# Transactions Letters

## On Soft-Input Soft-Output Decoding Using "Box and Match" Techniques

Philippa A. Martin, *Member, IEEE*, Antoine Valembois, Marc P. C. Fossorier, *Senior Member, IEEE*, and Desmond P. Taylor, *Fellow, IEEE*

***Abstract*—The *box and match decoding algorithm* (BMA) significantly reduces the computational complexity of the ordered statistic decoding algorithm at the expense of increased memory requirements. In this letter, a soft-input/soft-output version of the BMA is developed. Additional complexity-reduction techniques are also described.**

***Index Terms*—Iterative decoding, list decoding, product codes.**

### I. Introduction

SOFT-INPUT soft-output (SISO) decoding of linear block codes within a concatenated code structure can be prohibitively complex. This has led to reduced-complexity approaches being developed [1]–[5]. We focus here on list-based SISO *ordered statistic decoding* (OSD) algorithms, as they have been shown to retain performance as the minimum Hamming distance $d_{H,\min}$ of the component codes increases [1]. The SISO order-$2i$ reprocessing algorithm was introduced in [1], and a variant appeared in [6]. In [5], the complexity of this algorithm was reduced by estimating the extrinsic information in some positions, and performance was improved by scaling the extrinsic information. The *modified SISO order-$2i$ reprocessing* (MSISO-$2i$) algorithm of [5] was also found to outperform that of [2] at low bit-error rates (BERs). However, this algorithm still tends to be extremely complex for large values of $\min\{n-k, k\}$ and $d_{H,\min}$, where $n$ is the block length and $k$ is the dimension of a component code. To facilitate decoding, we develop a SISO version of the *box and match algorithm* (BMA-$(i,s)$) of [7], known as SISO BMA-$(i,s)$, using techniques from [1], [3], and [5]. It may be regarded as an extension of MSISO-$2i$ to more complex component codes.

BMA-$(i,s)$ is an OSD algorithm, which encodes a subset of all possible length-$k$ error patterns to produce a list of possible transmitted codewords and outputs the most likely codeword in the list. It allows the complexity of order-$2i$ reprocessing [8] to be reduced on the order of the square root of the number of computations at the cost of increased memory requirements. BMA-$(i,s)$ offers a range of tradeoffs between computational complexity, performance, and memory requirements through variation of the design parameters $i$ and $s$.

The soft information used and produced by the SISO BMA-$(i,s)$ is described in Section II. The algorithm is described and developed in Section III of this letter. In Section IV, we develop sufficient conditions for SISO optimality to decrease the number of error patterns considered when $i > 1$. In addition, complexity-reduction ideas from [5] are applied. We look at various tradeoffs between computational complexity and performance in Section V. Finally, conclusions are given in Section VI.

### II. Soft Information

SISO BMA-$(i,s)$ can be used to decode block codes in many concatenated structures, channels, and modulation schemes. In this letter, we consider quadrature phase-shift keying (QPSK) transmitted over a memoryless additive white Gaussian noise (AWGN) channel. We consider product codes (PCs) with two identical $(n, k, d_{H,\min})$ extended Bose–Chaudhuri–Hocquengem (BCH) component codes. The PC is iteratively decoded by using SISO BMA-$(i,s)$ to alternately decode the row and column component codes. Each decoding stage decodes all rows or all columns, so it takes two decoding stages to decode the PC once (called a single iteration). The overall iterative structure remains that of [5] with MSISO-$2i$ replaced by SISO BMA-$(i,s)$.

For ease of notation, we consider one component codeword block at a time (either a row or a column in the PC matrix). We begin by defining the received soft-input log-likelihood ratio (LLR) for position $v \in \{1, \ldots, n\}$ in a component codeword block as

$$L_{ch,v} = \log\left(\frac{\Pr(e_v = 1|r_v)}{\Pr(e_v = 0|r_v)}\right) = \frac{2r_v}{\sigma^2} \tag{1}$$

where $r_v$ is the $v$th received symbol, $e_v$ is the $v$th bit, and $\sigma^2$ is the noise variance, which is estimated or otherwise known.

The list of codewords found by SISO BMA-$(i,s)$ is used to calculate extrinsic information for iterative decoding. We assume equiprobable bits. As in [5], we define the LLR for the soft input to the $q$th decoding stage for the $v$th bit as

$$\lambda_v(q) = L_{ch,v} + \alpha(q) w_v(q-1) \tag{2}$$

Paper approved by T.-K. Truong, the Editor for Coding Theory and Techniques of the IEEE Communications Society. Manuscript received July 29, 2003; revised June 1, 2004. This work was supported in part by the Marsden Fund, in part by the Public Good Science Fund of New Zealand, and in part by the National Science Foundation under Grant CCR-00-98029. This paper was presented in part at the International Symposium on Information Theory, Yokohama, Japan, June–July 2003.

P. A. Martin and D. P. Taylor are with the Electrical and Computer Engineering Department, University of Canterbury, Christchurch 8020, New Zealand (e-mail: philippa_martin@ieee.org; taylor@elec.canterbury.ac.nz).

A. Valembois and M. P. C. Fossorier are with the Department of Electrical Engineering, University of Hawaii, Honolulu, HI 96822 USA (e-mail: marc@spectra.eng.hawaii.edu).

Digital Object Identifier 10.1109/TCOMM.2004.838733

0090-6778/04$20.00 © 2004 IEEE

where $\alpha(q)$ is a heuristically or adaptively chosen [4] scaling factor, and $w_v(q-1)$ is the extrinsic information for the $v$th bit after the $(q-1)$th decoding stage. We write, as in [5]

$$w_v(q) = \frac{1}{2}\left(\sum_{l=1}^{n}\left(z\left(e_l^{v1}\right) - z\left(e_l^{v0}\right)\right)\lambda_l(q)\right) - \lambda_v(q) \quad (3)$$

where $z(\cdot)$ maps $\{0,1\}$ to $\{-1,+1\}$ and $\boldsymbol{E}^{va} = (e_1^{va},\ldots,e_n^{va})$ is the best codeword found by SISO BMA-$(i,s)$ with $e_v = a, a \in \{0,1\}$.

Unlike other algorithms [3]–[5], SISO BMA-$(i,s)$ searches for a list of the most likely coset elements, $\boldsymbol{C}^{vb}$ ($c_v = b$, where $b = y_v \oplus a$), rather than the transmitted codewords, $\boldsymbol{E}^{va}$. Assuming the same code subspace is searched, we can, for each $v$, write $\boldsymbol{C}^{vb} = \boldsymbol{Y} \oplus \boldsymbol{E}^{va}$, where $\boldsymbol{Y} = (y_1,\ldots,y_n)$ is the hard decision on the soft input. SISO BMA-$(i,s)$ uses the *ellipsoidal weight* (EW) to choose the most likely coset elements, where the EW of coset element $\boldsymbol{C} = (c_1,\ldots,c_n)$ is defined as [7]

$$\mathrm{EW}(\boldsymbol{C}) = \sum_{l=1,c_l=1}^{n} z(y_l)\lambda_l(q) = \sum_{l=1,c_l=1}^{n} |\lambda_l(q)| \geq 0. \quad (4)$$

Then (3) can be written as

$$\begin{aligned} w_v(q) &= \frac{z(y_v)}{2}\left(\sum_{l=1}^{n} z(y_l)\left(z\left(c_l^{v1}\right) - z\left(c_l^{v0}\right)\right)\lambda_l(q)\right) - \lambda_v(q) \\ &= z(y_v)\left(\mathrm{EW}_v^{\min}(1) - \mathrm{EW}_v^{\min}(0)\right) - \lambda_v(q) \end{aligned} \quad (5)$$

where $\mathrm{EW}_v^{\min}(b)$ is the EW of the best coset element found by the decoder with $c_v = b, b \in \{0,1\}$. Thus, we can use (and store) a list of EWs to calculate the extrinsic information.

## III. SISO BMA-$(i,s)$

Now consider the processing of a single component codeword block by SISO BMA-$(i,s)$, based on the description of the BMA-$(i,s)$ in [7]. As in [1], SISO BMA-$(i,s)$ creates and operates on an equivalent reordered systematic code, $\mathbb{C}$, defined by matrix $\boldsymbol{G}$. The transmitted linear binary block code $\mathbb{C}''$ is defined by generator matrix $\boldsymbol{G}''$. The elements of $\boldsymbol{\lambda}(q)$ are reordered from most to least reliable to define permutation $\pi_1(\cdot)$, which is used to create an equivalent reordered code with generator matrix $\boldsymbol{G}' = \pi_1(\boldsymbol{G}'')$. Matrix $\boldsymbol{G}'$ is reduced to systematic form, $\boldsymbol{G}$, by making the $k$ *most reliable independent positions* (MRIPs) the information positions, thus defining permutation $\pi_2(\cdot)$, where $\boldsymbol{G} = \pi_2(\boldsymbol{G}') = \pi_2(\pi_1(\boldsymbol{G}''))$. The $k$ MRIPs, $v = 1,\ldots,k$, in the reordered soft input form the *most reliable basis* (MRB). These positions are ordered from most to least reliable, as are the $n-k$ parity positions of $\mathbb{C}$. The reordered vectors will be used in the remainder of the algorithm.

SISO BMA-$(i,s)$ encodes length-$k$ error patterns $\boldsymbol{P}$ in the MRB to produce coset elements $\boldsymbol{C}_p = (c_{p,1},\ldots,c_{p,n})$, where $\boldsymbol{C}_p = \boldsymbol{C}_0 \oplus \boldsymbol{P}\boldsymbol{G}$ and $\boldsymbol{C}_0$ is the coset element for $\boldsymbol{P} = \boldsymbol{0}$. The order-$i$ reprocessing algorithm of [1] considers all coset elements of Hamming weight $i$ or less over the MRB. SISO BMA-$(i,s)$ considers all of these coset elements, and, in addition, considers all coset elements of Hamming weight $2i$ or less over the first $s+k$ positions in $\mathbb{C}$, where $0 \leq s \leq n-k$. This covers the MRB and the $s$ most reliable positions outside the MRB.[1] The additional $s$ positions, $v = k+1,\ldots,k+s$, are called the *control band* (CB), and the value of $s$ is set by design. The binary pattern of $\boldsymbol{C}_p$ over the CB is called the $S$-complement of $\boldsymbol{P}$, and is denoted $\boldsymbol{\sigma}_p = (c_{p,k+1},\ldots,c_{p,k+s})$.

SISO BMA-$(i,s)$ consists of phases 0 to $i$ [7], similar to the OSD algorithm of [1]. During phase $j, 0 \leq j \leq i$, it stores previously processed error patterns $\boldsymbol{P}$ of Hamming weight $j$ in $2^s$ memory boxes. There is one box for each $S$-complement value. We denote the box containing previously processed error patterns $\boldsymbol{P}$ with $S$-complement $\boldsymbol{\sigma}_p$ by $B(\boldsymbol{\sigma}_p)$. Phase $j$ of SISO BMA-$(i,s)$ [7] computes all new coset elements with Hamming weight $j$ over the MRB. In addition, coset elements with Hamming weight $2j-1$ or $2j$ over the combination of the MRB and the CB, denoted MRB + CB, are computed.

Based on [5], an error pattern $\boldsymbol{P}$ is processed as follows.

1) Compute the coset element for $\boldsymbol{P}, \boldsymbol{C}_p$, and compute the $S$-complement of $\boldsymbol{P}, \boldsymbol{\sigma}_p$.
2) Compute the EW of $\boldsymbol{C}_p, \mathrm{EW}(\boldsymbol{C}_p)$. For $v = 1,\ldots,n$, if $\mathrm{EW}_v^{\min}(c_{p,v}) > \mathrm{EW}(\boldsymbol{C}_p)$, then update[2] $\mathrm{EW}_v^{\min}(c_{p,v})$. Also update the maximum stored EW, $\mathrm{EW}_{\max}$, if required. Update the minimum stored EW, $\mathrm{EW}_{\min} = \mathrm{EW}(\boldsymbol{C}_p)$, and the corresponding coset element, $\boldsymbol{C}_{\min} = \boldsymbol{C}_p$, if $\mathrm{EW}(\boldsymbol{C}_p) < \mathrm{EW}_{\min}$.

The procedure above is called Process $(\boldsymbol{P})$.

Many error patterns are created using the patterns stored in boxes. Consider two error patterns $\boldsymbol{P}$ and $\boldsymbol{P}'$ with Hamming weight $j$ and $j'$, respectively, where $1 \leq j' \leq j$. The process of creating error patterns $\boldsymbol{P} + \boldsymbol{P}'$, using error patterns from the boxes $\boldsymbol{P}$, with another error pattern $\boldsymbol{P}'$, is as follows [7].

1) For all $\boldsymbol{\sigma}$ with Hamming weight $j-j'-1$ or $j-j'$, locate box $B(\boldsymbol{\sigma} + \boldsymbol{\sigma}_0 + \boldsymbol{\sigma}_{p'})$.
2) For each pattern $\boldsymbol{P}$ of Hamming weight $j$ in box $B(\boldsymbol{\sigma} + \boldsymbol{\sigma}_0 + \boldsymbol{\sigma}_{p'})$, use procedure Process $(\boldsymbol{P} + \boldsymbol{P}')$. However, $\boldsymbol{P} + \boldsymbol{P}'$ is processed only if the most reliable 1 in $\boldsymbol{P}$ is less reliable than the least reliable 1 in $\boldsymbol{P}'$. Note $\boldsymbol{\sigma}_{p+p'} = \boldsymbol{\sigma}$ [7].

The procedure above is called Match $(\boldsymbol{P}', j', j)$.

We denote an error pattern of Hamming weight $j$ with 1's in positions $p_1 < p_2 < \cdots < p_j$ by $\boldsymbol{P}_{p_1 p_2 \ldots p_j}$. Based on [7], SISO BMA-$(i,s)$ can be summarized as follows.

1) Calculate the soft input to the $q$th decoding stage using (1) and (2).
2) Create the equivalent reordered systematic code $\mathbb{C}$.
3) Perform phase 0: Compute the coset element for $\boldsymbol{P} = \boldsymbol{0}, \boldsymbol{C}_0 = (c_{0,1},\ldots,c_{0,n})$. Initialize $\mathrm{EW}_v^{\min}(c_{0,v}) = \mathrm{EW}(\boldsymbol{C}_0)$ for $v = 1,\ldots,n, \mathrm{EW}_{\min} = \mathrm{EW}(\boldsymbol{C}_0)$, and $\boldsymbol{C}_{\min} = \boldsymbol{C}_0$.
4) *for* Phase $j = 1,2,\ldots,i$
   *for* $p_1 = k-j+1, k-j,\ldots,1$
   *Match* $(\boldsymbol{P}_{p_1}, 1, j)$.
   *for* $p_2 = k-j+2, k-j+1,\ldots,p_1+1$

[1] Note that the definition of $s$ used in [7] includes the $k$ MRB positions.

[2] The complexity of this operation can be significantly reduced by storing and updating the maximum value over $v$ of $\mathrm{EW}_v^{\min}(0)$ and the maximum value over $v$ of $\mathrm{EW}_v^{\min}(1)$. Only test if $\mathrm{EW}_v^{\min}(0)$ and $\mathrm{EW}_v^{\min}(1)$ need to be updated if the current EW is less than the respective stored maxima.

*Match* $(\boldsymbol{P}_{p_1p_2}, 2, j)$.

$\vdots$

*for* $p_j = k, k-1, \ldots, p_{j-1}+1$

*Process* $(\boldsymbol{P}_{p_1p_2\ldots p_j})$.

*Match* $(\boldsymbol{P}_{p_1p_2\ldots p_j}, j, j)$.

*Store* $\boldsymbol{P}_{p_1p_2\ldots p_j}$ in its box.

5) Calculate the extrinsic information vector $\boldsymbol{w}(q) = (w_1(q), \ldots, w_n(q))$ using (5). Perform the inverse permutation $\pi_1^{-1}(\pi_2^{-1}(\boldsymbol{w}(q)))$ to obtain the ordering of $\mathbb{C}''$.

## IV. Complexity-Reduction Techniques

Some relatively simple techniques can be used to further reduce the decoding complexity with virtually no loss in performance. We consider two such techniques.

### A. Suboptimal Error-Pattern Set Reduction

As in [5], we can significantly reduce the number of error patterns considered by only producing those with support in the $0 \leq \mathcal{N}_s \leq k$ *least reliable positions* (LRPs) within the MRB. Different values of $\mathcal{N}_s$ could be used for each phase to further reduce complexity.

If $\mathcal{N}_s < k$, then either $\boldsymbol{C}^{v0}$ or $\boldsymbol{C}^{v1}$ is not found for MRB positions $v = 1, \ldots, k - \mathcal{N}_s$ (and possibly in some parity positions). We estimate the extrinsic information in these positions using [4] and [5] as

$$w_v(q) = z(d_v)\frac{1}{|\theta|}\sum_{i=1, i\in\theta}^{n} z(d_i)w_i(q) \tag{6}$$

where $\theta$ is the set of positions where $\boldsymbol{C}^{v0}$ and $\boldsymbol{C}^{v1}$ were found, $|\theta|$ is the size of the set, and $\boldsymbol{D} = \boldsymbol{C}_{\min} \oplus \boldsymbol{Y} = (d_1, \ldots, d_n)$. If (6) makes $z(d_v)w_v(q) < 0$, we instead use the cruder estimate

$$w_v(q) = z(\tilde{d}_v)\min_{i\in\theta}\{z(d_i)w_i(q) > 0\} \tag{7}$$

where $\min(a > 0)$ is the minimum positive value of $a$.

The error-pattern set can be further reduced in later iterations when $i > 1$ for a small degradation in performance. If the input vector to the decoding stage is a codeword, so that $\mathrm{EW}(\boldsymbol{C}_0) = 0$, then only perform phase 0 and phase 1. In this case, we cannot find a better codeword than $\boldsymbol{C}_0$. Phase 1 is only used to calculate extrinsic information. This will be used in simulations when the sufficient conditions described in the next subsection are used.

### B. Sufficient Conditions for SISO Optimality

We now develop sufficient conditions for SISO optimality based on [5] and [7]–[9]. Note that the main difference between the sufficient conditions for SISO optimality used here and those derived in [7] comes from the fact that $\mathrm{EW}_{\max}$ instead of $\mathrm{EW}_{\min}$ is used as a threshold. The conditions can reduce the average complexity of step 4 when $i > 1, j > 1$ without changing performance, in that they allow the elimination of subsets of error patterns not needed to evaluate the extrinsic information. They can also be used with $\mathcal{N}_s \leq k$. Phase 0 and phase 1 are always processed, as they initialize the metrics in the algorithm.

TABLE I
Hamming Weight Over the MRB and CB of Coset Elements Considered During Phases 1–3

| phase, $j$ | MRB | CB | Procedure in step 4 |
|---|---|---|---|
| 1 | 1 | ? | Process$(\boldsymbol{P}_{p_1})$. |
| 1 | 1+1 | 0 | Match$(\boldsymbol{P}_{p_1}, 1, 1)$. |
| 2 | 2 | $\geq 1$ | Process$(\boldsymbol{P}_{p_1p_2})$. |
| 2 | 1+2 | $\leq 1$ | Match$(\boldsymbol{P}_{p_1}, 1, 2)$. |
| 2 | 2+2 | 0 | Match$(\boldsymbol{P}_{p_1p_2}, 2, 2)$. |
| 3 | 3 | $\geq 2$ | Process$(\boldsymbol{P}_{p_1p_2p_3})$. |
| 3 | 1+3 | $\leq 2$ | Match$(\boldsymbol{P}_{p_1}, 1, 3)$. |
| 3 | 2+3 | $\leq 1$ | Match$(\boldsymbol{P}_{p_1p_2}, 2, 3)$. |
| 3 | 3+3 | 0 | Match$(\boldsymbol{P}_{p_1p_2p_3}, 3, 3)$. |

The conditions depend on the order of processing of error patterns/coset elements. In Table I, the Hamming weight of coset elements over the MRB and CB are shown for phases 1–3. As can be seen at the end of phase $(j-1)$, all coset elements of Hamming weight $2j-3$ or $2j-2$ over the MRB + CB have already been considered. This means all remaining coset elements have Hamming weight greater than $2j-2$ over the MRB + CB.

We can develop an estimated lower threshold, $\tau_{\mathrm{EW}}(\boldsymbol{C}_p)$, on the ellipsoidal weight $\mathrm{EW}(\boldsymbol{C}_p)$ for coset element $\boldsymbol{C}_p$. It may be written as

$$\tau_{\mathrm{EW}}(\boldsymbol{C}_p) = \mathrm{EW}_{\mathrm{lb}}(\boldsymbol{P}) + \mathrm{EW}_{\mathrm{lb}}(\boldsymbol{\sigma}_p) + R[\omega(\boldsymbol{P}) + \omega(\boldsymbol{\sigma}_p)] \tag{8}$$

where $R[\omega(\boldsymbol{P}) + \omega(\boldsymbol{\sigma}_p)]$ is called the resource and is a lower bound on the EW over the positions outside the MRB + CB, $\mathrm{EW}_{\mathrm{lb}}$ denotes a lower bound on the EW, and $\omega(a)$ denotes the Hamming weight of $a$. To calculate $R[\omega(\boldsymbol{P}) + \omega(\boldsymbol{\sigma}_p)]$, we use the fact that the Hamming distance between $\boldsymbol{C}_p$ and $\boldsymbol{C}_{\min}$ is $d_{H,\min}$ or greater. For simplicity, we assume that $\boldsymbol{C}_{\min}$ and $\boldsymbol{C}_p$ do not overlap. In addition, we force $R[\cdot]$ to be nondecreasing. We define $R[\omega(\boldsymbol{P}) + \omega(\boldsymbol{\sigma}_p)]$ as the sum of the EW in the $\max(0, d_{H,\min} - \omega(\boldsymbol{C}_{\min}) - \omega(\boldsymbol{P}) - \omega(\boldsymbol{\sigma}_p))$ LRPs in $\boldsymbol{C}_{\min}$ which have value zero. From the previous discussion, we know $\omega(\boldsymbol{P}) + \omega(\boldsymbol{\sigma}_p) \geq 2j-1$ for phase $j$, and so, only $R[2j-1], \ldots, R[2i]$ need to be considered in phase $j$.

If $\tau_{\mathrm{EW}}(\boldsymbol{C}_p) \geq \mathrm{EW}_{\max}$, then $\boldsymbol{C}_p$ does not improve any of the minimum EWs and can be discarded, where $\mathrm{EW}_{\max}$ is the maximum stored EW. Due to the order in which error patterns are processed, we are often able to discard other error patterns [8] as discussed below.

For simplicity, consider phase $j = 2$ of step 4 for $i = 2$. The sufficient conditions can be extended to larger values of $j$ and $i$. Before processing phase 2, we estimate $\tau_{\mathrm{EW}}(\boldsymbol{C}_p)$ of (8) using

$$\mathrm{EW}_{\mathrm{lb}}(\boldsymbol{P}) = \sum_{l=k}^{k-j+1}|\lambda_l|, \quad \mathrm{EW}_{\mathrm{lb}}(\boldsymbol{\sigma}_p) = \sum_{l=k+s}^{k+s-j+2}|\lambda_l| \tag{9}$$

and $R[\omega(\boldsymbol{P}) + \omega(\boldsymbol{\sigma}_p)] = R[2j-1]$. If $\tau_{\mathrm{EW}}(\boldsymbol{C}_p) \geq \mathrm{EW}_{\max}$, then step 4 is terminated, as no remaining error patterns will improve $\mathrm{EW}_v^{\min}(c_{p,v})$. Note that $\mathrm{EW}_{\mathrm{lb}}(\boldsymbol{P}) + \mathrm{EW}_{\mathrm{lb}}(\boldsymbol{\sigma}_p)$ in (9) usually does not correspond to a valid coset element, but does provide a lower bound on the EW for any valid $\boldsymbol{C}_p$.

Next we consider the calls on procedure[3] Match $(\boldsymbol{P}, j', j)$. Using (8), (9), and Table I, we estimate the lowest EW that Match$(\boldsymbol{P}_{p_1}, j' = 1, j = 2)$ could calculate by

$$\tau_{\mathrm{EW}}(\boldsymbol{C}_p) = \mathrm{EW}(\boldsymbol{P}_{p1}) + \mathrm{EW}_{\mathrm{lb}}(\boldsymbol{P}) + \mathrm{EW}_{\mathrm{lb}}(\boldsymbol{\sigma}) + R[j + j' + \omega(\boldsymbol{\sigma})]. \quad (10)$$

For Match $(\boldsymbol{P}_{p_1}, 1, 2)$, if $\tau_{\mathrm{EW}}(\boldsymbol{C}_p) = |\lambda_{p_1}| + |\lambda_k| + |\lambda_{k-1}| + R[3] \geq \mathrm{EW}_{\max}$, then do not perform Match$(\boldsymbol{P}_{p_1}, 1, 2)$ again. Similarly, if $\tau_{\mathrm{EW}}(\boldsymbol{C}_p) = |\lambda_{p_1}| + |\lambda_{p_2}| + |\lambda_k| + |\lambda_{k-1}| + R[4] \geq \mathrm{EW}_{\max}$, then do not perform Match$(\boldsymbol{P}_{p_1 p_2}, 2, 2)$ during the current $p_2$ loop.

Finally, there is no need to store pattern $\boldsymbol{P}_{p_1 p_2}$ if it will not produce a lower EW when used in Match $(\boldsymbol{P}_{p_1}, 1, 2)$ and Match$(\boldsymbol{P}_{p_1 p_2}, 2, 2)$ during the remainder of phase 2. To this end, do not store $\boldsymbol{P}_{p_1 p_2}$ if $\tau_{\mathrm{EW}}(\boldsymbol{C}_p) = |\lambda_{p_1}| + |\lambda_{p_2}| + |\lambda_{p_1 - 1}| + R[3] \geq \mathrm{EW}_{\max}$.

Next consider Process $(\boldsymbol{P}_{p_1 p_2})$ with estimated threshold

$$\begin{aligned}\tau_{\mathrm{EW}}(\boldsymbol{C}_p) &= \mathrm{EW}(\boldsymbol{P}_{p_1 p_2}) + \mathrm{EW}_{\mathrm{lb}}(\boldsymbol{\sigma}_p) + R[j + \omega(\boldsymbol{\sigma}_p)] \\ &= |\lambda_{p_1}| + |\lambda_{p_2}| + |\lambda_{k+s}| + R[3]. \end{aligned} \quad (11)$$

If $\tau_{\mathrm{EW}}(\boldsymbol{C}_p) \geq \mathrm{EW}_{\max}$, then consider only larger values of $p_2$ from now on, and exit the $p_2$ loop. If, in addition, $p_2 = k$, then stop phase 2.

## V. Simulation Results

This section presents simulation results for iterative decoding of product codes using SISO BMA-$(i, s)$ as a component decoder. All simulations use $\alpha = 0.5$, which was found empirically to perform well. The optimization of $\alpha$ is outside the scope of this letter. The best approach depends on the modulation, list of codewords (determined in part by $i, s$, and $N_s$), maximum number of iterations, and codes considered.

For the same value of $\mathcal{N}_s$, the MSISO-2$i$ algorithm and SISO BMA-$(i, s = 0)$ have comparable computational complexity [7], consider the same error patterns, produce the same extrinsic information, and have the same performance. By making $s > 0$, SISO BMA-$(i, s > 0)$ can reduce computational complexity at the cost of increased memory requirements.[4]

In Fig. 1, we show the tradeoff between computational complexity (expressed as the average number of error patterns considered per component codeword block during the first decoding stage) and performance (expressed as the BER after ten iterations) for the rate-0.779 $(128, 113, 6)^2$ PC. The number of error patterns is approximately halved by using $\mathcal{N}_s = \lceil 2k/3 \rceil = 76$ instead of $\mathcal{N}_s = k = 113$, for a negligible change in performance (as in [5]). The only additional complexity for $\mathcal{N}_s < k$ is due to estimating $w_v(q)$ in some positions using (6) or (7). As can be seen, $s = 4$ or $s = 5$ with $i = 1$ and $\mathcal{N}_s = 76$ provides a good tradeoff between complexity and performance for this code. Simulations found $i = 1$ was sufficient for this code.[5]

[3]The Match( ) procedures process a large number of the error patterns. The number of such error patterns could be reduced by testing to see if patterns in the boxes can be erased. This could reduce complexity once there are many patterns in each box, but it would increase complexity, otherwise.

[4]For BMA-$(i, s)$ the memory requirements increase by $2^s + \binom{k}{i}$ [7].

[5]Note that using a larger $i$ than necessary can increase correlation and degrade performance in some cases.

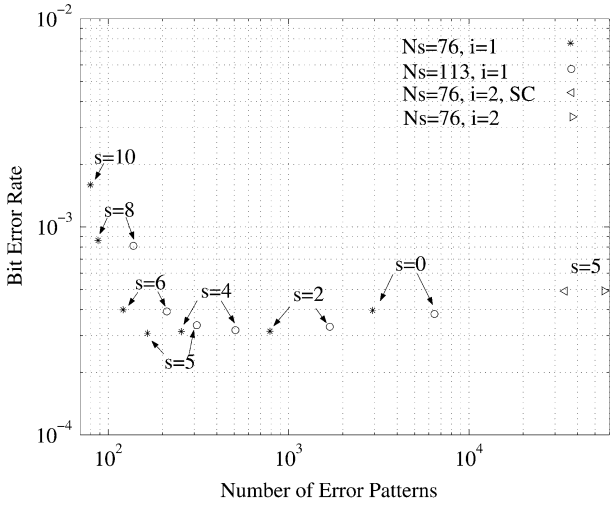


Fig. 1. BER versus the average number of error patterns per component codeword block for different values of $s, \mathcal{N}_s$, and $i$, for the $(128, 113, 6)^2$ PC with $\alpha = 0.5$ after ten iterations. Here $E_b/N_0 = 2.6$ dB, and SC stands for sufficient conditions.

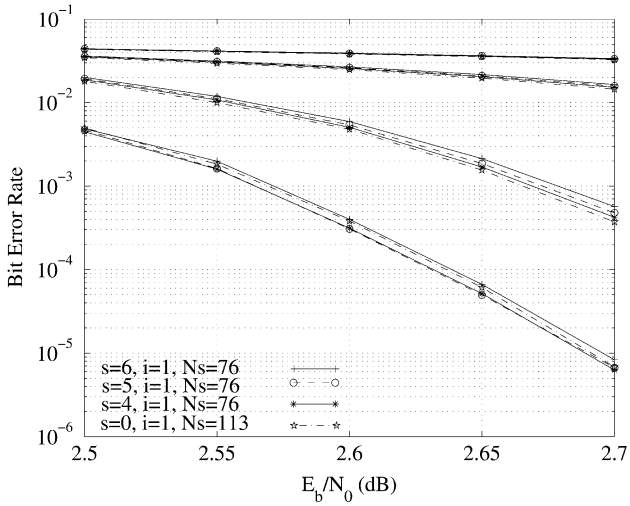


Fig. 2. BER for the $(128, 113, 6)^2$ PC, with $\alpha = 0.5, i = 1$ and different values of $\mathcal{N}_s$ and $s$, after 1, 2, 4, and 10 iterations. Performance improves with iteration.

In Fig. 2, the BER is shown for the $(128, 113, 6)^2$ PC with $i = 1, \mathcal{N}_s = 76$ or $\mathcal{N}_s = 113$, and $s = 0, s = 4, s = 5$, or $s = 6$. BER is plotted against $E_b/N_0$, where $E_b$ is the transmitted energy per data bit and $N_0$ is the noise spectral density. As a reference, capacity for this system is 1.9 dB. After ten iterations at a BER of $10^{-4}$ the performance (for $s = 4$) is approximately 0.37 dB better than that of the Chase-based algorithms of [3] and [4].

We also simulated the rate-0.685$(128, 106, 8)^2$ PC, which was slow to decode using the MSISO-2 algorithm or equivalently SISO BMA-(1, 0). The performance versus complexity tradeoffs for various values of $i, s$, and $\mathcal{N}_s$ are shown in Fig. 3. The average number of error patterns for $i = 2$ was slightly reduced by using the sufficient conditions of Section IV-B. The reduction was small for $\mathcal{N}_s = 71$ and $s = 4$, as many of the unlikely error patterns were already being ignored. As can be

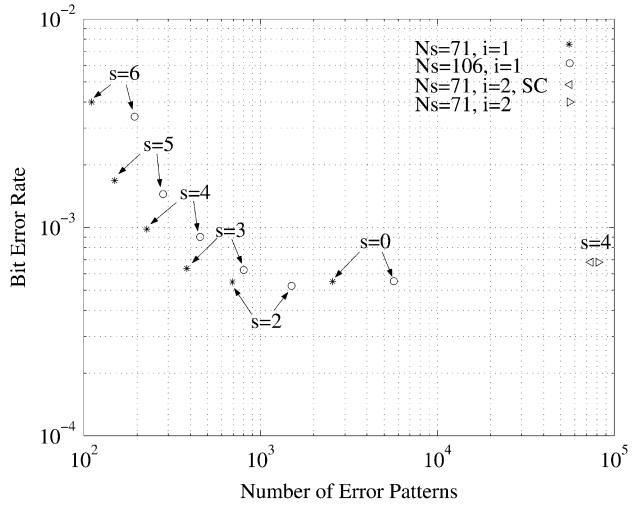


Fig. 3. BER versus the average number of error patterns per component codeword block for different values of $s, \mathcal{N}_s$, and $i$ for the $(128, 106, 8)^2$ PC using $\alpha = 0.5$ after ten iterations. Here $E_b/N_0 = 2.15$ dB, and SC stands for sufficient conditions.

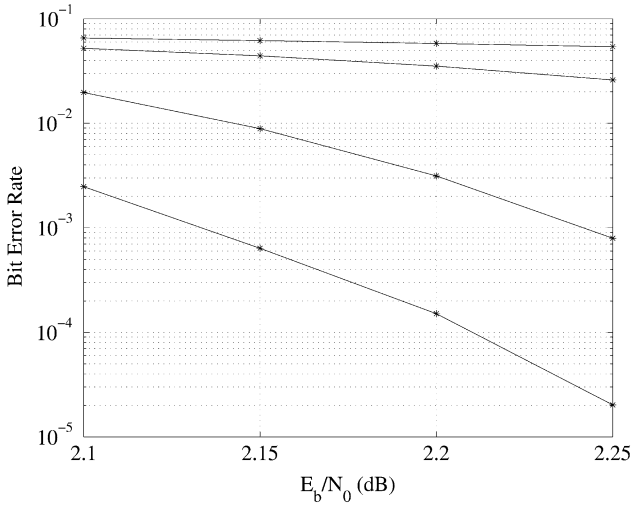


Fig. 4. BER for the $(128, 106, 8)^2$ PC with $\mathcal{N}_s = 71, i = 1, s = 3$, and $\alpha = 0.5$ after 1, 2, 4, and 10 iterations. Performance improves with iteration.

seen, $i = 1$ with $s = 2$ or $s = 3$ provide good tradeoffs between performance and computational complexity. Once again, $i = 1$ was found to be sufficient. However, it is expected that for larger values of $d_{H,\min}$, using $i > 1$ will provide significantly better performance. In Fig. 4, BER curves are given for the $(128, 106, 6)^2$ PC with $\mathcal{N}_s = 71, i = 1$, and $s = 3$.

Simulations using 16-ary pulse amplitude modulation (AM) or, equivalently, 256-ary quadrature AM were also considered for the memoryless AWGN channel [10]. At a BER of $10^{-5}$, after eight iterations, the $(128, 113, 6)^2$ PC (with $\mathcal{N}_s = k, i = 1, s = 4$, and $\alpha = 0.55$) was approximately 1.8 dB away from capacity.

## VI. Conclusion

We have developed a SISO version of BMA-$(i, s)$ for SISO decoding of component codes in an iterative structure. It allows computational complexity to be reduced, compared with previous SISO OSD algorithms [1], [5], [6] at the cost of negligible loss in performance and increased memory requirements. The approach appears best suited to codes with $d_{H,\min} \geq 6$ and moderate-to-large values of $n-k$. A range of tradeoffs between memory (storage) requirements, computational complexity, and performance are possible. Complexity can be further reduced by estimating the extrinsic information in some positions and by developing sufficient conditions for SISO optimality.

## Acknowledgment

The authors would like to thank Dr. D. M. Rankin for his help with the simulations. They also wish to thank the reviewers for their comments, which improved the presentation of this letter.

## References

[1] M. P. C. Fossorier and S. Lin, "Soft-input soft-output decoding of linear block codes based on ordered statistics," in *Proc. Globecom*, vol. 5, Nov. 1998, pp. 2828–2833.

[2] J. Fang, F. Buda, and E. Lemois, "Turbo product code: A well suited solution to wireless packet transmission for very low error rates," in *Proc. 2nd Int. Symp. Turbo Codes, Related Topics*, 2000, pp. 101–111.

[3] R. M. Pyndiah, "Near-optimum decoding of product codes: Block turbo codes," *IEEE Trans. Commun.*, vol. 46, pp. 1003–1010, Aug. 1998.

[4] P. A. Martin and D. P. Taylor, "On multilevel codes and iterative multistage decoding," *IEEE Trans. Commun.*, vol. 49, pp. 1916–1925, Nov. 2001.

[5] P. A. Martin, D. P. Taylor, and M. P. C. Fossorier, "Soft-input soft-output list-based decoding algorithm," *IEEE Trans. Commun.*, vol. 52, pp. 252–262, Feb. 2004.

[6] S. Vialle, "Construction et analyze de nouvelle structures de codage adaptées au traitement iteratif," Ph.D. dissertation (in French), Ecole Nat. Supérieure des Télécommun., Paris, France, 2000.

[7] A. Valembois and M. P. C. Fossorier, "Box and match techniques applied to soft-decision decoding," *IEEE Trans. Inform. Theory*, vol. 50, pp. 796–810, May 2004.

[8] M. P. C. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Trans. Inform. Theory*, vol. 41, pp. 1379–1396, Sept. 1995.

[9] D. J. Taipale and M. B. Pursley, "An improvement to generalized minimum-distance decoding," *IEEE Trans. Inform. Theory*, vol. 37, pp. 167–172, Jan. 1991.

[10] P. A. Martin, A. Valembois, M. P. C. Fossorier, and D. P. Taylor, "Reduced complexity soft-input soft-output "box and match" decoding," in *Proc. Int. Symp. Information Theory*, Yokohama, Japan, June 29–July 4, 2003.