Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2012, Article ID 609650, 8 pages
doi:10.1155/2012/609650

Research Article

Reduced Complexity Iterative Decoding of 3D-Product Block Codes Based on Genetic Algorithms

Abdeslam Ahmadi,[1] Faissal El Bouanani,[2] Hussain Ben-Azza,[1] and Youssef Benghabrit[1]

[1] *Department of Industrial and Production Engineering, Moulay Ismail University, Ecole Nationale Supérieure d'Arts et Métiers, Meknès 50000, Morocco*
[2] *Department of Communication Networks, Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes, Rabat 10000, Morocco*

Correspondence should be addressed to Abdeslam Ahmadi, ab_ahmadi@hotmail.com

Received 15 September 2011; Revised 21 December 2011; Accepted 8 February 2012

Academic Editor: Lisimachos Kondi

Two iterative decoding algorithms of 3D-product block codes (*3D-PBC*) based on genetic algorithms (*GAs*) are presented. The first algorithm uses the Chase-Pyndiah *SISO*, and the second one uses the list-based *SISO* decoding algorithm (LBDA) based on order-*i* reprocessing. We applied these algorithms over *AWGN* channel to symmetric *3D-PBC* constructed from *BCH* codes. The simulation results show that the first algorithm outperforms the Chase-Pyndiah one and is only 1.38 dB away from the Shannon capacity limit at BER of 10^{-5} for *BCH* $(31, 21, 5)^3$ and 1.4 dB for *BCH* $(16, 11, 4)^3$. The simulations of the LBDA-based *GA* on the *BCH* $(16, 11, 4)^3$ show that its performances outperform the first algorithm and is about 1.33 dB from the Shannon limit. Furthermore, these algorithms can be applied to any arbitrary 3D binary product block codes, without the need of a hard-in hard-out decoder. We show also that the two proposed decoders are less complex than both Chase-Pyndiah algorithm for codes with large correction capacity and LBDA for large *i* parameter. Those features make the decoders based on genetic algorithms efficient and attractive.

1. Introduction

Among the proposed codes in the history of error correcting, there are those who have performance very close to the Shannon limit, like Turbo codes [1] and LDPC codes [2]. Nevertheless, the remarkable reduction of BER is performed at the expense of their decoders complexity. The current challenge for researchers in this field is to find a compromise between performance and decoding complexity. Thus, several optimization works of decoding algorithms have emerged, in particular, those associated to product codes. These codes were first introduced in 1954 by Elias [3]. In 1981 and 1983, an iterative decoding method hard-in hard-out (*HIHO*) of these codes has been described, respectively by Tanner [4] and Lin and Costello [5]. In 1994, a soft-in soft-out (*SISO*) iterative decoding of the product block codes (*PBC*) was proposed by Pyndiah et al. [6], using the Chase algorithm as the elementary decoder [7]. This algorithm does not work alone, but together with another decoder *HIHO* which is not always easy to find for some codes, like quadratic residue (*QR*). Later, in 2004, an enhanced *SISO* iterative decoding algorithm of *PBC*, based on order reprocessing decoding, was developed by Martin et al. [8].

Recently, the researchers in the field of channel coding were inspired from artificial intelligence techniques to develop very good decoders for linear block codes. We quote from the first works in this sense, the decoding of linear block codes using algorithm A* [9], genetic algorithms [10], and neural networks [11].

We were interested in this work in decoders based on genetic algorithms (GAD) [10] applied to the 3D-product block code (*3D-PBC*). It was shown in [12], that these decoders applied to *BCH* codes outperform the Chase-2 algorithm and present a lower complexity for *BCH* codes with large block lengths. We note that their performances can

be improved further by optimizing some parameters such as the population size and the number of generations.

In this paper, which is the continuation of the work [13], we introduce and study two iterative decoding algorithms of an arbitrary 3D binary product block code based on GAD. The extrinsic information is computed in the first proposed algorithm according to the Chase-Pyndiah formulas [6] and is computed in the second one according to the list-based *SISO* decoding algorithm (LBDA) [8]. A comparison at the level of complexity of the proposed algorithms versus Chase-Pyndiah and LBDA algorithms was made.

This paper is organized as follows. Section 2 defines the *3D-PBC* code. Then, we explain in Section 3, the elementary decoding based on GAD. The presentation and complexity study of our iterative decoding algorithms using genetic algorithms IGAD, will be given in Section 4. Section 5 illustrates, through simulations, the IGAD performances and the effect of some parameters on these performances. It also presents a comparison of performances between the two proposed algorithms. Finally, Section 6 presents the conclusion and indicates how the performances of our decoders can be improved further.

2. 3D-Product Block Code (*3D-PBC*)

The product codes (or iterative codes) are a particular case of serial concatenated codes. They allow to construct codes of great length by concatenating two or more arbitrary block codes with short lengths. In our case, we considered two *symmetric 3D-PBC*, $(16, 11, 4)^3$ and $(31, 21, 5)^3$, which consists of three identical codes *BCH* each one.

Let $C^{(1)}(n_1, k_1, d_1)$, $C^{(2)}(n_2, k_2, d_2)$, and $C^{(3)}(n_3, k_3, d_3)$, three linear block codes. We encode an information block, using $3D\text{-}PBC = C^{(1)} \otimes C^{(2)} \otimes C^{(3)}$ given in the Figure 1, by

(1) filling a cube of k_2 rows, k_1 columns and k_3 as the depth by $k_1 \times k_2 \times k_3$ information bits;

(2) coding the $k_2 \times k_3$ rows (the cube contains k_3 lateral plans which are composed from k_2 rows each one) using code $C^{(1)}$. The check bits are placed at the right, and we obtain a new cube with $k_2 \times k_3 \times n_1$ bits;

(3) coding the $n_1 \times k_3$ columns of the cube obtained in the previous step using code $C^{(2)}$. This means that the check bits will be also encoded (the previous cube contains n_1 transverse plans which are composed from k_3 columns each one). The check bits are placed at the bottom of the cube obtained in step 2, and we get a new cube with $n_1 \times k_3 \times n_2$ bits;

(4) Coding, finally the obtained cube in step 3 from the front to the behind, that is, coding the $n_1 \times n_2$ columns, using code $C^{(3)}$ (the previous cube consists of n_2 horizontal plans which contains n_1 columns). The check bits are placed at the behind. So, the last cube which has $n_1 \times n_2 \times n_3$ bits is the codeword.

We can show by similar reasoning in [14] that the parameters of the *3D-PBC* are

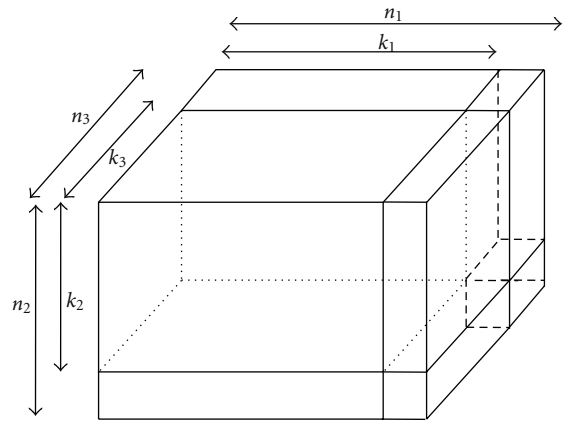

Figure 1: The 3D-product block code.

(i) length: $n = n_1 \times n_2 \times n_3$;

(ii) dimension: $k = k_1 \times k_2 \times k_3$:

(iii) minimum Hamming distance; $d = d_1 \times d_2 \times d_3$.

(iv) rate: $R = R_1 \times R_2 \times R_3 = k_1/n_1 \times k_2/n_2 \times k_3/n_3$.

This shows one of the best advantages of product block codes: building very long block codes with large minimum Hamming distance by concatenating short codes with small minimum Hamming distance.

3. Elementary Decoding of Linear Codes

Let $R = (R_1, \ldots, R_n)$ be the received sequence at the decoder input of a binary linear block code $C(n, k, d)$ with a generator matrix G.

3.1. Hard-Input Soft-Output Decoder

Step 1. Sort the elements of received vector R in descending order of magnitude. This will put reliable elements in the first ranks, since using an AWGN channel. Then, the vector is permuted such that its first k coordinates are linearly independent. We obtain a vector $R' = \pi(R) = (R'_1, \ldots, R'_n)$ such that $|R'_1| \geq |R'_2| \geq \cdots \geq |R'_n|$. Let G' be the permutation of G by π, that is, $G' = \pi(G)$.

Step 2. Quantize the first k bits of R' to obtain vector r and randomly generate $(N_i - 1)$ information vectors of k bits each one. This vectors form with vector r the initial population of N_i individuals $(I_1, \ldots, I_{N_i})$.

Step 3. Encode individuals of the current population, using G' to obtain codewords: $C_i = G' \cdot I_i (1 \leq i \leq N_i)$. Then, compute individuals fitness, defined as Euclidian distance between C_i and R'. Sort individuals in ascending order of fitness.

Step 4. Place the first N_e individuals (N_e: elite number $\leq N_i$) to the next population, which will be completed by offsprings

generated using reproduction operators: selection of two best individuals as parents (a, b) using the following linear ranking:

$$W_i = \frac{W_{\max} - 2(i-1)(W_{\max} - 1)}{N_i - 1}, \quad \forall i \in \{1, \ldots, N_i\}, \tag{1}$$

where W_i is the ith individual weight, and $W_{\max}$ weight is assigned to the fittest (nearest) individual.

Reproduce the $(N_e + 1)$ remaining individuals of the next population by crossover and mutation operations. Let p_c, p_m and Rand be respectively, probabilities of crossover and mutation, and a uniformly random value between 0 and 1, generated at each time.

if Rand $< p_c$, *then* for all $\in \{N_e + 1, \ldots, N_i\}, j \in \{1, \ldots, k\}$:

$$I_{ij} = \begin{cases} a_j & \text{if Rand} < \left(1 - a_j + a_j b_j\right) + \dfrac{a_j - b_j}{1 + e^{-4R'_j/N_0}} \\ b_j & \text{else,} \end{cases} \tag{2}$$

and then,

$$I_{ij} = 1 - I_{ij} \quad \text{if Rand} < p_m, \tag{3}$$

else

$$I_i = \begin{cases} a & \text{if Rand} < 0.5 \\ b & \text{else,} \end{cases} \tag{4}$$

end if

Repeat steps 3 and 4 for N_g generations.

Step 5. The first (fittest) individual D' of the last generation is the nearest to R'. So, the decided codeword is $D = \pi^{-1}(D')$.

3.2. Soft-Input Soft-Output Decoder. In this section, we present the SO_GAD decoders (soft-output GAD) used as the elementary decoder in our iterative decoding algorithms.

Let D denote the GAD decision of the input sequence R and w the extrinsic information.

Let $H^{(j)}$ be the competitor codeword of D corresponding to the jth bit defined by

$$\left\|H^{(j)} - R\right\| = \min_{2 \le p \le N_i} \left\{ \left\|Q^{(p)} - R\right\|, Q_j^{(p)} \neq D_j \right\}, \tag{5}$$

where $Q^{(p)}$ is the pth codeword of the last generation, $Q_j^{(p)}$ and D_j are the jth bits of $Q^{(p)}$, D and, $\|.\|$ is the Euclidean distance.

Algorithm 1. $(w, D) = \text{SO\_GAD}(k, n, R, p_c, p_m, N_i, N_g, \beta)$. Algorithm SO_GAD accepts as input k, n, p_c, p_m, N_i, N_g, the coefficient β. This coefficient is optimized according to the chosen code and SNR to enhance the algorithm performance.

For $j = 1$ *to* n *do*
if $H^{(j)}$ *exists, then*

$$w_j = \widetilde{D}_j \left[\frac{\left\|H^{(j)} - R\right\| - \|D - R\|}{4} \right] = \widetilde{D}_j \sum_{\substack{p=1, p \neq j \\ H_p^{(j)} \neq D_p}}^{n} R_P \widetilde{D}_p, \tag{6}$$

else

$$w_j = \beta \widetilde{D}_j, \tag{7}$$

where $\widetilde{D}_j = 2D_j - 1$.
end if
End for

Algorithm 2. $(w, D) = \text{SO\_GAD}(k, n, R, p_c, p_m, N_i, N_g, N_s)$. Let N_s be the LBDA parameter $(N_s \le k)$ enhancing the decoding performances [8]. The algorithm SO_GAD accepts as input k, n, p_c, p_m, N_i, N_g, and N_s. This parameter is usually chosen to be $\lceil 2k/3 \rceil$ or k.

For $j = 1$ *to* $k - N_s$ *do*

$$w_j = \frac{\widetilde{D}_j}{|\Gamma|} \sum_{l=k-N_s+1, l \in \Gamma}^{n} \widetilde{D}_l w_l \quad \text{if} \sum_{l=k-N_s+1, l \in \Gamma}^{n} \widetilde{D}_l w_l \ge 0, \qquad w_j = \widetilde{D}_j \min_{l \in \Gamma} \left\{ \widetilde{D}_l w_l > 0 \right\} \quad \text{otherwise,} \tag{8}$$

where Γ denotes the set of positions j where $H^{(j)}$ exists.
End for
For $j = k - N_s + 1$ *to* n *do*
if $H^{(j)}$ *exists, then*

$$w_j = \left[\frac{1}{2} \widetilde{D}_j \sum_{l=1}^{n} \left(\widetilde{D}_l - \widetilde{H}_l^{(j)} \right) R_l \right] - R_j, \tag{9}$$

else

$$w_j = \frac{\widetilde{D}_j}{|\Gamma|} \sum_{l=k-N_s+1, l \in \Gamma}^{n} \widetilde{D}_l w_l \quad \text{if} \sum_{l=k-N_s+1, l \in \Gamma}^{n} \widetilde{D}_l w_l \ge 0, \qquad w_j = \widetilde{D}_j \min_{l \in \Gamma} \left\{ \widetilde{D}_l w_l > 0 \right\} \quad \text{otherwise,} \tag{10}$$

where Γ denotes the set of positions j where $H^{(j)}$ exists.
end if
End for.

3.2.1. Decoding. The SO_GAD algorithm uses GAD for decoding the input sequence R. The decision codeword D is the top of the N_gth generation sorted in ascending order of fitness, and the competitor codeword $H^{(j)}$ corresponding to the jth bit of D, if it exists, is the first member of the last generation which have the different jth bit $H_j^{(j)} (H_j^{(j)} \neq D_j)$.

3.2.2. Extrinsic Information. The decision codeword D and the associated competitor codewords $(H^{(j)})_{1 \le j \le n}$ are used to calculate the extrinsic information from the formulas (6) and (7) for the first algorithm and (8) and (9) in the case of the second one.

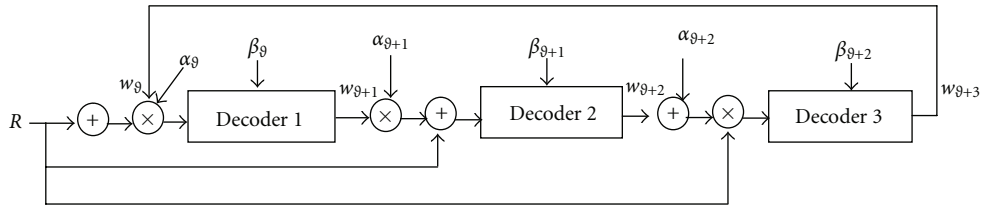

Figure 2: The $(\lfloor \theta/3 \rfloor + 1)$th iteration of IGAD1.

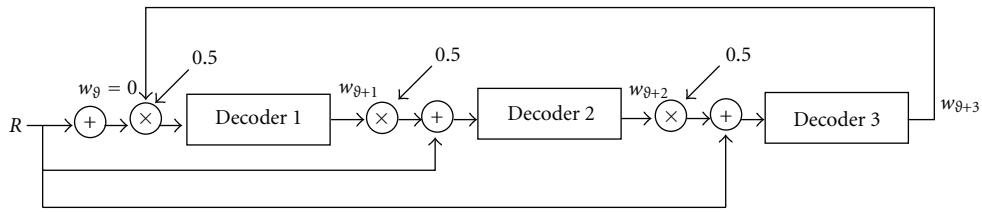

Figure 3: The $(\lfloor \theta/3 \rfloor + 1)$th iteration of IGAD2.

4. Iterative Decoding Algorithm and Complexity

In this section, we describe the iterative decoding algorithm of PBC based on GAD (IGAD), then we show that IGAD has a polynomial time complexity.

Let $\{C^{(i)}(n_i, k_i, d_i)\}_{1\le i\le 3}$ denotes three binary linear block codes of length n_i, dimension k_i, minimum Hamming distance d_i, and generator matrix $G^{(i)}$.

4.1. Iterative Decoding Algorithm. Let $(R_{ijk})_{1\le i\le n_2, 1\le j\le n_1, 1\le k\le n_3}$ be the received codeword. Figures 2 and 3 show the iterative decoding schemes of PBC based on GAD for the proposed algorithms. The following is an outline of IGADs.

Algorithm 3. IGAD$(k_1, k_2, k_3, n_1, n_2, n_3, R, p_c, p_m, N_i, N_g, N_{it}, \alpha, \{N_s|\beta\})$.

Algorithm IGAD accepts as input k_1, k_2, k_3, n_1, n_2, n_3, R, p_c, p_m, N_i, N_g, the iterations number N_{it}, the coefficients $(\alpha^{(\theta)})_{0\le\theta<3N_{it}}$. In the case of the first algorithm, we use the coefficients $(\beta^{(\theta)})_{0\le\theta<3N_{it}}$, and in the second, we use the N_s parameter. The α and β coefficients are optimized by simulation step by step for each code. For the second algorithm, we choose α to be 0.5.

Step 1. Extrinsic information *initialization*

$\theta = 0$, Iteration $= 1$.

Let $w_{ijk}^{(\theta)}$ is the extrinsic information given to θth elementary decoder by the other decoder:

$$w_{ijk}^{(0)} = 0, \quad 1 \le i \le n_2,\ 1 \le j \le n_1,\ 1 \le k \le n_3. \tag{11}$$

Step 2. Row, column, and depth decoding:

While $(It \le N_{it})$ do

Step 2.1. Decoding with SO_GAD the jth *column* and *estimating* the *extrinsic information* $w_{ijk}^{(\theta+1)}$, using (6) and (7), *of each vector* $s_{.j.}$ at the input of the elementary decoder

$$s_{ijk}^{(\theta)} = R_{ijk} + \alpha^{(\theta)} w_{ijk}^{(\theta)} \quad 1 \le i \le n_2,\ 1 \le k \le n_3. \tag{12}$$

Step 2.2. and 2.3. Repeat step 2.1 for decoding the rows and depths and estimating the extrinsic information. Let $D^{(\theta+3)}$ and $w^{(\theta+3)}$ be respectively the cubes decision and extrinsic information at the output of the depth elementary decoder.

Step 3. Iteration = Iteration + 1; $\theta = \theta + 1$.

End While.

Select the *decided codeword* $D^{(3N_{it})}$ *at the* N_{it}th iteration.

Stopping Criterion for the Second Algorithm.

Since the GAD decoder decides always a codeword, our second decoder does not need to use the NCB (nonconvergent block) decoder proposed in [8]. So, its complexity will be reduced.

4.2. Complexity Analysis. In this section, we present and compare the expressions of time complexities of the studied decoders.

4.2.1. IGADs Time Complexity. If we do not take into consideration the calculating step of the extrinsic information, the two algorithms have the same time complexity. The GAD algorithm for a linear block code $C(n, k)$ has polynomial time complexity $O(f(k, n, N_i, N_g))$, where the function f is given by [12]

$$f\left(k, n, N_i, N_g\right) = k^2 n + N_i N_g (kn + \log N_i). \tag{13}$$

Time Complexity of IGAD1

(i) Time complexity of extrinsic information computing: For each decision (row, column, or depth) at the

last generation of each iteration, the worst-case time complexity of competitors search is $O([N_i - 1]n)$.

From (6), the time complexity of extrinsic information calculating in the worst-case (if the competitor exists) at the last generation of each iteration is $O(n^2)$. So the total time complexity of extrinsic information computing is $O(\text{comp}_1(N_i, n))$, where

$$\text{comp}_1(N_i, n) = N_i n + n^2, \tag{14}$$

(ii) total time complexity:

At any iteration of IGAD1, the first elementary decoder has a time complexity of $O(k_2 k_3 f(k_1, n_1, N_i, N_g))$, the second decoder has a complexity of $O(n_1 k_3 f(k_2, n_2, N_i, N_g))$, and the third decoder has a time complexity of $O(n_1 n_2 f(k_3, n_3, N_i, N_g))$, so the total complexity is polynomial:

$$O\Big(N_{it}\Big[k_2 k_3 g\left(k_1, n_1, N_i, N_g\right) + n_1 k_3 g\left(k_2, n_2, N_i, N_g\right) + n_1 n_2 g\left(k_3, n_3, N_i, N_g\right)\Big]\Big), \tag{15}$$

where

$$g\left(k, n, N_i, N_g\right) = f\left(k, n, N_i, N_g\right) + \text{comp}_1(N_i, n). \tag{16}$$

For the symmetric *3D-PBC* $n_1 = n_2 = n_3 = n$ and $k_1 = k_2 = k_3 = k$, then the IGAD1 time complexity becomes

$$O\Big(N_{it}[k^2 + n^2 + kn]\Big[k^2 n + N_i N_g (kn + \log N_i) + nN_i + n^2\Big]\Big). \tag{17}$$

Time Complexity of IGAD2

(i) Time complexity of extrinsic information computing: The maximal number of competitors of each decision is $|\Gamma|_{\max} = n$. So, at the last generation of each iteration, the worst-case time complexity of the first step given by (8) is $O((k - N_s)\max(2n + 1, 2(n + N_s - k) + 3)) = O(n(k - N_s))$.

From (9), the worst-case time complexity of competitors search is is $O([N_i - 1](n - k + N_s))$.

From (9) and (10), the time complexity in the worst-case of the second step of extrinsic information calculating is

$$O((n - k + N_s)\max(2n + 3, 2(n + N_s - k) + 3, 2n + 1)) = O(n(n - k + N_s)). \tag{18}$$

So the total time complexity of extrinsic information computing is $O(\text{comp}_2(N_i, n, N_s, k))$, where

$$\text{comp}_2(N_i, n, N_s, k) = N_i(n - k + N_s) + n^2, \tag{19}$$

(ii) total time complexity:

The total complexity in this case is given from (16):

$$g\left(k, n, N_i, N_g\right) = f\left(k, n, N_i, N_g\right) + \text{comp}_2(N_i, n, N_s, k). \tag{20}$$

For the symmetric *3D-PBC* $n_1 = n_2 = n_3 = n$ and $k_1 = k_2 = k_3 = k$, then the IGAD2 time complexity becomes

$$O(N_{it}[k^2 + n^2 + kn]\times \Big[k^2 n + N_i N_g (kn + \log N_i) + N_i(n - k + N_s) + n^2\Big]\Big). \tag{21}$$

It is clear from (17) and (21) that IGAD2 is less complex than IGAD1, and their complexities are equal if $N_s = k$.

4.2.2. Chase-Pyndiah and LBDA Algorithms Time Complexities. We show that this algorithm has an exponential time complexity. Let $C(n, k, d)$ be a *BCH* code, and let M be the test patterns number used in both Chase and OSD-i (ordered statistic decoding) algorithms. The complexity of each algorithm is $O(Mn^2 \log_2 n)$.

The Euclidian distance computing of each codeword has a computational complexity of $O(n)$. So, the total time complexity of decoding and computing fitness of the M test patterns is $O(Mn^2 \log_2 n)$.

At any given decoding iteration of the Chase-Pyndiah algorithm, the sorting step of the M fitness has a time complexity of $O(M \log_2 M)$ and the the worst-case time complexity of competitors search is $O([M - 1]n)$. Thus, the total time complexity of the Chase-Pyndiah algorithm is

$$O(N_{it}[k_2 k_3 F(n_1, k_1, t_1) + k_3 n_1 F(n_2, k_2, t_2) + n_1 n_2 F(n_3, k_3, t_3)]), \tag{22}$$

where

$$F(n, k, t) = M\Big[n^2 \log_2 n + \log_2 M\Big]. \tag{23}$$

Thus, in the case of $n_1 = n_2 = n_3 = n$, $k_1 = k_2 = k_3 = k$, and, the exponential time complexity of the two algorithms is

$$O\Big(N_{it} M[n^2 + k^2 + kn]\Big[\log_2 M + n^2 \log_2 n\Big]\Big). \tag{24}$$

Note that in the case of Chase-2 algorithm, $M = 2^t$, where $t = \lfloor (d - 1)/2 \rfloor$.

From (17) and (24), it is shown that IGAD1 and IGAD2 are less complex than the two Chase-Pyndiah and LBDA algorithms for codes with large correction capacity t or for large i parameter or also with great length and low rate.

5. Simulation Results

The figures in this section plot the bit error rate (BER) versus the energy per bit to noise power spectral density ratio E_b/N_0 for the symmetric *3D-PBC* $(16, 11, 4)^3$ and $(31, 21, 5)^3$. The simulation parameters used in IGADs are given in Table 1.

5.1. IGAD1 Performances

5.1.1. Scaling Factors Optimization for IGAD1. As the iterations number increases, the extrinsic information gradually becomes more reliable. To take the effect into account, the scaling factors α are used to reduce the turbo decoder input

Table 1: Simulation default parameters.

N_e	1
N_{it}	12 for $(16, 11)^3$ and 15 for $(31, 21)^3$
Channel	AWGN
Modulation	BPSK
Min. number of transmitted frames	100
Min. of erroneous frames	30

Table 2: Optimized values of α and β for IGAD1.

Optimized values	$\alpha^{(\theta)}$
$(31, 21, 5)^3$	$(0, 1, 2, 3, 5, 7, 9, 10, 11, 15,$ $16, 17, 19, 19, 19, 15, 15, 15,$ $19, 19, 19, 17, 17, 17) \cdot 10^{-2}$ $\alpha^{(\theta)} = 0.2, 24 \le \theta \le 44$
$(16, 11, 4)^3$ Optimized α	$(0, 6, 20, 30, 30, 20, 30, 23,$ $20, 50, 20, 50, 20, 30, 30, 20,$ $33, 30, 30, 30, 40, 40, 40, 30,$ $40, 30, 35, 40, 40, 40, 40,$ $32, 40, 23, 30, 40) \cdot 10^{-2}$
$(16, 11, 4)^3$ Nonoptimized α	$0, 0.1, 0.01, 0.1, 0.1, 0.02,$ $0.1, 0.1, 0.03, 0.1, 0.1, 0.05,$ $0.1, 0.1, 0.07, 0.1, 0.1, 0.09$ $\alpha^{(\theta)} = 0.1, 18 \le \theta \le 35$
Optimized values	$\beta^{(\theta)}$
All codes	$(0, 20, 40, 60, 80) \cdot 10^{-2}$ $\beta^{(\theta)} = 1$, for all $\theta, \theta \ge 5$

impact. It has shown that these factors depend on the code and GAD. So, they are optimized step by step for each code. The optimized values α and β for our algorithm are shown in Table 2. However, as the scaling factors α and β are gradually increased or decreased from the optimal values, the decoding performance of IGAD1 decoder decreases. Figure 4 shows the gain with the optimized values of α, for $(16, 11, 4)^3$, compared to the values taken randomly. The values of genetic parameters used are $N_g = 18$, $N_i = 35$, $p_c = 0.97$ and $p_m = 0.03$.

5.1.2. Effect of Evaluated Codewords Number. Generally, increasing the number of evaluated codewords $N_i N_g$, the probability to find the codeword closest to the input sequence becomes high. This makes it possible to improve the BER performances. The effect of increasing the number of evaluated codewords on the BER improvement for code $(16, 11, 4)^3$ at the 12th iteration is presented in Figures 5 and 6. The values $N_g = 18$ and $N_i = 60$ can be the optimal values in a large range E_b/N_0. The other genetic parameters for the first optimization are $N_i = 35$, $p_c = 0.97$, and $p_m = 0.03$; $N_g = 18$, $p_c = 0.97$, and $p_m = 0.03$ for the second.

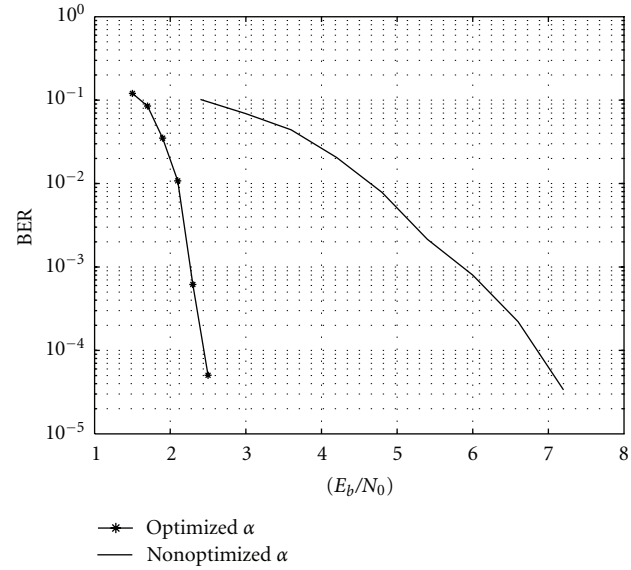

Figure 4: Effect scaling factor of $(16, 11, 4)^3$ for 12th iterations on IGAD1.

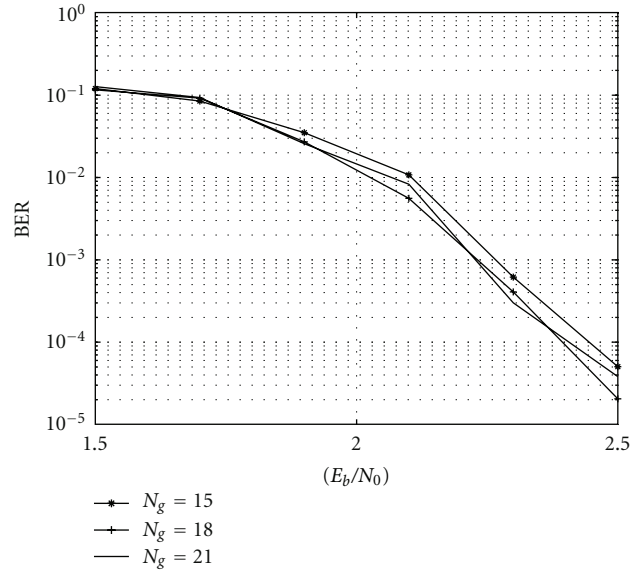

Figure 5: Effect of the generation number for $(16, 11, 4)^3$ at 12th iteration on IGAD1.

5.1.3. Cross-Over Rate Effect. Since the cross-over rate is one of the important features of a genetic algorithm, an optimization of this probability is necessary. Figure 7 shows the optimized value $p_c = 0.97$ for $(16, 11, 4)^3$ *3D-PBC* which improves the BER at a rather high SNR and at 12th iteration. This value closing to 1 means that IGAD1 requires a broad exploration and efficient exploitation, but increases somewhat the algorithm complexity. Indeed, when p_c is close to 0, the crossover operation will occur rarely. For this simulation, we fixed the other parameters as follows: $N_g = 18$, $N_i = 60$, and $p_m = 0.03$.

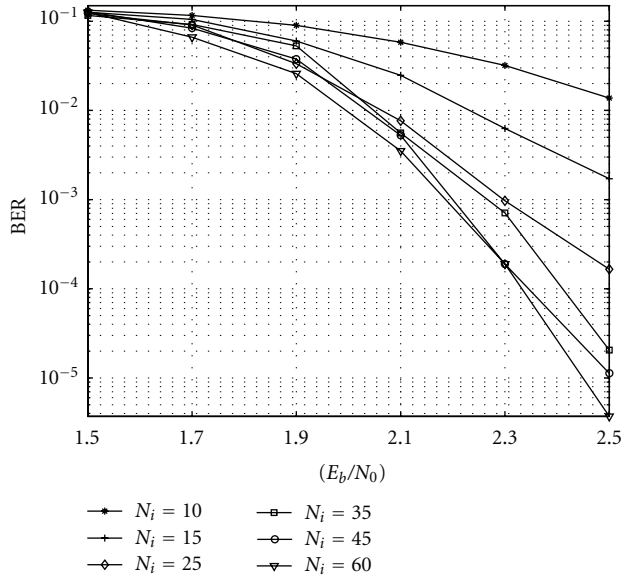

Figure 6: Effect of the population size for $(16, 11, 4)^3$ at 12th iteration on IGAD1.

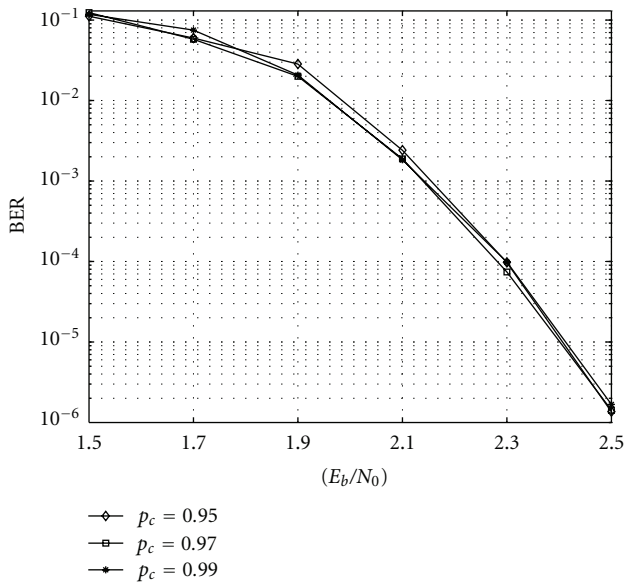

Figure 7: Effect of the crossover probability for $(16, 11, 4)^3$ at 12th iteration on IGAD1.

5.1.4. Mutation Rate Effect. The effect of mutation rate on IGAD1 for *BCH* $(16, 11, 4)^3$ *3D-PBC* is depicted in Figure 8. it is shown that $p_m = 0.05$ is the optimal value for BER at a high SNR and at 12th iteration. One reason of this value close to 0 may be the stability of members in vicinity of optima for low mutation rates. The fixed values are $N_g = 18$, $N_i = 60$, and $p_c = 0.97$.

5.1.5. Code Rate Effect. The Figure 9 shows the improvement/degradation of the BER performance of IGAD1 at

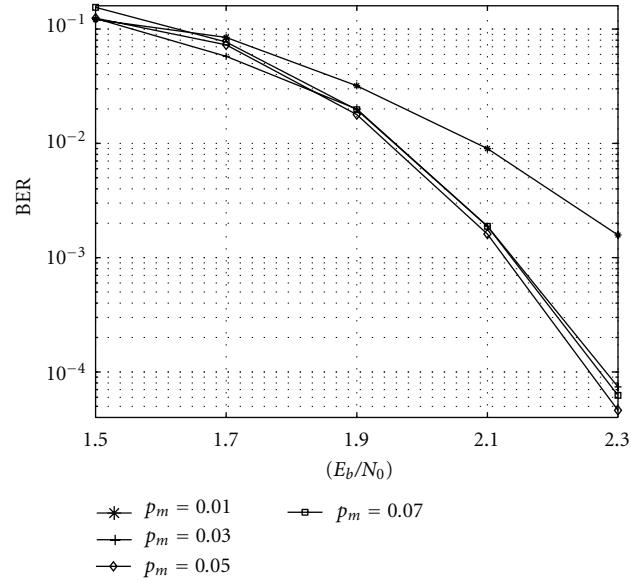

Figure 8: Effect of the mutation rate for $(16, 11, 4)^3$ *3D-PBC* at 12th iteration on IGAD1.

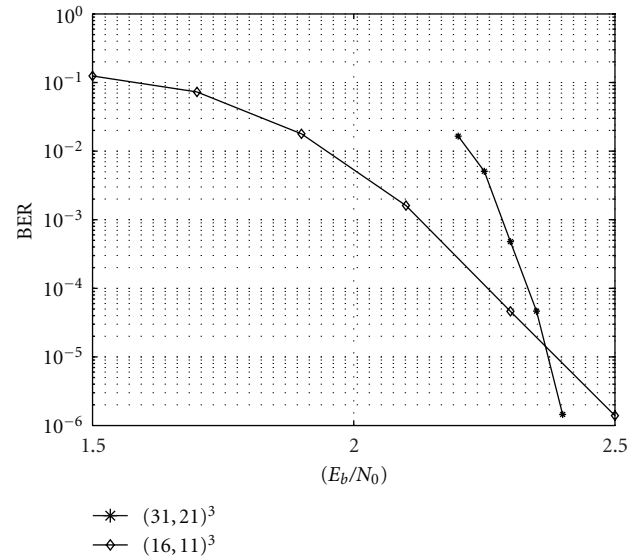

Figure 9: IGAD1 performances for *BCH* $(16, 11, 4)^3$ and *BCH* $(31, 21, 4)^3$ *3D-PBC* PBC at 12th iteration.

the 12th and 15ths, iteration respectively, with decreasing/increasing the code dimension or code rate. The rate 0.31 of $(31, 21, 4)^3$ is less than that of $(16, 11, 4)^3$ which equals to 0.32. This explains the better performances for the first *3D-PBC* code in the range $E_b/N_0 \geq 2.5$ dB. In this simulation, we adopted the optimal values previously found: $N_g = 18$, $N_i = 60$, $p_c = 0.97$, and $p_m = 0.03$.

5.1.6. Comparison between IGAD1 and IGAD2. As the iteration number increases, the IGADs performances improve approximately in this paper in the whole E_b/N_0 range for all *3D-PBC* studied. The performances of the IGAD decoders

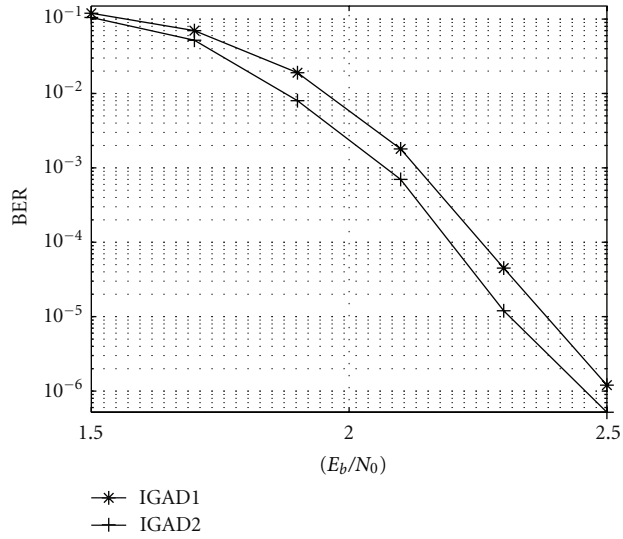

FIGURE 10: BER of IGAD1 and IGAD2 of $(16, 11, 4)^3$ for 12 iteration ($N_s = 8$).

is depicted in Figure 10 for *BCH* (16, 11, 4)^3 *3D-PBC*. These performances can be improved by increasing the total number of members as shown in Figure 6. The IGAD1 and IGAD2 performances are respectively about 1.4 dB and 1.33 dB away from the Shannon capacity limit, which is 0.97 dB for this code. We used the following optimized parameters: $N_g = 18$, $N_i = 60$, $p_c = 0.97$, and $p_m = 0.05$.

6. Conclusion

In this paper, we have presented two iterative decoding algorithms which can be applied to any arbitrary 3D-product block codes based on a genetic algorithm, without the need of a hard-in hard-out decoder. Our theoretical results show that these algorithms reduce the decoding complexity, for codes with a low rate and large correction capacity t or large i parameter used in LBDA algorithm. Furthermore, the performances of these algorithms can be improved by using asymmetric *3D-PBC* codes and also, by tuning some parameters like the selection method, the crossover/mutation rates, the population size, the number of generations, and the iterations number. These algorithms can be applied again on multipath fading channels in both CDMA systems and systems without spread-spectrum. Those features open broad prospects for decoders based on artificial intelligence.

References

[1] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Transactions on Communications*, vol. 44, no. 9, pp. 1261–1271, 1996.

[2] D. J. C. Mackay and R. M. Neal, "Good codes based on very sparse matrices," in *Proceedings of the 5th IMA Conference on Cryptography and Coding*, Springer, Berlin, Germany, 1995.

[3] P. Elias, "Error-free coding," *IRE Transactions on Information Theory*, vol. PGIT-4, pp. 29–37, 1954.

[4] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. IT-27, no. 5, pp. 533–547, 1981.

[5] S. Lin and D. Costello, *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, 1983.

[6] R. Pyndiah, A. Glavieux, A. Picart, and S. Jacq, "Near optimum decoding of product codes," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '94)*, vol. 1–3, pp. 339–343, San Francisco, Calif, USA, 1994.

[7] D. Chase, "A class of algorithms for decoding block codes with channel measurement information," *IEEE Transactions on Information Theory*, vol. 18, pp. 170–181, 1972.

[8] P. A. Martin, D. P. Taylor, and M. P. C. Fossorier, "Soft-input soft-output list-based decoding algorithm," *IEEE Transactions on Communications*, vol. 52, no. 2, pp. 252–262, 2004.

[9] Y. S. Han, C. R. P. Hartmann, and C.-C. Chen, "Efficient maximumlikelihood soft-decision decoding of linear block codes using algorithm A∗," Tech. Rep. SU-CIS-91-42, School of Computer and Information Science, Syracuse University, Syracuse, NY, USA, 1991.

[10] H. S. Maini, K. G. Mehrotra, C. Mohan, and S. Ranka, "Genetic algorithms for soft decision decoding of linear block codes," *Journal of Evolutionary Computation*, vol. 2, no. 2, pp. 145–164, 1994.

[11] J. L. Wu, Y. H. Tseng, and Y. M. Huang, "Neural networks decoders for linear block codes," *International Journal of Computational Engineering Science*, vol. 3, no. 3, pp. 235–255, 2002.

[12] F. El Bouanani, H. Berbia, M. Belkasmi, and H. Ben-azza, "Comparaison des décodeurs de Chase, l'OSD et ceux basés sur les algorithmes génétiques," in *Proceedings of the GRETSI*, Troyes, France, 2007.

[13] M. Belkasmi, H. Berbia, and F. El Bouanani, "Iterative decoding of product block codes based on the genetic algorithms," in *Proceedings of the 7th International ITG Conference on Source and Channel Coding (SCC'08)*, 2008.

[14] F. J. Macwilliams and N. J. A. Sloane, *The Theory Error Correcting Codes*, vol. North-Holland, Amsterdam, The Netherlands edition, 1978.