Int. J. Communications, Network and System Sciences, **2012, 5, 557-568**
http://dx.doi.org/10.4236/ijcns.2012.59066 Published Online September 2012 (http://www.SciRP.org/journal/ijcns)

Majority Voting Procedure Allowing Soft Decision Decoding of Linear Block Codes on Binary Channels

Saïd Nouh, Achraf El Khatabi, Mostafa Belkasmi
SIME Lab, National School of Computer Science and Systems Analysis (ENSIAS),
Mohammed V Souissi University, Rabat, Morocco
Email: nouh_ensias@yahoo.fr

Received July 3, 2012; revised July 30, 2012; accepted August 7, 2012

ABSTRACT

In this paper we present an efficient algorithm to decode linear block codes on binary channels. The main idea consists in using a vote procedure in order to elaborate artificial reliabilities of the binary received word and to present the obtained real vector *r* as inputs of a SIHO decoder (Soft In/Hard Out). The goal of the latter is to try to find the closest codeword to *r* in terms of the Euclidean distance. A comparison of the proposed algorithm over the AWGN channel with the Majority logic decoder, Berlekamp-Massey, Bit Flipping, Hartman-Rudolf algorithms and others show that it is more efficient in terms of performance. The complexity of the proposed decoder depends on the weight of the error to decode, on the code structure and also on the used SIHO decoder.

Keywords: Error Correcting Codes; Genetic Algorithms; HIHO Decoder; SIHO Decoder; Vote Procedure

1. Introduction

The current large development and deployment of wireless and digital communication encourages the research activities in the field of error correcting codes. Codes are used to improve the reliability of data transmitted over communication channels susceptible to noise. Coding techniques create codewords by adding redundant information to the user information.

There are two classes of error correcting codes: convolutional codes and block codes. The class of block codes contains two subclasses: nonlinear codes and linear codes. The principle of a block code $C(n, k)$ is as follows: the initial message is cut out into blocks of length k. The length of the redundancy is $n - k$ and thus the length of transmitted blocks is n. The main block codes are linear. If the code C is linear then the code $C^{\perp}(n, n - k)$ defined by (1) is also linear with "." denotes the scalar (dot) product.

$$h \in C^{\perp} \Leftrightarrow \forall c \in C : c \cdot h = 0 \qquad (1)$$

The code $C^{\perp}$ is called the dual code of C and each equation of the form given by (1) is called an orthogonal equation or a parity check equation.

There are two categories of decoding algorithms: Hard decision and Soft decision algorithms. Hard decision algorithms work on the binary form of the received information and generally they use the Hamming distance as a metric to minimize [1]. In contrast, soft decision algorithms work directly on the received symbols and generally they use the Euclidian distance as a metric to minimize [1].

The decoding category depends on the industrial requests and the communication channel. When the channel allows to measure the reliabilities $|r_i|_{i<n}$ (float symbols) of the sequence r of length n to decode the soft decision decoders working on these reliabilities allow to win generally about 2 dB more than the hard decision decoders working on the binary form of r can do. This difference between soft and hard decision decoders is justified by the proportionality between each reliability $|r_i|$ and the probability that the symbol r_j is correct.

Even if the soft decision decoders are more efficient than the hard decision decoders these latest are still an interesting subject for many scientists and industries thanks to their advantages. Some of those latest are listed below:

- When only the binary form of the received word is available as in the storage systems, the use of hard decision decoders becomes the only solution.
- In [2] we have given an efficient method to find the weight enumerator of a code which requires the use of a hard decision decoder.
- If a code can be efficiently decoded by a hard decision decoder then an efficient soft decoding algorithm of this code can be obtained by the Chasing technique described in [3].
- Some cryptographic systems use hard decoders to extract a noise added explicitly during the emission

phase in order to decrypt the message.

The hard decision decoding problem is in general NP-Hard. The wealth of the algebraic structure of some codes allows simplifying this hardness as in the BCH codes [4,5]. The famous permutation decoding algorithm [6] is a generic decoder, applicable to all systematic codes but it requires finding a PD-Set which is also a NP-Hard problem [7]. Another generic decoder of linear block codes is the Bit Flipping decoding algorithm (BF) based on the verification of orthogonal equations which was developed firstly for LDPC codes [8] and it is generalized thereafter for linear block codes but without ensuring good error correcting performances [9].

In [10,11], the authors have applied artificial intelligence to solve the decoding problem by genetic algorithms in [10] and by neural networks in [11]. However the authors of [12] have used a mathematical approach based on Coset Decomposition and syndrome decoding.

On the other side the authors of [13] and [14] have proposed a low complexity soft-Input Soft-Output module to decode convolutional codes. They use classical Viterbi algorithm and a module for computing softoutput from the hard output of the Viterbi algorithm.

The purpose of this work is to find a generic efficient hard decision decoding algorithm of linear block codes by combining a vote procedure with a soft decision decoding. Majority voting procedure is done by a module prior a soft decision decoder. Thus the efficiency of soft decision decoding algorithms becomes exploitable in the case of binary channels by creation of artificial reliabilities with a vote using a reduced number of orthogonal equations.

In the rest of this paper $C(n, k, d)$ designate a linear code of length n, dimension k, minimum distance d, error correcting capability t, generated by a matrix G and it can be checked by a matrix H and we note $dH(x,y)$ and $dE(x,y)$ respectively the Hamming and the Euclidean distance between two vectors x and y.

The remainder of this paper is structured as follows. In Section 2 we present some decoding algorithms as related works. In Section 3 we present the proposed decoder and we make a comparison with other decoders. Finally, a conclusion and a possible future direction of this research are outlined in Section 4.

2. Related Works

In this section we present some hard decision decoding algorithms with which we will compare our proposed decoding scheme in the next section.

2.1. The Bit Flipping (BF) Decoding Algorithm

The matrix H has n columns and $n - k$ rows or more, let V be a vector of length n and h a binary word to decode. The BF algorithm uses the following vote algorithm:

2.1.1. The Vote Algorithm

```
Inputs:
- L a list a dual codewords ($L \subseteq C^{\perp}$)
- M the number of dual codewords to use
- V a vector of length n.
- h a binary word of length n.
Outputs: V
Begin
for i from 1 to n do $V_i \leftarrow 0$; end for;
for i from 1 to M do
        u ← i^th element of L.
        if ($u.h \neq 0$)   then   for i from 1 to n do
                                       if ($u_i$=1)   then $V_j \leftarrow V_j + 1$;
                                       end for;
end for;
End;
```

We have observed that when the vote is efficient, the noised bits h_i have a big value of votes V_i.

2.1.2. The Gallager's Bit-Flipping Algorithm

The Gallager's bit flipping algorithm [8] works as follow:

```
Inputs:
- L: a list a dual codewords ($L \subseteq C^{\perp}$)
- iter_max: the maximum number of iterations
- threshold: the number of failed parity check equations to
        have for flipping.
- h: the received word.
Outputs: the decoded word h.
Begin
Continue ←true;
iter←0;
For j from 1 to n do: $z_j \leftarrow h_j$;
While (iter < iter_max and Continue = true)
{iter ← iter+1;
 Continue = false;
 Vote(h,L,V);
     For j from 1 to n
       If ($V_j \geq$ threshold) then
            { $h_j \leftarrow 1-h_j$; Continue=true ;}
     }
If (Continue=true) then For j from 1 to n $h_j \leftarrow z_j$;
End
```

This algorithm was developed firstly for decoding LDPC codes [8], its generalization for linear codes requires the use of a big number of parity check equations [9].

2.2. The Permutation Decoding Algorithm

The permutation decoding algorithm (PDA) was first developed by Jessie McWilliams [6]. It can be used when a certain number of permutations (automorphisms), leaving the code invariant, is known. The PDA correct a word by moving errors in the redundancy part of a permuted word, the hardness of finding the automorphism group restricts the use of this algorithm to codes with known stabilizers like the use of the projective special linear group for extended quadratic residue (EQR) codes.

2.3. The Hartman Rudolph Algorithm

The Hartman Rudolph (HR) decoder [15] is a symbol by symbol soft decision optimal decoding algorithm. It maximizes the probability that a bit corresponding to a symbol r_j of the sequence r to decode is equal to 1 or 0. Hartman and Rudolph have showed that this probability depends on all the codewords of the dual code $C^{\perp}$ generated by H. This algorithm has a high complexity because it uses 2^{n-k} dual codewords therefore it can be applicable only in the case of linear codes with height rate. More precisely it uses the formula (2) to decide if the mth bit of the decoded word c' is equal to 1 or 0.

$$\begin{cases} c'_m = 0 & \text{if } \sum_{j=1}^{2^{n-k}} \prod_{l=1}^{n} \left(\frac{1-\Phi_l}{1+\Phi_n} \right)^{c^{\perp}_{jl} \oplus \delta_{ml}} > 0 \\ c'_m = 1 & \text{otherwise} \end{cases} \qquad (2)$$

With the following notations:

- $\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$
- $\Phi_m = P_r(r_m/1) / P_r(r_m/0)$
- The bit $c^{\perp}_{jl}$ denotes the lth bit of the jth codeword of the code $C^{\perp}$.

A hard version of this algorithm can be obtained by using as inputs the binary form h_i of the received symbols r_i as follow:

$$\forall i \in \{1,2,3,\cdots,n\} : h_i = \begin{cases} -1 & \text{if } r_i \le 0 \\ +1 & \text{otherwise} \end{cases} \qquad (3)$$

3. The Proposed ARDec Decoder

3.1. The Principle

The proposed ARDec decoder (Artificial reliabilities based decoding algorithm) has the structure given in the **Figure 1**. It uses a generalized parity check matrix H^* to compute artificial reliabilities of the binary received word h.

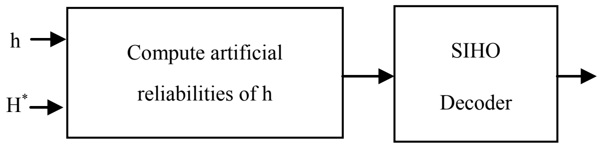

Figure 1. The proposed decoding algorithm ARDec scheme.

Let $C^{\perp}(n, k, d)$ be the dual code of C. The ARDec decoder uses the vote algorithm described in the Section 2.1 to compute artificial reliabilities. In the case of BPSK modulation, the bit 0 is represented by –1 and the bit 1 by 1 and ARDec works as follow:

```
Inputs:
- $H^*$ a generalized parity check matrix of C of M rows.
- h the binary word to decode.
Outputs: V
Begin
Vote(h, $H^*$, V);
Balance ($H^*$, V);
For i from 1 to n do:
    if ($h_i$=1) then $r_i = 1/(V_i+1)$ ; else $r_i = -1/(V_i+1)$ ;
Use a SIHO decoder to find the codeword c having
  the smallest Euclidean distance to r.
End
```

When the columns of the matrix H^* doesn't have the same weight, we propose to balance the vote vector V by the following Balance algorithm:

```
Inputs:
- $H^*$ a generalized parity check matrix of C of M rows.
- V a vector of voting values.
Outputs: V
Begin
For i from 1 to n do: $W[i] \leftarrow 1$ End For
For i from 1 to n do:
    For j from 1 to M do:
        $W[i] \leftarrow W[i] + H^*[i][j]$ ;
    End For
End For

For i from 1 to n do: $V[i] \leftarrow \frac{V[i]}{W[i]}$ End For
End
```

3.2. ARDec Based on Genetic Algorithms: ARDecGA

Genetic algorithms (GA) are heuristic search algorithms premised on the natural selection and genetic [16,17], we recall here some notions:

- Individual or chromosome: a potential solution of the problem, it's a sequence of genes.
- Population: a subset of the research space.
- Environment: the research space.
- Fitness function: the function to maximize/minimize.
- Encoding of chromosomes: it depends on the treated problem, the famous known schemes of coding are: binary encoding, permutation encoding, value encoding and tree encoding.
- Three operators of evolution:

1) Selection: it allows selecting the best individuals to insert in the intermediate generation and to create chil-

dren.

2) Crossover: For a pair of parents (p_1, p_2) it allows to create two children ch_1 and ch_2 with a crossover probability p_c.

3) Mutation: The genes of the individual are muted according to the mutation probability p_m.

The Maini algorithm [18], uses genetic algorithms to decode linear codes, the first step is to sort the received sequence r by reliabilities and to find a matrix G' of an equivalent code and the permutation π which binds the two codes. The information part $I(\pi(r))$ of $\pi(r)$ contains symbols with biggest reliabilities. A genetic algorithm works on $I(\pi(r))$ to find the codeword having the smallest Euclidean distance from $\pi(r)$.

The genetic operators of the Maini algorithm are given in [18] and we propose here a modification at the level of the initial population and the crossover operator.

For the crossover operator we choose to cross individuals in one point m, but this latest is randomly chosen between 1 and the length of the code.

For the initial population, we proceed as follow:

- Find h, the hard decision version of $\pi(r)$.
- Fix a value of p_i the probability to inverse a bit of h.
- $x \leftarrow h$.

For j from 1 to n do:

1) Generate uniformly a random value x, $0 \le x \le 1$.

2) If $(x \le p_i)$ then $x_i \leftarrow x_i \oplus 1$.

The value of p_i is chosen between 0.1 and 0.4.

For decoding a binary word, the sequence r of its artificial reliabilities can be created by the vote and balance procedures. After that the permutation π is obtained by sorting r. The ARDecGA algorithm based on the modified Maini decoder works on $\pi(r)$ as follow:

```
Inputs:
- The sequence π(r).
- N_g: Number of generations
- N_i: Population size
- N_e: elite number
- p_c: crossover probability
- p_m: mutation probability
Outputs: the codeword c.
Begin
h ← the hard version of π(r).
if (h∈C) then c ←h;
else
  begin
  Generate an initial population, of N_i individuals;
  Each individual is a binary word of length k. The first individual is
  the information part of h.
  Ngen   ← 1; continue← true;
While (Ngen ≤ N_g and continue=true) do
     Compute the fitness of each individual a:
     $fitness(a) = dE(b,\pi(r))$ , b is the codeword
       associated to the individual a in the code generated
       by G'.
     If( $dH(b,\pi(h)) \le t$ ) then {c←b
                          continue← false}
     end If
     Sort the population by increasing order of fitness.
     Copy the best Ne individuals (of small fitness) in the
     intermediate population.
     For i=Ne+1 to Ni do
        Select two individuals; p1 and p2 among the
          best individuals.
        Cross and mute p1 and p2 to obtain ch1 and ch2
        according to pc and pm.
        Among ch1 and ch2, insert the best individual in
          the intermediate population.
     End for.
     Ngen ← Ngen+1.
     End while.
   If(continue=true) then
              c← the first individual in the population.
   End If
   $c \leftarrow \pi^{-1}(c)$
  end;
end
```

3.3. ARDec Based on OSD Algorithm of Order M: ARDecOSD^m

The ordered statistic decoding algorithm of order m OSD^m is a SIHO decoder developed by Fossorier *et al.* [19]. It starts by sorting the received word by reliabilities, after that it passes to decode this last in an equivalent code by inversing in each time m bits and re-encoding. In this paper we will use the OSD^0, OSD^1, OSD^2 and OSD^3 decoders as a module in our hard decision decoder, they have a polynomial complexity and they yield to good error correcting performances. This algorithm requires only the generator matrix of the code, this characteristics allows its use to decode linear codes without restriction.

3.4. Construction of the Generalized Check Matrix H^*

The choice of the generalized parity check matrix H^* is a key factor in the success of the ARDec decoder. For representing a linear code for the soft decision Belief Propagation decoding algorithm, Yedidia *et al.* [20,21] have showed that the check matrix should have the following characteristics:

1) The number of ones in each row is small.

2) The number of ones in each column is large.

3) For all pairs of rows of the check matrix, the number of columns that have a one in both rows is small; ideally zero or one.

In this work we will show that these characteristics are good criteria for choosing H^*. We use the algorithm given in [2] for finding a list L of codewords from the dual code of C, this list respect then the first characteristic given above. Here we propose a genetic algorithm GA-GPC for extracting H^* from L. This algorithm tries to improve the chosen generalized parity check matrix by considering the second and the third characteristics as fitness.

For explaining the GA-GPC algorithm we give the following definition.

Definition: Let C be a linear code and $C^{\perp}$ its dual, $d^{\perp}$ the minimum distance of $C^{\perp}$, v and w two elements of $C^{\perp}$ of weight $d^{\perp}$. The degree of cooperation between v and w is the difference between $d^{\perp}$ and the sum of their inner product. The degree of cooperation of a list $L' \subseteq C^{\perp}$ is the sum of the cooperation degrees between all its elements, two by two, it is given by:

$$co(L') = \sum_{L_i, L_j \in L', i \neq j} d^{\perp} - L'_i \cdot L'_j \quad (4)$$

The degree of cooperation allows checking if a list satisfies sufficiently the second and the third characteristics recommended by Yedidia *et al.* [20,21].

In the GA-GPC algorithm the list L is indexed from 1 to z, an individual is a subset containing exactly M elements of $J_z = \{1, 2, 3, \cdots, z\}$; it represent M elements of L. The mutation of a gene from an individual consists in replacing it by another element of J_z. The cross between two individuals in one point gives two children which can be repaired by mutation if they contain a multiple copies of the same gene.

The GA-GPC algorithm works as follow:

```
Inputs:
- M, the number of rows in $H^*$.
- n, the length of the code
- L a list of a minimum weight dual-codewords of size z.
- $N'_g$, number of generations
- $N'_i$, population size
- $N'_e$, elite number
- $p'_c$, crossover probability
- $p'_m$, mutation probability
Begin
Generate an initial population, of $N'_i$ individuals; each individual is a subset of size M of $J_z$.
N ← 1;
While (N ≤ $N'_g$) do
  {Compute the fitness of each individual A:
     $fitness(A) = co(A)$
    Sort the population by decreasing order of fitness.
    Copy the best $N'_e$ individuals (of big fitness) in the intermediate population.
    For i= $N'_e + 1$ to $N'_i$ :
      Select two individuals p1 and p2 among the best $N'_e$ individuals.
      Cross and mute p1 and p2 for obtaining ch1 and ch2 according to $p'_c$ and $p'_m$.
      Among ch1 and ch2, insert the best individual in the intermediate population.
      N← N+1 }
End
Outputs: the individual (matrix) having the best fitness.
```

3.5. Simulation Results of ARDec and Comparison to other Decoders

To show the efficiency of ARDec algorithm and the impact of its parameters we give in this subsection its error correcting performances for some linear codes form many classes with a comparison with other decoding algorithms over the AWGN channel (Additive White Gaussian Noise). It is known that the AWGN channel can be viewed as a binary channel. All simulations are obtained by using the parameters given in the **Table 1** and the default parameters of the GA-GPC algorithm given in the **Table 2**.

3.5.1. Simulation Results of ARDec Algorithm

The **Figure 2** presents the error correcting performances

Table 1. Default simulation parameters.

Simulation parameter	value
Channel	AWGN
Modulation	BPSK
Minimum number of residual bit in errors	200
Minimum number of transmitted blocks	5000

Table 2. Default GA-GPC parameters.

GA-GPC parameter	Parameter value
Crossover probability p'_c	0.91
Mutation probability p'_m	0.07
Number of generations N'_g	200
Population size N'_i	200
Elite number N'_e	$N'_i/2$

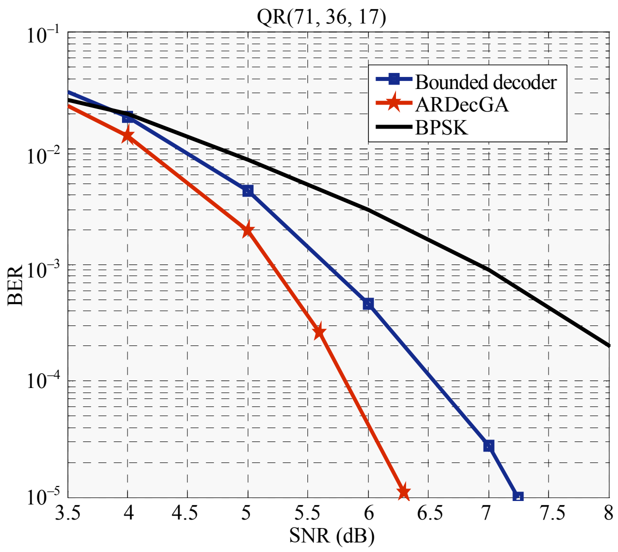

Figure 2. Error correcting performances of the ARDecGA algorithm for the QR(71, 36, 11) code.

of ARDecGA with $M = 500$, $N_g = 50$, $N_i = 150$, $N_e = 2$, $p_c = 0.95$, $p_m = 0.03$ for the Quadratic Residue code QR(71, 36, 11) compared to those of a bounded decoder (pseudo-decoder) which correct all configurations of errors of weight less than or equal to 5 (the error correcting capability of this code). For this code, the ARDecGA algorithm allows to win about 1 dB more than the 2.3 dB guaranteed by the bounded decoder, therefore 3.3 dB as coding gain is obtained.

In [22], authors have found a double circulant code DCC(62, 31, 12) which is optimal in the sense that it has the maximum possible minimum distance for the length 62 and the dimension 31. The **Figure 3** presents the error correcting performances of ARDecOSD^2 with $M = 100$ for this code and we have verified statistically that all errors of weight less than or equal to 5 are corrected thus about 2.6 dB as coding gain is obtained.

3.5.2. Impact of the Parameter *M* on ARDecGA Algorithm Performances

To show the impact of the parameter M on the error correcting performances of the ARDecGA algorithm for a DSC (Difference-Set Cyclic Code) code, we give in the **Figure 4** the simulation results for the DSC(73, 45, 10) code with $N_e = 2$, $N_g = 10$, $p_c = 0.95$, $p_m = 0.03$ and $N_i = 50$.

The parameter M has an important effect on the efficiency of ARDecGA algorithm. At BER = 10^{-4} there is a difference of more than 3 dB between $M = 10$ and $M = 300$ but the difference between $M = 300$ and $M = 500$ is negligible.

3.5.3. Impact of the Parameter N_g on the ARDecGA Performances

To show the impact of the number of generations N_g on the error correcting performances of the ARDecGA algorithm, we give in the **Figure 5** the simulation for the BCH(63, 30, 13) code with $N_e = 2$, $M = 1000$, p_c=0.95, $p_m = 0.03$ and $N_i = 300$. For this code 20 generations are sufficient.

3.5.4. Impact of the Parameter N_i on the ARDecGA Performances

To show the impact of the population size N_i on the error correcting performances of the ARDecGA algorithm, we give in the **Figure 6** the simulation for the BCH(63, 30, 13) code with $N_e = 2$, $M = 1000$ and $N_g = 100$. For this code 150 individuals in each generation are sufficient.

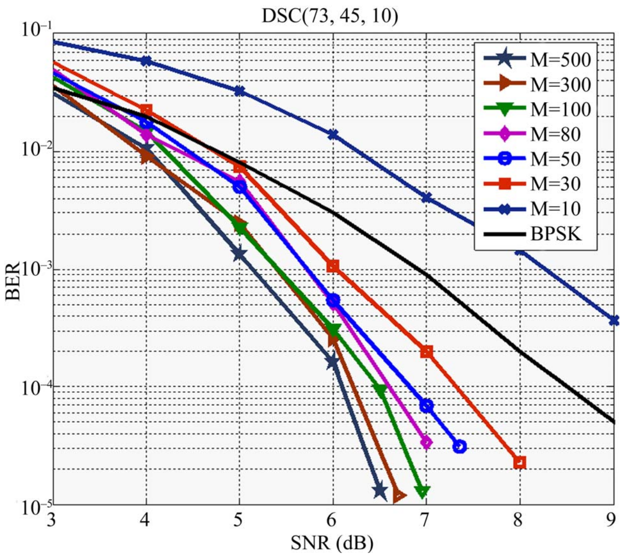

Figure 4. Impact of the parameter *M* on the ARDecGA error correcting performances for DSC(73, 45, 10) code.

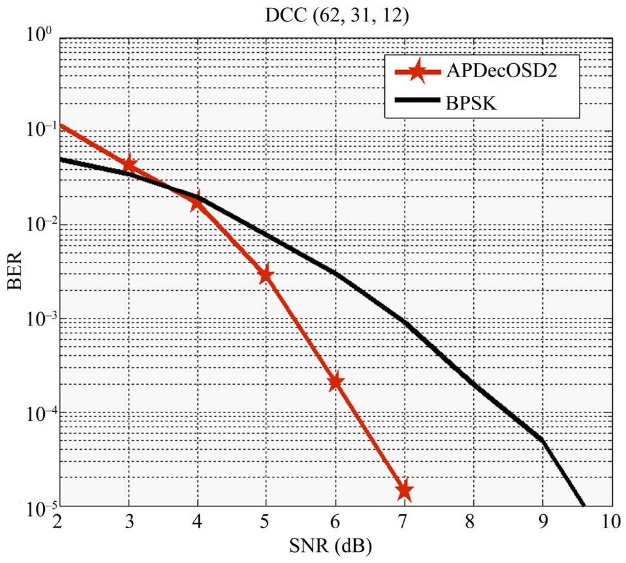

Figure 3. Error correcting performances of the ARDecOSD^2 algorithm for a DCC(62, 31, 12) code.

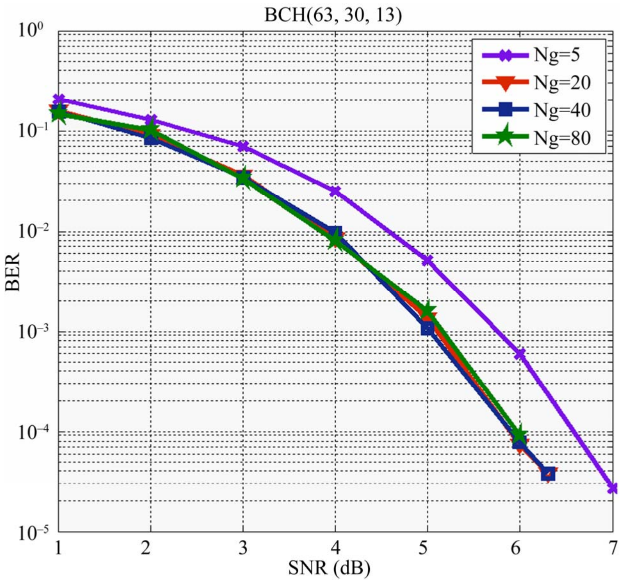

Figure 5. Impact of the parameter N_g on the ARDecGA error correcting performances for the BCH(63, 30, 13) code.

3.5.5. Impact of the Order m on the ARDecOSD^m Performances

To show the impact of the parameter m on the error correcting performances of the ARDecOSD^m algorithm we give in the **Figure 7** the simulation results for the QR(89, 45, 17) code with M = 1500. At BER = 10^{-5} there is a gain of about 1.3 dB between the orders 0 and 3.

3.5.6. Comparison between ARDecOSD^3 and ARDecGA Algorithms

To compare the error correcting performances of the ARDecOSD^3 ($M = 1000$) and ARDecGA ($N_i = 200$, $N_g = 50$, $N_e = 2$, $p_c = 0.95$, $p_m = 0.03$, $M = 1000$) algorithms we give

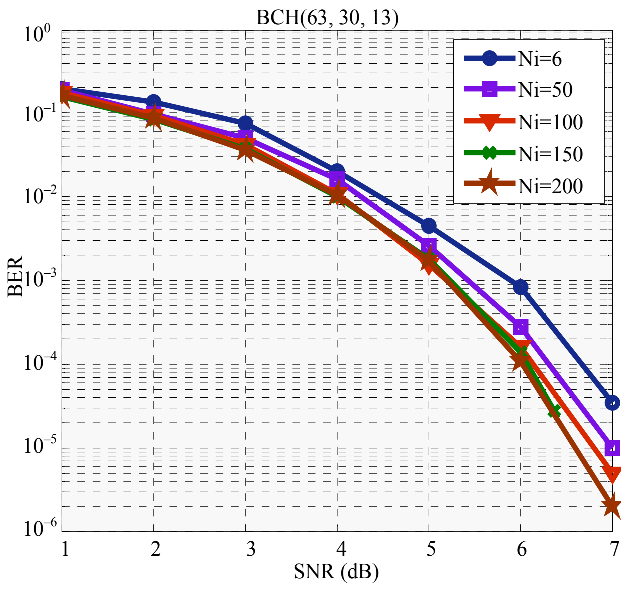

Figure 6. Impact of the parameter N_i on the ARDecGA error correcting performances for the BCH(63, 30, 13) code.

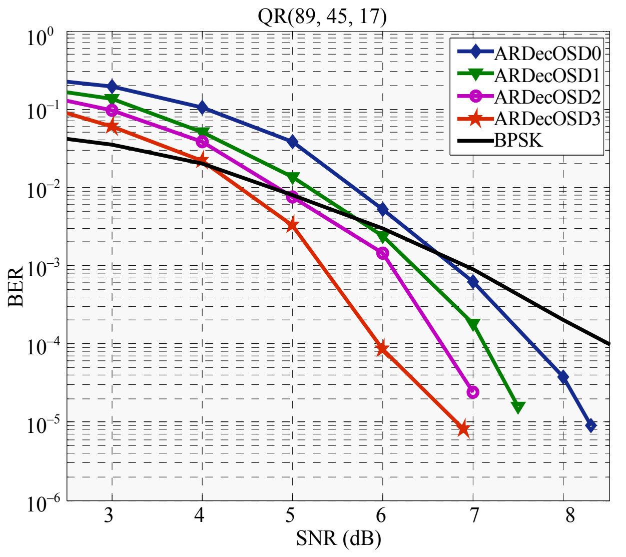

Figure 7. Impact of the order m on the ARDecOSD^m error correcting performances for the QR(89, 45, 17) code.

in the **Figure 8** the simulation results for the BCH(63, 39, 9) code. The ARDecGA is relatively better than the ARDecOSD^3 and it allows to win about 0.5 dB compared to the Berlekamp-Massey decoder (BM) at BER = 10^{-5}.

3.5.7. Comparison between ARDecGA and Hard Decision Decoding Algorithm of OSMLD Codes

The vote technique is often used to decode linear code with a particular structure like the class of the OSMLD (One Step Logic Majority Decodable) codes which contains the DSC(73, 45), DSC(273, 191) and BCH(15, 7) codes. The famous known decoder of OSMLD code is the majority logic decoder [23]. The **Figure 9** presents a comparison between error correcting performances of

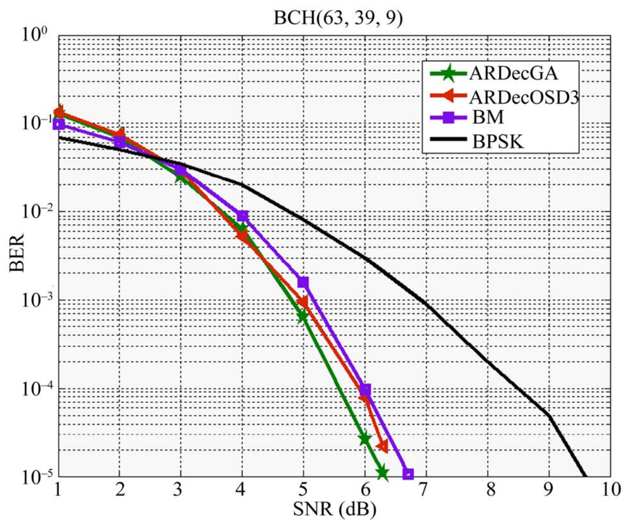

Figure 8. Comparison between ARDecOSD^3 and ARDecGA performances for the BCH(63, 39, 9) code.

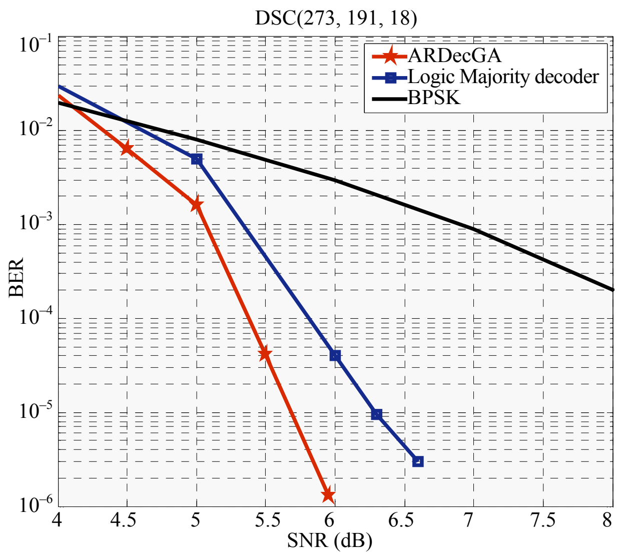

Figure 9. Comparison between ARDecGA and OSMLD decoder performances for DSC(273, 191, 18) code.

ARDecGA (M = 273, N_g = 50, N_i = 300, p_c = 0.95, p_m = 0.03, N_e = 2) and the majority logic decoder, it shows that ARDecGA allows to win about 0.7 dB at BER = 10^{-5} than this classic decoder.

At BER = 10^{-5} the ARDecGA decoder allows to win about 0.6 dB for the BCH(15, 7) code (**Figure 10**) and 0.9 dB for the DSC(73, 45) code (**Figure 4**).

3.5.8. Comparison between ARDecGA and Gallager Bit Flipping Algorithms

The **Figure 11** presents a comparison between the error correcting performances of ARDecGA (N_g = 6, N_i = 20, p_c = 0.95, p_m = 0.03, N_e = 2) and the Bit Flipping Algorithm (BF) applied to the BCH(63, 39, 9) code with the same number of parity check equations M = 450. The maximum number of iteration in the BF algorithm is fixed at 100 and the value of the threshold is optimized. At BER = 10^{-5} the ARDecGA decoder allows to win about 0.6 dB.

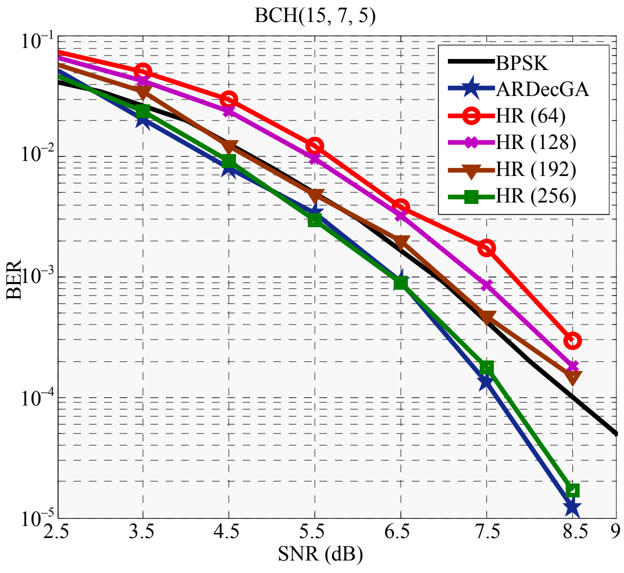

Figure 10. Comparison between the performances of ARDecGA and HR decoders for the BCH(15, 7, 5) code.

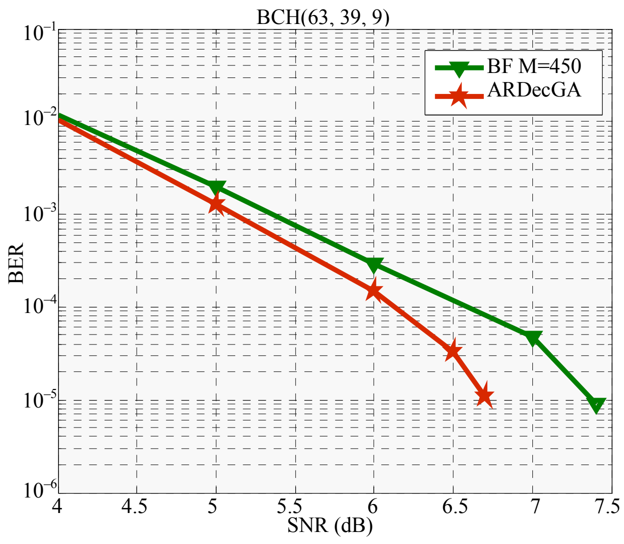

Figure 11. Comparison between ARDecGA and BF error correcting performances for the BCH(63, 39, 9) code.

3.5.9. Comparison between ARDecGA and the Hard Decision Hartman-Rudolf Algorithm

The **Figure 10** presents a comparison between the error correcting performances of ARDecGA algorithm (M = 10, N_g = 3, N_i = 8, p_c = 0.95, p_m = 0.03, N_e = 2) and the Hartman Rudolf decoder (hard decision version) applied to the BCH(15, 7, 5) code with the value of M as parameter. This figure shows that the ARDecGA decoder with 10 vectors from $C^{\perp}$ and 26 individuals in the genetic algorithm allows to win more than 1 dB at BER = 10^{-4} compared to the HR decoder with 64, 128 and 192 vectors from $C^{\perp}$. The ARDecGA decoder has the same performances of the HR decoder when it uses all the 256 vectors.

When $n - k$ increase, the hard decision version of the Hartman-Rudolf algorithm becomes very complex. For the BCH(63, 39, 9) code there are 16777216 codewords in its dual code BCH$^{\perp}$(63, 24, 14). Here, we propose to use only the 450 codewords of the BCH$^{\perp}$(63, 24, 14) code having the minimum weight 14 in the decoding by the HR algorithm. The **Figure 12** presents a comparison between the performances of ARDecGA decoder (M = 450, N_g = 6, N_i = 20, Ne = 2, p_c = 0.95, p_m = 0.03) and the Hartman Rudolf decoder (M = 450) applied to the BCH(63, 39, 9) code. This figure shows that the ARDecGA allows to win about 1 dB compared to the HR decoder at BER = 10^{-5}.

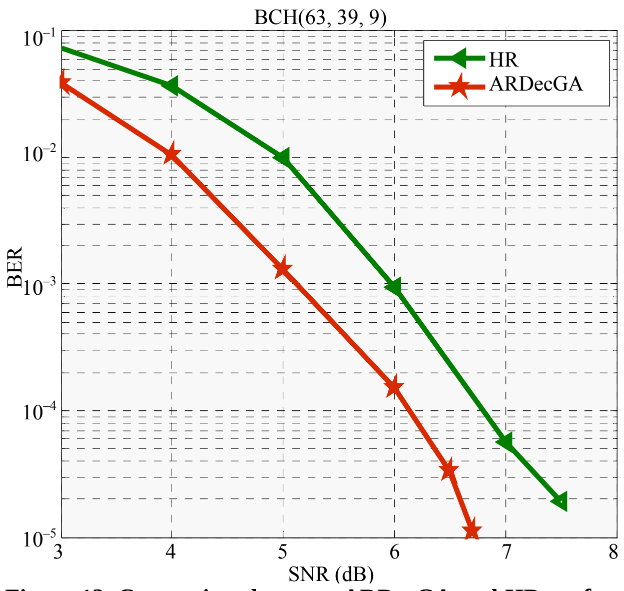

Figure 12. Comparison between ARDecGA and HR performances for the BCH(63, 39, 9) code.

3.5.10. Comparison between ARDecGA, Berlekamp-Massey (BM) and Chase-2 Algorithms

The **Figure 13** presents a comparison between the error correcting performances on the BCH(63, 30, 13) code of the following three algorithms:

1) Chase-2 algorithm [3] working on the channel reliabilities measurements of the received sequences.

2) Berlekamp-Massey decoder (BM) [4,5] working on the binary form of the received sequences.

3) ARDecGA ($M = 1000$, $N_i = 300$, $N_g = 50$, $N_e = 2$, $p_c = 0.95$, $p_m = 0.03$) working on the computed artificial reliabilities.

The **Figure 13** shows that the error correcting performances of ARDecGA are between those of the Chase-2 and the Berlekamp-Massey decoders.

3.5.11. Comparison between ARDecOSD^3 and the Permutation Decoding Algorithm

The **Figure 14** presents a comparison between the error correcting performances of ARDecOSD^3 ($M = 50$) and the permutation decoding algorithm (PDA) [6,7] for the extended quadratic residue code EQR(48, 24, 12); the error correcting performances are the same.

3.5.12. Comparison between ARDecGA and the Hard Decision Maximum Likelihood Decoder

The hard decision maximum likelihood decoder (MLD hard) is the most efficient decoder, it decides by the closest codeword. The **Figure 15** presents a comparison between the error correcting performances of ARDecGA ($M = 20$, $N_i = 15$, $N_g = 8$, $p_c = 0.95$, $p_m = 0.03$, $N_e = 2$) and this decoder for the QR(17, 9, 5) code.

The ARDecGA, which search only in 101 codewords,

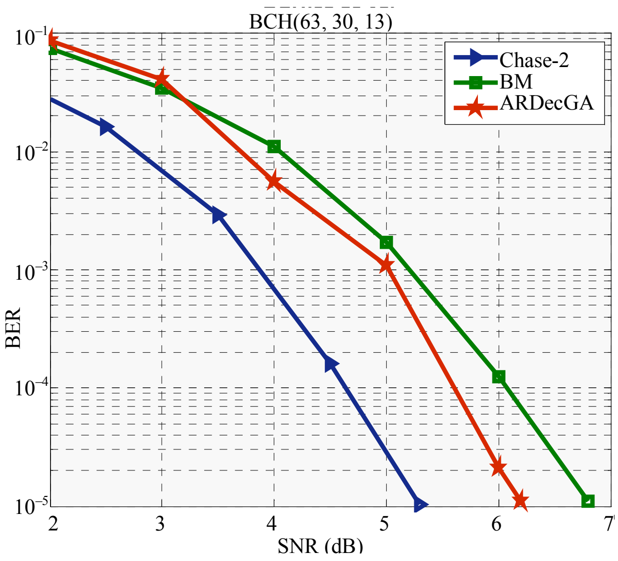

Figure 13. Comparison between ARDecGA, Chase-2 and BM performances for the BCH(63, 30, 13) code.

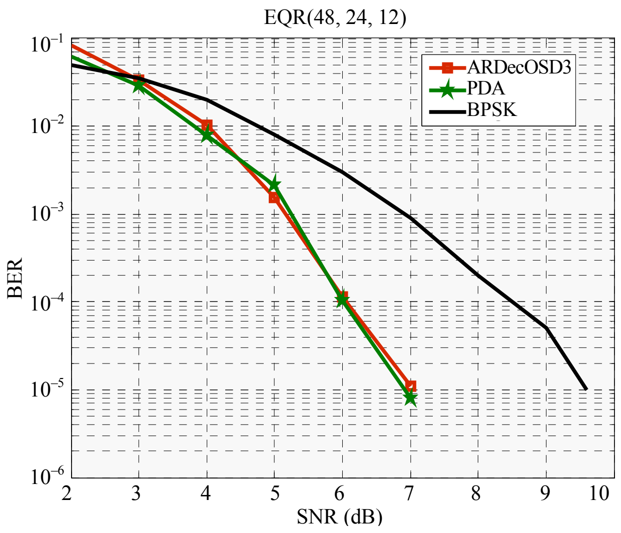

Figure 14. Comparison between ARDecOSD^3 and PDA performances for the EQR(48, 24, 12) code.

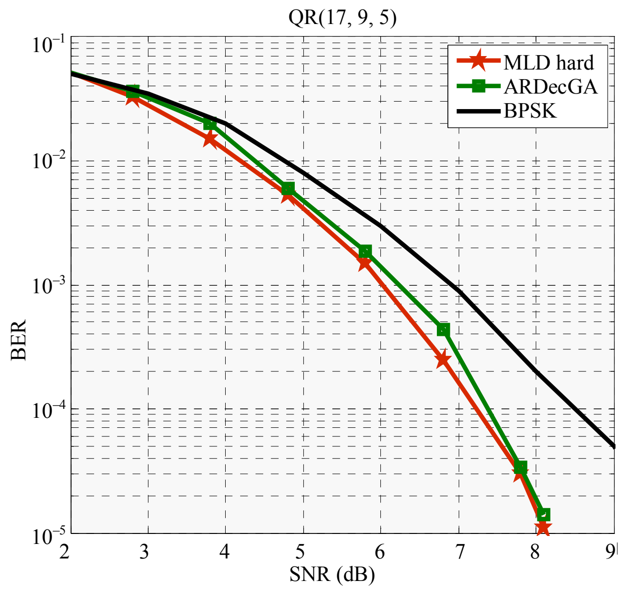

Figure 15. Comparison between ARDecGA and the hard decision MLD performances for the QR(17, 9, 5) code.

reach the error correcting performances of the MLD decoder which uses all the $2^9 = 512$ codewords.

3.5.13. Comparison between ARDecGA and the HDGA Decoder

The hard HDGA decoder [10] is one of the most recent and efficient new decoding algorithms, it uses genetic algorithms and information sets to decode linear block codes. The **Figure 16** presents a comparison between the error correcting performances of ARDecGA ($M = 300$, $N_i = 100$, $N_g = 20$, $p_c = 0.95$, $p_m = 0.03$, $N_e = 2$) and this decoder. It shows that ARDecGA and HDGA have the same performances.

3.6. Complexity of ARDec

The complexity of ARDec is variable and it depends on the following parameters:

- The parameter M.
- The code length n.
- The code dimension k.
- The weight of the error to decode.
- The complexity of the auxiliary SIHO decoder.

The **Table 3** presents an upper bound of the complexity of ARDecGA and ARDecOSD^m and it gives those of BM and HDGA algorithms. This table shows that the complexity of ARDecGA is polynomial in n however the one of ARDecOSD^m is polynomial in n^m. Both the complexity of HDGA and the one of the BM algorithms are polynomial in n^2.

The **Figure 17** shows a basic property of error correcting code in the ambient space. If h is a binary received word then there exists at most one codeword c in the sphere of radius t (error correcting capability t of the code) centered on h.

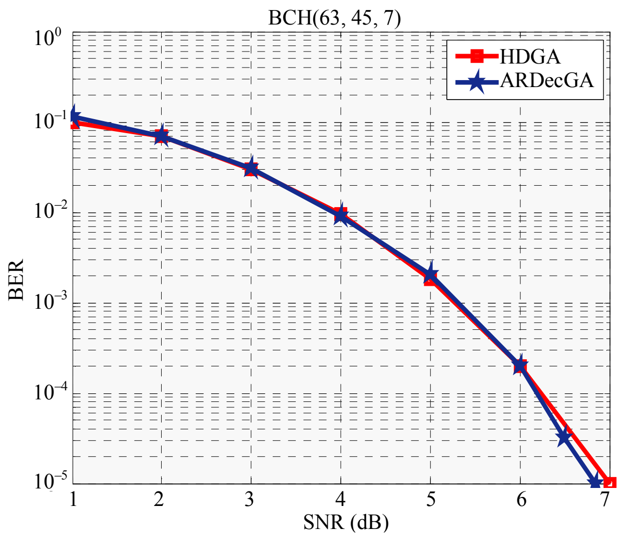

Figure 16. Comparison between ARDecGA and the hard decision HDGA performances for the BCH(63, 45, 7) code.

Table 3. Complexity of BM, HDGA and ARDec algorithms.

Decoder	Complexity
ARDecOSD^m	Less than or equal to $O\left(M \cdot n + n^{m+1}\right)$
ARDecGA	Less than or equal to $O\left(M \cdot n + N_i \times N_g \left[k_n + \log\left(N_i\right)\right]\right)$
ARDec based on SIHO decoder	Less than or equal to $O\left(M \cdot n\right) + O\left(\text{SIHO decoder}\right)$
Berlekamp-Massey	$O(n^2)$
HDGA	$O\left(N_i \times N_g \left[k_n^2 + k_n + \log\left(N_i\right)\right]\right)$

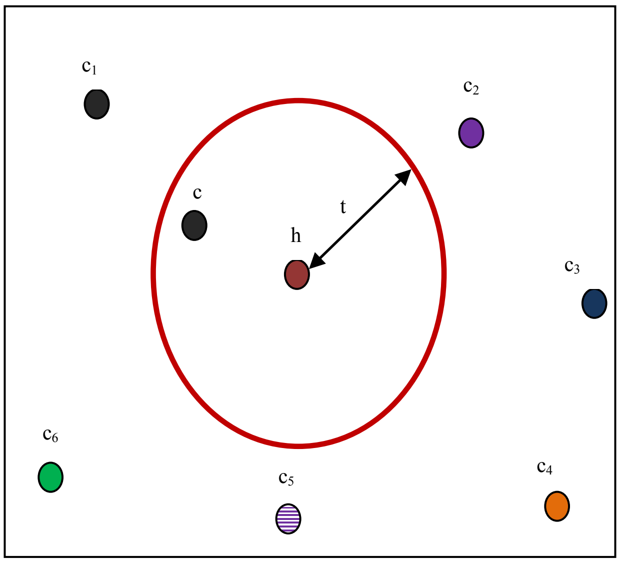

Figure 17. Basic property of error correcting codes.

In the decoding steps of h, the algorithm stops when the codeword c at Hamming distance less than or equal to t is found. This stop criterion allows reducing considerably the complexity of ARDec. The **Figure 18** shows the average number of generations required to decode errors of weight between 0 and 9 for the BCH(63, 30, 13) code of error correcting capability $t = 6$ by the ARDecGA algorithm. It shows that the errors of weight less than or equal to $t - 1$ are decoded in the first generation; the errors of weight t requires about 2.86 generations at average and the errors of weight greater than t requires the use of 100 generations because generally the stop criterion isn't verified in this case.

The **Figure 18** shows that the use of only three generations allows to correct errors of weight less than or equal to t (correctable errors) and justifies that ARDecGA has in practice a small complexity comparing to the upper bound given in the **Table 3**.

When the number of generations N_g and the population size N_i are sufficient the ARDecGA algorithm allows correcting some errors of weight more than t. The **Figure 19** shows that the use of ARDecGA with $M = 1000$, $N_g = 100$, $p_c = 0.95$, $p_m = 0.03$ and $N_i = 300$ allows the correction of about 90% of errors of weight 7 and 46% of errors of weight 8 for the BCH(63, 30, 13) code of error correcting capability equal to 6.

4. Conclusion and Perspectives

In this paper we have presented an efficient hard decision decoding algorithm which uses the dual space of a linear code C to compute artificial reliabilities of binary the received word h by a majority voting procedure. The symbols with lowest reliabilities are moved in the redundancy part of the word $\pi(h)$ by using a permutation

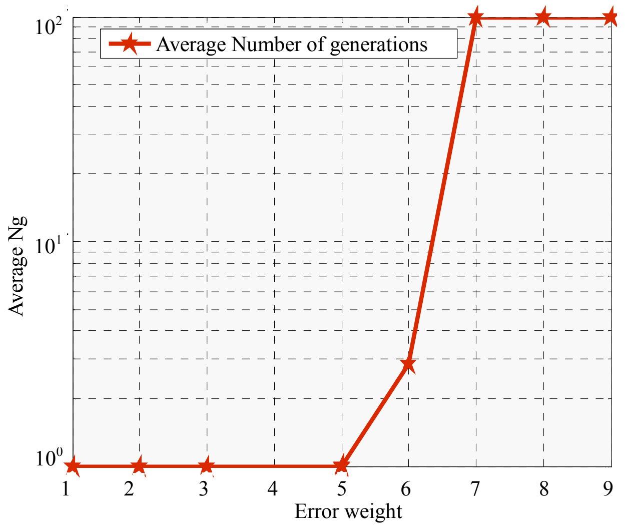

Figure 18. Average number of generations for the BCH(63, 30, 13) code.

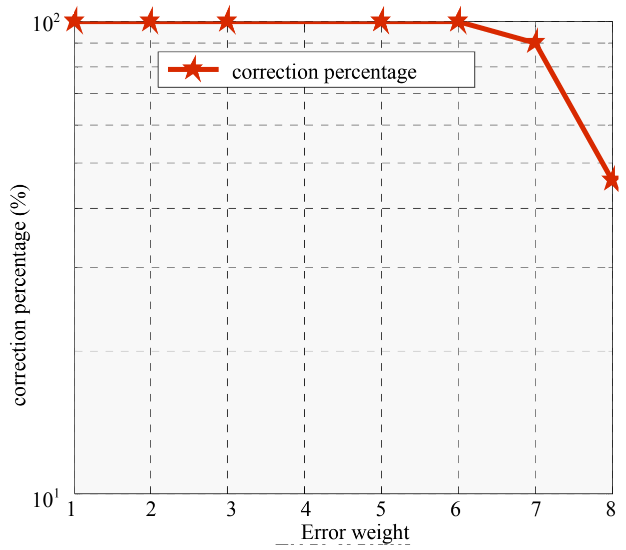

Figure 19. Correction percentage for the BCH(63, 30, 13) code.

π which binds the two codes C and $\pi(C)$. The word $\pi(h)$ contains generally a few noised symbols in its information part which can be cleaned by a genetic algorithm or by some test sequences of the OSD decoder. The simulation results show that the ARDec decoder allows to correct all errors of weight less than or equal to the error correcting capability of the code C. We have showed by simulation that more correcting capability can be gained by using a vote procedure and a powerful SIHO decoder.

REFERENCES

[1] G. C. Clarck and J. B. Cain, "Error-Correction Coding for Digital Communication," Plenum, New York, 1981.

[2] S. Nouh and M. Belkasmi, "Genetic Algorithms for Finding the Weight Enumerator of Binary Linear Block Codes," *International Journal of Applied Research on Information Technology and Computing*, Vol. 2, No. 3, 2011, pp. 80-93.

[3] D. Chase, "A Class of Algorithms for Decoding Block Codes with Channel Measurement Information," *IEEE Transactions on Information Theory*, Vol. 18, No. 1, 1972, pp. 170-181. doi:10.1109/TIT.1972.1054746

[4] J. L. Massey, "Shift-Register Synthesis and BCH Decoding," *IEEE Transaction on Information Theory*, Vol. 15, No. 1, 1969, pp. 122-127.

[5] E. R. Berlekamp, "Algebraic Coding Theory," Aegean Park Press, Walnut Creek, 1984.

[6] F. J. MacWilliams, "Permutation Decoding of Systematic Codes," *The Bell System Technical Journal*, Vol. 63, No. 1, 1964, pp. 485-505.

[7] S. Nouh, M. Askali and M. Belkasmi, "Efficient Genetic Algorithms for Helping the Permutation Decoding Algorithm," *International Conference on Intelligent Systems*, Mohammedia, 16-17 May 2012.

[8] R. G. Gallager, "Low-Density Parity-Check Codes," *IRE Transactions on Information. Theory*, Vol. 8, No. 1, 1962, pp. 21-28. doi:10.1109/TIT.1962.1057683

[9] R. H. Morelos-Zaragoza, "The Art of Error Correcting Coding," 2nd Edition, John Wiley & Sons, Hoboken, 2006.

[10] Azouaoui, I. Chana and M. Belkasmi "Efficient Information Set Decoding Based on Genetic Algorithms," *International Journal of Communications, Network and System Sciences*, Vol. 5, No. 7, 2012, pp. 423-429.

[11] R. Sujan, *et al.*, "Adaptive 'Soft' Sliding Block Decoding of Convolutional Code Using the Artificial Neural Net-Work," *Transactions on Emerging Telecommunications Technologies*, 2012.

[12] M. Sayed, "Coset Decomposition Method for Decoding Linear Codes," *International Journal of Algebra*, Vol. 5, No. 28, 2011, pp. 1395-1404.

[13] M. Kerner and O. Amrani, "Iterative Decoding Using Optimum Soft Input—Hard Output Module," *IEEE Transactions on Communications*, Vol. 57, No. 7, 2009, pp. 1881-1885. doi:10.1109/TCOMM.2009.07.070167

[14] B. Cristea, "Viterbi Algorithm for Iterative Decoding of Parallel Concatenated Convolutional Codes," *Proceedings of 18th European Signal Processing Conference*, Aalborg, 23-27 August 2010.

[15] C. R. P. Hartmann and L. D. Rudolph, "An Optimum Symbol-by-Symbol Decoding Rule for Linear Codes," *IEEE Transactions on Information Theory*, Vol. 22, No. 5, 2009, pp. 514-517.

[16] D. E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning," Addison Wesley, Reading, 1989.

[17] J. McCall, "Genetic Algorithms for Modelling and Optimizationm" *Journal of Computational and Applied Mathematics*, Vol. 184, No. 1, 2005, pp. 205-222. doi:10.1016/j.cam.2004.07.034

[18] H. Maini, K. Mehrotra, C. Mohan and S. Ranka, "Soft Decision Decoding of Linear Block Codes Using Genetic

Algorithms," *IEEE International Symposium on Information Theory*, Trondheim, 27 June-1 July 1994.

[19] M. P. C. Fossorier and S. Lin, "Soft Decision Decoding of Linear Block Codes Based on Ordered Statistics," *IEEE Transactions on Information Theory*, Vol. 41, No. 5, 1995, pp. 1379-1396. doi:10.1109/18.412683

[20] J. S. Yedidia, J. Chen and M. Fossorier, "Generating Code Representations Suitable for Belief Propagation Decoding," Tech. Report TR-2002-40, Mitsubishi Electric Research Laboratories, Broadway, 2002.

[21] J. S. Yedidia, J. Chen and M. Fossorier, "Representing Codes for Belief Propagation Decoding," *Proceedings of IEEE International Symposium on Information Theory*, Yokohama, 29 June-4 July 2003, p. 176.

[22] A. Azouaoui, M. Askali and M. Belkasmi, "A Genetic Algorithm to Search of Good Double-Circulant Codes," *IEEE International Conference on Multimedia Computing and Systems Proceeding*, Ouarzazate, 7-9 April 2011, pp. 829-833.

[23] J. L. Massey, "Threshold Decoding," M.I.T. Press, Cambridge, 1963.