This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the WCNC 2007 proceedings.

Product Codes and Parallel Concatenated Product Codes

Tina D.-H. Huang, Chi-Yuan Chang, Yan-Xiu Zheng and Yu T. Su
Department of Communications Engineering
National Chiao Tung University
Hsinchu, 30056, TAIWAN,
Email: ytsu@mail.nctu.edu.tw

***Abstract*— We study the decoding of product codes and a class of parallel concatenated product codes (PCPC). PCPC improves the minimum distance while retaining the merit of low decoding complexity of turbo product codes (TPC). We prove that using the Fibonacci interleaver does help increasing the minimum distance. The regularity of the interleaver also reduces the implementation complexity and makes parallel interleaving feasible. We show that Pyndiah's algorithm for decoding product codes produce an annealing effect similar to that of the so-called annealed belief propagation (ABP) algorithms which adjusts the "temperature" of an augmented cost surface. Decoding methods based on modified Pyndiah and annealed BCJR algorithms for both PC and PCPC are proposed and their performance is compared.**

I. Introduction

The Product code concept is very simple and relatively efficient for building very long block codes using two or more short block codes. Fig. 1 shows a typical (two-dimensional) product coding scheme that arranges the information symbols in a rectangular array and encodes each row and column individually by two linear block codes $\mathcal{C}^1$ and $\mathcal{C}^2$. The resulting augmented rectangular array of Fig. 1 forms a codeword of the product code $\mathcal{C}^1 \otimes \mathcal{C}^2$ with rows and columns being $\mathcal{C}^1$ and $\mathcal{C}^2$ codewords.

Pyndiah [3] proposes an iteratively decoding method that applies the Chase-II algorithm [5] to each component code to generate extrinsic information. The resulting turbo-decoded performance is quite impressive, considering the low complexity of the overall decoding algorithm as low minimum distance linear block code applied. As moderate minimum distance linear block code applied, such as double error-correcting (BCH) code, cyclic searching or parity check searching are not usable for candidate codewords search; the overall complexity grows fast.

The class of parallel concatenated product codes (PCPCs) is an extension of the product codes (PCs). Its structure is similar to that of a classic turbo code [2], replacing the constituent convolutional codes in a turbo code by product codes. The code can use simple component codes to pertain high free distance without significant decoding complexity increase. [1] has studied the same structure applying single parity check code as linear product code.

We extend the idea of [1] and use extended Hamming codes as the component codes of the associated product codes. The availability of low-complexity soft output decoding algorithm for extended Hamming codes and flexible choices of code rates and codeword lengths make PCs and PCPCs attractive options for high speed wireless communication applications.

We use Fibonacci interleaver for PCPCs for it has a regular structure and renders simple implementation. It will be shown that this class of interleavers guarantees that the resulting PCPC has a minimum distance larger than that of its constituent product codes if a proper component codeword length is selected.

As the soft-output algorithm of [3] does not explicitly consider the a priori information, we modify the decoding algorithm using a new metric and derive a new extrinsic information formula. Hanzo *et al.* [6] used maximum a BCJR-based iterative decoding algorithm to decode PCs. However, as a PC has a girth of four only, the sum-product decoder does not offer near-optimal performance and is outperformed by Pyndiah's algorithm in high signal-to-noise ratio (SNR) region. We show that Pyndiah's list decoding approach undergoes a cooling process similar to that of annealed belief propagation (ABP) algorithms [7] and demonstrate that incorporating dynamic temperature(s) in the BCJR-based approach also result in improved performance.

The rest of this paper is organized as follows. Section 2 describes a modified Pyndiah-Chase algorithm and Hanzo's BCJR-based PC decoding algorithm, and an annealed version of the BCJR-based decoder. Performance comparison of all three TPC decoders is given as well. In Section 3, we present the encoder and decoder structures of PCPCs and the special interleaver used. We also describe the corresponding decoding schedule. A lower bound for the minimum distance of a PCPC using the Fibonacci interleaver is derived. The last section presents some simulated performance of various PCs and PCPCs.

II. Turbo product codes

A. Product codes

Consider the product code $\mathcal{C}^1 \otimes \mathcal{C}^2$ shown in Fig. 1, where $\mathcal{C}^1$ and $\mathcal{C}^2$ are (n_1, k_1, δ_1) and (n_2, k_2, δ_2) systematic linear block codes. The product code $\mathcal{P} = \mathcal{C}^1 \otimes \mathcal{C}^2$ has a length of $n = n_1 \times n_2$ bits with $k = k_1 \times k_2$ information bits and minimum Hamming distance $\delta = \delta_1 \times \delta_2$. Its code rate $\mathcal{R}$ is given by $\mathcal{R} = \mathcal{R}_1 \times \mathcal{R}_2$, where $\mathcal{R}_i = k_i/n_i$ is the code rate of

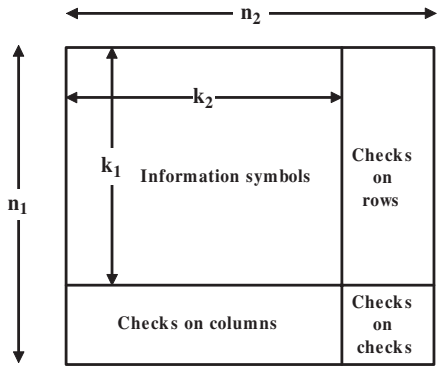

Fig. 1. A typical two-dimensional product code, $\mathcal{P} = \mathcal{C}^1 \otimes \mathcal{C}^2$.

the component code $\mathcal{C}^i$. The $k_1 \times k_2$ matrix within the product codeword is called the *information array* and the remaining entries form the codeword *parity part*.

B. A modified Pyndiah-Chase algorithm

There are a variety of soft-decision decoding algorithms for block codes. The Chase algorithm and its variations offer a good balance and tradeoff between complexity and performance. List-decoding often consists of two stages: (i) finding candidate codewords based on the received samples and (ii) generating soft output. As in a turbo decoder, the soft output is then passed to the ensuing decoding round as the extrinsic (a priori) information. For convenience of reference we summarize the Chase list decoding algorithm for a linear block code as follows.

1) Selecting candidate codewords: Suppose an (n, k, δ) linear block code $\mathcal{C}$ is BPSK-modulated and transmitted over an additive white Gaussian noise (AWGN) channel. $\boldsymbol{X} = (x_1, \cdots, x_l, \cdots, x_n)$ and $\boldsymbol{R} = (r_1, \cdots, r_l, \cdots, r_n)$ denote the transmitted codeword and received vector, where $x_l \in \{+1, -1\}$. Then $\boldsymbol{R} = \boldsymbol{X} + \boldsymbol{E}$, where the noise vector is $\boldsymbol{E} = (e_1, e_2, \cdots, e_n)$ in which e_i are i.i.d. zero-mean Gaussian random variables with variance σ^2. $\boldsymbol{A} = (a_1, \cdots, a_l, \cdots, a_n)$ denotes the a priori information of the codeword bits where

$$a_l = \ln \frac{\Pr\{c_l = +1\}}{\Pr\{c_l = -1\}}. \tag{1}$$

Following the spirit of Chase' list decoding approach, we propose the following three-step algorithm, which is a modified version of Pyndiah's algorithm [3].

A1. Use $\boldsymbol{R}$ and $\boldsymbol{A}$, if available, to obtain the hard decision vector $\boldsymbol{Y} = (y_1, y_2, \cdots, y_n)$ as well as their reliability (extrinsic information). Determine the $p = \lfloor \delta/2 \rfloor$ positions associated with the least reliable binary elements of $\boldsymbol{Y}$, where the reliability of y_j is given by $\Lambda(x_j)$ and is related to the log-likelihood ratio (LLR) via

$$\Lambda(x_j) = \ln \frac{\Pr\{r_j | x_j = +1\}}{\Pr\{r_j | x_j = -1\}} + a_j = \frac{2}{\sigma^2} r_j + a_j \tag{2}$$

A2. Bit-flipping the most unreliable p positions on $\boldsymbol{Y}$ to form the set of 2^p test patterns $\boldsymbol{T}^q$ $(0 \le q \le 2^p - 1)$.

A3. Form test sequence $\boldsymbol{Z}^q$ where $z_l^q = y_l \oplus t_l^q$ and decode $\boldsymbol{Z}^q$ using an algebraic (or hard) decoder. Denoted by Ω the set of all decoded codewords, decision $\boldsymbol{D} = (d_1, d_2, \cdots, d_n)$ of a row (or column) of the product code is then obtained by applying decision rule:
$\boldsymbol{D} \in \Omega$ is a local maximum likelihood codeword if

$$|\boldsymbol{R} - \boldsymbol{D}|^2 \le |\boldsymbol{R} - \boldsymbol{C}^i|^2 \ \forall \ \boldsymbol{C}^i \in \Omega. \tag{3}$$

where $\mathbf{C}^i = (c_1^i \ c_2^1 \ \cdots \ c_n^i)$ and

$$|\boldsymbol{R} - \boldsymbol{C}^i|^2 = \sum_{l=1}^{n} \frac{(r_l - c_l^i)^2}{2\sigma^2} - \frac{c_l^i}{2} a_l \tag{4}$$

is the metric between $\boldsymbol{R}$ and $\boldsymbol{C}^i$.

2) Soft output generation: The reliability of decision d_j about the transmitted symbol x_j, given the observation $\boldsymbol{R}$, is

$$\Lambda'(x_j) = \ln \left(\frac{\Pr\{x_j = +1 | \boldsymbol{R}\}}{\Pr\{x_j = -1 | \boldsymbol{R}\}} \right) \tag{5}$$

Let $S_j^{+1} \subset \boldsymbol{C}$ be the set of codewords whose jth coordinate c_j^i is $+1$ and $S_j^{-1} \subset \boldsymbol{C}$ be the set of codewords with -1 in their jth coordinate. Then we have

$$\Pr\{x_j = +1 | \boldsymbol{R}\} = \sum_{\mathbf{C}^i \in S_j^{+1}} \Pr\left\{\mathbf{X} = \mathbf{C}^i | \mathbf{R}\right\} \tag{6}$$

$$\Pr\{x_j = -1 | \mathbf{R}\} = \sum_{\mathbf{C}^i \in S_j^{-1}} \Pr\left\{\mathbf{X} = \mathbf{C}^i | \mathbf{R}\right\}, \tag{7}$$

and (5) becomes

$$\begin{aligned} \Lambda'(x_j) \quad &= \ln \left(\frac{\sum\limits_{\mathbf{C}^i \in S_j^{+1}} \mathrm{p}\{\mathbf{X}=\mathbf{C}^i|\mathbf{R}\}}{\sum\limits_{\mathbf{C}^i \in S_j^i} \mathrm{p}\{\mathbf{X}=\mathbf{C}^i|\mathbf{R}\}} \right) \\ &\approx \ln \left(\frac{\sum\limits_{\mathbf{C}^i \in S_j^{+1} \cap \Omega} \mathrm{p}\{\mathbf{X}=\mathbf{C}^i|\mathbf{R}\}}{\sum\limits_{\mathbf{C}^i \in S_j^{-1} \cap \Omega} \mathrm{p}\{\mathbf{X}=\mathbf{C}^i|\mathbf{R}\}} \right) \\ &\approx \ln \left(\frac{\max\limits_{\mathbf{C}^i \in S_j^{+1} \cap \Omega} \mathrm{p}\{\mathbf{X}=\mathbf{C}^i|\mathbf{R}\}}{\max\limits_{\mathbf{C}^i \in S_j^{-1} \cap \Omega} \mathrm{p}\{\mathbf{X}=\mathbf{C}^i|\mathbf{R}\}} \right) \end{aligned} \tag{8}$$

At high SNRs, (8) can be approximated by

$$\Lambda''(x_j) = |\boldsymbol{R} - \boldsymbol{C}^{-1(j)}|^2 - |\boldsymbol{R} - \boldsymbol{C}^{+1(j)}|^2 \tag{9}$$

where $C^{+1(j)} \in S_j^{+1} \cap \Omega$ and $C^{-1(j)} \in S_j^{-1} \cap \Omega$ are the codewords with the minimum metric and one of $C^{+1(j)}$ and $C^{-1(j)}$ is $\boldsymbol{D}$. Substituting (4) into (9), we obtain

$$\Lambda''(x_j) = \frac{2r_j}{\sigma^2} + a_j + w_j \tag{10}$$

where the extrinsic information w_j is

$$w_j = \sum_{l=1, l \ne j}^{n} \left(\frac{2}{\sigma^2} r_l + a_l \right) c_l^{+1(j)} p_l. \tag{11}$$

if there is a competing codeword at position j, i.e., there exists codewords in Ω whose jth coordinate differs from d_j, and

$$p_l = \begin{cases} 0, & \text{if } c_l^{+1(j)} = c_l^{-1(j)} \\ 1, & \text{if } c_l^{+1(j)} \neq c_l^{-1(j)} \end{cases} . \quad (12)$$

We compute extrinsic information for a position without competing codeword by

$$w_j(\beta) = \beta \times \overline{w}, \ j \notin V, \quad (13)$$

where $0 < \beta < 1$ and

$$\overline{w} = \frac{1}{|V|} \sum_{j \in V} |w_j|. \quad (14)$$

$$V = \{ \ j \mid \exists \ \boldsymbol{C}^i \in \Omega \ s.t. \ c_j^i \neq d_j \} \quad (15)$$

An iterative decoder then use the extrinsic information $w_j^{(m)}$ obtained at the mth decoding round (DR) as the a priori information for the $(m+1)$th DR by (see also Fig. 4)

$$\begin{aligned} a_j^{(m+1)} &= w_j^{(m)}(\beta(m)) \\ \Lambda^{(m+1)}(x_j) &= \frac{2}{\sigma^2} r_j + \alpha(m) a_j^{(m+1)}, \end{aligned} \quad (16)$$

$0 < \alpha(m) \leq 1$ is a weighting factor which is often chosen as a monotonic increasing function of m.

C. BCJR-based TPC decoding algorithm

Hanzo *et al.* [6] suggested an iterative decoding algorithm for PCs by using a modified BCJR algorithm to calculate extrinsic information. Incorporating the parity check bit in calculating the LLR of a decoded bit (see Fig 2(a)), they obtain the a-posteriori LLR

$$\Lambda'(x_j) = \ln \left[\frac{\sum_{(\acute{s},s) \Rightarrow x_j = +1} \gamma_k^{(+1)}(\acute{s}, s) \cdot [(\alpha^e \beta^e + \alpha^o \beta^o) \cdot}{\sum_{(\acute{s},s) \Rightarrow x_j = -1} \gamma_k^{(-1)}(\acute{s}, s) \cdot [(\alpha^e \beta^o + \alpha^o \beta^e) \cdot} \right.$$
$$\left. \frac{P(r_n | x_n = +1) + (\alpha^e \beta^o + \alpha^o \beta^e) \cdot P(r_n | x_n = -1)]}{P(r_n | x_n = +1) + (\alpha^e \beta^e + \alpha^o \beta^o) \cdot P(r_n | x_n = -1)]} \right] \quad (17)$$

where the superscript $e(o)$ indicates the path arriving at this state gives an even (odd) number of $+1$ bits. Note that the forward and backward conditional probabilities $\alpha_k(s)$ and $\beta_k(s)$ are separated into two groups; see Fig. 2 (b).

1) Annealed BCJR based TPC decoding algorithm: Our simulation concludes that the bit error rate (BER) performance of the BCJR-based approach is worse than that of the Pyndiah-Chase algorithm in high SNR region, although it is known that the BCJR algorithm, being an MAP decoding rule, gives the best BER performance when decoding the component block codes. This is due to the fact that the BCJR-based PC decoder follows the conventional BP schedule which yields inferior performance if the corresponding factor graph has short cycles. To remedy this shortcoming, we borrow the concept of the recently developed ABP algorithm [7].

The ABP approach incorporates a dynamic temperature into the free energy formulation so that message passing is performed on a dynamic surface of energy cost. A proper cooling process is in place to randomize the BP schedule over an ensemble of codes and to help BP converge to a stable fixed point with lower energy value, which gives more accurate global minimum estimations.

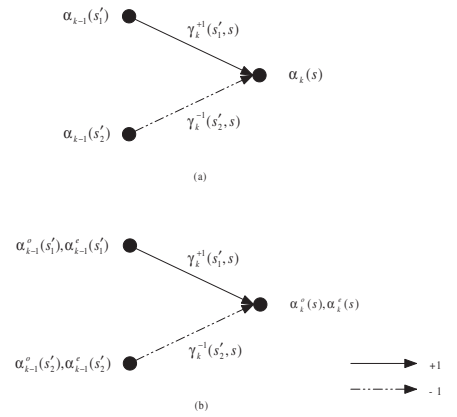

Fig. 2. (a) Forward recursion of the BCJR algorithm,(b) Forward recursion of modified BCJR (MAP) algorithm. $\acute{s}_1$ and $\acute{s}_2$ are particular states of $\acute{s}$, where $\acute{s}$ and s represent states in time $k-1$ and k, respectively.

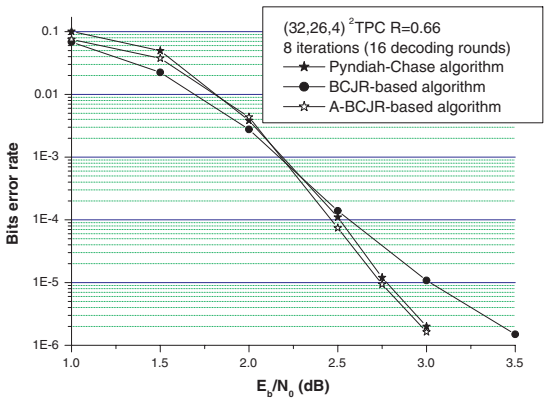

Fig. 3. Performance of Pyndiah-Chase algorithm, BCJR based algorithm, and Annealed BCJR (A-BCJR) based algorithm.

Following the concept of the ABP algorithms, we replace the *branch metric* of the BCJR algorithm in log-domain (denoted by $*$) by

$$\begin{aligned} \gamma^*_{k,\frac{1}{T_1},\frac{1}{T_2}}(\acute{s}, s) &\equiv \left(\frac{1}{T_1}\right) \frac{x_k L_a(x_k)}{2} + \left(\frac{1}{T_2}\right) \frac{L_c}{2}(r_k x_k) \\ &= \gamma_{m_{\mu k,\frac{1}{T_1}}} + \gamma_k^{\frac{1}{T_2}} \end{aligned} \quad (18)$$

where $L_c = 4E_s/N_0$, $L_a(x)$ stands for the extrinsic information of x, and T_1, T_2 are the temperature parameters such that $0 < 1/T_1, 1/T_2 < 1$. $\gamma^*_{k,\frac{1}{T_1},\frac{1}{T_2}}(\acute{s}, s)$ degenerated to the conventional transition probability $\gamma_k^*(\acute{s}, s) = x_k La(x_k)/2 + L_c(r_k x_k)/2$ when $T_1 = T_2 = 1$.

Note that the parameters $0 < \alpha(m), \beta(m) < 1$ used in (16), (17) play the same roles as $1/T_1$ and $1/T_2$ do in the annealed BCJR algorithm for cooling the associated cost function. Fig. 3 reveals that both A-BCJR based algorithm and Pyndiah-Chase algorithm outperform the BCJR-based algorithm by 0.5 dB at BER $= 2 * 10^{-6}$. The performance of the A-BCJR-based algorithm can be further improved by fine-tuning the temperatures.

III. Parallel Concatenated Product Codes

A. Encoder

The structure of parallel concatenated product code (PCPC) is similar to that of the classic turbo code in which the component recursive systematic convolutional encoders are replaced by PCs. shown in Fig. 1. A codeword consists of the PC codeword of the upper branch and the parity part of the lower branch output. The only interleaver considered here will be the Fibonacci interleaver.

B. Decoder

The decoder consists of four a posteriori probability (APP) component decoders, each is responsible for decoding a component block code. Fig. 4 depicts the structure of a component decoder which is similar to that presented in [3]. Denote by $[\boldsymbol{R}]$ the receiving matrix and by $[\boldsymbol{W}_t(m)]$ the output extrinsic information matrix of the tth APP decoder at the end of the mth APP decoding round. The a priori information matrix given at input of the tth APP decoder is

$$\boldsymbol{W}_{sum}(m) = \sum_{l \neq t} \boldsymbol{W}_l(m). \tag{19}$$

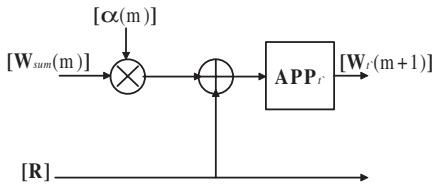

Fig. 4. Block diagram of elementary APP decoder $\boldsymbol{APP}_{t^i}$.

The message passing operations among these four APP decoders is shown in Fig. 5, where $\boldsymbol{I}$ and $\boldsymbol{P}$ are extrinsic information corresponding to information bits and parity check bits, respectively. A complete iteration follows the APP decoding schedule: $\boldsymbol{APP}_0 \rightarrow \boldsymbol{APP}_1 \rightarrow \boldsymbol{APP}_2 \rightarrow \boldsymbol{APP}_3$. Each APP decoder also sends related extrinsic information to the other two decoders that do not sit next to it in the schedule. Decoding of a component TPC usually converges after 8 APP decoding rounds but a PCPC may takes an average of 16 APP decoding rounds to converge.

Similar to [3], the choice of the *weighting factor* α and *reliability factor* β is very critical in determining the performance. The sets of α and β we used for different decoding rounds are given by

$$\alpha(m) = [0.0, 0.1, 0.2, 0.3, 0.4, 0.6, 0.8, 1.0]. \tag{20}$$

$$\beta(m) = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.8, 1.0]. \tag{21}$$

and

$$\alpha(m) = [0.0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.7, 0.8, 1.0]. \tag{22}$$

$$\beta(m) = [0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.8, 0.9, 1.0]. \tag{23}$$

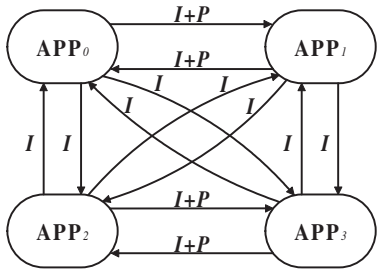

Fig. 5. An flow chart showing the message-passing of various component APP decoders and related decoding schedule.

C. Fibonacci interleaver

The position of each bit in the $(k_1 \times k_2)$ information array is represented by $(\bar{i}, \bar{j})$, where $0 \leq \bar{i} < k_1$ and $0 \leq \bar{j} < k_2$. Using the definition $|W|_Z = W \pmod{Z}$, the Fibonacci interleaver is characterized by the permutation rule

$$(\bar{i}, \bar{j}) \rightarrow (|\bar{i} + \bar{j}|_{k_1}, ||\bar{i} + \bar{j}|_{k_1} + \bar{j}|_{k_2}). \tag{24}$$

For a PC with extended Hamming codes as component codes, a minimum-weight codeword has nonzero entries at the same locations of some four nonzero-weight rows and columns within the information array, i.e., the parity part of the product codeword has weight 0. The basic requirement of the interleaver used in a PCPC is to make sure the interleaved information array will not contain such a 4×4 all-1 subarray. The Fibonacci interleaver does possess such a desired property. In particular, one can show that

Lemma 1: A PCPC using $(n_1, k_1, 4) \times (n_2, k_2, 4)$ product codes as constituent codes has a minimum Hamming distance greater than 16 if k_1 or k_2 is not a multiple of 4.

Proof: Since each component block code of a constituent PC has minimum distance 4, we use a 4×4 matrix called minimum-weight error event array to represent the nonzero positions of a minimum weight product codeword. As it is possible that all entries of this matrix lie within the information array of a product codeword, we want the interleaver to guarantee that the permuted information array will not produce an all-zero parity part.

Without loss of generosity, let the nonzero positions be located at the four columns $(\bar{j}, \bar{j} + g_1, \bar{j} + g_2, \bar{j} + g_3)$ and four rows $(\bar{i}, \bar{i} + h_1, \bar{i} + h_2, \bar{i} + h_3)$, where $0 < h_1 < h_2 < h_3 < k_1$

and $0 < g_1 < g_2 < g_3 < k_2$. We have the error-position array $[\gamma_{w,z}]_{4\times 4}$

$$\begin{bmatrix} (\bar{i},\bar{j}) & \cdots & \cdots & (\bar{i},\bar{j}+g_3) \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ (\bar{i}+h_3,\bar{j}) & \cdots & \cdots & (\bar{i}+h_3,\bar{j}+g_3) \end{bmatrix} \quad (25)$$

where $1 \le w,z \le 4$, $\gamma_{w,z} = (\bar{i}+u_w, \bar{j}+v_z)$, $\mathbf{u} = (u_1,u_2,u_3,u_4) = (0,h_1,h_2,h_3)$ and $\mathbf{v} = (v_1,v_2,v_3,v_4) = (0,g_1,g_2,g_3)$.

After Fibonacci interleaving, the positions will be permuted to

$$\tilde{\gamma}_{w,z} = (|\bar{i}+u_w+\bar{j}+v_z|_{k_1}, ||\bar{i}+u_w+\bar{j}+v_z|_{k_1}+\bar{j}+v_z|_{k_2}).$$

which can be decomposed as

$$\begin{aligned} \tilde{\gamma}_{w,z} &= (|\bar{i}+\bar{j}|_{k_1}, ||\bar{i}+\bar{j}|_{k_1}+\bar{j}|_{k_2}) \\ &\quad \bigoplus_{k_1\times k_2} (|u_w+v_z|_{k_1}, ||u_w+v_z|_{k_1}+v_z|_{k_2}), \\ &= \dot{\gamma}_{w,z} \bigoplus_{k_1\times k_2} \ddot{\gamma}_{w,z}. \end{aligned}$$

where $\bigoplus_{k_1\times k_2}$ represents the operation that takes module k_1 and k_2 on the first and the second entries, respectively. Because all entries of the array $[\dot{\gamma}_{w,z}]_{4\times 4}$ are all the same, $[\tilde{\gamma}_{w,z}]_{4\times 4}$ does not form a matrix with all entries forming a 4 by 4 position array only if $[\ddot{\gamma}_{w,z}]_{4\times 4}$ does not. Therefore, we only need to consider an error-position array

$$\begin{aligned} [\ddot{\gamma}_{w,z}]_{4\times 4} &= [|u_w+v_z|_{k_1}, ||u_w+v_z|_{k_1}+v_z|_{k_2}]_{4\times 4}, \\ &= [\hat{\gamma}_{w,z}, \check{\gamma}_{w,z}]_{4\times 4}. \end{aligned}$$

where $\hat{\gamma}_{w,z}$ and $\check{\gamma}_{w,z}$ represents the rows and columns of this position array, respectively. As we know, the array $[\ddot{\gamma}_{w,z}]_{4\times 4}$ will form a 4×4 position array without generating parity bits only if there are four distinct values in rows or columns. Therefore, we firstly consider an array of the form

$$[\hat{\gamma}_{w,z}]_{4\times 4} = \begin{bmatrix} |0|_{k_1} & |g_1|_{k_1} & |g_2|_{k_1} & |g_3|_{k_1} \\ |h_1|_{k_1} & |h_1+g_1|_{k_1} & |h_1+g_2|_{k_1} & |h_1+g_3|_{k_1} \\ |h_2|_{k_1} & |h_2+g_1|_{k_1} & |h_2+g_2|_{k_1} & |h_2+g_3|_{k_1} \\ |h_3|_{k_1} & |h_3+g_1|_{k_1} & |h_3+g_2|_{k_1} & |h_3+g_3|_{k_1} \end{bmatrix}.$$

If these 16 positions are permuted to the same four rows $0, g_1, g_2, g_3$, then h_1, h_2, h_3 should be moved to g_1, g_2, g_3. If h_1 is moved to g_2 or g_3, either h_2 or h_3 should be mapped to g_1, which is obviously a contradiction. Therefore, we have $h_1 = g_1$. If h_2 is mapped to g_3 then h_3 should be mapped to g_2, which is not possible, therefore, $h_2 = g_2$ and $h_3 = g_3$ and the resulting array becomes

$$[\hat{\gamma}_{w,z}]_{4\times 4} = \begin{bmatrix} |0|_{k_1} & |g_1|_{k_1} & |g_2|_{k_1} & |g_3|_{k_1} \\ |g_1|_{k_1} & |2g_1|_{k_1} & |g_1+g_2|_{k_1} & |g_1+g_3|_{k_1} \\ |g_2|_{k_1} & |g_2+g_1|_{k_1} & |2g_2|_{k_1} & |g_2+g_3|_{k_1} \\ |g_3|_{k_1} & |g_3+g_1|_{k_1} & |g_3+g_2|_{k_1} & |2g_3|_{k_1} \end{bmatrix}.$$

If $\hat{\gamma}_{w,w} \neq 0$ for $w = 2,3,4$, then there are three entries $\hat{\gamma}_{w,z} = 0,\ w \neq z,\ 0 < w,z \le 4$, which again is a contradiction. Therefore, at least one of $\hat{\gamma}_{w,w}$ is 0 for $w = 2,3,4$. We now proceed to discuss these three cases.

1) $|2g_1|_{k_1} = 0 : g_1 = \frac{k_1}{2}$ implies $g_1+g_2 \neq 0$, $g_1+g_3 \neq 0$. Thus g_2+g_3 must be equal to 0 but then we have $k_1 > g_3 > g_2 > g_1 = \frac{k_1}{2}$, a contradiction.
2) $|g_2|_{k_1} = 0 : g_2 = \frac{k_1}{2}$ implies $g_2+g_3 \neq 0$, $g_1+g_2 \neq 0$ and $g_1+g_3 = 0$. Therefore $\mathbf{v} = (0, g_1, \frac{k_1}{2}, k_1-g_1)$ and $[\hat{\gamma}_{w,z}]_{4\times 4}$ becomes

$$\begin{bmatrix} 0 & g_1 & \frac{k_1}{2} & k_1-g_1 \\ g_1 & 2g_1 & g_1+\frac{k_1}{2} & 0 \\ \frac{k_1}{2} & g_1+\frac{k_1}{2} & 0 & \frac{k_1}{2}-g_1 \\ k_1-g_1 & 0 & \frac{k_1}{2}-g_1 & -2g_1 \end{bmatrix}. \quad (26)$$

   Because there should be four entries equal to "$\frac{k_1}{2}$", $2g_1$ and $-2g_1$ are equal to "$\frac{k_1}{2}$" and $g_1 = \frac{k_1}{4}$, hence $\mathbf{v} = (0, \frac{k_1}{4}, \frac{k_1}{2}, \frac{3k_1}{4})$.
3) $|2g_3|_{k_1} = 0 : g_3 = \frac{k_1}{2}$ implies $g_1+g_3 \neq 0$, $g_2+g_3 \neq 0$, which forces g_1+g_2 to be equal to 0 but then we have the contradictory result, $0 < g_1 < g_2 < g_3 = \frac{k_1}{2}$.

Next let us consider the sixteen values of the columns error position array $[\check{\gamma}_{w,z}]_{4\times 4} = [||v_w+v_z|_{k_1}+v_z|_{k_2}]_{4\times 4}$. By substituting $\mathbf{v} = (0, \frac{k_1}{4}, \frac{k_1}{2}, \frac{3k_1}{4})$ into the array, we have

$$\begin{bmatrix} 0 & |\frac{k_1}{2}|_{k_2} & |k_1|_{k_2} & |\frac{3k_1}{2}|_{k_2} \\ |\frac{k_1}{4}|_{k_2} & |\frac{3k_1}{4}|_{k_2} & |\frac{5k_1}{4}|_{k_2} & |\frac{3k_1}{4}|_{k_2} \\ |\frac{k_1}{2}|_{k_2} & |k_1|_{k_2} & |\frac{k_1}{2}|_{k_2} & |k_1|_{k_2} \\ |\frac{3k_1}{4}|_{k_2} & |\frac{k_1}{4}|_{k_2} & |\frac{3k_1}{4}|_{k_2} & |\frac{5k_1}{4}|_{k_2} \end{bmatrix}. \quad (27)$$

There are four "$|\frac{3k_1}{4}|_{k_2}$", three "$|k_1|_{k_2}$", three "$|\frac{k_1}{2}|_{k_2}$", two "$|\frac{1k_1}{4}|_{k_2}$", two "$|\frac{5k_1}{4}|_{k_2}$", one "$|\frac{6k_1}{4}|_{k_2}$" and one "0" in these entries. As only four distinct values are allowed for these entries and $|\frac{1k_1}{4}|_{k_2}$, $|\frac{5k_1}{4}|_{k_2}$ and $|\frac{6k_1}{4}|_{k_2}$ can not be "0", we need only to consider the remaining possibilities.

1) $|k_1|_{k_2} = 0 : k_1 = l \times k_2, l \in \mathbf{Z}$. $0, |\frac{lk_2}{4}|_{k_2}, |\frac{lk_2}{2}|_{k_2}, |\frac{3lk_2}{4}|_{k_2}$ should be different, which immediately implies that l is odd.
2) $|\frac{k_1}{2}|_{k_2} = 0 : k_1 = 2l \times k_2, l \in \mathbf{Z}$. This condition leads to $|\frac{k_1}{4}|_{k_2} = |\frac{3k_1}{4}|_{k_2}$ a contradiction.

Since neither k_1 nor k_2 is a multiple of 4, there is no weight-16 PCPC codeword. ■

Employing an analogous argument we can also prove

Lemma 2: A PCPC using $(n_1,k_1,4)\times(n_2,k_2,4)$ product codes as constituent codes cannot have a minimum Hamming distance equals to 17, 18 or 19.

Lemmas 1 and *2* then give us

Theorem 1: A PCPC using $(n_1,k_1,4)\times(n_2,k_2,4)$ product codes as constituent codes has a minimum Hamming distance no less than 20 if k_1 or k_2 is not a multiple of 4.

We apply an extra interleaver to enlarge free distance but there exists at least one weight 31 codeword in PCPC for all kinds of interleaver. We have done computer search on a $(8,4,4)^4$ PCPC and the resultant distance has codeword weight 28. By the way, the number of the searched codeword

is 5. The upper and the searched codeword weights have a difference by 3. It implies that Fibonacci interleaver renders large distance property with small number codeword of neighbor for PCPC although we do not provide a complete prove up to codeword weight 28.

IV. Simulation results and discussion

The performance of two PCPCs and two PCs is presented in this section. The first PCPC uses $(32, 26, 4)$ extended Hamming code as its PC component code and is denoted by $(32, 26, 4)^4$, resulting in a code rate of $26^2/[2(32^2) - 26^2] = .493$. The second one, denoted by $(64, 57, 4)^2 \times (16, 11, 4)^2$, uses $(64, 57, 4)$ and $(16, 11, 4)$ extended Hamming codes and has a code rate of $(57 \times 11)/[2(64 \times 16) - (57 \times 11)] = .441$. We also examine the performance of two PCs. The first PC, denoted by $(16, 11, 4)^2$, uses $(16, 11, 4)$ extended Hamming code as its component code so that the code rate becomes .473. The second PC, $(32, 26, 4)^2$, uses the $(32, 26, 4)$ extended Hamming code and has a code rate of .66.

A decoding round (DR) is defined as one-pass decoding of a product code $\boldsymbol{APP}_i \rightarrow \boldsymbol{APP}_{i+1}$. A 4-iteration PC decoder such as that given in [3] thus needs 8 decoding rounds. PCPC requires more iterations. As expected, more DRs lead to improved performance as is shown in Fig. 6 where the performance of $(32, 26, 4)^4$ is given.

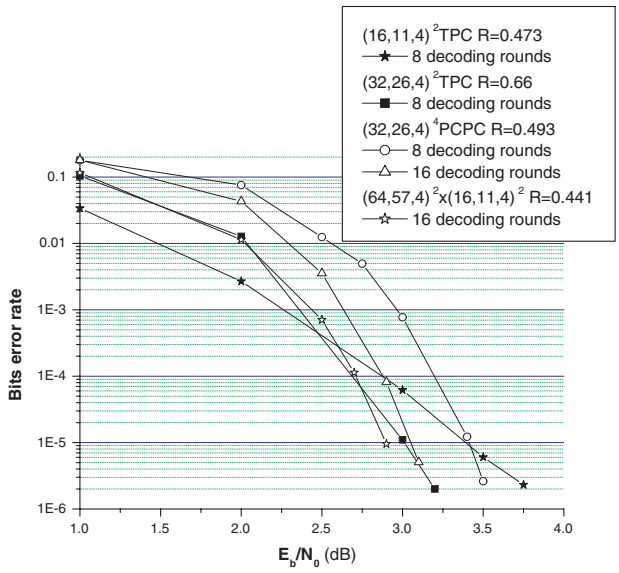

Fig. 6. BER performance of various PCs and PCPCs.

We compare BER performance of two PCPCs in Fig. 6. The one with the lower rate yields better performance. Performance of two PCs are shown in the same figure. The two PCs provide performance superior to PCPCs at lower SNR region while PCPCs begin to show their advantage for BER $< 10^{-5}$. Due to larger free distances, PCPCs outperform PCs at high SNR. Although the performance is not entirely fair as far as code rate and codeword length are concerned, these performance curves do serve to demonstrate the influence of a code's distance spectrum. It also indicates that there is much room left for improving the performance of PCPCs.

Fig. 7 compares the performance of the A-BCJR based algorithm and the conventional BCJR-based algorithm. The former gives a 0.2 dB improvement at BER$\approx 2 * 10^{-5}$. The effectiveness of the cooling mechanism during the turbo decoding process seems not as impressive as that for PCs. Both Figs. 6 and 7 show that the proposed decoding algorithms are not capable of rendering ML performance of a PCPC that has a relatively large free distance. Better performance can be obtained if one optimizes the temperature parameters and the decoding schedule of the A-BCJR algorithm. It is clear, however, a better decoding algorithm for codes with short cycles in general is still lacking.

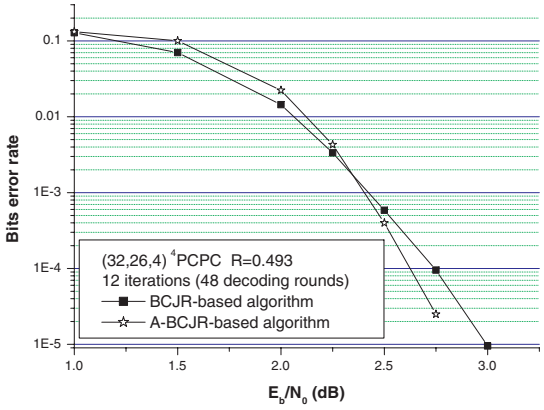

Fig. 7. Bit error rate performance of A PCPC using BCJR-based and A-BCJR based algorithms.

References

[1] D.M. Rankin and T.A. Gulliver, "Serial and parallel concatenation of SPC product codes," *Proc. IEEE Int. Symp. on Inf. Theory and Applications*, Honolulu, USA, pp 778-781, Nov. 2000.

[2] C. Berrou, A. Glavieux, "Near optimum error correcting coding and decoding: Turbo code," *IEEE Trans. Commun.*,vol. 44, pp. 1261-1271, Oct. 1996.

[3] Pyndiah, R.M, "Near-Optimum Decoding of Product Codes: Block Turbo codes," *IEEE Trans. Commun.*, vol. 46, no. 8, pp. 1003-1010, Aug. 1998.

[4] H. Burton, E. Weldon, Jr, "Cyclic product codes," *IEEE Trans. Inform. Theory,* vol. 11, pp. 433-439, Jul. 1965.

[5] D. Chase, "A class of algorithm for decoding block codes with channel measurement information," *IEEE Trans. Inform. Theory*, vol. 18, pp. 170-182, Jan. 1972.

[6] L. Hanzo, T.H. Liew, B.L. Yeap *Turbo Coding, Turbo Equalisation and Space-Time coding*, pp. 232-239, Jan. 1972.

[7] Yen-Chih Chen, Yu T. Su "Constraint Relaxation and Annealed Belief Propagation for Binary Networks," *submitted ISIT2007*, Jan. 2007.

[8] J.-S. Liao, Y. T. Su and S.-C. Chen, "Type II ARQ schemes based on turbo product codes," in Proc. IEEE GlobeCom 2002, vol. 1, pp. 846-850, Nov. 2002.

[9] M. Janani, A. Hedayat, T.E. Hunter, A. Nosratinia, "Coded cooperation in wireless communications: space-time transmission and iterative decoding," *Signal Processing, IEEE Trans. on*, vol. 52, no. 2, pp. 362-371, Feb. 2004 .