REPORTS IN INFORMATICS

ISSN 0333-3590

Random Edge-Local Complementation With Applications to Iterative Decoding of HDPC Codes

Joakim Grahl Knudsen and
Constanza Riera and Lars Eirik Danielsen
and Matthew G. Parker and Eirik Rosnes

REPORT NO 395 August 2010

Department of Informatics

UNIVERSITY OF BERGEN

Bergen, Norway

This report has URL http://www.ii.uib.no/publikasjoner/texrap/ps/2010-395.ps

Reports in Informatics from Department of Informatics, University of Bergen, Norway, is available at http://www.ii.uib.no/publikasjoner/texrap/.

Requests for paper copies of this report can be sent to:

Department of Informatics, University of Bergen, Høyteknologisenteret,
P.O. Box 7800, N-5020 Bergen, Norway

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

Department of Informatics
University of Bergen
N-5020 Bergen
Norway

August 31, 2010

Abstract

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 [31]. The advent of turbo codes in 1993 [3] and the rediscovery of low-density parity-check (LDPC) codes at around the same time [29] (although LDPC codes were actually invented in 1962 [12] and re-discovered once

already, in 1981 [33]) 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 [33]. 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, $d_{\min}$, 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 $d_{\min}$ 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 $d_{\min}(\mathcal{C}^{\perp})$. It is known that many families of codes – specifically BCH and RS codes – do not have Tanner graphs without cycles of length 4 [15]. Furthermore, these codes do typically not have sparse duals [34], 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 suboptimal 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) [10, 27]. 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 [21, 22]. 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) [20]. One state-of-the-art decoder for HDPC codes, the iterative permutation decoder (SPA-PD) [14], 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 [9, 14, 25], over the additive white Gaussian noise (AWGN) channel. Also, alternatively or in conjunction [9, 24], 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 [18, 25].

This paper describes the pseudorandom use of a simple graph operation known as edge-local complementation (ELC) [4, 7] to improve the performance of iterative decoding [23, 26]. 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 $d_{\min}$ is denoted by $[n, k, d_{\min}]$, where $d_{\min}$ 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, $a_i = 0,\ i < d_{\min}$. 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 $v_0 \to u_1,\ v_1 \to u_2,\ v_2 \to u_0,\ v_3 \to u_4$, and $v_4 \to u_3$, while $v_5 \to u_5$. 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 $\mathrm{Aut}(\mathcal{C}) = \mathrm{Aut}(\mathcal{C}^\perp)$ [19], 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 [14]. 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

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

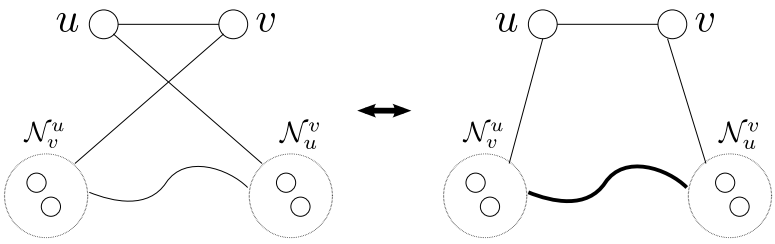


Figure 1: ELC on edge (u, v) of a bipartite simple graph. Curved links indicate arbitrary edges. Bold links mean that the edges connecting the two sets have been complemented; edges are replaced by nonedges, and vice versa. This graph may be a subgraph of a larger graph, in which case the rest of the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

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 $\mathcal{N}_v \backslash \{u\}$. 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 $\{u', v'\}$.

2 Edge-Local Complementation

ELC is defined on an edge of a simple graph, $(u, v) \in \mathcal{G}$ [4]. We consider only bipartite graphs in this work, which simplifies the description. ELC on an edge

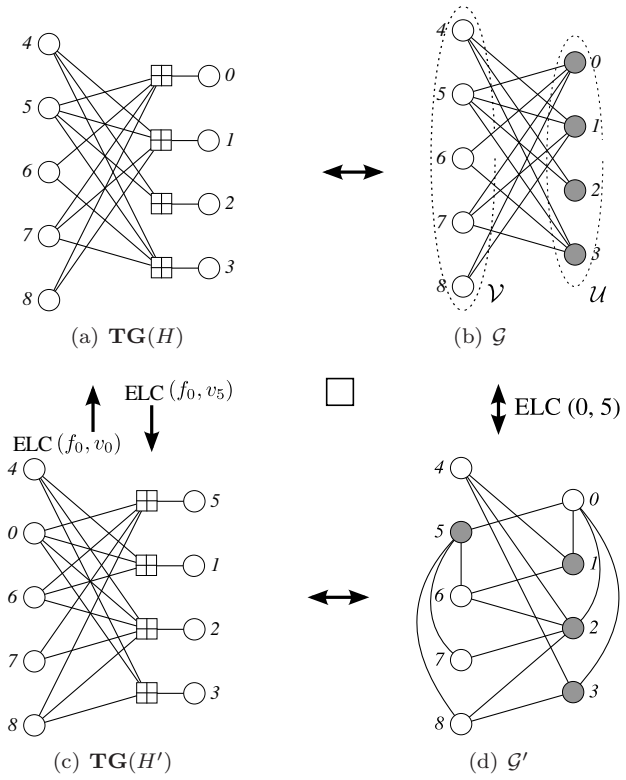


Figure 2: Example of ELC on a small $[9, 4, 4]$ code, showing also the corresponding Tanner graphs. White and grey nodes correspond to $\mathcal{V}$ and $\mathcal{U}$, respectively.

(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.

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

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.

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

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*

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

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, 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, orbit($\mathcal{C}$). Structural distinctness is in terms of graph isomorphism. By using the software package Nauty [30], 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, 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) [8].

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* $\mathbf{e}$ *is* $0 \leq l \leq \min(n - k,\ k)$*. Depending on* H*, the sequence* $\mathbf{e}$ *may not be unique, so equivalent sequences may be derived from* $\mathcal{P}$ *and* $\mathcal{P}'$*.*

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**

Proof: ELC generates the entire orbit [7], and in particular all systematic parity-check matrices for the corresponding code, so such a sequence $\mathbf{e}$ 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 [5]. ELC has the effect of swapping exactly one pair of positions between $\mathcal{I}$ and $\mathcal{P}$, so the length of $\mathbf{e}$ 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 $(\text{row}(r), s) \in \mathbf{TG}(H)$. When several valid choices of r exist for a coordinate $s \in \mathcal{S}$, a branch point arises in the algorithm leading to an 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 $(\mathrm{row}(1), 13) = (1, 13) \in \mathbf{TG}(H)$. 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

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

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* [24].

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 [5].

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

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

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 $\sigma_2 = \theta_2 \circ \sigma \in \mathcal{D}_{\alpha(H)} \circ \sigma$ 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, $|\mathcal{K}^l_H| = |\mathcal{K}^l_{\sigma(H)}|,\ 0 \leq l \leq \min(n-k,\ k)$ and $\sigma \in \mathrm{Aut}(\mathcal{C})$,

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)$\rangle$

and we call the set $\{|\mathcal{K}_H^l|\},\ 0 \le l \le \min(n-k,\ k)$, the *profile* of the iso-orbit of H. This profile varies with H, but is invariant over the iso-orbit of H.

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*

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

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*

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	-	-	-	-

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

another rare code with only two structures in the orbit

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

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

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*

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

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 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 [14]), 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 $\mathbf{e}$ 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*

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

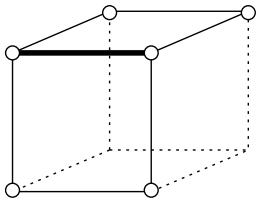
This is due to the perfect symmetry of $\mathcal{G}$*, in which all local neighborhoods are identical, forming a cube – see Fig. 3(a).*

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).

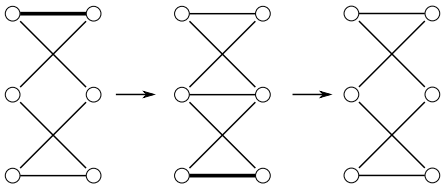
Example 10 *For the "bow-tie" graph, consisting of a single 6-cycle, an isomorphism is found in the application of ELC to any of the three pairs of diametrically opposite edges, inserting and removing a chord (i.e., a 4-cycle), as shown in Fig. 3(b). As expected, the iso-orbit of the corresponding* $[6, 3]$ *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 [24]. 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.

Figure 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*

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

Proof: ELC on (u, v) complements the edges between $\mathcal{N}^v_u$ and $\mathcal{N}^u_v$, 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

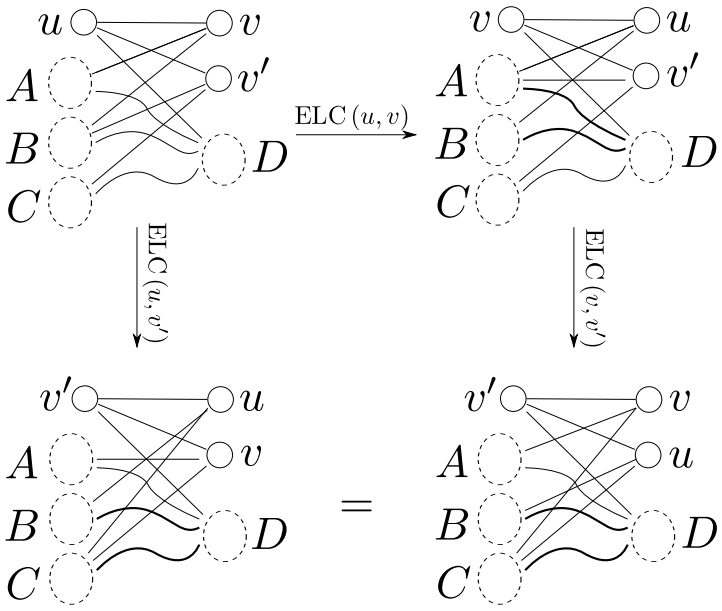


Figure 4: Proof of Lemma 2. ELC on adjacent edges, e.g. (u, v) and (v, v'), will always result in the exact same graph as a single ELC on the second edge, (u, v'). Note that, due to the swap of nodes u and v in the first ELC, the second ELC edge is labeled (v, v') – however, this is the same edge as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

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.

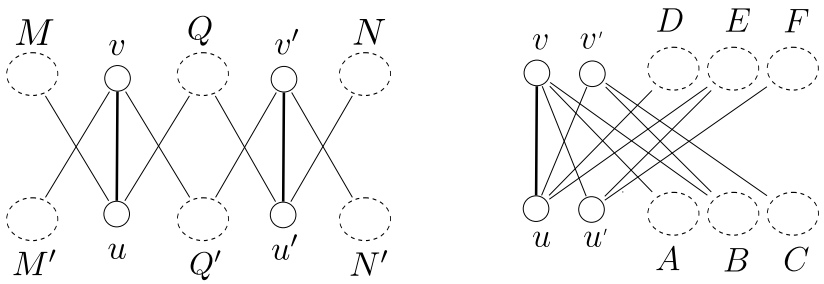
Lemma 2 (Adjacent edges [2]) *ELC on* $\{(u, v), (v, v')\}$*, where* $v' \in \mathcal{N}_u^v$*, gives the same graph as ELC on* (u, v')*.*

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.



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

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

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 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, similarly, $Q' = \mathcal{N}_{v'}^{u'} \cap \mathcal{N}_v^u \neq \emptyset$.

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, the sequence

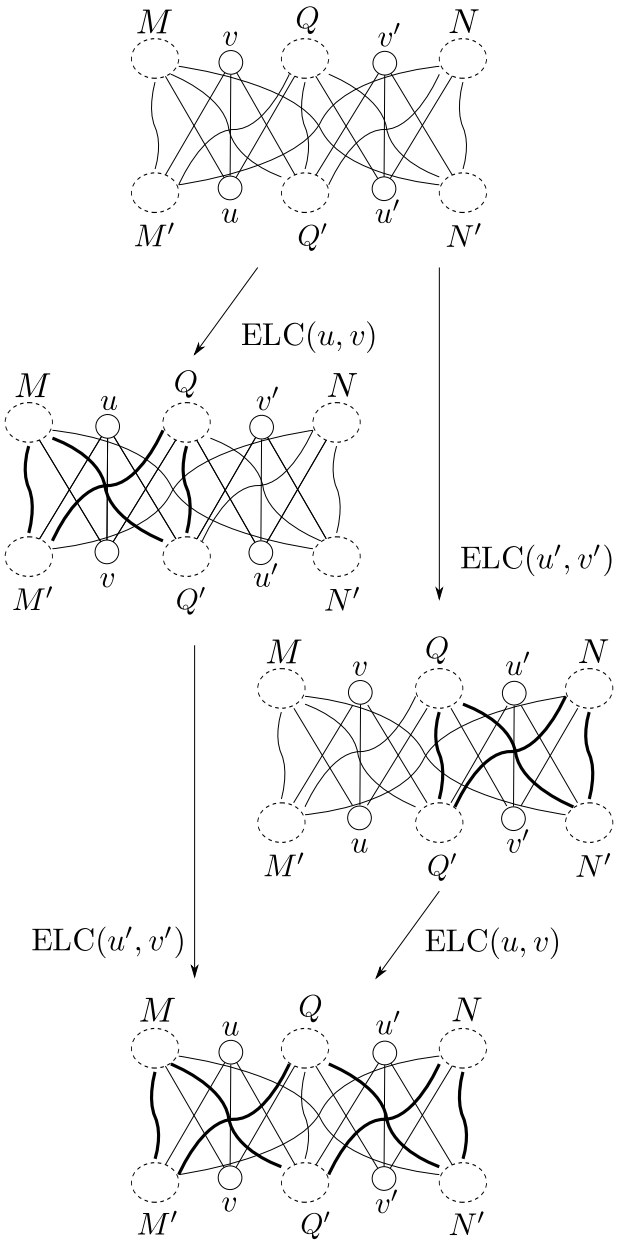


Figure 6: Proof of Theorem 5. A special case of independence gives the equivalent sequence, $(u', v'), (u, v)$; although the local subgraphs $\mathcal{E}_{u,v}$ and $\mathcal{E}_{u',v'}$ are *not* independent, the overlap is confined to Q and Q' (which is complemented twice) [5].

$\{(u',v'),(u,v)\}$ gives the same graph as the sequence $\{(u,v),(u',v')\}$ [5]. 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*

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

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 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 $\{(u',v),(u,v')\}$ [5], 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 $\{(u,v'),(u',v)\}$ [5], 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.

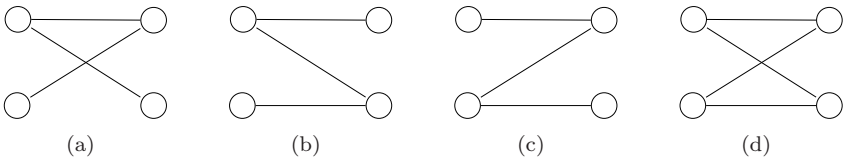


Figure 7: Figs. 7(a) - 7(c) represent special cases of ELC equivalences [5], where the first ELC is on the upper edge and the second ELC is on the bottom edge. In Fig. 7(a), this edge results from the first ELC. In the same way, Fig. 7(d) shows where double ELC is not possible (the second edge is removed by the first ELC).

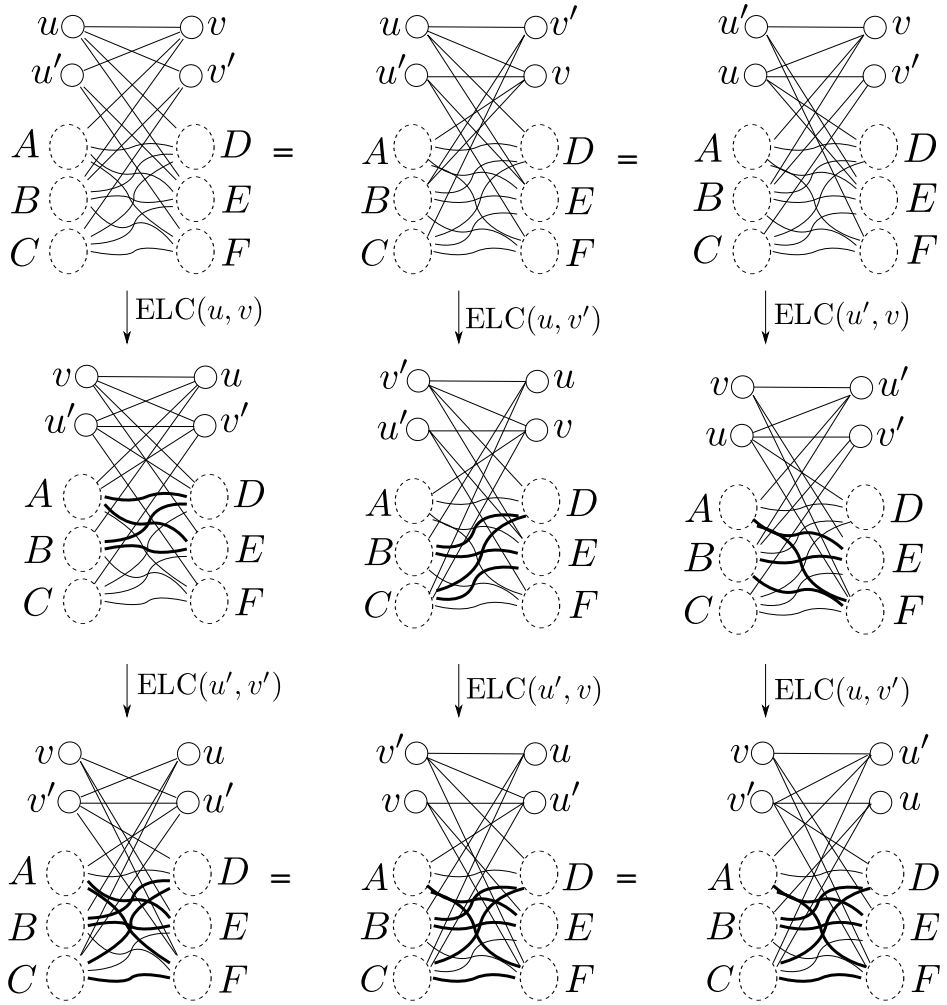


Figure 8: Proof of Theorem 6.

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

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

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 [16], and an extremal double circulant self-dual $[68,34,12]$ code ("$C_{68,1}$") from [13]. 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,

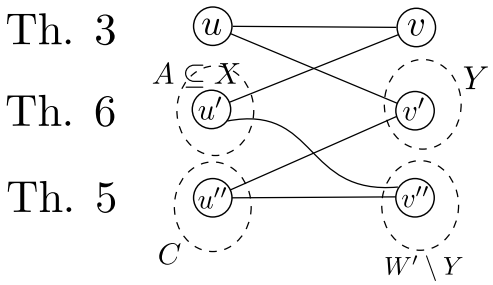


Figure 9: Illustration of Alg. 2, where $X := \mathcal{N}_v^u$, $Y := \mathcal{N}_u^v$, and $W := \mathcal{N}_{u'}^v$. Note that the edge (u', v') will result from the first ELC on (u, v). The curved line indicates the possibility of an edge (u', v'').

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 $T \geq$ -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 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.

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

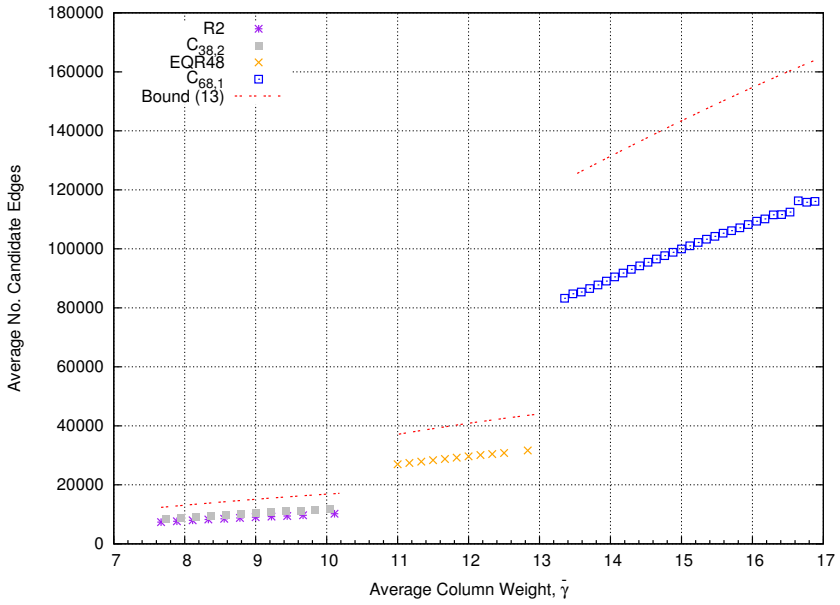


Figure 10: Complexity simulations for WB_ELC$_{(a)}$ compared against the bound of (13) (dashed lines). Each code is simulated over its entire weight range, from the minimum weight to the maximum weight encountered by ELC (which is around 50% weight), so the complexity of WB_ELC$_{(a)}$ is completely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

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 $\bar{\gamma}$.

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).

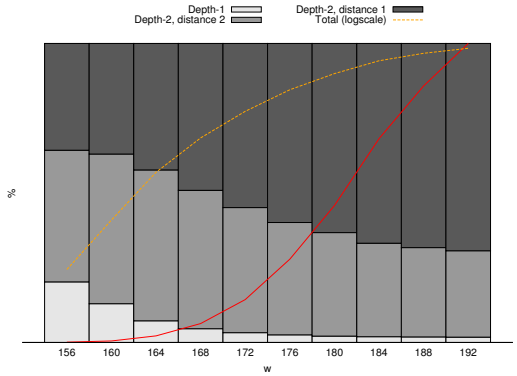
For enumeration purposes, it may also be interesting to enumerate only those WB-ELC operations involving a specific node, $v^\star$. By restricting the operation of Alg. 2 to $v^\star$ only, rather than $\mathcal{V}$, the single iteration of the outer loop returns the subset of operations which include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

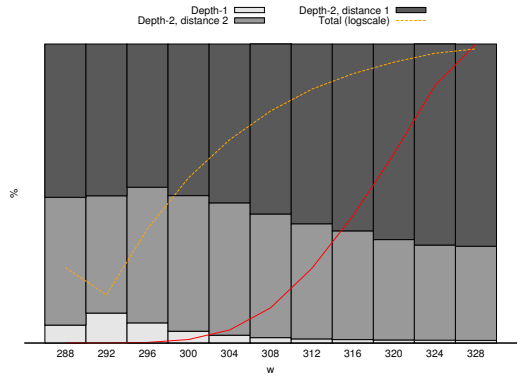
The WB-ELC algorithm can be used to reduce the weight of a bipartite graph. By definition (3), the parity-check matrix related to $\mathcal{G}$ is in systematic form. The purpose of reducing the weight of a systematic matrix may be motivated by the SPA-WBELC algorithm described in Section 5, but is in 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 [20].

We define instance (b) of Alg. 2, denoted WB_ELC$_{(\mathrm{b})}$, as one which terminates upon the first encountered WB-ELC operation to satisfy the threshold, T. The returned

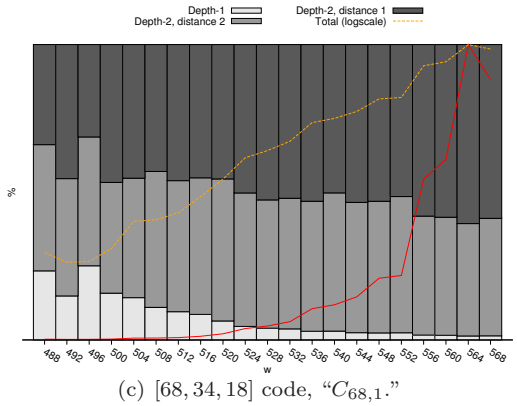
[5] The distribution of $|\mathcal{G}|$ usually resembles a normal distribution, so g is to ensure a minimum number of observations are achieved at the tails (high and low ends). The total number of graphs, $\sum \mathbf{d}(|\mathcal{G}|)$, is much greater than $10\,(|\mathcal{G}|_{\max} - |\mathcal{G}|_{\min})$.



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



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



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

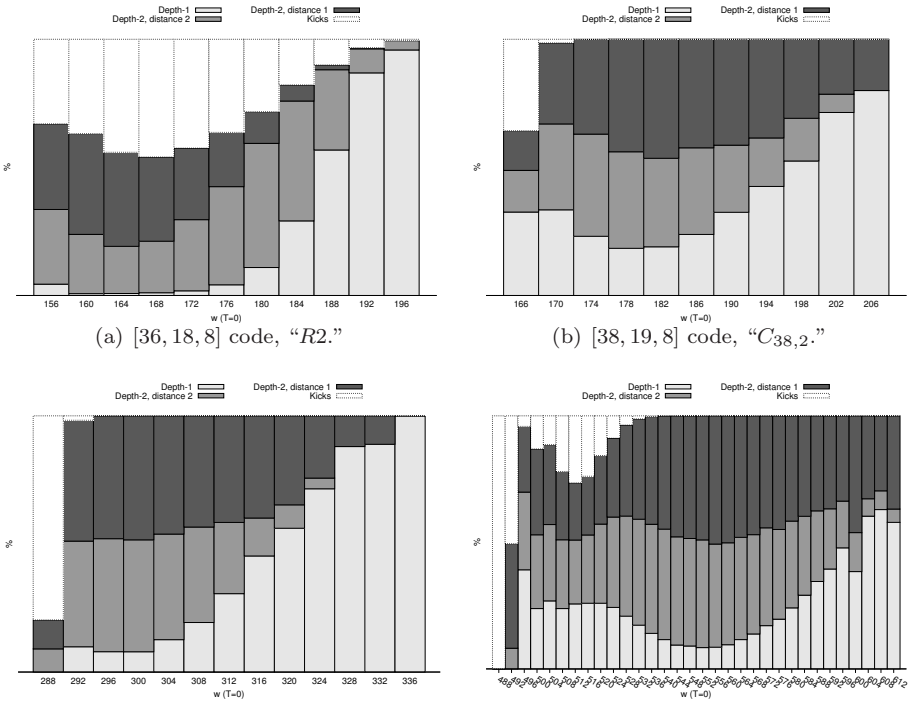
Figure 11: The percentage of depth-1 (Theorem 3) and depth-2 (Theorems 5 and 6) WB-ELC found by WB_ELC$_{(a)}$, for increasing weights, $|\mathcal{G}|$. Also shown (in red) is the (normalized) total count of WB-ELC sequences, and the plot in logarithmic scale (orange).

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$_{(\mathrm{b})}$ will find a WB-ELC operation, $\mathbf{e}$.

A simple algorithm is proposed, called Alg. minIP (listing omitted), which repeatedly invokes WB_ELC$_{(\mathrm{b})}(\mathcal{G}$, -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$_{(\mathrm{b})}$ 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$_{(\mathrm{b})}$ 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) 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 of weight $(n-k)d_{\min}(\mathcal{C}^{\perp})$, 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$_{(\mathrm{b})}$ 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$_{(\mathrm{b})}$ 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$_{(\mathrm{b})}$ 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 $|\mathcal{G}|$.



(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, "$C_{68,1}$."

Figure 12: The percentage of depth-1 (Theorem 3) and depth-2 (Theorems 5 and 6) returned (picked) by WB_ELC$_{\text{(b)}}$, 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 $L(\mathcal{G}, 0) = \emptyset$.

It should be emphasized, however, that WB_ELC$_{\text{(b)}}$, 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$_{\text{(b)}}$ 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,

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 $d_{\min}(\mathcal{C}^{\perp}) = 4$.

$\star$ Initial matrix described in [13, 16] (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, $|H| = (n-k)d_{\min}(\mathcal{C}^{\perp})$.

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.

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 operations can be used to give diversity during decoding. The framework is based on the SPA-PD algorithm [14], and centers around applying a damping routine [20] 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

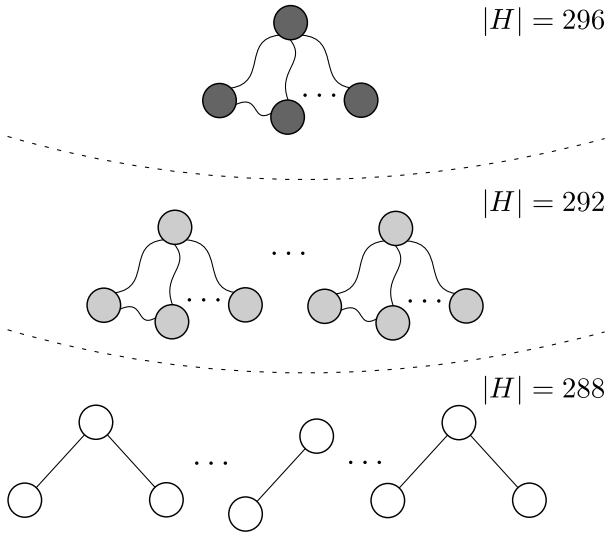


Figure 13: Sub-orbits of "EQR48" code, for which $|\mathcal{G}_0| = 288$. Nodes represent distinct graphs of the indicated weight, and edges represent a WB-ELC operation connecting two graphs. Curved edges mean arbitrary WB-ELC operations may exist. Between weight classes, this is indicated by dashed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

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) [20, 32]. 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 $\mathbf{L}$ [14]. After this, the relationship (15) between extrinsic information

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

$\alpha = \alpha_0$
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

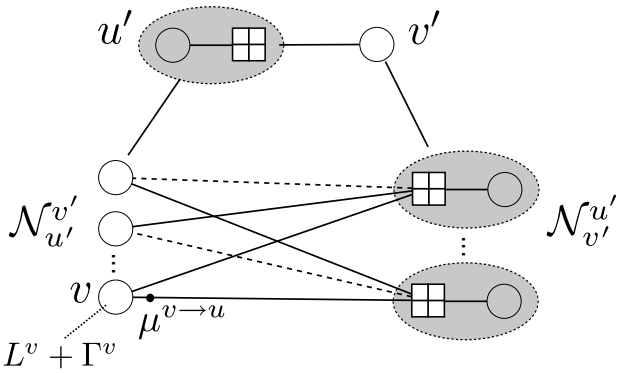


Figure 14: Description of LD, affecting all edges inserted by ELC on (u', v'). These new edges (solid lines), $(u, v)\ \forall\ v \in \mathcal{N}_{u'}^{v'}$, are initialized for the next iteration with $\mu_{j+1}^{v\to u} = L_j^v + \alpha(\Gamma_j^v - \mu_j^{v\leftarrow u}) = L_j^v + \alpha\Gamma_j^v$,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

(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 [25, 26]. 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

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

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 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) [25]. 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 [26], 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.

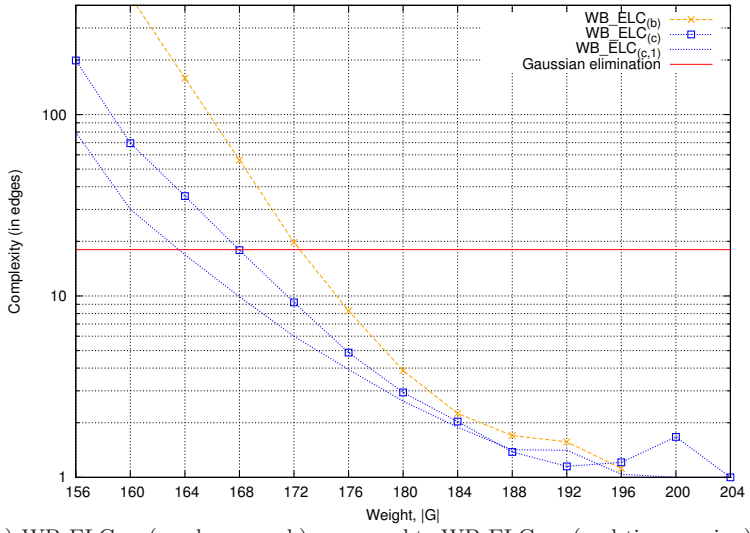
5.2 Real-Time WB-ELC Algorithm

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 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*

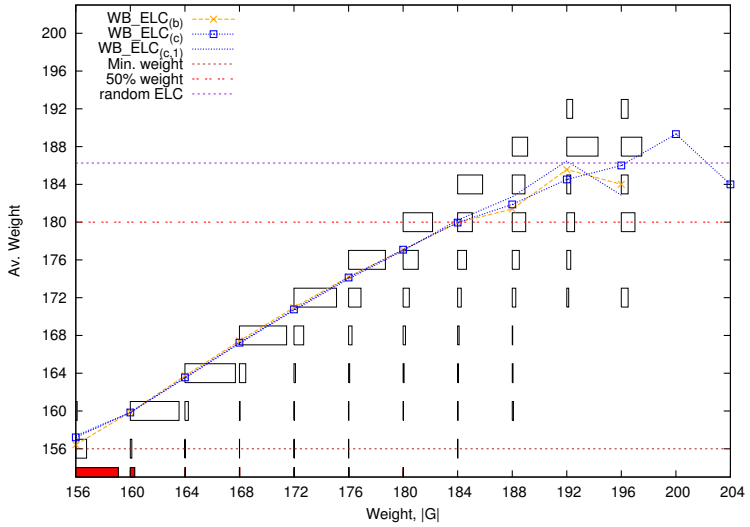
$$T' = |H_0| + T - |H| \tag{18}$$

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$_{(\mathrm{b})}$, we now consider a set of further modifications to give a *real-time* version, which we refer to as WB_ELC$_{(\mathrm{c})}$. 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$_{(\mathrm{c})}$ 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$_{(\mathrm{a})}$ (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$_{(\mathrm{c})}$ to WB_ELC$_{(\mathrm{b})}$. 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$_{(\mathrm{c})}$ which only considers depth-1 WB-ELC, denoted by WB_ELC$_{(\mathrm{c},1)}$. 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 [21, 26]. Implemented in terms

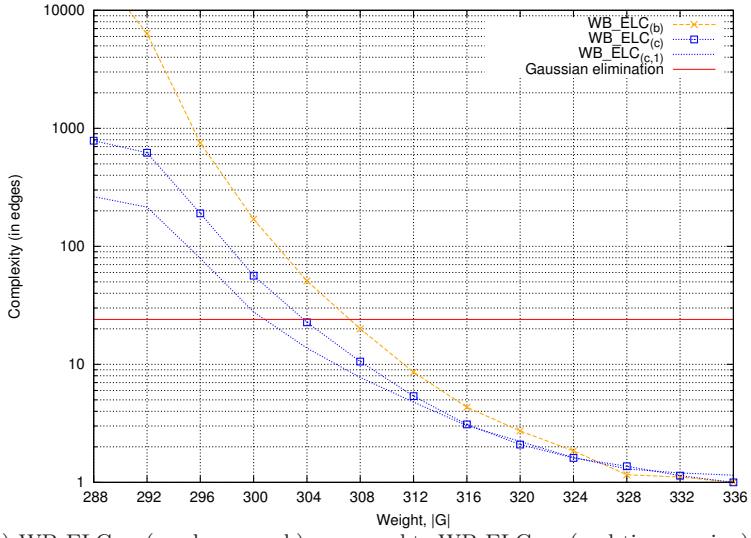


(a) WB_ELC$_{(b)}$ (random search) compared to WB_ELC $_{(c)}$ (real-time version) and a reduced real-time version searching only at depth-1.

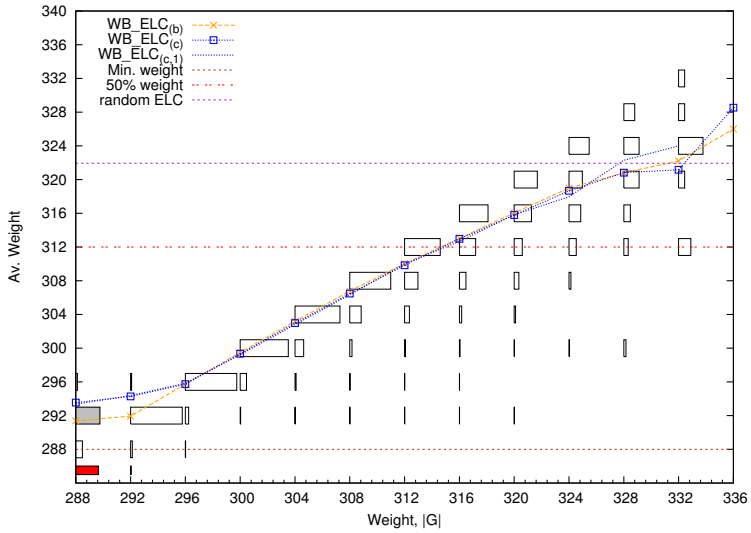


(b) For increasing weight, $|\mathcal{G}|$, the weight distribution of the graphs encountered during random traversal of the sub-orbit of the code approaches a normal distribution, with average centered at the expected weight found by random (unbounded) ELC.

Figure 15: Code "R2."

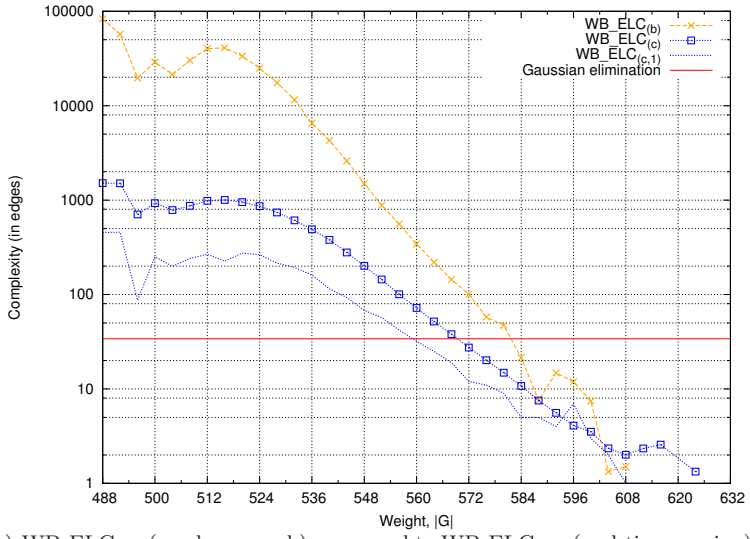


(a) WB_ELC$_{(b)}$ (random search) compared to WB_ELC $_{(c)}$ (real-time version) and a reduced real-time version searching only at depth-1.

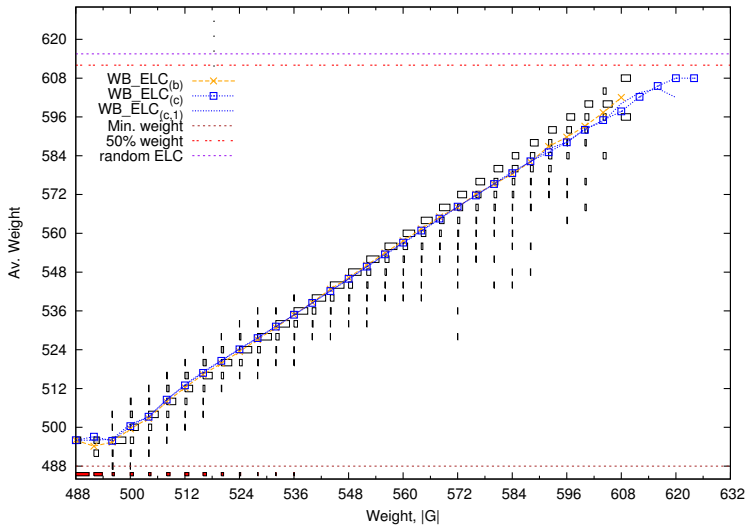


(b) For increasing weight, $|\mathcal{G}|$, the weight distribution of the graphs encountered during random traversal of the sub-orbit of the code approaches a normal distribution, with average centered at the expected weight found by random (unbounded) ELC.

Figure 16: Code "EQR48."

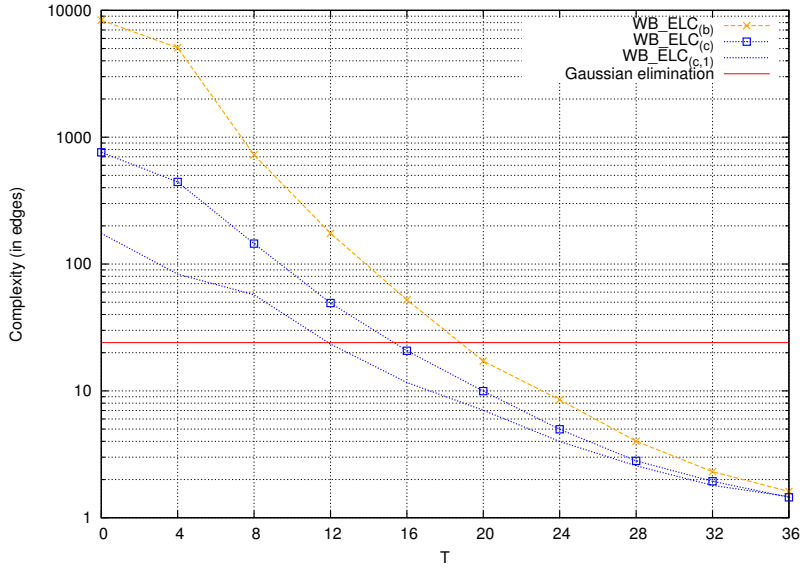


(a) WB_ELC$_{(b)}$ (random search) compared to WB_ELC $_{(c)}$ (real-time version) and a reduced real-time version searching only at depth-1.

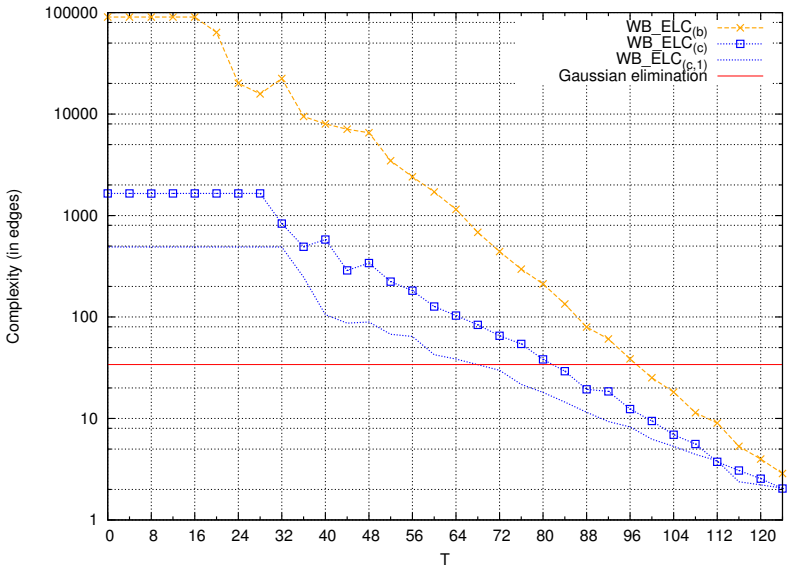


(b) For increasing weight, $|\mathcal{G}|$, the weight distribution of the graphs encountered during random traversal of the sub-orbit of the code approaches a normal distribution, with average centered at the expected weight found by random (unbounded) ELC.

Figure 17: Code "C_{68}."



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



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

Figure 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, $\mathcal{G}_0$.

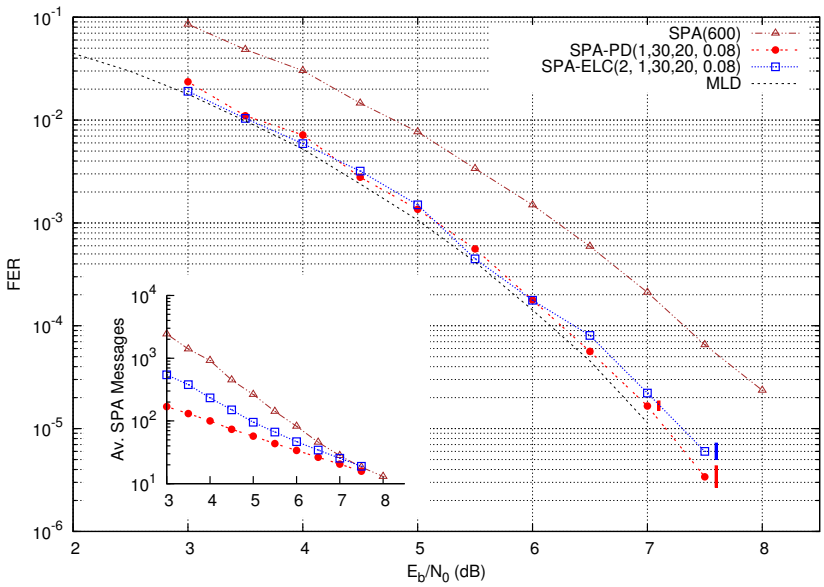
of ELC operations, the complexity of a GE stage is $n-k$, which is included as the red reference line.

Figs. 15(b), 16(b), and 17(b) illustrate the graph weights resulting from the previous simulations. For WB_ELC$_{\text{(b)}}$, a (horizontal) histogram is plotted for each weight class $|\mathcal{G}|$, showing the distribution of the weights 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 WB_ELC$_{\text{(b)}}(\mathcal{G}', 0) = \emptyset$. 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 WB_ELC$_{\text{(c)}}$ and WB_ELC$_{\text{(c,1)}}$ 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 WB_ELC$_{\text{(c,1)}}$ also succeeds in bounding the weight.

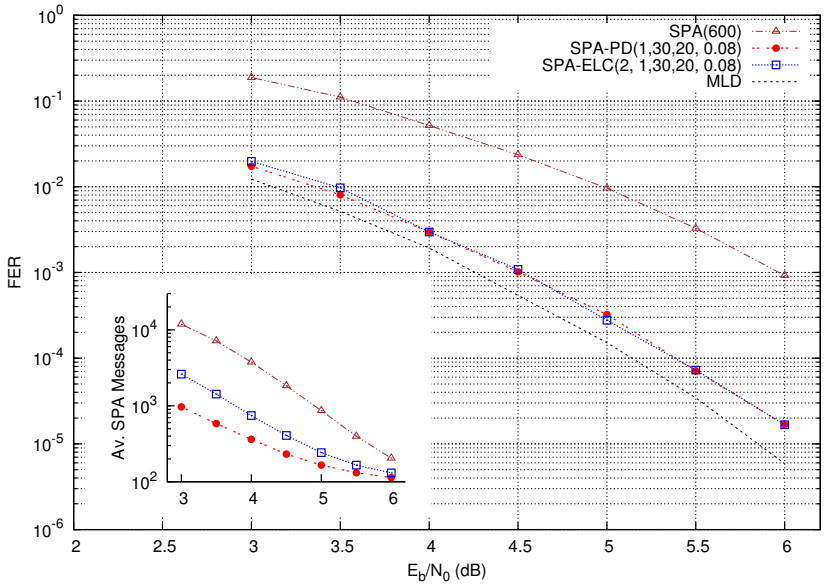
Fig. 18 shows a similar experiment which focuses on the performance of WB-ELC in a decoding setting. Starting from a reduced-weight graph, $\mathcal{G}_0$, we simulate the average performance of keeping $|\mathcal{G}|$ upper-bounded by $|\mathcal{G}_0| + T$. Reflecting the intended usage, $\mathbf{e} =$ WB_ELC$_{\text{(b)}}(\mathcal{G}, T')$ (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).

5.3 Error-Rate Observations

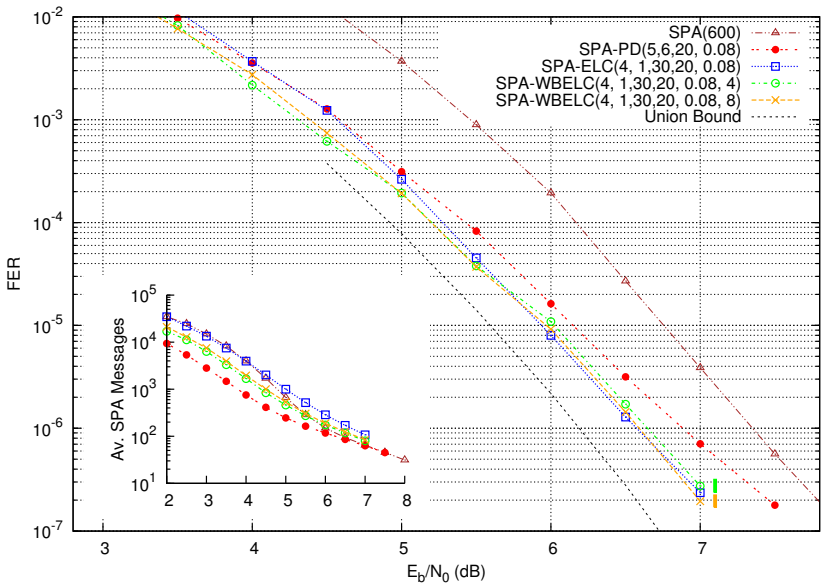
The SPA-ELC decoding algorithm, which uses random ELC operations in between SPA iterations to gain diversity, has previously been shown effective [25]. The simulation results in Fig. 19 compare the proposed SPA-ELC decoder against various decoding algorithms, where we ensure that $\tau = I_1 I_2 I_3$ 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



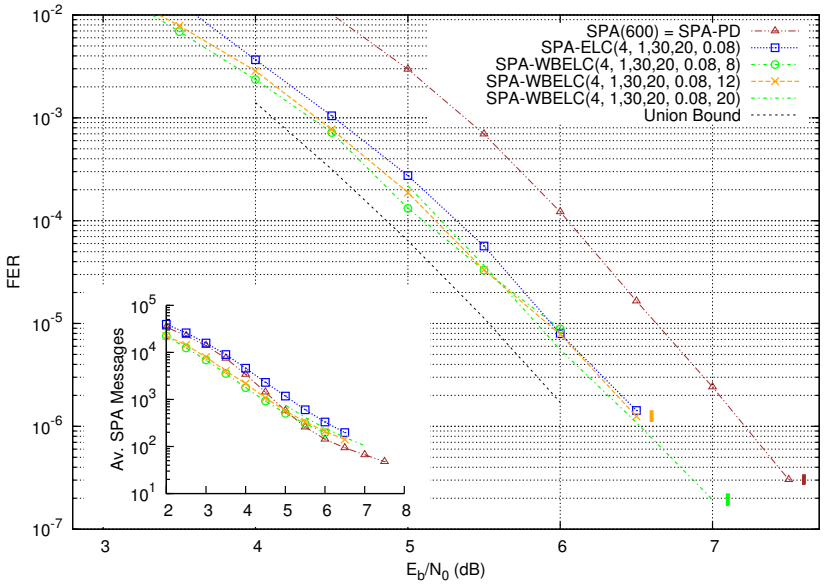
(a) BCH15 $= [15, 5, 7]$, with $|\mathrm{Aut}(\mathcal{C})| = 20\,160$



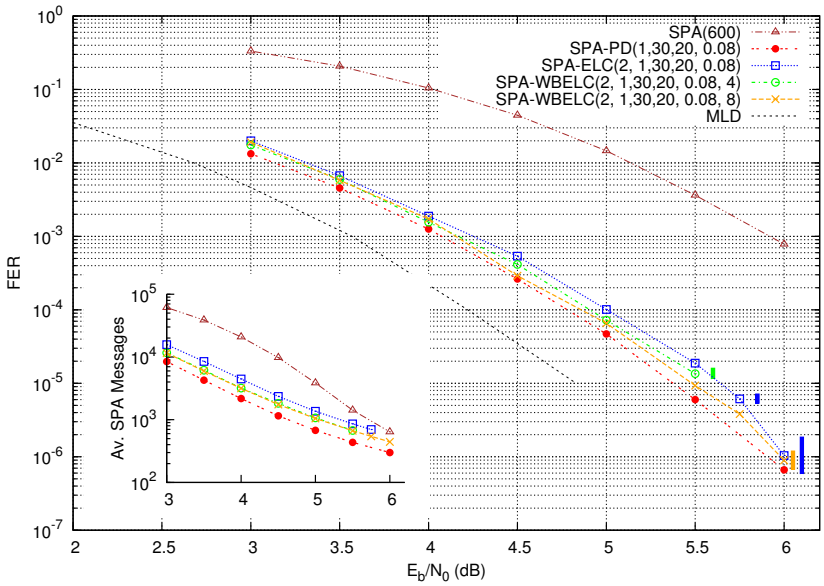
(b) Ext. Golay $= [24, 12, 8]$, with $|\mathrm{Aut}(\mathcal{C})| = 244\,823\,040$



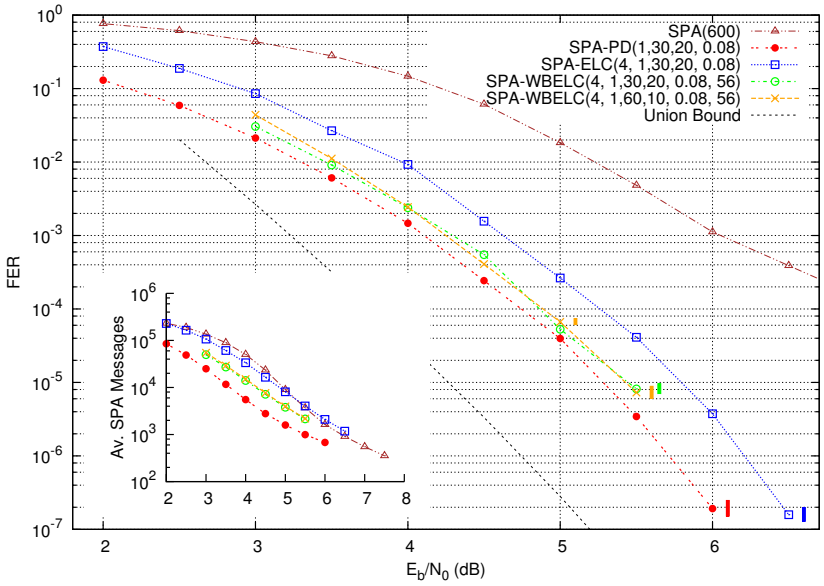
(c) $R2 = [36, 18, 8]$, with $|\mathrm{Aut}(\mathcal{C})| = 32$



(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 $|\mathrm{Aut}(\mathcal{C})| = 68$

Figure 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 [28]). The union bound is calculated based on the full weight enumerator of the code.

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$(p, I_1, I_2, I_3, \alpha_0, \text{WB_ELC}_{(\text{b})}(\mathcal{G}, T), \text{LD})$

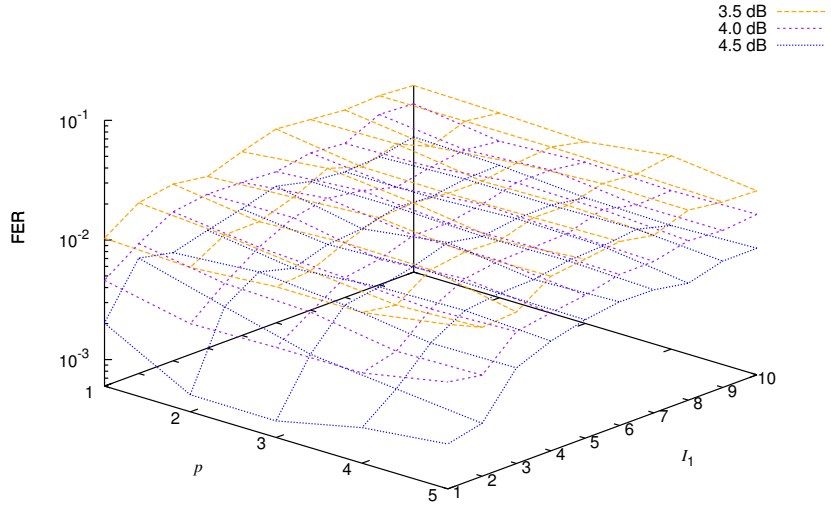
values of I_1 which we know to have the greatest influence on performance. The nearly identical data obtained for fixed I_2 and fixed I_3 verifies that the performance is dominated by I_1, and that the best performance is found for low values of I_1, specifically for $I_1 = 1$. This is verified also 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 [20].

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-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 [14] when $\text{Aut}(\mathcal{C})$ 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.

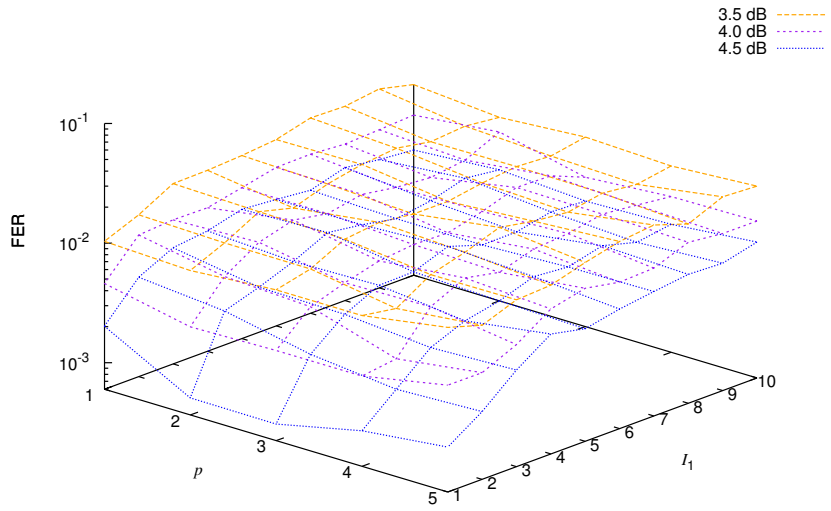
In Fig. 19, the performance of SPA-ELC and SPA-WBELC is simulated for various codes of different blocklength when signalling over the AWGN channel. The graphs used for decoding were optimized on weight in a preprocessing stage, as reported in Table 3. For SPA and SPA-PD, the same graph is used throughout 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 [14], the operation is permutation from $\text{Aut}(\mathcal{C})$, selected at random by taking random combinations of generators of $\text{Aut}(\mathcal{C})$ [6]. 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 [1]. We observe a performance gain of ~ 0.5dB at bit-error rate 10^{-4} over this scheme (we still observe a gain of ~ 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

[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 $I_3 = \tau/(I_1 I_2)$.

Figure 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 $p \approx 2$.

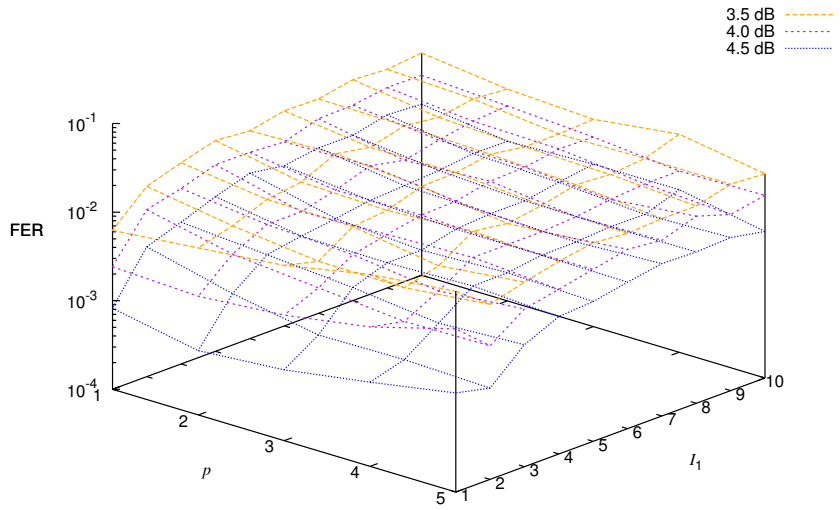
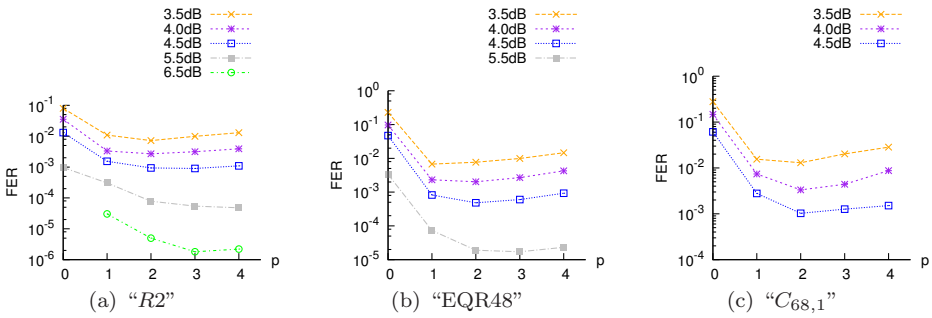


Figure 21: Similar response for SPA-ELC on code "EQR48" ($I_3 = 20$ fixed and $I_2 = \tau/(I_1 I_3)$).

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 ~ 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.

The overall observation is that SPA-ELC outperforms SPA for all codes, with a significant gain (over 1dB) which increases with blocklength. 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 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



(a) "$R2$" (b) "EQR48" (c) "$C_{68,1}$"

Figure 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.

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 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 $\mathrm{WB\_ELC}_{(\mathrm{b})}$ 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 $\mathrm{WB\_ELC}_{(\mathrm{c})}$ (or even $\mathrm{WB\_ELC}_{(\mathrm{c},1)}$), 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

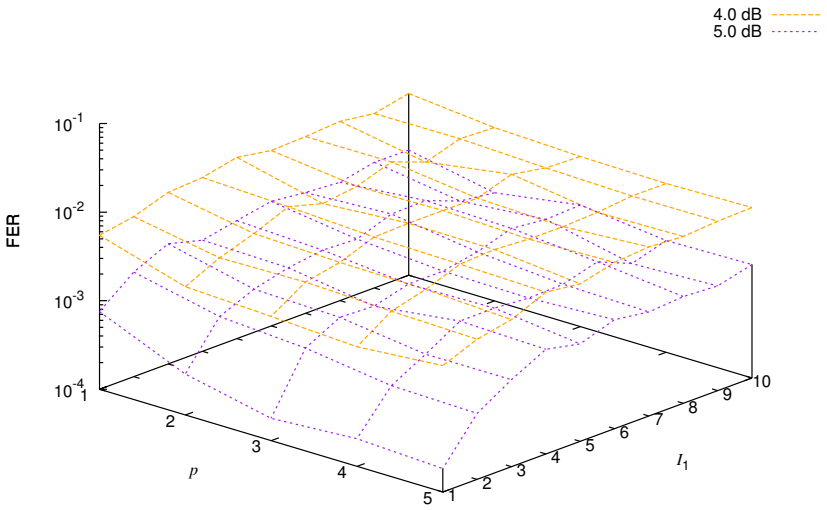


Figure 23: Simultaneous determination of parameters p and I_1 for SPA-WBELC decoding on code "$R2$." Here, $I_3 = 20$ and $I_2 = \tau/(I_1 I_3)$.

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$_{(\mathrm{b})}$, 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 [9, 11], 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

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

where double complementation cancels out, e.g., $A \not\nsim D = A \sim D$. 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] K. Andrews, S. Dolinar, and F. Pollara. LDPC decoding using multiple representations. In *Proc. IEEE Int. Symp. Inform. Theory*, page 456, Lausanne, Switzerland, June/July 2002.

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

[3] C. Berroux, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo codes. In *Proc. IEEE Int. Conf. Commun.*, pages 1064–1070, Geneva, Switzerland, May 1993.

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

[5] R. Brijder, T. Harju, and H. J. Hoogeboom. Pivots, determinants, and perfect matchings of graphs. arXiv:0811.3500, 2008.

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

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

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

[9] I. Dimnik and Y. Be'ery. Improved random redundant iterative HDPC decoding. *IEEE Trans. Commun.*, 57(7):1982–1985, July 2009.

[10] M. P. C. Fossorier. Reliability-based soft-decision decoding with iterative information set reduction. *IEEE Trans. Inform. Theory*, 48(12):3101–3106, December 2002.

[11] M. P. C. Fossorier, M. Mihaljevic, and H. Imai. Reduced complexity iterative decoding of low-density parity check codes based on belief propagation. *IEEE Trans. Commun.*, 47(5):673–680, May 1999.

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

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

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

[15] T. R. Halford and A. J. Grant. Which codes have 4-cycle-free Tanner graphs. *IEEE Trans. Inform. Theory*, 52(9):4219–4223, September 2006.

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

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

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

[19] W. C. Huffman. Handbook of coding theory. In V. S. Pless and W. C. Huffman, editors, *Handbook of Coding Theory*. Elsevier, North-Holland, Amsterdam, 1998.

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

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

[22] W. Jin and M. P. C. Fossorier. Reliability-based soft-decision decoding with multiple biases. *IEEE Trans. Inform. Theory*, 53(1):105–120, January 2007.

[23] J. G. Knudsen. Randomised construction and dynamic decoding of LDPC codes. Master's thesis, University of Bergen, Bergen, Norway, 2006.

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

[25] J. G. Knudsen, C. Riera, L. E. Danielsen, M. G. Parker, and E. Rosnes. Iterative decoding on multiple Tanner graphs using random edge local complementation. In *Proc. IEEE Int. Symp. Inform. Theory*, pages 899–903, Seoul, Korea, June/July 2009.

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

[27] A. Kothiyal, O. Y. Takeshita, W. Jin, and M. P. C. Fossorier. Iterative reliability-based decoding of linear block codes with adaptive belief propagation. *IEEE Commun. Lett.*, 9(12):1067–1069, December 2005.

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

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

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

[31] C. E. Shannon. A mathematical theory of communication. *Bell System Tech. J.*, 27:379–423, 623–656, July and October 1948.

[32] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Tech. rep.: CS-94-125, Carnegie Mellon University, Pittsburgh, PA, 1994.

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

[34] J. S. Yedidia, J. Chen, and M. P. C. Fossorier. Generating code representations suitable for belief propagation decoding. In *Proc. 40th Allerton Conf. Commun., Contr. and Comp.*, pages 447–456, Monticello, IL, October 2002.