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). Inferred least reliable codeword positions are targeted by an ELC stage in between sum-product algorithm iterations. A gain is shown over related iterative decoding algorithms–mainly due to an improved heuristic to determine optimum ELC edges in the Tanner graph–both in error-rate performance, as well as complexity in terms of a significant reduction in the required number of ELC operations. We also present a novel damping operation, generalized to the graph-local setting where extrinsic information remains on edges not affected by ELC.**

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, non-trivial automorphism group and large minimum distance), as well as convenient algebraic descriptions to simplify hardware implementation. We have previously described a graph-local operation known as edge-local complementation (ELC) and its applications to improve the performance of SPA decoding by providing diversity during decoding. In addition to random application of a small number of ELC operations [3], we have considered the controlled application of ELC such as to preserve graph isomorphism, or to maintain a bound on graph weight [7]. We have previously compared against iterative permutation decoding (SPA-PD), in which permutations from the automorphism group of the code are used to gain diversity [1]. This work is an extension of our work on ELC-based SISO HDPC decoding, where the aim is now to target inferred error positions during decoding. In addition to improved diversity, the structural effects of ELC on the Tanner graph are such that affected positions are set in a listening state (degree-1 nodes), such that they may continue to converge without influencing other nodes (e.g., via cycles). The adaptive belief propagation decoder from [5], which we denote by ABP, uses Gaussian elimination (GE) on the $(n-k) \times n$ parity-check matrix H, in an attempt to reduce the columns corresponding to the $n-k$ least reliable parity positions to an identity matrix. We show how the ELC operation is related to GE. The novelty of the proposed ABP-ELC decoder lies in the significant reduction in the number of positions (columns of H) affected by the adaptive stage, while simultaneously achieving a gain in error-rate performance over SPA-PD and ABP. We also maintain the locality argument of the ELC operation, which leads to several modifications to improve the performance of the ABP-ELC heuristic (i.e., how to choose the least reliable positions).

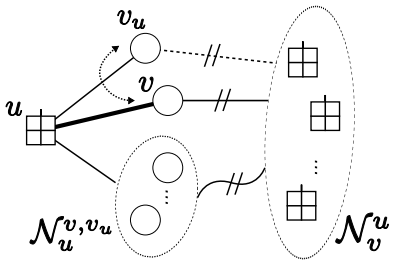


Fig. 1. ELC on edge (u, v) of a (systematic) Tanner graph, where v_u is the systematic node for node u. Straight links between two sets mean that these are completely connected, while curved links mean arbitrary connections. Dashed lines indicate non-edges. Doubly slashed links are complemented (edges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

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

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 parity-check matrix $H = [I\, P]$, or, equivalently,

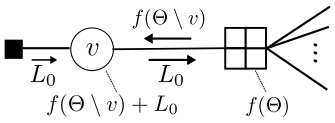


Fig. 2. The SPA update of a systematic variable node v.

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 now discuss the implementation of ELC as directly applied to $\mathbf{TG}(H)$. ELC requires that H is systematic, and since it is natural to assume there are no repeated columns, each row u has a unique pivotal. Let this single node adjacent to u be denoted by $v_u \in \mathcal{P}$. The implementation of ELC on an edge $(u, v) \in \mathbf{TG}(H)$ is to complement the edges connecting $\mathcal{N}_u$ and $\mathcal{N}_v^u$, as shown in Fig. 1. By definition, v is adjacent to all nodes in $\mathcal{N}_v$, whereas the systematic node $v_u \in \mathcal{N}_u$ is not connected to $\mathcal{N}_v^u$. Thus, the complementation entails the effect of swapping the connections of these nodes, or, equally, the corresponding two columns in H. Since $v_u \in \mathcal{P}$ and $v \in \mathcal{I}$, positions v and v_u are swapped between $\mathcal{I}$ and $\mathcal{P}$, changing the bipartition. Note that ELC on (u, v_u) has no effect, as $\mathcal{N}_{v_u}^u = \emptyset$. A maximum of $n - k$ *independent* ELC operations may be applied to $\mathbf{TG}(H)$, one per row (see *cancellation issues* in Section III). It is readily verified that ELC is a graph implementation of the row-additions performed to reduce a column to systematic form during GE. As such, we may define GE (on a systematic matrix) as a sequence of $n - k$ ELC operations. In fact, it has been shown that all information sets of the code are found via ELC [8].

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], SPA-PD, so as to further benefit from SISO decoding. In a similar fashion, ABP attempts to produce an identity submatrix in the columns corresponding to error positions, by means of GE on H [5]. The GE operation affects individual columns independently, and can be targeted directly to the desired $n-k$ positions, whereas a permutation generally affects all n positions of the codeword.

Over an additive white Gaussian noise (AWGN) channel, the error positions are obviously unknown to the receiver. However, simple heuristics exist to infer the reliability at a codeword position. The received noisy vector is $\mathbf{y} = (-1)^{\mathbf{x}} + \mathbf{e}$, where $\mathbf{x}$ is a codeword and $\mathbf{e}$ is AWGN. In the log-likelihood ratio (LLR) domain, the initial LLR at position v is $L_0^v \triangleq \frac{2}{\eta^2} y_v$, where η is the standard deviation of the AWGN. The magnitude $|L_j^v|$ of the LLR L_j^v in iteration j (see (1)) serves as a measure on the reliability of position v. The ABP algorithm begins by sorting the n codeword positions according to increasing reliability (we assume an unique ordering always exists, since the output alphabet of the AWGN channel has infinite support). Let the corresponding permutation be σ, such that $\sigma_j = i$, $0 \leq j, i < n$, if i is the jth least reliable position, and initialize a counter $\delta = 0$. Then, for each row $0 \leq j - \delta < n - k$ of H, a pivotal is attempted made in column h_{σ_j} at row index $j - \delta$. The (row) coordinate of a pivotal in a column is not important for SPA decoding, yet an important aspect of ABP is to avoid cancellations where a second pivotal is made in the same row, thus replacing the initial pivotal. As σ is processed from left to right, any such cancellation must have a counter-productive effect on performance (this is easily verified by simulations). To avoid this, ABP processes the rows in an ordered fashion. If h_{σ_j} is zero in row entry $j - \delta$, then let $j' > j - \delta$ be the first non-zero row entry (if any) in h_{σ_j}. Row j' is then added onto row $j - \delta$, such that a pivotal may now be made here. As such, the GE stage performs redundant work when position σ_j is already in $\mathcal{P}$. Only when h_{σ_j} is zero in all coordinates $j' \geq j - \delta$ will ABP skip to the next position, σ_{j+1}, and increase δ by 1. The GE stage ends after $j - \delta = n - k$ pivotals have been made, which is always possible since H is of full rank.

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 is a variable node of degree 1, not counting the Forney-style input half-edge, L_0. This node is minimally connected to $\mathbf{TG}(H)$, and is not part of any cycles. The SPA update rules adhere to an extrinsic principle, in which the message passed out on any edge is independent of the incoming message along that same edge. One iteration of the flooding schedule consists of the execution of an SPA rule for all variable nodes, followed by executing all check nodes. The SPA rule for a check node is the parity function (XOR) of its incoming messages, which we denote by $f(\Theta)$. Thus, the message to v is $f(\Theta \setminus v)$, providing v with updated information from the rest of the graph. For a variable node in the LLR domain, the update rule is summation. When v is systematic, the updated soft value is $L = f(\Theta \setminus v) + L_0$, and the node can only relay its input message, the channel value $L - f(\Theta \setminus v) = L_0$, to its single adjacent check node. As such, v may be said to be in a listening or passive state. If this position is unreliable, it will not disturb the rest of the graph, while still receiving information such that L may converge.

Still, the gain of ABP can not be accredited to this isolation effect alone. Specifically, it is known that modifying the structure of $\mathbf{TG}(H)$ during SPA decoding may, in itself, improve performance [1–7, 10, 11]. Such diversity will change the structure of cycles and node-degree distributions, and, generally, alter the flow of messages during SPA decoding. After the GE stage, the new $\mathbf{TG}(H)$ is initialized for the next SPA iteration, by damping each variable node. Damping involves scaling down the extrinsic contribution to the LLR by a damping coefficient, $0 < \alpha < 1$. This is then accumulated on the input to the next iteration,

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

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

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

where $\Gamma_j^v = \Sigma_{u \in \mathcal{N}_v} \mu_j^{v \leftarrow u}$ is the extrinsic contribution to variable node v (the sum of all incoming messages, $\mu_j^{v \leftarrow u}$) in iteration j, and $\Gamma_0^v \triangleq 0$. The global damping stage (GD) applies (1) to all variable nodes in $\mathbf{TG}(H)$, followed by discarding all current extrinsic information; setting $\mu_j^{v \leftarrow u} = 0$ for all $v \in \mathbf{TG}(H)$. The next flooding iteration is initialized based on $\mathbf{L}_{j+1}$ only. This slightly complicates the argument of isolating a variable node, as the input to v is no longer fixed to the channel value, L_0^v. As ABP uses a constant damping coefficient for each iteration, the accumulation may be expressed as $L_{j+1}^v = L_0^v + \alpha \Sigma_{j'=1}^{j} \Gamma_{j'}^v$. Thus, damping never affects the channel value, and we see that the accumulation is negligible when α is small, such that the isolation argument still holds.

IV. ABP-ELC

Let us initially describe the proposed ABP-ELC algorithm simply as SPA iterations interspersed with a novel 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 ABP algorithm arise naturally.

The first distinction is that SPA-ELC decoding has been shown to be effective for $p \ll n - k$ [3], meaning a reduction in complexity – the major concern with ABP. Some further distinctions arise from the description of ELC. ELC requires that H is in systematic form, just as the GE stage of ABP will immediately reduce any H to systematic form. As ELC on a systematic position has no effect, we may immediately skip any position already in $\mathcal{P}$ while processing σ. The cancellation problem is handled by flagging the check node, u, as *used,* such that it will not be considered again in this ELC stage.

A. Improved Heuristic to Select ELC Positions

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

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

where $|v|$ is shorthand for $|L^v|$. The ABP-ELC(p) algorithm processes the p first information positions in σ, applying (2) to determine the ELC locations. In the event where $u^\star = \emptyset$, no ELC is possible for this position, and ABP-ELC moves on to the next-worst information position. No additional measures are needed to avoid the cancellation problem. The algorithm simply works from both ends of σ, pairing the weakest information position with the strongest parity position. The resulting ELC induces a swap of the corresponding columns across the bipartition, and requires that we keep track of the bipartition.

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

Proof: Consider two positions, v and w, which are both adjacent to u. Without loss of generality, say $|v| < |w|$, so we first consider v. Then, assume (2) gives $u^\star = u$, and we perform ELC on (u, v) such that v_u becomes non-systematic, and the systematic node of u is now v. When we later consider w, choosing the same $u^\star = u$ would entail a swap of w and v across the bipartition, thus cancelling the previous swap. However, this choice of $u^\star$ is not possible since $|v| < |w|$. ■

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

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

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

Proof: Say the previous ELC was on (u, v), which swapped v_u into $\mathcal{I}$. If the heuristic later reaches this same $w = v_u$ (before exhausting p), no subsequent ELC is possible for this position, according to (2). In the initial ELC, v_u was the most reliable parity position adjacent to any $\tilde{u} \in \mathcal{N}_v$ such that $|v_{\tilde{u}}| > |v|$. However, this now means that $|w| > |v_{u'}|\ \forall\, u' \in \mathcal{N}_w$, since $w = v_u$, and (2) yields $\emptyset$. ■

Again, Example 1 shows a case in the last stage, when $v = v_0$ which was swapped in the first stage. We have now covered all the possible cancellation issues, without any extra bookkeeping or complexity in the ABP-ELC heuristic (apart from keeping track of the bipartition).

B. Local-Neighborhood Damping

We have previously described a simplified, edge-local damping operation (LD), whose action is restricted to the local subgraph affected by ELC [3]. We now generalize LD

to include the incoming message $\mu_j^{v\leftarrow u}$ on an edge (u, v), producing a new outbound message,

$$\mu_{j+1}^{v\rightarrow u} = L_j^v + \alpha(L_j^v + \Gamma^v - \mu_j^{v\leftarrow u}). \tag{3}$$

Edges inserted by ELC contain no information, $\mu_j^{v\leftrightarrow u} \triangleq 0$, so these are initialized using (3), in accordance with the description in [3]. With LD, edges unaffected by ELC are left unchanged, so the reasoning for accumulating on the input no longer applies (**L** remains the channel vector for the entire frame). We define local-neighborhood damping (ND) as (3) applied to *all* edges (not just new edges) adjacent to all variable nodes affected by ELC on an edge (u', v'), namely $v \in \mathcal{N}_{u'}$. Since all edges adjacent to every $v \in \mathcal{N}_{u'}$ are damped, it is advantageous to also accumulate, according to (1), on the input of each v. Hence, ND is a step towards GD, in that it operates on a node rather than edge level, while taking advantage of any extrinsic information on edges unaffected by ELC.

V. Results

The frame error-rate (FER) performance of ABP-ELC is simulated on the $[31, 25, 7]$ 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 with the configurations specified in Table I. For all algorithms, the maximum number of iterations is $T = I_1 I_2 I_3$. For ABP-ELC, the damping rule, D, is LD or ND. When $I_2 = T$ and $I_1 = I_3 = 1$, damping is constant ($\alpha = \alpha_0$ for the entire frame), while damping is disabled by configuring $\alpha_0 = 1$. In order to compare with other algorithms, the same parameters were used for our simulations. Most importantly, this regards α_0 and T. For ABP, we use "Variation A" (avoid weight-1 columns), whereas we do not use "Variation B" (list decoding) [5]. For SPA and SPA-PD we use a non-systematic matrix optimized on weight. For ABP and ABP-ELC, the initial matrix is less important, due to the effects of the GE or ELC stages.

It is interesting to note that ABP-ELC outperforms SPA-PD for the QR code, which has a very large automorphism group [3]. For both codes, Figs. 3(a) and 4(a) show that the ABP-ELC decoder also has a gain over the successful ABP algorithm, even when ABP uses "Variation C," a symbol level hard-decision list decoding (HDD) stage for the RS code (denoted ABP & HDD). This gain is attributed mainly to our improved ABP-ELC heuristic, but we also observe a gain due to the modified damping. For both codes, the performance of ABP-ELC shows an improvement over ABP, and even over ABP & HDD until a FER of 10^{-4}. With ABP-ELC & HDD, an additional constant gain in FER is achieved, most importantly in terms of a reduced flooring effect at high signal-to-noise ratios (SNRs). Even without the HDD stage on the RS code, ABP-ELC is only about 0.07 dB away from ABP-ELC & HDD, until a FER of 10^{-4}. For the QR code, ABP-ELC approaches the union bound (based on the full weight enumerator of the code) within 0.3 dB at a FER of 10^{-5}. To examine the benefit of reducing p, we simulate ABP-ELC over SNRs 3.5, 4.5, and 5.0 dB for $0 < p \leq n - k$, Figs. 3(b) and 4(b). The performance of ABP at the corresponding SNR points is indicated by the horizontal grey lines. We observe that the performance of ABP-ELC improves with increasing p only initially, before flattening out at an optimal value, $p^\star \ll n - k$, of approximately 7 and 10 for the QR and RS code, respectively. For the QR code, the performance of ABP is matched (intersect the grey lines) for $p \approx 3$. We also see that the optimal type of damping (LD or ND) depends on the code, and can have a significant impact on performance, as shown in Fig. 4(b).

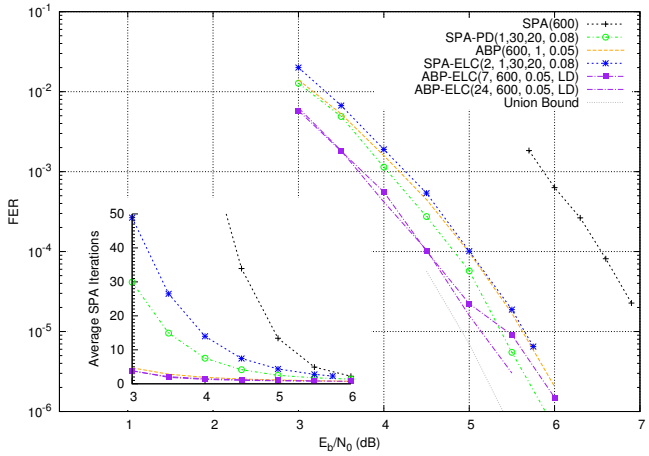
We now focus on the reduction in decoding complexity, measured in terms of average total number of SPA iterations. The HDD stage is implemented using a "genie-aided stopping criterion" correcting up to $\left\lfloor \frac{d_{\min}-1}{2} \right\rfloor = 3$ symbol errors, and may stop early [5]. An actual implementation, e.g. using the Berlekamp-Massey algorithm, would require a list implementation. Since the number of iterations would always be T, the complexity in terms of SPA iterations is not included. We also consider the complexity in terms of average number of ELC operations per GE or ELC stage. Due to cancellation effects, an ELC stage may perform less than p ELC operations. Define a function, $\bar{p}(p)$, to give the average number of ELC operations performed per ELC stage, for a given p. Figs. 3(c) and 4(c) compare $\bar{p}(p)$ against the linear (possibly redundant) ELC complexity of ABP. As stated also in [6], a reduction can be achieved by simply improving the implementation of the GE stage to do no work whenever an unreliable position is already in $\mathcal{P}$. The plots verify that ABP performs $n - k - \bar{p}(n - k)$ redundant operations (pivotals). Even comparing against such

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

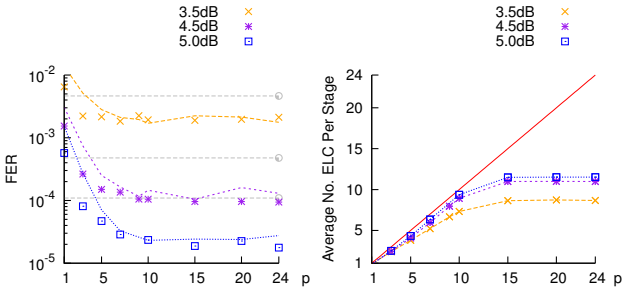
$\alpha := \alpha_0$
for I_3 times **do**
  Restart decoder from channel vector
  for I_2 times **do**
    (C) HDD stage. RS code only
    Stop if syndrome check is satisfied
    Apply damping rule, D, with coefficient α
    Apply at random p operations, OP
    (A) Random row additions (avoid weight-1 columns)
    for I_1 times **do**
      Apply SPA iteration (flooding scheduling)
    end for
  end for
  Increment α towards 1
end for

TABLE I
Decoding algorithms simulated in this work, and the corresponding configurations of Algorithm 1

SPA(T)	SISO-HDPC$(0, 1, T, 1, 1, -, -)$
ABP$(T, 1, \alpha)$	SISO-HDPC$(1, 1, T, 1, \alpha, \mathrm{GE}, \mathrm{GD})$
SPA-PD$(I_1, I_2, I_3, \alpha_0)$	SISO-HDPC$(1, I_1, I_2, I_3, \alpha_0, \mathrm{PD}, \mathrm{GD})$
SPA-ELC$(p, I_1, I_2, I_3, \alpha_0)$	SISO-HDPC$(p, I_1, I_2, I_3, \alpha_0, \mathrm{ELC}, \mathrm{LD})$
ABP-ELC(p, T, α, D)	SISO-HDPC$(1, 1, T, 1, \alpha, \mathrm{ABP{-}ELC}(p), D)$



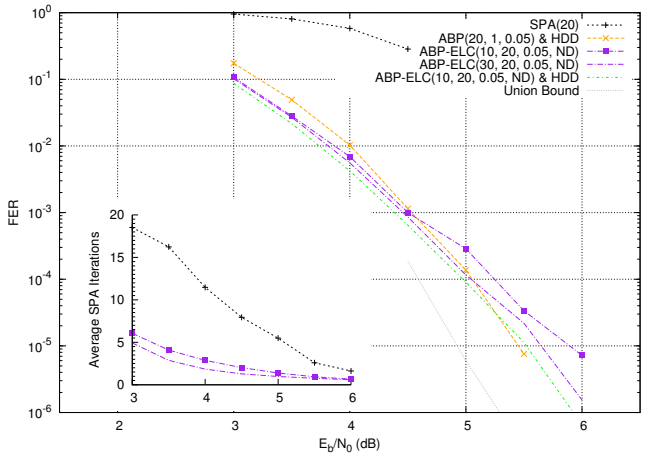
(a) FER performance, and average number of SPA iterations per frame.



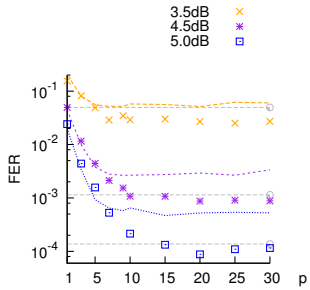
(b) FER plotted against p.

(c) ELC usage, $\bar{p}(p)$, of ABP-ELC.

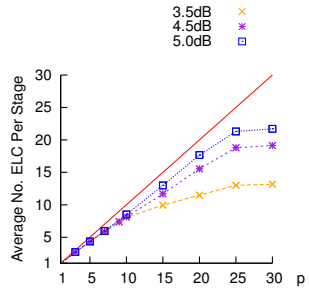
Fig. 3. For the $[48, 24, 12]$ extended QR code, the best performance is achieved using LD. The performance using ND is shown by the curved lines in Fig. 3(b). Maximum $T = 600$ iterations.



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



(b) FER plotted against p.



(c) ELC usage, $\bar{p}(p)$, of ABP-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. The performance using LD is indicated by the curved lines in Fig. 4(b). Maximum $T = 20$ iterations.

an improved, reduced-complexity GE stage, we see that $\bar{p}(p)$ grows linearily before flattening out at $p' \gtrsim 2p^\star$. Thus, we may achieve a further reduction in ELC complexity (over an improved GE stage) of $\bar{p}(n-k) - \bar{p}(p^\star)$.

VI. Conclusion and Future Work

We have described an ABP-ELC decoder to target the least reliable positions (inferred error positions) during decoding, as well as a generalized, *local-neighborhood* damping rule. An improvement is shown over related algorithms, both in terms of FER performance and complexity, which is ascribed mainly to an improved heuristic to apply the ELC operations. Simultaneously, the amount of ELC operations can be reduced significantly, compared to a GE stage, for an improvement also in complexity. Extensive simulation data is presented for an extended QR code and a RS code.

Future work is concerned with further extensions of the ABP-ELC heuristic, and using a non-systematic variant of ELC to avoid degree-1 nodes. We are also working on a distributed implementation of the sorting stage, where each check node sorts its adjacent variable nodes.

References

[1] 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, "Iterative decoding on multiple Tanner graphs using random edge local complementation," in *Proc. IEEE Int. Symp. Inform. Theory*, Seoul, Korea, Jun./Jul. 2009, pp. 899–903.

[4] J. Jiang and K. R. Narayanan, "Iterative soft 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, Mar. 2006.

[7] 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, pp. 98–101.

[8] L. E. Danielsen and M. G. Parker, "Edge local complementation and equivalence of binary linear codes," *Des. Codes Cryptogr.*, vol. 49, no. 1-3, pp. 161–170, Dec. 2008.

[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. Y. 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.