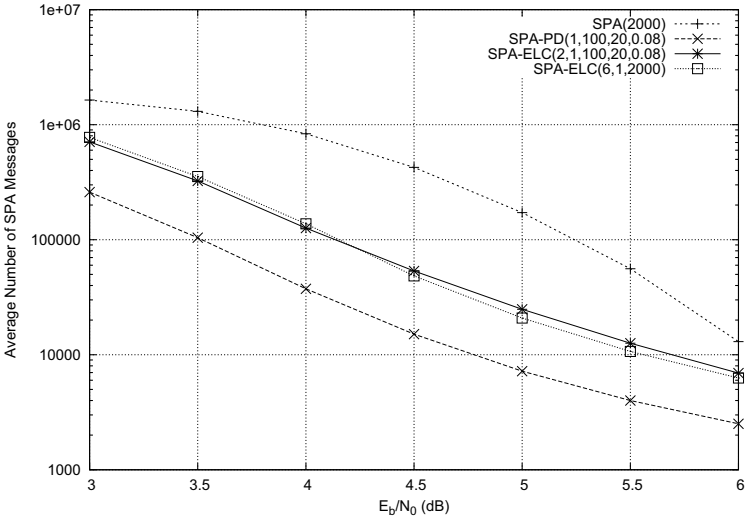
(b) Average number of SPA messages

Fig. 4: [48, 24, 12] *EQR48 code*





(a) FER performance



(b) Average number of SPA messages

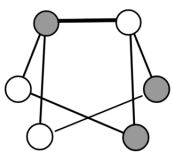
Fig. 5: [104, 52, 20] *EQR104 code*

complexity of one ELC operation is much smaller than the complexity of a SPA iteration, the total number of SPA messages may serve as a common measure for the complexity of the decoders. We have no initial syndrome check, so the number of iterations approaches 1 at high E_b/N_0. In the same way, the complexity approaches the average weight of the matrices encountered during decoding. Each FER point was simulated until at least 100 frame errors were observed.

From the figures, we observe that the SPA-ELC decoder outperforms standard SPA decoding, both in terms of FER and decoding complexity. The extended Golay code is a perfect example for demonstrating the benefits of SPA-ELC. The orbit of this code contains only two structures, where one is of minimum weight (weight 96) and the other only slightly more dense (weight 102), while the iso-orbit of the code is very large. Thus, we can extend SPA-PD with multiple Tanner graphs (two structures) while keeping the density low. Not surprisingly, SPA-ELC achieves the FER performance of SPA-PD, albeit with some complexity penalty. Note that the simple SPA-ELC decoder, without damping, approaches closely the complexity of SPA-PD at the cost of a slight loss in FER. For the larger codes, the sizes of the orbits are very large, and many structures are less suited for SPA-PD. Still, the same tradeoff between FER performance and complexity holds, based on whether or not we use damping. For the EQR48 code, we have observed a rich subset of the orbit containing minimum weight structures (weight 288). The optimum value of p (see line 10 in Alg. 2) was determined empirically.

5 Conclusion and Future Work

We have described a local diversity decoder, based on the SPA and the ELC operation. The SPA-ELC algorithm outperforms the standard SPA both in terms of error rate performance and complexity, and compares well against SPA-PD, despite the fact that SPA-PD uses global operations. Ongoing efforts are devoted to further improvements, and include; selective application of ELC, rather than random; devise techniques such that diversity may be restricted to sparse structures in the orbit; identify a code construction suited to SPA-ELC, for which the orbit contains several desirable structures even for large blocklengths.



ACKNOWLEDGMENT

The authors wish to thank Alban Goupil for providing the MLD curve for the EQR48 code.

REFERENCES

[1] GALLAGER, R. G.: Low-density parity-check codes. *IRE Trans. Inform. Theory* 8(1), 21–28, Jan. 1962.

[2] MACKAY, D. J. C., NEAL, R. M.: Good codes based on very sparse matrices. In *Proc. 5th IMA Conf. on Cryptography and Coding LNCS 1025*, pp. 100–111. Royal Agricultural College, Cirencester, UK, Dec. 1995.

[3] JIANG, J., NARAYANAN, K. R.: Iterative soft-input soft-output decoding of Reed-Solomon codes by adapting the parity-check matrix. *IEEE Trans. Inform. Theory* 52(8), 3746–3756, Aug. 2006.

[4] JIANG, J., NARAYANAN, K. R.: Iterative soft decoding of Reed-Solomon codes. *IEEE Commun. Lett.* 8(4), 244–246, Apr. 2004.

[5] HALFORD, T. R., CHUGG, K. M.: Random redundant iterative soft-in soft-out decoding. *IEEE Trans. Commun.* 56(4), 513–517, Apr. 2008.

[6] HEHN, T., HUBER, J. B., LAENDNER, S., MILENKOVIC, O.: Multiple-bases belief-propagation for decoding of short block codes. In *Proc. IEEE Int. Symp. Inform. Theory*, pp. 311–315. Nice, France, Jun. 2007.

[7] DIMNIK, I., BE'ERY, Y.: Improved random redundant iterative HDPC decoding. *IEEE Trans. Commun.* 57(7), 1982–1985, Jul. 2009.

[8] KNUDSEN, J. G., RIERA, C., PARKER, M. G., ROSNES, E.: Adaptive soft-decision decoding using edge local complementation. In *Proc. Second Int. Castle Meeting on Coding Theory and Applications, LNCS 5228*, pp. 82–94. Castillo de la Mota, Medina del Campo, Spain, Sep. 2008.

[9] MACWILLIAMS, F. J.: Permutation decoding of systematic codes. *Bell System Tech. J.* 43, 485–505, 1964.

[10] RICHARDSON, T., URBANKE, R. L.: *Modern Coding Theory*. Cambridge University Press, 2008.

[11] Richardson, T.: Error floors of LDPC codes. In *Proc. 41st Annual Allerton Conf. on Commun., Control, and Computing*, pp. 1426–1435. Monticello, IL, Oct. 2003.

[12] Bouchet, A.: Isotropic systems. *European J. Comb.* 8, 231–244, Jul. 1987.

[13] Danielsen, L. E., Parker, M. G.: Edge local complementation and equivalence of binary linear codes. *Des. Codes Cryptogr.* 49(1-3), 161–170, Dec. 2008.

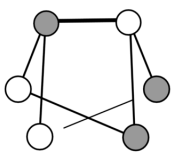
[14] Riera, C., Parker, M. G.: On Pivot orbits of Boolean functions, optimal codes and related topics. In *Fourth Int. Workshop on Optimal Codes and Related Topics*, pp. 248–253. Sofia, Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Jun. 2005.

[15] Hein, M., Dür, W., Eisert, J., Raussendorf, R., den Nest, M. V., Briegel, H. J.: Entanglement in graph states and its applications. In *Proc. Int. School of Physics "Enrico Fermi" on "Quantum Computers, Algorithms and Chaos"*. Varenna, Italy, 2005.

[16] Kschischang, F. R., Frey, B. J., Loeliger, H. A.: Factor graphs and the sum-product algorithm. *IEEE Trans. Inform. Theory* 47(2), 498–519, Feb. 2001.

[17] Celler, F., Leedham-Green, C. R., Murray, S. H., Niemeyer, A. C., O'Brien, E. A.: Generating random elements of a finite group. *Commun. in Algebra* 23, 4931–4948, 1995.

[18] MacWilliams, F. J., Sloane, N. J. A.: *The Theory of Error-Correcting Codes*. North Holland, 1977.



PAPER III

RANDOM EDGE-LOCAL COMPLEMENTATION WITH APPLICATIONS TO ITERATIVE DECODING OF HDPC CODES *

Joakim Grahl Knudsen Constanza Riera

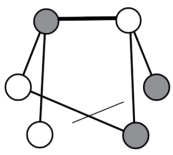
Lars Eirik Danielsen Matthew G. Parker

Eirik Rosnes

*Knudsen, J. G., Riera, C., Danielsen, L. E., Parker, M. G., Rosnes, E.: Random edge-local complementation with applications to iterative decoding of HDPC codes. Tech. Report no. 395, Department of Informatics, University of Bergen, Norway, Aug. 2010.

Submitted to IEEE Trans. Inform. Theory, 2010.

The material in this paper was presented in part at the International Zürich Seminar on Communications (IZS), Zürich, Switzerland, Mar. 2010, and at the Workshop on Linear Programming and Message-Passing Approaches to High-Density Parity-Check Codes and High-Density Graphical Models, Tel Aviv, Israel, Mar. 2010.



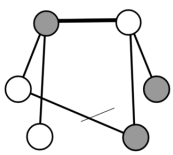
RANDOM EDGE-LOCAL COMPLEMENTATION WITH APPLICATIONS TO ITERATIVE DECODING OF HDPC CODES

Joakim Grahl Knudsen Constanza Riera

Lars Eirik Danielsen Matthew G. Parker

Eirik Rosnes

This paper describes the application of edge-local complementation (ELC), defined for a simple bipartite graph, to a Tanner graph associated with a binary linear code, $\mathcal{C}$. From a code perspective, various properties of ELC are described and discussed, mainly the special case of *isomorphic* ELC operations and the relationship to the automorphism group of the code, $\mathrm{Aut}(\mathcal{C})$, as well as the generalization of ELC to *weight-bounding* ELC (WB-ELC) operations under which the number of edges remains upper-bounded. The main motivation is the use of ELC to improve iterative soft-input soft-output decoding of high-density parity-check (HDPC) codes using the sum-product algorithm (SPA). By updating the edges of the Tanner graph using ELC additional *diversity* is achieved, while maintaining control on the weight of the Tanner graph (which also influences the number of short cycles) via WB-ELC. One motivation of ELC-based SPA decoding is the *locality* argument; that diversity is achieved by local graph action, and so is well-suited to the local actions that constitute the SPA and allows a parallel implementation.



Further applications of WB-ELC are described, including a heuristic to search for a systematic parity-check matrix (i.e., a Tanner graph) of reduced weight – a problem which has not received much focus in the literature. Extensive simulation data is shown for a range of HDPC codes, both in terms of matrix weight reduction, and error-rate performance of a proposed SPA-WBELC iterative decoding algorithm. A gain is reported over SPA decoding, and over a state-of-the-art algorithm to decode HDPC codes using permutations from $\mathrm{Aut}(\mathcal{C})$.

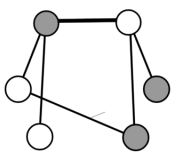
1 INTRODUCTION

Iterative soft decision decoding algorithms, applied to properly designed codes, have been shown to give results which, asymptotically, closely approach the theoretical limits established by Shannon [1]. The advent of turbo codes in 1993 [2] and the rediscovery of low-density parity-check (LDPC) codes at around the same time [3] (although LDPC codes were actually invented in 1962 [4] and re-discovered once already, in 1981 [5]) caused much attention to be focused on iterative decoding of large, random or pseudo-random, sparse block codes. The sum-product algorithm (SPA) is the standard soft decision iterative algorithm for decoding of LDPC codes on Tanner graphs [5]. The sparse, random nature of these codes make them well-suited for graph-based implementations, for which the SPA approximates optimum decoding at a complexity linear in blocklength. However, the large size and random nature of turbo and LDPC codes have negative implications when they are to be used in practice. This inspired researchers to adapt SPA decoding to small-size linear block codes, with blocklengths in the hundreds of bits or below. Small LDPC codes suffer a performance degradation due to finite-length effects and topological problems with the Tanner graph. At small blocklengths, however, one has the benefit of using strong, nonrandom codes – i.e., "classical codes" – for which useful properties are known, such as large minimum distance, $d_{\mathbf{min}}$, and nontrivial automorphism group. Today, these codes remain important components in technological devices, such as compact disc players and satellite communications, in which computational efficiency is still of vital importance (e.g., in low-power battery or solar powered circuitry). Important legacy codes are Bose-Chaudhuri-Hocquenghem (BCH), Reed-Solomon (RS), and quadratic residue (QR) codes. However, a large $d_{\mathbf{min}}$ or nontrivial

automorphism group are not obviously applicable to soft decision SPA decoding. For instance, as a parity-check matrix, H, can at best consist of $n-k$ linearly independent rows (codewords of the dual code, $\mathcal{C}^{\perp}$) of minimum weight, obviously the weight of H must increase with $d_{\mathbf{min}}(\mathcal{C}^{\perp})$. It is known that many families of codes – specifically BCH and RS codes – do not have Tanner graphs without cycles of length 4 [6]. Furthermore, these codes do typically not have sparse duals [7], so, when such codes are revisited from the context of iterative soft decoding, these are commonly referred to as *high-density parity-check* (HDPC) codes.

This has resulted in numerous creative approaches to adapting sub-optimal soft decision decoding to HDPC codes. These approaches can roughly be grouped into two categories, where one is characterized by an adaptive decoding based on the decoder state (the received noisy channel vector and the current codeword estimate) [8, 9]. The main idea is based on producing an error-free information set, which can, then, be re-encoded to produce a codeword. Such a most reliable basis (MRB) process can also be iterative, as in *order statistic decoding* (OSD) using different MRBs, either in terms of increased-order OSD (i.e., involving also some less reliable positions), or by simply using SPA iterations to update the codeword estimate and change the MRB [10, 11]. This way, a list of candidate codewords may be produced, from which an output is selected typically in terms of Euclidean distance from the received vector.

The other category is characterized by pseudorandom processes, involving code-preserving row operations or column permutations on the parity-check matrix, mainly to achieve increased diversity (i.e., different parity-check equations) during SPA decoding. Our work focuses on such random diversity-based algorithms. The aim of increased diversity is to decrease the effect of topological problems with the Tanner graph of the code, so that structural errors can be suppressed, e.g., by using randomized cyclic shifts on a cyclic code (stochastic shifting iterative decoding, SSID) [12]. One state-of-the-art decoder for HDPC codes, the iterative permutation decoder (SPA-PD) [13], generalizes SSID to applying random permutations from the automorphism group of the code and performs very well on BCH codes, as well as on QR codes [13–15], over the additive white Gaussian noise (AWGN) channel. Also, alternatively or in conjunction [14, 16], multiple bases (matrices) for the same dual code may be used to gain diversity. These matrices are



usually preprocessed and typically optimized on weight [17], but can also be produced in real-time [15, 18].

This paper describes the pseudorandom use of a simple graph operation known as edge-local complementation (ELC) [19, 20] to improve the performance of iterative decoding [21, 22]. One advantage of ELC-based SPA decoding is the *locality* argument; that diversity is achieved by local graph action, and so is well-suited to the local actions that constitute the SPA. Diversity stems from the change in Tanner graph due to the complementation of edges in a local subgraph, corresponding to row-additions on the associated parity-check matrix. The locality property, which allows for parallel implementation of the SPA, also has beneficial effects on the overall complexity of an ELC-based decoding algorithm in many contexts. The effect of ELC on a graph is explored, and we define a subset of ELC operations under which the weight of the graph is upper-bounded (to within some threshold value). We identify and describe all possible occurrences of single and double application of ELC which is *weight-bounding* ELC (WB-ELC). We also present a further specialization of WB-ELC to *isomorphic* ELC (iso-ELC), under which the *structure* of the graph is invariant. These properties (weight and structure) are important from a coding perspective (where the graph is a Tanner graph), and are used to improve the error-rate performance of a soft-input soft-output (SISO) HDPC decoder based on interleaving SPA iterations with random ELC operations; giving a novel SPA-ELC and a SPA-WBELC decoding algorithm. A one-to-one relationship between iso-ELC operations and permutations from the automorphism group of the code is presented, such that the SPA-PD algorithm may be used as a relevant benchmark in terms of performance, simulated for various HDPC codes. We also propose a related application of WB-ELC to reduce (or even minimize) the weight of a graph, i.e., finding a reduced-weight systematic parity-check matrix for the code – an instance of weight reduction which has not received much focus in the literature.

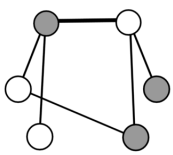
1.1 Outline

This paper is organized as follows. The ELC operation, which is defined for a simple graph, is described in Section 2. A discussion on the action of ELC, in terms of the resulting graphs, focuses, firstly, on structurally distinct graphs, and, secondly, on isomorphic graphs with a link to $\mathrm{Aut}(\mathcal{C})$. Section 3 presents a generalization of iso-ELC to WB-ELC, i.e.,

the action of ELC is discussed in terms of a maximum permitted weight of the resulting graphs. We identify the specific subgraphs on which one or a sequence of two (depth-1 or 2) ELC operations are WB-ELC, and go on to prove how these cases cover all possible subgraphs within depth-2. Adhering to the locality argument, we also prove how the search space for depth-2 WB-ELC is limited to neighboring pairs of edges (distance 1 or 2 edges apart), such that the impact of WB-ELC is confined to a local subgraph of maximum diameter 4. Section 4 describes an algorithm to enumerate all WB-ELC operations (within depth-2) on a given graph, and to within some threshold. A bound on the complexity of this algorithm is derived, and is verified using simulations on graphs of different sizes. Several applications of this algorithm are described, centered around the use of WB-ELC in a iterative decoding setting. As a preprocessing stage, the algorithm can be used to minimize the weight of a graph (i.e., a systematic H), and also to find such a graph from which many other distinct reduced-weight graphs can be reached using WB-ELC. Finally, in Section 5, the use of ELC as a source of diversity during SPA decoding is described. Two proposed decoding algorithms – SPA-ELC and SPA-WBELC – are described, simulated, and compared against other relevant decoding algorithms on a range of HDPC codes. To facilitate fair comparisons, a common framework for iterative SISO HDPC decoding is presented, within which all decoding algorithms are implemented. Empirical data is presented on both choices of decoder parameters, resulting error-rate performance, and decoding complexity. Certain implementational remarks (for WB-ELC) are presented in an appendix.

1.2 Preliminaries

We begin by introducing some notation used in the following sections. We use uppercase italics and square brackets for matrices; script notation and curly brackets for sets; and boldface notation for vectors. A binary linear code $\mathcal{C}$ of length n, dimension k, and minimum distance $d_{\mathbf{min}}$ is denoted by $[n, k, d_{\mathbf{min}}]$, where $d_{\mathbf{min}}$ is defined as the minimum Hamming weight of any nonzero codeword. The weight enumerator of $\mathcal{C}$ is a vector, $\mathbf{a}$, for which a_i contains the number of codewords of weight i. Necessarily, $a_i = 0,\ i < d_{\mathbf{min}}$. If $a_i = 0,\ i \neq 0 \bmod 2$ (odd weights), then $\mathcal{C}$ is *even.* If this also holds for mod 4, $\mathcal{C}$ is *doubly even.* The column indices $0, 1, \ldots, n-1$ are referred to as the *coordinates* of the code. The dual code is $\mathcal{C}^{\perp}$, containing the codewords



orthogonal to $\mathcal{C}$, and if $\mathcal{C} = \mathcal{C}^{\perp}$ we say the code is self-dual. Permutations are written in cycle notation, where we only specify the indices of the affected positions. For example, given a length-6 vector $\mathbf{v}$ and a permutation $\pi = (0, 1, 2)(3, 4)$, then $\mathbf{u} = \pi(\mathbf{v})$ means $v_0 \rightarrow u_1$, $v_1 \rightarrow u_2$, $v_2 \rightarrow u_0$, $v_3 \rightarrow u_4$, and $v_4 \rightarrow u_3$, while $v_5 \rightarrow u_5$. Similarily, $\pi(H)$ permutes the columns of a matrix, H. The identity permutation, affecting no positions, is, then, $\pi = \emptyset$. The automorphism group of the code, $\mathrm{Aut}(\mathcal{C})$, is the group of permutations which preserve the code, $\mathrm{Aut}(\mathcal{C}) = \{\sigma : \sigma(\mathcal{C}) = \mathcal{C}\}$. It is well known that $\mathrm{Aut}(\mathcal{C}) = \mathrm{Aut}(\mathcal{C}^{\perp})$ [23], and permutations are typically applied to H (which generates $\mathcal{C}^{\perp}$) during decoding, or to the soft-input vector containing the *a posteriori* probability (APP) values [13]. If $\mathrm{Aut}(\mathcal{C})$ consists of the identity permutation alone, we say that $\mathrm{Aut}(\mathcal{C})$ is trivial.

Let I_k be the identity matrix of size k, where we use the shorthand notation I when the dimension is obvious. The generator matrix, G, generates $\mathcal{C}$ (contains k linearly independent codewords forming a basis for the code), and the parity-check matrix, H, generates $\mathcal{C}^{\perp}$. This gives $GH^T = 0$, where $(\cdot)^T$ denotes the transpose of its argument. In the context of iterative graph-based decoding of $\mathcal{C}$, the focus is on H rather than G. H is said to be *systematic* if its columns can be reordered into the *standard form*

$$\pi(H) = [I_{n-k} \mid P] \tag{1}$$

by some column permutation, π. In turn, a standard form generator matrix is $\pi(G) = [P^T \mid I_k]$. This permutation, π, does not in general preserve the code. An information set, $\mathcal{I}$, of the code corresponds to a set of k columns in G which can be reduced to an identity submatrix by means of Gaussian elimination (GE). The $n - k$ columns at positions $\mathcal{P} := \{0, 1, \ldots, n-1\} \setminus \mathcal{I}$ form a parity set. Note that an information set corresponds to a parity set of the dual code, such that $\mathcal{I}$ refers to the P-part of H. In a systematic parity-check matrix, the columns indexed by $\mathcal{P}$ are referred to as systematic (weight-1) columns, while the remaining columns (weight greater than one) are *nonsystematic.* The (row) index of the single nonzero entry of a systematic column $\mathbf{h}_i$, $i \in \mathcal{P}$, is denoted by $\mathrm{row}(i) \in [0, n-k)$. In standard form, $\mathrm{row}(i) = i$, $0 \leq i < n - k$. In systematic form, the (Hamming) weight of H, denoted by $|H|$, is the number of nonzero entries in H, and the weight of H is lower-bounded by

$$\max\left(k(d_{\mathbf{min}}(\mathcal{C}) - 1) + n - k,\ (n-k)d_{\mathbf{min}}(\mathcal{C}^{\perp})\right). \tag{2}$$

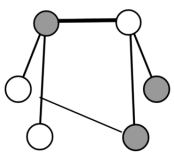
The Tanner graph, $\mathbf{TG}(H)$, associated with H is a $(2n-k)$-node bipartite graph with adjacency matrix $\mathbf{TG}(H) = \left[\begin{smallmatrix} 0 & H \\ H^T & 0 \end{smallmatrix}\right]$. (At some abuse of notation, we denote both the graph and its adjacency matrix by, in this case, $\mathbf{TG}(H)$.) In the remainder of this paper, we will assume that H is systematic. The n variable nodes, corresponding to the columns of H, are partitioned into $|\mathcal{P}| = n-k$ systematic and $|\mathcal{I}| = k$ nonsystematic nodes, where the former have degree one (disregarding the Forney style "half-edge" containing the channel input to each variable node). The $n-k$ check nodes of $\mathbf{TG}(H)$, corresponding to the rows of H, each have an associated (adjacent) systematic variable node. By grouping each check node with its associated systematic (variable) node, an n-node, $(n-k,\ k)$-bipartite, simple (i.e., undirected, with no double edges or loops) graph is produced, with adjacency matrix

$$\mathcal{G} = (\mathcal{U} \cup \mathcal{V},\ \mathcal{E}) = \pi^{-1} \left[\begin{smallmatrix} 0 & P \\ P^T & 0 \end{smallmatrix}\right] \tag{3}$$

where π^{-1} undoes the reordering in (1). The bipartition $(\mathcal{U},\ \mathcal{V})$ contains the $n-k$ grouped check/systematic variable nodes and the nonsystematic variable nodes, respectively. Furthermore, a permutation (here, π^{-1}) acts on both columns and rows of $\mathcal{G}$. By keeping a record of the bipartition, $(\mathcal{U},\ \mathcal{V})$, at all times, this amounts to a one-to-one mapping between a Tanner graph (i.e., a code) and a simple bipartite graph. In summary, given a code represented by $\mathbf{TG}(H)$, we construct a simple graph by ignoring the systematic variable nodes – see Example 1. The number of edges in $\mathcal{G}$ is

$$|\mathcal{G}| = |\mathcal{E}| = |H| - (n-k) \tag{4}$$

which we refer to as the weight of $\mathcal{G}$. If nodes in $\mathcal{U}$ and $\mathcal{V}$ have average degree $\bar{\rho}$ and $\bar{\gamma}$, respectively, we have that $|\mathcal{G}| = k\bar{\gamma} = (n-k)\bar{\rho}$. The local neighborhood of a node v is the set of nodes adjacent to v, and is denoted by $\mathcal{N}_v$, while $\mathcal{N}_v^u$ is shorthand notation for $\mathcal{N}_v \setminus \{u\}$. Let $\mathcal{E}_{A,B}$ denote the subgraph induced by the nodes in $A \cup B$ – i.e., it is a set of $|\mathcal{E}_{A,B}|$ edges. Furthermore, $\mathcal{E}_{u,v}$ is shorthand notation for $\mathcal{E}_{\mathcal{N}_u^v, \mathcal{N}_v^u}$, the local neighborhood of the edge (u,v). We use the compact notation $\{(u,v), \ldots, (u',v')\}$ for an ordered list of edges. Define the *distance* between edges (or *nonedges*) (u,v) and (u',v') as the shortest path between the sets of endpoints (nodes), $\{u,v\}$ and $\{u',v'\}$.



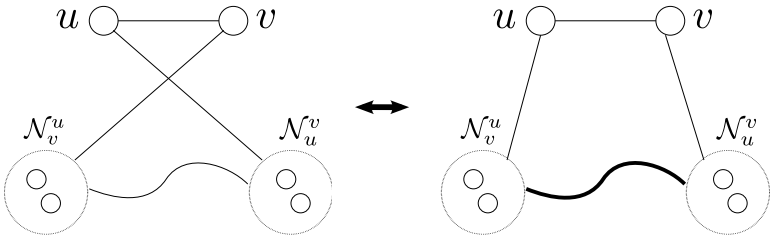


Fig. 1: *ELC on edge (u, v) of a bipartite simple graph. Curved links indicate arbitrary edges. Bold links mean that the edges connecting the two sets have been complemented; edges are replaced by nonedges, and vice versa. This graph may be a subgraph of a larger graph, in which case the rest of the graph remains unchanged.*

2 Edge-Local Complementation

ELC is defined on an edge of a simple graph, $(u, v) \in \mathcal{G}$ [19]. We consider only bipartite graphs in this work, which simplifies the description. ELC on an edge (u, v) will *complement* the edges of $\mathcal{E}_{u,v}$, replacing edges with nonedges and vice versa, followed by swapping the nodes u and v – see Fig. 1. In this sense, we say that ELC is a local operation as it only affects edges within a distance of one from the ELC edge, (u, v). The resulting graph, after ELC, is denoted by $\mathcal{G}_{(u,v)}$. ELC is a self-invertible operation as two ELC operations on the same edge is the identity operation, $\mathcal{G}_{(u,v),(u,v)} = \mathcal{G}$. The number of edges affected (inserted or removed) by the complementation of ELC is, on average,

$$|\mathcal{N}_u^v||\mathcal{N}_v^u| \approx (\bar{\gamma} - 1)(\bar{\rho} - 1). \tag{5}$$

Assuming $k = n - k$ and $\bar{\gamma} = \bar{\rho}$,[1] we may express the complexity of ELC in terms of number of edge-operations performed, by using the average node degree, $\bar{\gamma}$. The complementation has the effect of inverting a local neighborhood, which may increase or decrease the weight, depending on the particular graph on which we perform an ELC operation. The effect of repeated ELC (on random edges) is seen in Section 5 to stabilize the weight of $|\mathcal{G}|$ at around 50%, i.e.

$$|\mathcal{G}| \approx k^2/2 \tag{6}$$

[1]This is a fairly realistic assumption for rate-1/2 HDPC codes.

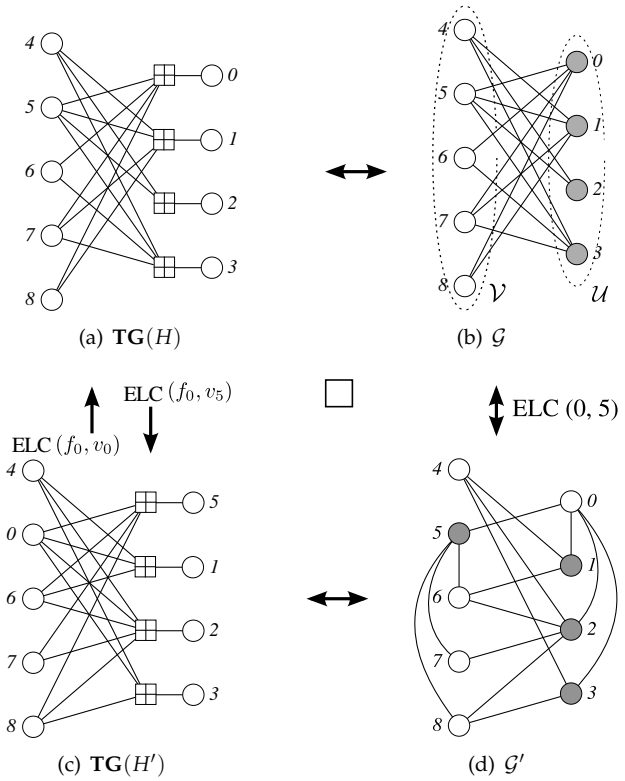


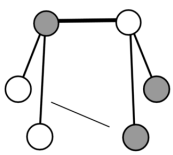
Fig. 2: *Example of ELC on a small* $[9, 4, 4]$ *code, showing also the corresponding Tanner graphs. White and grey nodes correspond to* $\mathcal{V}$ *and* $\mathcal{U}$*, respectively.*

or, equivalently, $|H| \approx \frac{k(n-k)}{2} + (n-k) = k(k+2)/2$. The complexity of ELC at this expected weight is important to identify. Taking $\bar{\gamma} = k/2$, (5) gives $(k/2-1)^2 = k^2/4 - k + 1$.

From the matrix perspective, it is easily seen that one ELC operation implements the reduction stage of GE on a single column, as shown in the following example.

Example 1. *Consider the optimal (in terms of maximum* $d_{\min}$ *for blocklength 9 and dimension 4)* $[9, 4, 4]$ *code, with parity-check matrix*

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



and the corresponding Tanner graph as shown in Fig. 2(a). Fig. 2(b) shows the corresponding simple bipartite graph, while Fig. 2(d) shows an example of ELC on the edge $(0,5)$*, with the resulting Tanner graph,* $\mathbf{TG}(H')$*, in Fig. 2(c).*

Note that it may be convenient to implement ELC directly on $\mathbf{TG}(H)$*, at the cost of minor modifications to the implementation; for instance, the inverse of ELC on* (f_0, v_5) *is* (f_0, v_0)*, due to the swap. By considering the resulting* H'*, it can be seen that ELC is, in fact, a graph implementation of a single stage (column) of GE; adding row 0 to rows 1, 2, and 3 to get*

$$H' = \begin{bmatrix} \mathbf{1}\,0\,0\,0\,0\,\mathbf{1}\,1\,1\,1 \\ \mathbf{1}\,1\,0\,0\,1\,\mathbf{0}\,1\,0\,0 \\ \mathbf{1}\,0\,1\,0\,1\,\mathbf{0}\,1\,1\,1 \\ \mathbf{1}\,0\,0\,1\,1\,\mathbf{0}\,0\,0\,1 \end{bmatrix}.$$

Now, column 5 has been reduced to systematic form, and the row additions have effectively swapped columns 0 and 5 between $\mathcal{I}$ *and* $\mathcal{P}$*, giving a new information (and parity) set of the code.*

The link to GE emphasizes that ELC will always preserve the code (i.e., the null space of H). Implemented on the Tanner graph, the inverse operation must reflect the changed information set, as shown in Fig. 2. In this work, we refer to ELC on $\mathcal{G}$ and on $\mathbf{TG}(H)$ interchangeably, using the simple graph definition mainly to simplify descriptions and proofs on ELC, while using the Tanner graph version for practical implementations.

2.1 Minimum-Length ELC Sequence Between two Structures

The set of structurally distinct graphs which arise by iteratively doing ELC on all edges of a bipartite simple graph $\mathcal{G}$, pruning the recursion tree on repeated structures, is known as the orbit, $\text{orbit}(\mathcal{G})$, of the graph. This orbit is the same for all graphs corresponding to the same code, $\mathcal{C}$, so we may refer to it as the orbit of the code, $\text{orbit}(\mathcal{C})$. Structural distinctness is in terms of graph isomorphism. By using the software package Nauty [24], we obtain a canonical form of a simple graph, denoted by $N(\mathcal{G})$. Thus, for two simple graphs $\mathcal{G}$ and $\mathcal{G}'$, we have that $\mathcal{G} \overset{\text{iso}}{=} \mathcal{G}' \Leftrightarrow N(\mathcal{G}) = N(\mathcal{G}')$. The one-to-one relationship between a graph and a parity-check matrix means that we may also speak of the orbit as a set of parity-check matrices, $\text{orbit}(H)$. We will use these references interchangeably in the following.

If a code has only one structure in its orbit, we say that it is an *ELC-preserved* code (or, equivalently, since this graph is unique, we may say that the graph is ELC-preserved) [25].

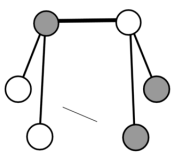
Theorem 1 (ELC sequence)**.** *A minimum-length ELC sequence*

$$\mathbf{e} = \{(u_0, v_0), (u_1, v_1), \ldots, (u_{l-1}, v_{l-1})\}$$

can be found to convert a systematic matrix H into another systematic matrix H', where H and H' span the same space (they are in the same orbit), by comparing the corresponding bipartitions as represented by the parity sets $\mathcal{P}$ and $\mathcal{P}'$. The length, l, of **e** *is $0 \leq l \leq \min(n-k,\ k)$. Depending on H, the sequence* **e** *may not be unique, so equivalent sequences may be derived from $\mathcal{P}$ and $\mathcal{P}'$.*

Proof. ELC generates the entire orbit [20], and in particular all systematic parity-check matrices for the corresponding code, so such a sequence **e** must exist. Since a systematic basis for a (dual) code is uniquely defined (up to row permutations) by its parity set, the information set (i.e., the P-part of H) is a function of the parity set. Thus, by comparing $\mathcal{P}$ and $\mathcal{P}'$, we determine which coordinates are in opposite partitions, and shall be swapped. Each ELC operation preserves the (dual) code, and has the effect of swapping a pair of columns in H from $\mathcal{I}$ to $\mathcal{P}$, along with some "residual" modifications to H resulting from the row-additions. To modify H into H', we may thus focus on swapping the corresponding pairs of columns from $\mathcal{P}$ into $\mathcal{P}'$, thus giving the I-part of H', and the residual modifications must "resolve" into the required P-part (since the P-part is unique given the I-part). Then, the submatrices I and I' are equal, from which it follows that $P = P'$, such that $H = H'$ (up to row-equivalence). Alg. 1 is a constructive proof of this theorem, showing how $\mathcal{P}$ and $\mathcal{P}'$ are used to determine a corresponding ELC sequence. Due to the possible row-equivalence, several equivalent ELC sequences (of equal length) may exist [26]. ELC has the effect of swapping exactly one pair of positions between $\mathcal{I}$ and $\mathcal{P}$, so the length of **e** must be exactly $l = |\mathcal{P} \setminus \mathcal{P}'|$, which is upper-bounded by $\min(n-k,\ k)$. □

The difference (coordinates to swap) corresponds to the sets $\mathcal{L} = \mathcal{P} \setminus \mathcal{P}'$ and $\mathcal{S} = \mathcal{P}' \setminus \mathcal{P}$. As each position in the identity (sub) matrix is unique, $r \in \mathcal{L}$ can be viewed as a row-index, where r is chosen such that $(\mathrm{row}(r), s) \in \mathbf{TG}(H)$. When several valid choices of r exist for a coordinate $s \in \mathcal{S}$, a branch point arises in the algorithm leading to an



Algorithm 1 MIN_ELC(H, H')

1: $\mathcal{L} := \mathcal{P} \setminus \mathcal{P}'$
2: $\mathcal{S} := \mathcal{P}' \setminus \mathcal{P}$
3: $\mathbf{e} := \emptyset$
4: **while** $\mathcal{S} \neq \emptyset$ **do**
5: choose and remove any $s \in \mathcal{S}$
6: choose and remove any $r \in \mathcal{L}$ s.t. $(\text{row}(r), s) \in \mathbf{TG}(H)$
7: ELC on $(\text{row}(r), s)$ on $\mathbf{TG}(H)$
8: $\mathbf{e} := \mathbf{e} \cup (\text{row}(r), s)$
9: **end while**

equivalent ELC sequence. The resulting Tanner graphs are exactly the same (although the *matrices* may be different, but only in terms of row permutations).

Example 2. *Consider the* $[14, 7, 3]$ *doubly circulant QR code, represented by a parity-check matrix*

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

The orbit of this code consists of 11 graphs. Choosing two distinct graphs, $\mathcal{G}$ *and* $\mathcal{G}'$*, from the orbit of the code we must have that* $N(\mathcal{G}) \neq N(\mathcal{G}')$*. Let* H *be a parity-check matrix corresponding to* $\mathcal{G}$*, and let* H' *correspond to* $\mathcal{G}'$*, where*

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

It is easily seen that $\mathcal{G}$ *and* $\mathcal{G}'$ *are indeed nonisomorphic, simply by verifying that* $|H| \neq |H'|$*. The parity sets are* $\mathcal{P} = \{1, 2, 3, 5, 6, 9, 11\}$ *and* $\mathcal{P}' = \{0, 2, 3, 5, 9, 11, 13\}$*. Now, Alg. 1 computes* $\mathcal{L} = \{1, 6\}$ *and* $\mathcal{S} = \{0, 13\}$*. Choosing, say,* $s = 13$*, we find that* $r = 1$ *gives* $(\text{row}(1), 13) = (1, 13) \in \mathbf{TG}(H)$*. ELC on edge* $(1, 13)$ *of* H *gives the following matrix*

$$H_{(1,13)} = \begin{bmatrix} 0&1&0&0&0&0&0&0&1&1&1&0&1&0 \\ 1&1&0&0&1&0&0&1&1&0&1&0&0&1 \\ 0&1&1&0&1&0&0&0&0&0&0&0&0&0 \\ 1&1&0&1&0&0&0&0&0&0&0&0&0&0 \\ 1&1&0&0&0&0&0&1&0&0&1&1&1&0 \\ 1&0&0&0&1&1&0&0&0&0&0&0&0&0 \\ 1&1&0&0&1&0&1&0&0&0&0&0&0&0 \end{bmatrix}.$$

Then, the remaining choice of $s = 0$ *gives* $r = 6$*, where* $(\mathrm{row}(6), 0) = (6, 0) \in \mathbf{TG}(H)$ *and, after ELC on* $(6, 0)$ *on* $H_{(1,13)}$*, we get*

$$H_{\{(1,13),(6,0)\}} = \begin{bmatrix} 0&1&0&0&0&0&0&0&1&1&1&0&1&0 \\ 0&0&0&0&0&0&1&1&1&0&1&0&0&1 \\ 0&1&1&0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&1&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&1&1&0&0&1&1&1&0 \\ 0&1&0&0&0&1&1&0&0&0&0&0&0&0 \\ 1&1&0&0&1&0&1&0&0&0&0&0&0&0 \end{bmatrix}$$

which gives the same Tanner graph as $\mathbf{TG}(H')$ *(by swapping rows* 1 *and* 6*). That the ELC sequence* $\mathbf{e} = \{(1, 13), (6, 0)\}$ *is not unique is reflected by Alg. 1. Different choices would result in the sequences* $\{(1, 0), (6, 13)\}$ *and* $\{(6, 13), (1, 0)\}$*, which both give the "target" matrix,* H'*. The sequence* $\{(6, 0), (1, 13)\}$ *is not possible, since the edge* $(6, 0) \notin H$*.*[2]

2.2 Tanner Graph Invariants

In the context of graph-based, iterative decoding, we are interested in discerning distinct Tanner graphs, when these may correspond to isomorphic simple bipartite graphs. A code is preserved under elementary row operations (i.e., row additions and permutations) on the associated basis (parity-check matrix), so we define two parity-check matrices, H and H', as isomorphic if and only if the rows of H' can be permuted to give the exact same matrix H (or vice versa). A parity-check matrix, H, can be put in canonical form, denoted by $R(H)$, by sorting its rows in lexicographical order, $\mathbf{TG}(H) = \mathbf{TG}(H') \Leftrightarrow R(H) = R(H')$.

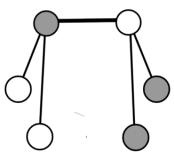
A sequence of ELC operations, $\mathbf{e}$, connecting two parity-check matrices for the same code, $H \neq H'$, with the same canonical form, i.e., $N(\mathcal{G}) = N(\mathcal{G}')$, has previously been defined as an *iso-ELC sequence* [16].

Definition 1. *A permutation* $\theta \in \mathrm{Aut}(\mathcal{C})$ *is called trivial if and only if* $\mathbf{TG}(H) = \mathbf{TG}(\theta(H))$.

Theorem 2 (ELC finds entire $\mathrm{Aut}(\mathcal{C})$)**.** *Each nontrivial permutation in* $\mathrm{Aut}(\mathcal{C})$*, for a given* H*, is associated with an iso-ELC sequence,* $\mathbf{e}$*, of length* l*, for* $1 \leq l \leq \min(n - k,\ k) = \min(\dim(\mathcal{C}^{\perp}), \dim(\mathcal{C}))$*. The particular sequence depends on the parity set,* $\mathcal{P}$*, (i.e., on* H*), and is not unique.*

Proof. For each nontrivial permutation $\sigma \in \mathrm{Aut}(\mathcal{C})$, H and $\sigma(H)$ are two (nonisomorphic) systematic parity-check matrices for $\mathcal{C}$, i.e., they both span the same space, and the result follows from Theorem 1. □

[2]These equivalent ELC sequences also follow from [26].



Proposition 1 (Trivial permutation)**.** *A permutation $\theta \in \mathrm{Aut}(\mathcal{C})$ is trivial if and only if it permutes no positions between $\mathcal{I}$ and $\mathcal{P}$ for the given H. Furthermore, the set of trivial permutations forms a subgroup $\mathcal{D}_H \preceq \mathrm{Aut}(\mathcal{C})$.*

Proof. If a permutation θ is trivial for a given parity-check matrix H, then (by definition) H and $\theta(H)$ are row-equivalent. Since H and $\theta(H)$ are row-equivalent, θ is constrained to permute the columns from $\mathcal{P}$ (i.e., the I-part of H) to indices from $\mathcal{P}$ (and thus permute the columns from $\mathcal{I}$ (i.e., the P-part of H) to indices from $\mathcal{I}$, and the result follows.

Conversely, if a permutation θ permutes no positions between $\mathcal{I}$ and $\mathcal{P}$ for the given H, then the resulting matrix $\theta(H)$ will have weight-1 columns in exactly the same positions as H, i.e., in the positions in $\mathcal{P}$. Permuting the rows of $\theta(H)$ such that the I-parts of H and $\theta(H)$ become identical will also make the P-parts identical (the P-part is a function of the I-part), from which it follows that H and $\theta(H)$ are row-equivalent, and the permutation θ is (by definition) trivial.

Finally, we need to prove that the set of trivial permutations forms a subgroup of $\mathrm{Aut}(\mathcal{C})$. This follows directly from the first result (i.e., that a permutation $\theta \in \mathrm{Aut}(\mathcal{C})$ is trivial if and only if it permutes no positions between $\mathcal{I}$ and $\mathcal{P}$), since the composition of two such permutations obviously permutes no positions between $\mathcal{I}$ and $\mathcal{P}$. □

In the following, we may denote the subgroup $\mathcal{D}_H$ simply by $\mathcal{D}$ when the matrix is obvious from the context. The subgroup $\mathcal{D}$ is *not* a code property, but a property of H. Furthermore, since $\mathcal{D}$ is a subgroup, we can decompose $\mathrm{Aut}(\mathcal{C})$ into a union of cosets of $\mathcal{D}$;

$$\mathrm{Aut}(\mathcal{C}) = \{\mathcal{D} \circ \sigma_0\} \cup \{\mathcal{D} \circ \sigma_1\} \cup \cdots \cup \{\mathcal{D} \circ \sigma_{|\mathrm{Aut}(\mathcal{C})|/|\mathcal{D}|-1}\}$$

where $\mathcal{K}_H = \{\sigma_0, \ldots, \sigma_{|\mathrm{Aut}(\mathcal{C})|/|\mathcal{D}|-1}\}$ is a set of coset leaders, given H, which we may denote simply by $\mathcal{K}$ (as we do for the subgroup $\mathcal{D}$) when the matrix is obvious from the context, and σ_0 is the identity permutation.

Alg. 1 can be used to convert any $\sigma \in \mathrm{Aut}(\mathcal{C})$ into an equivalent iso-ELC sequence, $\mathbf{e}$, by taking as input H and $H' = \sigma(H)$. The corresponding iso-ELC sequence depends on both σ and H, and we may emphasize this by the notation, $\mathbf{e}_{\sigma,H}$. Then we have that $R(\sigma(H)) = R(\mathbf{e}_{\sigma,H}(H))$.

Proposition 2. *Given a parity-check matrix H, $\mathbf{e}_{\sigma,H}$ is an iso-ELC sequence representation of* all *permutations in the coset $\mathcal{D} \circ \sigma$, $\sigma \in \mathrm{Aut}(\mathcal{C})$.*

Proof. The coset decomposition is in terms of row equivalence, i.e., $R(\sigma(H) = R(\sigma'(H))$ for any $\sigma' \in \mathcal{D} \circ \sigma$, and the result follows. □

The set $\mathcal{K}_H \setminus \{\sigma_0\}$ contains permutations from $\mathrm{Aut}(\mathcal{C})$ which give a distinct (i.e., not row-equivalent) parity-check matrix $\sigma(H)$, where $\sigma \in \mathcal{K}_H \setminus \{\sigma_0\}$. Each coset leader σ corresponds to a matrix $R(\sigma(H))$, representing the $|\mathcal{D}|$ row-equivalent matrices $\theta(\sigma(H))$, $\forall\ \theta \in \mathcal{D}$. In other words, these all correspond to the same Tanner graph for $\mathcal{C}$. In this sense, the set of coset leaders is not unique (any $\sigma' \in \mathcal{D} \circ \sigma$, where $\sigma \neq \sigma_0$, could be used as a coset leader), which means that $\mathcal{K}_H$ is not unique even for a given H. Since σ_0 is the identity mapping, $\mathcal{K}_H$ can be a group.

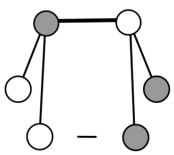
The set of (distinct) Tanner graphs resulting from the permutations in $\mathcal{K}_H$ comprise the *iso-orbit* of H,[3] $\{\sigma_0(H), \ldots, \sigma_{|\mathcal{K}|-1}(H)\}$. These Tanner graphs are all distinct, but correspond to isomorphic simple graphs, $R(H) \neq R(\sigma(H))$, but $N(\mathcal{G}) = N(\sigma(\mathcal{G}))$, $\forall\ \sigma \in \mathcal{K}_H \setminus \{\sigma_0\}$. The iso-orbit can be partitioned into disjoint subsets according to the (minimal) length, $0 \leq l \leq \min(n-k,\ k)$, of the corresponding ELC sequences, $\mathcal{K}_H^l = \{\sigma \in \mathcal{K}_H : |\mathcal{P} \setminus \sigma(\mathcal{P})| = l\}$. In particular, $\mathcal{K}^0 = \{\sigma_0\}$. Thus, for $l > 0$, $\mathcal{K}^l$ is *not* a group since it does not contain the identity permutation, σ_0.

Proposition 3. *For any permutation α, not necessarily in* $\mathrm{Aut}(\mathcal{C})$*, the trivial subgroup $\mathcal{D}_{\alpha(H)} = \alpha \circ \mathcal{D}_H \circ \alpha^{-1}$, for a given H. Furthermore, $\mathcal{K}_{\alpha(H)} = \alpha \circ \mathcal{K}_H \circ \alpha^{-1}$ and $\mathcal{K}^l_{\alpha(H)} = \alpha \circ \mathcal{K}^l_H \circ \alpha^{-1}$, for all l, $0 \leq l \leq \min(n-k,\ k)$.*

Proof. Let $\sigma = \alpha \circ \theta \circ \alpha^{-1}$ where $\theta \in \mathcal{D}_H$. After applying α^{-1} to $\alpha(H)$, the original matrix H is reconstructed. Then, the effect of applying θ to H is to permute the rows of H. Finally, the columns are permuted according to α, and the resulting matrix will be row-equivalent to $\alpha(H)$. Thus, σ is trivial with respect to $\alpha(H)$, from which it follows that $\alpha \circ \mathcal{D}_H \circ \alpha^{-1}$ is a subset of $\mathcal{D}_{\alpha(H)}$. To prove equality, we use this result with $H' = \alpha(H)$, from which it follows that $\kappa \circ \mathcal{D}_{H'} \circ \kappa^{-1} \subseteq \mathcal{D}_{\kappa(H')}$, where κ is any permutation. Choosing $\kappa = \alpha^{-1}$, we get $\alpha^{-1} \circ \mathcal{D}_{\alpha(H)} \circ \alpha \subseteq \mathcal{D}_H$, from which it follows that $\mathcal{D}_{\alpha(H)} \subseteq \alpha \circ \mathcal{D}_H \circ \alpha^{-1}$. Since $\mathcal{D}_{\alpha(H)}$ is both a subset and a super-set of $\alpha \circ \mathcal{D}_H \circ \alpha^{-1}$, we have equality.

To prove the second part, i.e., to show that $\mathcal{K}_{\alpha(H)} = \alpha \circ \mathcal{K}_H \circ \alpha^{-1}$, we use the fact that for any two permutations $\sigma_1 = \theta_1 \circ \sigma \in \mathcal{D}_{\alpha(H)} \circ \sigma$ and

[3] The iso-orbit of H, containing Tanner graphs, should not be confused with the orbit of $\mathcal{C}$, which contains simple graphs.



$\sigma_2 = \theta_2 \circ \sigma \in \mathcal{D}_{\alpha(H)} \circ \sigma$ from the same coset (based on $\mathcal{D}_{\alpha(H)}$), where σ denotes the coset leader and $\theta_1, \theta_2 \in \mathcal{D}_{\alpha(H)}$,

$$\sigma_1 \circ \sigma_2^{-1} = \theta_1 \circ \sigma \circ \sigma^{-1} \circ \theta_2^{-1} = \theta_1 \circ \theta_2^{-1} \in \mathcal{D}_{\alpha(H)}.$$

Thus, if for any two permutations σ_1 and σ_2 from a given set, the composition $\sigma_1 \circ \sigma_2^{-1} \notin \mathcal{D}_{\alpha(H)}$, then σ_1 and σ_2 belong to two different cosets (based on $\mathcal{D}_{\alpha(H)}$). Now, let $\sigma_1 = \alpha \circ \kappa_1 \circ \alpha^{-1}$ and $\sigma_2 = \alpha \circ \kappa_2 \circ \alpha^{-1}$, where $\kappa_1, \kappa_2 \in \mathcal{K}_H$, from which it follows that

$$\sigma_1 \circ \sigma_2^{-1} = \alpha \circ \kappa_1 \circ \alpha^{-1} \circ \alpha \circ \kappa_2^{-1} \circ \alpha^{-1} = \alpha \circ (\kappa_1 \circ \kappa_2^{-1}) \circ \alpha^{-1}.$$

Since $\kappa_1 \circ \kappa_2^{-1} \notin \mathcal{D}_H$ (κ_1 and κ_2 are from different cosets based on $\mathcal{D}_H$), $\sigma_1 \circ \sigma_2^{-1} \notin \mathcal{D}_{\alpha(H)}$, and it follows that σ_1 and σ_2 are from two different cosets based on $\mathcal{D}_{\alpha(H)}$, and the result follows since $|\mathcal{K}_H| = |\mathrm{Aut}(\mathcal{C})|/|\mathcal{D}_H| = |\mathrm{Aut}(\mathcal{C})|/|\mathcal{D}_{\alpha(H)}| = |\mathcal{K}_{\alpha(H)}|$.

To prove the third part, i.e., to show that $\mathcal{K}^l_{\alpha(H)} = \alpha \circ \mathcal{K}^l_H \circ \alpha^{-1}$ for all l, we use the fact that the *depth* of σ, i.e., the length of the corresponding ELC sequence, based on H, is the same as the depth of $\alpha \circ \sigma \circ \alpha^{-1}$ based on $\alpha(H)$ for any σ in $\mathrm{Aut}(\mathcal{C})$. To show this, we write the depth of $\alpha \circ \sigma \circ \alpha^{-1}$ based on $\alpha(H)$ as

$$\begin{aligned} &|\{\alpha \circ \sigma \circ \alpha^{-1}(\mathcal{P}_{\alpha(H)}) \cap \mathcal{I}_{\alpha(H)}\}| \\ &\qquad = |\{\alpha \circ \sigma \circ \alpha^{-1}(\alpha(\mathcal{P}_H)) \cap \alpha(\mathcal{I}_H)\}| \\ &\qquad = |\{\alpha \circ \sigma(\mathcal{P}_H) \cap \alpha(\mathcal{I}_H)\}| \\ &\qquad = |\{\alpha(\sigma(\mathcal{P}_H) \cap \mathcal{I}_H)\}| \\ &\qquad = |\{\sigma(\mathcal{P}_H) \cap \mathcal{I}_H\}|. \end{aligned}$$

Now, we can conclude that the depth of all coset leaders in $\mathcal{K}^l_{\alpha(H)}$ (based on $\mathcal{D}_{\alpha(H)}$) is the same and equal to the depth of the coset leaders from $\mathcal{K}^l_H$ (based on $\mathcal{D}_H$), from which the result follows. □

As discussed above, $\mathcal{D}$ depends on H, so the iso-orbit is not a code property. The partitioning of permutations in $\mathcal{K}_H$ into disjoint subsets according to the length of the corresponding iso-ELC sequence may vary for each $H' = \sigma(H),\ \sigma \in \mathrm{Aut}(\mathcal{C})$. Still, from Proposition 3, $|\mathcal{K}^l_H| = |\mathcal{K}^l_{\sigma(H)}|,\ 0 \le l \le \min(n-k,\ k)$ and $\sigma \in \mathrm{Aut}(\mathcal{C})$, and we call the set $\{|\mathcal{K}^l_H|\},\ 0 \le l \le \min(n-k,\ k)$, the *profile* of the iso-orbit of H. This profile varies with H, but is invariant over the iso-orbit of H.

Table 1: *Pairs of permutations from* $\mathrm{Aut}(\mathcal{C})$ *which generate* $\mathcal{K}$ *for the* $[8, 4, 4]$ *extended Hamming code, as represented by* (7)*. These 8 groups are all isomorphic to one group, which is unique.*

$\langle(0,4,2,7,6,3,1),\ (0,6,7,4,5,2,3)\rangle$	$\langle(0,1,3,6,5,7,2),\ (0,6,1,7,4,5,2)\rangle$
$\langle(0,6,4,5,1,2,3),\ (0,7,5,2,1,4,3)\rangle$	$\langle(0,6,7,4,2,3,1),\ (0,4,5,2,7,6,3)\rangle$
$\langle(0,2,1,6,4,5,3),\ (0,6,7,5,4,2,1)\rangle$	$\langle(0,6,2,1,5,7,3),\ (0,7,5,3,4,2,1)\rangle$
$\langle(0,5,7,2,4,3,1),\ (0,2,6,4,7,5,3)\rangle$	$\langle(0,4,5,1,2,7,3),\ (0,6,7,5,2,1,3)\rangle$

Since the profile varies with $H \in \mathrm{orbit}(\mathcal{C})$, it may be desirable to search the orbit for a graph that has certain properties with respect to the profile. We will illustrate this with some examples.

Example 3. *For the* $[8, 4, 4]$ *extended Hamming code, which is ELC-preserved, the parity-check matrix*

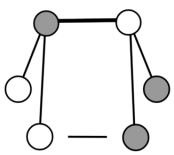
$$H = \begin{bmatrix} 1&0&0&0&1&1&0&1 \\ 0&1&0&0&0&1&1&1 \\ 0&0&1&0&1&1&1&0 \\ 0&0&0&1&1&0&1&1 \end{bmatrix} \tag{7}$$

has the profile listed in Table 2. For this code, there exists only one conjugacy class of subgroups of $\mathrm{Aut}(\mathcal{C})$ *of size* $|\mathcal{K}| = |\mathrm{Aut}(\mathcal{C})|/|\mathcal{D}| = 1344/24 = 56$*. $\mathcal{K}$ can be any of the eight distinct (but isomorphic) subgroups in this class. The eight subgroups may all be generated by two permutations, as listed in Table 1. This shows that $\mathcal{K}$ can be a group, and the minimum number of generators is 2 ($\mathcal{K}$ can not be a cyclic subgroup). Generators for $\mathcal{D}$ are* $\langle(0,2)(6,7),\ (1,3)(4,5),\ (2,3)(5,7)\rangle$*.*

Example 4. *The* $[15, 5, 7]$ *BCH code is a rare example of a code with only two graphs in the orbit. The corresponding systematic parity-check matrices, of weight 42 and 40, are*

$$H_0 = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&0&1&0&1&0&1 \\ 0&1&0&0&0&0&0&0&0&0&0&1&1&1&1&1 \\ 0&0&1&0&0&0&0&0&0&0&0&1&1&0&1&0 \\ 0&0&0&1&0&0&0&0&0&0&0&1&1&0&1&1 \\ 0&0&0&0&1&0&0&0&0&0&0&1&0&0&1&1 \\ 0&0&0&0&0&1&0&0&0&0&0&1&1&1&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&1&1&1&1&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0&1&1&1&1 \\ 0&0&0&0&0&0&0&0&1&0&1&0&1&1&1&0 \\ 0&0&0&0&0&0&0&0&0&1&0&1&0&1&1&1 \end{bmatrix} \text{ and } H_1 = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&0&1&0&1&0&1 \\ 0&1&0&0&0&0&1&0&0&0&1&0&0&0&0&1 \\ 0&0&1&0&0&0&1&0&0&0&1&0&1&0&0&0 \\ 0&0&0&1&0&0&1&0&0&0&0&0&0&1&1&1 \\ 0&0&0&0&1&0&0&0&0&0&1&0&0&1&1&1 \\ 0&0&0&0&0&1&1&0&0&0&1&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1&0&0&1&0&0&1&0&1 \\ 0&0&0&0&0&0&0&0&1&0&0&0&0&1&1&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&1 \\ 0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&0 \end{bmatrix}$$

respectively. The corresponding trivial subgroups of $\mathrm{Aut}(\mathcal{C})$ *are of size* $|\mathcal{D}| = 12$ *and* $|\mathcal{D}| = 120$*, where* $|\mathrm{Aut}(\mathcal{C})| = 20\,160$*. For the first structure, there*



are no subgroups of $\mathrm{Aut}(\mathcal{C})$ *of size* $|\mathcal{K}| = |\mathrm{Aut}(\mathcal{C})|/|\mathcal{D}| = 1\,680$. *For the second structure there are 5 conjugacy classes of subgroups of size* $|\mathcal{K}| = 168$ *(two of which are nonisomorphic), but none of these represent* $\mathcal{K}$ *(they all contain trivial permutations). Thus, for this code,* $\mathcal{K}$ *can not be a group. The two profiles for* $\mathcal{K}$ *are listed in Table 2.*

Example 5. *The* $[24, 12, 8]$ *extended Golay code, where* $|\mathrm{Aut}(\mathcal{C})| = 244\,823\,040$ *is another rare code with only two structures in the orbit*

$$H_0 = \begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&1&1&0&0&0&1&1\\
0&1&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&0&0&1&0&0&1&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&1&1&0&1&0&0&1&0&1&0&1&1\\
0&0&0&1&0&0&0&0&0&0&0&0&0&1&1&0&0&0&1&1&1&0&1&1&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&1&1&0&0&1&1&0&1&1&0&0&1\\
0&0&0&0&0&1&0&0&0&0&0&0&0&1&1&0&0&1&1&0&1&1&0&1&1\\
0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&0&0&1&1&0&1&1&1&1&1\\
0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&1&1&0&1&1&1&1&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&1&1&0&1&1&1&1&1&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&1&1&0&1&1&1&1&1&1&0\\
0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&1&1&0&0&0&1&1&0&1&1\\
0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&1&1&0&0&0&1&1&1&1
\end{bmatrix}$$

$$H_1 = \begin{bmatrix}
1&0&1&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&0&0&1&0&0&0&0\\
1&0&0&1&0&0&0&0&0&0&0&0&0&1&1&0&1&0&0&1&0&1&0&1&1\\
1&0&0&0&0&1&0&0&0&0&0&0&0&1&1&0&0&1&0&0&0&1&1&1&0\\
1&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&1&1&1&0&1&0&1&0&0\\
1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&1&0&0&1&1&0&1&1\\
1&0&0&0&0&0&0&0&1&0&0&0&0&1&1&1&1&0&1&0&1&1&1&1&1\\
1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&1&1&0&0&0&1&1\\
0&1&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&0&0&1&0&0&1&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&1&1&0&0&1&1&0&1&1&0&0&1\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&1&1&0&1&1&1&1&1&0\\
0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&1&1&0&0&0&1&1&0&1&1\\
0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&1&1&0&0&0&1&1&1&1
\end{bmatrix}.$$

These are of weight 96 and 100, and $|\mathcal{D}| = 240$ *and* 660, *respectively. The two profiles for* $\mathcal{K}$ *are listed in Table 2. As for the first structure of the BCH code in Example 4, no subgroups of* $\mathrm{Aut}(\mathcal{C})$ *exist of size* $|\mathcal{K}|$ *for neither of the two structures for the extended Golay code. Thus,* $\mathcal{K}$ *can not be a group.*

Example 6. *The* $[31, 21, 5]$ *BCH code has* $118\,208$ *graphs in its orbit. Among these there is a total of 8 structures for which* $|\mathcal{D}| > 1$, *such that* $\mathcal{K}$ *may be a true subgroup (we disregard the case when* $\mathcal{K} = \mathrm{Aut}(\mathcal{C})$*). One of these is*

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

These are all of weight 120 (with one exception, at weight 140) and all have $|\mathcal{D}| = 5$, *and* $\mathcal{K}$ *is indeed a group, with the profile and generator (the group*

Table 2: *Profiles of $\mathcal{K}$ as split into subsets according to the length of the corresponding ELC sequence. "E.H." and "E.G." are the extended Hamming and Golay codes.*

Code	$\lvert H\rvert$	0	1	2	3	4	5	6	7	8	9	10	11	12
E.H.	16	1	12	30	12	1	-	-	-	-	-	-	-	-
"BCH15"	42	1	29	246	678	585	141	-	-	-	-	-	-	-
"	40	1	0	30	60	65	12	-	-	-	-	-	-	-
E.G.	100	1	22	616	6 490	33 935	85 712	117 392	85 712	33 935	6 490	616	22	1
"	96	1	60	1 650	18 140	92 655	236 520	322 044	236 520	92 655	18 140	1 650	60	1
"BCH31"	120	1	0	0	0	0	0	10	10	10	-	-	-	-
"	140	1	0	0	0	0	0	10	10	10	-	-	-	-

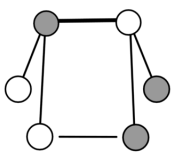
" $\mathcal{K} = \langle (0,7,14,21,28,4,11,18,25,1,8,15,22,29,5,12,19,26,2,9,16,23,30,6,13,20,27,3,10,17,24) \rangle$

is cyclic) listed in Table 2. This group is unique as there is only one subgroup (in the single conjugacy class) of $\mathrm{Aut}(\mathcal{C})$ *of the required size,* $|\mathcal{K}| = 31$*. The size of* $\mathrm{Aut}(\mathcal{C})$ *is* 155*.*

Example 7. *For the* [48, 24, 12] *extended QR (EQR) code, the possible sizes of* $\mathcal{D}$ *are* $\{1,2,3,4,6,8,12,23,24\}$*. Only for* $|\mathcal{D}| = 1$ *(the trivial case) can* $\mathcal{K}$ *be a group. In the orbit of the code, we may determine the number of distinct structures corresponding to the different values of* $|\mathcal{D}|$ *to be* $\{-, 43\,838, 128, 120, 13, 5, 2, 1, 1\}$*. We omit counting for* $|\mathcal{D}| = 1$*, which would entail enumerating the entire orbit of the code.*

Example 8. *For the* [63, 51, 5] *BCH code and the* [63, 39, 9] *BCH code (used in [13]), the possible sizes of* $\mathcal{D}$ *are* $\{1,2,3,6\}$*. For the first code, the corresponding number of structures is* $\{-, 7\,398, 222, 14\}$*, and for the second code, we find* $\{-, > 500\,000, 3\,675, 38\}$*. For all these structures (and both codes),* $\mathcal{K}$ *can be a group.*

As ELC complements edges in the local subgraph, i.e., at distance 1 from the ELC edge, we may alternatively say that ELC has the effect of complementing *4-cycles* (cycles of length 4) in the graph. This perspective leads to some observations on the relationship between iso-ELC and the girth of the graph. Specific requirements must be satisfied for a graph operation to be an isomorphism; most importantly, that the number of edges in the graph remains invariant. In other words, for



any **e** to be an isomorphism, the most basic requirement is that the number of edges *inserted* is matched by (equals) the number of edges *removed.* For this to be possible using a single ELC operation, the girth (length of the smallest cycle in $\mathcal{G}$) must be 4.

Example 9. *For the* $[8, 4, 4]$ *extended Hamming code, again, we have that* $|\mathcal{K}^1| = |\mathcal{G}| = 12$*, which is to say that the code is ELC-preserved. As such, the coset leaders in* $\mathcal{K}$ *form telescoping sequences. Given* H *in standard form, the single iso-ELC sequence in* $\mathcal{K}^4$ *contains shorter iso-ELC sequences as subsets, as follows*

$$\{\overbrace{\overbrace{\overbrace{(0,6)}^{\in \mathcal{K}^1}, (1,7)}^{\in \mathcal{K}^2}, (2,4)}^{\in \mathcal{K}^3}, (3,5)\} = \mathcal{K}^4.$$

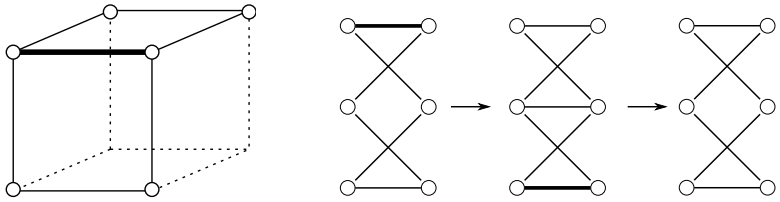
This is due to the perfect symmetry of $\mathcal{G}$*, in which all local neighborhoods are identical, forming a cube – see Fig. 3(a).*

ELC on any edge of an ELC-preserved graph (corresponding to an ELC-preserved code) is an iso-ELC operation, yet, according to the definition of ELC, a pair of columns are indeed swapped between $\mathcal{I}$ and $\mathcal{P}$. This proves that $\mathrm{Aut}(\mathcal{C})$ for an ELC-preserved code can not be trivial. On the other hand, for a large random code we expect that $\mathrm{Aut}(\mathcal{C})$ is trivial, but that the orbit is vast (e.g. for LDPC codes).

Example 10. *For the "bow-tie" graph, consisting of a single 6-cycle, an isomorphism is found in the application of ELC to any of the three pairs of diametrically opposite edges, inserting and removing a chord (i.e., a 4-cycle), as shown in Fig. 3(b). As expected, the iso-orbit of the corresponding* $[6, 3]$ *code gives* $|\mathcal{K}^0| = 1$ *and* $|\mathcal{K}^2| = 3$*.*

3 Weight-Bounding ELC

In the discussion on isomorphic ELC operations, a requirement is that the number of edges in the graph must be preserved [16]. We will generalize this, and introduce a notion of *weight-bounding* ELC (WB-ELC) operations, in which the weight of H after ELC, denoted by $|H'|$, is upper-bounded by some threshold, T, such that $|H'| \leq |H| + T$. So, the depth-i iso-ELC sequences described previously are depth-i WB-ELC for $T = 0$. In this work, we will restrict our focus to single and double



(a) The identical local neighborhoods of the "Hamming cube."

(b) The "bow-tie" graph, consisting of one 6-cycle.

Fig. 3: *Relationships between an isomorphism and the girth of the underlying graph.*

ELC operations (depth-1 or 2), in order to facilitate the locality argument of the ELC operation. However, the concept of WB-ELC extends to an arbitrary length sequence of ELC operations. In this section, we only allow an ELC sequence on a certain edge in the graph if $|H| \leq |H_0| + T$, where H_0 is the initial graph and T is the weight-bounding threshold. We give necessary and sufficient conditions to achieve this bound, both for single ELC and for two consecutive ELCs.

Let $A \sim B$ be a shorthand notation for the edges in the subgraph $\mathcal{E}_{A,B}$, i.e., those connecting nodes in A to nodes in B. Also, $\mathcal{E}^{C}_{A,B}$ denotes the subgraph after complementing $A \sim B$. The net difference in edges before and after complementation is $\Delta\mathcal{E}_{A,B} \triangleq |\mathcal{E}^{C}_{A,B}| - |\mathcal{E}_{A,B}|$.

Lemma 1. *The number of edges complemented between sets A and B can be expressed as $\Delta\mathcal{E}_{A,B} \triangleq |\mathcal{E}^{C}_{A,B}| - |\mathcal{E}_{A,B}| = |A||B| - 2|\mathcal{E}_{A,B}|$.*

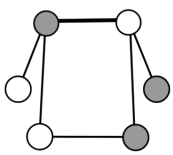
Proof. The complete bipartite graph between A and B has $|A||B|$ edges. This means that, for any graph between A and B, $|\mathcal{E}_{A,B}| + |\mathcal{E}^{C}_{A,B}| = |A||B|$, so $\Delta\mathcal{E}_{A,B} = |\mathcal{E}^{C}_{A,B}| - |\mathcal{E}_{A,B}| = |A||B| - |\mathcal{E}_{A,B}| - |\mathcal{E}_{A,B}|$. □

3.1 Depth-1, Single Edge WB-ELC

If the weight change due to the action of a single ELC is upper-bounded, then the weight of the entire graph is upper-bounded, and since ELC is a local operation, we say that the ELC is WB-ELC.

Theorem 3. *The weight change of $\mathcal{G}$ under ELC on (u, v) is upper-bounded by a threshold T iff*

$$\Delta\mathcal{E}_{u,v} = |\mathcal{N}_u^v||\mathcal{N}_v^u| - 2|\mathcal{E}_{u,v}| \leq T. \tag{8}$$



Proof. ELC on (u, v) complements the edges between $\mathcal{N}_u^v$ and $\mathcal{N}_v^u$, where (8) follows from Lemma 1. The weight change of $\mathcal{G}$ under ELC on (u, v) is therefore $\Delta\mathcal{E}_{u,v}$. □

3.2 Depth-2, Double Edge WB-ELC

For many graphs, it is difficult (or even impossible) to bound the weight change by any reasonable threshold (i.e., small T), using only a single ELC. We now determine the WB-ELC operations which exist for double application of ELC on a graph. Given a graph, $\mathcal{G}$, and a theshold, T, the definition of a depth-2 WB-ELC operation is an ordered sequence of two ELC operations, where the first ELC operation must change the weight of $\mathcal{G}$ by more than T to a graph $\mathcal{G}^\star$, whereas the second ELC must compensate, by reducing the weight of $\mathcal{G}^\star$ by at least $|\mathcal{G}^\star| - |\mathcal{G}| - T$. (Note that this amount is always positive, as $|\mathcal{G}^\star| > T + |\mathcal{G}|$; otherwise the first ELC would change the weight by less or equal than T.) This follows from the fact that, if the first ELC did *not* exceed the weight-bounding threshold, then it would, in itself, be a depth-1 WB-ELC operation.

A very important first observation is that the search space for depth-2 WB-ELC can be significantly reduced from that of checking all pairs of edges in $\mathcal{G}$. First, ELC on two adjacent edges, i.e., at distance 0, reduces to a single ELC operation.

Lemma 2 (Adjacent edges [27])**.** *ELC on* $\{(u, v), (v, v')\}$*, where* $v' \in \mathcal{N}_u^v$*, gives the same graph as ELC on* (u, v')*.*

Proof. See Fig. 4. The full proof is found in Appendix A □

From Lemma 2, we see that ELC on adjacent edges reduces to a single ELC, which has already been covered by the discussion of depth-1 WB-ELC. So, in order to find additional WB-ELC instances at depth-2, we need not consider adjacent pairs of edges. We now present an important result regarding depth-2 WB-ELC; that the distance between a pair of edges can not be greater than two, for $T \geq$ -1.[4]

Lemma 3 (Disjoint edges)**.** *Let* $T \geq$ -1*. Any depth-2 WB-ELC where the pair of edges are at a distance greater than two will always reduce to either one instance, or two separate instances, of depth-1 WB-ELC.*

[4] A special case exists for $T <$ -1, which is accounted for in Proposition 4.

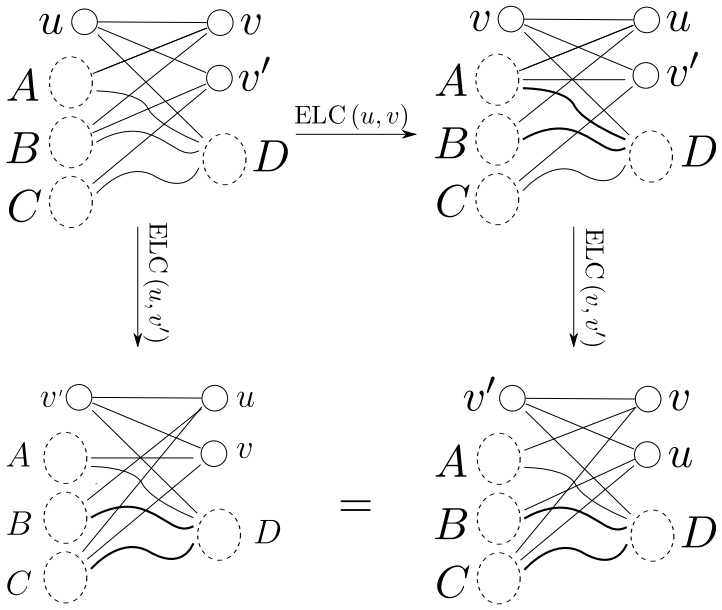
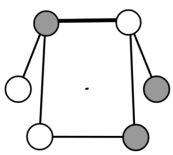
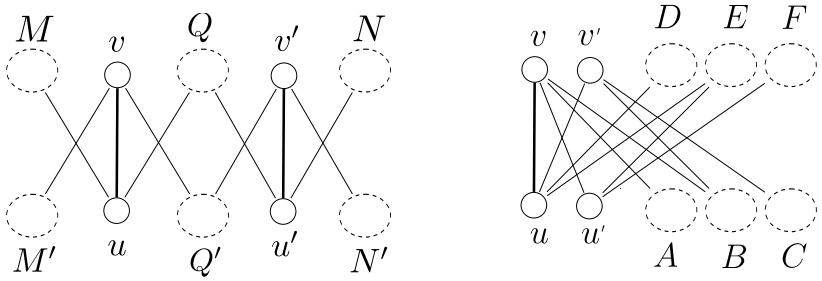


Fig. 4: *Proof of Lemma 2. ELC on adjacent edges, e.g.* (u, v) *and* (v, v')*, will always result in the exact same graph as a single ELC on the second edge,* (u, v')*. Note that, due to the swap of nodes u and v in the first ELC, the second ELC edge is labeled* (v, v') *– however, this is the same edge as* (u, v') *in the initial graph.*

Proof. Consider two disjoint subgraphs, $\mathcal{E}_{u,v}$ and $\mathcal{E}_{u',v'}$, of the same graph. In this case, ELC on $\{(u, v), (u', v')\}$ gives the same graph as ELC on $\{(u', v'), (u, v)\}$, since the neighborhoods do not interact. Consider first the case where $T \geq 0$. The only possibilities for WB-ELC are: Both ELC operations preserve weight, in which case they both classify as depth-1 WB-ELC operations, or one ELC operation increases weight by $\Delta\mathcal{E}_{u,v} > T$, while the other ELC reduces the weight by at least $\Delta\mathcal{E}_{u,v} - T$. Since they commute, we can assume without loss of generality that ELC on (u, v) is the operation which reduces the weight, but then this classifies as a depth-1 WB-ELC.

For the case where $T =$ -1, again we have two possibilities; that both ELC operations are depth-1 WB-ELC, or that one of them does not decrease the weight, while the other reduces the weight. Again, and since they commute, we can assume without loss of generality that





(a) Distance 2 (Theorem 5) (b) Distance 1 (Theorem 6)

Fig. 5: *Depth-2 WB-ELC. Potential connections between sets in the (bipartite) subgraphs are not shown.*

ELC on (u, v) is the operation which reduces the weight, but then this classifies as a depth-1 WB-ELC. □

Theorem 4 (Reduced search space)**.** *Let $T \geq$ -1. All depth-2 WB-ELC can be found by considering pairs of edges at distance one or two.*

Proof. The proof follows from Lemmas 2 and 3. □

In this sense, we define WB-ELC (both depth-1 and depth-2) as a *local* graph operation, in that its effect is confined to a subgraph of diameter at most 4. The corresponding subgraphs are shown in Fig. 5. We have restricted the search space considerably, and shall now cover all the possible cases for depth-2 WB-ELC, for $T \geq$ -1.

Let us first consider the case where the pair of edges are at a distance of exactly two edges apart, Fig. 5(a). Given an edge (u, v), let $u', v' \notin \mathcal{N}_u \cup \mathcal{N}_v$ be such that $(u', v') \in \mathcal{G}$, $Q = \mathcal{N}_u^v \cap \mathcal{N}_{u'}^{v'} \neq \emptyset$, and, similarily, $Q' = \mathcal{N}_{v'}^{u'} \cap \mathcal{N}_v^u \neq \emptyset$.

Theorem 5 (Distance 2)**.** *The weight change of $\mathcal{G}$ under ELC on $\{(u, v), (u', v')\}$ is upper-bounded by a threshold T iff*

$$\Delta\mathcal{E}_{u,v} + \Delta\mathcal{E}_{u',v'} - 2\Delta\mathcal{E}_{Q',Q} \leq T. \tag{9}$$

This case covers all instances of depth-2 WB-ELC where the edges are at a distance two apart.

Proof. The edges (u, v) and (u', v') comprise a special case of independence since Q and Q' are adjacent to both edges – see Fig. 5(a). As such,

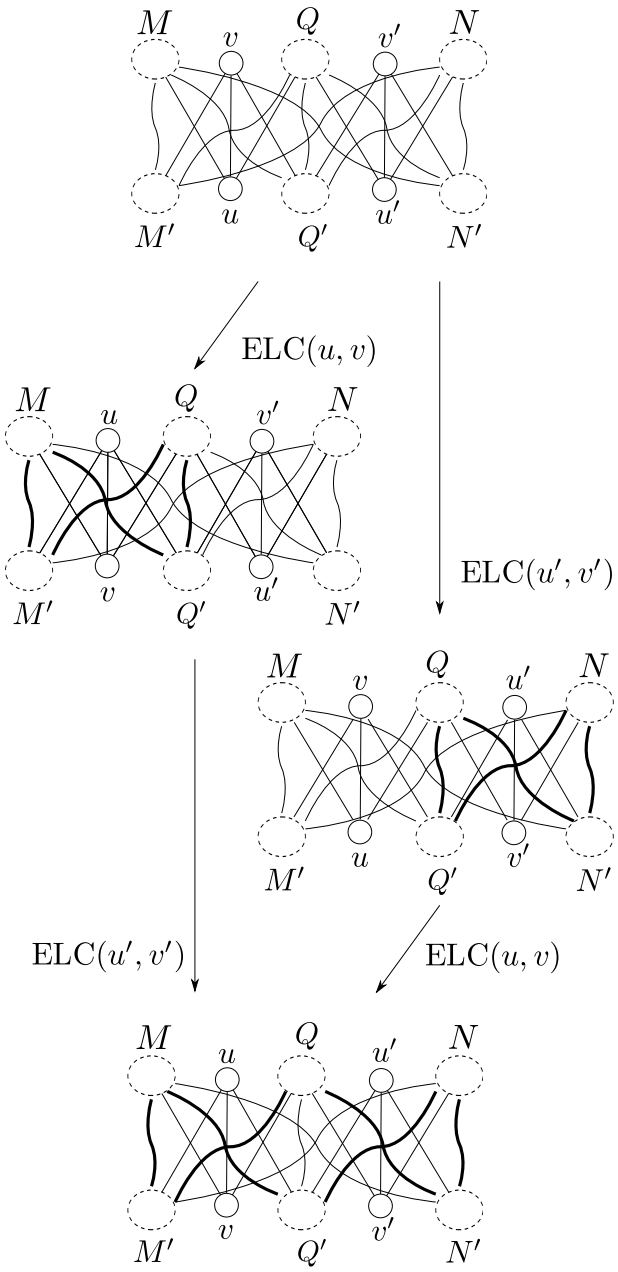
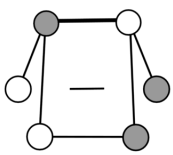


Fig. 6: *Proof of Theorem 5. A special case of independence gives the equivalent sequence, $(u', v'), (u, v)$; although the local subgraphs $\mathcal{E}_{u,v}$ and $\mathcal{E}_{u',v'}$ are not independent, the overlap is confined to Q and Q′ (which is complemented twice) [26].*



the sequence $\{(u',v'),(u,v)\}$ gives the same graph as the sequence $\{(u,v),(u',v')\}$ [26]. In addition to the case where Q and Q' are both nonempty, there are two other cases at distance 2. We consider first the case where either Q or Q' is empty (note that if both are empty the distance is greater than 2). In this case, the two ELC operations are independent and this case reduces to Theorem 3 (one or two instances of depth-1 WB-ELC). The second of these cases is where $(u',v') \notin \mathcal{G}$. As u' and v' are not in the common neighbourhood of (u,v), we cannot obtain this edge by ELC on (u,v), and thus the second ELC (on (u',v')) is not possible. We have now seen that, for depth 2 at distance 2, we only need to consider the sequence $\{(u,v),(u',v')\}$ where Q and Q' are both nonempty. Since the distance is not greater than two, this case is not covered by Theorem 3. The net effect of ELC on $\{(u,v),(u',v')\}$ is $\Delta\mathcal{E}_{M',M} + \Delta\mathcal{E}_{Q',M} + \Delta\mathcal{E}_{Q',N} + \Delta\mathcal{E}_{N',Q} + \Delta\mathcal{E}_{M',Q} + \Delta\mathcal{E}_{N',N}$. Fig. 6 illustrates that we have $\Delta\mathcal{E}_{u,v} = \Delta\mathcal{E}_{M',M} + \Delta\mathcal{E}_{M',Q} + \Delta\mathcal{E}_{M,Q'} + \Delta\mathcal{E}_{Q',Q}$, and $\Delta\mathcal{E}_{u',v'} = \Delta\mathcal{E}_{N',N} + \Delta\mathcal{E}_{N',Q} + \Delta\mathcal{E}_{N,Q'} + \Delta\mathcal{E}_{Q',Q}$. This is the same as the result of ELC on (u,v) summed to the result of ELC on (u',v') independently, except for the double complementation of $\mathcal{E}_{Q',Q}$, which gives the desired formula. □

Fig. 3(b) shows an example, where the weight-bounding is implicit from the isomorphism.

We now consider distance one. Given an edge (u,v) and two nodes u' and v', we denote by $B = \mathcal{N}_v^{u,u'} \cap \mathcal{N}_{v'}^{u,u'}$, $A = \mathcal{N}_v^{u,u'} \setminus B$, $C = \mathcal{N}_{v'}^{u,u'} \setminus B$, $E = \mathcal{N}_u^{v,v'} \cap \mathcal{N}_{u'}^{v,v'}$, $D = \mathcal{N}_u^{v,v'} \setminus E$, and $F = \mathcal{N}_{u'}^{v,v'} \setminus E$, see Fig. 5(b). We now consider the case where both u' and v' are in the neighborhood of (u,v), and where $(u',v') \notin \mathcal{G}$ is created by the first ELC.

Theorem 6 (Distance 1)**.** *The weight change of $\mathcal{G}$ under ELC on $\{(u,v),(u',v')\}$ is upper-bounded by a threshold T iff*

$$\Delta\mathcal{E}_{A,E\cup F} + \Delta\mathcal{E}_{B,D\cup E} + \Delta\mathcal{E}_{C,D\cup F} + |C| + |F| - |B| - |E| \leq T. \qquad (10)$$

This case covers all instances of depth-2 WB-ELC where the edges are at distance one apart.

Proof. As in the case for distance 2, we will begin by defining the search space. There are four possible choices of two edges at a distance of one apart; these are shown in Fig. 7, where the upper edge is referred to as the *first* edge (for ELC). The case in Fig. 7(d) is a degenerate case, in that the second egde is removed by the first ELC, so this case is not

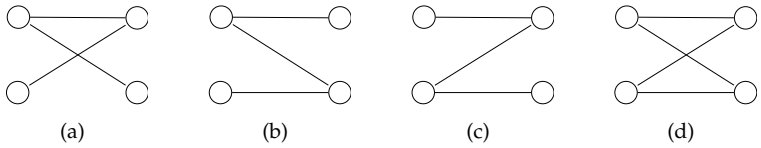


Fig. 7: *Figs. 7(a) - 7(c) represent special cases of ELC equivalences [26], where the first ELC is on the upper edge and the second ELC is on the bottom edge. In Fig. 7(a), this edge results from the first ELC. In the same way, Fig. 7(d) shows where double ELC is not possible (the second edge is removed by the first ELC).*

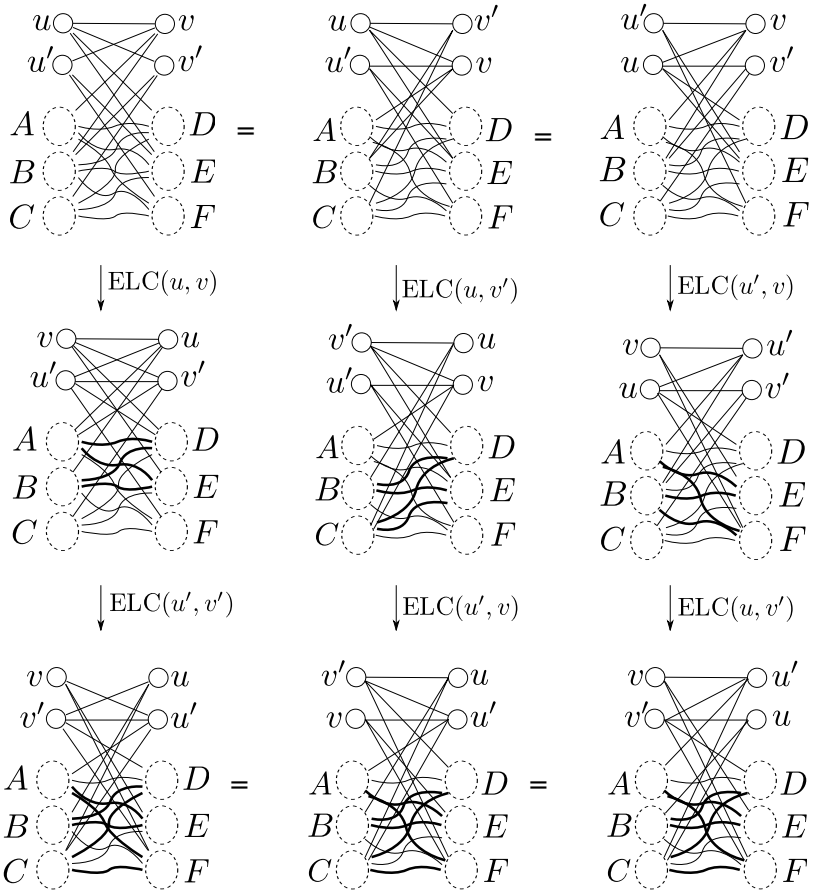
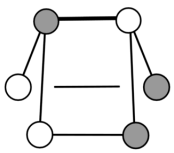


Fig. 8: *Proof of Theorem 6.*



possible. We show in Fig. 8 that the three possible cases of double ELC within distance 1 reduce to the case in Fig. 7(a). Note that, in this case, the second edge results from the first ELC.

Consider the case in Fig. 7(a). We denote the first edge (u, v) and the second (u', v'), giving the sequence $\{(u, v), (u', v')\}$. This illustrates therefore the case where u' and v' belong to $\mathcal{N}_u^v \cup \mathcal{N}_v^u$, and where $(u', v') \notin \mathcal{G}$. This case is shown in the left column of Fig. 8.

Since ELC on $\{(u, v), (u', v')\}$ gives the same graph as ELC on $\{(u', v), (u, v')\}$ [26], this covers the case illustrated in Fig. 7(b). This case is shown in the middle column of Fig. 8, and the equivalence is shown in the upper and lower rows.

In the same way, as ELC on $\{(u, v), (u', v')\}$ gives the same graph as ELC on $\{(u, v'), (u', v)\}$ [26], this covers the case illustrated in Fig. 7(c). This case is shown in the right column of Fig. 8, and the equivalence is shown in the upper and lower rows.

Thus, without loss of generality, we may focus on the case in the left column of Fig. 8. The effect of ELC on $\{(u, v), (u', v')\}$, as illustrated in the leftmost column of Fig. 8, is to complement (u, v), (u', v'), $A \sim E$, $A \sim F$, $B \sim D$, $B \sim E$, $C \sim D$, and $C \sim F$. In addition, node u (node v in the initial graph, before the swap) is now connected to C instead of A, v (i.e., u) is connected to F instead of D. In the same way, v' (i.e., u') is connected to D instead of E, and u' (v') to A instead of B. This all amounts to

$$\begin{aligned}\Delta\mathcal{E} = 1 - 1 + \Delta\mathcal{E}_{A,E\cup F} + \Delta\mathcal{E}_{B,D\cup E} + \Delta\mathcal{E}_{C,D\cup F} + \\ |C| - |A| + |F| - |D| + |D| - |E| + |A| - |B|, \end{aligned} \tag{11}$$

where $1 - 1$ is included to emphasize that the edge (u, v) is removed and the edge (u', v') is created, for a zero contribution to the weight difference. Note that all sets are pairwise disjoint. This gives the desired formula. □

We have shown that, for $T \geq$ -1, the depth-2 WB-ELC cases must occur on pairs of edges spaced by distance at most two. Let us now, for completeness, consider the case where $T <$ -1.

Proposition 4. *Let $T <$ -1. In this case a pair of edges at a distance of more than two may give depth-2 WB-ELC that does not reduce to (neither a single nor a double instance of) depth-1 WB-ELC.*

Proof. Let $T <$ -1. Consider again the disjoint edges, $\{(u, v), (u', v')\}$, i.e., at distance greater than two. Say ELC on (u, v) reduces the weight

by $\Delta\mathcal{E}_{u,v}$, while ELC on (u', v') gives a further reduction of $\Delta\mathcal{E}_{u',v'}$. Since we have $T <$ -1, it can happen that $\Delta\mathcal{E}_{u,v}$ and $\Delta\mathcal{E}_{u',v'}$ are both greater than T, while $\Delta\mathcal{E}_{u,v} + \Delta\mathcal{E}_{u',v'} \leq T$. In this case, the individual ELC operations would *not* be found as depth-1 WB-ELC, so this is a special case of depth-2 WB-ELC which can not be restricted to a local subgraph. □

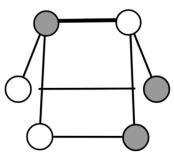
4 Complexity and Enumeration of WB-ELC

The main results discussed in this work are those on the generalization of isomorphic ELC operations to WB-ELC operations. Most importantly, the locality argument of ELC is maintained as the search space of (depth-1 and 2) WB-ELC is restricted to edges spaced by at most distance 2; that we need only consider single edges, and pairs of edges no more than two edges apart, in order to enumerate all WB-ELC operations for a given graph and threshold, $T \geq$ -1. We now discuss a selection of applications based on WB-ELC operations, based on an enumeration algorithm.

For this work, we consider various strong classical codes of practical blocklengths, which all qualify as HDPC codes. The $[15, 5, 7]$ BCH code and the $[24, 12, 8]$ EQR code (commonly referred to as the extended Golay code) serve a special purpose due to their extremely small orbit (see Examples 4 and 5). Correspondingly, as discussed for ELC-preserved codes, $\mathrm{Aut}(\mathcal{C})$ is large for these small codes; $|\mathrm{Aut}(\mathcal{C})|$ is 20 160 and 244 823 040, respectively. At a slightly larger blocklength, we have chosen two *extremal* (in terms of minimum distance) self-dual $[36, 18, 8]$ (called "R2") and $[38, 19, 8]$ ("$C_{38,2}$") codes from [28], and an extremal double circulant self-dual $[68, 34, 12]$ code ("$C_{68,1}$") from [29]. For these codes, $\mathrm{Aut}(\mathcal{C})$ is small ($|\mathrm{Aut}(\mathcal{C})| \approx n$); 32, 1 (trivial), and 68, respectively. We also consider the $[48, 24, 12]$ EQR code, as a next step from the extended Golay code, but for which the orbit size is large. Correspondingly, $\mathrm{Aut}(\mathcal{C})$ is small compared to the extended Golay code, containing "only" 51 888 permutations. These codes (except BCH15) are all even or doubly even; all codewords have Hamming weight divisible by 2 or 4, respectively.

4.1 Enumeration Algorithm

As Theorems 3 - 6 cover all possible single and double applications of ELC where the weight change is upper-bounded by some threshold



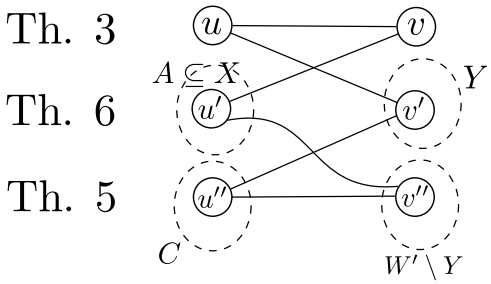


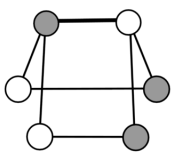
Fig. 9: *Illustration of Alg. 2, where* $X := \mathcal{N}_v^u$, $Y := \mathcal{N}_u^v$, *and* $W := \mathcal{N}_{u'}^v$. *Note that the edge* (u', v') *will result from the first ELC on* (u, v). *The curved line indicates the possibility of an edge* (u', v'').

$T \geq$ -1, we propose an enumeration algorithm to identify all (depth-1 and 2) WB-ELC operations on $\mathcal{G}$. The search space defined by Theorem 4 suggests an implementation. For each edge $(u, v) \in \mathcal{G}$, after checking Theorem 3 (depth-1), we want to check Theorem 6 on every $(u', v') \in \mathcal{E}_{u,v}$. Then, for each such choice of u' and v', we check Theorem 5 on every $(u'', v'') \in \mathcal{E}_{u',v'}$, see Fig. 9. The commutativity and isomorphisms discussed in Section 3 require additional measures to be taken in order to avoid duplicate WB-ELC sequences (giving the exact same Tanner graph). This corresponds to pruning of the search space, giving a complexity benefit. For Theorem 5, it suffices to ensure that the edges of the search space are considered in one direction only. For Theorem 6, it is possible to handle both cases (isomorphisms) by slight modifications to the sets of nodes from which the candidate edges are picked. Since $(u', v') \in \mathcal{E}_{u,v}$, we automatically avoid the two isomorphic cases described in the proof of Theorem 6. Furthermore, by restricting $u' \in A = \mathcal{N}_v^u \setminus \mathcal{N}_{v'}^u$ we avoid the degenerate case where $\{(u, v), (u', v')\}$ forms a 4-cycle since v' is not adjacent to A (by definition). Further details on a practical and efficient implementation are given in Appendix B

The most straight-forward implementation is to enumerate the search space, apply ELC to all candidate edges of $\mathcal{G}$ (as identified by Theorems 3 - 6), and check the weight of the resulting graphs, $\mathcal{G}'$. If the weight is upper-bounded by T, i.e., $|\mathcal{G}'| \leq |\mathcal{G}| + T$, then the corresponding ELC sequence is a (depth-1 or 2) WB-ELC operation. In order to enumerate all WB-ELC operations, such an implementation may apply ELC to the next candidate edge after undoing (i.e., repeating) the most

Algorithm 2 WB_ELC$_{(a)}(\mathcal{G}, T)$, to enumerate all WB-ELC sequences given $\mathcal{G}$ and threshold T.

$\kappa := 0$ // complexity counter (#edges)
for $\forall\, u \in \mathcal{U}, v \in \mathcal{V} : (u,v) \in \mathcal{G}$ **do**
  $X := \mathcal{N}_v^u$, $Y := \mathcal{N}_u^v$, $S := \emptyset$
  if $|X||Y| - 2|\mathcal{E}_{X,Y}| - T \leq 0$ **then**
    ELC(u,v) is T WB-ELC // Theorem 3
  end if
  κ++
  for $\forall\, v' \in Y$ **do**
    $Z := \mathcal{N}_{v'}^u$, $B := X \cap Z$, $A := X \setminus B$, $C := Z \setminus B$
    for $\forall\, u' \in A$ **do**
      $W := \mathcal{N}_{u'}^v$, $E := Y \cap W$, $D := Y \setminus E$, $F := W \setminus E$
      if $\Delta\mathcal{E}_{A,E\cup F} + \Delta\mathcal{E}_{B,D\cup E} + \Delta\mathcal{E}_{C,D\cup F} + 2|C| + 2|F| - T \leq 0$ **then**
        ELC$\{(u,v),(u',v')\}$ is T WB-ELC // Theorem 6
      end if
      κ++
    end for
    for $\forall\, u'' \in C \;:\; u'' > u$ **do**
      $W' := \mathcal{N}_{u''}$
      for $\forall\, v'' \in W' \setminus Y : (u'',v'') \notin S$ **do**
        $S := S \cup \{(u'',v'')\}$
        $X' := \mathcal{N}_{v''}^{u''}$, $Q := W' \cap Y$, $Q' := X \cap X'$
        if $Q, Q' \neq \emptyset$ **then**
          if $\Delta\mathcal{E}_{u,v} + \Delta\mathcal{E}_{X',W'} - 2\Delta\mathcal{E}_{Q,Q'} + |X'| - T \leq 0$ **then**
            ELC$\{(u,v),(u'',v'')\}$ is T WB-ELC // Theorem 5
          end if
        end if
        κ++
      end for
    end for
  end for
end for



recent ELC operation. For some purposes, however, it may be desirable to avoid unnecessary modifications of the graph, so an alternative implementation is to use the counting formulas (8) - (10) of the WB-ELC theorems to determine whether a set of (one or two) candidate edges is indeed a WB-ELC operation, without explicitly performing any ELC operations. The complexity of computing the sets and set operations required by the counting formulas is proportional to that of doing (and undoing) the corresponding ELC operations, so the preferred approach may be decided according to application requirements other than complexity.

For the following discussions, we assume the approach using counting formulas, with an implementation given in Alg. 2. This algorithm, referred to as WB_ELC$_{(a)}$, will serve as the framework for other algorithms discussed in this work. Let $L(\mathcal{G}, T)$ denote the set of WB-ELC sequences for $\mathcal{G}$, given T. As described, the search is rooted in the depth-1 candidate edge, (u, v), and works its way out to distance 1 and 2 (depth-2). As such, there will be a large overlap of the sets of nodes and edges required by the WB-ELC theorems. An intuitive approach is to reuse as many of these sets as possible by carefully nesting the theorems. To facilitate this, the counting formulas of Theorems 5 and 6 are modified as detailed in Appendix B Recall from (3) that $\mathcal{G} = (\mathcal{U} \cup \mathcal{V}, \mathcal{E})$ is an $(n-k,\ k)$-bipartite simple graph, with average column and row weights $\bar{\gamma}$ and $\bar{\rho}$, respectively.

Defining complexity, χ, in terms of the number of candidate edges checked to enumerate WB-ELC sequences, an analysis of the loop structure of WB_ELC$_{(a)}$ gives

$$\chi = |\mathcal{G}| \left(1 + |Y||A| + \min \left\{ |Y| \frac{|C|}{2} |W' \setminus Y|, |\mathcal{G}| - |\Omega| \right\} \right) \tag{12}$$

where we assume that, on average, half of the elements in C satisfy $u'' > u$ in line 17, and note that each edge of the graph can only satisfy $(u'', v'') \notin S$ in line 19 once, and further that edges with endpoints in $\mathcal{N}_u$ or $\mathcal{N}_v$, which we denote $\Omega = \{(a, b) \in \mathcal{G} \mid a \in \mathcal{N}_u \cup \mathcal{N}_v\}$, will never be considered as candidate edges (u'', v'') for checking Theorem 5 in line 24. Hence $|\mathcal{G}| - |\Omega|$, the number of edges at distance ≥ 2 from (u, v), is an upper bound on the number of times line 24 is executed. The minimization in (12) is necessary as the first argument, $|Y| \frac{|C|}{2} |W' \setminus Y|$, an upper bound on the number of times line 24 is executed derived from analysing the loop structure, will be higher than $|\mathcal{G}| - |\Omega|$ except for very small $\bar{\gamma}$.

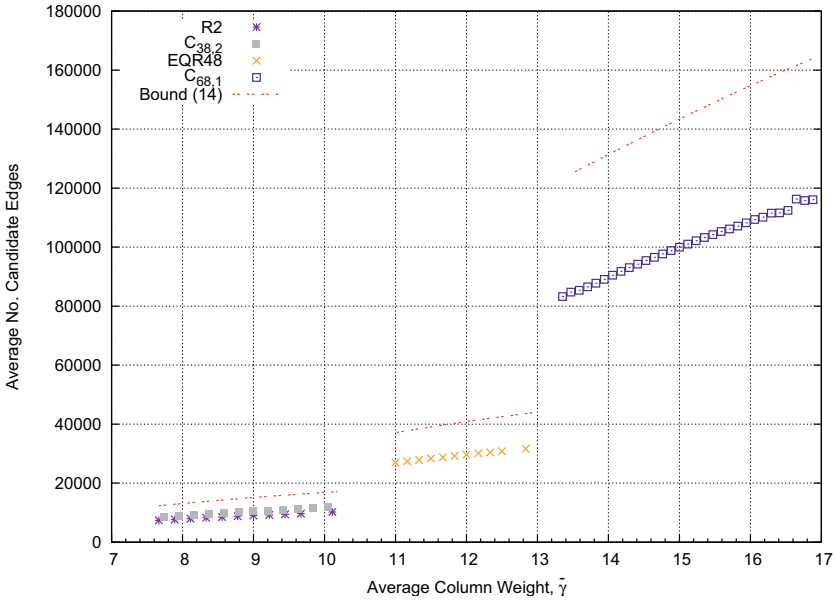


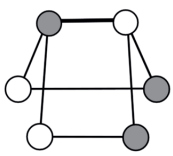
Fig. 10: *Complexity simulations for WB_ELC$_{(a)}$ compared against the bound of* (13) *(dashed lines). Each code is simulated over its entire weight range, from the minimum weight to the maximum weight encountered by ELC (which is around 50% weight), so the complexity of WB_ELC$_{(a)}$ is completely described.*

To obtain an estimate of the complexity of WB_ELC$_{(a)}$, we assume that $\mathcal{G} = (\mathcal{U} \cup \mathcal{V}, \mathcal{E})$ is a (k,k)-bipartite graph where every vertex has degree γ. Moreover, we assume that every pair of vertices from the same partition have λ common neighbors.

Proposition 5.

$$\lambda = \frac{\gamma(\gamma-1)}{k-1}.$$

Proof. Consider a vertex $u \in \mathcal{U}$. This vertex has γ neighbors, each of which have $\gamma - 1$ neighbors in $\mathcal{U} \setminus \{u\}$. Hence the number of edges between $\mathcal{N}_u$ and $\mathcal{U} \setminus \{u\}$ is $\gamma(\gamma-1)$. There are $k-1$ vertices in $\mathcal{U} \setminus \{u\}$ and each of these have λ neighbors in common with u. Hence the number of edges between $\mathcal{N}_u$ and $\mathcal{U} \setminus \{u\}$ is $(k-1)\lambda$. We then have that $\gamma(\gamma-1) = (k-1)\lambda$, and the result follows. □



Proposition 6.

$$|\Omega| = 2\gamma^2 - \frac{\gamma^3}{k}.$$

Proof. Each of the 2γ vertices in $\mathcal{N}_u \cup \mathcal{N}_v$ have γ neighbors, so clearly $|\Omega| \leq 2\gamma^2$. However, this counts edges (x, y) where $x \in \mathcal{N}_u$ and $y \in \mathcal{N}_v$ twice. Assuming that all edges are equally likely, we have that the probability that there is an edge between x and an arbitrary node $z \in \mathcal{U}$ is $\frac{\gamma}{k}$. Then the number of edges from x to $\mathcal{N}_v$ is, on average, $\frac{\gamma^2}{k}$. It follows that the total number of edges between $\mathcal{N}_u$ and $\mathcal{N}_v$ is, on average, $\frac{\gamma^3}{k}$. Subtracting the edges that have been doubly counted, we get that $|\Omega| = 2\gamma^2 - \frac{\gamma^3}{k}$. □

By our assumptions, we then have that $|\mathcal{G}| = k\gamma$, $|Y| = \gamma - 1$, and $|A| = |C| = |W' \setminus Y| = \gamma - \lambda = \frac{\gamma(k-\gamma)}{k-1}$. By substituting into (12), we obtain

$$\chi = k\gamma\left(1 + \frac{\gamma(\gamma-1)(k-\gamma)}{k-1} + \min\left\{\frac{\gamma^2(\gamma-1)(k-\gamma)^2}{2(k-1)^2}, \frac{\gamma(k-\gamma)^2}{k}\right\}\right). \tag{13}$$

In the extreme cases, for very sparse and dense graphs, we have $\lim_{\gamma\to 1} \chi = k$ and $\lim_{\gamma\to k} \chi = k^2$. In the intermediate case, for instance for $\gamma = \frac{k}{2}$, we have complexity $\mathcal{O}(k^4)$.

This bound is verified by simulations on various graphs (codes), i.e., running WB_ELC$_{(a)}$ and counting the number of candidate edges checked by an implementation of Theorems 3 - 6, whether it uses recursive ELC or computes the counting formulas (we assume the overhead is the same). Note the counter, κ, in Alg. 2. Given a graph, $\mathcal{G}$, representing a code, the simulation consists of counting the number of candidate edges considered to produce $L(\mathcal{G}, T)$. The specific threshold, T, is not important for this simulation, as it does not affect the complexity in enumerating all WB-ELC sequences (when we count complexity in number of edges considered), so we use $T = 0$. This count is added to position $|\mathcal{G}|$ of a vector, $\mathbf{c}$, denoted by $\mathbf{c}(|\mathcal{G}|)$. The number of distinct graphs of weight $|\mathcal{G}|$ is counted in a similar vector, $\mathbf{d}$. In this work, we focus on self-dual codes which are also even or doubly even. Since H consists of codewords of the dual code, the weight of $\mathcal{G}$ also increases in steps of 2 or 4, respectively.

Random WB-ELC is then applied until a new (i.e., distinct) graph is found, and the process is repeated until all entries of $\mathbf{d}$ are greater

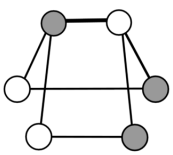
than some minimum number, g (we used $g = 10$).[5] Let $\mathcal{G}_{\max}$ denote a graph of "weight 50%" (6), corresponding to random ELC, beyond which we no longer think of the weight as bounded. Fig. 10 shows the simulated average complexity of Alg. 2 on the codes described previously, plotted against average column weight, $\bar{\gamma}(\mathcal{G}) = |\mathcal{G}|/k$, in the interval $[\bar{\gamma}(|\mathcal{G}|_{\min}),\ \bar{\gamma}(|\mathcal{G}|_{\max})]$. This interval completely describes the encountered graphs found by ELC for the given code. These simulations are compared against the bound of (13) for the same values.

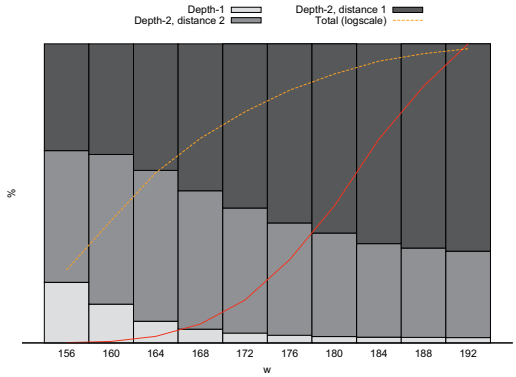
A related set of data from the same simulations is the relative fraction of WB-ELC sequences in $L(\mathcal{G}, T)$ attributed to Theorems 3, 5, and 6, and how this distribution varies with $|\mathcal{G}|$. Let $\mathcal{G}' = \mathbf{e}(\mathcal{G})$ for some $\mathbf{e} \in L(\mathcal{G}, T)$. Recall that the simulations produced $L(\mathcal{G}, 0)$, so the WB-ELC sequences found must give $|\mathcal{G}|_{\min} \leq |\mathcal{G}'| \leq |\mathcal{G}|$. Define the matrix W by expanding the array $\mathbf{c}$ by a dimension of length 3. By grouping the WB-ELC sequences in $L(\mathcal{G}, T)$ by theorem, $W(|\mathcal{G}|, t)$ counts the number of WB-ELC sequences corresponding to distance $t = 0, 1, 2$ (Theorems 3, 5, and 6, respectively). The total number of WB-ELC sequences for a weight class is $\sum_t W(|\mathcal{G}|, t)$. Fig. 11 shows this data plotted as stacked histograms, where the height of each segment gives the relative fraction (percentage of $\sum_t W(|\mathcal{G}|, t)$) of the corresponding theorem for that weight class. These show a trend, which holds for various codes, that the fraction of depth-1 WB-ELC sequences decreases for increasing weight, before stabilizing at a small but constant fraction. As $|\mathcal{G}|$ increases, it must necessarily become easier to find edges for which ELC will preserve the weight, culminating at $|\mathcal{G}|_{\max}$. Accordingly, it must be even easier to bound the weight even more by using double ELC (depth-2 WB-ELC). Furthermore, we note a dominance of Theorem 6 over Theorem 5, which validates our implementation checking Theorem 5 last (nested in the innermost loop).

Also shown is the average total number of WB-ELC operations found for each weight class, normalized by $\nu = \max_W \sum_t W(|\mathcal{G}|, t)/\mathbf{d}(|\mathcal{G}|)$ (the largest observation) to produce a percentage (i.e., to fit in the same plot), $\nu^{-1} \sum_t \frac{W(|\mathcal{G}|,t)}{\mathbf{d}(|\mathcal{G}|)}$, for each position in $\mathbf{c}$ (weight class).

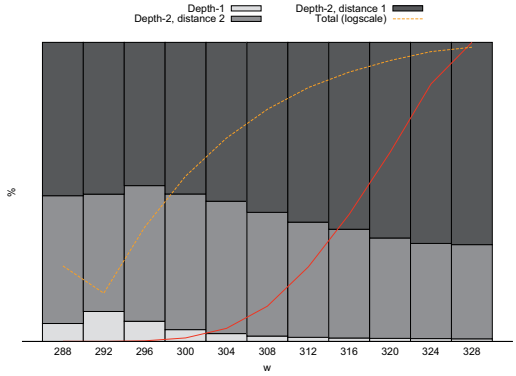
For enumeration purposes, it may also be interesting to enumerate only those WB-ELC operations involving a specific node, $v^\star$. By restricting the operation of Alg. 2 to $v^\star$ only, rather than $\mathcal{V}$, the single

[5] The distribution of $|\mathcal{G}|$ usually resembles a normal distribution, so g is to ensure a minimum number of observations are achieved at the tails (high and low ends). The total number of graphs, $\sum \mathbf{d}(|\mathcal{G}|)$, is much greater than $10\,(|\mathcal{G}|_{\max} - |\mathcal{G}|_{\min})$.

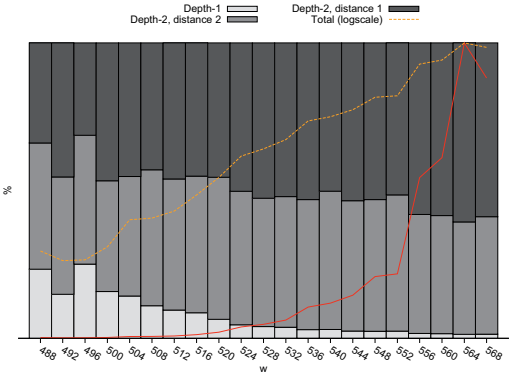




(a) $[36, 18, 8]$ code, "$R2$."



(b) $[48, 24, 12]$ code, "EQR48."



(c) $[68, 34, 18]$ code, "$C_{68,1}$."

Fig. 11: *The percentage of depth-1 (Theorem 3) and depth-2 (Theorems 5 and 6) WB-ELC found by WB_ELC$_{(a)}$, for increasing weights, $|\mathcal{G}|$. Also shown (in red) is the (normalized) total count of WB-ELC sequences, and the plot in logarithmic scale (orange).*

iteration of the outer loop returns the subset of operations which include $v^\star$. Due to the locality argument (Theorem 4), this does indeed find the entire subset of $L(\mathcal{G}, T)$ rooted in $v^\star$, i.e., of the form $(u, v^\star)$ or $\{(u, v^\star),\ (u', v')\}$.

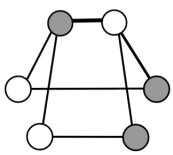
4.2 Reduce Weight in IP-form

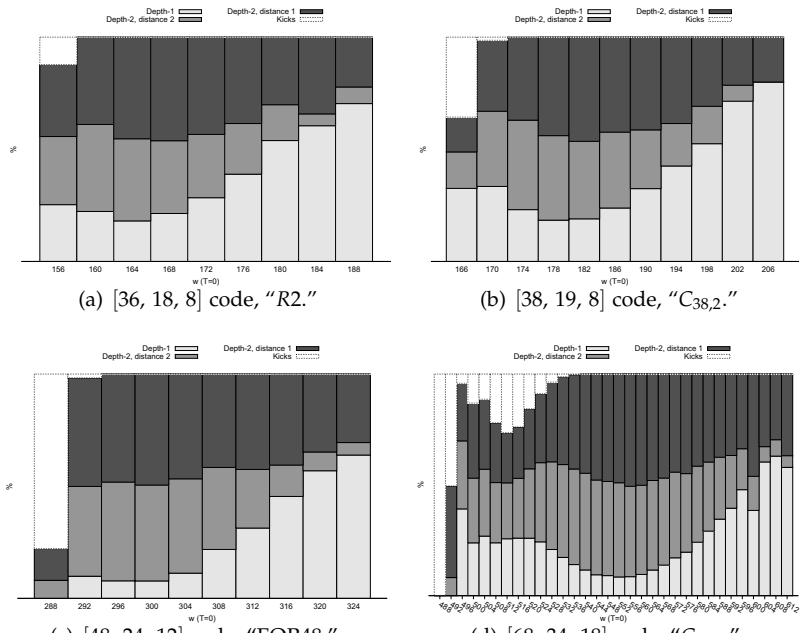
The WB-ELC algorithm can be used to reduce the weight of a bipartite graph. By definition (3), the parity-check matrix related to $\mathcal{G}$ is in systematic form. The purpose of reducing the weight of a systematic matrix may be motivated by the SPA-WBELC algorithm described in Section 5, but is in itself an interesting special (and more difficult) instance of the often encountered problem of finding a reduced-weight basis for a (dual) code, which has been shown to be very hard [12].

We define instance (b) of Alg. 2, denoted WB_ELC$_{\text{(b)}}$, as one which terminates upon the first encountered WB-ELC operation to satisfy the threshold, T. The returned WB-ELC operations should ideally be uniform samples of $L(\mathcal{G},\ T)$. To improve the uniformity of the sampling, all sets in Alg. 2 are permuted randomly before being traversed in the (for) loops. This modification does not affect the ability to enumerate all WB-ELC operations (if the stopping criterion mentioned above is removed) only the order in which these are encountered, as is now desired. As long as the current graph, $\mathcal{G}$, is of weight $|\mathcal{G}|_{\min} \leq |\mathcal{G}| \leq |\mathcal{G}|_{\min} + T$, and $L(\mathcal{G},\ T') \neq \emptyset$, where $T' = |\mathcal{G}|_{\min} + T - |\mathcal{G}|$, WB_ELC$_{\text{(b)}}$ will find a WB-ELC operation, $\mathbf{e}$.

A simple algorithm is proposed, called Alg. minIP (listing omitted), which repeatedly invokes WB_ELC$_{\text{(b)}}(\mathcal{G}$, -1$)$ to determine a random WB-ELC sequence, $\mathbf{e}$, such that $|\mathbf{e}(\mathcal{G})| < |\mathcal{G}|$. We refer to this as Alg. minIP. Eventually, a graph is reached from which it is not possible to further reduce the weight, such that WB_ELC$_{\text{(b)}}$ returns $\mathbf{e} = \emptyset$. At this point, the proposed algorithm proceeds by random (unbounded) ELC until the weight is increased and the reduction may resume. Alternatively, WB_ELC$_{\text{(b)}}$ may be modified to return the operation encountered which gives the smallest weight increase. Both these heuristics are referred to as a *kick.*

Table 3 compares the performance of Alg. minIP against other algorithms to reduce weight. Recall the codes selected for this work. The corresponding parity-check matrices are optimized on weight, both in nonsystematic form, as well as systematic form. The weight of the initial matrix, H_0, e.g., as produced by Magma, is denoted by $|H_0|$ (where (4)





(a) $[36, 18, 8]$ code, "$R2$."
(b) $[38, 19, 8]$ code, "$C_{38,2}$."
(c) $[48, 24, 12]$ code, "EQR48."
(d) $[68, 34, 18]$ code, "$C_{68,1}$."

Fig. 12: *The percentage of depth-1 (Theorem 3) and depth-2 (Theorems 5 and 6) returned (picked) by WB_ELC$_{(b)}$, for increasing weight, $|\mathcal{G}|$. The fraction of kicks is also shown, corresponding to when Alg.* minIP *gives a new graph, $\mathcal{G}$, for which $L(\mathcal{G}, 0) = \emptyset$.*

gives the weight of $\mathcal{G}$). Alg. IP is a recursive, deterministic depth-first algorithm to traverse the orbit of a code, by means of ELC operations on $\mathcal{G}$. Unless the orbit of the code is impractically large, this approach will determine the minimum-weight systematic matrix. However, for most codes, the search space is exponential in n. The corresponding column in Table 3 shows the lowest weight systematic matrix found. For "$R2$" and "$\mathcal{C}_{38,2}$," we were able to compute the entire ELC orbit of the codes, to find optimal-weight matrices in systematic form. For "$\mathcal{C}_{68,1}$," the orbit is infeasibly large, yet, using WB-ELC preprocessing, we were able to find a systematic matrix of weight 488. For nonsystematic form, Alg. nonIP takes random combinations of minimum-weight codewords of the dual code (to generate the parity-check matrix), and attempts to find combinations of $n - k$ linearly independent rows. Generally, this algorithm succeeded in finding minimum-weight matrices

Table 3: *Reduced-weight matrices. Results of Alg.* MINIP *compared to other algorithms. Column* $|H_0|$ *specifies the weight of the initial parity-check matrix,* H_0*, (e.g., as constructed by* MAGMA*). The bound does not assume a systematic form matrix.*

Code		$\lvert H_0\rvert$	Bound (2)	MINIP	IP	NONIP
BCH15	$[15, 5, 7]^{\dagger}$	42	40	40 [1]	40	40 [4]
Ext. Golay	$[24, 12, 8]$	96	96	96 [1]	96	96 [4]
$R2$	$[36, 18, 8]$	188 $\star$	144	156	156 [2]	152
$C_{38,2}$	$[38, 19, 8]$	240 $\star$	152	166	166 [2]	154
EQR48	$[48, 24, 12]$	320	288	288 [1]	288	288 [4]
$C_{68,1}$	$[68, 34, 12]$	612 $\star$	408	488	492	410 [3]

$\dagger$ Only code which is *not* self-dual, and $d_{\mathbf{min}}(\mathcal{C}^{\perp}) = 4$.

$\star$ Initial matrix described in [28, 29] (not generated by MAGMA).

[1] Lower bound (2) on matrix minimum weight achieved.

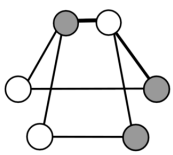
[2] Entire orbit enumerated, so these weights are optimum for systematic matrices.

[3] Minimum-weight matrix possible to construct using 33 minimum-weight (dual) codewords, and one of weight 14.

[4] Optimal weight in nonsystematic form, $|H| = (n-k)d_{\mathbf{min}}(\mathcal{C}^{\perp})$.

of weight $(n-k)d_{\mathbf{min}}(\mathcal{C}^{\perp})$, or at least coming quite close. In terms of search time, only the largest code required more than a few seconds to find the reported matrices, using a standard desktop computer (Alg. IP and Alg. NONIP ran for ~ 1 day, while Alg. MINIP used ~ 3 days).

Fig. 12 shows the response of WB_ELC$_{\text{(b)}}$ in terms of which theorem is returned from the randomized algorithm with $T = 0$, for increasing weight, $|\mathcal{G}|$. As discussed previously, the fraction of all WB-ELC sequences (i.e., when considering the entire $L(\mathcal{G}, T)$) corresponding to depth-1 is small but constant (Fig. 11), but comes to dominate the response of WB_ELC$_{\text{(b)}}$ simply because it is natural and convenient to check Theorem 3 first. Similarily, the fraction corresponding to depth-2 reflects the position of these theorems in Alg. 2, based on the results of Fig. 11. As in Fig. 11, random WB-ELC sequences are used to find the next distinct Tanner graph, $\mathcal{G}'$. Then, WB_ELC$_{\text{(b)}}$ with $T = 0$ is used to choose a random WB-ELC for $\mathcal{G}'$, and the corresponding theorem is added to the counter for the weight class $|\mathcal{G}'|$. In order to increase the observations (number of distinct Tanner graphs) at the low-weight



weight classes, Alg. MINIP (i.e., T = -1) is used to find the next distinct Tanner graph, $\mathcal{G}'$. This simulation illustrates the relative occurrence of Theorems 3, 5, and 6 (and kicks) for a graph of a given weight, when $T = 0$. This response can give an indication on which theorem occurs more frequently as a function of $|\mathcal{G}|$.

It should be emphasized, however, that WB_ELC$_{(b)}$, if run in succession with $T \geq 0$, does *not* get stuck unless $L(\mathcal{G}, T) = \emptyset$ for the starting graph, $\mathcal{G}$. The overall trend for the various codes simulated is that the number of kicks (for $T = 0$) decreases for increasing graph weight – which makes sense, as finding a WB-ELC sequence is, intuitively, more difficult for sparser graphs. Figs. 15(a), 16(a), and 17(a) show the performance of WB_ELC$_{(b)}$ in terms of number of candidate edges considered on average in order to find a random WB-ELC sequence, for increasing T – see Section 5 for a discussion.

4.3 BOUNDED-WEIGHT SUB-ORBIT

The orbit of a code is the set of distinct (nonisomorphic) graphs one finds by any sequence of ELC operations. Specializing this to WB-ELC operations, the same procedure gives a bounded-weight sub-orbit of the code in which all graphs are of weight $|\mathcal{G}'| \leq |\mathcal{G}_0| + T$. The size of this sub-orbit depends on $\mathcal{G}_0$, and, obviously, T. Using $T = |\mathcal{G}|_{\max} - |\mathcal{G}_0|$, corresponding to unbounded weight (i.e., random ELC), we enumerate the entire orbit of the code. Otherwise, another (disjoint) bounded-weight sub-orbit may be found "around" a graph $\mathcal{G}'$, where $|\mathcal{G}'| \leq |\mathcal{G}_0| + T$, which is not already in some previously enumerated bounded-weight sub-orbit.

For increasing values of T, certain sub-orbits which are disjoint for lower T may become linked to form supercomponents, as shown figuratively in Fig. 13. For the "EQR48" code, the minimum weight is 288. Using data from a random search, we found 110 distinct minimum weight structures (all connected in either pairs or triples), and, within $T = 4$, the largest supercomponent found contains 371 structures of weight 288 and 292. Including also $T = 8$, the resulting sub-orbit contains only one supercomponent, connecting all $\sim 70\,000$ graphs found in our random search. In contrast, a brute force (recursive) attempt to enumerate the orbit of this code, only found 30 minimum weight graphs before running out of memory on a standard desktop computer.

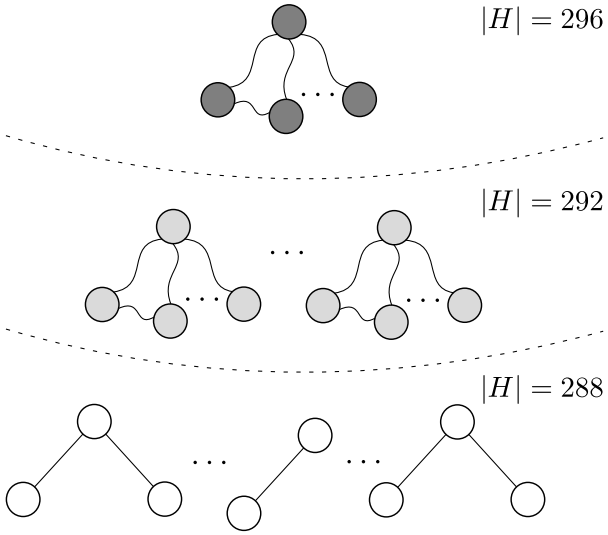


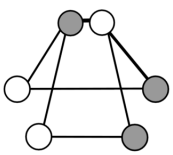
Fig. 13: *Sub-orbits of "EQR48" code, for which $|\mathcal{G}_0| = 288$. Nodes represent distinct graphs of the indicated weight, and edges represent a WB-ELC operation connecting two graphs. Curved edges mean arbitrary WB-ELC operations may exist. Between weight classes, this is indicated by dashed lines. The components within the same weight class are disjoint (if we do not cross dashed lines).*

5 Generalized SISO HDPC Decoder

For this work, the most important application is the use of WB-ELC operations during SISO HDPC decoding, where the aim is to have increased diversity (i.e., more parity-check matrices for the same code) which are all well-suited for use in iterative decoding. Several parameters of a parity-check matrix affect its suitability for decoding, where one of these is the weight, or density, of the matrix. Let the received noisy channel vector be $\mathbf{y} = (\text{-}1)^{\mathbf{x}} + \mathbf{n}$, where $\mathbf{x}$ is a codeword and $\mathbf{n}$ is AWGN. In the log-likelihood ratio (LLR) domain, the initial LLR at position v is $L_0^v \triangleq \frac{2}{\eta^2} y_v$, where η is the standard deviation of the AWGN.

5.1 Edge-Local Damping Rule

The generalized SISO HDPC decoder, Alg. 3, gives a common framework for iterative decoding of HDPC codes, where different opera-



Algorithm 3 SISO-HDPC$(p, I_1, I_2, I_3, \alpha_0, \text{OP}, \text{DR})$

$\alpha = \alpha_0$
for I_3 times **do**
  Restart decoder from channel vector
  for I_2 times **do**
    Stop if syndrome check is satisfied
    Apply damping rule, DR, with coefficient α
    Apply at random p operations, OP
    for I_1 times **do**
      Apply SPA iteration ('flooding' scheduling)
    end for
  end for
  Increment damping coefficient, $\alpha := \alpha_0 + (1 - \alpha_0)\frac{I_3}{I_3 - 1}$
end for

tions can be used to give diversity during decoding. The framework is based on the SPA-PD algorithm [13], and centers around applying a damping routine [12] to SPA iterations interspersed with operations to give increased diversity during decoding. Three nested loops control the dynamic damping scheme. The maximum number of iterations is $\tau = I_1 I_2 I_3$, and for every I_1 iterations a diversity stage is executed, in which the extrinsic contribution of the LLRs, Γ_j^v, of each variable node, v, is scaled down by a damping coefficient, α, $0 < \alpha < 1$, and accumulated on the input to the next iteration according to

$$L_{j+1}^v = L_j^v + \alpha \Gamma_j^v. \tag{14}$$

The extrinsic contribution to variable node v (the sum of all incoming messages, $\mu_j^{v \leftarrow u}$) in iteration j is

$$\Gamma_j^v = \sum_{u \in \mathcal{N}_v} \mu_j^{v \leftarrow u} \tag{15}$$

where we define $\Gamma_0^v \triangleq 0$. The damping coefficient may be viewed as a measure of *trust* in the information produced by the Tanner graph, which is scaled down, as opposed to the information received from the channel, which is never damped. Each time we increment α, "the contribution from the Tanner graph" is strengthened, so the outer loop is an implementation of I_3 independent serial decoders of the received channel vector, for varying values of α (allowing a parallel implementation).

The rationale behind damping originates from gradient algorithms, where α is the *step width,* which is varied in order to prevent the algorihm (in our case, the convergence of the iterative decoding process, in terms of negative sum of soft syndromes) from getting stuck at pseudo-equilibrium points (local minima) [12, 30]. The contribution from the received noisy channel vector is never damped, which is obvious if we rewrite (14) as $L_{j+1}^{v} = L_0^v + \alpha\Sigma_{j'=1}^{j}\Gamma_{j'}^{v}$. These new, damped LLRs are then used to re-initialize the decoder. So, after *resetting* all messages to neutral LLRs,

$$\mu_j^{v\leftarrow u} := 0,\ \forall\ (u,v) \in \mathcal{G} \tag{16}$$

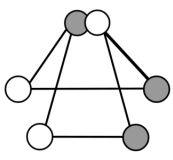
iteration $j+1$ begins by, in effect, forwarding the new, damped input towards the check nodes. This global reset stage is necessary when the operation used in the SISO HDPC decoder acts on the variable node level, e.g., as in SPA-PD which permutes $\mathbf{L}$ [13]. After this, the relationship (15) between extrinsic information (on edges) and LLRs (in nodes) does not hold. The global stage of (14) and (16), followed by re-initializing all edges, is referred to as *global damping* (GD).

In contrast to GD, we have previously proposed edge-local damping schemes more suited for the edge-local action of ELC [15, 22]. The damping rule (14) can be generalized to include and take advantage of extrinsic information on an edge. Formulated for the edge-local perspective (damping each edge separately), the damping rule (14) becomes

$$\mu_{j+1}^{v\rightarrow u} = L_j^v + \alpha(\Gamma_j^v - \mu_j^{v\leftarrow u}) \tag{17}$$

where the extrinsic contribution $\mu_j^{v\leftarrow u}$ is subtracted, to adhere to the extrinsic principle of the SPA.

ELC on (u',v') complements the edges of $\mathcal{E}_{u',v'}$. By defining a flooding SPA iteration as the update of all check nodes followed by all variable nodes (this is usually done in the opposite order, at no general significance), we ensure that all soft information on edges which are *removed* from any $v \in \mathcal{N}_{u'}^{v'}$ is contained (summed) in Γ^v. Thus, we need only focus on edges *inserted* by ELC, i.e., precisely $(u,v) \in \mathcal{E}_{u',v'}$ after ELC. Fig. 14 shows an example situation, using the mapping from Tanner graph to simple graph where a check node is grouped with its systematic variable node. The figure shows the situation *after* ELC on (u',v'), where the solid lines are the inserted edges. These new edges must be initialized with some outgoing message, $\mu_{j+1}^{v\rightarrow u}$, before



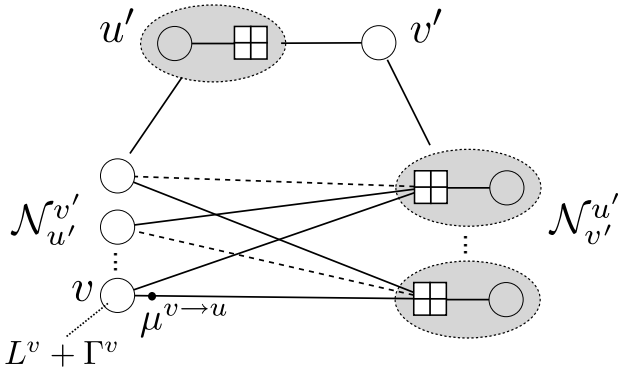


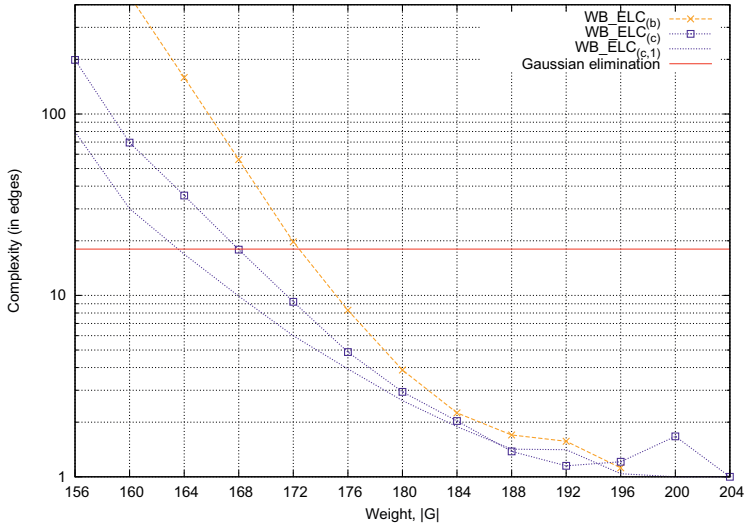
Fig. 14: *Description of LD, affecting all edges inserted by ELC on* (u', v')*. These new edges (solid lines),* $(u, v)\ \forall\ v \in \mathcal{N}_{u'}^{v'}$*, are initialized for the next iteration with* $\mu_{j+1}^{v\to u} = L_j^v + \alpha(\Gamma_j^v - \mu_j^{v\leftarrow u}) = L_j^v + \alpha\Gamma_j^v$*, since* $\mu_j^{v\leftarrow u} = 0$*.*

the next SPA iteration (iteration $j+1$, which begins with check nodes), so (17) implements a damping-and-initialization rule. However, since $\mu_j^{v\leftarrow u} = 0$ for new edges, (17) reduces to (14). We emphasize that the edges connected to $\mathcal{N}_v \setminus \mathcal{N}_{v'}$, i.e., those unaffected by ELC on (u', v'), are *not* damped and retain their extrinsic messages for the next iteration. Restricting damping to the edges affected by ELC is referred to as *edge-local damping* (LD) [15]. There is still some loss of extrinsic information due to ELC, since only the *sum* of adjacent messages is stored in a variable node, yet the loss is significantly smaller than that resulting from the reset stage involved in GD. We also propose a more advanced local damping rule in [22], which was not found effective for the codes used in this work.

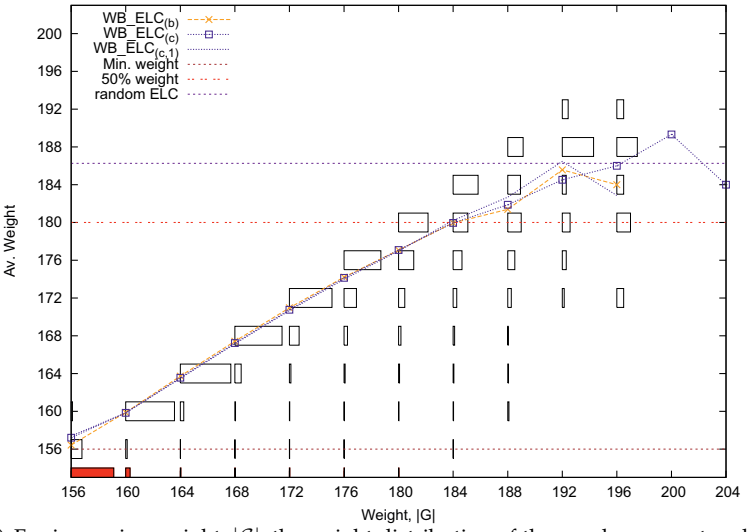
Every $I_1 I_2$ iterations, α is incremented towards 1, and the decoder is restarted (I_3 times) from the received noisy channel vector. This constitutes, in effect, the starting of a new, serial decoder, only with a new and increased (i.e., reduced effect) damping coefficient.

5.2 Real-Time WB-ELC Algorithm

In a decoding setting, where the Tanner graph to be modified contains soft information on the edges, it is natural to focus on the approach of using the counting formulas to minimize loss of soft information in the WB-ELC stage. The aim is to produce a random set of reduced-weight

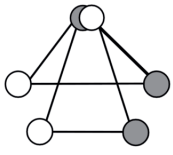


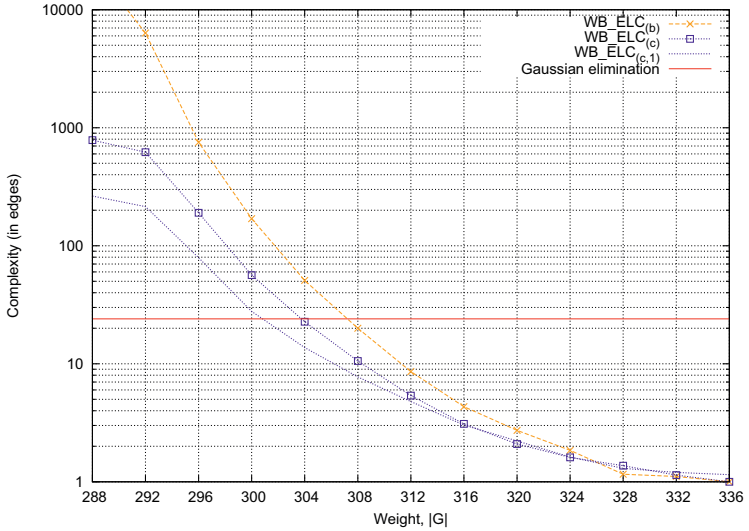
(a) $\text{WB\_ELC}_{(b)}$ (random search) compared to $\text{WB\_ELC}_{(c)}$ (real-time version) and a reduced real-time version searching only at depth-1.



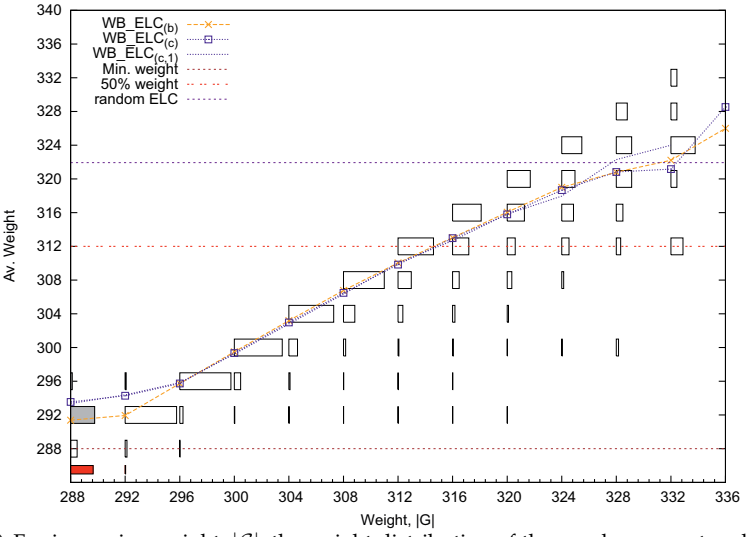
(b) For increasing weight, $|\mathcal{G}|$, the weight distribution of the graphs encountered during random traversal of the sub-orbit of the code approaches a normal distribution, with average centered at the expected weight found by random (unbounded) ELC.

Fig. 15: *Code "R2."*



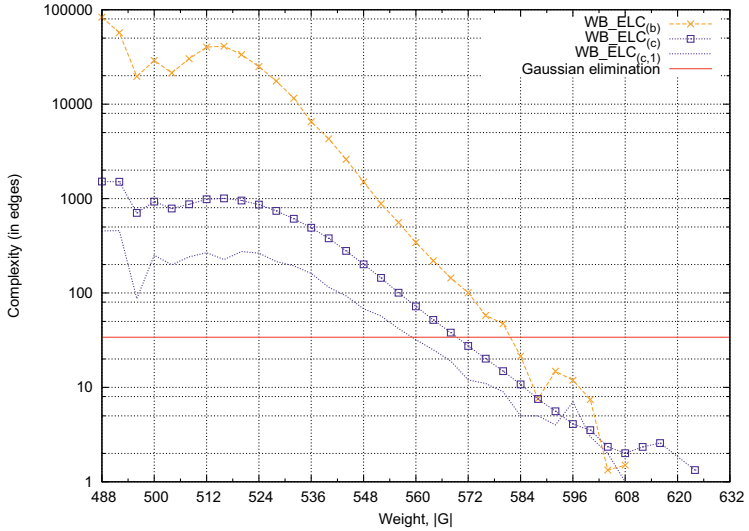


(a) $\text{WB\_ELC}_{(b)}$ (random search) compared to $\text{WB\_ELC}_{(c)}$ (real-time version) and a reduced real-time version searching only at depth-1.

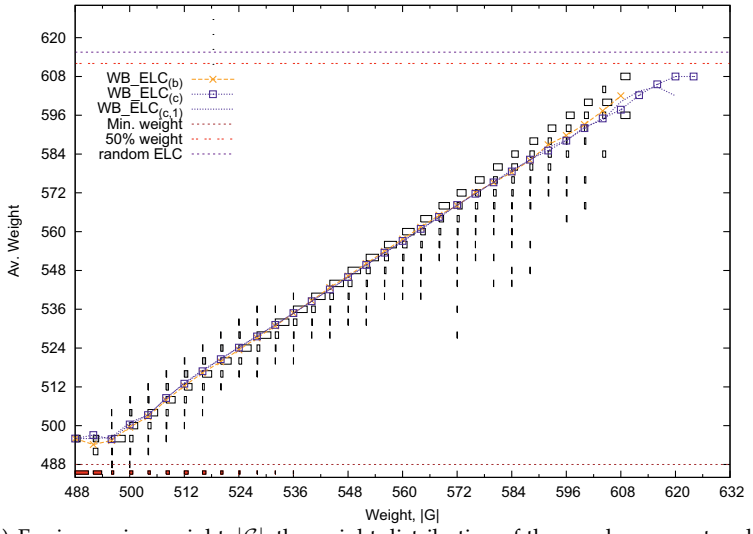


(b) For increasing weight, $|\mathcal{G}|$, the weight distribution of the graphs encountered during random traversal of the sub-orbit of the code approaches a normal distribution, with average centered at the expected weight found by random (unbounded) ELC.

Fig. 16: *Code "EQR48."*

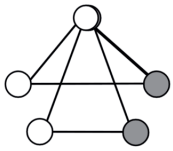


(a) $\text{WB\_ELC}_{(b)}$ (random search) compared to $\text{WB\_ELC}_{(c)}$ (real-time version) and a reduced real-time version searching only at depth-1.



(b) For increasing weight, $|\mathcal{G}|$, the weight distribution of the graphs encountered during random traversal of the sub-orbit of the code approaches a normal distribution, with average centered at the expected weight found by random (unbounded) ELC.

Fig. 17: *Code "C_{68}."*



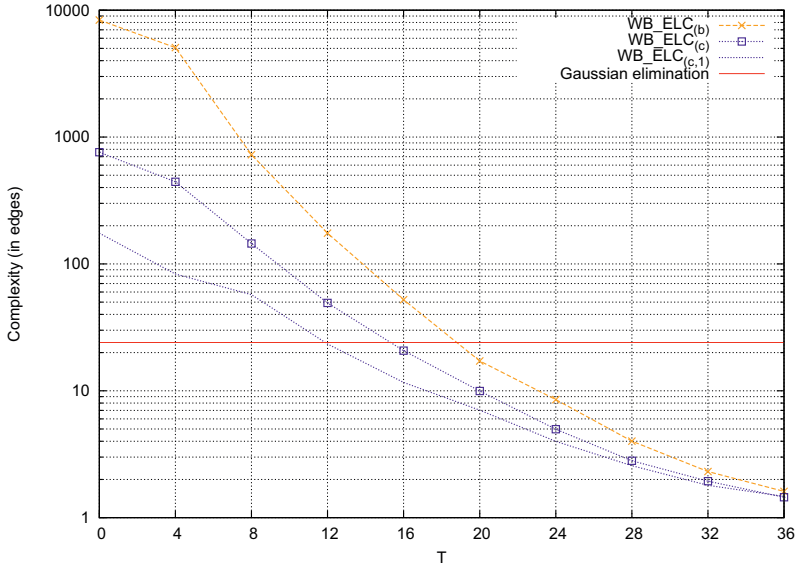
Tanner graphs for $\mathcal{C}$, at minimum (search) overhead. As before, the weight of each graph should be upper-bounded, $|H| \leq |H_0| + T$, and we determine the *local threshold*

$$T' = |H_0| + T - |H| \tag{18}$$

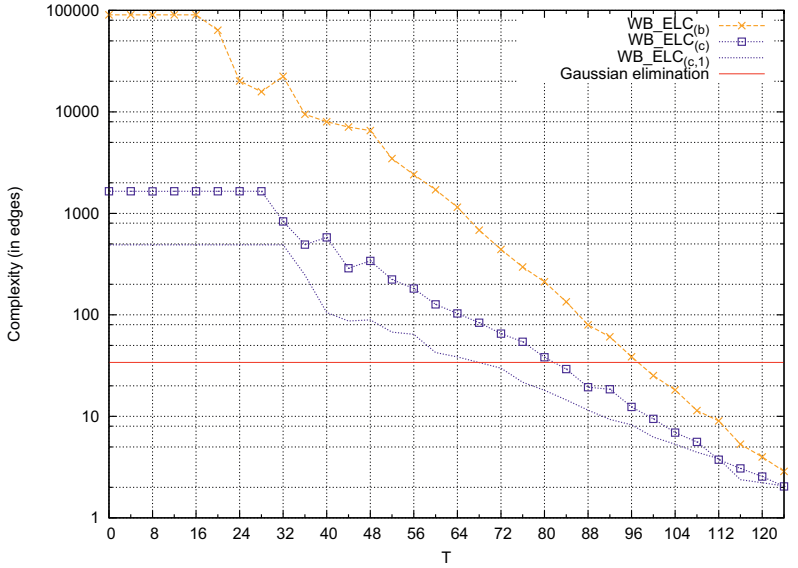
based on the weight $|H|$ of the current graph, H. The WB-ELC stage is to be done during decoding, in between SPA iterations, so efficiency is a key concern. Based on WB_ELC$_{(b)}$, we now consider a set of further modifications to give a *real-time* version, which we refer to as WB_ELC$_{(c)}$. By increasing the coarseness of the search, a random WB-ELC sequence may be found at a decreased average complexity, in terms of number of candidate edges considered. Rather than exhaustively traversing the entire local subgraphs induced (entire sets A, C, etc.) by the theorems around the *root edge,* (u, v), the search may be confined to checking the depth-2 theorems for only one, random choice of (u', v'), and one for (u'', v'') – see Fig. 9. This way, Theorem 3 is checked while producing the subsets required for Theorems 5 and 6, and WB_ELC$_{(c)}$ applies this coarse search using all edges $(u, v) \in \mathcal{G}$ as root edge. In this sense, depth-1 has a natural precedence over depth-2, (single ELC is half the complexity of double ELC). As shown in Fig. 11, the percentage of depth-2 WB-ELC sequences constitute a large fraction of the total sequences output by WB_ELC$_{(a)}$ (they are relatively easy to find), validating the heuristic of only checking Theorems 5 and 6 on a subset of the candidate edges.

Figs. 15(a), 16(a), and 17(a) compare the complexity of WB_ELC$_{(c)}$ to WB_ELC$_{(b)}$. The complexity shows the average number of edges checked to find the first random WB-ELC operation on a graph of weight $|\mathcal{G}|$, and for $T = 0$. As before, each weight class is simulated independently, and new graphs are found using Alg. MINIP. We also compare a variant of WB_ELC$_{(c)}$ which only considers depth-1 WB-ELC, denoted by WB_ELC$_{(c,1)}$. The reduced search space gives a further improvement in number of candidate edges checked. From a decoding perspective, it makes sense to compare this *ELC complexity* to the adaptive belief propagation (ABP) algorithm, which uses a GE stage in between SPA iterations [10, 22]. Implemented in terms of ELC operations, the complexity of a GE stage is $n - k$, which is included as the red reference line.

Figs. 15(b), 16(b), and 17(b) illustrate the graph weights resulting from the previous simulations. For WB_ELC$_{(b)}$, a (horizontal) histogram is plotted for each weight class $|\mathcal{G}|$, showing the distribution of the weights

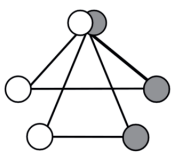


(a) [48, 24, 12] code, "EQR48." $|\mathcal{G}_0| = 288$.



(b) [68, 34, 18] code, "$C_{68,1}$." $|\mathcal{G}_0| = 488$.

Fig. 18: *Average number of edges checked to determine a sequence of* 1 000 *random WB-ELC sequences, simulated for increasing parameters T. All simulations begin with the same reduced-weight graph,* $\mathcal{G}_0$.



of the graphs, $\mathcal{G}'$, found when the search starts from $\mathcal{G}$. This distribution is over $|\mathcal{G}_0| \leq |\mathcal{G}'| \leq |\mathcal{G}|$, with an average around $|\mathcal{G}|$, until $|\mathcal{G}|$ reaches the average weight resulting from random (i.e., unbounded) ELC, beyond which the distributions begin forming a normal distribution centered on the average weight described in (6). The average weight due to repeated random (unbounded) ELC is slightly higher than the "50% weight." The fraction of repeated Tanner graphs (as in row-equivalent matrices) encountered is indicated by the red bar below the histograms. Only at the low-weight end of the scale, when $|\mathcal{G}|$ is near $|\mathcal{G}_0|$, are such repetitions encountered, which indicates that, generally, a very high degree of diversity results from using random WB-ELC operations during decoding. Some fraction of the encountered graph weights in the histograms is *above* the maximum weight of the class, $|\mathcal{G}'| > |\mathcal{G}|$ (recall that we simulate $T = 0$). This indicates the fraction of graphs for which the weight can not be upper-bounded by $T = 0$, resulting in a kick to a higher weight. Any such kicks are a result of Alg. MINIP giving a new graph, $\mathcal{G}'$, for which $|\mathcal{G}'| < |\mathcal{G}|$ (using $T =$ -1), such that $\text{WB\_ELC}_{(b)}(\mathcal{G}', 0) = \emptyset$. However, this occurs only at the low-weight end of the scale and we may conclude that the search is, in fact, able to maintain the desired weight-bounding. The average weight plots for $\text{WB\_ELC}_{(c)}$ and $\text{WB\_ELC}_{(c,1)}$ show only a slight increase in average weight at the low-weight end, which indicates the number of kicks resulting from the reduced-complexity search. This indicates a complexity reduction with negligible performance penalty. Furthermore, we verify the above assumption that the weight can be bounded by depth-1 WB-ELC alone, by noting that $\text{WB\_ELC}_{(c,1)}$ also succeeds in bounding the weight.

Fig. 18 shows a similar experiment which focuses on the performance of WB-ELC in a decoding setting. Starting from a reduced-weight graph, $\mathcal{G}_0$, we simulate the average performance of keeping $|\mathcal{G}|$ upper-bounded by $|\mathcal{G}_0| + T$. Reflecting the intended usage, $\mathbf{e} = \text{WB\_ELC}_{(b)}(\mathcal{G}, T')$ (or (c) or (c,1)) is used (rather than Alg. MINIP) to find the next WB-ELC operation, where $T' = |\mathcal{G}_0| + T - |\mathcal{G}|$ using (18). This is then repeated for $\mathcal{G}' = \mathbf{e}(\mathcal{G})$ until 1 000 graphs are simulated for this T. Then, the experiment proceeds the same way for the next value of T. Again, we compare the complexity against a GE stage. Let T_{GE} denote the threshold for which the complexity of WB_ELC intersects this measure. For thresholds $T > T_{GE}$ the average complexity of a WB-ELC stage is lower than that of a GE stage, while still giving a weight-bounding effect (as compared to random ELC).

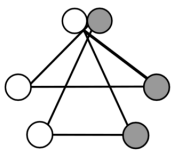
Table 4: *Decoding algorithms simulated in this work, and the corresponding configurations of Alg. 3*

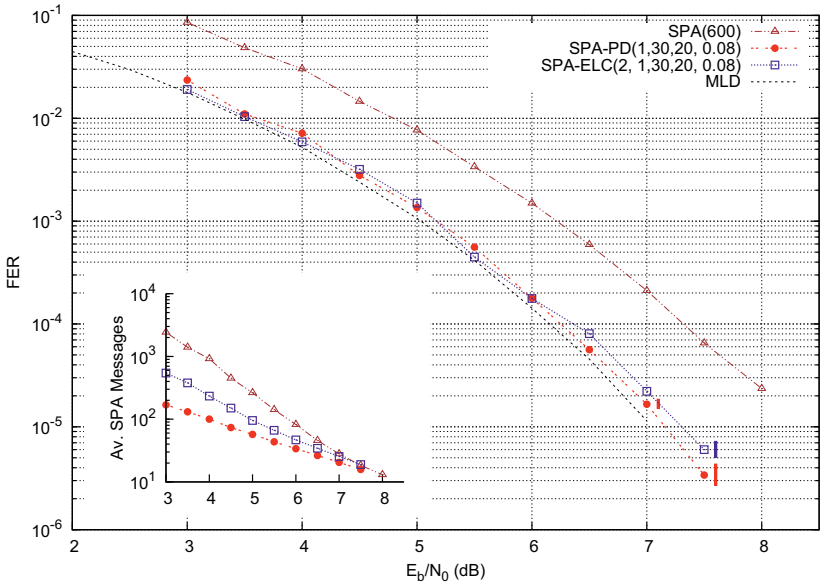
Decoding Algorithm	Configuration
SPA(τ)	SISO-HDPC$(0, 1, \tau, 1, 1, -, -)$
SPA-PD$(I_1, I_2, I_3, \alpha_0)$	SISO-HDPC$(1, I_1, I_2, I_3, \alpha_0, \text{PD}, \text{GD})$
SPA-ELC$(p, I_1, I_2, I_3, \alpha_0)$	SISO-HDPC$(p, I_1, I_2, I_3, \alpha_0, \text{ELC}, \text{LD})$
SPA-WBELC$(p, I_1, I_2, I_3, \alpha_0, T)$	SISO-HDPC$(p, I_1, I_2, I_3, \alpha_0, \text{WB_ELC}_{(\text{b})}(\mathcal{G}, T), \text{LD})$

5.3 Error-Rate Observations

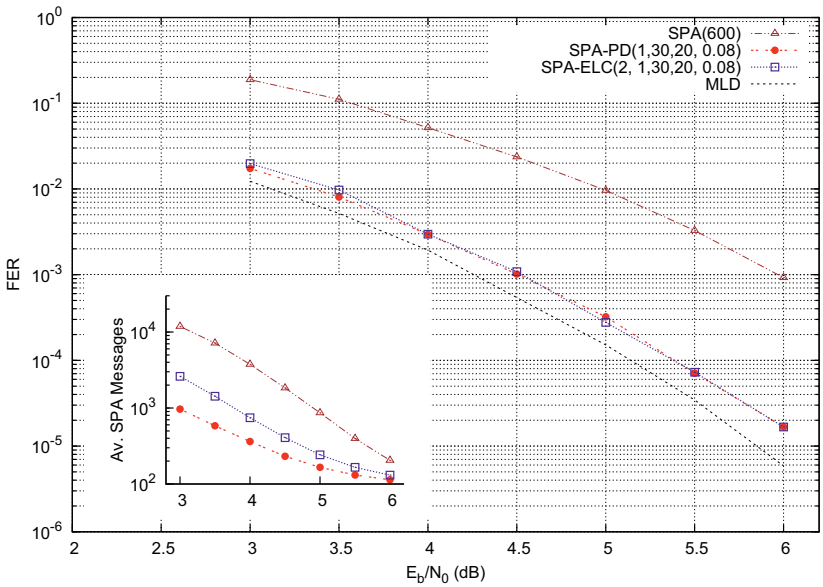
The SPA-ELC decoding algorithm, which uses random ELC operations in between SPA iterations to gain diversity, has previously been shown effective [15]. The simulation results in Fig. 19 compare the proposed SPA-ELC decoder against various decoding algorithms, where we ensure that $\tau = I_1 I_2 I_3$ is fixed. The algorithms are all implemented using the SISO HDPC framework, with the configurations summarized in Table 4. The parameter p specifies the number of ELC operations to employ every I_1 iterations. The values of p, I_1, I_2, and I_3 are chosen empirically, based on frame error-rate (FER) simulations. In Fig. 20, using code "$R2$," we fix one of the loop constants (and fix $\tau = 600$), and determine the second for varying values of I_1 which we know to have the greatest influence on performance. The nearly identical data obtained for fixed I_2 and fixed I_3 verifies that the performance is dominated by I_1, and that the best performance is found for low values of I_1, specifically for $I_1 = 1$. This is verified also for the "EQR48" code. The value of p is then selected based on data shown in Fig. 22, where we find an optimal value at reasonably low values of p. This value is only slightly sensitive to SNR. The initial damping coefficient, $\alpha_0 = 0.08$, is borrowed from [12].

The drawback of SPA-ELC is an increase in matrix weight, so unless the orbit of the code contains only "low" weight (i.e., near $|\mathcal{G}|_{\min}$) graphs (e.g., the BCH15 and the extended Golay code, for which the orbit contains only two graphs), the performance of the SPA is negatively affected by increased weight. Not only does this increase the complexity of computing the SPA update rules, but there is also a well-

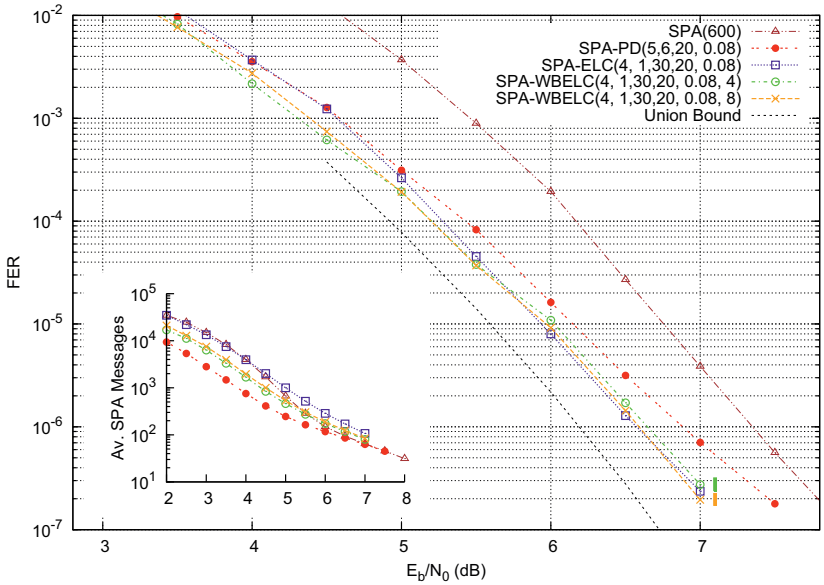




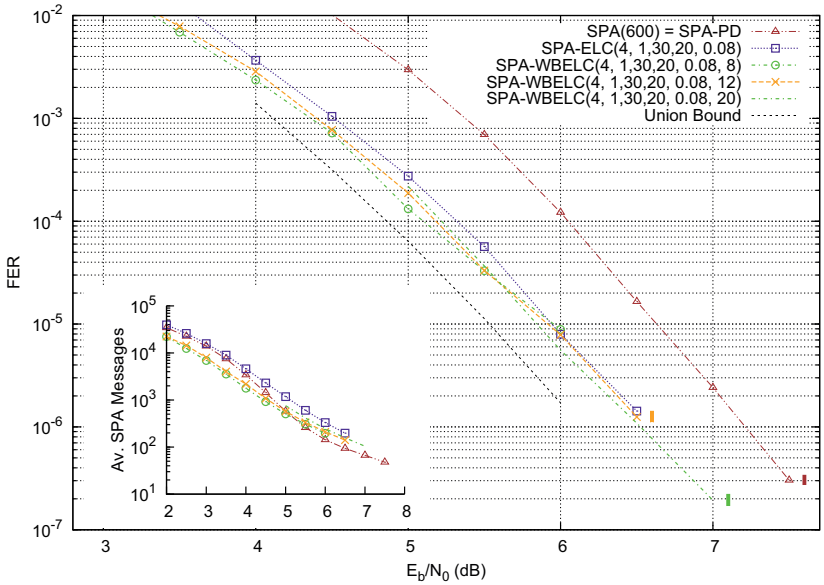
(a) BCH15 $= [15, 5, 7]$, with $|\mathrm{Aut}(\mathcal{C})| = 20\,160$



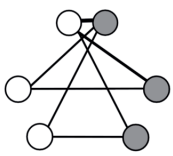
(b) Ext. Golay $= [24, 12, 8]$, with $|\mathrm{Aut}(\mathcal{C})| = 244\,823\,040$

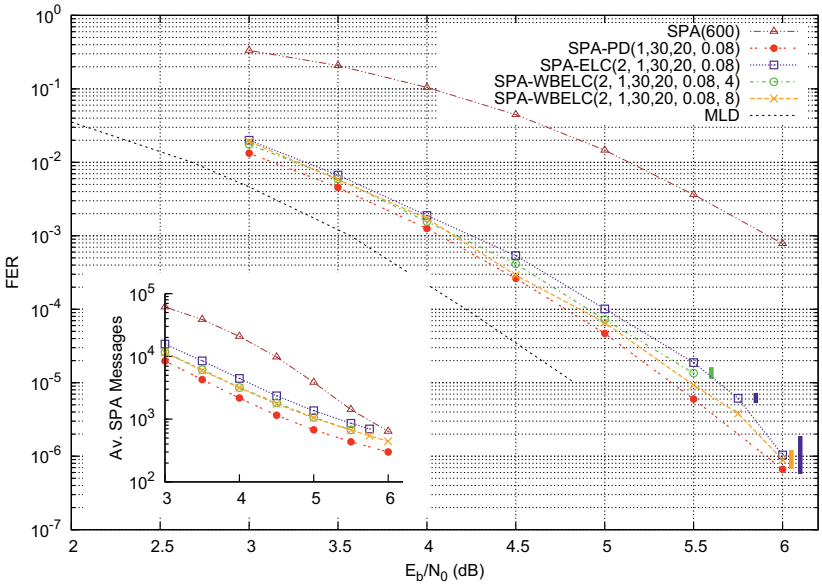


(c) $R2 = [36, 18, 8]$, with $|\mathrm{Aut}(\mathcal{C})| = 32$

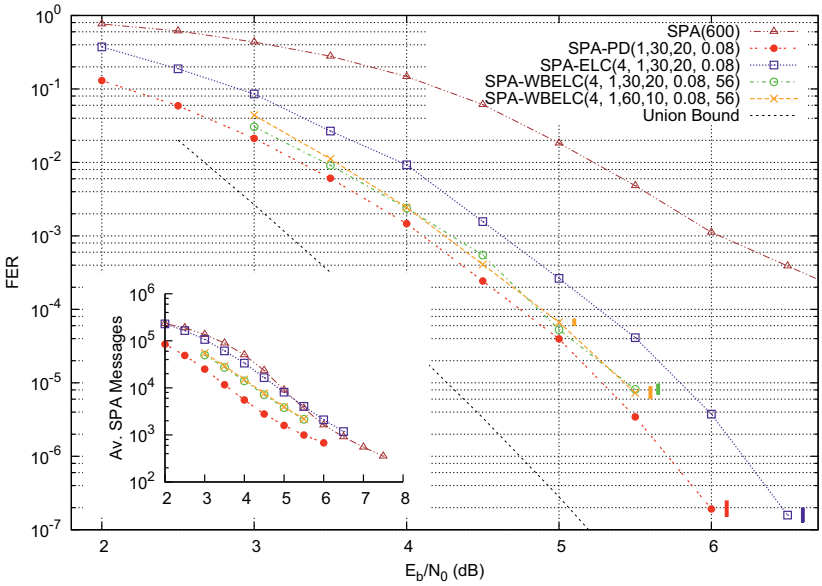


(d) $\mathcal{C}_{38,2} = [38, 19, 8]$, with $|\mathrm{Aut}(\mathcal{C})| = 1$





(e) EQR48 $= [48, 24, 12]$, with $|\mathrm{Aut}(\mathcal{C})| = 51\,888$



(f) $\mathcal{C}_{68,1} = [68, 34, 12]$, with $|\mathrm{Aut}(\mathcal{C})| = 68$

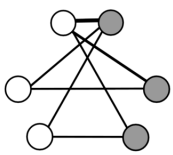
Fig. 19: *Simulation results. Each SNR point is simulated until at least 100 frame-error events were observed (otherwise, error bars indicate a 95% confidence interval [31]). The union bound is calculated based on the full weight enumerator of the code.*

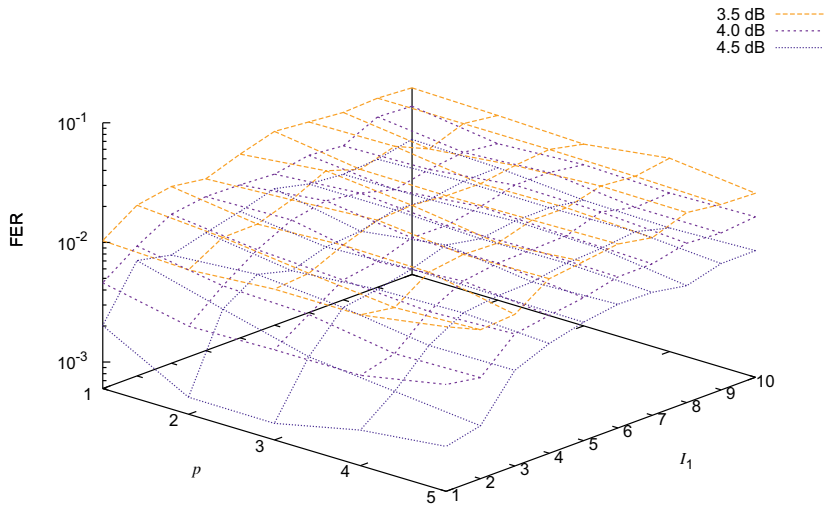
known adverse effect on convergence due to short cycles in the Tanner graph. The aim of the SPA-WBELC decoder is to use WB-ELC to give structurally distinct matrices of bounded weight. We will show that the SPA-WBELC decoder outperforms SPA-PD [13] when $\mathrm{Aut}(\mathcal{C})$ is small. For SPA-WBELC, the operation is WB-ELC, which amounts to either one or two ELC operations. For a similar degree of diversity, a budget of p ELC operations per I_1 iterations is allocated for the SPA-WBELC decoder, which is decremented by $|\mathbf{e}|$. (This means the SPA-WBELC decoder may use $p+1$ ELC operations per ELC stage, but the average is very close to p, so we ignore this.) As for SPA-ELC, optimal values of parameters are determined empirically, as shown in Fig. 23.

In Fig. 19, the performance of SPA-ELC and SPA-WBELC is simulated for various codes of different blocklength when signalling over the AWGN channel. The graphs used for decoding were optimized on weight in a preprocessing stage, as reported in Table 3. For SPA and SPA-PD, the same graph is used throughout the decoding process, and this graph is also used in a nonsystematic form for these decoders.[6] For the ELC-based decoders, the initial (preprocessed) $\mathbf{TG}(H)$ is restored at the beginning of each frame (codeword simulated). For SPA-PD [13], the operation is permutation from $\mathrm{Aut}(\mathcal{C})$, selected at random by taking random combinations of generators of $\mathrm{Aut}(\mathcal{C})$ [32]. As a reference, the performance of the optimal maximum-likelihood decoder (MLD) is simulated where this is feasible, and is otherwise approximated by a union bound using the full weight enumerator of the code.

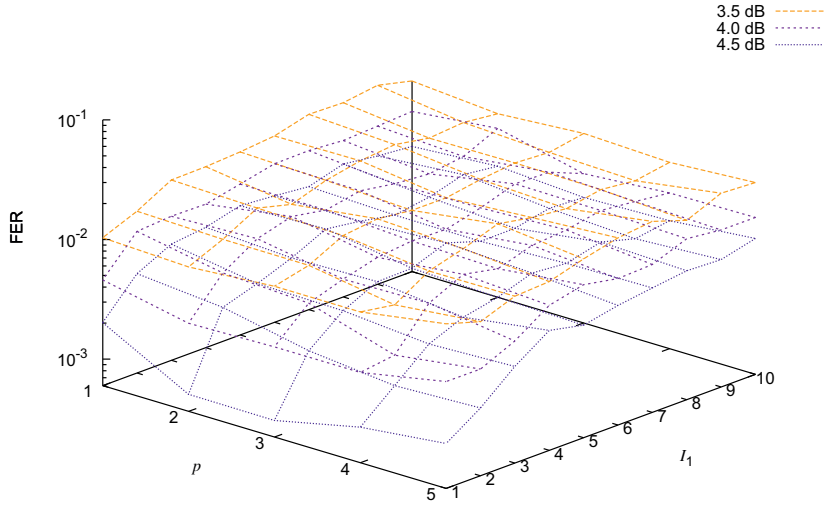
A simple scheme running SPA on seven distinct minimum-weight matrices for the extended Golay code gives an improvement over SPA [33]. We observe a performance gain of $\sim$ 0.5dB at bit-error rate 10^{-4} over this scheme (we still observe a gain of $\sim$ 0.25dB when we limit SPA-ELC to $\tau = 200$ iterations). We also observe an improvement in error-rate on this code over the more advanced multiple-bases belief propagation (MBBP) algorithm, which uses 15 $n \times n$ matrices (based on cyclic shifts of minimum-weight codewords in $\mathcal{C}^{\perp}$) in a parallel (i.e., list) decoding scheme [17]. At FER $3 \cdot 10^{-3}$ we observe a gain of $\sim$ 0.2dB when using $\tau = 600$ iterations. In addition to this improvement in performance, we also achieve a significant reduction in complexity, by avoiding parallelism, using fewer iterations (they use a maximum of 1 050 iterations), and avoiding redundant parity-check matrices.

[6] We have also simulated SPA and SPA-PD on systematic matrices (not shown), to verify that FER performance is not significantly sensitive to this.





(a) $I_3 = 20$ fixed, and $I_2 = \tau/(I_1 I_3)$.



(b) $I_2 = 30$ fixed, and $I_3 = \tau/(I_1 I_2)$.

Fig. 20: *Simultaneous determination of parameters p and I_1 for SPA-ELC decoding on code "R2." The minimum FER is for low I_1 and $p \approx 2$.*

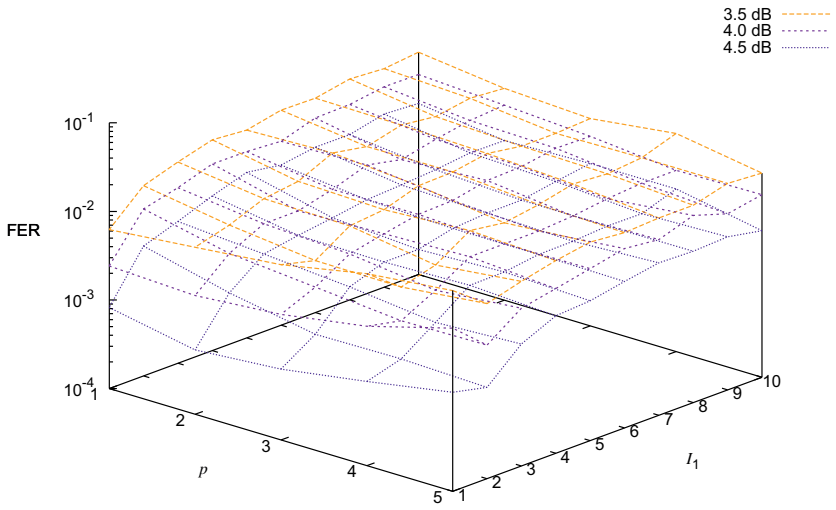
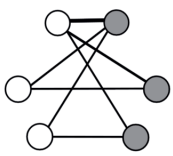


Fig. 21: *Similar response for SPA-ELC on code "EQR48" ($I_3 = 20$ fixed and $I_2 = \tau/(I_1 I_3)$).*

The overall observation is that SPA-ELC outperforms SPA for all codes, with a significant gain (over 1dB) which increases with block-length. Most interestingly, we observe that the flooring effect of SPA on the "$C_{68,1}$" code (occurring already at FER 10^{-4}) is avoided by adding random ELC operations to the decoding process. For the smaller-size codes, BCH15 and the extended Golay code, the performance of SPA-ELC coincides with that of the SPA-PD decoder. This is an interesting observation, as the SPA-PD is regarded among the state-of-the-art iterative SISO decoders for HDPC codes. However, the overhead of generating elements from $\mathrm{Aut}(\mathcal{C})$ is not trivial, which may make the random, graph-local SPA-ELC decoder an interesting alternative. For the larger codes, it is apparent that SPA-ELC cannot keep up with SPA-PD. This results from the weight increase due to random ELC, which is the main motivation for this work. When the weight is not bounded, random ELC will give graphs sampled from the orbit of the code, which is generally extremely large. However, rare exceptions exist, including the ELC-preserved codes, as well as "BCH15" and the extended Golay



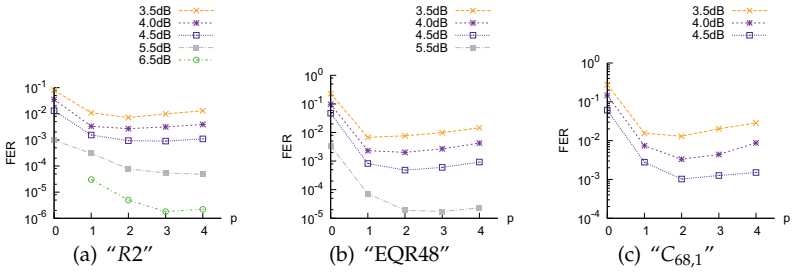


Fig. 22: *Details for SPA-ELC with $I_1 = 1$, $I_2 = 30$, and $I_3 = 20$. Here, $p = 0$ denotes SPA decoding (with no damping). p may be increased to slightly reduce flooring effect.*

code for which the orbit is of size 2. This follows from the fact that the orbits of these codes are *well behaved* for ELC decoding; they contain only "low" weight graphs – see Examples 4 and 5. Thus, these codes are nearly ELC-preserved, such that any sequence of random ELC operations will preserve the structure of *two*, rather than one, graphs. SPA-ELC on these codes can be thought of as SPA-PD, but now using two distinct graphs; in a sense, a combination of SPA-PD and MBBP.

Consider next the $R2$ code, for which $\mathrm{Aut}(\mathcal{C})$ is small, and even more importantly the "$C_{38,2}$" code, for which $\mathrm{Aut}(\mathcal{C})$ is trivial. These codes are included to report the performance of SPA-PD on codes which are arguably not the optimal choice for this algorithm. For these codes, it is seen that SPA-ELC has a gain over SPA-PD, especially in terms of a removing a floor-effect on "$R2$." The "$C_{38,2}$" code is a propos of an important class of codes, for which the structure is not so strong as to facilitate a nontrivial $\mathrm{Aut}(\mathcal{C})$. Although not considered in this work, for most random codes (such as LDPC codes) it may be expected that $\mathrm{Aut}(\mathcal{C})$ is trivial, in which case SPA-PD "reduces" to SPA (conceptually, only applying the identity permutation), giving further meaning to the aforementioned gain of SPA-ELC over SPA. To emphasize; SPA-ELC can improve SPA decoding on codes where SPA-PD cannot.

For the larger "EQR48" and "$C_{68,1}$" codes, however, $|\mathrm{Aut}(\mathcal{C})|$ *and* the orbit size are both large (in fact, the actual size of the orbit appears to be impractical to compute), so additional measures are required to maintain the gain of ELC-based decoding. Using WB-ELC, it is possible to bound the weight due to ELC, and the simulations verify the assumption that the performance of iterative SISO decoding is

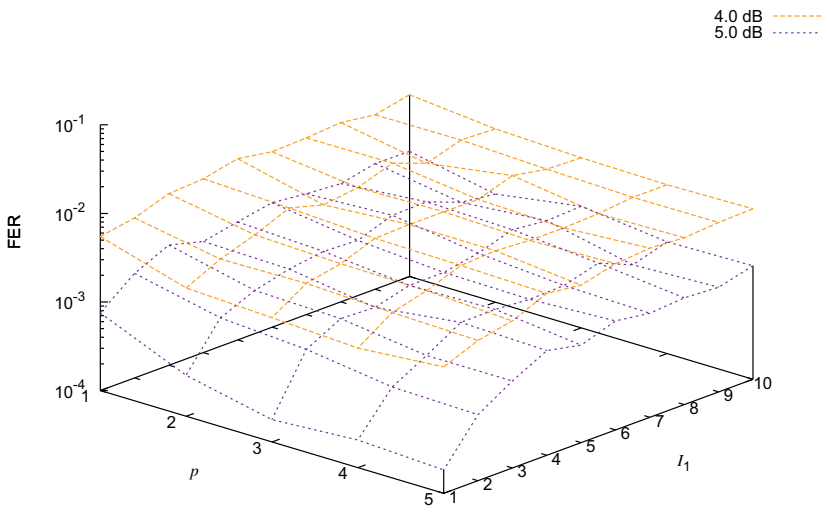
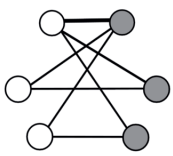


Fig. 23: *Simultaneous determination of parameters p and I_1 for SPA-WBELC decoding on code "R2." Here, $I_3 = 20$ and $I_2 = \tau/(I_1 I_3)$.*

sensitive to increased graph weight; SPA-WBELC shows a consistent gain over SPA and SPA-ELC for all codes considered, closing the gap to SPA-PD also for these largest codes. For the special "BCH15" and extended Golay code, we verify that the performance of SPA-ELC and SPA-WBELC is the same.

The choice of parameters has a large impact on SPA-WBELC performance. As discussed, a preprocessing stage is required not only to obtain a reduced-weight initial graph H_0, but also to check that the "low" weight sub-orbit of this graph (within $|H_0| + T$) is sufficiently large to provide the required amount of diversity during decoding. The SPA-WBELC decoder uses the exhaustive algorithm WB_ELC$_{(b)}$ to determine random WB-ELC operations, rather than one of the real-time versions. This algorithm does not employ heuristics (i.e., kicks), so it is important to choose an initial graph for which a sufficiently large sub-orbit is "available" during decoding. Assuming $T \geq 0$, it is always possible to go back to the previous graph by undoing the previous WB-ELC operation. The SPA-WBELC decoder will perform $pI_3I_2 = p\tau/I_1$



WB-ELC operations during decoding, so a *sufficient* amount of diversity is, arguably, $\sim p\tau/I_1$ distinct Tanner graphs. Alternatively, a speed-up could be achieved by using WB_ELC$_{(c)}$ (or even WB_ELC$_{(c,1)}$), but then at a small penalty in FER due to the number of kicks required by these real-time implementations. As the aim of this work is to report the benefits of bounding the weight due to ELC, we have not gone into detail on this and do not include a dedicated set of simulations. Using Alg. MINIP, we are able to reach (or, at least closely approach) the minimum-weight bound from (2), producing a set of candidate bounded-weight graphs. The procedure outlined in Section 4.3 gives an indication on whether the number of distinct Tanner graphs reachable via WB-ELC within threshold T is sufficient. This way, the initial graph H_0 is chosen, and the parameter T is set. In summary, using $T = 4$ or 8, diversity and weight-bounding is achieved for "$R2$" and "EQR48." All codes were simulated using WB_ELC$_{(b)}$, so, for "$C_{68,1}$," the threshold used in simulations was increased to $T = 56$ in order to reduce simulations time for this code.

The FER performance of SPA-WBELC has a deterministic response to increase in T. As shown in the simulations, by increasing T, a gain is found in the low-SNR range. Yet, for increasing SNR, the FER performance "breaks off" at some point, approaching that of SPA-ELC. As such, this should not be considered a floor-effect (as the FER resumes its initial slope), but rather an indication that the distinction between SPA-WBELC and SPA-ELC is stronger at low SNR. This may be explained by considering the general performance of SPA; at low SNR (high noise), the number of iterations is, generally, higher than that at high SNR. Thus, the number of operations (in this sense, ELC operations) is greater per frame simulated at low SNR. At high SNR, as the average number of iterations per frame approaches 1 (an initial syndrome check is not used in this work, which would otherwise give an average of 0 iterations), the result of ELC and WB-ELC is to a large extent the same, as the weight increase of SPA-ELC is limited to at most one stage consisting of p ELC operations.

5.4 Complexity Observations

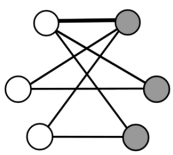
We also report simulations data on the average complexity of the various decoding algorithms. Since the SPA-ELC and SPA-WBELC decoders use a systematic matrix and modify the corresponding graph during

decoding, whereas the SPA and SPA-PD decoders use a static, nonsystematic matrix, the complexity cannot be reported simply in terms of average number of iterations per codeword. However, the complexity of all stages of SPA decoding and the ELC operation is proportional to the number of edges involved, so decoding complexity may be measured in average number of SPA messages [14, 34], counted as

$$\chi_D = \frac{1}{F} \sum_{F} \sum_{j=1}^{J \leq \tau} |H_j| \tag{19}$$

where J is the number of iterations used for a frame (which varies with SNR and the specific noise pattern), and F is the total number of frames simulated per SNR point. In terms of messages (edges processed), the complexity of one (flooding) SPA iteration is $2(|\mathcal{G}| + n - k) = 2(k\bar{\gamma} + n - k)$. For the following argument, we assume again that $k = n - k$. At the expected "50% weight" the complexity of one SPA iteration is $2k(\bar{\gamma} + 1) = 2k(k/2 + 1) = k^2 + 2k$, which is significantly higher (by at least a factor of 4) than the ELC complexity, $k^2/4 - k + 1$, from (5). As such, we do not take the overhead of applying ELC operations into account in the comparisons – especially when SPA iterations are in the log-likelihood domain, where the check node update rule involves use of the computationally heavy hyperbolic tangent rule (this is not taken into account in the analysis above).

For complexity, as defined in (19), we observe the desired effect of bounding the weight increase due to ELC. For SPA-ELC, the average weight of H quickly settles around "50% weight," i.e., $k(k+2)/2$ following from (6), whereas for SPA-WBELC, the average weight is around $|H_0| + T$ – as shown by the histograms in Figs. 15(b), 16(b), and 17(b). The inset plots, comparing number of SPA messages, in Fig. 19 indicate a general trend where the SPA-PD decoder has the lowest complexity, while the SPA is the most complex decoder. As these two algorithms use the exact same graph (for a given code), any difference must be entirely in terms of number of *iterations* used per codeword. In other words, this shows how the SPA-PD is a very important benchmark, as it gives an improvement in *both* FER and complexity. Similarily, our proposed SPA-WBELC algorithm also has an improvement in complexity (in terms of average number of SPA messages), over SPA and SPA-ELC, and is not far from the benchmark complexity of SPA-PD. The complexity improvement (over SPA-ELC) is a direct benefit from bounding the weight. However, as we have discussed, there is a significant search overhead incurred by the SPA-WBELC decoder. This is



highly implementation-dependant, and more efficient implementations (perhaps even tailored to the specific code used) may be possible.

CONCLUSION

In this work, we have presented a mapping from a Tanner graph to a simple, bipartite graph such as to facilitate the use of a graph operation known as ELC during iterative, graph-based decoding. It is known that ELC modifies locally the structure (i.e., the edges) of a graph, without changing the associated code. We have identified and described how the ELC operation – or more generally a sequence of ELC operations – may induce a graph isomorphism, and how this is linked to code automorphism, i.e. to $\mathrm{Aut}(\mathcal{C})$. From the code perspective, we have also defined a notion of Tanner graph isomorphism (row-equivalence of matrices), and shown the relationship to the corresponding trivial (in terms of decoding) subgroup of $\mathrm{Aut}(\mathcal{C})$. This gives a natural relationship with a state-of-the-art decoding algorithm for classical (HDPC) codes, SPA-PD, which improves decoding by employing random permutations from $\mathrm{Aut}(\mathcal{C})$ during decoding.

The concept of isomorphic ELC operations has been generalized to a weight-bounding application of ELC, WB-ELC, for which the effect of ELC on the weight of the graph is upper-bounded by a threshold. All possible instances of WB-ELC due to single and double application of ELC on a graph are classified, where we show that all double instances occur on adjacent edges. In this sense, WB-ELC adheres to the locality of ELC and SPA, which, in turn, simplifies the implementation and complexity of an algorithm to enumerate all WB-ELC operations on a graph, within a threshold value. The complexity of such an algorithm is analyzed theoretically, and verified empirically by simulations on a set of different graphs (corresponding to typical HDPC codes of different blocklength).

Several applications of WB-ELC are suggested, which all relate to the context of graph-based decoding on a Tanner graph of a HDPC code. First of all, a set of reduced-weight (Tanner) graphs may be produced using WB-ELC with a negative threshold, from which a graph suitable for decoding is chosen in terms of having a large bounded-weight sub-orbit, to within some threshold. To facilitate the use in decoding, various heuristics are proposed to devise a real-time, reduced-complexity version of the enumeration algorithm. Again, simulations are used to assess the benefits of these heuristics on the same set of

graphs (HDPC codes). The resulting operation is finally used to describe a SPA-WBELC decoder.

A generalized framework for SISO decoding of HDPC codes is proposed, based on the SPA-PD algorithm. By abstracting the operation used to gain diversity in between SPA iterations, various decoding algorithms may be implemented in this common framework. This ensures a fair comparison in simulations results, which are presented both in terms of FER and complexity (in terms of number of SPA messages computed per codeword). In this context, we also describe a novel edge-local damping rule, which is suitable in the local context of ELC-based decoding. In total, extensive simulations data show a consistent gain of SPA-ELC and SPA-WBELC over SPA, and where SPA-WBELC approaches closely the performance of SPA-PD. Furthermore, we emphasize types of graphs (or, codes) well-suited for SPA-WBELC, but for which SPA-PD can *not* be used.

ACKNOWLEDGMENT

The authors wish to thank Alban Goupil for providing the MLD curve for the "EQR48" code.

APPENDIX

A PROOF OF LEMMA 2, ELC ON ADJACENT EDGES

Proof. Let $A \not\sim B$ denote the complementation of the edges between nodes in A and B, where double complementation cancels out: $A \not\sim B = A \sim B$. Define the sets $A = \mathcal{N}_v^u \setminus \mathcal{N}_{v'}^u, B = \mathcal{N}_v^u \cap \mathcal{N}_{v'}^u, C = \mathcal{N}_{v'}^u \setminus \mathcal{N}_v^u$, and $D = \mathcal{N}_u^{v,v'}$. ELC on two adjacent edges, $\{(u,v),(v,v')\}$ gives the same graph as ELC directly on (u,v'). Consider the initial graph, $\mathcal{G}$, consisting of the following components

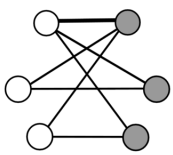
$$\mathcal{G} = (u,v),(u,v'),(u,D),(v,A\cup B),(v',B\cup C),(A\cup B\cup C)\sim D.$$

We may then denote the graph after ELC on (u,v) as $\mathcal{G}_{(u,v)}$, for any edge

$$\mathcal{G}_{(u,v)} = (v,u),(v,v'),(v,D),(u,A\cup B),(v',A\cup C),(A\cup B)\not\sim D, C\sim D$$

and

$$\mathcal{G}_{\{(u,v),(v,v')\}} = (v',u),(v',v),(v',D),(u,B\cup C),(v,A\cup C),A\not\sim D,(B\cup C)\not\sim D$$



where double complementation cancels out, e.g., $A \not\sim D = A \sim D$. It is then readily seen that

$$\begin{aligned}\mathcal{G}_{(u,v')} &= (v',v),(v',u),(v',D),(v,A\cup C),(u,B\cup C),A\sim D,(B\cup C)\not\sim D\\ &= \mathcal{G}_{\{(u,v),(v,v')\}}.\end{aligned}$$

Note that, due to the swap in the first ELC on (u,v), we have that (v,v') and (u,v') refer to the same edge, before and after ELC on (u,v). See also Fig. 4. □

This proof can be extended to nonbipartite graphs, although this is outside the scope of this paper.

B Implementation Notes for WB-ELC Algorithm

To facilitate an efficient reuse of sets, slight modifications are made to the counting formulas. Theorem 6 considers pairs of edges adjacent at distance one, so, given (u,v) from Theorem 3, we can reuse the sets $\mathcal{N}_v^u$ and $\mathcal{N}_u^v$ and $|\mathcal{E}_{u,v}|$ for all depth-2 instances rooted in the edge (u,v).

For Theorem 6, v' is picked from $\mathcal{N}_u^v$. Then, for all possibilities of $u' \in A = \mathcal{N}_v^u \setminus \mathcal{N}_{v'}^u$, it is possible to reuse the left-hand sets A, B, and C for all instances of Theorem 6. By choosing $u' \in A$ rather than $\mathcal{N}_v^u$, we know that $(u',v') \notin \mathcal{G}$. Thus, we know that Theorem 6 applies, and, since $A \subseteq \mathcal{N}_v^u$, we generally save some search time. With $W = E \cup F = \mathcal{N}_{u'}^v$, we get the following modifications to (10)

$$\begin{aligned}u' \in A \subset X = \mathcal{N}_v^u \;\Rightarrow\; & |A| \to |A|-1,\ |\mathcal{E}_{A,E\cup F}| \to |\mathcal{E}_{A,W}| - |W|\\ v' \in D \subset Y = \mathcal{N}_u^v \;\Rightarrow\; & |D| \to |D|-1,\ |\mathcal{E}_{B,D\cup E}| \to |\mathcal{E}_{B,Y}| - |B|\\ & |\mathcal{E}_{C,D\cup F}| \to |\mathcal{E}_{C,D\cup F}| - |C|.\end{aligned}$$

From Theorem 4 we know that u'' must be selected among nodes within distance 2 from v, which gives a significant reduction in search space. This means that, for Theorem 5, we choose $u'' \in C = \mathcal{N}_{v'}^u \setminus \mathcal{N}_v^u$, since the node must be at distance two from (u,v). By choosing $v'' \in \mathcal{N}_{u''} \setminus \mathcal{N}_u^v$, we can reuse $W' = \mathcal{N}_{u''}$. We also reuse the sets corresponding to the root edge, (u,v) (namely, Y and X). For all valid choices of v'', $W' = \mathcal{N}_{u''}^{v''} \cup \{v''\}$. This means that $|\mathcal{N}_{u''}^{v''}| = |W'| - 1$ is invariant (does not need to be recomputed). Furthermore, as $v'' \in W'$ is connected to $X' = \mathcal{N}_{v''}^{u''}$, we have that $|\mathcal{E}_{X',\mathcal{N}_{v''}^{u''}}| = |\mathcal{E}_{X',W'}| - |X'|$.

For Theorem 5, we need to do a little more bookkeeping to avoid repetitions. First of all, a simple check, $u'' > u$, is to avoid combinations of

ELC that give the same graph, i.e., $\{(u,v),(u'',v'')\} = \{(u'',v''),(u,v)\}$. However, additional steps need to be taken to avoid repetitions. Consider the situation when we arrive at Theorem 5: The edge (u,v) is reused from Theorem 3, and v' is reused from Theorem 6 (indirectly, in the use of C). Now we can check Theorem 5, $\{(u,v)(u'',v'')\}$, for all possibilities of u'', v''. However, for different choices of v', there may be an overlap in the resulting sets C, resulting in repeated enumeration of WB-ELC according to Theorem 5. Such challenges arise as a consequence of nested evaluation of the theorems, and a quick solution to the problem is to simply keep track of the 'used' edges (u'', v'') in a set S, and avoid checking these repeatedly.

REFERENCES

[1] SHANNON, C. E.: A mathematical theory of communication. *Bell System Tech. J.* 27, 379–423, 623–656, Jul. and Oct. 1948.

[2] BERROUX, C., GLAVIEUX, A., THITIMAJSHIMA, P.: Near Shannon limit error-correcting coding and decoding: Turbo codes. In *Proc. IEEE Int. Conf. Commun.*, pp. 1064–1070. Geneva, Switzerland, May 1993.

[3] MACKAY, D. J. C.: Good error-correcting codes based on very sparse matrices. *IEEE Trans. Inform. Theory* 45(2), 399–431, Mar. 1999.

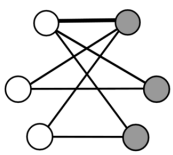
[4] GALLAGER, R. G.: Low-density parity-check codes. *IRE Trans. Inform. Theory* 8(1), 21–28, Jan. 1962.

[5] TANNER, R. M.: A recursive approach to low complexity codes. *IEEE Trans. Inform. Theory* 27(5), 533–547, Sep. 1981.

[6] HALFORD, T. R., GRANT, A. J.: Which codes have 4-cycle-free Tanner graphs. *IEEE Trans. Inform. Theory* 52(9), 4219–4223, Sep. 2006.

[7] YEDIDIA, J. S., CHEN, J., FOSSORIER, M. P. C.: Generating code representations suitable for belief propagation decoding. In *Proc. 40th Allerton Conf. Commun., Contr. and Comp.*, pp. 447–456. Monticello, IL, Oct. 2002.

[8] FOSSORIER, M. P. C.: Reliability-based soft-decision decoding with iterative information set reduction. *IEEE Trans. Inform. Theory* 48(12), 3101–3106, Dec. 2002.



[9] Kothiyal, A., Takeshita, O. Y., Jin, W., Fossorier, M. P. C.: Iterative reliability-based decoding of linear block codes with adaptive belief propagation. *IEEE Commun. Lett.* 9(12), 1067–1069, Dec. 2005.

[10] Jiang, J., Narayanan, K. R.: Iterative soft-input soft-output decoding of Reed-Solomon codes by adapting the parity-check matrix. *IEEE Trans. Inform. Theory* 52(8), 3746–3756, Aug. 2006.

[11] Jin, W., Fossorier, M. P. C.: Reliability-based soft-decision decoding with multiple biases. *IEEE Trans. Inform. Theory* 53(1), 105–120, Jan. 2007.

[12] Jiang, J., Narayanan, K. R.: Iterative soft decoding of Reed-Solomon codes. *IEEE Commun. Lett.* 8(4), 244–246, Apr. 2004.

[13] Halford, T. R., Chugg, K. M.: Random redundant iterative soft-in soft-out decoding. *IEEE Trans. Commun.* 56(4), 513–517, Apr. 2008.

[14] Dimnik, I., Be'ery, Y.: Improved random redundant iterative HDPC decoding. *IEEE Trans. Commun.* 57(7), 1982–1985, Jul. 2009.

[15] Knudsen, J. G., Riera, C., Danielsen, L. E., Parker, M. G., Rosnes, E.: Iterative decoding on multiple Tanner graphs using random edge local complementation. In *Proc. IEEE Int. Symp. Inform. Theory*, pp. 899–903. Seoul, Korea, Jun./Jul. 2009.

[16] Knudsen, J. G., Riera, C., Parker, M. G., Rosnes, E.: Adaptive soft-decision decoding using edge local complementation. In *Proc. Second Int. Castle Meeting on Coding Theory and Applications, LNCS 5228*, pp. 82–94. Castillo de la Mota, Medina del Campo, Spain, Sep. 2008.

[17] Hehn, T., Huber, J. B., Laendner, S., Milenkovic, O.: Multiple-bases belief-propagation for decoding of short block codes. In *Proc. IEEE Int. Symp. Inform. Theory*, pp. 311–315. Nice, France, Jun. 2007.

[18] Hehn, T., Huber, J., Milenkovic, O., Laendner, S.: Multiple-bases belief-propagation decoding of high-density cyclic codes. *IEEE Trans. Commun.* 58(1), 1–8, Jan. 2010.

[19] Bouchet, A.: Isotropic systems. *European J. Comb.* 8, 231–244, Jul. 1987.

[20] Danielsen, L. E., Parker, M. G.: Edge local complementation and equivalence of binary linear codes. *Des. Codes Cryptogr.* 49(1-3), 161–170, Dec. 2008.

[21] Knudsen, J. G.: *Randomised Construction and Dynamic Decoding of LDPC Codes*. Master's thesis, University of Bergen, Bergen, Norway, 2006.

[22] Knudsen, J. G., Riera, C., Danielsen, L. E., Parker, M. G., Rosnes, E.: Improved adaptive belief propagation decoding using edge-local complementation. In *Proc. IEEE Int. Symp. Inform. Theory*, pp. 774–778. Austin, Texas, Jul. 2010.

[23] Huffman, W. C.: Handbook of coding theory. In Pless, V. S., Huffman, W. C. (eds.), *Handbook of Coding Theory*. Elsevier, North-Holland, Amsterdam, 1998.

[24] McKay, B. D.: Nauty; software for computing automorphism groups of graphs and digraphs. Web page, 2007. `http://cs.anu.edu.au/people/bdm/nauty/`.

[25] Danielsen, L. E., Parker, M. G., Riera, C., Knudsen, J. G.: On graphs and codes preserved by edge local complementation, 2010. arXiv:1006.5802.

[26] Brijder, R., Harju, T., Hoogeboom, H. J.: Pivots, determinants, and perfect matchings of graphs, 2008. arXiv:0811.3500.

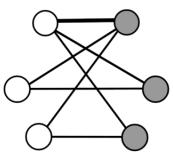
[27] Arratia, R., Bollobás, B., Sorkin, G. B.: The interlace polynomial of a graph. *J. Comb. Theory, Series B* 92(2), 199–233, Nov. 2004.

[28] Harada, M.: New extremal self-dual codes of lengths 36 and 38. *IEEE Trans. Inform. Theory* 45(7), 2541–2543, Nov. 1999.

[29] Gulliver, T. A., Harada, M.: Classification of extremal double circulant self-dual codes of lengths 64 to 72. *Des. Codes Cryptogr.* 13(3), 257–269, Mar. 1998.

[30] Shewchuk, J. R.: An introduction to the conjugate gradient method without the agonizing pain. Tech. rep.: CS-94-125, Carnegie Mellon University, Pittsburgh, PA, 1994.

[31] MacKay, D. J. C.: *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.



[32] Celler, F., Leedham-Green, C. R., Murray, S. H., Niemeyer, A. C., O'Brien, E. A.: Generating random elements of a finite group. *Commun. in Algebra* 23, 4931–4948, 1995.

[33] Andrews, K., Dolinar, S., Pollara, F.: LDPC decoding using multiple representations. In *Proc. IEEE Int. Symp. Inform. Theory*, p. 456. Lausanne, Switzerland, Jun./Jul. 2002.

[34] Fossorier, M. P. C., Mihaljevic, M., Imai, H.: Reduced complexity iterative decoding of low-density parity check codes based on belief propagation. *IEEE Trans. Commun.* 47(5), 673–680, May 1999.

PAPER IV

IMPROVED ADAPTIVE BELIEF PROPAGATION DECODING USING EDGE-LOCAL COMPLEMENTATION *

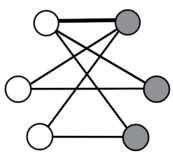
Joakim Grahl Knudsen Constanza Riera

Lars Eirik Danielsen Matthew G. Parker

Eirik Rosnes

*Knudsen, J. G., Riera, C., Danielsen, L. E., Parker, M. G., Rosnes, E.: Improved adaptive belief propagation decoding using edge-local complementation. In *Proc. IEEE Int. Symp. Inform. Theory*, pp. 774–778. Austin, Texas, Jul. 2010.

The notation in this chapter has been edited to make it consistent with the thesis. A misprint has been corrected in formula (3).



IMPROVED ADAPTIVE BELIEF PROPAGATION DECODING USING EDGE-LOCAL COMPLEMENTATION

Joakim Grahl Knudsen Constanza Riera

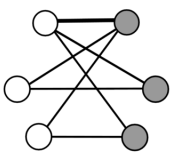
Lars Eirik Danielsen Matthew G. Parker

Eirik Rosnes

This work is an extension of our previous work on an iterative soft-decision decoder for high-density parity-check codes, using a graph-local operation known as edge-local complementation (ELC). Inferred least reliable codeword positions are targeted by an ELC stage in between sum-product algorithm iterations. A gain is shown over related iterative decoding algorithms – mainly due to an improved heuristic to determine optimum ELC locations in the Tanner graph – both in error-rate performance, as well as complexity in terms of a significant reduction in the required number of ELC operations. We also present a novel damping operation, generalized to the graph-local setting where extrinsic information remains on edges not affected by ELC.

1 INTRODUCTION

Iterative soft-input soft-output (SISO) decoding of graph-based codes has been shown to give near-optimum results, when the sum-product algorithm (SPA) is used on low-density parity-check codes. Recently, these results have been extended to high-density parity-check (HDPC)



codes, in order to facilitate the use of well-known strong families of codes, such as Bose-Chaudhuri-Hocquenghem [1, 2], quadratic residue (QR) [3], and Reed-Solomon (RS) codes [4–6]. Constructions of these types have strong structural properties (most importantly, non-trivial automorphism group and large minimum distance), as well as convenient algebraic descriptions to simplify hardware implementation. We have previously described a graph-local operation known as edge-local complementation (ELC) and its applications to improve the performance of SPA decoding by providing diversity during decoding. In addition to random application of a small number of ELC operations [3], we have considered the controlled application of ELC such as to preserve graph isomorphism, or to maintain a bound on graph weight [7]. We have previously compared against iterative permutation decoding (SPA-PD), in which permutations from the automorphism group of the code are used to gain diversity [1]. This work is an extension of our work on ELC-based SISO HDPC decoding, where the aim is now to target inferred error positions during decoding. In addition to improved diversity, the structural effects of ELC on the Tanner graph are such that affected positions are set in a listening state (degree-1 nodes), such that they may continue to converge without influencing other nodes (e.g., via cycles). The adaptive belief propagation decoder from [5], denoted by ABP, uses Gaussian elimination (GE) on the $(n-k) \times n$ parity-check matrix H, in an attempt to reduce the columns corresponding to the $n-k$ inferred least reliable parity positions to an identity matrix. We will show how the ELC operation is related to GE. The novelty of the proposed ABP-ELC decoder lies in the significant reduction in the number of positions (columns of H) affected by the adaptive stage, while simultaneously achieving a gain in error-rate performance over SPA-PD and ABP. We also maintain the locality argument of the ELC operation [3], which leads to several modifications to improve the performance of the ABP-ELC heuristic (i.e., where to apply the ELC operations).

We use boldface notation for vectors, and italics uppercase for matrices. A binary linear code $\mathcal{C}$ of length n, dimension k, and minimum distance $d_{\mathbf{min}}$ is denoted by $[n, k, d_{\mathbf{min}}]$. H is said to be systematic if the columns can be reordered into the form $[I\, P]$, where I is the identity matrix of size $n-k$. The corresponding codeword positions comprise a parity set $\mathcal{P}$ and an information set $\mathcal{I}$, respectively, referring to the $k \times n$ generator matrix $[P^T\, I]$. The single non-zero entry of a systematic column is referred to as a *pivotal.* Unless stated otherwise, codes discussed in this paper are represented by H in systematic form. The

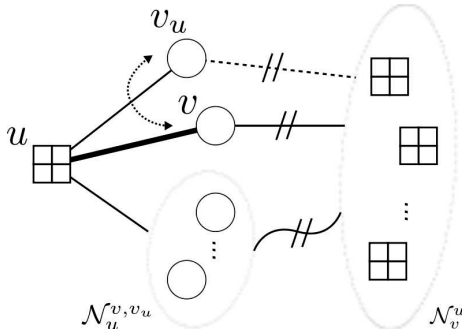
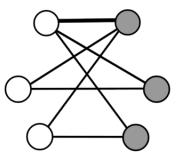


Fig. 1: *ELC on edge (u, v) of a (systematic) Tanner graph, where v_u is the systematic node for node u. Straight links between two sets mean that these are completely connected, while curved links mean arbitrary connections. Dashed lines indicate non-edges. Doubly slashed links are complemented (edges are replaced by non-edges, and vice versa) resulting in v and v_u swapping connections. This graph may be a subgraph of a larger graph.*

local neighborhood of a node v is the set of nodes adjacent to v, and is denoted by $\mathcal{N}_v$, while $\mathcal{N}_v^u$ is shorthand for $\mathcal{N}_v \setminus \{u\}$. We refer to variable node v_i as simply v when the index is obvious from the context, and use the shorthand notation $v \in \mathcal{P}$ for a node v_i where $i \in \mathcal{P}$.

2 ELC ON A TANNER GRAPH

ELC is defined on a simple graph, $G = \begin{pmatrix} 0 & P \\ P^T & 0 \end{pmatrix}$, corresponding to a parity-check matrix $H = [I \mid P]$, or, equivalently, a Tanner graph $\mathbf{TG}(H) = \begin{pmatrix} 0 & H \\ H^T & 0 \end{pmatrix}$. We have previously described ELC on $\mathbf{TG}(H)$ by going via G [3]. We now discuss the implementation of ELC as directly applied to $\mathbf{TG}(H)$. ELC requires that H is systematic, and since it is natural to assume there are no repeated columns, each row u has a unique pivotal. Let this single node adjacent to u be denoted by $v_u \in \mathcal{P}$. The implementation of ELC on an edge $(u, v) \in \mathbf{TG}(H)$ is to complement the edges connecting $\mathcal{N}_u$ and $\mathcal{N}_v^u$, as shown in Fig. 1. By definition, v is adjacent to all nodes in $\mathcal{N}_v$, whereas the systematic node $v_u \in \mathcal{N}_u$ is not connected to $\mathcal{N}_v^u$. Thus, the complementation entails the effect of swapping the connections of these nodes, or, equally, the corresponding two columns in H. Since $v_u \in \mathcal{P}$ and $v \in \mathcal{I}$, positions v and v_u are swapped between $\mathcal{I}$ and $\mathcal{P}$, changing the bipartition. Note



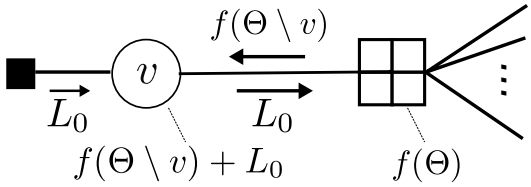


Fig. 2: *The SPA update of a systematic variable node v.*

that ELC on (u, v_u) has no effect, as $\mathcal{N}_{v_u}^{u} = \emptyset$. A maximum of $n - k$ *independent* ELC operations may be applied to $\mathbf{TG}(H)$, one per row (see *cancellation issues* in Section 3). It is readily verified that ELC is a graph implementation of the row-additions performed to reduce a column to systematic form during GE. As such, we may define GE (on a systematic matrix) as a sequence of $n - k$ ELC operations. In fact, it has been shown that all information sets of the code are found via ELC [8].

3 Adaptive Belief Propagation

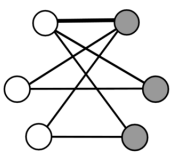
The idea of incorporating the inferring and moving of errors into solvable positions as part of a decoding algorithm, was suggested by MacWilliams and Sloane for their algebraic permutation decoder (PD) [9]. PD has recently been extended to an iterative algorithm [1], SPA-PD, so as to further benefit from SISO decoding. In a similar fashion, ABP attempts to produce an identity submatrix in the columns corresponding to error positions, by means of GE on H [5]. The GE operation affects individual columns independently, and can be targeted directly to the desired $n - k$ positions, whereas a permutation generally affects all n positions of the codeword.

Over an additive white Gaussian noise (AWGN) channel, the error positions are obviously unknown to the receiver. However, simple heuristics exist to infer the reliability at a codeword position. The received noisy vector is $\mathbf{y} = (-1)^{\mathbf{x}} + \mathbf{e}$, where $\mathbf{x}$ is a codeword and $\mathbf{e}$ is AWGN. In the log-likelihood ratio (LLR) domain, the initial LLR at position v is $L_0^v \triangleq \frac{2}{\eta^2} y_v$, where η is the standard deviation of the AWGN. The magnitude $|L_j^v|$ of the LLR L_j^v in iteration j (see (1)) serves as a measure on the reliability of position v. The ABP algorithm begins by sorting the n codeword positions according to increasing reliability

(we assume an unique ordering always exists, since the output alphabet of the AWGN channel has infinite support). Let the corresponding permutation be σ, such that $\sigma_j = i$, $0 \leq j, i < n$, if i is the jth least reliable position, and initialize a counter $\delta = 0$. Then, for each row $0 \leq j - \delta < n - k$ of H, a pivotal is attempted made in column h_{σ_j} at row index $j - \delta$. The (row) coordinate of a pivotal in a column is not important for SPA decoding, yet an important aspect of ABP is to avoid cancellations where a second pivotal is made in the same row, thus replacing the initial pivotal. As σ is processed from left to right, any such cancellation must have a counter-productive effect on performance (this is easily verified by simulations). To avoid this, ABP processes the rows in an ordered fashion. If h_{σ_j} is zero in row entry $j - \delta$, then let $j' > j - \delta$ be the first non-zero row entry (if any) in h_{σ_j}. Row j' is then added onto row $j - \delta$, such that a pivotal may now be made here. As such, the GE stage performs redundant work when position σ_j is already in $\mathcal{P}$. Only when h_{σ_j} is zero in all coordinates $j' \geq j - \delta$ will ABP skip to the next position, σ_{j+1}, and increase δ by 1. The GE stage ends after $j - \delta = n - k$ pivotals have been made, which is always possible since H is of full rank.

3.1 Isolating Weak Positions

The presence of weight-1 columns in H has a significant impact on the flow of messages in SPA decoding. As illustrated in Fig. 2, the Tanner graph equivalent of a weight-1 column is a variable node of degree 1, not counting the Forney-style input half-edge, L_0. This node is minimally connected to $\mathbf{TG}(H)$, and is not part of any cycles. The SPA update rules adhere to an extrinsic principle, in which the message passed out on any edge is independent of the incoming message along that same edge. One iteration of the flooding schedule consists of the execution of an SPA rule for all variable nodes, followed by executing all check nodes. The SPA rule for a check node is the parity function (XOR) of its incoming messages, which we denote by $f(\Theta)$. Thus, the message to v is $f(\Theta \setminus v)$, providing v with updated information from the rest of the graph. For a variable node in the LLR domain, the update rule is summation. When v is systematic, the updated soft value is $L = f(\Theta \setminus v) + L_0$, and the node can only relay its input message, the channel value $L - f(\Theta \setminus v) = L_0$, to its single adjacent check node. As such, v may be said to be in a listening or passive state. If this position is



unreliable, it will not disturb the rest of the graph, while still receiving information such that L may converge.

Still, the gain of ABP can not be accredited to this isolation effect alone. Specifically, it is known that modifying the structure of $\mathbf{TG}(H)$ during SPA decoding may, in itself, improve performance [1–7, 10, 11]. Such diversity will change the structure of cycles and node-degree distributions, and, generally, alter the flow of messages during SPA decoding. After the GE stage, the new $\mathbf{TG}(H)$ is initialized for the next SPA iteration, by damping each variable node. Damping involves scaling down the extrinsic contribution to the LLR by a damping coefficient, $0 < \alpha < 1$. This is then accumulated on the input to the next iteration,

$$L_{j+1}^{v} = L_{j}^{v} + \alpha \Gamma_{j}^{v}, \tag{1}$$

where $\Gamma_{j}^{v} = \Sigma_{u \in \mathcal{N}_v} \mu_{j}^{v \leftarrow u}$ is the extrinsic contribution to variable node v (the sum of all incoming messages, $\mu_{j}^{v \leftarrow u}$) in iteration j, and $\Gamma_{0}^{v} \triangleq 0$. The global damping stage (GD) applies (1) to all variable nodes in $\mathbf{TG}(H)$, followed by discarding all current extrinsic information; setting $\mu_{j}^{v \leftarrow u} = 0$ for all $v \in \mathbf{TG}(H)$. The next flooding iteration is initialized based on $\mathbf{L}_{j+1}$ only. This slightly complicates the argument of isolating a variable node, as the input to v is no longer fixed to the channel value, L_{0}^{v}. Still, as ABP uses a constant damping coefficient for each iteration, the accumulation may be expressed as $L_{j+1}^{v} = L_{0}^{v} + \alpha \Sigma_{j'=1}^{j} \Gamma_{j'}^{v}$. Thus, damping never affects the channel value, and we see that the accumulation is negligible when α is small, such that the isolation argument still holds.

4 ABP-ELC

Let us initially describe the proposed ABP-ELC algorithm as SPA iterations interspersed with a novel ELC stage acting on a number $0 < p \leq n - k$ of the least reliable information positions. From the graph-local ELC perspective, however, certain distinctions from the ABP algorithm arise naturally.

The first distinction is that SPA-ELC decoding has been shown to be effective for $p \ll n - k$ [3], meaning a reduction in complexity – the major concern with ABP. Some further distinctions arise from the description of ELC. ELC requires that H is in systematic form, just as the GE stage of ABP will immediately reduce any H to systematic form. As ELC on a systematic position has no effect, we may immediately

Example 1 (ELC stage)**.** *Consider the extended Hamming* $[8, 4, 4]$ *code,* H*, and some decoder state (vector of LLRs),* $\mathbf{L}$*. The actual values are not important for an example, so we focus directly on the resulting permutation* σ*. The bipartition is indicated (in* σ*) by underlining the indices of positions in* $\mathcal{I}$*. The current position (column) to consider for ELC is indicated by a star symbol over* H*. This ELC stage ends after two ELC operations.*

$$
\begin{array}{c}
\\
H = \\
\\
\sigma =
\end{array}
\begin{array}{c}
\begin{array}{cccccccc} & & & & & \star & & \end{array} \\
\begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\
\begin{bmatrix}
\mathbf{1} & 0 & 0 & 0 & 1 & \mathbf{1} & 1 & 0 \\
0 & \mathbf{1} & 0 & 0 & 1 & \mathbf{1} & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & \mathbf{1} & 0 & \mathbf{1} & 1 & 1
\end{bmatrix} \\
\begin{pmatrix} \underline{5} & 1 & \underline{4} & 3 & 0 & \underline{7} & 2 & \underline{6} \end{pmatrix}
\end{array}
$$

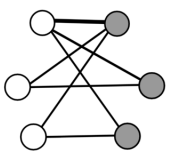
Initial matrix gives $u^\star = u_0$*.*

$$
\begin{array}{c}
\\
H = \\
\\
\sigma =
\end{array}
\begin{array}{c}
\begin{array}{cccccccc} & & & & \star & & & \end{array} \\
\begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\
\begin{bmatrix}
1 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & 1 & 1 \\
1 & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 & 1
\end{bmatrix} \\
\begin{pmatrix} 5 & 1 & \underline{4} & 3 & \underline{0} & \underline{7} & 2 & \underline{6} \end{pmatrix}
\end{array}
$$

After ELC(u_0, v_5)*, we get* $u^\star = u_2$*.Note that* $v_1 \in \mathcal{P}$ *is skipped.*

$$
\begin{array}{c}
\\
H = \\
\\
\sigma =
\end{array}
\begin{array}{c}
\begin{array}{cccccccc} \star & & & & & & & \end{array} \\
\begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\
\begin{bmatrix}
\mathbf{1} & 0 & 1 & 0 & 0 & \mathbf{1} & 0 & 1 \\
\mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\
\mathbf{1} & 0 & 1 & \mathbf{1} & 0 & 0 & 1 & 0
\end{bmatrix} \\
\begin{pmatrix} 5 & 1 & 4 & 3 & \underline{0} & \underline{7} & \underline{2} & \underline{6} \end{pmatrix}
\end{array}
$$

After ELC(u_2, v_4)*,* $u^\star = \emptyset$*, and the procedure ends.*



skip any position already in $\mathcal{P}$ while processing σ. The cancellation problem is handled by flagging the check node, u, as *used*, such that it will not be considered again in this ELC stage.

4.1 Improved Heuristic to Select ELC Positions

Consider the implicit swap effect involved in ELC on $\mathbf{TG}(H)$. Extending the argument made for ABP – namely, to move (isolate) the least reliable positions into $\mathcal{P}$ – it is equally reasonable to ensure also the converse; that the positions moved into $\mathcal{I}$ are the most reliable positions. The procedure is simple and graph-local. Given a position v, rather than choosing arbitrarily among the unused adjacent check nodes $u \in \mathcal{N}_v$, the ELC stage chooses the check node $u^\star \in \mathcal{N}_v$ for which the swap is with the most reliable position in $\mathcal{P}$ adjacent to any $u \in \mathcal{N}_v$, i.e.,

$$u^\star = \underset{u \in \mathcal{N}_v,\ |v_u| > |v|}{\arg\max} |v_u|, \tag{2}$$

where $|v|$ is shorthand for $|L^v|$. The ABP-ELC(p) algorithm processes the p first information positions in σ, applying (2) to determine the ELC locations. In the event where $u^\star = \emptyset$, no ELC is possible for this position, and ABP-ELC moves on to the next-worst information position. No additional measures are needed to avoid the cancellation problem. The algorithm simply works from both ends of σ, pairing the weakest information position with the strongest parity position. The resulting ELC induces a swap of the corresponding columns across the bipartition, and requires that we keep track of the bipartition.

Proposition 1. *Although variable nodes share the same check nodes, the proposed heuristic will never repeatedly choose the same check node $u^\star$ within an ELC stage.*

Proof. Consider two positions, v and w, which are both adjacent to u. Without loss of generality, say $|v| < |w|$, so we first consider v. Then, assume (2) gives $u^\star = u$, and we perform ELC on (u, v) such that v_u becomes non-systematic, and the systematic node of u is now v. When we later consider w, choosing the same $u^\star = u$ would entail a swap of w and v across the bipartition, thus cancelling the previous swap. However, this choice of $u^\star$ is not possible since $|v| < |w|$. □

Example 1 shows a case, where $v = v_5$ and $w = v_4$, both adjacent to $u = u_0$ (row 0). When considering v_4, we can not choose again $u^\star = u_0$, since $v_{u_0} = v_5$ and $|v_5| < |v_4|$.

As a final issue, consider the situation when we reach a position $v_{u\star}$ which has already been involved in a previous swap with some other position v. This means that $v_{u\star}$ has been moved from $\mathcal{P}$ to $\mathcal{I}$ within this ELC stage. The proposed scheme will never do a second ELC on an edge adjacent to this node, which would cancel the previous swap.

Proposition 2. *A position swapped from $\mathcal{P}$ to $\mathcal{I}$ will not be subject to any subsequent ELC, within the same ELC stage.*

Proof. Say the previous ELC was on (u, v), which swapped v_u into $\mathcal{I}$. If the heuristic later reaches this same $w = v_u$ (before exhausting p), no subsequent ELC is possible for this position, according to (2). In the initial ELC, v_u was the most reliable parity position adjacent to any $\tilde{u} \in \mathcal{N}_v$ such that $|v_{\tilde{u}}| > |v|$. However, this now means that $|w| > |v_{u'}| \; \forall \, u' \in \mathcal{N}_w$, since $w = v_u$, and (2) yields $\emptyset$. □

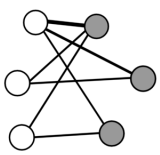
Again, Example 1 shows a case in the last stage, when $v = v_0$ which was swapped in the first stage. We have now covered all the possible cancellation issues, without any extra bookkeeping or complexity in the ABP-ELC heuristic (apart from keeping track of the bipartition).

4.2 Local-Neighborhood Damping

We have previously described a simplified, edge-local damping operation (LD), whose action is restricted to the local subgraph affected by ELC [3]. We now generalize LD to include the incoming message $\mu_j^{v \leftarrow u}$ on an edge (u, v), producing a new outbound message,

$$\mu_{j+1}^{v \rightarrow u} = L_j^v + \alpha(\Gamma_j^v - \mu_j^{v \leftarrow u}). \tag{3}$$

Edges inserted by ELC contain no information, $\mu_j^{v \leftrightarrow u} \triangleq 0$, so these are initialized using (3), in accordance with [3]. With LD, edges unaffected by ELC are left unchanged, so the reasoning for accumulating on the input no longer applies ($\mathbf{L}$ remains the channel vector for the entire frame). We define local-neighborhood damping (ND) as (3) applied to *all* edges (not just new edges) adjacent to all variable nodes affected by ELC on an edge (u', v'), namely $v \in \mathcal{N}_{u'}$. Since all edges adjacent to every $v \in \mathcal{N}_{u'}$ are damped, it is advantageous to also accumulate, according to (1), on the input of each $v \in \mathcal{N}_{u'}$. Hence, ND is a step towards GD, in that it operates on a node rather than edge level, while taking advantage of any extrinsic information on edges unaffected by ELC.



Algorithm 1 SISO-HDPC$(p, I_1, I_2, I_3, \alpha_0, \text{OP}, D)$. Stages **A** and **C** may only apply when implementing the ABP or ABP-ELC algorithm

```
1: α := α_0
2: for I_3 times do
3:    Restart decoder from channel vector
4:    for I_2 times do
5:       (C) HDD stage. RS code only
6:       Stop if syndrome check is satisfied
7:       Apply damping rule, D, with coefficient α
8:       Apply at random p operations, OP
9:       (A) Random row additions (avoid weight-1 columns)
10:      for I_1 times do
11:         Apply SPA iteration (flooding scheduling)
12:      end for
13:   end for
14:   Increment α towards 1
15: end for
```

5 Results

The frame error-rate (FER) performance of ABP-ELC is simulated on the [31, 25, 7] RS code over GF(2^5) (we use a binary [155, 125] image), and the [48, 24, 12] extended QR code. Various decoding algorithms are implemented using our generalized SISO HDPC decoder, Algorithm 1 [3], with the configurations specified in Table 1. For all decoders, the maximum number of iterations is $\tau = I_1 I_2 I_3$. For ABP-ELC, the damping rule, D, is LD or ND. When $I_2 = \tau$ and $I_1 = I_3 = 1$, damping is constant ($\alpha = \alpha_0$ for the entire frame), while damping is disabled by configuring $\alpha_0 = 1$. For comparison with other decoders, the same parameters were used for our simulations; most importantly, α_0 and τ. For ABP, we use "Variation A" (avoid weight-1 columns), whereas we do not use "Variation B" (list decoding) [5]. For SPA and SPA-PD we use a non-systematic matrix optimized on weight. For ABP and ABP-ELC, the initial matrix weight is less important, due to the effects of the GE or ELC stages [7].

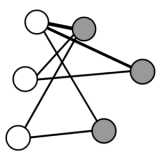
It is interesting to note that ABP-ELC outperforms SPA-PD for the QR code, which has a very large automorphism group [3]. For both codes, Figs. 3(a) and 4(a) show that the ABP-ELC decoder also has a gain over the successful ABP algorithm, even when ABP uses "Variation

Table 1: *Decoding algorithms simulated in this work, and the corresponding configurations of Algorithm 1*

Decoding Algorithm	Configuration
SPA(τ)	SISO-HDPC$(0, 1, \tau, 1, 1, -, -)$
ABP$(\tau, 1, \alpha)$	SISO-HDPC$(1, 1, \tau, 1, \alpha, \text{GE}, \text{GD})$
SPA-PD$(I_1, I_2, I_3, \alpha_0)$	SISO-HDPC$(1, I_1, I_2, I_3, \alpha_0, \text{PD}, \text{GD})$
SPA-ELC$(p, I_1, I_2, I_3, \alpha_0)$	SISO-HDPC$(p, I_1, I_2, I_3, \alpha_0, \text{ELC}, \text{LD})$
ABP-ELC(p, τ, α, D)	SISO-HDPC$(1, 1, \tau, 1, \alpha, \text{ABP–ELC}(p), D)$

C," a symbol level hard-decision list decoding (HDD) stage for the RS code (denoted ABP & HDD). This gain is attributed mainly to our improved ABP-ELC heuristic, but we also observe a gain due to the modified damping. For both codes, the performance of ABP-ELC shows an improvement over ABP, and even over ABP & HDD until a FER of 10^{-4}. With ABP-ELC & HDD, an additional constant gain in FER is achieved, most importantly in terms of a reduced flooring effect at high signal-to-noise ratios (SNRs). Even without the HDD stage on the RS code, ABP-ELC is only about 0.07 dB away from ABP-ELC & HDD, until a FER of 10^{-4}. For the QR code, ABP-ELC approaches the union bound (based on the full weight enumerator of the code) within 0.3 dB at a FER of 10^{-5}. To examine the benefit of reducing p, we simulate ABP-ELC over SNRs 3.5, 4.5, and 5.0 dB for $0 < p \leq n - k$, Figs. 3(b) and 4(b). The performance of ABP at the corresponding SNR points is indicated by the horizontal grey lines. We observe that the performance of ABP-ELC improves with increasing p only initially, before flattening out at an optimal value, $p^\star \ll n - k$, of approximately 7 and 10 for the QR and RS code, respectively. For the QR code, the performance of ABP is matched (intersecting the grey lines) for $p \approx 3$. We also see that the optimal type of damping (LD or ND) depends on the code, and can have a significant impact on performance, as shown in Fig. 4(b).

We now focus on the reduction in decoding complexity, measured in terms of average total number of SPA iterations. The HDD stage is implemented using a "genie-aided stopping criterion" correcting up to $\left\lfloor \frac{d_{\min}-1}{2} \right\rfloor = 3$ symbol errors, and may stop early [5]. An actual implementation, e.g. using the Berlekamp-Massey algorithm, would



require a list implementation. Since the number of iterations would always be τ, the complexity in terms of SPA iterations is not included.

We also consider the complexity in terms of average number of ELC operations per GE or ELC stage. Due to cancellation effects, an ELC stage may perform less than p ELC operations. Define a function, $\bar{p}(p)$, to give the average number of ELC operations performed per ELC stage, for a given p. Figs. 3(c) and 4(c) compare $\bar{p}(p)$ against the linear (possibly redundant) ELC complexity of ABP. As stated also in [6], a reduction can be achieved by simply improving the implementation of the GE stage to do no work whenever an unreliable position is already in $\mathcal{P}$. The plots verify that ABP performs $n-k-\bar{p}(n-k)$ redundant operations (pivotals). Even comparing against such an improved, reduced-complexity GE stage, we see that $\bar{p}(p)$ grows linearily before flattening out at $p' \gtrsim 2p^\star$. Thus, we may achieve a further reduction in ELC complexity (over an improved GE stage) of $\bar{p}(n-k) - \bar{p}(p^\star)$.

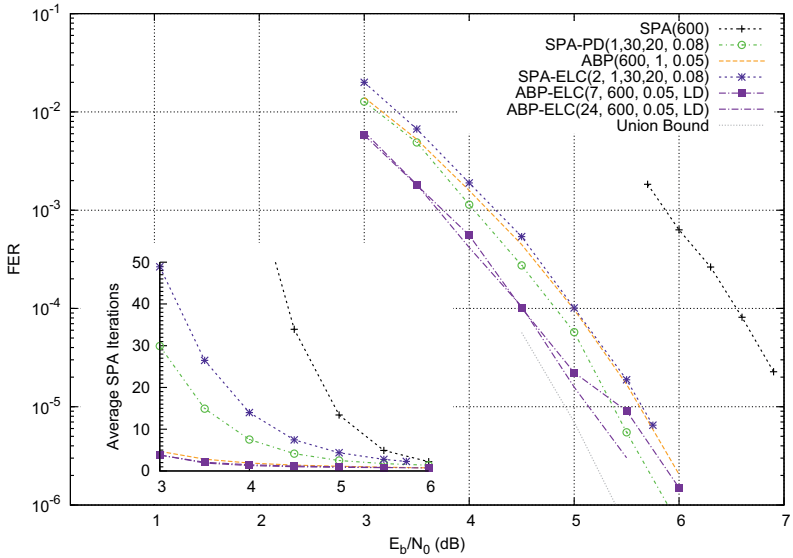
6 Conclusion and Future Work

We have described an ABP-ELC decoder to target the least reliable positions (inferred error positions) during decoding, as well as a generalized, *local-neighborhood* damping rule. An improvement is shown over related algorithms, both in terms of FER performance and complexity, which is ascribed mainly to an improved heuristic to apply the ELC operations. Simultaneously, the amount of ELC operations can be reduced significantly, compared to a GE stage, for an improvement also in complexity. Extensive simulation data is presented for an extended QR code and an RS code. Future work is concerned with further extensions of the ABP-ELC heuristic, and using a non-systematic variant of ELC to avoid degree-1 nodes. We are also working on a distributed implementation of the sorting stage, where each check node sorts its adjacent variable nodes.

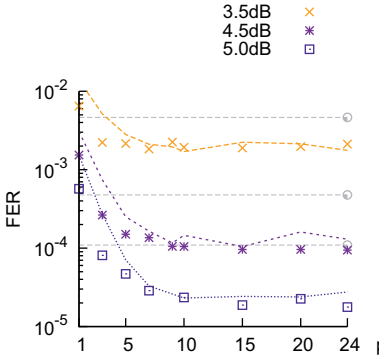
References

[1] Halford, T. R., Chugg, K. M.: Random redundant iterative soft-in soft-out decoding. *IEEE Trans. Commun.* 56(4), 513–517, Apr. 2008.

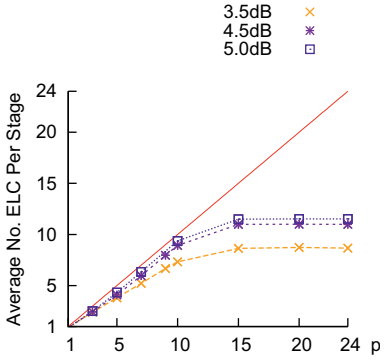
[2] Dimnik, I., Be'ery, Y.: Improved random redundant iterative HDPC decoding. *IEEE Trans. Commun.* 57(7), 1982–1985, Jul. 2009.



(a) FER performance, and average number of SPA iterations per frame.

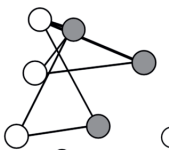


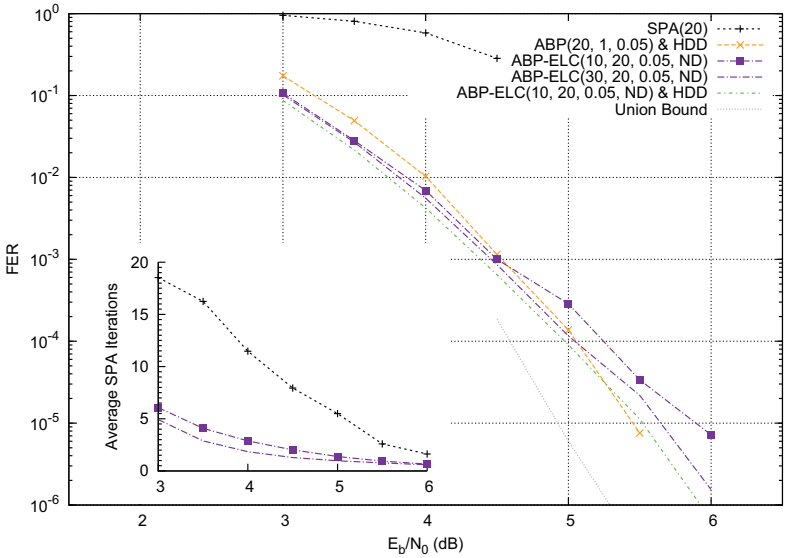
(b) FER plotted against p.



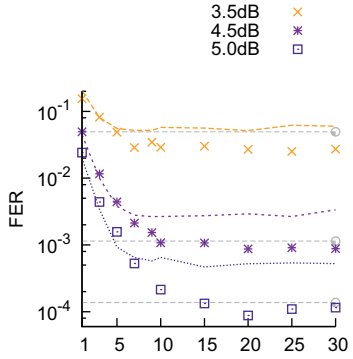
(c) Average ELC usage, $\bar{p}(p)$.

Fig. 3: *For the* [48, 24, 12] *extended QR code, the best performance is achieved using* LD. *The performance using* ND *is shown by the curved lines in Fig. 3(b). Maximum* $\tau = 600$ *iterations.*

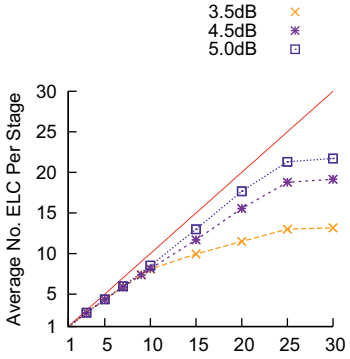




(a) FER performance, and average number of SPA iterations per frame.



(b) FER plotted against p.



(c) Average ELC usage, $\bar{p}(p)$.

Fig. 4: *For the binary image of the* $[31, 25, 7]$ *RS code over* $\mathrm{GF}(2^5)$*, the gain is greater using* ND*. The performance using* LD *is indicated by the curved lines in Fig. 4(b). Maximum* $\tau = 20$ *iterations.*

[3] Knudsen, J. G., Riera, C., Danielsen, L. E., Parker, M. G., Rosnes, E.: Iterative decoding on multiple Tanner graphs using random edge local complementation. In *Proc. IEEE Int. Symp. Inform. Theory*, pp. 899–903. Seoul, Korea, Jun./Jul. 2009.

[4] Jiang, J., Narayanan, K. R.: Iterative soft decoding of Reed-Solomon codes. *IEEE Commun. Lett.* 8(4), 244–246, Apr. 2004.

[5] Jiang, J., Narayanan, K. R.: Iterative soft-input soft-output decoding of Reed-Solomon codes by adapting the parity-check matrix. *IEEE Trans. Inform. Theory* 52(8), 3746–3756, Aug. 2006.

[6] El-Khamy, M., McEliece, R. J.: Iterative algebraic soft-decision list decoding of Reed-Solomon codes. *IEEE J. Sel. Areas Commun.* 24(3), 481–490, Mar. 2006.

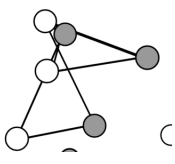
[7] Knudsen, J. G., Riera, C., Danielsen, L. E., Parker, M. G., Rosnes, E.: On iterative decoding of HDPC codes using weight-bounding graph operations. In *Proc. Int. Zürich Seminar on Commun.*, pp. 98–101. Zürich, Switzerland, Mar. 2010.

[8] Danielsen, L. E., Parker, M. G.: Edge local complementation and equivalence of binary linear codes. *Des. Codes Cryptogr.* 49(1-3), 161–170, Dec. 2008.

[9] MacWilliams, F. J., Sloane, N. J. A.: *The Theory of Error-Correcting Codes*. North Holland, 1977.

[10] Kothiyal, A., Takeshita, O. Y.: A comparison of adaptive belief propagation and the best graph algorithm for the decoding of linear block codes. In *Proc. IEEE Int. Symp. Inform. Theory*, pp. 724–728. Adelaide, Australia, Sep. 2005.

[11] Hehn, T., Huber, J. B., Laendner, S., Milenkovic, O.: Multiple-bases belief-propagation for decoding of short block codes. In *Proc. IEEE Int. Symp. Inform. Theory*, pp. 311–315. Nice, France, Jun. 2007.



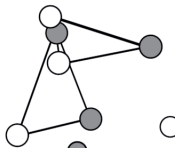
PAPER V

ON GRAPHS AND CODES PRESERVED BY EDGE LOCAL COMPLEMENTATION *

Lars Eirik Danielsen Matthew G. Parker

Constanza Riera Joakim Grahl Knudsen

*Danielsen, L. E., Parker, M. G., Riera, C., Knudsen, J. G.: On graphs and codes preserved by edge local complementation, 2010. arXiv:1006.5802.
This paper was presented at the 10th Nordic Combinatorial Conference (NORCOM 2010), Reykjavik, Iceland, May 2010.



ON GRAPHS AND CODES PRESERVED BY EDGE LOCAL COMPLEMENTATION

Lars Eirik Danielsen Matthew G. Parker

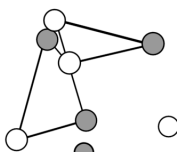
Constanza Riera Joakim Grahl Knudsen

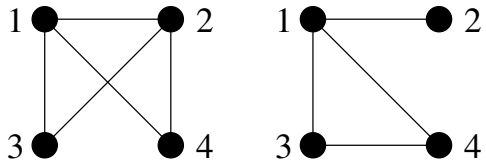
Orbits of graphs under local complementation (LC) and edge local complementation (ELC) have been studied in several different contexts. For instance, there are connections between orbits of graphs and error-correcting codes. We define a new graph class, ELC-preserved graphs, comprising all graphs that have an ELC orbit of size one. Through an exhaustive search, we find all ELC-preserved graphs of order up to 12 and all ELC-preserved bipartite graphs of order up to 16. We provide general recursive constructions for infinite families of ELC-preserved graphs, and show that all known ELC-preserved graphs arise from these constructions or can be obtained from Hamming codes. We also prove that certain pairs of ELC-preserved graphs are LC equivalent. We define ELC-preserved codes as binary linear codes corresponding to bipartite ELC-preserved graphs, and study the parameters of such codes.

1 Introduction

1.1 Graphs

A *graph* is a pair $G = (V, E)$ where V is a set of *vertices*, and $E \subseteq V \times V$ is a set of *edges*. The *order* of G is $n = |V|$. A graph with n vertices can be





(a) The Graph G (b) The Graph $G * 1$

Fig. 1: *Example of local complementation*

represented by an $n \times n$ *adjacency matrix* Γ, where $\Gamma_{i,j} = 1$ if $\{i, j\} \in E$, and $\Gamma_{i,j} = 0$ otherwise. We will only consider *simple undirected* graphs, whose adjacency matrices are symmetric with all diagonal elements being 0, i.e., all edges are bidirectional and no vertex can be adjacent to itself. The *neighborhood* of $v \in V$, denoted $N_v \subset V$, is the set of vertices connected to v by an edge. The number of vertices adjacent to v is called the *degree* of v. The *induced subgraph* of G on $W \subseteq V$ contains vertices W and all edges from E whose endpoints are both in W. The *complement* of G is found by replacing E with $V \times V - E$, i.e., the edges in E are changed to non-edges, and the non-edges to edges. Two graphs $G = (V, E)$ and $G' = (V, E')$ are *isomorphic* if and only if there exists a permutation π on V such that $\{u, v\} \in E$ if and only if $\{\pi(u), \pi(v)\} \in E'$. A *path* is a sequence of vertices, $(v_1, v_2, \ldots, v_i)$, such that $\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{i-1}, v_i\} \in E$. A graph is *connected* if there is a path from any vertex to any other vertex in the graph. A graph is *bipartite* if its set of vertices can be decomposed into two disjoint sets such that no two vertices within the same set are adjacent. We call the graph (a, b)*-bipartite* if these sets are of size a and b, respectively.

Definition 1 ([1–3])**.** *Given a graph $G = (V, E)$ and a vertex $v \in V$, let $N_v \subset V$ be the neighborhood of v.* Local complementation *(LC) on v transforms G into $G * v$ by replacing the induced subgraph of G on N_v by its complement. (Fig. 1)*

Definition 2 ([2])**.** *Given a graph $G = (V, E)$ and an edge $\{u, v\} \in E$,* edge local complementation *(ELC) on $\{u, v\}$ transforms G into $G^{(u,v)} = G * u * v * u = G * v * u * v$.*

Definition 3 ([2])**.** *ELC on $\{u, v\}$ can equivalently be defined as follows. Decompose $V \setminus \{u, v\}$ into the following four disjoint sets, as visualized in Fig. 2.*

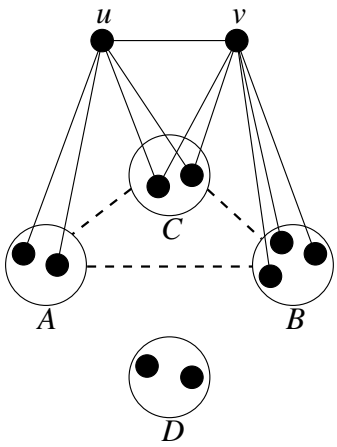


Fig. 2: *Visualization of the ELC operation*

A Vertices adjacent to u, but not to v.

B Vertices adjacent to v, but not to u.

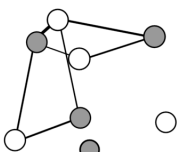
C Vertices adjacent to both u and v.

D Vertices adjacent to neither u nor v.

To obtain $G^{(u,v)}$, perform the following procedure. For any pair of vertices $\{x, y\}$, where x belongs to class A, B, or C, and y belongs to a different class A, B, or C, "toggle" the pair $\{x, y\}$, i.e., if $\{x, y\} \in E$, delete the edge, and if $\{x, y\} \notin E$, add the edge $\{x, y\}$ to E. Finally, swap the labels of vertices u and v.

Definition 4. *The graphs G and G′ are* LC-equivalent *(resp.* ELC-equivalent*) if a graph isomorphic to G′ can be obtained by applying a finite sequence of LC (resp. ELC) operations to G. The* LC orbit *(resp.* ELC orbit*) of G is the set of all non-isomorphic graphs that can be obtained by performing any finite sequence of LC (resp. ELC) operations on G.*

The LC operation was first defined by de Fraysseix [3], and later studied by Fon-der-Flaas [1] and Bouchet [2]. Bouchet defined ELC as "complementation along an edge" [2], but this operation is also known as *pivoting* on a graph. LC orbits of graphs have been used to study *quantum graph states* [4, 5], which are equivalent to *self-dual additive codes*



over $\mathbb{F}_4$ [6]. We have previously used LC orbits to classify such codes [7]. There are also connections between graph orbits and properties of *Boolean functions* [8, 9]. *Interlace polynomials* of graphs have been defined with respect to both ELC [10] and LC [11]. These polynomials encode certain properties of the graph orbits, and were originally used to study a problem related to DNA sequencing [12]. We have previously studied connections between interlace polynomials and error-correcting codes [13]. Bouchet [14] proved that a graph is a *circle graph* if and only if certain subgraphs, or obstructions, do not appear anywhere in its LC orbit. Similarly, circle graph obstructions under ELC were described by Geelen and Oum [15]. As we will see later, bipartite graphs correspond to binary linear error-correcting codes. ELC can be used to generate orbits of equivalent codes, which has been used to classify codes [16]. ELC also has applications in *iterative decoding* of codes [17–20].

For bipartite graphs, we can simplify the ELC operation, since the set C in Fig. 2 must be empty. Given a bipartite graph $G = (V, E)$ and an edge $\{u, v\} \in E$, $G^{(u,v)}$ can be obtained by "toggling" all edges between the sets $N_u \setminus \{v\}$ and $N_v \setminus \{u\}$, followed by a swapping of vertices u and v. Moreover, if G is an (a, b)-bipartite graph, then, for any edge $\{u, v\} \in E$, $G^{(u,v)}$ must also be (a, b)-bipartite [8]. Note that LC does not, in general, preserve bipartiteness. It follows from Definition 2 that every LC orbit can be partitioned into one or more ELC orbits. If $G = (V, E)$ is a connected graph, then, for any vertex $v \in V$, $G * v$ must also be connected. Likewise, for any edge $\{u, v\} \in E$, $G^{(u,v)}$ must be connected.

Definition 5. *A graph $G = (V, E)$ is called* ELC-preserved *if for any edge $\{u, v\} \in E$, $G^{(u,v)}$ is isomorphic to G. In other words, G is ELC-preserved if and only if the ELC orbit of G contains only G itself.*

We only consider connected graphs, since a disconnected graph is ELC-preserved if and only if its connected components are ELC-preserved. Trivially, empty graphs are ELC-preserved. We could also define an LC-preserved graph as a graph where LC on any vertex preserves the graph, up to isomorphism. A search of all connected graphs of order up to 12 reveals that only the unique connected graph of order two has this property.

1.2 Codes

A binary linear code, $\mathcal{C}$, is a linear subspace of $\mathbb{F}_2^n$ of dimension k. The 2^k elements of $\mathcal{C}$ are called *codewords*. The *Hamming weight* of a codeword is the number of nonzero components. The *minimum distance* of $\mathcal{C}$ is equal to the smallest nonzero weight of any codeword in $\mathcal{C}$. A code with minimum distance d is called an $[n, k, d]$ code. Two codes are *equivalent* if one can be obtained from the other by a permutation of the coordinates. A permutation that maps a code to itself is called an *automorphism*. All automorphisms of $\mathcal{C}$ make up its *automorphism group*. We define the *dual code* of $\mathcal{C}$ with respect to the standard inner product, $\mathcal{C}^\perp = \{\boldsymbol{u} \in \mathbb{F}_2^n \mid \boldsymbol{u} \cdot \boldsymbol{c} = 0 \text{ for all } \boldsymbol{c} \in \mathcal{C}\}$. $\mathcal{C}$ is called *self-dual* if $\mathcal{C} = \mathcal{C}^\perp$, and *isodual* if $\mathcal{C}$ is equivalent to $\mathcal{C}^\perp$. The code $\mathcal{C}$ can be defined by a $k \times n$ *generator matrix*, C, whose rows span $\mathcal{C}$. By column permutations and elementary row operations C can be transformed into a matrix of the form $C' = (I \mid P)$, where I is a $k \times k$ identity matrix, and P is some $k \times (n-k)$ matrix. The matrix C', which is said to be of *standard form*, generates a code which is equivalent to $\mathcal{C}$. The matrix $H' = (P^{\mathrm{T}} \mid I)$, where I is an $(n-k) \times (n-k)$ identity matrix is the generator matrix of $\mathcal{C}'^\perp$ and is called the *parity check matrix* of $\mathcal{C}'$.

Definition 6 ([21, 22])**.** *Let $\mathcal{C}$ be a binary linear $[n, k]$ code with generator matrix $C = (I \mid P)$. Then the code $\mathcal{C}$ corresponds to the $(k, n-k)$-bipartite graph on n vertices with adjacency matrix*

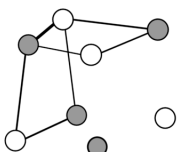
$$\Gamma = \begin{pmatrix} \mathbf{0}_{k\times k} & P \\ P^T & \mathbf{0}_{(n-k)\times(n-k)} \end{pmatrix},$$

where $\mathbf{0}$ denotes all-zero matrices of the specified dimensions.

Theorem 1 ([16])**.** *Applying any sequence of ELC operations to a graph corresponding to a code $\mathcal{C}$ will produce a graph corresponding to a code equivalent to $\mathcal{C}$. Moreover, graphs corresponding to equivalent codes will always belong to the same ELC orbit.*

Note that, up to isomorphism, one bipartite graph corresponds to both the code $\mathcal{C}$ generated by $(I \mid P)$, and the code $\mathcal{C}^\perp$ generated by $(P^{\mathrm{T}} \mid I)$. When $\mathcal{C}$ is isodual, the ELC-orbit of the associated graph correspond to a single equivalence class of codes. Otherwise, the ELC-orbit correspond to two equivalence classes, that of $\mathcal{C}$ and that of $\mathcal{C}^\perp$ [16].

Definition 7. *An* ELC-preserved code *is a binary linear code corresponding to an ELC-preserved bipartite graph.*



It follows from Theorem 1 that ELC allows us to jump between all standard form generator matrices of a code. Hence an ELC-preserved code is a code that has only one standard form generator matrix, up to column permutations.

Theorem 2 ([16]). *The minimum distance of an $[n, k, d]$ binary linear code $\mathcal{C}$ is $d = \delta + 1$, where δ is the smallest vertex degree of any vertex in a fixed partition of size k over all graphs in the associated ELC orbit. The minimum vertex degree in the other partition over the ELC orbit gives the minimum distance of $\mathcal{C}^{\perp}$.*

For an ELC-preserved graph, Theorem 2 means that the minimum distance of the associated code, and its dual code, can be found simply by finding the minimum vertex degree in each partition of the graph.

It has been shown that ELC can improve the performance of iterative decoding [17–20]. This technique, which will not be described in detail here, involves applying ELC operations to a bipartite graph between iterations of a *sum-product algorithm* which attempts to decode a received noisy vector to the nearest codeword in the the corresponding code. In this application, labeled graphs are used, so that ELC is equivalent to row additions on an initial generator matrix of the form $(I \mid P)$, which means that the corresponding code is preserved. (It is the parity check matrix of the code that is actually used in iterative decoding, but we have already seen that, up to isomorphism, the bipartite graph corresponding to the generator matrix and parity check matrix of a code is the same.) For an ELC-preserved code, all generator matrices must be column permutations of one unique generator matrix, and hence these permutations must all be automorphisms of the code. It follows that iterative decoding with ELC on an ELC-preserved code is equivalent to a variant of *permutation decoding* [17, 23].

1.3 Outline

In Section 2, we show that there do exist non-trivial bipartite and non-bipartite ELC-preserved graphs. We find all ELC-preserved graphs of order up to 12 and all ELC-preserved bipartite graphs of order up to 16. In Section 3, we show that *star graphs* and *complete graphs* as well as graphs corresponding to *Hamming codes* and *extended Hamming codes* are ELC-preserved. We then prove that more ELC-preserved graphs can be obtained from four recursive constructions. Given a bipartite ELC-preserved graph, a larger bipartite ELC-preserved graph is constructed

by *star expansion*. Similarly, *clique expansion* produces non-bipartite ELC-preserved graphs. *Hamming expansion* and the related *Hamming clique expansion* use a special graph of order seven, corresponding to a Hamming code, to obtain new ELC-preserved graphs. In Section 4, we show that all ELC-preserved graphs of order up to 12, and all ELC-preserved bipartite graphs of order up to 16, are obtained from these constructions. We also prove that certain pairs of ELC-preserved graphs are LC equivalent. In particular, from extended Hamming codes, we obtain new non-bipartite ELC-preserved graphs via LC. The properties of ELC-preserved codes obtained from star expansion and Hamming expansion are described in Section 5. In particular, we enumerate and construct new self-dual ELC-preserved codes. In Section 6 we briefly consider the generalization from ELC-preserved graphs to graphs with orbits of size two, and study the corresponding codes. Finally, in Section 7, we conclude with some ideas for future research.

2 Enumeration

From previous classifications [7, 16], we know the ELC orbit size for all graphs of order $n \leq 12$, and all bipartite graphs of order $n \leq 15$. (A database of ELC orbits is available on-line at `http://www.ii.uib.no/~larsed/pivot/`.) We find that a small number of ELC orbits of size one exist for each order n. Despite the much smaller number of bipartite graphs, there are approximately the same number of ELC-preserved bipartite and non-bipartite graphs for $n \leq 12$. The numbers of ELC-preserved graphs, together with the total numbers of ELC orbits, are given in Table 1. Note that all numbers are for connected graphs.

By using an extension technique we were also able to generate all ELC-preserved bipartite graphs of order $n = 16$. Given the 1,156,716 ELC orbit representatives for $n = 15$, we extend each (a, b)-bipartite graph in $2^a + 2^b - 2$ ways, by adding a new vertex and connecting it to all possible combinations of at least one of the old vertices. The complete set of extended graphs is significantly smaller than that set of all bipartite connected graphs of order 16, but it must contain at least one representative from each ELC orbit. To see that this is true, consider a connected bipartite graph G of order 16. The induced subgraph on any 15 vertices of G must be ELC-equivalent to one of the graphs that were extended to form the extended set, and hence there must be at least one graph in the extended set that is ELC-equivalent to G. We check each member of the extended set, and find that there are 6

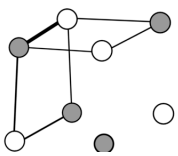


Table 1: *Number of non-bipartite ELC orbits (nb_n), non-bipartite ELC-preserved graphs (nbp_n), bipartite ELC orbits (b_n), and bipartite ELC-preserved graphs (bp_n)*

n	nb_n	nbp_n	b_n	bp_n
2	-	-	1	1
3	1	1	1	1
4	2	1	2	1
5	7	1	3	1
6	27	2	8	2
7	119	1	15	2
8	734	2	43	3
9	6,592	3	110	2
10	104,455	3	370	2
11	3,369,057	2	1,260	1
12	231,551,924	6	5,366	5
13			25,684	1
14			154,104	5
15			1,156,716	4
16			?	6

connected bipartite ELC-preserved graphs of order 16. Note that this is the same extension technique that was used to classify ELC orbits [16], but checking if a graph is ELC-preserved is much faster than generating its entire ELC orbit, since we only need to consider ELC on each edge of the graph, and can stop and reject the graph as soon as a second orbit member is discovered.

3 Constructions

For all $n \geq 2$, there is a bipartite ELC-preserved graph of order n, namely the *star graph*, denoted s^n. This graph has one vertex, v, of

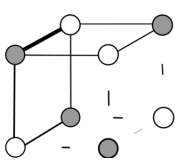
degree $n-1$ and $n-1$ vertices, $u_1, u_2, \ldots, u_{n-1}$, of degree 1. Clearly the graph is ELC-preserved, since for all edges $\{u_i, v\}$, $N_{u_i} \setminus \{v\} = \emptyset$. The construction given in Theorem 3 gives us more bipartite ELC-preserved graphs. For brevity, we will denote $N_v^u = N_v \setminus (N_u \cup \{u\})$. Let e^n denote the *empty graph* on n vertices, i.e., a graph with no edges.

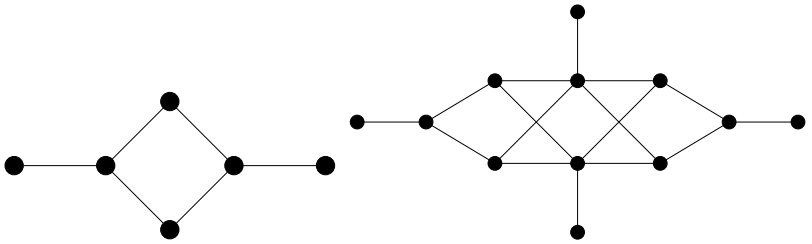
Definition 8 ([10, 24])**.** *Given a graph $G = (V, E)$, a vertex $v \in V$, and another graph $H = (V', E')$, where $V \cap V' = \emptyset$, by* substituting *v with H, we obtain the graph $G' = (V \setminus \{v\} \cup V', E'')$, where E'' is obtained by taking the union of E and E', removing all edges incident on v, and joining all vertices in V' to w whenever $\{v, w\} \in E$.*

Definition 9 ([25])**.** *Given a graph $G = (V, E)$, and a vertex $v \in V$, we add a* pendant *at v by adding a new vertex w to V and a new edge $\{v, w\}$ to E.*

Theorem 3 (Star expansion)**.** *Given an ELC-preserved bipartite graph $G = (V, E)$ on k vertices and an integer $m > 1$, we obtain an ELC-preserved bipartite graph $S^m(G)$ on $n = km$ vertices by substituting all vertices in one partition of G with e^m and adding $m-1$ pendants to all vertices in the other partition.*

Proof. Let $\{u, v\} \in E$. Without loss of generality, assume that u is substituted by $u_1, \ldots, u_m$, all incident on v. Moreover, pendant vertices $w_1, \ldots, w_{m-1}$ are added, with v as their only neighbor. Clearly ELC on $\{v, w_i\}$ is ELC-preserving. Due to symmetries, it only remains to show that ELC on an edge $\{u_i, v\}$ preserves $S^m(G)$. In the graph G, let $A = N_u^v$ and $B = N_v^u$. In the graph $S^m(G)$, $N_{u_i}^v = A$, and $N_v^{u_i} = (B_1 \cup \cdots \cup B_m) \cup C \cup D$, where $C = \{w_1, \ldots, w_{m-1}\}$ and $D = \{u_1, \ldots, u_m\} \setminus \{u_i\}$. The subgraph induced on $A \cup B_j$ in $S^m(G)$, for $1 \leq j \leq m$, is isomorphic to the subgraph induced on $A \cup B$ in G. ELC on $\{u_i, v\}$ means that we toggle all pairs of vertices between $N_{u_i}^v$ and $N_v^{u_i}$. Toggling pairs between A and B_j, for $1 \leq j \leq m$, preserves $S^m(G)$, since toggling pairs between A and B preserves G. (The fact that all vertices in A have $m-1$ added pendants has no effect on this.) Finally, in addition to swapping u_i and v, ELC has the effect of toggling pairs of vertices between A and C, and between A and D. In $S^m(G)$, all vertices in A are connected to all vertices in D, and no vertex in A is connected to any vertex in C. The sets C and D are both of size $m-1$, the vertices in C have no other neighbors than v, and the vertices in D have no other neighbors than $A \cup \{v\}$. Hence ELC on $\{u_i, v\}$ simply swaps the vertices in C with the vertices in D. This means that $S^m(G)^{(u_i, v)}$ is isomorphic to $S^m(G)$, and it follows that $S^m(G)$ is ELC-preserved.





(a) The graph $S^2_-(s^3)$ (b) The graph $S^2_-(S^2_-(s^3))$

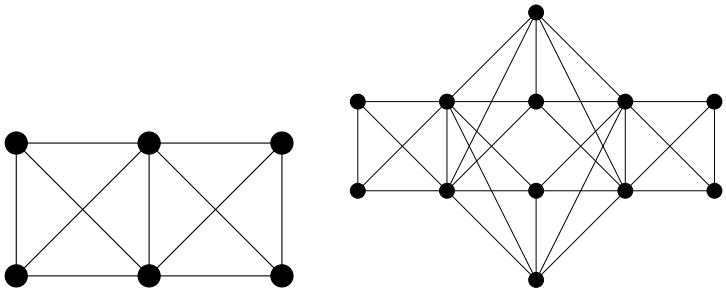
Fig. 3: *Examples of star expansion*

Furthermore, $S^m(G)$ must be bipartite, since substituting vertices by empty graphs and adding pendants cannot make a bipartite graph non-bipartite. □

Examples of graphs obtained by star expansion are shown in Fig. 3. From Theorem 3 we can obtain two different graphs, by choosing in which partition of G we substitute vertices with e^m. In our examples, when the partitions of G are of unequal size, we write $S^m_+(G)$ when we substitute the vertices in the largest partition, and $S^m_-(G)$ when we substitute the vertices in the smallest partition. In the cases where the partitions are of equal size, $S^m(G)$ will give the same graph for both partitions in all examples in this paper. If G is an $(r, k-r)$-bipartite graph, then $S^m(G)$ will be $(r+k(m-1), k-r)$-bipartite. Since its output is always bipartite, the star expansion construction can be iterated to obtain new ELC-preserved graphs, such as the graph $S^2_-(S^2_-(s^3))$ of order 12, shown in Fig. 3(b). However, some of these iterated constructions can be simplified. For instance, it is easy to verify that $S^m_+(s^k) = s^{km}$ and $S^{m_2}_+(S^{m_1}_-(s^k)) = S^{m_1 m_2}_-(s^k)$.

For all $n \geq 3$, there is a non-bipartite ELC-preserved graph on n vertices, namely the *complete graph*, denoted c^n. This graph has n vertices, $v_1, v_2, \ldots, v_n$, of degree $n-1$. Clearly the graph is ELC-preserved, since for all edges $\{v_i, v_j\}$, $N_{v_i} = N_{v_j}$, and hence the sets A and B in Fig. 2 are empty. The following more general construction gives us more non-bipartite ELC-preserved graphs.

Theorem 4 (Clique expansion)**.** *Given an ELC-preserved graph G on k vertices and an integer $m > 1$, we obtain an ELC-preserved non-bipartite graph $C^m(G)$ on $n = km$ vertices by substituting all vertices of G with c^m.*

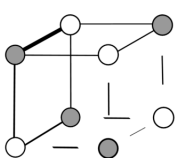


(a) The graph $C^2(s^3)$ (b) The graph $C^2(S^2_{-}(s^3))$

Fig. 4: *Examples of clique expansion*

Proof. Let $\{u, v\} \in E$. Let u be substituted by $u_1, \ldots, u_m$, and let v be substituted by $v_1, \ldots, v_m$. ELC on any edge within a substituted subgraph, such as $\{u_i, u_j\}$, must preserve $C^m(G)$, since $N_{u_i} = N_{u_j}$. Due to symmetries, it only remains to show that ELC on an edge $\{u_i, v_j\}$ preserves $C^m(G)$. In the graph G, let $A = N_u^v$, $B = N_v^u$, and $C = N_u \cap N_v$. In the graph $C^m(G)$, $N_{u_i}^{v_j} = A_1 \cup \cdots \cup A_m$, $N_{v_j}^{u_i} = B_1 \cup \cdots \cup B_m$, and $N_{u_i} \cap N_{v_j} = (C_1 \cup \cdots \cup C_m) \cup U \cup V$, where $U = \{u_1, \ldots, u_m\} \setminus \{u_i\}$ and $V = \{v_1, \ldots, v_m\} \setminus \{v_j\}$. Let $X, Y \in \{A, B, C\}$, $X \neq Y$. All subgraphs in $C^m(G)$ induced on X_r are isomorphic to subgraphs in G induced on X. A vertex $x_r \in X_r$ is connected to a vertex $y_s \in Y_s$ in $C^m(G)$, for $1 \leq r, s \leq m$ if and only if $x \in X$ is connected to $y \in Y$ in G. Hence, toggling pairs between X_r and Y_s, for $1 \leq r, s \leq m$, preserves $C^m(G)$ since toggling pairs between X and Y preserves G. (The fact that edges have been added between X_r and X_t, for $1 \leq r, t \leq m$, by the clique substitution, has no effect on this, since the subgraphs in $C^m(G)$ induced on $X_r \cup X_t$ are isomorphic for all $1 \leq r, t \leq m$.) The final effect of ELC on $\{u_i, v_j\}$ is to toggle all pairs between $U \cup V$ and $A_1 \cup \cdots \cup A_m$, and all pairs between $U \cup V$ and $B_1 \cup \cdots \cup B_m$. But, since we also swap u_i and v_j, the total effect is equivalent to swapping u_r and v_r for all $1 \leq r \leq m$. It follows that $C^m(G)^{(u_i, v_j)}$ is isomorphic to $C^m(G)$, and hence that $C^m(G)$ is ELC-preserved. □

Examples of graphs obtained by clique expansion are shown in Fig. 4. The output of a clique expansion will always be a non-bipartite graph, except for the trivial case $C^2(e^1) = s^2$. However, the input can be a bipartite graph, and hence the construction can be combined with star



expansion to obtain new ELC-preserved graphs, such as the graph $C^2(S^2_-(s^3))$ of order 12, shown in Fig. 4(b). Iterating clique expansion on its own does not produce new graphs, since, trivially, $C^m(c^k) = c^{mk}$ and $C^{m_2}(C^{m_1}(G)) = C^{m_1 m_2}(G)$.

Theorem 5. *The graph h^r, for $r \geq 3$, is an ELC-preserved $(r, 2^r - r - 1)$-bipartite graph on $n = 2^r - 1$ vertices. To obtain h^r, let one partition, U, consist of r vertices, and the other partition, W, be divided into $r - 1$ disjoint subsets, W_i, for $2 \leq i \leq r$, where W_i contains $\binom{r}{i}$ vertices. Let each vertex in W_i be connected to i vertices in U, such that $N_a \neq N_b$ for all $a, b \in W$. The graph h^r corresponds to the $[2^r - 1, 2^r - r - 1, 3]$ Hamming code.*

Proof. From the construction of the graph h^r, we see that it corresponds to a code with parity check matrix $(I \mid P)$, where the columns are all vectors from $\mathbb{F}_2^r$, which is the parity check matrix of a Hamming code [26]. We know from Theorem 1 that any ELC operation on h^r must give a graph that corresponds to an equivalent code. Since the distance of the code is greater than two, all columns of the parity check matrix must be distinct. It follows that all parity check matrices of equivalent codes must contain all vectors from $\mathbb{F}_2^r$, in some order. Hence the corresponding graphs are isomorphic, and h^r must be ELC-preserved. □

Definition 10. *A graph is called* Eulerian *if all its vertices have even degree, and* anti-Eulerian *if all its vertices have odd degree.*

An anti-Eulerian graph must have even order, and is always the complement of an Eulerian graph. Anti-Eulerian graphs have been shown to correspond to *Type II* self-dual additive codes over $\mathbb{F}_4$ [7].

Lemma 1. *Let $G = (V, E)$ be an anti-Eulerian graph. After performing any LC or ELC operation on G, we obtain a graph G' which is also anti-Eulerian.*

Proof. Let $v \in V$ and $w \in N_v$. LC on v transforms N_w into $N'_w = (N_w \cup N_v) \setminus (N_w \cap N_v) \setminus \{w\}$, where $|N'_w| = |N_w| + |N_v| - (2\,|N_w \cap N_v| + 1)$. Since G is anti-Eulerian, $|N_w|$ and $|N_v|$ must be odd. We then see that $|N'_w|$ is the sum of three odd numbers, and must therefore be odd. The same argument holds for all neighbors of v, so $G * v$ is anti-Eulerian. That ELC also preserves anti-Eulerianicity then follows from Definition 2. □

Theorem 6. *The graph h^r_e, for $r \geq 3$, is an ELC-preserved $(r + 1, 2^r - r - 1)$-bipartite graph on $n = 2^r$ vertices. To obtain h^r_e, first construct h^r, as in*

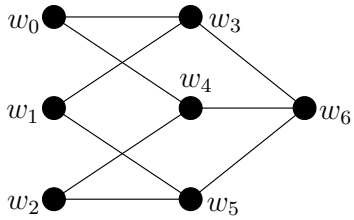


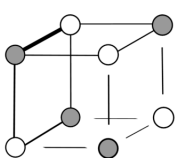
Fig. 5: *The graph* h^3

Theorem 5, and then add a new vertex which is connected by edges to all existing vertices of even degree. The graph h_e^r *corresponds to the* $[2^r, 2^r - r - 1, 4]$ extended Hamming code.

Proof. h_e^r must be bipartite, since all vertices of h^r in the partition of size r have degree $\sum_{i=2}^{r} \binom{r}{i} \frac{i}{r} = 2^{r-1} - 1$, which is odd. The new vertex added to h^r also has odd degree, since the number of vertices in h^r of even degree is $\sum_{i=1}^{\lfloor \frac{r}{2} \rfloor} \binom{r}{2i} = 2^{r-1} - 1$. Hence h_e^r is anti-Eulerian. It follows from the construction that h_e^r corresponds to a code with parity check matrix $(I \mid P)$, where the columns are all odd weight vectors from $\mathbb{F}_2^{r+1}$, which is the parity check matrix of an extended Hamming code [26]. We know from Theorem 1 that any ELC operation on h_e^r must give a graph that corresponds to an equivalent code. Since the distance of the code is greater than two, all columns of the parity check matrix must be distinct. The graph h_e^r is anti-Eulerian, and must remain so after ELC, according to Lemma 1. It follows that all parity check matrices of equivalent codes must contain all odd weight vectors from $\mathbb{F}_2^{r+1}$, in some order. Hence the corresponding graphs are isomorphic, and h_e^r must be ELC-preserved. □

For $n = 7$, we obtain from Theorem 5 the bipartite ELC-preserved graph h^3, shown in Fig. 5, corresponding to the Hamming code of length 7. This is an important graph, as it forms the basis for the general constructions given by Theorems 7 and 8.

Theorem 7 (Hamming expansion)**.** *Given an ELC-preserved graph* $G = (V, E)$ *on* k *vertices, we obtain an ELC-preserved graph* $H(G)$ *on* $n = 7k$ *vertices. For all vertices* $v_i \in V$, $0 \leq i < k$, *we replace* v_i *with the subgraph* $h_i = (\{w_{7i}, \ldots, w_{7i+6}\}, \{\{w_{7i}, w_{7i+3}\}, \{w_{7i}, w_{7i+4}\}, \{w_{7i+1}, w_{7i+3}\}, \{w_{7i+1}, w_{7i+5}\}, \{w_{7i+2}, w_{7i+4}\}, \{w_{7i+2}, w_{7i+5}\}, \{w_{7i+3}, w_{7i+6}\}, \{w_{7i+4}, w_{7i+6}\}, \{w_{7i+5}, w_{7i+6}\}\})$.



(Note that h_i is a specific labeling of the graph h^3. The labeled graph h_0 is depicted in Fig. 5.) If $\{v_i, v_j\} \in E$, we connect each of the vertices w_{7i}, w_{7i+1}, and w_{7i+2} to all the vertices w_{7j}, w_{7j+1}, and w_{7j+2}. (Note that this differs from the graph substitution in Definition 8.)

Proof. Let $a = w_6$, $b = w_3$, and $c = w_0$. If $k > 1$, let $d = w_7$, and assume (without loss of generality) that there is an edge $\{v_0, v_1\} \in E$. Due to the symmetry of $H(G)$ and ELC-preservation of G, we only need to consider ELC on the three edges $\{a, b\}$, $\{b, c\}$, and $\{c, d\}$ to prove the ELC-preservation of $H(G)$. That h_0 is ELC-preserved, and hence that $\{a, b\}$ preserves $H(G)$ is easily verified by hand. We then consider the edge $\{b, c\}$. Note that $N_b^c = \{a, c' = w_1\}$, where c' has exactly the same neighbors as c outside h_0, and a has no common neighbors with c outside h_0. Since we know that the subgraph h_0 is ELC-preserved, the effect of ELC on $\{b, c\}$ is simply to swap a and c'. The edge $\{c, d\}$ corresponds to the edge $\{v_0, v_1\} \in E$. In the graph G, let $A = N_{v_0}^{v_1}$, $B = N_{v_1}^{v_0}$, and $C = N_{v_0} \cap N_{v_1}$. In the graph $H(G)$, c is connected to three copies of A, d is connected to three copies of B, and both c and d are connected to three copies of C. Since ELC on $\{v_0, v_1\}$ preserves G, toggling pairs between these multiplied neighborhoods must preserve $H(G)$, as in Theorem 4. There are only eight remaining vertices to consider: c is connected to $D = \{w_3, w_4\}$ and $E = \{w_8, w_9\}$, and d is connected to $F = \{w_{10}, w_{11}\}$ and $G = \{w_1, w_2\}$. The vertices in D has no neighbors outside h_0, and the vertices in F have no neighbors outside h_1. The vertices in E share the same neighbors as d outside h_1, and the vertices in G share the same neighbors as c outside h_0. The effect of ELC on $\{c, d\}$ is to swap D with E and F with G. Hence $H(G)$ must be preserved, except for the local structure of h_0 and h_1, which it remains to check. ELC on $\{c, d\}$ has the effect of toggling pairs between D and G and between E and F. Finally we swap u and v. The result is that the structure of h_0 and h_1 is preserved, as illustrated in Fig. 6 and Fig. 7. It follows that $H(G)$ is ELC-preserved. □

Theorem 8 (Hamming clique expansion)**.** *For $k \geq 1$ and $m \geq 1$, we obtain an ELC-preserved graph H_k^m on $n = 7k + m$ vertices by taking the union of $G = H(c^k)$ and $K = c^m$. We add edges from each vertex in K to all the $3k$ vertices in G labeled (as in Theorem 7) w_{7i}, w_{7i+1}, and w_{7i+2}, for $0 \leq i < k$. (Note that $H_1^1 = h_e^3$.)*

Proof. Without loss of generality, let $a = w_6$, $b = w_3$, $c = w_0$, $d = w_7$, and let e and f be two distinct vertices in K. (For $k = 1$, ignore d, and for

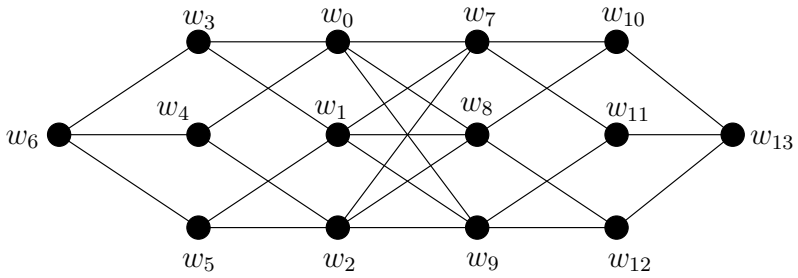


Fig. 6: *The graph* $H(s^2)$

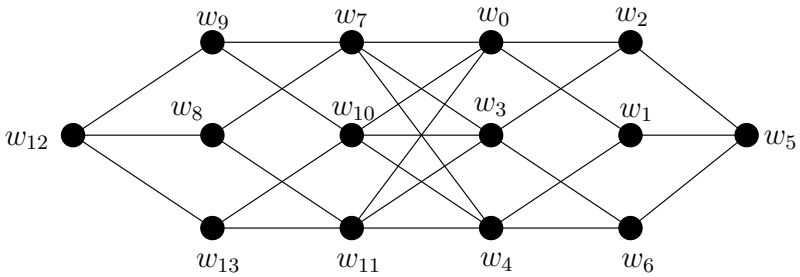
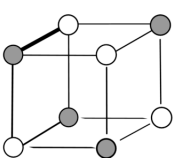


Fig. 7: *The graph* $H(s^2)^{(w_0,w_7)}$



$m = 1$, ignore f.) Due to the symmetry of H_k^m, we only need to consider ELC on the five edges $\{a, b\}$, $\{b, c\}$, $\{c, d\}$, $\{c, e\}$, and $\{e, f\}$ to prove the ELC-preservation of H_k^m. The proof for $\{a, b\}$, $\{b, c\}$, and $\{c, d\}$ are the same as in Theorem 7. (The proof still works with $K = c^m$ added to N_c and N_d.) The edge $\{e, f\}$ is trivial, since $N_e = N_f$. It only remains to show that ELC on $\{c, e\}$ preserves H_k^m. Observe that $N_c^e = \{w_3, w_4\}$ and $N_e^c = \{w_1, w_2\}$. All other neighbors of c and e are in $N_c \cap N_e$, since the underlying graph of $G = H(c^k)$ is a complete graph. Furthermore, w_1 and w_2 are connected to all vertices in $N_c \cap N_e$, and w_3 and w_4 are not connected to any vertex in $N_c \cap N_e$. The effect of ELC is to swap the vertices in N_c^e with the vertices in N_e^c. h_0 is preserved as before. It follows that H_k^m is ELC-preserved. □

Proposition 1. *$H(G)$ is bipartite when $G = (V, E)$ is bipartite. H_k^m is bipartite only in the trivial case where $k = m = 1$.*

Proof. Let $V = \{v_0, \ldots v_{k-1}\}$. In $H(G)$, each v_i is replaced by a bipartite subgraph, h_i, and edges are added between these subgraphs, such that the induced subgraph on $\{w_{7i}, w_{7i+1}, w_{7i+2}, w_{7j}, w_{7j+1}, w_{7j+2}\}$ in $H(G)$ is a complete bipartite graph if there is an edge $\{v_i, v_j\} \in E$ and an empty graph otherwise. It follows that $H(G)$ is bipartite whenever G is bipartite. (The trivial case $H(e^1) = h^3$ is clearly also bipartite.) H_k^m is clearly non-bipartite if $k > 2$ or $m > 2$, since it contains a 3-clique. It is easily checked that for the remaining cases, only $H_1^1 = h_e^3$ is bipartite. □

4 Classification

Tables 2 and 3 shows how all bipartite ELC-preserved graphs of order $n \leq 16$, and all non-bipartite ELC-preserved graphs of order $n \leq 12$ arise from the constructions described in the previous section.

We observe that certain pairs of ELC-preserved graphs are LC-equivalent. It is easy to verify that c^n and s^n form a complete LC orbit, for all $n \geq 3$. The following theorem explains all the remaining pairs of LC-equivalent ELC-preserved graphs for $n \leq 12$, namely $\{S_-^2(s^4), C^2(s^4)\}$, $\{S_-^2(s^6), C^2(s^6)\}$, $\{S_-^3(s^4), C^3(s^4)\}$, and $\{S_-^2(S_-^2(s^3)), C^2(S_-^2(s^3))\}$. (Note that all these pairs of graphs are part of larger LC orbits whose other members are not ELC-preserved.)

Theorem 9. *Let $G = (U \cup W, E)$ be a $(r, n-r)$-bipartite graph with partitions $U = \{u_1, \ldots, u_r\}$ and $W = \{w_1, \ldots, w_{n-r}\}$. Let $S^m(G)$ be the graph*

Table 2: *Classification of bipartite ELC-preserved graphs*

n	
2	s^2
3	s^3
4	s^4
5	s^5
6	s^6, $S^2_-(s^3)$
7	s^7, h^3
8	s^8, $S^2_-(s^4)$, h^3_e
9	s^9, $S^3_-(s^3)$
10	s^{10}, $S^2_-(s^5)$
11	s^{11}
12	s^{12}, $S^2_-(s^6)$, $S^3_-(s^4)$, $S^4_-(s^3)$, $S^2_-(S^2_-(s^3))$
13	s^{13}
14	s^{14}, $S^2_-(s^7)$, $S^2_-(h^3)$, $S^2_+(h^3)$, $H(s^2)$
15	s^{15}, $S^3_-(s^5)$, $S^5_-(s^3)$, h^4
16	s^{16}, $S^2_-(s^8)$, $S^4_-(s^4)$, $S^2_-(S^2_-(s^4))$, $S^2(h^3_e)$, h^4_e

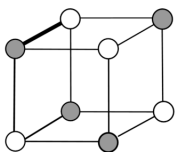


Table 3: *Classification of non-bipartite ELC-preserved graphs*

n	
3	c^3
4	c^4
5	c^5
6	c^6, $C^2(s^3)$
7	c^7
8	c^8, $C^2(s^4)$
9	c^9, $C^3(s^3)$, H_1^2
10	c^{10}, $C^2(s^5)$, H_1^3
11	c^{11}, H_1^4
12	c^{12}, $C^2(s^6)$, $C^3(s^4)$, $C^4(s^3)$, $C^2(S_-^2(s^3))$, H_1^5

where the vertices in U are substituted with e^m. If all vertices in U have odd degree, and all pairs of vertices from U have an even number of (or zero) common neighbors, then $C^m(G) = S^m(G) * w_1 * \cdots * w_{n-r}$, i.e., we can get from $S^m(G)$ to $C^m(G)$ by performing LC on all vertices in W. (The order of the LC operations is not important.)

Proof. Consider performing LC on a vertex w_i in $S^m(G)$. This vertex will be connected to the set X of $m-1$ pendant vertices, and to km other vertices, where k is the degree of w_i in G. Let u be a neighbor of w_i in G, and let Y be the set of m vertices that u is replaced with in $S^m(G)$. The subgraph induced on Y is e^m. After LC on w_i, the induced subgraph on Y will be c^m. Moreover, the induced subgraph on $X \cup \{w_i\}$ will also be c^m, and all vertices in Y will be connected to all vertices of $X \cup \{w_i\}$. Subsequent LC on another vertex w_j, where w_j is also connected to u in G, will change the subgraph induced on Y back to e^m. To ensure that the induced subgraph on Y is c^m in the final graph, we must require u to have odd degree in G. If w_i is also connected to another vertex u' in G, which is replaced by Y' in $S^m(G)$, LC on w_i will connect all vertices in Y to all vertices in Y'. Since we require that u and u' share an even number of neighbors, none of these edges will remain in the final graph. With these considerations, it follows that after performing

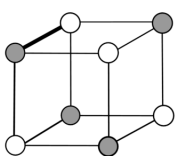
LC on all vertices in W, we obtain a graph where every vertex of G is substituted by c^m, which is the definition of $C^m(G)$. □

New non-bipartite ELC-preserved graphs, h^r_*, of order $n = 2^r$ for $r \geq 4$, can be obtained from the following theorem, by applying specific LC operations to ELC-preserved bipartite graphs corresponding to extended Hamming codes, h^r_e. (Note that $h^3_* = h^3_e$. For $r \geq 4$, h^r_* is a non-bipartite ELC-preserved graph that cannot be obtained from any of our other constructions.)

Theorem 10. *Given the bipartite ELC-preserved graph h^r_e, defined in Theorem 6, LC operations applied, in any order, to all vertices in the partition of size $2^r - r - 1$ preserves the graph, while LC operations applied, in any order, to all vertices in the partition of size $r + 1$ gives an ELC-preserved graph h^r_* which is non-bipartite when $r \geq 4$.*

Proof. Let U denote the set of vertices in the partition of size $r + 1$, and W denote the set of vertices in the partition of size $2^r - r - 1$. After performing LC on all vertices in W, two vertices $u, v \in U$ will be connected by an edge if and only if u and v have an odd number of common neighbors in W. To show that LC on all vertices in W preserves h^r_e, we must show that all pairs of vertices from U have an even number of common neighbors. Let u_e be the extension vertex that was added to h^r to form h^r_e, as described in Theorem 6, and let u_i and u_j be two other vertices in U. The number of neighbors common between u_e and u_i is $\sum_{i=1}^{\lfloor \frac{r}{2} \rfloor} \binom{r}{2i} \frac{2i}{r} = 2^{r-2}$. The number of neighbors common between u_i and u_j is $\sum_{i=2}^{r} \binom{r-2}{i-2} = 2^{r-2}$.

We will now show that LC on all vertices in U transforms h^r_e into the ELC-preserved graph h^r_*. The adjacency matrix of h^r_e can be written $\Gamma = \begin{pmatrix} \mathbf{0}_{r\times r} & P \\ P^{\mathrm{T}} & \mathbf{0}_{(n-r)\times(n-r)} \end{pmatrix}$, where $(I \mid P)$ is the parity check matrix of $\mathcal{C}$, an extended Hamming code. LC on a vertex $u \in U$ can be implemented on Γ by adding row u to all rows in N_u and then changing the diagonal elements $\Gamma_{v,v}$, for all $v \in N_u$, from 1 to 0. After performing LC on all vertices in U, the adjacency matrix of h^r_* is $M = \begin{pmatrix} \mathbf{0} & P \\ P^{\mathrm{T}} & X \end{pmatrix}$. Since each vertex in W has an odd number of neighbors in U, each row of X is the linear combinations of an odd number of rows from P, except that all diagonal elements of X have been changed from 1 to



0. Moreover, the nonzero coordinates of row i of P^T indicates which rows of P were added to form row i of X. It follows that the rows of the matrix $\begin{pmatrix} I & P \\ P^{\mathrm{T}} & X+I \end{pmatrix}$ are the 2^r codewords of $\mathcal{C}^{\perp}$ formed by taking all linear combination of an odd number of rows from $(I \mid P)$, since $(I \mid P)$ contains all odd weight columns from $\mathbb{F}_2^{r+1}$. After performing ELC on an edge $\{u, v\}$ in h_*^r, where $u \in U$ and $v \in W$, and then swapping vertices u and v, we obtain an adjacency matrix $M' = \begin{pmatrix} \mathbf{0} & P' \\ P'^{\mathrm{T}} & X' \end{pmatrix}$. After ELC on an edge $\{u, v\}$ where $u, v \in W$, the vertices in U will no longer be an independent set, but by permuting vertices from U with vertices from N_u or N_v, we can obtain the form M'. We need to show that the rows of $M' + I$ are 2^r codewords of a code equivalent to $\mathcal{C}^{\perp}$ formed by taking linear combinations of an odd number of rows from $(I \mid P')$. Since, according to Theorem 6, the extended Hamming code only has one parity check matrix, up to column permutations, this implies that h_*^r is ELC-preserved. ELC on $\{u, v\}$ is the same as LC on u, followed by LC on v, followed by LC on u again. We have seen that LC corresponds to row additions and flipping diagonal elements. We only need to show that all diagonal elements of M are flipped from 1 to 0 an even number of times to ensure that all rows of $M' + I$ are the codewords described above. If we swap vertices u and v after performing ELC, it follows from the definition of ELC that rows u and v of M' must be the same as in M. As for the other rows, LC on u flips $M_{i,i}$ for $i \in N_u \setminus \{v\}$, LC on v then flips $M_{i,i}$ for $i \in (N_v \cup N_u) \setminus (N_v \cap N_u)$, and finally, LC on u flips $M_{i,i}$ for $i \in N_v \setminus \{v\}$. In total, this means that for each $i \in N_u \cup N_v \setminus \{u, v\}$, the diagonal element $M_{i,i}$ has been flipped from 1 to 0 two times.

The graph h_*^r is non-bipartite if there is at least one pair of vertices from W with an odd number of common neighbors in U. For $r \geq 4$, there must be a pair of vertices from $W_2 \subset W$, the set of $\binom{r}{2}$ vertices of degree 2 in h^r, with no common neighbors in h^r and hence one common neighbor, i.e. the extension vertex, in h_e^r. □

5 ELC-PRESERVED CODES

As we have already shown, the graph h^3 corresponds to the $[7, 4, 3]$ *Hamming code*, and its dual $[7, 3, 4]$ *simplex code*. The graph h_e^3 corresponds to the self-dual $[8, 4, 4]$ extended Hamming code. The star graph s^n

corresponds to the $[n, 1, n]$ *repetition code*, and its dual $[n, n-1, 2]$ code. We can obtain larger ELC-preserved bipartite graphs using Hamming expansion or star expansion, and the parameters of the corresponding codes are given by the following theorems.

Theorem 11. *$H(G)$, for G a connected ELC-preserved $(r, k-r)$-bipartite graph on $k \geq 2$ vertices, corresponds to a $[7k, 3k+r, 4]$ code $\mathcal{C}$, and to the dual $[7k, 4k-r, 4]$ code $\mathcal{C}^{\perp}$.*

Proof. From the construction of $H(G)$, we get that $\mathcal{C}$ must have length $n = 7k$. The codes $\mathcal{C}$ and $\mathcal{C}^{\perp}$ have dimension $3k+r$ and $4k-r$, respectively, since $H(G)$ has partitions of size $3k+r$ and $4k-r$ when G has partitions of size r and $k-r$. That both $\mathcal{C}$ and $\mathcal{C}^{\perp}$ have minimum distance 4 follows from the fact that the minimum vertex degree in both partitions of $H(G)$ is 3. This is verified by observing that the subgraph h_0, shown in Fig. 5, has one vertex w_6 of degree 3, and three vertices w_3, w_4, and w_5 of degree 3, belonging to different partitions. Moreover, the degrees of w_0, w_1, and w_2 must be at least 5, since G is connected. □

Theorem 12. *Let G be a connected ELC-preserved $(r, k-r)$-bipartite graph on $k \geq 2$ vertices and assume, without loss of generality, that $r \leq k-r$. Let G correspond to a $[k, r, d]$ code and its dual $[k, k-r, d']$ code. Then $S_+^m(G)$ corresponds to an $[mk, r, md]$ code and its dual $[mk, mk-r, 2]$ code. $S_-^m(G)$ corresponds to an $[mk, k-r, md']$ code and its dual $[mk, mk-k-r, 2]$ code.*

Proof. From the construction of $S^m(G)$, we get that all the codes must have length $n = mk$. In G, the minimum vertex degree in the partition of size r must be $d-1$, and the minimum vertex degree in the other partition must be $d'-1$. In $S_+^m(G)$, $k-r$ vertices of G have been substituted by e^m and m pendants have been added to the other r vertices. Hence, $S_+^m(G)$ must contain a partition of size r with minimum vertex degree $md-1$, since the vertex of degree $d-1$ in G is now connected to $d-1$ copies of e^m plus $m-1$ pendants. The other partition of $S_+^m(G)$ has size $mk-r$, and contains pendants, i.e., vertices of degree one. By similar argument, $S_-^m(G)$ has a partition of size $k-r$ with minimum vertex degree $md'-1$ and a partition of size $mk-k-r$ with minimum vertex degree one. The dimensions and minimum distances of the corresponding codes follow. □

We observe that the ELC-preserved graphs h_e^3 and $H(s^2)$ correspond to $[8, 4, 4]$ and $[14, 7, 4]$ self-dual codes. A natural question to ask is whether there are other ELC-preserved self-dual codes. All self-dual

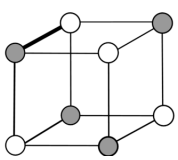


Table 4: *ELC orbit size of graphs corresponding to self-dual codes*

n	d	Codes	ELC-preserved	Size two ELC orbits
8	≥ 4	1	1	-
10	≥ 4	-	-	-
12	≥ 4	1	-	1
14	≥ 4	1	1	-
16	≥ 4	2	-	1
18	≥ 4	2	-	-
20	≥ 4	6	-	1
22	≥ 4	8	-	-
24	≥ 4	26	-	2
26	≥ 4	45	-	-
28	≥ 4	148	-	1
30	≥ 4	457	-	-
32	≥ 4	2523	-	2
34	≥ 6	938	-	-

binary codes of length $n \leq 34$ have been classified by Bilous and van Rees [27, 28]. A database containing one representative from each equivalence class of codes with $n \leq 32$ and $d \geq 4$, and one representative from each equivalence class of codes with $n = 34$ and $d \geq 6$ is available on-line at `http://www.cs.umanitoba.ca/~umbilou1/`. We have generated the ELC orbits of all the corresponding bipartite graphs, and found that h_e^3 and $H(s^2)$ are the only ELC-preserved graphs, as shown in Table 4. However, as the following theorem shows, we can construct an infinite number of ELC-preserved self-dual codes with $n \geq 56$ by iterated Hamming expansion of h_e^3 and $H(s^2)$.

Theorem 13. *Let $H^r(G) = H(\cdots H(G))$ denote the r-fold Hamming expansion of G. Then for $r \geq 1$, $H^r(h_e^3)$ corresponds to an ELC-preserved self-dual $[8 \cdot 7^r, 4 \cdot 7^r, 4]$ code, and $H^{r+1}(s^2)$ corresponds to an ELC-preserved self-dual $[2 \cdot 7^{r+1}, 7^{r+1}, 4]$ code.*

Proof. The parameters of the codes follows from Theorem 11. It remains to show that they are self-dual. A code with generator matrix $(I \mid P)$ is self-dual if the same code is also generated by $(P^{\mathrm{T}} \mid I)$, i.e., if $P^{-1} = P^{\mathrm{T}}$. The codes associated with both h_e^3 and $H(s^2)$ have the property that $P = P^{\mathrm{T}}$, and Hamming expansion must preserve this symmetry since it has the same effect on both partitions of the graph. In general, $P = P^{\mathrm{T}}$ only implies that a code is isodual, but we can prove a stronger property in this case. Note that P corresponding to $H(G)$ will have full rank when P corresponding to G has full rank, since we know that P corresponding to $H(s^2)$, which is the Hamming expansion of the induced subgraph on any pair of vertices connected by an edge in G, has full rank. Since an ELC-preserved code only has one generator matrix, up to column permutations, and the inverse of a symmetric matrix is symmetric, we must have that $P^{-1}(I \mid P) = (P \mid I)$. Hence the code is self-dual. □

6 Orbits of Size Two

ELC-preserved codes with good properties could have practical applications in iterative decoding [17–20]. However, there seem to be extremely few such codes, and, except for the perfect Hamming codes, graphs arising from the constructions in Section 3 correspond to $[n, k, d]$ codes with either low minimum distance d or low rate $\frac{k}{n}$, compared to the best known codes of the same length. Iterative decoding with ELC also works for graphs with larger ELC orbits, such as *quadratic residue codes* [18], and has performance close to that of iterative permutation decoding [23] for graphs with small ELC orbits, such as the *extended Golay code* [18]. The self-dual $[24, 12, 8]$ extended *Golay code* corresponds to a bipartite graph with an ELC orbit of size two. As a generalization of ELC-preserved graphs, we therefore briefly consider graphs with ELC orbit of size two. The number of size two orbits are listed in Table 5. We have also counted LC orbits of size two. Clearly there is an LC orbit $\{s^n, c^n\}$ for all $n \geq 3$. The only other size two LC orbit we find for $n \leq 12$ is comprised of the two graphs of order six depicted in Fig. 8 (These two graphs correspond to the self-dual additive *Hexacode* over $\mathbb{F}_4$ [7].)

We have also looked at the ELC orbits corresponding to self-dual codes of length $n \leq 34$, as seen in Table 4. Except for the $[24, 12, 8]$ extended Golay code and a $[32, 16, 4]$ code, the remaining self-dual codes in this table with ELC orbits of size two, all with minimum distance four, can be constructed by the following theorem. It remains

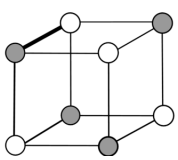


Table 5: *Number of orbits of size two*

n	Bipartite ELC	Non-bipartite ELC	LC
3	-	-	1
4	1	1	1
5	2	3	1
6	4	9	2
7	6	10	1
8	9	21	1
9	12	22	1
10	22	43	1
11	22	41	1
12	33	91	1
13	35		
14	53		
15	48		

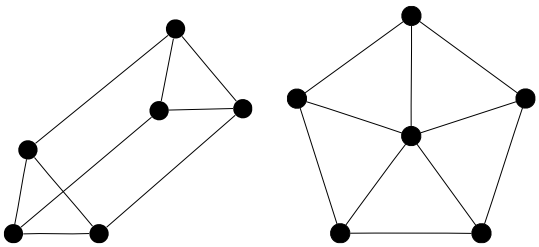


Fig. 8: *LC orbit of size two*

an open problem to devise a general construction for self-dual codes with ELC orbits of size two and minimum distance greater than four.

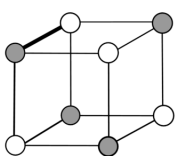
Theorem 14. *Let G be a $(2m, 2m)$-bipartite graph on $4m$ vertices, where $m \geq 3$. Let the vertices in one partition be labeled $v_1, v_2, \ldots, v_{2m}$, and the vertices in the other partition be labeled $w_1, w_2, \ldots, w_{2m}$. Let there be an edge $\{v_i, w_j\}$ whenever $i \neq j$. Then G has an ELC orbit of size two and corresponds to a self-dual $[4m, 2m, 4]$ code $\mathcal{C}$.*

Proof. The code $\mathcal{C}$ has generator matrix $(I \mid P)$ where P is circulant with first row $(01 \cdots 1)$. It can be verified that $P^{-1} = P = P^{\mathrm{T}}$ when P is of this form with even dimensions. Hence $P^{-1}(I \mid P) = (P^{\mathrm{T}} \mid I)$ and $\mathcal{C}$ is self-dual. (An (m, m)-bipartite graph constructed as above for odd $m \geq 7$ would still have an ELC orbit of size two but would correspond to a non-self-dual $[2m, m, 4]$ code.) Note that $m = 1$ and $m = 2$ must be excluded, since they produce the ELC-preserved graphs s^2 and h_e^3, respectively.

Due to the symmetry of G we only need to consider ELC on one edge $\{v_i, w_j\}$. This will take us to a graph G' where the neighborhoods of v_i, v_j, w_i, w_j are unchanged, but where $N_{v_k} = \{v_i, v_j, w_k\}$ and $N_{w_k} = \{w_i, w_j, v_k\}$, for all $k \neq i, j$. We need to consider ELC on three types of edges in G'. ELC on $\{v_i, w_j\}$ or $\{v_j, w_i\}$ will take us back to G. ELC on an edge $\{v_k, w_k\}$ will preserve G', since it simply removes edges $\{v_i, w_j\}$ and $\{v_j, w_i\}$ and adds edges $\{v_i, w_i\}$ and $\{v_j, w_j\}$, thus in effect swapping vertices v_i and v_j. Finally, ELC on an edge $\{v_i, w_k\}$ also preserves G', since it swaps the roles of vertices v_j and v_k. (ELC on $\{v_k, w_i\}$ similarly swaps w_j and w_k.) This can be seen by noting that w_k has neighbors v_j and v_k, with v_k being connected to w_j in $N_{v_i}^{w_k}$ and v_j being connected to all vertices in $N_{v_i}^{w_k}$ except w_j. Hence these relations are reversed after complementation. Furthermore, $N_{v_k} \setminus N_{v_i}^{w_k} = N_{v_j} \setminus N_{v_i}^{w_k} = \{w_k, w_i\}$, so isomorphism is preserved. We have shown that the ELC orbit of G has size two. Since the minimum vertex degree over the ELC orbit is 3, the minimum distance of $\mathcal{C}$ is 4. □

7 Conclusions

We have introduced ELC-preserved graphs as a new class of graphs, found all ELC-preserved graphs of order up to 12 and all ELC-preserved bipartite graphs of order up to 16, and shown how all these graphs arise from general constructions. It remains an open problem to prove



that all ELC-preserved graphs arise from these constructions, or give an example to the contrary. We therefore pose the question: Is a connected ELC-preserved graph of order n always either s^n, where n is prime, H_k^m, where $n = 7k + m$, h^r, where $n = 2^r - 1$, h_e^r or h_*^r, where $n = 2^r$, or can it be obtained as $S_+^m(G)$, $S_-^m(G)$, or $C^m(G)$, where G is an ELC-preserved graph of order $\frac{n}{m}$, or $H(G)$, where G is an ELC-preserved graph of order $\frac{n}{7}$? (Note that not all star graphs and complete graphs are primitive ELC-preserved graphs, since most of them can be obtained as follows. From the graph e^1, we can obtain all $c^n = C^n(e^1)$. From $s^2 = C^2(e^1)$, we obtain all $s^n = S^{\frac{n}{2}}(s^2)$ where n is even. More generally, for $n = pq$ a composite number, $s^n = S_+^p(s^{\frac{n}{q}}) = S_+^q(s^{\frac{n}{p}})$, so only s^p with p an odd prime is a primitive ELC-preserved graph.)

Another challenge is to enumerate or classify ELC-preserved graphs of order $n > 12$ and ELC-preserved bipartite graphs of order $n > 16$. Our classification used a previous complete classification of ELC orbits [16], and a graph extension technique to obtain all bipartite ELC-preserved graphs of order 16. Perhaps the complexity of classification could be reduced by further exploiting restrictions on the structure of ELC-preserved graphs.

ELC-preserved graphs are an interesting new class of graphs from a theoretical point of view. As discussed in Section 1, LC and ELC orbits of graphs show up in many different fields of research, and ELC-preserved graphs may also be of interest in these contexts. We have seen that one possible use for bipartite ELC-preserved graphs is in iterative decoding of error-correcting codes. Hamming codes are perfect, but for this application we would like codes with rate $\frac{k}{n} \approx \frac{1}{2}$. Such ELC-preserved codes obtained from our constructions do not have minimum distance that can compete with the best known codes of similar length, except for the optimal $[8, 4, 4]$ code (h_e^3), for which iterative decoding has been simulated with good results [17], and the optimal $[14, 7, 4]$ code ($H(s^2)$). Longer codes obtained from Hamming expansion will always have minimum distance 4, as shown in Theorem 11. Codes that have a negligible number of low weight codewords can still have good decoding performance, but the number of weight 4 codewords in these codes grows linearly with the length, since the number of degree 3 vertices in the corresponding graphs does so, and hence the codes are not well suited for this application. It is therefore interesting to consider ELC orbits of size two, one of them corresponding the extended Golay code of length 24, for which iterative decoding with ELC has been simulated with good results [18]. For codes of higher length, however,

this criteria is probably also too restrictive. Graphs with ELC orbits of bounded size could be more suitable for this application, and would be interesting to study from a graph theoretical point of view. For some graphs, ELC on certain edges will preserve the graph, while ELC on other edges may not. Iterative decoding where only ELC on the subset of edges that preserve the graph are allowed has been studied [17]. Graphs where ELC on certain edges preserve the number of edges in the graph, or keep the number of edges within a given bound, have also been considered in iterative decoding [19]. ELC-preserved graphs are clearly a subclass of the graphs where all ELC orbit members have the same number of edges. This class of graphs, and other possible generalizations of ELC-preserved graphs, would be interesting to study further.

ACKNOWLEDGEMENTS This research was supported by the Research Council of Norway.

REFERENCES

[1] FON-DER FLAAS, D. G.: On local complementations of graphs. In *Combinatorics (Eger, 1987)*, vol. 52 of *Colloq. Math. Soc. János Bolyai*, pp. 257–266. North-Holland, Amsterdam, 1988.

[2] BOUCHET, A.: Graphic presentations of isotropic systems. *J. Combin. Theory Ser. B* 45(1), 58–76, 1988.

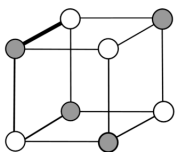
[3] DE FRAYSSEIX, H.: Local complementation and interlacement graphs. *Discrete Math.* 33(1), 29–35, 1981.

[4] HEIN, M., EISERT, J., BRIEGEL, H. J.: Multi-party entanglement in graph states. *Phys. Rev. A* 69(6), 062 311, 2004.

[5] VAN DEN NEST, M., DEHAENE, J., DE MOOR, B.: Graphical description of the action of local Clifford transformations on graph states. *Phys. Rev. A* 69(2), 022 316, 2004.

[6] CALDERBANK, A. R., RAINS, E. M., SHOR, P. M., SLOANE, N. J. A.: Quantum error correction via codes over GF(4). *IEEE Trans. Inform. Theory* 44(4), 1369–1387, 1998.

[7] DANIELSEN, L. E., PARKER, M. G.: On the classification of all self-dual additive codes over GF(4) of length up to 12. *J. Combin. Theory Ser. A* 113(7), 1351–1367, 2006.



[8] Riera, C., Parker, M. G.: On pivot orbits of Boolean functions. In *Proc. Fourth International Workshop on Optimal Codes and Related Topics*, pp. 248–253. Bulgarian Acad. Sci. Inst. Math. Inform., Sofia, 2005.

[9] Riera, C., Parker, M. G.: Generalised bent criteria for Boolean functions (I). *IEEE Trans Inform. Theory* 52(9), 4142–4159, 2006.

[10] Arratia, R., Bollobás, B., Sorkin, G. B.: The interlace polynomial of a graph. *J. Combin. Theory Ser. B* 92(2), 199–233, 2004.

[11] Aigner, M., van der Holst, H.: Interlace polynomials. *Linear Algebra Appl.* 377, 11–30, 2004.

[12] Arratia, R., Bollobás, B., Coppersmith, D., Sorkin, G. B.: Euler circuits and DNA sequencing by hybridization. *Discrete Appl. Math.* 104, 63–96, 2000.

[13] Danielsen, L. E., Parker, M. G.: Interlace polynomials: Enumeration, unimodality, and connections to codes. *Discrete Appl. Math.* 158(6), 636–648, 2010.

[14] Bouchet, A.: Circle graph obstructions. *J. Combin. Theory Ser. B* 60(1), 107–144, 1994.

[15] Geelen, J., Oum, S.-i.: Circle graph obstructions under pivoting. *J. Graph Theory* 61(1), 1–11, 2009.

[16] Danielsen, L. E., Parker, M. G.: Edge local complementation and equivalence of binary linear codes. *Des. Codes Cryptogr.* 49, 161–170, 2008.

[17] Knudsen, J. G., Riera, C., Parker, M. G., Rosnes, E.: Adaptive soft-decision iterative decoding using edge local complementation. In *Proc. Second International Castle Meeting on Coding Theory and Applications, LNCS 5228*, pp. 82–94. Castillo de la Mota, Medina del Campo, Spain, 2008.

[18] Knudsen, J. G., Riera, C., Danielsen, L. E., Parker, M. G., Rosnes, E.: Iterative decoding on multiple Tanner graphs using random edge local complementation. In *Proc. IEEE Int. Symp. Inform. Theory*, pp. 899–903. 2009.

[19] Knudsen, J. G., Riera, C., Danielsen, L. E., Parker, M. G., Rosnes, E.: On iterative decoding of HDPC codes using weight-bounding graph operations. In *Proc. Int. Zürich Seminar on Com-*

munications, pp. 98–101. Zürich, Switzerland, Mar. 2010.

[20] Knudsen, J. G., Riera, C., Danielsen, L. E., Parker, M. G., Rosnes, E.: Improved adaptive belief propagation decoding using edge-local complementation. In *Proc. IEEE Int. Symp. Inform. Theory*, pp. 774–778. Austin, Texas, Jul. 2010.

[21] Curtis, R. T.: On graphs and codes. *Geom. Dedicata* 41(2), 127–134, 1992.

[22] Parker, M. G., Rijmen, V.: The quantum entanglement of binary and bipolar sequences. In *Sequences and Their Applications – SETA '01*, Discrete Math. Theor. Comput. Sci., pp. 296–309. Springer, London, 2002.

[23] Halford, T. R., Chugg, K. M.: Random redundant iterative soft-in soft-out decoding. *IEEE Trans. Commun.* 56(4), 513–517, 2008.

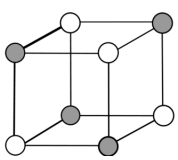
[24] Bollobás, B.: *Modern graph theory*, vol. 184 of *Graduate Texts in Mathematics*. Springer, New York, 1998.

[25] Ellis-Monaghan, J. A., Sarmiento, I.: Distance hereditary graphs and the interlace polynomial. *Combin. Probab. Comput.* 16(6), 947–973, 2007.

[26] Pless, V. S., Huffman, W. C. (eds.): *Handbook of Coding Theory*. North-Holland, 1998, Amsterdam.

[27] Bilous, R. T., van Rees, G. H. J.: An enumeration of binary self-dual codes of length 32. *Des. Codes Cryptogr.* 26, 61–86, 2002.

[28] Bilous, R. T.: Enumeration of the binary self-dual codes of length 34. *J. Combin. Math. Combin. Comput.* 59, 173–211, 2006.



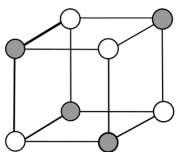
6 Future Work

- **SPA-PD & MBBP**
  Preprocess a code (using WB-ELC) so as to obtain several structurally distinct graphs of reduced (or minimum) weight, and run SPA-PD on each of these in turn, changing graph periodically to increase diversity. Alternatively, starting from a reduced-weight graph within a sub-orbit of some size, the decoder can use random WB-ELC to change structure, thus alleviating the need for any stored graphs.

- **WB-ELC at depth > 2**
  How does WB-ELC extend to cases involving edges at a distance of more than 2? Motivated by the preserved locality of depth 2 WB-ELC (i.e., that the distance is at most 2), and the number of isomorphic cases (which can be ignored), it seems feasible to identify all possible cases for higher depths. Such an extension might help clarify the relationship between distance (locality), depth and girth of the corresponding graph.

- **Generalized ELC**
  The operation of ELC on a Tanner graph implies its operation on a bipartite simple graph, associated with a systematic parity-check matrix. However, as ELC amounts to elementary row additions on H, there are several possible generalizations of ELC to the nonsystematic case. We have identified one such generalized ELC (GELC) operation, which reduces to ELC when H is in systematic form, and have begun exploring how GELC relates to $\mathrm{Aut}(\mathcal{C})$. For instance, an interesting question is a generalized notion of triviality to the nonsystematic case (where a permutation can be implemented by row permutations alone). Our current work strongly suggests a one-to-one relationship between $\mathrm{Aut}(\mathcal{C})$ and GELC operations (as for ELC), and that $\mathcal{D}$ is always a group. Furthermore, many properties of ELC extend naturally to GELC, such as GELC-preserved graphs, and a partitioning of $\mathcal{K}$ into subsets according to the minimum length of the corresponding iso-GELC sequence.

- **Truly local WB-ELC**
  As an alternative to using the WB-ELC algorithm to maintain a bound on the weight of $\mathcal{G}$, it is possible to determine code-specific



rules which identify whether an edge will *preserve* $|\mathcal{G}|$, if subject to an ELC. Such rules are based on the edges of a specific graph, and use only locally available information, so this corresponds to a distributive implementation of WB-ELC for $T = 0$ (if we ignore those WB-ELC operations which *reduce* the weight). For instance, such rules may be derived for the strongly structured (extended) Golay code, where the aim is to remain within the iso-orbit of the lowest-weight graph in the orbit of the code (which contains only two nonisomorphic graphs). The node initiating the process may keep an ongoing record of those adjacent edges on which it is *permitted* (from a weight-bounding perspective) to perform an ELC by passing messages to the corresponding incident nodes which return a description of their neighborhoods. The information in these messages may also be used to implement the ELC operation itself – facilitating a *truly local* implementation of ELC (where a node only knows its adjacent edges). Tentative data indicate that the complexity of this message passing approach is of comparable complexity to that of the WB-ELC algorithms proposed in this thesis, but is only suited to certain codes which exhibit significant symmetries.