Improved Adaptive Belief Propagation Decoding Using Edge-Local Complementation

Joakim Grahl Knudsen, Constanza Riera*, Lars Eirik Danielsen, Matthew G. Parker, and Eirik Rosnes
Dept. of Informatics, University of Bergen, Thormøhlensgt. 55, 5008 Bergen, Norway
email: {joakimk, larsed, matthew, eirik}@ii.uib.no
*Bergen University College, Nygårdsgt. 112, 5008 Bergen, Norway, email: csr@hib.no

***Abstract*—This work is an extension of our previous work on an iterative soft decision decoder for high-density parity-check codes, using a graph-local operation known as edge-local complementation (ELC). The random application of ELC is replaced by ELC operations to target the inferred least reliable codeword positions, in between sum-product algorithm iterations. This is related to other decoding algorithms employing similar ideas, where our experience with graph-local operations allow us to improve the heuristics, and thus the performance of the decoder. This gain is both in terms of error-rate performance, and complexity in terms of a significant reduction in the required number of such operations. Simulations results are shown for two types of high-density parity-check codes (Reed-Solomon and quadratic residue codes), for which a gain is shown over our previous ELC-decoders, as well as iterative permutation decoding (SPA-PD) and adaptive belief-propagation (JN-ADP).**

I. Introduction

Iterative soft-input soft-output (SISO) decoding of graph-based codes has been shown to give near-optimum results, when the sum-product algorithm (SPA) is used on low-density parity-check codes. Recently, these results have been extended to high-density parity-check (HDPC) codes, in order to facilitate the use of well-known strong families of codes, such as Bose-Chaudhuri-Hocquenghem [1, 2], quadratic residue (QR) [3], and Reed-Solomon (RS) codes [4–6]. Constructions of these types have strong structural properties (most importantly, large automorphism group and minimum distance), as well as convenient algebraic descriptions to simplify hardware implementation. We have previously described a graph-local operation known as edge-local complementation (ELC), to improve the performance of SPA decoding by providing diversity during decoding. In addition to random application of a small number of ELC operations [3], we have considered the controlled application of ELC such as to preserve graph isomorphism [7], or to maintain a bound on graph density [8]. This work is an extension on our work on ELC-based SISO HDPC decoding, where the aim is now to affect inferred error positions during decoding. By using the structural effects of ELC on the Tanner graph, these positions are arguably set in a 'listening state,' such that they may continue to converge, but without influencing other nodes (e.g., via cycles). This approach is similar to iterative permutation decoding (SPA-PD), which uses permutations from the automorphism group of the code in an attempt to confine errors to a parity-set of the code, such that these may be corrected by simply re-encoding the corresponding information set [1]. The adaptive belief propagation (JN-ADP) decoder [4] uses Gaussian elimination (GE) on the $(n-k) \times n$ parity-check matrix H, to reduce the columns corresponding to the $n-k$ least reliable positions to an identity matrix. We will show how the ELC operation is related to GE. The novelty of the proposed SPA-ADP-ELC decoder lies in the significant reduction in the number of positions (columns of H) affected by the 'ADP-stage,' whether it is implemented using GE or ELC, while simultaneously achieving a gain in error-rate performance over SPA-PD and JN-ADP. We also maintain the locality argument of the ELC operation, which leads to several modifications to improve the performance of the ADP-ELC heuristic (i.e., how to choose the least reliable positions).

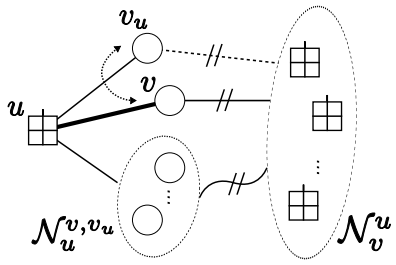


Fig. 1. ELC on edge (u, v), as implemented on a (systematic) Tanner graph, where v_u is the systematic node for node u. Straight links between two sets mean that these are completely connected, while curved links mean arbitrary connections. Dashed lines indicate non-edges. Doubly slashed links are complemented (edges are replaced by non-edges, and vice versa) resulting in v and v_u swapping connections. This graph may be a subgraph of a larger graph.

First, some notation and definitions. We will use boldface notation for vectors, and italics uppercase for matrices. A binary linear code $\mathcal{C}$ of length n, dimension k, and minimum distance $d_{\min}$ is denoted by $[n, k, d_{\min}]$. The $(n-k) \times n$ parity-check matrix is denoted by H, and is said to be systematic if its columns can be reordered into the form $[I\, P]$, where I is the identity matrix of size $n-k$. The corresponding codeword positions comprise a parity set $\mathcal{I}$, and an information set $\mathcal{P}$, respectively, referring to the $k \times n$ generator matrix $[P^T\, I]$. The single non-zero entry of a systematic column is referred to as a *pivotal*. Unless stated otherwise, codes discussed in this paper are represented by H in systematic form. The local neighborhood of a node v is the set of nodes adjacent to v, and is denoted by $\mathcal{N}_v$, while $\mathcal{N}_v^u$ is shorthand for $\mathcal{N}_v \setminus \{u\}$. We will refer to variable node v_i as simply v when the index is obvious from the context.

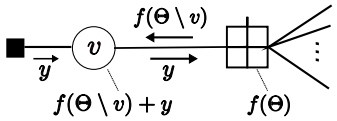


Fig. 2. The SPA update of a degree-2 variable node, v.

II. ELC on a Tanner graph

ELC is defined on a simple graph, $G = \left(\begin{smallmatrix} 0 & P \\ P^T & 0 \end{smallmatrix}\right)$, corresponding to a code in systematic form $H = [I \ P]$, or, equivalently, a Tanner graph $\mathbf{TG}(H) = \left(\begin{smallmatrix} 0 & H \\ H^T & 0 \end{smallmatrix}\right)$. We have previously described ELC on $\mathbf{TG}(H)$ by going via G [3]. We will now discuss the implementation of ELC as directly applied to $\mathbf{TG}(H)$. ELC requires that H is systematic, and since it is natural to assume there are no repeated columns, each row u has a unique pivotal. Let this single adjacent node to u be denoted by $v_u \in \mathcal{I}$. The implementation of ELC on $\mathbf{TG}(H)$ is to complement the edges connecting $\mathcal{N}_u$ and $\mathcal{N}_v^u$, as shown in Fig. 1. By definition, v is adjacent to all nodes in $\mathcal{N}_v$, whereas the systematic node $v_u \in \mathcal{N}_u$ is not connected to $\mathcal{N}_v^u$ at all. Thus, the complementation entails the effect of swapping the connections of these nodes, or, equally, the corresponding two columns in H. Since $v_u \in \mathcal{I}$ and $v \in \mathcal{P}$, this is a swap between $\mathcal{I}$ and $\mathcal{P}$. Note that ELC on (u, v_u) has no effect, as $\mathcal{N}_{v_u}^u = \emptyset$. A maximum of $n-k$ ELC 'independent' ELC operations may be applied to $\mathbf{TG}(H)$–one per row.

It is readily verified that ELC is a graph implementation of the row-additions performed to reduce a column to systematic form during GE. As such, we may define GE (on a systematic matrix) as $n-k$ ELC operations. For completeness, we mention that the above described operation applied to a non-systematic $\mathbf{TG}(H)$ will reduce the corresponding column to systematic form. For reference, we may define this as non-systematic ELC.

III. Adaptive Belief Propagation

The idea of incorporating the inferring and moving of errors into solvable positions as part of a decoding algorithm, was suggested by MacWilliams and Sloane for their algebraic permutation decoder (PD) [9]. PD has recently been extended to an iterative algorithm [1, 7], SPA-PD, so as to further benefit from SISO decoding. In a similar fashion, JN-ADP attempts to produce an identity submatrix in the columns corresponding to error positions, by means of GE on H [4]. The GE operation affects individual columns independently, and can be targeted directly to the desired $n-k$ positions, whereas a permutation generally affects all n positions of the code.

Over an additive white Gaussian noise (AWGN) channel, the error positions are obviously unknown to the receiver. However, simple heuristics exist to infer the reliability of the *a posteriori* probability (APP), or state value, at a position. Using binary phase-shift keying (mapping $0 \to +1, 1 \to -1$), the received noisy vector is $y_i = (-1)^{x_i} + e_i$, $0 \le i < n$, where $\mathbf{x}$ is a codeword and $\mathbf{e}$ is AWGN. In the log-likelihood ratio (LLR) domain, the magnitude $|L_j^v|$ serves as a measure on the reliability of the APP at position v, in iteration j. Note that $L_0^v \triangleq \frac{2}{\eta^2} y$, where η is the standard deviation of the AWGN. The JN-ADP algorithm begins by sorting the n codeword positions according to increasing magnitude. Let the corresponding permutation be σ, such that $\sigma_j = i$, $0 \le j, i < n$, if v_i is the j'th least reliable position. Let $\mathbf{K} = \sigma(\mathbf{L})$ be the sorted sequence of APPs, and define a counter, δ. Then, for each row $0 \le j - \delta < n - k$ of H, a pivotal is attempted in column h_{K_j} (position K_j) at row index $j - \delta$. The placement of a pivotal in a column is not important for SPA decoding, yet, an important aspect of JN-ADP is to avoid cancellations, where several pivotals affect the same row. Due to the nature of GE, a pivotal is unique not only to its column, but also to its row, in that any repeated application (within the same GE-stage) on the same row will replace the previous pivotal in that row. As $\mathbf{K}$ is processed in order of improving reliabilities, any such cancellation must have a counter-productive effect on performance (this is easily verified by simulations). To avoid this, JN-ADP processes the rows in an ordered fashion. If h_{K_j} is zero in row entry $j - \delta$, then let $j' > j - \delta$ be the first non-zero row entry in h_{K_j}. Row j' is then added onto row $j - \delta$, such that a pivotal may be made here. As such, the GE-stage does work even when position K_j is already in $\mathcal{I}$. Only when the column is zero in all coordinates $j' \ge j - \delta$ will JN-ADP skip to the next position, K_{j+1}, and increase δ by 1. The GE-stage ends when $j - \delta = n - k$, or, failing that, when $\mathbf{K}$ is exhausted. As described in [5], the aim of JN-ADP is to affect as many unreliable positions as possible in each GE-stage. Usually, δ is small, such that one GE-stage does $p \approx n - k$ pivotals to produce a new $\mathbf{TG}(H)$.

A. Isolating Weak Positions

The presence of weight-1 columns in H has a significant impact on the flow of messages in SPA decoding. As illustrated in Fig. 2, the Tanner graph equivalent of a weight-1 column in H is a variable node of degree 1, not counting the Forney-style input half-edge, y. This node is minimally connected to $\mathbf{TG}(H)$, and is not part of any cycles. The SPA-rules must adhere to an extrinsic principle, in which the message passed out on any edge is independent of the incoming message along that same edge. This message is the marginal for the variable encoded by the edge. The SPA rule for a check node is the parity (XOR) function of its incoming messages, which we denote by $f(\Theta)$. Thus, the marginal on v is $f(\Theta \setminus v)$, providing v with updated information from the rest of the graph. For a variable node in the LLR domain, the SPA rule is simple summation. When v is systematic, the APP is $L^v = y + f(\Theta \setminus v)$, and the node can only relay its input message, $L^v - f(\Theta \setminus v) = y$, to the single adjacent check node. This variable node may be said to be in a listening or passive state. If such a position is unreliable, it will not disturb the rest of the graph, while receiving information to help it converge.

Still, the gain of JN-ADP can not be accredited to this isolation effect alone. Specifically, it is known that modify-

Example 1 (ADP-ELC-stage)**.** *Consider the extended Hamming* $[8, 4, 4]$ *code,* H*, and some decoder state (vector of APPs),* $\mathbf{L}$*. The actual values are not important for an example, so we focus directly on a permutation,* σ*. The bipartition is indicated (in* σ*) by underlining the indices of positions in* P*. The current position (column) to consider for ELC is indicated by a star symbol over* H*. This ELC-stage ends after two ELC operations.*

$$H = \begin{array}{c} \overset{\star}{\begin{matrix} 0&1&2&3&4&5&6&7 \end{matrix}} \\ \begin{bmatrix} \mathbf{1}&0&0&0&1&\mathbf{1}&1&0 \\ 0&\mathbf{1}&0&0&1&\mathbf{1}&0&1 \\ 0&0&1&0&1&0&1&1 \\ 0&0&0&\mathbf{1}&0&\mathbf{1}&1&1 \end{bmatrix} \end{array} \quad \sigma = (\underline{5}\,1\,\underline{4}\,3\,0\,\underline{7}\,2\,\underline{6}) \quad u^\star = u_0 \quad \xrightarrow[ELC(u_0, v_5)]{} \quad H = \begin{array}{c} \overset{\star}{\begin{matrix} 0&1&2&3&4&5&6&7 \end{matrix}} \\ \begin{bmatrix} 1&0&0&0&\mathbf{1}&\mathbf{1}&1&0 \\ 1&1&0&0&0&0&1&1 \\ 0&0&\mathbf{1}&0&\mathbf{1}&0&1&1 \\ 1&0&0&\mathbf{1}&\mathbf{1}&0&0&1 \end{bmatrix} \end{array} \quad \sigma = (5\,1\,\underline{4}\,3\,\underline{0}\,\underline{7}\,2\,\underline{6}) \quad u^\star = u_2 \quad \xrightarrow[ELC(u_2, v_4)]{} \quad H = \begin{array}{c} \overset{\star}{\begin{matrix} 0&1&2&3&4&5&6&7 \end{matrix}} \\ \begin{bmatrix} \mathbf{1}&0&1&0&0&\mathbf{1}&0&1 \\ \mathbf{1}&\mathbf{1}&0&0&0&0&1&1 \\ 0&0&1&0&1&0&1&1 \\ \mathbf{1}&0&1&\mathbf{1}&0&0&1&0 \end{bmatrix} \end{array} \quad \sigma = (5\,1\,4\,3\,\underline{0}\,\underline{7}\,\underline{2}\,\underline{6}) \quad u^\star = \emptyset.$$

ing the structure of $\mathbf{TG}(H)$ during SPA decoding may, in itself, improve performance [2, 3, 5, 10, 11]. Such diversity will change the structure of cycles, change the node-degree distributions (as with irregular low-density parity-check codes), and, generally, alter the flow of messages during SPA decoding. After the GE-stage, the new $\mathbf{TG}(H)$ is initialized for the next SPA iteration, by damping each variable node. Damping involves scaling down the extrinsic contribution to the APP by a damping coefficient, $0 < \alpha < 1$. This is then accumulated on the input, to initialize the next iteration with

$$L^v_{j+1} := L^v_j + \alpha \Gamma^v_j, \tag{1}$$

where $\Gamma^v_j = \Sigma_{u \in \mathcal{N}_v} \mu^{v \leftarrow u}_j$, the sum of messages incoming to v in iteration j, and $\Gamma^v_0 \triangleq L^v_0$. The (global) damping stage, (GD) applies (1) to all variable nodes in $\mathbf{TG}(H)$. This slightly complicates the argument of isolating a variable node, as the input to v is no longer fixed to the channel value y. As JN-ADP uses a constant damping coefficient, the accumulation may be expressed as,

$$L^v_{j+1} = y + \alpha \Sigma^j_{j'=1} (\Gamma^v_{j'} - \Gamma^v_{j'-1}).$$

Thus, damping never affects the channel value, and we see that the accumulation is negligible when α is small, such that the isolation argument may still hold.

IV. SPA-ADP-ELC

Let us initially describe the proposed SPA-ADP-ELC algorithm simply as SPA iterations interspersed with an ADP-ELC-stage acting on a number $0 < p \leq n - k$ of the least reliable positions. From the graph-local ELC perspective, however, certain distinctions from the JN-ADP algorithm naturally occur. Let us first point out that, although the process of sorting the APPs is indeed a global operation, it is possible to approach the problem from a graph-local perspective.

The first distinction is that SPA-ELC decoding has been shown to be effective for $p \ll n - k$, meaning a reduction in complexity – the major concern with JN-ADP. Some further distinctions arise from the description of ELC. ELC requires that H is in systematic form, and we observe that the GE-stage of JN-ADP will immediately reduce any H to systematic form. For a systematic node $v_u \in \mathcal{I}$, $\mathcal{N}^u_v = \emptyset$, such that ELC on (u, v_u) has no effect. While processing $\mathbf{K}$, we may immediately skip any position already in $\mathcal{I}$. The cancellation issue is handled by flagging the check node, u, as 'used,' such that it will not be considered again in this ADP-ELC-stage.

We have previously described how the technique of damping may be simplified to affect only the edges affected by ELC [3]. If an operation acts locally on only certain nodes in $\mathbf{TG}(H)$, we may restrict (1) to these nodes. For example, ELC on (u, v) entails damping of variable node v, which also serves to place outgoing messages onto the new edges inserted by ELC. This, by itself, has a beneficial effect on performance, and we refer to this as local damping (LD). The following sections discuss some further benefits of a local ADP-ELC-stage.

A. Improved Heuristic to Select ELC Positions

Consider the implicit 'swap effect' involved in ELC on $\mathbf{TG}(H)$. Extending the argument made for JN-ADP – namely, to move (isolate) the least reliable positions into $\mathcal{P}$ (I-part of H) – it is equally reasonable to ensure also the converse; that the positions moved into $\mathcal{I}$ are the most reliable positions. The procedure is simple and graph-local. Given a position, v, rather than only choosing arbitrarily among the unused adjacent check nodes $u \in \mathcal{N}_v$, the ADP-ELC algorithm chooses the check node $u^\star \in \mathcal{N}_v$ for which the induced swap is with the most reliable systematic position adjacent to $\mathcal{N}_v$,

$$u^\star = \underset{u \in \mathcal{N}_v,\ |v_u| > |v|}{\operatorname{argmax}} |v_u|. \tag{2}$$

In the event where $u^\star = \emptyset$, no ELC is possible for this position, and ADP-ELC moves on to the next-worst position. For completeness, we mention that in the event that the best edge is not unique, the heuristic will simply select and return the first instance encountered. However, as this is unlikely to occur for real-valued APPs, we will simply view these edges as equally good choices, and not attempt to refine the response to this situation. The ADP-ELC(p) algorithm processes the p first information positions in $\mathbf{K}$, applying (2) to determine the ELC locations. No additional measures are needed to avoid the cancellation problem. The algorithm simply works from both ends of $\mathbf{K}$, pairing the leftmost (weakest) information position with the rightmost (strongest) parity position. The resulting ELC swaps the corresponding positions across the bipartition, and requires that we keep track of the bipartition ($\mathcal{I}$ and $\mathcal{P}$). This simple approach avoids the cancellation issue, in that the systematic positions corresponding to the 'used' rows, become non-systematic – but, with large magnitude – due to the swap.

Lemma 1. *Although variable nodes share the same check nodes, the proposed heuristic will never repeatedly choose the same check node* $u^\star$ *within an ADP-ELC-stage.*

Proof: Consider two positions, v and w, which are both adjacent to the same check node, u. Without loss of generality, say $|v| < |w|$, so we first consider v. Then, (2) gives $u^\star = u$, and we perform ELC on (u, v) such that v_u becomes non-systematic, and the systematic node of u is now v. When we later consider w, choosing the same $u^\star = u$ would entail a

swap of w with v, thus cancelling the previous swap. However, this choice of $u^\star$ is not possible, because $|v| < |w|$. ∎

Example 1 shows a case, where $v = v_5$ and $w = v_4$, both adjacent to $u = u_0$ (row 0). When considering v_4, we can not choose again $u^\star = u_0$, since $v_u = v_5$ and $|v_5| < |v_4|$.

As a final issue, consider the situation when we reach a position $v_{u\star}$, which has already been involved in a swap with some other position v. This means that $v_{u\star}$ has been moved from $\mathcal{P}$ to $\mathcal{I}$ (I to P-part of H) within this ADP-ELC-stage. The proposed scheme will never do a second ELC on an edge adjacent to this node, which would cancel the previous swap.

Lemma 2. *A position swapped from $\mathcal{P}$ into $\mathcal{I}$ will not be subject to any subsequent ELC, within the same ADP-ELC-stage.*

Proof: Say the preceding ELC was on (u, v), which swapped v_u into $\mathcal{I}$. If the heuristic later reaches this same $w = v_u$ (before exhausting p), no subsequent ELC is possible for this position. In the initial ELC, v_u was the most reliable systematic position adjacent to any $u \in \mathcal{N}_v$, such that $|v_u| > |v|$. However, this now means that $|w| > |v_{u'}|,\ \forall u' \in \mathcal{N}_w$, since $w = v_u$. ∎

Consequently, (2) yields $\emptyset$, and ADP-ELC proceeds to consider the next position in $\mathbf{K}$. Again, Example 1 shows a case, in the last stage, when $v = v_0$. We have now covered all the possible cancellation issues, without any extra bookkeeping or complexity in the ADP-ELC heuristic (apart from keeping track of the bipartition).

B. Local-Neighborhood Damping

As an increasing number of ELC operations is applied to $\mathbf{TG}(H)$, the overall modification to the graph can no longer be considered to be local. In this situation, we have observed by experiment that local damping does not give the best results. In ELC on (u, v), local damping may be extended to all variable nodes $v \in \mathcal{N}_u$, without violating the locality restriction. This is referred to as local-neighborhood damping (ND).

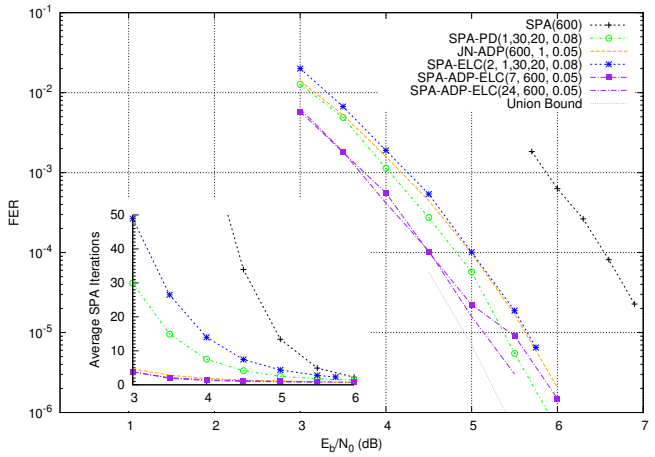
V. Results

The frame error-rate (FER) performance of SPA-ADP-ELC is simulated on the $[31, 25, 8]$ RS code over $\mathrm{GF}(2^5)$ (we use a binary $[155, 125]$ image), and the $[48, 24, 12]$ extended QR code. The algorithms are implemented using our generalized SISO HDPC decoder, Algorithm 1. SPA-ADP-ELC(p, T, α) = SISO-HDPC$(1, 1, T, 1, \alpha, \mathrm{ADP\text{–}ELC}(p), \mathrm{DR})$, where T is the maximum number of SPA iterations and the damping rule is either LD or ND. JN-ADP$(T, 1, \alpha)$ = SISO-HDPC$(1, 1, T, 1, \alpha, \mathrm{GE}, \mathrm{GD})$, where the GE-stage may be implemented by $n - k$ ELC operations. Since $I_2 = T$, and $I_1 = I_3 = 1$, these decoders use constant damping. For JN-ADP, we use "Variation A" (avoid degree-1 columns), whereas we do not use "Variation B" (list decoding) [5]. SPA-PD$(I_1, I_2, I_3, \alpha_0)$ = SISO-HDPC$(1, I_1, I_2, I_3, \alpha_0, \mathrm{PD}, \mathrm{GD})$ and SPA-ELC$(p, I_1, I_2, I_3, \alpha_0)$ = SISO-HDPC$(p, I_1, I_2, I_3, \alpha_0, \mathrm{ELC}, \mathrm{LD})$ update α from α_0 to 1. It is interesting to note that the SPA-ADP-ELC decoder outperforms SPA-PD on the QR code, which has a very large automorphism group [3]. For SPA(T) and SPA-PD we use a non-systematic matrix optimized on weight. For both codes, Figs. 3(a) and 4(a) show that the SPA-ADP-ELC decoder also has a gain over the successful JN-ADP algorithm, even when JN-ADP uses "Variation C," a symbol-level hard-decision list-decoding (HDD) stage for the RS code. This gain is attributed mainly to our improved ADP-ELC heuristic, but we also observe a gain due to the modified damping. We may also benefit from an HDD stage, mainly to reduce an error-floor effect. For both codes, the performance is quite near the union bound. We also observe a reduction in the average number of SPA messages computed per frame, due to a reduction in number of SPA iterations. The HDD stage is implemented using a "genie-aided stopping criterion," rather than a list-decoder [5]. As such, the complexity of these two-stage can not be counted in terms of SPA iterations as implemented.

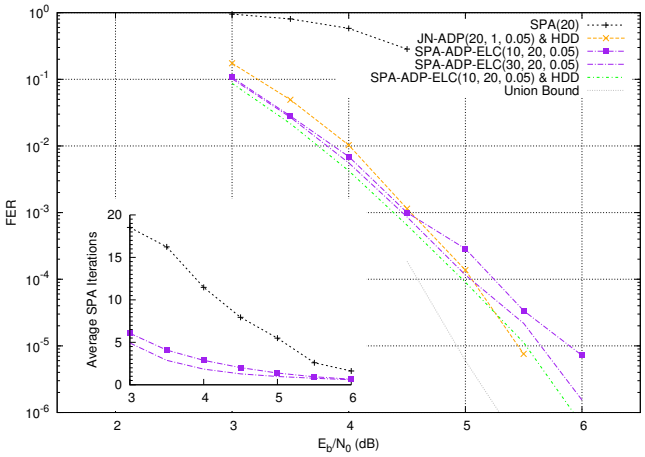
We now focus on the reduction in complexity, in terms of ELC operations. As we increase p from 1 to $n-k$, we simulate signal-to-noise ratio (SNR) range 3.5 to 5.0 dB. Figs. 3(b) and 4(b) show the results for SPA-ADP-ELC, for which the FER improves with increasing p, until an optimal value, $p^\star$, is reached. As postulated initially, a competetive FER is achieved for $p^\star \ll n-k$, for a significant reduction in ELC complexity. JN-ADP performance for the same SNR is indicated by the horizontal lines. As we have discussed, an ADP-ELC (or GE) stage is sometimes unable to perform an ELC operation for a given unreliable position. The GE-stage of JN-ADP always does work equivalent to $n - k$ ELC operations. Define a function, $\bar{p}(p)$, to give the average number of ELC operations performed per ADP-ELC-stage, for a given p. Figs. 3(c) and 4(c) compare $\bar{p}(p)$ against the linear ELC-complexity of JN-ADP. As stated also in [6], a reduction can be achieved by simply improving the implementation of GE to do no work whenever an unreliable position is already in $\mathcal{P}$. The plots

Algorithm 1 SISO-HDPC$(p, I_1, I_2, I_3, \alpha_0, \mathrm{OP}, \mathrm{DR})$. Stages **A** and **C** may only apply when implementing the JN-ADP or SPA-ADP-ELC algorithm

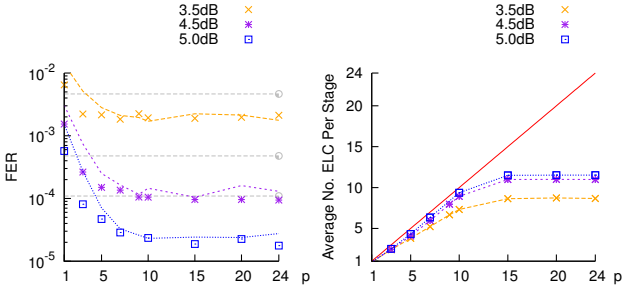
```
α = α_0
for I_3 times do
    Restart decoder from channel vector
    for I_2 times do
        (C) HDD stage. RS-code only.
        Stop if syndrome check is satisfied
        Apply damping rule, DR, with coefficient α
        Apply at random p operations, OP
        (A) Random row additions (no degree-1 columns)
        for I_1 times do
            Apply SPA iteration ('flooding' scheduling)
        end for
    end for
    Increment α towards 1
end for
```



(a) FER performance, and average number of SPA iterations per frame.

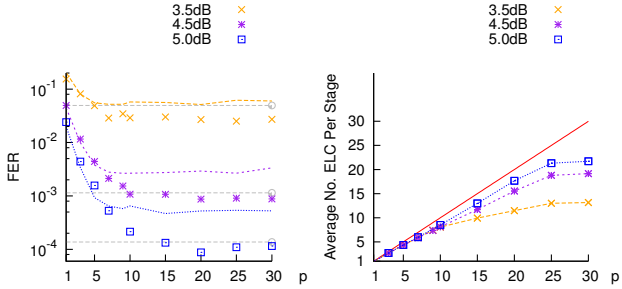


(a) FER performance, and average number of SPA iterations per frame.



(b) Optimal value, $p^\star = 7$ (c) ELC usage, $\bar{p}(p)$, of ADP-ELC

Fig. 3. For the $[48, 24, 12]$ extended QR code, the best performance is using local damping. The performance using ND damping is indicated by the lines in Fig. 3(b). Maximum $T = 600$ iterations.



(b) Optimal value, $p^\star = 10$ (c) ELC usage, $\bar{p}(p)$, of ADP-ELC

Fig. 4. For the binary image of the $[31, 25, 7]$ Reed-Solomon code over $\mathrm{GF}(2^5)$, the gain is greater using ND damping. The performance using LD damping is indicated by the lines in Fig. 4(b). Maximum $T = 20$ iterations.

verify that JN-ADP performs $n - k - \bar{p}(n-k)$ redundant ELC operations. Even comparing against this improved GE implementation in [6], the figures show that $\bar{p}(p)$ grows linearly before flattening out at high p. By using $p^\star$, we may then get a further reduction in average number of ELC operations.

VI. Conclusion and Future Work

We have described an SPA-ADP-ELC decoder to target the least reliable positions during decoding. An improvement is shown over related algorithms, both in terms of error-rate performance and complexity, which is ascribed to an improved heuristic to apply the ELC operations. Most importantly, the amount of ELC operations can be reduced significantly, compared to a full GE-stage. Future work is concerned with further extensions of the ADP-ELC heuristic, and using a non-systematic variant of ELC to avoid degree-1 nodes. We are also working on a distributed implementation of the sorting stage, where each check node sorts its adjacent variable nodes.

References

[1] T. R. Halford and K. M. Chugg, "Random redundant iterative soft-in soft-out decoding," *IEEE Trans. Commun.*, vol. 56, no. 4, pp. 513–517, Apr. 2008.

[2] I. Dimnik and Y. Be'ery, "Improved random redundant iterative HDPC decoding," *IEEE Trans. Commun.*, vol. 57, no. 7, pp. 1982–1985, Jul. 2009.

[3] J. G. Knudsen, C. Riera, L. E. Danielsen, M. G. Parker, and E. Rosnes, "Random edge local complementation and iterative soft-decision decoding," in *Proc. IEEE Int. Symp. Inform. Theory*, Seoul, Korea, Jul. 2009, pp. 899–903.

[4] J. Jiang and K. R. Narayanan, "Iterative soft decision decoding of Reed-Solomon codes," *IEEE Commun. Lett.*, vol. 8, no. 4, pp. 244–246, Apr. 2004.

[5] ——, "Iterative soft-input soft-output decoding of Reed-Solomon codes by adapting the parity-check matrix," *IEEE Trans. Inform. Theory*, vol. 52, no. 8, pp. 3746–3756, Aug. 2006.

[6] M. El-Khamy and R. J. McEliece, "Iterative algebraic soft-decision list decoding of Reed-Solomon codes," *IEEE J. sel. Areas Commun.*, vol. 24, no. 3, pp. 481–490, 2006.

[7] J. G. Knudsen, C. Riera, M. G. Parker, and E. Rosnes, "Adaptive soft-decision decoding using edge local complementation," in *Proc. Second Int. Castle Meeting on Coding Theory and Applications (2ICMCTA), LNCS 5228*, Castillo de la Mota, Medina del Campo, Spain, Sep. 2008, pp. 82–94.

[8] J. G. Knudsen, C. Riera, L. E. Danielsen, M. G. Parker, and E. Rosnes, "On iterative decoding of HDPC codes using weight-bounding graph operations," in *Proc. Int. Zürich Seminar on Commun.*, Zürich, Switzerland, Mar. 2010.

[9] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. North Holland, 1977, ch. 16.

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

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