Adaptive Soft-Decision Iterative Decoding Using Edge Local Complementation

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

Dept. of Informatics, University of Bergen, Thormøhlensgt. 55, 5008 Bergen, Norway
{joakimk,riera,matthew,eirik}@ii.uib.no

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

1 Introduction

Inspired by the success of iterative decoding of LDPC codes, originally introduced by Gallager [1] and later rediscovered in the mid 1990's by MacKay and Neal [2], on a wide variety of communication channels, the idea of iterative, soft-decision decoding has recently been applied to classical algebraically constructed codes in order to achieve low-complexity Belief Propagation decoding [3,4,5]. Both Reed-Solomon and Bose-Chaudhuri-Hocquenghem (BCH) codes have been considered in the context of iterative decoding. Certain algebraically constructed bipartite graphs are known to exhibit good code properties, such as large minimum distance and a non-trivial automorphism group. However, these typical 'classical properties' do not necessarily lend themselves well to modern graph-based coding theory. Factors which influence the performance of iterative, soft-decision decoders are pseudo-codewords [6], stopping and trapping sets [7,8], sparsity, girth, and degree distributions [9]. Structural weaknesses of graphical codes are inherent to the particular parity-check matrix, H, which can be said to implement the code in the decoder. This matrix is a non-unique

A. Barbero (Ed.): ICMCTA, LNCS 5228, pp. 82–94, 2008.

$(n-k)$-dimensional basis for the null space of the code, $\mathcal{C}$, which, in turn, is a k-dimensional subspace of $\{0,1\}^n$. Although any basis (for the dual code, $\mathcal{C}^{\perp}$) is a parity-check matrix for $\mathcal{C}$, their performance in decoders is not uniform. In this work, we assume that H is of full rank and in 'standard' $[\, I \,|\, P \,]$-form, where I is the identity matrix.

We propose a class of adaptive decoders which facilitate message-passing on classical linear codes, by taking advantage of (non-trivial) graph structure. It is well known that H can be mapped into a bipartite (Tanner) graph, $\mathbf{TG}(H)$, which is described by its adjacency matrix, $\begin{pmatrix} 0 & H \\ H^T & 0 \end{pmatrix}$. With H being in standard form, a specific information set (on the codeword positions) is implied. We will refer to bit nodes (i.e., columns of H) corresponding to I and P as 'parity' and 'information' nodes, respectively,[1] and rows of H correspond to 'constraint' (check) nodes. Using a localized, low-complexity graph edge-operation, we update the parity-check matrix, but still stay within the automorphism group of the code, $\mathrm{Aut}(\mathcal{C})$. Thus, the graph update rule can be viewed as a particular relabelling (isomorphism) of the bit nodes. Furthermore, by selectively or randomly shifting sensitive substructures (e.g., short cycles, or weight-1 nodes) within the graph, we aim to influence the flow of extrinsic information through $\mathbf{TG}(H)$ in a way helpful to the decoding process.

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

2 Edge Local Complementation

The operation of edge local complementation (ELC) [10,11,12], also known as Pivot, is a local operation on a simple graph. Fig. 1(a) shows $G_{\mathcal{N}_u \cup \mathcal{N}_v}$, the local subgraph of a bipartite graph induced by nodes u, v, and their disjoint neighborhoods which we denote $\mathcal{N}_u^v \triangleq \mathcal{N}_u \setminus \{v\}$ and $\mathcal{N}_v^u \triangleq \mathcal{N}_v \setminus \{u\}$, respectively.

Pivot on a bipartite graph is described as the complementation of edges between these two sets; $\forall\ v' \in \mathcal{N}_u^v,\ u' \in \mathcal{N}_v^u$, check whether edge $(u', v') \in G$, in which case it is deleted (otherwise, it is created). Finally, the edges adjacent to u and v are swapped. As such, Pivot updates the set of constraints (rows of H) by changing the edges of $\mathbf{TG}(H)$, whereas nodes are invariant. The complexity of the graph-based algorithm is $\mathcal{O}(\deg(u)\deg(v))$. The fact that Pivot amounts to row

[1] Note that these terms refer to the generator matrix of the code, $G_{\mathcal{C}} \triangleq \left[\, P^T \,|\, I_k \,\right]$.

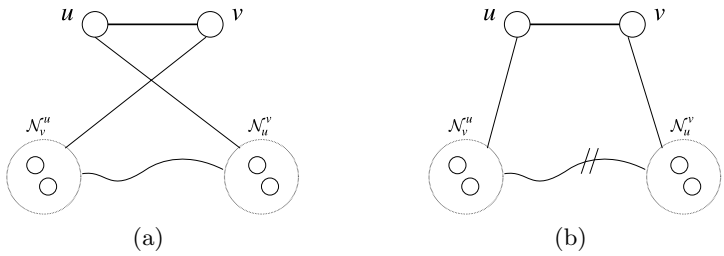


Fig. 1. Pivot (ELC) on edge (u, v) of a bipartite graph. Doubly slashed links mean the edges connecting two sets have been complemented.

additions assures that the code is preserved. In the following, we use the notation G^i to denote a graph G that has been subject to i Pivots (similarly for H^i).

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

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

As can be readily verified, although Pivot preserves the code, it can have a negative impact on parameters of its implementation, H, as a decoder. Edges complemented are at distance 2 from (u, v), so for a typical sparse, girth-6 graph, many 4-cycles result, and density increases [14]. Pivot does not generally preserve graph isomorphism (structure), so the operation will often given us a different structure in the (Pivot) *orbit* of G [15]. The matrices H^i in this orbit are the set of structurally different parity-check matrices for the same code, as discussed in the Introduction. We briefly mention that all information sets of $\mathcal{C}$ may be enumerated by traversing this orbit of G [11].

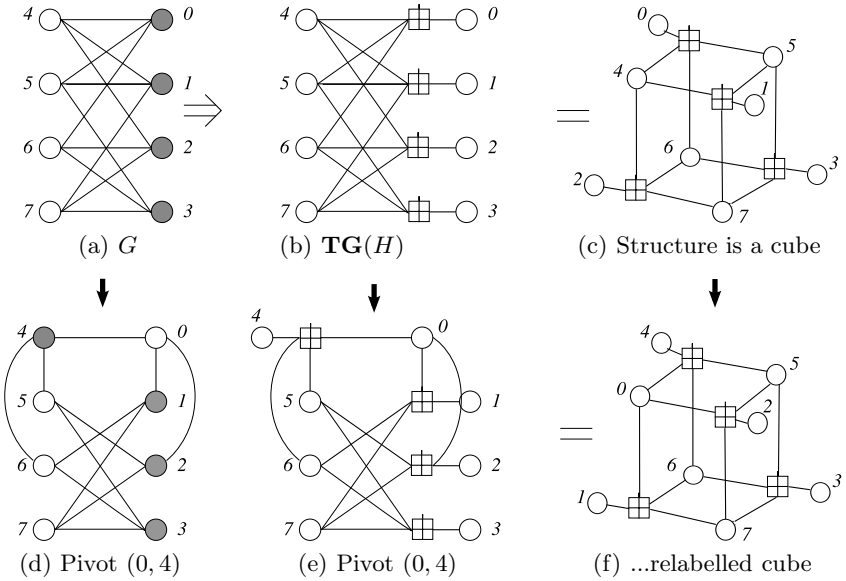


Fig. 2. (a) through (c) are three equivalent representations of the $(8, 4)$ extended Hamming code. (d) through (e) show the corresponding representation after Pivot is applied to edge $(0, 4)$. Fig. 5 shows the parity-check matrices of (b) and (e), respectively.

2.1 Iso-Pivot

In this section we describe an application of Pivot to preserve key features of the graph, to remedy the drawbacks enumerated in the previous section. We define Iso-Pivot as a sequence of Pivot operations over which (global) graph isomorphism is preserved. Such an operation will be in $\mathrm{Aut}(\mathcal{C})$, in that its action has the appearance of a relabelling on the nodes of a graph, or–equivalently–a permutation on the columns of a matrix (H). If there exist sequences of Pivots which preserve the structure of G, then $\mathrm{Aut}(\mathcal{C})$ must be non-trivial. Isomorphism is a certificate on the properties of the resulting graphs (matrices) used during decoding; that these remain the same as for the initial G (H), which can be assumed to have been carefully selected. The relabelling, however, alters the flow of messages in $\mathbf{TG}(H)$, i.e., which nodes are exchanging information. Note in particular how, after the Iso-Pivot in Fig. 2(f), node 4 is no longer part of a 4-cycle (whereas node 0 now is).

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

A. Most generally, to have an isomorphism, the number of edges in G must remain invariant under Pivot. Pivot is a local operation, so we only have to consider the subgraph $G_{\mathcal{N}_u \cup \mathcal{N}_v}$. Edge complementation can then be achieved by complementing the corresponding $\deg(u) \times \deg(v)$ submatrix, H_{uv}. Define $\mathrm{wt}(H)$ as the weight (number of non-zero entries) of H. In order for

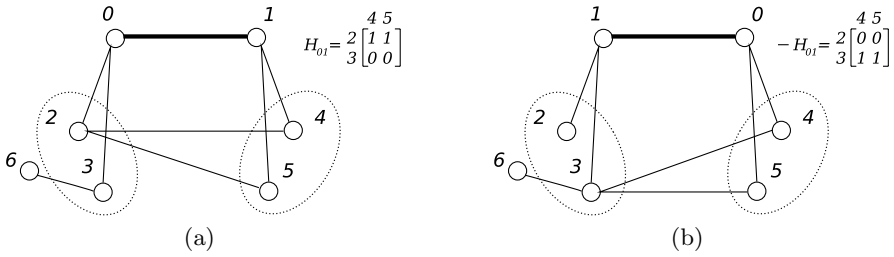


Fig. 3. Pivot on $(0, 1)$ is (A) edge-count preserving and (B) a local isomorphism, but not (C) a global isomorphism due to node 6. This node is not local to the Pivot edge.

$\mathrm{wt}(H_{uv}) = \mathrm{wt}(\overline{H_{uv}})$, at least one of the dimensions must be an even number, and $\mathrm{wt}(H_{uv})$ must equal $uv/2$. If these conditions are met, we define the Pivot operation as *edge-count preserving.*

B. More specifically, we define a *local isomorphism* as an operation which preserves the structure of subgraph $G_{\mathcal{N}_u \cup \mathcal{N}_v}$, without making any assumptions on the overall (global) structure of G. We then define the Pivot operation to be *local Iso-Pivot iff* H_{uv} can be recovered from $\overline{H_{uv}}$ by row/column permutations only. Fig. 3 shows a small example.
C. Finally, most specifically, we say that Pivot is a (global) *Iso-Pivot iff* H can be restored from H^1, using only row/column permutations, considering the entire matrix.

These requirements lead to the following observation,

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

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

2.2 Iso-Orbit

The definitions of Iso-Pivot are naturally extended to the case where a single Pivot can not by itself be an isomorphism. Consider, for instance, a girth-6 graph. Here, the local neighborhood (of any edge) must be empty, and, after a single Pivot, this neighborhood becomes a complete (bipartite) (sub)graph at distance 2 from the Pivot edge (all 4-cycles). This violates requirement A, and the resultant graph can not be isomorphic to the initial one–neither locally, nor globally.[2]

In the general case, Iso-Pivot is described as an ordered set of d edges on which Pivot must be applied to achieve an isomorphism. This is referred to as a d-iso sequence (or, a length-d iso sequence). The set of all isomorphisms of G (reachable via Iso-Pivot, for $d \geq 1$) is called the *Iso-Orbit* of G, which

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

corresponds to a subset of $\mathrm{Aut}(\mathcal{C})$. Pivot can be used, in a preprocessing stage, to recursively search for Iso-Pivot sequences. For each such relabelling of G, we keep the corresponding iso sequence leading to it. Since Pivot is reversible, identical isomorphisms may be found via sequences of different length, and involving different edges, where certain operations cancel each other out. As such, for each unique labelling, we keep only the minimum length sequence in the Iso-Orbit.

From a decoding perspective, row permutations of H give the same $\mathbf{TG}(H)$. By canonising the rows of H (in our case, sorting according to decimal value of the binary rows), we ensure that the Iso-Orbit contains only non-trivial isomorphisms of G. The complexity of this search is $\mathcal{O}(n|\mathrm{Aut}(\mathcal{C})|)$, where G has n nodes, so for strongly structured (and large) graphs it may be necessary to bound the recursion with a maximum depth, $d_{\max}$. This possibly partial Iso-Orbit is then simply referred to as the d-Iso-Orbit of G.

2.3 Local Iso Criterions

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

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

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

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

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

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

Proof. – Either $\mathcal{N}_u^v = \emptyset$, or $\mathcal{N}_v^u = \emptyset$. Let $\mathcal{N}_u^v = \emptyset$. Then Pivot on (u, v) has the effect of disconnecting v from $\mathcal{N}_v^u$, while connecting u to $\mathcal{N}_v^u$. The permutation that gives us the isomorphism is $\sigma = (u\ v)$. The same permutation applies when $\mathcal{N}_v^u = \emptyset$.

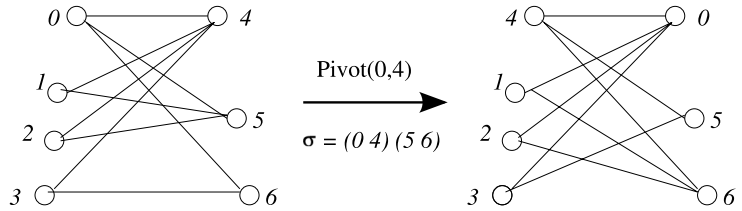


Fig. 4. Example of Lemma 1, where $\alpha = 0,\ \alpha' = 4,\ w_0 = 5,$ and $w'_0 = 6$. These graphs are isomorphic.

– For every $w_i \in \mathcal{N}_\alpha^{\alpha'}, \exists\ w'_i \in \mathcal{N}_\alpha^{\alpha'}$ such that $\mathcal{N}_{w_i} \ominus \mathcal{N}_{w'_i} = \mathcal{N}_{\alpha'}^{\alpha}$: The permutation that gives us the isomorphism is $\sigma = (u\ v) \prod (w_i\ w'_i)$. □

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

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

3 Structure of the (8, 4) Extended Hamming Code

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

The associated graph has one structure in its (non-iso) orbit; the cube of Fig. 2(c), meaning that its structure is so strong that any edge of any G^i satisfies Lemma 1 (is an Iso-Pivot). Starting from G as in Fig. 2(b), with parity-check matrix in Fig. 5(a), we find the Iso-Orbit of the graph. Grouped by length, $d = 1$ to 4, this orbit consists of 12, 30, 12, and 1 isomorphisms, respectively, all

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

(a) Standard form (b) Pivot (0, 4)

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

resembling a cube. Including the initial labelling, this sums up to 56 structurally distinct, non-trivial parity-check matrices for the code.[3] Necessarily, the 12 1-iso sequences correspond to the 12 edges of G (P-part of H).

4 Simulation Results

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

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

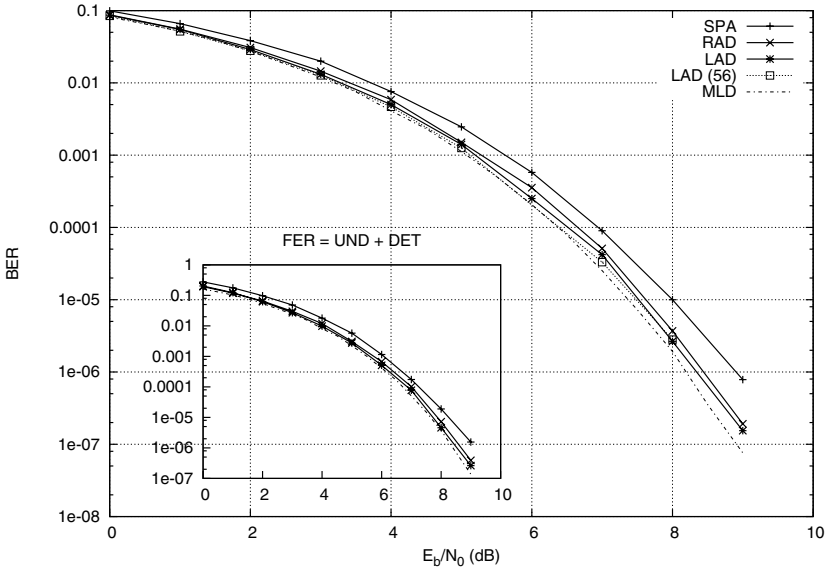
The *random adaptive decoder* (RAD) is a (flooding) SPA decoder, but which is designed to adapt (via random Iso-Pivot) to another G^i, with regular iteration interval, t. The decoder stops as soon as the syndrome check is satisfied (valid codeword, though not necessarily the one sent), or when T iterations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

The *list adaptive decoder* (LAD) is an extension of this idea, but where we apply Iso-Pivot operations from a precomputed list, $L \subseteq \text{Iso-Orbit}(G)$. In addition to the initial labelling, the range of this decoder is $D = |L| + 1$. A pool of T flooding iterations is allocated. Graph G^i, $0 \leq i < D$, is allocated $h_i = \lceil (T - I)/(D - i) \rceil$ iterations to come to a decoder decision, where I is the total number of iterations used by previous decoders G^j, $j < i$. Depending on T and L, h_i may go to 0, so an overall minimum, $h_{\min}$, should be set. This means that, although the list L may not be employed in its entirety, we ensure that the graphs used are doing useful work (more than 1 iteration). This

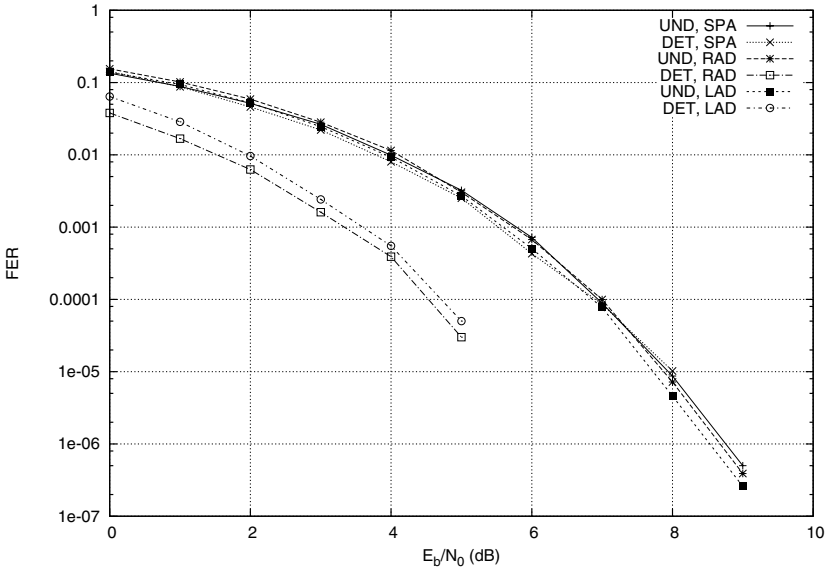
[3] Note that the full automorphism group of this code may be found by row permutations on these generators; $56 \cdot 4! = 1344 = |\mathrm{Aut}(\mathcal{C})|$.

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

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



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



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

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

minium should reflect parameters of the graph and code. Before applying the next Iso-Pivot from L, G^i compares its local decision to a running optimum kept in the decoder, and overwrites if a better decoder output is found (in squared Euclidean distance from $\mathbf{y}$). This comparison is devised to favor valid decoder states, in that distance measures of detected failures are only considered as long as no valid state has been found. The LAD does not stop on reaching a valid decoder state, but continues until "timeout" (T iterations). The final graph, G^δ, $\lfloor T/h_{\min} \rfloor \leq \delta \leq D-1$, outputs the optimum decision as the decoder result. In case no graph found a valid syndrome, the error state nearest to $\mathbf{y}$ (of the δ timeout states) was output. This is in an effort to reduce the bit-error contribution.

Fig. 6 benchmarks the performance of RAD/LAD against standard SPA and the optimal maximum likelihood decoder (MLD), in terms of bit-error rate (BER) and frame-error rate (FER), where an improvement is seen. The LAD plot is slightly nearer to the optimal MLD plot than the RAD, but the gain is not significant compared to the complexity tradeoff (Fig. 7). A detailed look at the (detected and undetected) frame errors, Fig. 6(b), reveals that adaptive decoders outperform SPA in terms of detected errors (timeout), where RAD shows the best gain. The RAD performs a random walk around the 56 sequences in the Iso-Orbit of G, while, for the LAD, we chose the subset of 12 1-iso sequences (defined by the 12 edges of the initial G) such that $h_{\min} = \lceil 100/13 \rceil = 7$.

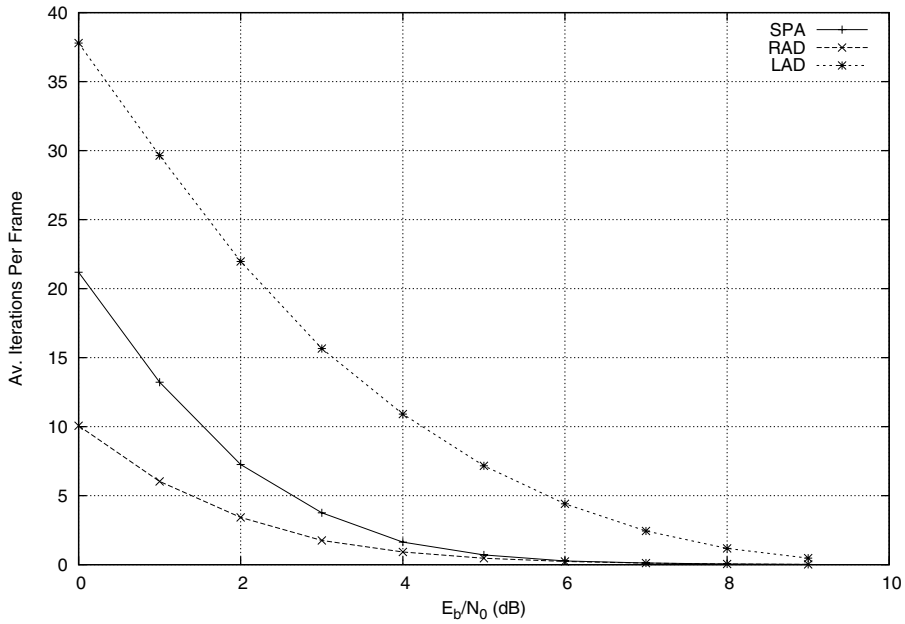


Fig. 7. The total number of iterations used (where timeout states contribute T iterations, and error-free frames contribute 0) averaged over total number of simulated frames

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

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

5 Conclusion and Future Work

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

As Iso-Pivot rotates sensitive substructures in $\mathbf{TG}(H)$, we expect a gain in selectively applying Iso-Pivot based on local convergence assessments (e.g., using entropy or reliability measures). We mention shifting short cycles away from unreliable bit nodes–as seen in the cube of Fig. 2. Pivoting adjacent to unreliable positions also causes these to become temporarily 'isolated' in terms of message-passing (weight-1 node), such that these are set in a 'listening state,' rather than confusing the adjacent nodes with its (presumed) unreliable APP [16,17,3]. In our scheme, we achieve this effect without the overhead of Gaussian elimination.

Local iso-criterions for $d = 2$ have been identified, and we are also working on further generalizations. This is interesting, as, due to the link between Pivot and 4-cycles, girth-6 graph isomorphisms can not be preserved with less than 2 Pivots. Our results on global isomorphisms indicate that it is not trivial to find graphs which exhibit a non-empty Iso-Orbit, which simultaneously are good codes (i.e., sparse and girth greater than 4). A reasonable next step is a more methodical search through all codes up to some length, yet we are also looking towards the use of local isomorphisms during decoding. For instance, when girth

is not preserved (see Section 2.1), cycle-splitting or cycle-reduction comes to mind–as in the way two 4-cycles can sometimes be split into one 6-cycle.

We are working on a generalized Pivot operation, which does not depend on the matrix (graph) being in standard form. With this tool, we expect to be able to compare our results with the realistically sized BCH code of [5]. We anticipate more significant results where larger Tanner graphs allow more true localization. Euclidean geometry LDPC codes [18] are also potential, sufficiently structured candidates for adaptive decoding.

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

Acknowledgments

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

References

1. Gallager, R.G.: Low-density parity-check codes. IRE Trans. Inform. Theory 8(1), 21–28 (1962)
2. MacKay, D.J.C., Neal, R.M.: Good codes based on very sparse matrices. In: Cryptography and Coding 5th IMA Conf., December 1995, pp. 100–111 (1995)
3. Jiang, J., Narayanan, K.R.: Iterative soft-input soft-output decoding of Reed-Solomon codes by adapting the parity-check matrix. IEEE Trans. Inform. Theory 52(8), 3746–3756 (2006)
4. Jiang, J., Narayanan, K.R.: Iterative soft decision decoding of Reed-Solomon codes. IEEE Commun. Lett. 8(4), 244–246 (2004)
5. Halford, T.R., Chugg, K.M.: Random redundant iterative soft-in soft-out decoding. IEEE Trans. on Commun. 56(4), 513–517 (2008)
6. Vontobel, P.O., Koetter, R.: Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes. IEEE Trans.Inform.Theory (2005) (submitted for publication)
7. Di, C., Proietti, D., Telatar, I.E., Richardson, T.J., Urbanke, R.L.: Finite-length analysis of low-density parity-check codes on the binary erasure channel. IEEE Trans. Inform. Theory 48(6), 1570–1579 (2002)
8. Richardson, T.: Error floors of LDPC codes. In: Proc.41st Annual Allerton Conf.on Commun., Control, and Computing, Monticello, IL, October 2003, pp. 1426–1435 (2003)
9. Richardson, T.J., Urbanke, R.: The capacity of low-density parity-check codes under message-passing decoding. IEEE Trans. Inform. Theory 47(2), 599–618 (2001)
10. Bouchet, A.: Isotropic systems. European Journal of Combinatorics 8, 231–244 (1987)
11. Danielsen, L.E., Parker, M.G.: Edge local complementation and equivalence of binary linear codes. Des. Codes Cryptogr. (to appear, 2008)

12. Riera, C., Parker, M.G.: On Pivot orbits of Boolean functions, optimal codes and related topics. In: Fourth International Workshop on Optimal Codes and Related Topics, Sofia, Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, June 2005, pp. 248–253 (2005)
13. Kschischang, F.R., Frey, B.J., Loeliger, H.A.: Factor graphs and the sum-product algorithm. IEEE. Trans. on Inform. Theory 47(2), 498–519 (2001)
14. Knudsen, J.G.: Randomised construction and dynamic decoding of LDPC codes. Master's thesis, University of Bergen (2006)
15. Danielsen, L.E., Parker, M.G.: On the classification of all self-dual additive codes over GF(4) of length up to 12. Journ. of Comb. Theory, Series A 113(7), 1351–1367 (2006)
16. Catherine, C.: Enhancing the error-correction performance of low-density parity-check codes. PhD thesis, University of Mauritius (2008)
17. Kothiyal, A., Takeshita, O.: A comparison of adaptive belief propagation and the best graph algorithm for the decoding of linear block codes. In: International Symposium on Information Theory, September 2005, pp. 724–728 (2005)
18. Kou, Y., Lin, S., Fossorier, M.P.C.: Low-density parity-check codes based on finite geometries: A rediscovery and new results. IEEE Trans. Inform. Theory 47(7), 2711–2736 (2001)