Dynamic Message-Passing Decoding on Simple Graphs for the Quaternary Symmetric Channel

Matthew G. Parker et al *

August 31, 2014

1 Introduction

The aim of this note is to characterise a decoding scenario that is well-suited to message-passing decoding of $\mathbb{F}_4$-additive codes, where the natural graph structure on which to decode is a simple graph. To force this scenario we assume that we are transmitting our codewords over a *quaternary symmetric channel* - I don't think much, if any, LDPC/turbo work has been done for this channel? $\mathbb{F}_4$-additive codes whose duals are taken with respect to the *Hermitian inner product* are well-suited to this channel. This then allows us to exploit the optimal/maximal $\mathbb{F}_4$-additive codes, self-dual with respect to the Hermitian inner product, that have been previously classified by Lars and myself. Moreover, we have simple graph representations for such codes so this motivates a description of the message-passing process that uses simple graphs as opposed to the standard bipartite graphs - and we have ideas as to how to construct larger graphs which are both sparse and represent codes with high distance. The trick here is that Hermitian self-dual $\mathbb{F}_4$ codes can always, to within code equivalence, be written in a so-called *graph form*, i.e. their parity = generator matrices can always be expressed in the form $wI + A$, where A is a symmetric binary matrix, I is the identity, and w is the generator of $\mathbb{F}_4$. Therefore we call such codes *graph codes*. This means

*

that we can propose a 4-state message-passing decoder that only ever has to deal with parity matrices of the form $wI + A$, so it should be possible to construct more efficient decoders than are needed for more general 4-state decoding. It is important to stress that simple-graph decoding using the sum-product algorithm, as described in this note, is an identical algorithm to that used to do standard 4-state decoding using the sum-product algorithm, i.e. that uses bipartite graphs. The bipartite graphs are implicit in the simple-graphs. However, because we consider pairs of bits, the natural channel model for this scheme is the quaternary symmetric channel and its generalisations. Moreover our claims, in this context, are two-fold:

- The description of the simple graph (i.e. the adjacency matrix) is all that is needed to define the decoder for simple graph decoding - and for the dynamic-decoding version we also need to store a permutation description at each vertex, but this is a minor tweak. So the bipartite description is an overkill and, one can argue, is less natural.

- Simple graph decoding is a special subset of 4-state decoding so, as a result, one has a more efficient decoder for simple graph decoding, as it only has to deal with the special case.

We then propose dynamic decoding schemes for message-passing decoding of these graph codes on a quaternary symmetric channel. Local complementation is a graph update rule for simple graphs that, if applied to the graph codes in graph form, traverses an orbit of equivalent graph codes - note the codes are not, in general, identical, just equivalent, which is a problem for the decoding process. We propose a matrix-form of local complementation that can be applied to the parity matrices associated with the graph codes, and involves just row additions. This preserves the code, but the graph form of the parity check matrix is lost. To recover the graph form we also need to propose permutations of the representations inherent to each 4-state symbol after each local complementation step. This recovers the graph form but now we only have an equivalent code. So, to ensure that we are decoding on the correct code, we have also to update the soft information of each symbol according to the same permutations. In summary, we are able to dynamically apply local complementation to the Hermitian self-dual $\mathbb{F}_4$-additive code (graph code) in graph form to preserve both code and graph form.

A second type of dynamic decoding can be proposed when we only have a 2-state message-passing decoder to decode our 4-state symbols. Such a

scenario requires that we initially approximate each of our 4-state received symbols by two 2-state symbols, before passing on to the 2-state decoder. But that approximation can be done in a number of ways, and it is this diversity of approximation which allows us to propose *dynamic symbol permutation decoding* for such scenarios.

Before we flesh out these scenarios in more detail we first describe these graph codes using a purely binary notation. The point here is that it is not necessary to describe the code structures by using the language of $\mathbb{F}_4$. It is done later just for notational convenience. So, to begin with, we define the *graph-class* of binary codes that can be defined by a binary $n \times 2n$ parity matrix, where the dual is defined with respect to the *symplectic inner product* of pairs of bits. We characterise the graph-class and show that a subset of it, called the *self-dual graph-class* (SD graph-class), is a class of half-rate binary codes that are self-dual with respect to the symplectic inner product. We then show that all these definitions map naturally over to the $\mathbb{F}_4$-world so that a symplectic self-dual binary code of length $2n$ from the SD graph-class is equivalent to a Hermitian self-dual $\mathbb{F}_4$-additive code (graph code) of length n. After establishing this bridge we then develop the decoding scenarios primarily using the language of $\mathbb{F}_4$.

2 Matrix structure $\Leftrightarrow$ Graph Structure

The matrix structure we deal with, primarily, in this note comprises $n \times 2n$ binary parity matrices, $\{H\}$, with a particular constraint:

$$H = K|A, \quad K \circ A \text{ symmetric }, \quad K, A \ n \times n \text{ binary matrices},$$

where '$\circ$' means element-wise binary 'OR'. A special case occurs when $K = I$, the identity, in which case H is in systematic form and A is symmetric. We refer to this class of matrices (rather unimaginatively) as the *graph-class* and, more precisely, as the n-vertex graph-class. We primary consider rows of H as existing $\in (\mathbb{F}_2^2)^n$ as each row comprises n binary pairs, where bits at indices $(0, n)$, $(1, n+1)$, $\ldots$, $(n-1, 2n-1)$ comprise the binary pairs.

For H in the n-vertex graph-class, let $\Gamma = (K \circ A) \oplus I$, where '$\oplus$' means mod 2 addition. Then, up to a diagonal, Γ can be viewed as the adjacency matrix of a simple graph of n vertices. So the n-vertex graph-class of matrices

has a mapping to the n-vertex simple graphs, but this mapping is not one-to-one as more than one graph-class matrix maps to the same simple graph.

3 Local complementation on the graph-class

Local complementation is a graph operation that acts on simple graphs. Let v be a vertex in a simple graph, $\mathcal{G}$, and let $\mathcal{N}_v$ be the set of neighbouring vertices of v in $\mathcal{G}$. Let $\mathcal{G}_v$ be the subgraph of $\mathcal{G}$ induced by the vertices in $\mathcal{N}_v$. Let $\mathcal{G}'$ be the remaining edges in $\mathcal{G}$, such that $\mathcal{G} = \mathcal{G}' + \mathcal{G}_v$ is the edge-disjoint union of two graphs. Then local complementation on $\mathcal{G}$ at vertex v acts by complementing the edges in $\mathcal{G}_v$ and results in the graph $\mathcal{G}^v$, where $\mathcal{G}^v = \mathcal{G}' + \bar{\mathcal{G}}_v$, where $\bar{\mathcal{G}}_v$ is the graph complement of $\mathcal{G}_v$.

We now define the action of local complementation on the graph-class of matrices.

Let H be a member of the n-vertex graph-class, and let Γ be its associated adjacency matrix, which represents the simple graph, $\mathcal{G}$. Then local complementation at vertex v (row v) of H gives H^v, and is realised as follows.

$H^v \leftarrow H$:

- row $i \leftarrow$ row $i \oplus$ row v, iff (i, v) is an edge in $\mathcal{G}$.
- row i unchanged otherwise.

One can verify that

$$H^v \leftarrow H \quad \equiv \quad \Gamma^v \leftarrow \Gamma \quad \equiv \quad \mathcal{G}^v \leftarrow \mathcal{G},$$

where Γ^v and $\mathcal{G}^v$ are the adjacency matrix and graph associated with the graph-class matrix, H^v.

The action of local complementation on matrices from the graph-class keeps them in the graph-class and, moreover, preserves the binary linear code, as it involves only row operations.

4 Example

Consider the following half-rate $[10, 5]$ binary linear code, with parity matrix

$$H = \begin{array}{ccc} 10000 & | & a1001 \\ 01000 & | & 1b100 \\ 00100 & | & 01c10 \\ 00010 & | & 001d1 \\ 00001 & | & 1001e \end{array} = I|A,$$

where the binary symbols comprising the binary vector $D = (abcde) \in \mathbb{F}_2^5$ are non-determined at present. Then H is in the graph-class, where $K = I$, and H maps to a simple graph of n vertices with adjacency matrix,

$$\Gamma = \begin{array}{c} 01001 \\ 10100 \\ 01010 \\ 00101 \\ 10010, \end{array}$$

which is the adjacency matrix for C_5.

One can verify that

$$H^0 = \begin{array}{ccccccc} 10000 & | & a & 1 & 0 & 0 & 1 \\ 11000 & | & 1 \oplus a & 1 \oplus b & 1 & 0 & 1 \\ 00100 & | & 0 & 1 & c & 1 & 0 \\ 00010 & | & 0 & 0 & 1 & d & 1 \\ 10001 & | & 1 \oplus a & 1 & 0 & 1 & 1 \oplus e \end{array} = K^0|A^0,$$

where

$$\Gamma^0 = (K^0 \circ A^0) \oplus I = \begin{array}{c} 01001 \\ 10101 \\ 01010 \\ 00101 \\ 11010, \end{array}$$

which is the adjacency matrix for $\mathcal{G}^0$.

One can further verify that

$$(H^0)^1 = H^{01} = \begin{array}{ccccccc} 01000 & | & 1 & b & 1 & 0 & 0 \\ 11000 & | & 1\oplus a & 1\oplus b & 1 & 0 & 1 \\ 11100 & | & 1\oplus a & b & 1\oplus c & 1 & 1 \\ 00010 & | & 0 & 0 & 1 & d & 1 \\ 01001 & | & 0 & b & 1 & 1 & e \end{array} = (K^0)^1|(A^0)^1,$$

where

$$(\Gamma^0)^1 == \Gamma^{01} = ((K^0)^1 \circ (A^0)^1) \oplus I = \begin{array}{c} 01100 \\ 10101 \\ 11011 \\ 00101 \\ 01110, \end{array}$$

which is the adjacency matrix for $(\mathcal{G}^0)^1 = \mathcal{G}^{01}$.

In the example, the graph-class was preserved by local complementation, irrespective of the choice of $D = (abcde)$ and, moreover, all matrices are parity matrices for the same binary linear code.

5 Some pathological examples leading to two extra constraints on the graph-class

THIS SECTION NEEDS TO BE MADE A LITTLE MORE PRECISE

Consider the following parity matrix

$$H = \begin{array}{ccc} 00000 & | & a1001 \\ 00000 & | & 1b100 \\ 00000 & | & 01c10 \\ 00000 & | & 001d1 \\ 00000 & | & 1001e \end{array} = 0|A,$$

where the binary symbols comprising the binary vector $D = (abcde) \in \mathbb{F}_2^5$ are non-determined at present. Then H is in the graph-class, where $K = 0$,

and H maps to a simple graph of n vertices with adjacency matrix,

$$\Gamma = \begin{array}{ccccc} 1\oplus a & 1 & 0 & 0 & 1 \\ 1 & 1\oplus b & 1 & 0 & 0 \\ 0 & 1 & 1\oplus c & 1 & 0 \\ 0 & 0 & 1 & 1\oplus d & 1 \\ 1 & 0 & 0 & 1 & 1\oplus e \end{array}$$

which, up to a diagonal, is the adjacency matrix for C_5.

But we see that this is not a particularly interesting example as the first 5 columns of H are zero. Moreover, for some choices of $D = (abcde)$, A may not even have maximum rank as a binary matrix. Furthermore, even if A has maximum rank, we may still obtain graphs which are not fully-connected. For instance, $H = K|A$, where $K = 0$, and $A = I$, represents an unconnected graph, and this is not interesting.

To avoid these degenerate and/or unconnected members of the graph-class we require two extra constraints on the graph-class:

- At least one of A or K must have maximum rank as a binary matrix.
- The graph, $\mathcal{G}$, associated with H, must be connected.

We refer to this sub-class of the graph-class as the *connected non-degenerate graph-class* but, in order to enhance the admittedly miniscule possibility of retaining some sort of social life we, for the rest of this note, refer to this subclass simply as the 'graph-class'.

6 Channel models suited to the graph-class

The graph-class interpretation of the matrix, H, implicitly pairs off matrix columns $(0, n)$, $(1, n+1)$, $\ldots$, $(n-1, 2n-1)$, mapping them to vertices $0, 1, \ldots, n-1$, respectively, of the associated simple graph, $\mathcal{G}$. Thus it is natural to associate the graph-class parity matrix, H, with the transmission and reception of a series of length-n codewords, where each codeword comprises a series of n 4-state symbols, where each 4-state symbol is associated with one of the n vertices of $\mathcal{G}$.

We consider two possible transmit/receive scenarios:

- The communications system independently transmits and receives each 4-state symbol using some 4-state modulation scheme - we refer to this as 4-state modulation.

- The communications system independently transmits and receives each 4-state symbol using some 2-state modulation scheme by first describing each 4-state symbol using two bits, then sending each bit independently - we refer to this as 2-state modulation.

For each of these modulation schemes we can consider certain basic error models (there are, of course, more):

- 4-state modulation - alphabet $\mathcal{A} = \{u, v, w, x\}$:

  - The probability of transmitting symbol $i \in \mathcal{A}$ and receiving symbol $j \in \mathcal{A}$, $j \neq i$, is a constant, independent of both i and j. This means that the probability of transmitting symbol $i \in \mathcal{A}$ and receiving symbol i is also a constant, independent of i. We refer to this error model as a 'quaternary-symmetric channel'.

  - The probability of transmitting symbol $i \in \mathcal{A}$ and receiving symbol i is a constant, independent of i. But the probability of transmitting symbol $i \in \mathcal{A}$ and receiving symbol $j \in \mathcal{A}$, $j \neq i$, depends on both i and j. For instance, standard 4-phase-shift-keying could be an example of this where, for instance, the probability of sending symbol u as phase '1' and receiving symbol x as phase '-1' is, typically, less than the probability of sending symbol $u = 1$ as phase '1' and receiving either symbol v as phase '$\frac{1}{\sqrt{2}}(1 + i)$' or symbol w as phase '$\frac{1}{\sqrt{2}}(1 - i)$', where $i = \sqrt{-1}$.

- 2-state modulation: - alphabet $\mathcal{A} = \{u, v\}$:

  - The probability of transmitting symbol $i \in \mathcal{A}$ and receiving symbol $j \in \mathcal{A}$, $j \neq i$, is a constant, independent of both i and j. This means that the probability of transmitting symbol $i \in \mathcal{A}$ and receiving symbol i is also a constant, independent of i We refer to this error model as a 'binary-symmetric channel'.

The channel model that we are, here, particularly interested in decoding over is the quaternary symmetric channel. A physical system for which a

quaternary symmetric channel is appropriate could be a three-dimensional channel where the 4 modulation points occur as vertices of a tetrahedron. We can later consider the AWGN (i.e. assuming Gaussian noise) variant for the quaternary symmetric channel.

7 The self-dual graph-class and self-dual $\mathbb{F}_4$-additive codes

Let H be a $2n \times n$ binary matrix. Then, for each row of H, each binary pair at column indices $(i, i+n)$ can be represented by a 4-state symbol. In particular it is convenient to represent the binary pair by a symbol from $\mathbb{F}_4$. Define the map, M:

$$M: \quad 0 \leftarrow 00, \quad 1 \leftarrow 01, \quad w \leftarrow 10, \quad w^2 \leftarrow 11,$$

where $w^3 = w^2 + w = 1$. (In fact any invertible map between $\{0, 1, w, w^2\}$ and $\{00, 01, 10, 11\}$ with the restriction $0 \leftarrow 00$ will do).

Using this map we obtain $H_4 = M(H)$. For example, for the $[10, 5]$ example discussed previously, and with $D = (abcde) = (00000)$, we get,

$$H_4 = M(H) = \begin{matrix} w1001 \\ 1w100 \\ 01w10 \\ 001w1 \\ 1001w. \end{matrix}$$

Lemma 1 *Binary addition of rows of H maps to $\mathbb{F}_4$ addition of rows of H_4.*

Proof. Self-evident. □

Likewise, binary multiplication of rows of H is mapped to binary multiplication of rows of H_4. Usually, one would consider the $\mathbb{F}_4$-linear code constrained by $\{0, 1, w, w^2\}$ multiples of rows of H. But if, instead, we consider only the $\mathbb{F}_4$-additive code constrained by $\{0, 1\}$ multiples of rows of H, then it is evident that the graph-class matrix can be viewed as a parity-check matrix for an $\mathbb{F}_4$-additive code.

This mapping from binary-linear to $\mathbb{F}_4$-additive is not strictly required - it is just another view of the same thing. But we give it here because we already have a lot of results that have been proved in the context of $\mathbb{F}_4$-additive codes. Moreover it is, perhaps, more natural if we are dealing with modulation schemes that are communicating 4-state symbols.

Define the *Hermitian inner product* of $\mathbb{F}_4$ vectors u and v to be given by $u \star_4 v = \sum_i u_i v_i^2 \oplus u_i^2 v_i$. Then H_4 is defined to be the parity-check matrix of a *self-dual* $\mathbb{F}_4$-additive code with respect to the Hermitian inner product iff all its rows are pairwise orthogonal with respect to the Hermitian inner product, i.e. $u \star_4 v = 0$ $\forall$ distinct rows u and v of H_4.

Define the *symplectic inner product* of length-$2n$ binary vectors, $u = u_l|u_r$ and $v = v_l|v_r$, where each of u_l, u_r, v_l, v_r are of length n, to be given by $u \star v = \sum_i u_{l,i} v_{r,i} \oplus u_{r,i} v_{l,i}$. Then H is defined to be the parity-check matrix of a *self-dual* binary code with respect to the symplectic inner product iff all its rows are pairwise orthogonal with respect to the symplectic inner product, i.e. $u \star v = 0$ $\forall$ distinct rows u and v of H.

Lemma 2 *If H is self-dual with respect to the symplectic inner product, then $H_4 = M(H)$ is self-dual with respect to the Hermitian inner product, and vice versa.*

Proof. The mapping from binary to $\mathbb{F}_4$ may be described by the mappings $u \leftarrow (u_l, u_r)$, $v \leftarrow (v_l, v_r)$, where $0 \leftarrow (0,0)$. Observe that $u^2 v + u v^2 = uv(u+v) \in \mathbb{F}_2$. If $u = 0$ or $v = 0$ then $uv(u+v) = 0 \leftrightarrow u_l v_r + u_r v_l = 0$ is self-evident. So it remains to prove for $u \neq 0$ and $v \neq 0$. For this case $uv(u+v) = 1$ iff $(u_l \neq v_l)$ or $(u_r \neq v_r)$, i.e. iff $(u_l \neq v_l) \circ (u_r \neq v_r)$. But $(u_l \neq v_l) = (u_l + v_l)$ and $a \circ b = ab + a + b$, so $(u_l \neq v_l) \circ (u_r \neq v_r) = (u_l + v_l)(u_r + v_r) + u_l + v_l + u_r + v_r$. Therefore $uv(u+v) = 1$ iff $u_l v_r + u_r v_l + (u_l + 1)(u_r + 1) + (v_l + 1)(v_r + 1) = 1$. But we are only considering the case $u \neq 0$ and $v \neq 0$, so the above reduces to $uv(u+v) = 1$ iff $u_l v_r + u_r v_l = 1$. The converse follows simply. $\square$

Consider the set, $\mathcal{P}_4$, of 6 possible permutations of the non-zero elements, $\{1, w, w^2\}$, of $\mathbb{F}_4$.

Lemma 3 *The matrix obtained by applying members of $\mathcal{P}_4$ to the elements of each column of H_4 (one member per column) is Hermitian self-dual iff H_4 is Hermitian self-dual.*

Proof. Proved elsewhere. □

Consider the set $\mathcal{P}$, of 6 possible permutations of the non-zero bit-pairs, $\{01, 10, 11\}$, of $\mathbb{F}_2^2$.

Corollary 1 *The matrix obtained by applying members of $\mathcal{P}$ to the elements of each column-pair of H (one member per column-pair) is symplectic self-dual iff H is symplectic self-dual.*

Let $\sigma \in \mathcal{P}_4^n = (\sigma_0, \sigma_1, \ldots, \sigma_{n-1})$ be some n-fold permutation of the elements in columns 0 to $n-1$ of H_4, i.e. σ_j is the permutation applied to column j of H_4.

Lemma 4 *If $H_4 = wI \oplus A$ then we say that H_4 is in* graph form. *Such a matrix always generates a Hermitian self-dual $\mathbb{F}_4$-additive code. Conversely, all Hermitian self-dual $\mathbb{F}_4$-additive codes can, to within some n-fold permutation σ, be described by parity matrices in graph form. Therefore we refer to such codes as* graph codes.

Proof. One can easily convince oneself that any two rows of a matrix of the form $H_4 = wI \oplus A$ are always orthogonal with respect to the Hermitian inner-product. The converse is less trivial but has been proved, for instance, in theorem 6 of [2]. □

The fact that all $\mathbb{F}_4$-additive codes that are self-dual with respect to the Hermitian inner product can, up to permutation equivalence, be written in the graph form $H_4 = wI \oplus A$ is particularly powerful as it allows us to classify such codes using graph theory [2].

Lemma 5 *If H is in the graph-class then it is not necessarily self-dual with respect to the symplectic inner product.*

Proof. Consider $H = K|A = \begin{array}{c|c} 100 & 011 \\ 010 & 100 \\ 101 & 100 \end{array}$. Then $K \circ A$ is symmetric but the last two rows are not orthogonal with respect to the symplectic inner product. □

Lemma 6 *If H is from the graph-class and is self-dual with respect to the symplectic inner product, then we say it is in the* self-dual graph-class *(SD graph-class) and it defines a code, C, which can be generated by $G = H$. Likewise, if H_4 is in graph-form then it has a corresponding generator matrix $G_4 = H_4$.*

Proof. The arguments follows immediately from the definition of self-duality (so this is probably not really a separate lemma). □

The *symplectic weight*, $\mathrm{wt}(c)$, of a codeword, c, from an even-length binary linear code, C, of blocklength $2n$ and with parity-check matrix $K|A$, is defined as follows.

$$\mathrm{wt}(c) = \sum_{i=0}^{n-1} c_i \circ c_{i+n},$$

where the sum is over the integers. For instance, for the C_5 example code discussed previously, $G = H$ and C_5 contains the codeword 10000 : 01001 comprising the five bit-pairs $(1,0), (0,1), (0,0), (0,0), (0,1)$ and is of symplectic weight 3. The *symplectic Hamming distance*, d, of an even-length binary linear code, C, with parity-check matrix $K|A$, then follows immediately from the definition of symplectic weight, and is the non-zero codeword from C with the minimum number, d, of non-zero bit-pairs. For instance, for the C_5 example code discussed previously, $d = 3$, and a minimum symplectic weight codeword from such a code is: 10000 : 01001. For the corresponding $\mathbb{F}_4$-additive code, the Hermitian weight is just the number of non-zero elements in a codeword. For instance, for our example code C_5 we have $G_4 = H_4$ and our example codeword maps to $w1001$ which has weight 3. Moreover the Hermitian distance of C_5 is 3. Note that our examples are for self-dual codes, but the above statements hold for any even-length binary symplectic codes (resp. $\mathbb{F}_4$-additive codes) - they don't even have to be half-rate. For example, let $G = \left.\begin{matrix} 111 \\ 010 \end{matrix}\right| \begin{matrix} 110 \\ 001 \end{matrix}$. Then $G_4 = \begin{matrix} w^2 & w^2 & w \\ 0 & w & 1 \end{matrix}$. Then G generates the four codewords $(0,0)(0,0)(0,0)$ and $(1,1)(1,1)(1,0)$ and $(0,0)(1,0)(0,1)$ and $(1,1)(0,1)(1,1)$ of symplectic weights $0, 3, 2, 3$, respectively. Likewise G_4 generates the four codewords 000, w^2w^2w, $0w1$, and w^21w^2, respectively, again of (Hermitian) weights $0, 3, 2, 3$.

For the quaternary symmetric channel it is natural to use the Hermitian (or symplectic) weight to quantify code distance, as the three single symbol

errors on symbol 0 take it to 1, w, and w^2, respectively, with equal probability, and therefore these errors should be quantified by equal weight, which is what Hermitian (or symplectic) weight does (in contrast, for instance, QPSK channels are more accurately modeled by *Lee distance*).

Let $c \in C$. Then $H \star c = 0$. Let S be the matrix that swaps the first n columns of H with the last n columns, such that $E = HS$. (Then S can be written as $S = I_n \otimes X$, where I_n is the $n \times n$ identity matrix, and $X = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix} \right)$). It follows from the definition of the symplectic product that, for $c \in C$,

$$H \star c = HSc = Ec = 0.$$

But, for H from the SD graph-class, $G = H$ is a generator matrix for the code. Therefore $H \star H^t = H(HS)^t = HE^t = 0$. So another way of defining the SD graph-class is as the class of $n \times 2n$ binary matrices that generate codes that are self-dual with respect to the standard inner-product to within a swap of the first n columns and last n columns of the matrix, i.e. they are a special class of half-rate binary *isodual codes*.

Lemma 7 *The SD graph-class is contained in the class of half-rate binary isodual codes of even length with respect to column permutations of order two. (ARE THE TWO CLASSES THE SAME??)*

We shall not persevere with this alternative isodual definition explicitly, but it is worth bearing in mind.

8 The sum-product algorithm for simple graph decoding on a quaternary symmetric channel

As discussed, the $n \times 2n$ graph-class can be associated with the class of n-vertex simple graphs. Instead of working with the graph-class directly, we use, in this section, the $\mathbb{F}_4$-additive representation - the only reason for this is that it is notationally more concise. So we use the mapping $M : 00 \rightarrow 0, 01 \rightarrow 1, 10 \rightarrow w, 11 \rightarrow w^2$. Moreover we focus on the SD graph-class, which translates to self-dual $\mathbb{F}_4$-additive codes in graph form.

In this section we consider 4-state modulation over the quaternary symmetric channel, and propose to use 4-state decoding for graph codes. The message-passing used is the standard sum-product algorithm adapted to the 4-state case. But we argue that the message-passing structure, usually modelled by some bipartite graph, is here more naturally modelled by a simple graph. Moreover, when we generalise to dynamic decoding, then we can argue that, up to an initial internal permutation mapping of each of the n received 4-state soft symbol vectors, and a final internal inverse permutation of the decoded symbols, the message-passing structure is still naturally modelled by a simple graph.

Consider, once more, the previous symplectic $[10, 5, 3]$ code example, but this time expressed using a Hermitian $(5, 2^5, 3)$ $\mathbb{F}_4$-additive code. Then,

$$H_4 = wI + A = \begin{matrix} w1001 \\ 1w100 \\ 01w10 \\ 001w1 \\ 1001w, \end{matrix}$$

where, without loss of generality, one can assume the diagonal $D = (abcde) = (00000)$. Then, because H_4 is in graph form, the graph, $\mathcal{G}$, associated with H_4 has adjacency matrix $\Gamma = A$ and, in our example Γ is the adjacency matrix for C_5. Conversely, if we have the promise that our matrix H_4 is in graph form, then the description provided by Γ is sufficient for decoding of the code using, say, message-passing - each vertex describes both variable and function node, where the local function is precisely defined by the graph. For instance, vertex 1 of $\mathcal{G}$ in our example describes the Hermitian constraint $1w100$. Of course H_4 could be described using an equally valid more conventional bipartite graph, where function and variable nodes are separated, but when H_4 is guaranteed to be in graph form, then this bipartite description is somewhat redundant and, arguably, less natural. So, if we restrict our class of codes to graph codes, and further restrict to message-passing only on parity matrices, H_4, in graph form (which are guaranteed to exist for each such code), then the n-vertex simple graph representation is sufficient and concise. Moreover, as we only have to deal with one type of 4-state constraint, the implementation of each vertex can be made more efficient, both in hardware and software, than a general 4-state constraint. We call this *simple graph decoding*.

We now have sufficient detail to realise both simple graph decoding, and

simple graph dynamic decoding. This is for the scenario where we have 4-state modulation over a quaternary-symmetric channel, with 4-state decoding.

Consider the trivial parity matrix $H_4 = \left(\begin{smallmatrix} w & 1 \\ 1 & w. \end{smallmatrix} \right)$.

Then the first row, $(w, 1)$, describes the constraint f_0. Number the columns as 0 and 1, so column 0 contains the w. Remembering that we are constraining with respect to the Hermitian inner product, then $(w, 1)$ constrains to the 8 codewords $\{(0,0), (w,0), (1,w), (w^2,w), (0,1), (w,1), (1,w^2), (w^2,w^2)\}$. We assume the input of 4-state soft symbol vectors into the decoder. Likewise $(1, w)$ describes the constraint f_1 and constrains to the 8 codewords $\{(0,0), (0,w), (w,1), (w,w^2), (1,0), (1,w), (w^2,1), (w^2,w^2)\}$. We can visualise this situation by the factor graph in Fig 1:

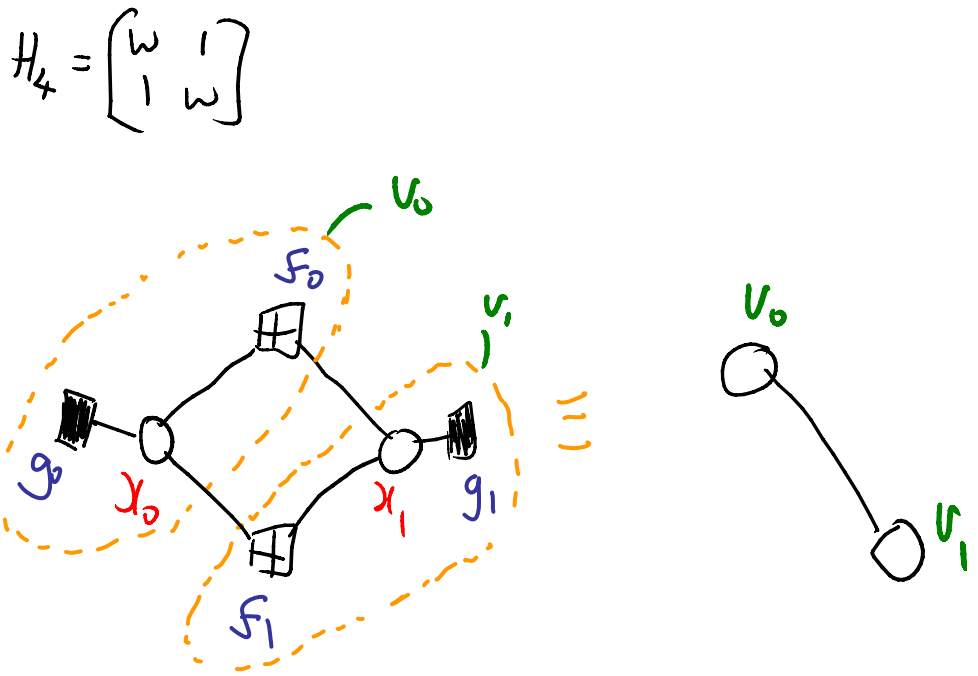

Figure 1: Factor graph for a simple graph

In Fig 1 the received soft information is passed into the graph by the functions $g_0 = (p_{0,0}, p_{1,0}, p_{w,0}, p_{w^2,0})$ and $g_1 = (p_{0,1}, p_{1,1}, p_{w,1}, p_{w^2,1})$. It is helpful to consider the situation as a truth-table:

x_0	x_1	g_0	g_1	f_0	f_1	$f_0 f_1$
0	0	$p_{0,0}$	$p_{0,1}$	1	1	1
0	1	$p_{0,0}$	$p_{1,1}$	1	0	0
0	w	$p_{0,0}$	$p_{w,1}$	0	1	0
0	w^2	$p_{0,0}$	$p_{w^2,1}$	0	0	0
1	0	$p_{1,0}$	$p_{0,1}$	0	1	0
1	1	$p_{1,0}$	$p_{1,1}$	0	0	0
1	w	$p_{1,0}$	$p_{w,1}$	1	1	1
1	w^2	$p_{1,0}$	$p_{w^2,1}$	1	0	0
w	0	$p_{w,0}$	$p_{0,1}$	1	0	0
w	1	$p_{w,0}$	$p_{1,1}$	1	1	1
w	w	$p_{w,0}$	$p_{w,1}$	0	0	0
w	w^2	$p_{w,0}$	$p_{w^2,1}$	0	1	0
w^2	0	$p_{w^2,0}$	$p_{0,1}$	0	0	0
w^2	1	$p_{w^2,0}$	$p_{1,1}$	0	1	0
w^2	w	$p_{w^2,0}$	$p_{w,1}$	1	0	0
w^2	w^2	$p_{w^2,0}$	$p_{w^2,1}$	1	1	1

The two constraints, f_0, f_1 are represented as indicator functions $f_j : \mathbb{F}_4^2 \to \mathcal{R}$, and the one positions of their product $f_0 f_1$ identify the codewords defined by the parity-check matrix, H_4 and generated by the matrix $G_4 = H_4$. These codewords are $(0,0), (1,w), (w,1), (w^2, w^2)$. Note from Fig 1 that the vertex v_j comprises the 4-state variable x_j, the 16-state constraint f_j, and the 4-state soft input g_j. We must consider the following messages:

- Two messages from $v_0 \to v_1$:
  - $f_0 \to x_1$ (length-4 vector: v_0's belief on v_1).
  - $x_0 \to f_1$ (length-4 vector: v_0's projected self-belief).
- Two messages from $v_1 \to v_0$:
  - $f_1 \to x_0$ (length-4 vector: v_1's belief on v_0).
  - $x_1 \to f_0$ (length-4 vector: v_1's projected self-belief).
- Three internal messages for v_0:
  - $g_0 \to x_0$ (length-4 vector: public belief on v_0).
  - $f_0 \to x_0$ (length-4 vector: v_0's contextual self-belief).

- $x_0 \to f_0$ (length-4 vector: v_0's internal self-belief).

- Three internal messages for v_1:
  - $g_1 \to x_1$ (length-4 vector: public belief on v_1).
  - $f_1 \to x_1$ (length-4 vector: v_1's contextual self-belief).
  - $x_1 \to f_1$ (length-4 vector: v_1's internal self-belief).

For example, after the information from g_0 has reached x_0, and assuming that $f_1 \to x_0$ is initially $(1,1,1,1)$, the next message $x_0 \to f_0$ is

$$x_0 \to f_0 = (f_1 \to x_0)g_0 = g_0 = (p_{0,0}, p_{1,0}, p_{w,0}, p_{w^2,0})^T.$$

After the information from g_0 has reached f_0 the next message $f_0 \to x_1$ is

$$f_0 \to x_1 = \sum_{x_0} f_0(x_0 \to f_0) = \begin{pmatrix} p_{0,0}+p_{w,0} & p_{0,0}+p_{w,0} & p_{1,0}+p_{w^2,0} & p_{1,0}+p_{w^2,0} \end{pmatrix}^T.$$

After the information from g_0 has reached f_1 as the message $x_0 \to f_1$, the next message $f_1 \to x_1$ is

$$f_1 \to x_1 = \sum_{x_0} f_1(x_0 \to f_1) = \begin{pmatrix} p_{0,0}+p_{1,0} & p_{w,0}+p_{w^2,0} & p_{0,0}+p_{1,0} & p_{w,0}+p_{w^2,0} \end{pmatrix}^T.$$

After the information from g_0 has reached x_1 the next message $x_1 \to f_0$ is

$$x_1 \to f_0 = (f_1 \to x_1)g_1 = \begin{pmatrix} p_{0,1}(p_{0,0}+p_{1,0}) & p_{1,1}(p_{w,0}+p_{w^2,0}) & p_{w,1}(p_{0,0}+p_{1,0}) & p_{w^2,1}(p_{w,0}+p_{w^2,0} \end{pmatrix}$$

After the information from g_0 has reached x_1 the next message $x_1 \to f_0$ is

$$x_1 \to f_0 = (f_1 \to x_1)g_1 = \begin{pmatrix} p_{0,1}(p_{0,0}+p_{1,0}) & p_{1,1}(p_{w,0}+p_{w^2,0}) & p_{w,1}(p_{0,0}+p_{1,0}) & p_{w^2,1}(p_{w,0}+p_{w^2,0} \end{pmatrix}$$

8.1 A binary interpretation?

We are considering the sum-product algorithm for $H_4 = \begin{pmatrix} w & 1 \\ 1 & w. \end{pmatrix}$. The equivalent binary parity-check matrix is $H = \left(\begin{array}{cc|cc} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{array}\right)$. Labeling the columns of H by x_{00}, x_{01}, x_{10}, and x_{11}, we can visualise H by the factor graph in Fig 2.

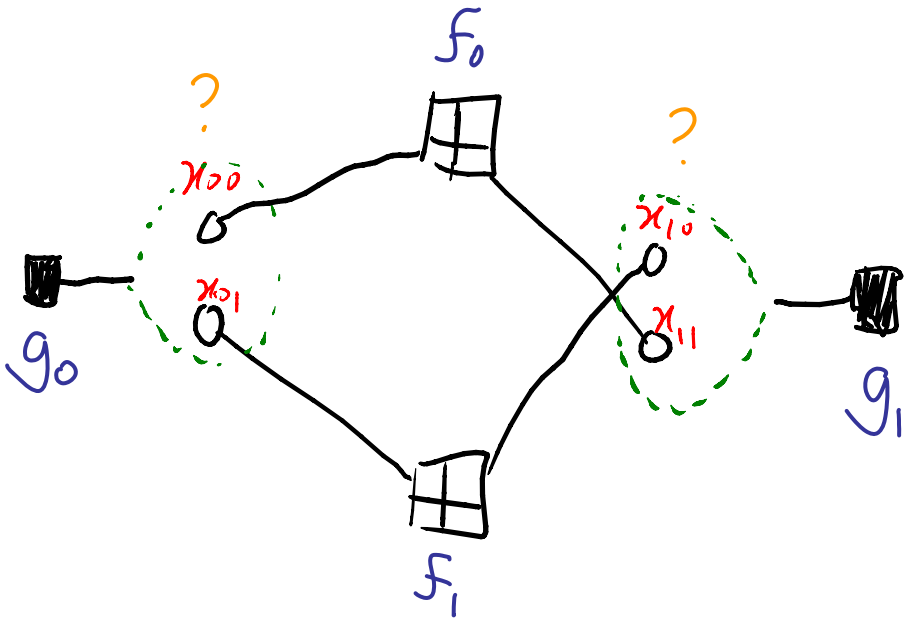

Figure 2: Factor graph for a simple graph in binary form

But, as identified in figure 2, there is a problem with passing the soft information from g_j to (x_{j0}, x_{j1}). This is because we are considering a quaternary symmetric channel - so g_j is a length-4 vector, but x_{j0} and x_{j1} are both length-2 vectors. This explains why, although the matrix H represents two *disjoint* binary codes, the quaternary channel 'joins' these two codes together. However, if we consider the approximation $g_j \approx x_{j0} \otimes x_{j1}$ then we reduce the system to standard binary message-passing on two parallel binary codes.

Note that the binary representation of the graph form does not always lead to disjoint codes. Consider the binary representation of the simple graph of fig 3.

One sees that the binary factor graph of fig 3 is not disjoint. Moreover it has a cycle. The general rule is as follows.

Lemma 8 *The binary factor graph representation of the simple graph,* $\mathcal{G}$*,*

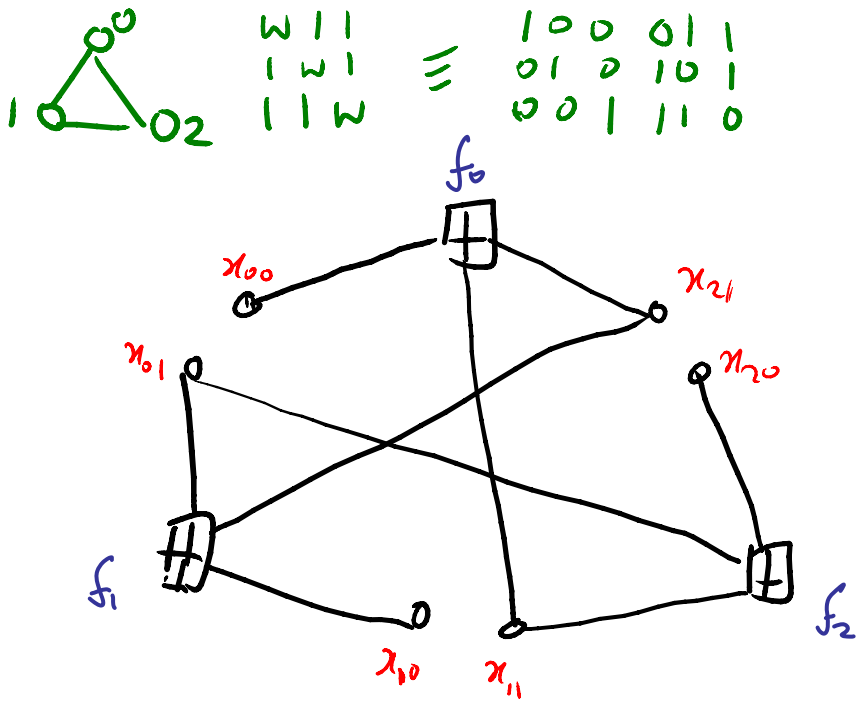

Figure 3: Factor graph for K_3 in binary form

splits into two disjoint parts iff $\mathcal{G}$ is bipartite.

Note, however, that $LC_0(K_3) = 01, 02$ which is a bipartite graph for which the binary factor graph representation splits into two disjoint parts. We are particularly interested in graphs, $\mathcal{G}$, that do not have a bipartite graph in their LC orbit. The first graph of this type is $C_5 = 01, 12, 23, 34, 40$ - up to relabelling the LC orbit of C_5 comprises three graphs, none of which are bipartite. Moreover, it is the graphs which do not have a bipartite member in their LC orbit that potentially have the highest distance when viewed as $\mathbb{F}_4$-additive codes. For instance the self-dual $\mathbb{F}_4$-additive code generated by C_5 has distance 3, which is the highest distance over all length-5 half-rate $\mathbb{F}_4$-additive codes.

9 Local complementation as a matrix transform

The simple graph $\mathcal{G}$ of n vertices can be represented by the homogeneous quadratic Boolean function, $g(x_0, x_1, \ldots, x_{n-1}) : \mathbb{F}_2^n \to \mathbb{F}_2$, where $x_i x_j$ is a monomial in g iff (i, j) is an edge in $\mathcal{G}$. For instance, the 3-vertex graph $\mathcal{G} = 01, 02$ can be represented by $g(x_0, x_1, x_2) = x_0x_1 + x_0x_2$. Let $N = \frac{\alpha}{\sqrt{2}} \begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix}$ be a 2×2 unitary matrix, where $i = \sqrt{-1}$ and $\alpha = e^{-\pi i/12}$. Let $D = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$. For a vertex v, let $\mathcal{B}_v(\mathcal{G})$ be the set of vertices in $\mathcal{G}$ that are either connected to v by an edge (i.e. the neighbours of v) or v itself. In other words $\mathcal{B}_v(\mathcal{G})$ is the ball of vertices in $\mathcal{G}$ of radius 1 with v at the centre. Let U be a 2×2 unitary matrix over the complex numbers, let '$\otimes$' be the tensor product (or Kronecker product), and let $U_k = I \otimes I \otimes \ldots \otimes I \otimes U \otimes I \otimes \ldots \otimes I$ be the $2^n \times 2^n$ unitary matrix where U acts at tensor position k, and where $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. For instance, for $n = 4$, then $N_2 = I \otimes I \otimes N \otimes I$. Let S be a subset of the integers $\{0, 1, \ldots, n-1\}$. Then U_S is the $2^n \times 2^n$ unitary matrix where U acts at all tensor positions with indices in S. For instance, for $n = 4$ and $S = \{0, 2\}$, then $N_S = N \otimes I \otimes N \otimes I$.

If $\mathcal{G}$ is represented by the Boolean function g, then we say that $\mathcal{G}^v$ (being the graph obtained from $\mathcal{G}$ by local complementation at vertex v) is represented by the Boolean function g^v.

Lemma 9

$$(-1)^{g^v} = \alpha^2 D_{\mathcal{B}_v(\mathcal{G})} N_v (-1)^g.$$

Proof. Proved elsewhere (double check that it is really α^2. □

For instance, if $n = 4$ and $g(x_0, x_1, x_2, x_3) = x_0x_1 + x_0x_2 + x_2x_3$ then $\mathcal{B}_0(\mathcal{G}) = \{0, 1, 2\}$, $D_{\mathcal{B}_0(\mathcal{G})} = D \otimes D \otimes D \otimes I$, $N_0 = N \otimes I \otimes I \otimes I$, $D_{\mathcal{B}_0(\mathcal{G})} N_0 = DN \otimes D \otimes D \otimes I$, and $g^0(x_0, x_1, x_2, x_3) = x_0x_1 + x_0x_2 + x_1x_2 + x_2x_3$.

Let $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ and $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$. Now observe that $N^3 = I$, $ND = \alpha^{19} XDN^2$, $NX = \alpha^{19} XZN$, and $NZ = \alpha XN$. Therefore any re-

peated local complementations on $\mathcal{G}$, can be written in terms of a Boolean function g', where

$$(-1)^{g'} = \alpha^k X_P Z_Q D_R N_S N_T^2 (-1)^g, \tag{1}$$

for some integer k, and for some subsets P, Q, R, S, T of $\{0, 1, \ldots, n-1\}$, where $S \cap T = \emptyset$.

We can re-express (1) as

$$(-1)^{g'} = VU(-1)^g,$$

where $V = \alpha^k X_P Z_Q D_R$ and $U \in \{I, N, N^2\}^{\otimes n}$. The part of the LC action that modifies edges is characterised by U, and V is an adjustment to eliminate affine terms, mod 8, for g', thereby ensuring that g' is a homogeneous quadratic, i.e. the action of V does not modify the quadratic part of g'. U can be one of 3^n transforms. For instance, for $n = 3$, then $U \in \{I \otimes I \otimes I, I \otimes I \otimes N, I \otimes I \otimes N^2, I \otimes N \otimes I, \ldots, N^2 \otimes N^2 \otimes N, N^2 \otimes N^2 \otimes N^2\}$.

Let $\mathrm{LC}(\mathcal{G})$ be the LC orbit of $\mathcal{G}$.

Lemma 10

$$|LC(\mathcal{G})| \leq 3^n.$$

Proof: As $U \in \{I, N, N^2\}^{\otimes n}$, then U can be one of 3^n matrices, each one representing a different transform of $(-1)^g$, namely $U(-1)^g$. A subset of these U can be identified (to within a post-multiplication by V) with a certain sequence of LC operations on $\mathcal{G}$. Moreover, from lemma 9 and (1), any sequence of LC operations can be identified with a unique $U \in \{I, N, N^2\}^{\otimes n}$. Therefore the maximum size of an LC orbit is 3^n.

The upper-bound of lemma 10 is very loose and it remains open to derive a tight upper bound on the size of an LC orbit both for the case of labeled and unlabeled graphs (where graph isomormphism reduces the graph count).

Let $P_b(\mathcal{G})$ be the probability that the LC orbit of $\mathcal{G}$ contains a bipartite graph, given that $\mathcal{G}$ is selected at random from the set of n-vertex simple graphs.

Lemma 11

$$P_b(\mathcal{G}) \to 0, \qquad as\ n \to \infty.$$

Proof. A lower-bound on the number of n-vertex simple graphs can be obtained from the literature. Let's call this bound $R(\mathcal{G})$. Similarly an upper-bound on the number of n-vertex simple bipartite graphs can be obtained from the literature. Let's call this bound $B(\mathcal{G})$. Then the best we can hope for is that each LC orbit is of size 3^n and each LC orbit contains just one bipartite graph, in which case we can cover $3^n B(\mathcal{G})$ graphs. Therefore $P_b(\mathcal{G}) \leq \frac{3^n B(\mathcal{G})}{R(\mathcal{G})}$. But from the values of $R(\mathcal{G})$ and $B(\mathcal{G})$ in the literature we see that $\frac{3^n B(\mathcal{G})}{R(\mathcal{G})} \to 0$ as $n \to \infty$. (Of course this proof is not complete until I have found some equations for $R(\mathcal{G})$ and $B(\mathcal{G})$, but they will exist somewhere in the literature I am sure, and will satisfy the asymptote that I claim). □

10 Generalising to directed simple graphs

Our model up until now has considered only simple graphs, i.e. where $K \circ A$ is symmetric and, moreover, we have focused on Hermitian self-dual $\mathbb{F}_4$-additive codes. A recent paper classified all half-rate $\mathbb{F}_4$-additive codes by interpreting them as directed simple graphs [3], where a directed simple graph no longer requires that its adjacency matrix, Γ, be symmetric. This motivates us to consider whether we can generalise the idea of message-passing on simple graphs to directed simple graphs. This is what we do in this section.

10.1 Local complementation on a directed simple graph

Local complementation on a directed simple graph operates on directed simple graphs and is a generalisation of local complementation on simple graphs. Let v be a vertex in a directed simple graph, $\mathcal{G}$, and let $\mathcal{N}_v$ be the set of neighbouring vertices of v in $\mathcal{G}$. Let $\mathcal{G}$ be represented by its adjacency matrix, $\Gamma = (\Gamma_{ij})$. We say that there is an arrow from vertex i to vertex j of G iff $\Gamma_{ij} = 1$, $i \neq j$. We say that there is an edge from i to j via v iff there is an arrow from i to v and an arrow from v to j.

Local complementation, $\mathcal{G}^v = \mathbf{LC}_v(\mathcal{G})$:
For all vertex pairs $(i, j) \in \mathcal{N}_v \times \mathcal{N}_v$, $i, j \neq v$, if there is an arrow from i to j via v, then add (resp. remove) to $\mathcal{G}$ an arrow from i to j if, currently, there doesn't exist (resp. does exist) an arrow from i to j.

It can easily be verified that this definition of LC on directed simple graphs is a generalisation of LC on (undirected) simple graphs.

We can also describe this LC rule in terms of operations on the adjacency matrix Γ:

Local complementation (adjacency matrix), Γ^v:

$\Gamma^v = \Gamma$

$\forall i \neq v$:

if $\Gamma_{iv} = 1$

$\forall j \neq v, i$:

$\Gamma^0_{ij} = \Gamma_{ij} + \Gamma_{vj}, \mod 2.$

The LC operation as described above for the adjacency matrix Γ is not a series of row additions, and this highlights that LC does not, in general, preserve the code. But we can show that the code is preserved up to permutations of the elements in certain columns. To see this, consider how LC acts on $H_4 = wI + \Gamma$, being a series of row additions followed by permutations of elements in a subset of the columns. Let $\theta : \mathbb{F}_4^* \to \mathbb{F}_4^*$, where $\mathbb{F}_4^* = \{1, w, w^2\}$ be the permutation $\theta(1) = w^2$, $\theta(w) = w$, and $\theta(w^2) = 1$. Similarly, let $\phi : \mathbb{F}_4^* \to \mathbb{F}_4^*$ be the permutation $\phi(1) = 1$, $\phi(w) = w^2$, and $\phi(w^2) = w$. We use the notation $\theta_j(H_4)$ to mean that θ is applied to every element in column j of H_4 (and similarly for $\phi_j(H_4)$):

Local complementation (H_4 matrix in graph form $wI + \Gamma$), H_4^v:

$H_4^v = H_4$

$\forall i \neq v$:

if $H_{4,iv} = 1$

$\forall j$:

$H^v_{4,ij} = H_{4,ij} + H_{4,vj}, \mod 2.$

$H_4^v = \theta_v(H_4^v)$
$\forall i \neq v$:

if $(H_{4,iv} H_{4,vi} = 1)$

$H_4^v = \phi_i(H_4^v)$

It easy to see that LC on Γ and on H_4, as described above, are equivalent, so the code is preserved up to permutations of the non-zero $\mathbb{F}_4$ elements of a subset of the columns.

Consider the example of a directed graph in fig 4 before and after an LC acting on vertex 0.

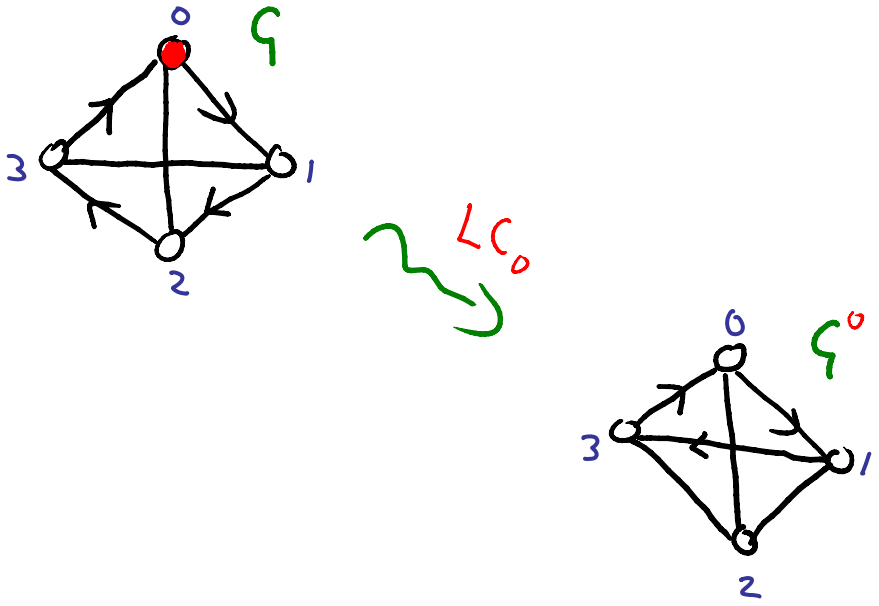

Figure 4: Local complementation on vertex 0 of a directed simple graph

In terms of the adjacency matrix, Γ, we have:

$$\Gamma = \begin{matrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{matrix}. \text{ Then } \Gamma^0 = \begin{matrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{matrix}.$$

For the parity-check matrix, H_4, we obtain H_4^0 as follows:

Let $H_4 = \begin{matrix} w & 1 & 1 & 0 \\ 0 & w & 1 & 1 \\ 1 & 0 & w & 1 \\ 1 & 1 & 0 & w \end{matrix}$. Then, to begin with, $H_4^0 = \begin{matrix} w & 1 & 1 & 0 \\ 0 & w & 1 & 1 \\ w^2 & 1 & w^2 & 1 \\ w^2 & 0 & 1 & w \end{matrix}$, and, after permuting the elements in column 0 by θ and the elements in columns 2 by ϕ, we obtain. $H_4^0 = \begin{matrix} w & 1 & 1 & 0 \\ 0 & w & 1 & 1 \\ 1 & 1 & w & 1 \\ 1 & 0 & 1 & w \end{matrix}$

We see that $H_4^0 = wI + \Gamma^0$ as required.

The LC rule for a directed graph may well have been previously defined by Bouchet in [1] but I haven't had time yet to check whether Bouchet's definition is the same.

Define a *unicycle* of an n-vertex directed graph, $\mathcal{G}$, to be a subset, S, of the n vertices such that at least one uni-directional cycle exists in $\mathcal{G}$ comprising all vertices in S. Such a cycle must not be bi-directional.

Lemma 12 *Let $\mathcal{G}$ be an n-vertex directed simple graph, and let $U \subset \{0, 1, \ldots, n-1\}$ comprise all vertices of $\mathcal{G}$ that are involved in one or more unicycles and, consequently, $\{0, 1, \ldots, n-1\} \setminus U$ are those vertices not involved in any unicycles. Then local complementation at any vertex v of $\mathcal{G}$ preserves the sets U and $\{0, 1, \ldots, n-1\} \setminus U$.*

Proof. Let (i, j) be an edge in one or more unicycles of $\mathcal{G}$. W.l.o.g. let there be an arrow from vertex i to vertex j that contributes to a unicycle, S, (there may or may not be an arrow from j to i). Then the arrow from i to j, and hence the unicycle S, can only be removed by LC if there is another vertex, v, such that the edges (i, v) and (v, j) exist, with arrows from i to v and from v to j (there may or may not exist arrows from v to i and/or from j to v). Performing LC at vertex v then removes the arrow from i to j. But, prior to LC at v, if such a path from i to j via v exists, then,

- if $v \notin S$, then i, v, j are members of the unicycle $S \cup \{v\}$, and remain so after LC at v.
- if $v \in S$, then there exist two unicycles, S', and S'', where $\{i, v\} \subset S'$, $j \notin S'$ and $\{v, j\} \subset S''$, $i \notin S''$, and where $S' \cup S'' = S$. After LC at v, both unicycles S' and S'' remain.

So, irrespective of whether $v \in S$ or not, after LC at v all members of $S \cup \{v\}$ remain part of some unicycle or other. The same argument can be used for any unicycle which is removed by LC at v. As LC is an involution, all vertices which are not part of a unicycle must also remain not part of a unicycle after LC at v. □

10.2 The sum-product algorithm for directed simple graph decoding on a quaternary symmetric channel

Consider $H_4 = \begin{pmatrix} w & 1 \\ 0 & w \end{pmatrix}$. The associated graph is a directed graph and the code constrained by H_4 is no longer self-dual. However it is still $\mathbb{F}_4$-additive. It turns out that we can still message-pass on this graph. The factor graph for H_4 is shown in fig 5.

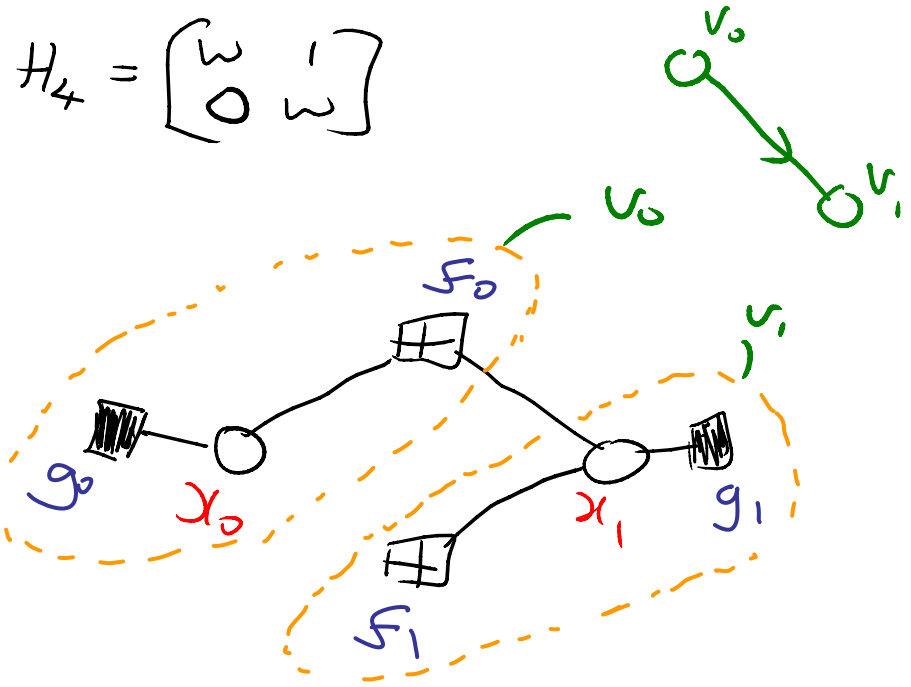

Figure 5: Factor graph for simple directed graph

We need only consider the following messages (in general, x_0 and f_1 may

be connected to external graph nodes - clearly this is not the case in this explicit example):

- One message from $v_0 \to v_1$:
  - $f_0 \to x_1$ (length-4 vector: v_0's belief on v_1).
- One message from $v_1 \to v_0$:
  - $x_1 \to f_0$ (length-4 vector: v_1's projected self-belief).
- Three internal messages for v_0:
  - $g_0 \to x_0$ (length-4 vector: public belief on v_0).
  - $f_0 \to x_0$ (length-4 vector: v_0's contextual self-belief).
  - $x_0 \to f_0$ (length-4 vector: v_0's internal self-belief).
- Three internal messages for v_1:
  - $g_1 \to x_1$ (length-4 vector: public belief on v_1).
  - $f_1 \to x_1$ (length-4 vector: v_1's contextual self-belief).
  - $x_1 \to f_1$ (length-4 vector: v_1's internal self-belief).

In the directed scenario of fig 5 v_1 sends information about itself to v_0 but v_0 does not send information about itself to v_1. Conversely v_0 sends its belief about v_1 to v_1 but v_1 does not send its belief about v_0 to v_0. Thus the directed graph implies a *hierarchy* of vertices, where information flows against the direction of the arrow and where 'control' (belief) flows in the direction of the arrow - one can imagine many levels of hierarchy, depending on the arrow structure.

For some blocklengths the directed graphs outperform the undirected graphs for code distance. For example, the self-dual $\mathbb{F}_4$-additive codes of length 7 are of distance 3 or less. But there exists an $\mathbb{F}_4$-additive code of length 7 and distance 4 that is not self-dual. It is represented by the directed graph shown in fig 6.

It turns out (apart from trivial exceptions) that all half-rate $\mathbb{F}_4$-additive codes can be represented, to within column permutations, by directed graphs. A classification of these half-rate $\mathbb{F}_4$-additive codes using the directed graph approach can be found in [3].

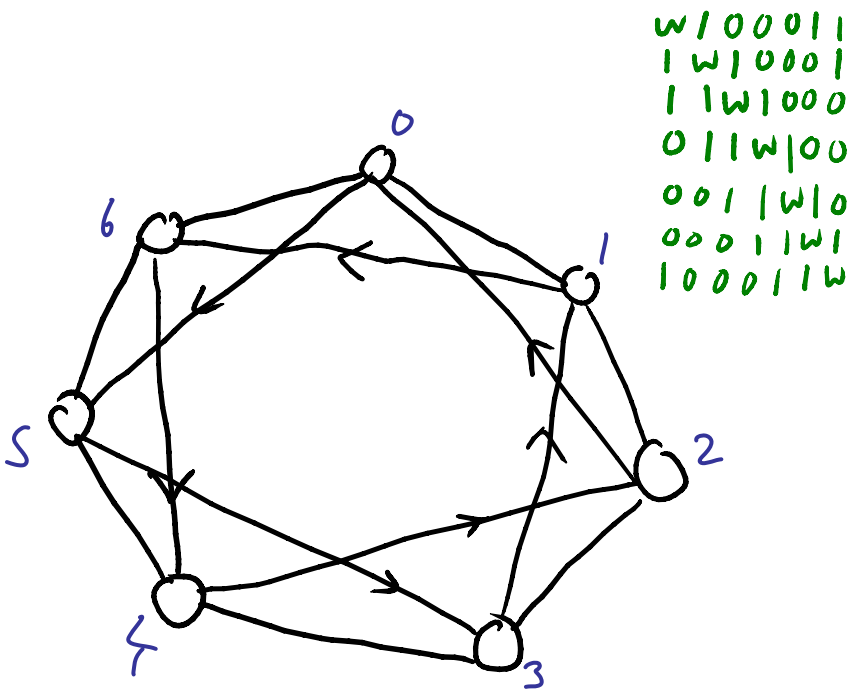

Figure 6: Factor graph for simple directed graph

10.3 Message-passing on Dynamic Graphs

Our aim now is to show that we can extend the implementation of the sum-product algorithm for directed simple graphs to dynamic scenarios where the graph changes during decoding, and where simple graphs are a special case of directed simple graphs. Thus we wish to do *directed simple graph dynamic decoding* where LC is used to update the graphs. The problem here, as mentioned before, is that the matrix version of LC destroys the graph form of H_4 so the information provided by Γ is not, in general, sufficient to describe the complete decoding process. We show that directed simple graph dynamic decoding can still be realised, however, by the three step process:

- Permutation of the elements of each of the received 4-state symbol vectors.
- directed simple graph decoding

- Inverse permutation of the elements of each of the soft-decoded 4-state symbol vectors.

The elemental permutations, as mentioned previously, are:

- $\theta : \mathbb{F}_4^* \to \mathbb{F}_4^* : w^2 \leftrightarrow 1$, w unchanged.
- $\phi : \mathbb{F}_4^* \to \mathbb{F}_4^* : w^2 \leftrightarrow w$, 1 unchanged.

Using these two permutations, and observing that $\theta^2 = \phi^2 = I$, and $\theta\phi\theta = \phi\theta\phi$, we can represent the 6 permutations of $\mathbb{F}_4^* = \{1, w, w^2\}$, i.e. the set $\mathcal{P}_4$, by

$$\mathcal{P}_4 = \{I, \theta, \phi, \phi\theta, \theta\phi, \theta\phi\theta\},$$

where I is, in this context, the identity permutation.

Consider doing matrix LC on row 0 (vertex 0) of H_4 for our C_5 simple graph example, i.e. where

$$H_4 = \begin{matrix} w & 1 & 0 & 0 & 1 \\ 1 & w & 1 & 0 & 0 \\ 0 & 1 & w & 1 & 0 \\ 0 & 0 & 1 & w & 1 \\ 1 & 0 & 0 & 1 & w. \end{matrix}$$

Then

$$H_4^0 = \begin{matrix} w & 1 & 0 & 0 & 1 \\ w^2 & w^2 & 1 & 0 & 1 \\ 0 & 1 & w & 1 & 0 \\ 0 & 0 & 1 & w & 1 \\ w^2 & 1 & 0 & 1 & w^2. \end{matrix}$$

Noting that vertex 0 of $\mathcal{G}$ has neighbours 1 and 4, we can convert H_4^0 back to graph form by applying θ to column 0, and ϕ to columns 1 and 4, to get,

$$\tilde{H_4^0} = \phi_1\phi_4\theta_0(H_4^0) = wI + \Gamma^0 = \begin{matrix} w & 1 & 0 & 0 & 1 \\ 1 & w & 1 & 0 & 1 \\ 0 & 1 & w & 1 & 0 \\ 0 & 0 & 1 & w & 1 \\ 1 & 1 & 0 & 1 & w. \end{matrix}$$

Therefore, to do the correct message-passing decoding on $\mathcal{G}^0$, we first have to apply permutation θ to the received 4-state soft vector at index 0, and

apply ϕ to the received 4-state soft vectors at indices 1 and 4, then apply (directed) simple graph decoding using 4-state message-passing. Finally we apply $\theta^{-1} = \theta$, and $\phi^{-1} = \phi$, to the decoded 4-state soft vectors at index 0, and at indices 1 and 4, respectively.

Similarly,

$$(H_4^0)^1 = \begin{array}{ccccc} 1 & w & 1 & 0 & 0 \\ w^2 & w^2 & 1 & 0 & 1 \\ w^2 & w & w^2 & 1 & 1 \\ 0 & 0 & 1 & w & 1 \\ 0 & w & 1 & 1 & w. \end{array}$$

Noting that vertex 1 of $\mathcal{G}^0$ has neighbours 0, 2, and 4, we can convert $(H_4^0)^1$ back to graph form by applying θ to column 1, and ϕ to columns 0, 2, and 4, then, for the previous LC step, applying θ to column 0, and ϕ to columns 1 and 4. Combining these two steps, we can convert $(H_4^0)^1$ back to graph form by applying $\phi\theta$ (reading from the right) to column 0, $\theta\phi$ to column 1, ϕ to column 2, I to column 3, and $\phi^2 = I$ to column 4, to give

$$(\tilde{H_4^0})^1 = \phi_2\theta_1\phi_1\phi_0\theta_0((H_4^0)^1) = wI + (\Gamma^0)^1 = \begin{array}{ccccc} w & 1 & 1 & 0 & 0 \\ 1 & w & 1 & 0 & 1 \\ 1 & 1 & w & 1 & 1 \\ 0 & 0 & 1 & w & 1 \\ 0 & 1 & 1 & 1 & w. \end{array}$$

Therefore, to do the correct message-passing decoding on $(\mathcal{G}^0)^1$, we first have to apply permutation $\phi\theta$ to the received 4-state soft vector at index 0, and apply $\theta\phi$ to the received 4-state soft vector at index 1, and ϕ to the received 4-state soft vector at index 2. Then apply (directed) simple graph decoding using 4-state message-passing. Finally, apply $(\phi\theta)^{-1} = \theta\phi$ to the decoded 4-state soft vector at index 0, $(\theta\phi)^{-1} = \phi\theta$ to the decoded 4-state soft vector at index 1, and $\phi^{-1} = \phi$ to the decoded 4-state soft vector at index 2.

... and so on .... The above example is for a simple graph, but it should be clear that exactly the same strategy applies for the more general class of directed simple graphs.

Therefore, to do directed simple graph dynamic decoding we need only keep a record of the current permutation to be applied at each symbol, being

one of six possible permutations for each symbol. This modification is of trivial complexity in comparison to the complexity of the message-passing.

****
****
THE STUFF FOLLOWING NEEDS TO BE RE-READ AND RE-CHECKED - SUGGEST READERS IGNORE FOR NOW.

11 Message-passing with dynamic symbol permutation

In this section we consider 4-state modulation over the quaternary symmetric channel, and propose to use 2-state decoding for codes defined by SD graph-class matrices (we could, alternatively, develop the arguments in terms of $\mathbb{F}_4$ graph codes, but it seems more natural to do it here in terms of the SD graph-class as we are considering 2-state decoding). The message-passing used is the standard 2-state sum-product algorithm. The receiver has to approximate each received 4-state soft vector by the tensor product of two 2-state soft vectors, before passing these two vectors to the decoder. However, this approximation allows for diversity, as the approximation depends on the 2-bit representation chosen for each 4-state symbol. Thus if, during decoding, we dynamically permute the representation of each symbol by a member of $\mathcal{P}$, then we can potentially enhance the performance of our decoding. This is a way of possibly compensating for the performance loss resulting from the approximation of the 4-state soft vector by two 2-state soft vectors.

This *dynamic symbol permutation decoding* preserves the graph, $\mathcal{G}$, and preserves symplectic self-duality and weight-distribution of the code. To further ensure correct decoding, one must keep track of the representational meaning of $00, 01, 10, 11$ for each symbol after the action of a member of $\mathcal{P}$ on one or more column pairs of H. Let $r_j = (p_{00,j}, p_{01,j}, p_{10,j}, p_{11,j})$ be the received soft information for the jth received symbol, and approximate r_j by $r_j \approx (p_{0,j}, p_{1,j}) \otimes (p_{0,j+n}, p_{1,j+n})$ for input to the 2-state decoder, where $p_{0,j} = p_{00,j} + p_{10,j}$, $p_{1,j} = p_{01,j} + p_{11,j}$, $p_{0,j+n} = p_{00,j} + p_{01,j}$, and $p_{1,j+n} = p_{10,j} + p_{11,j}$. Then applying $\sigma \in \mathcal{P}$ to the $(j, j+n)$ column pair of H takes r_j to $(p_{\sigma(00),j}, p_{\sigma(01),j}, p_{\sigma(10),j}, p_{\sigma(11),j})$, and now r_j is approximated by $r_j \approx$

$(p_{0,j}, p_{1,j}) \otimes (p_{0,j+n}, p_{1,j+n})$, where

$$\begin{aligned} p_{0,j} &= p_{\sigma(00),j} + p_{\sigma(10),j}, \\ p_{1,j} &= p_{\sigma(01),j} + p_{\sigma(11),j}, \\ p_{0,j+n} &= p_{\sigma(00),j} + p_{\sigma(01),j}, \\ p_{1,j+n} &= p_{\sigma(10),j} + p_{\sigma(11),j}. \end{aligned}$$

Such a permutation causes a linear update to the structure of the function nodes of the 2-state decoder, which translates into linear operations on the columns of H.

As an example of the effect of dynamic symbol permutation decoding on the approximations, and on the function nodes, consider, once again, the code, C, generated by the SD graph-class parity matrix:

$$H = (h_0, h_1, \ldots, h_{2n-1}) = \begin{array}{ccc} 10000 & | & 01001 \\ 01000 & | & 11100 \\ 00100 & | & 01010 \\ 00010 & | & 00101 \\ 00001 & | & 10011 \end{array} = I|A,$$

where, for simplicity, we have explicitly assigned the diagonal, and where h_j is the jth column of H. Applying $\sigma_{0,5} = (f_1^0, f_0^0)$ and $\sigma_{2,7} = (f_1^2, f_0^2)$ to each row of column-pairs (h_0, h_5), and (h_2, h_7), respectively, where, for binary variables, x_0, x_1,

$$\sigma_{0,5}: \quad \begin{array}{cc} x_1x_0 & \rightarrow \; f_1^0 f_0^0 \\ \hline 00 & \rightarrow \; 00 \\ 01 & \rightarrow \; 10 \\ 10 & \rightarrow \; 11 \\ 11 & \rightarrow \; 01, \end{array} \qquad f_0^0 = x_0 + x_1, \quad f_1^0 = x_1.$$

$$\sigma_{2,7}: \quad \begin{array}{cc} x_1x_0 & \rightarrow \; f_1^2 f_0^2 \\ \hline 00 & \rightarrow \; 00 \\ 01 & \rightarrow \; 11 \\ 10 & \rightarrow \; 10 \\ 11 & \rightarrow \; 01, \end{array} \qquad f_0^2 = x_0 + x_1, \quad f_1^2 = x_0.$$

leads to

$$H' = \sigma_{2,7}\sigma_{0,5}H = \begin{array}{ccc} 10000 & | & 11001 \\ 01100 & | & 11100 \\ 00000 & | & 01110 \\ 00110 & | & 00101 \\ 00001 & | & 10011 \end{array} = K'|A'.$$

Writing H in terms of its columns, $H = (h_0, h_1, \ldots, h_9)$, we see that $H' = (f_1^0, h_1, f_1^2, h_3, h_4, f_0^0, h_6, f_0^2, h_8, h_9) = (h_0, h_1, h_7, h_3, h_4, h_0 + h_5, h_6, h_2 + h_7, h_8, h_9)$.

It is straightforward to check that this code is still in the SD graph-class and still has symplectic distance 3. Moreover, its associated graph is still $\mathcal{G}$. One can view it as exactly the same code as before where the code symbols are represented differently by the two bits for bit-pairs $(0, 5)$ and $(2, 7)$. Therefore we must update the corresponding 2-state soft inputs to the decoder, to reflect this different viewpoint, such that

$$\begin{aligned} p_{0,0} &= p_{00,0} + p_{11,0}, \\ p_{1,0} &= p_{10,0} + p_{01,0}, \\ p_{0,n} &= p_{00,0} + p_{10,0}, \\ p_{1,n} &= p_{11,0} + p_{01,0}, \end{aligned}$$

and

$$\begin{aligned} p_{0,2} &= p_{00,2} + p_{10,2}, \\ p_{1,2} &= p_{11,2} + p_{01,2}, \\ p_{0,2+n} &= p_{00,2} + p_{11,2}, \\ p_{1,2+n} &= p_{10,2} + p_{01,2}. \end{aligned}$$

It should be emphasised that dynamic symbol permutation is pointless if one is doing 4-state decoding, as any internal 're-arrangement' of the symbol statistics is, in that case, fully compensated for by the 4-state variable and function nodes. In other words there is no performance loss to be clawed back.

12 Using codes from the SD graph-class over the quaternary-symmetric channel

We restrict ourselves to a parity matrix, H, from the SD graph-class, whose associated graph is $\mathcal{G}$. Then we wish for two things from the length-$2n$, half-rate code, C, whose symplectic parity-check matrix is H:

- $[2n, n, d]$ has d large, maximal or optimal.
- $\mathcal{G}$ is sparse.

Let $G = H$ be the corresponding $n \times 2n$ symplectic self-dual generator matrix from the SD graph-class.

We consider/propose some specific coding/decoding scenarios.

12.1 4-state modulation, 4-state decoding

Encoder/transmitter:

- Construct codewords as $c = (c_0, c_1, \ldots, c_{2n-1}) = aG \in (\mathbb{F}_2^2)^n$, given information vectors $a \in \mathbb{F}_2^n$.
- Send n bit-pairs, $\in \mathbb{F}_2^2$, of each codeword as n 4-state modulated symbols over a quaternary-symmetric channel.

Receiver/decoder:

- Each codeword is received as n length-4 probability vectors $r = (r_0, r_1, \ldots, r_{n-1})$, where, for $0 \leq j < n$, $r_j = (p_{00,j}, p_{01,j}, p_{10,j}, p_{11,j})$, such that $p_{00,j} + p_{01,j} + p_{10,j} + p_{11,j} = 1$, and $p_{ab,j}$ is the probability that the transmitted bit-pair, (c_j, c_{n+j}), is (a, b), $a, b \in \mathbb{F}_2$.
- Soft vectors, r_j, are passed to a standard 4-state message-passing decoder, using a 4-state version of the sum-product algorithm, where all variable nodes are 4-state, and where all function nodes take in and give out 4-state probability vectors. Choose the graph code parity matrix in graph form to realise the decoding using simple graph decoding, in which case a simplified version of 4-state decoding can be implemented.

Implementation considerations: For 2-state message-passing it is normal to use log-likelihood ratios. This is because each soft vector input to the decoder is of the form $r_j = (p_{0,j}, p_{1,j}) = 1$, where $p_{0,j} + p_{1,j} = 1$, so we only need to represent one real number, due to the normalisation requirement. But, for 4-state message-passing, we have three independent real numbers to represent after normalisation. Probably good solutions to this already exist in the literature, e.g. MacKay, Declercq, Mickey Mouse, ..., but maybe the intuitively natural LLRs to use are l_a, l_b, and l_{ab}, where

$$l_a = \log_2 \frac{p_{00,j} + p_{01,j}}{p_{10,j} + p_{11,j}}, \quad l_b = \log_2 \frac{p_{00,j} + p_{10,j}}{p_{01,j} + p_{11,j}}, \quad l_{ab} = \log_2 \frac{p_{00,j} + p_{11,j}}{p_{01,j} + p_{10,j}}.$$

Then one re-expresses all SPA operations in terms of l_a, l_b, and l_{ab}.

... go on then ... ??

12.2 4-state modulation, 4-state decoding + modifications

For the previous scheme we propose the following possible enhancements:

- Choose graph codes where $\mathcal{G}$ is sparse but distance is large - e.g. the nested-clique graphs that we have found.
- Simple graph dynamic decoding via interleaved, 'random', SD graph-class local complementations - preserves the code and traverses the local complementation orbit of the graph. Choose the graph code parity matrix in graph form.

12.3 4-state modulation, 2-state decoding

Encoder/transmitter:

- Construct codewords as $c = (c_0, c_1, \ldots, c_{2n-1}) = aG \in (\mathbb{F}_2^2)^n$, given information vectors $a \in \mathbb{F}_2^n$.
- Send n bit-pairs, $\in \mathbb{F}_2^2$, of each codeword as n 4-state modulated symbols over a quaternary-symmetric channel.

Receiver/decoder:

- Each codeword is received as n length-4 probability vectors $r = (r_0, r_1, \ldots, r_{n-1})$, where, for $0 \leq j < n$, $r_j = (p_{00,j}, p_{01,j}, p_{10,j}, p_{11,j})$, such that $p_{00,j} + p_{01,j} + p_{10,j} + p_{11,j} = 1$, and $p_{ab,j}$ is the probability that the transmitted bit-pair, (c_j, c_{n+j}), is (a, b), $a, b \in \mathbb{F}_2$.
- The length-4 received soft vectors, r_j, are approximated by two length-2 soft vectors, i.e. $r_j \approx (p_{0,j}, p_{1,j}) \otimes (p_{0,j+n}, p_{1,j+n})$, where $p_{0,j} = p_{00,j} + p_{10,j}$, $p_{1,j} = p_{01,j} + p_{11,j}$, $p_{0,j+n} = p_{00,j} + p_{01,j}$, and $p_{1,j+n} = p_{10,j} + p_{11,j}$. These length-2 soft vectors are then passed to a 2-state message-passing decoder, where all variable nodes are 2-state, and where all function nodes take in and give out 2-state probability vectors. Choose the graph code parity matrix in graph form to realise using simple graph decoding, in which case a simplified version of 4-state dedcoding can be implemented.

Implementation considerations: For 2-state message-passing it is normal to use log-likelihood ratios. In the scenario under consideration, these LLRs are $\log_2 \frac{p_{0,j}}{p_{1,j}}$ and $\log_2 \frac{p_{0,j+n}}{p_{1,j+n}}$.

12.4 4-state modulation, 2-state decoding + modifications

For the previous scheme we propose the following possible enhancements:

- Choose graph codes where $\mathcal{G}$ is sparse but distance is large - e.g. the nested-clique graphs that we have found.
- Simple graph dynamic decoding via interleaved, 'random', SD graph-class local complementations - preserves the code and traverses the local complementation orbit of the graph. Choose the graph code parity matrix in graph form, where the soft input permutations are performed before the approximation of the 4-state vector by two 2-state vectors.
- Dynamic symbol permutation decoding via interleaved, 'random', symbol permutations, $\mathcal{P}$.

12.5 2-state modulation, 4-state decoding

Encoder/transmitter:

- Construct codewords as $c = (c_0, c_1, \ldots, c_{2n-1}) = aG \in (\mathbb{F}_2^2)^n$, given information vectors $a \in \mathbb{F}_2^n$.
- Send $2n$ bits, $\in \mathbb{F}_2$, of each codeword as $2n$ 2-state modulated symbols over a binary-symmetric channel.

Receiver/decoder:

- Each codeword is received as $2n$ length-2 probability vectors $r = (r_0, r_1, \ldots, r_{2n-1})$, where, for $0 \leq j < 2n$, $r_j = (p_{0,j}, p_{1,j})$, such that $p_{0,j} + p_{1,j} = 1$, and $p_{a,j}$ is the probability that the transmitted bit, c_j, is $a \in \mathbb{F}_2$.
- Each pair of two length-2 received soft vectors, (r_j, r_{j+n}), is represented by one length-4 soft vector, s_j, i.e. $s_j = r_j \otimes r_{j+n}$, $0 \leq j < n$. The soft vectors, s_j, are passed to a standard 4-state message-passing decoder, using a 4-state version of the sum-product algorithm, where all variable nodes are 4-state, and where all function nodes take in and give out 4-state probability vectors. Choose the graph code parity matrix in graph form to realise using simple graph decoding, in which case a simplified version of 4-state dedcoding can be implemented.

Implementation considerations: For normal 4-state message-passing, we have three independent real numbers to represent after normalisation, per symbol. But in this case, each 4-state received soft vector is really the tensor product of two two-state soft vectors. So we have:

$$l_a = \log_2 \frac{p_{0,j}}{p_{1,j}}, \quad l_b = \log_2 \frac{p_{0,j+n}}{p_{1,j+n}}, \quad l_{ab} = \log_2 \frac{p_{0,j} + p_{1,j+n}}{p_{1,j} + p_{0,j+n}},$$

where l_{ab} is redundant at input to the decoder. But, internal to the decoder, all three LLR values are required.

12.6 2-state modulation, 4-state decoding + modifications

For the previous scheme we propose the following possible enhancements:

- Choose graph codes where $\mathcal{G}$ is sparse but distance is large - e.g. the nested-clique graphs that we have found.
- Simple graph dynamic decoding via interleaved, 'random', SD graph-class local complementations - preserves the code and traverses the local complementation orbit of the graph. Choose the graph code parity matrix in graph form.

12.7 4-state modulation, 2-state reception, 2-state decoding

Encoder/transmitter:

- Construct codewords as $c = (c_0, c_1, \ldots, c_{2n-1}) = aG \in (\mathbb{F}_2^2)^n$, given information vectors $a \in \mathbb{F}_2^n$.
- Send $2n$ bits, $\in \mathbb{F}_2$, of each codeword as n 4-state modulated symbols over a quaternary-symmetric channel, but receive as $2n$ 2-state soft vectors, two vectors for every 4-state symbol sent.

Receiver/decoder:

- Each codeword is received as $2n$ length-2 probability vectors $r = (r_0, r_1, \ldots, r_{2n-1})$, where, for $0 \leq j < 2n$, $r_j = (p_{0,j}, p_{1,j})$, such that $p_{0,j} + p_{1,j} = 1$, and $p_{a,j}$ is the probability that the transmitted bit, c_j, is $a \in \mathbb{F}_2$.

- The length-2 received soft vectors, r_j, are passed to a 2-state message-passing decoder, where all variable nodes are 2-state, and where all function nodes take in and give out 2-state probability vectors.

12.8 4-state modulation, 2-state reception, 2-state decoding + modifications

For the previous scheme we propose the following possible enhancements:

- Choose graph codes where $\mathcal{G}$ is sparse but distance is large - e.g. the nested-clique graphs that we have found.
- Simple graph dynamic decoding via interleaved, 'random', SD graph-class local complementations - preserves the code and traverses the local complementation orbit of the graph. Choose the graph code parity matrix in graph form, where the soft input permutations are performed on each pair of two 2-state vectors to realise two new 2-state vectors.
- Dynamic symbol permutation decoding via interleaved, 'random', symbol permutations, $\mathcal{P}$.

13 Edge local complementation for graph code

Edge-local complementation is just three local complementations, so is easy to characterise and implement at the simple graph level as we already know how to do local complementation on such graphs. It will, therefore, preserve both code and graph form.

Edge-local complementation can also be performed at the bit-level, i.e. on the binary code from the SD graph-class. In this case the code is preserved but, in general, the graph structure is not preserved. How do we ensure that the graph structure is preserved?

.....

References

[1] Andre Bouchet, Digraph decompositions and Eulerian systems, Siam J. Alg. Disc. Meth., **8** 3, July 1987.

[2] Lars Eirik Danielsen, Matthew G. Parker, On the classification of all self-dual additive codes over GF(4) of length up to 12, J. Combin. Theory Ser. A **113** (7), 1351–1367 (2006). arXiv:math.CO/0504522, also `http://www.ii.uib.no/~matthew/selfdual.pdf`

[3] Lars Eirik Danielsen, Matthew G. Parker, Directed graph representation of half-rate additive codes over GF(4), Des. Codes Cryptogr., **59**, 1–3, April 2011. arXiv:0902.3883, also `http://www.ii.uib.no/~matthew/directedFinal.pdf`