# Improved Decoding of linear Block Codes using compact Genetic Algorithms with larger tournament size

**Ahlam Berkani[1] , Ahmed Azouaoui[2], Mostafa Belkasmi[3] and Bouchaib Aylaj[4]**

**[1] Mohammed V-Rabat University, SIME labo, ENSIAS
Rabat, Morocco**

**[2] FS - El Jadida, Chouaib Doukkali University
El Jadida, Morocco**

**[3] Mohammed V-Rabat University, SIME labo, ENSIAS
Rabat, Morocco**

**[4] FS - El Jadida, Chouaib Doukkali University
El Jadida, Morocco**

**Abstract**
Soft-decision decoding is a very important NP-hard problem for developers of communication systems. In this work we propose two new dual domain soft decision decoders that use compact Genetic Algorithm (cGA) with larger tournament size: the first algorithm investigates tournament selection with larger size using mutation, and the second employs higher selection pressure with randomly generated individuals. The obtained results are compared to known previous works and show the effectiveness of using larger tournament size in dual domain soft decision decoding problem. Behind performances analysis, a complexity study is done which shows that both proposed decoders are not very complex in comparison with the standard compact Genetic Algorithm based decoder (cGAD).
***Keywords:*** *compact Genetic Algorithm, dual domain Soft decision-decoding, higher selection pressure, mutation, optimization, tournament size, updating step size.*

## 1. Introduction

In digital communication, one of the major problems is how to deliver the message of the source to the destination as faithfully as possible. Error correcting codes are a method among others, used to insure reliable transmission, by removing errors due to channel noise. Indeed, error correcting codes are used to improve the reliability of the transmitted data over communication channels susceptible to noise. During data transmission, error-correcting codes are used twice: for coding and decoding. Coding techniques create code words in the emission by adding redundant information to the initial message, and decoding methods try to find in the reception the nearest code word to the received message (Figure 1).

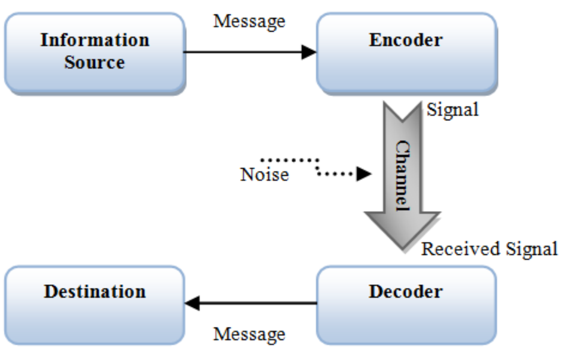

Fig. 1 Simplified model of communication systems

Decoding algorithms are divided in two categories: hard decision and soft decision. Hard decision decoding works with the binary form of the received word. Soft decision one works directly on the received symbols [10]. Soft-decision decoding is an NP-hard problem and has been approached in different ways.
Artificial intelligence techniques were recently introduced to solve this problem. Among the different methods used, Genetic Algorithms (GAs) [14] have taken a big interest in decoding problems like many other scientific areas, and was used in many optimization problems. To our knowledge, Maini et al. were the first to introduce *GAs* in the soft-decision decoding of linear bloc codes [16]. Cardoso et al have also published an influential work on hard-decision decoding of linear bloc codes using *GAs* [8]. Belkasmi's team proposed in 2012 [2] a new decoder

based on *GAs* which deals with the dual code unlike other genetic decoders that use the code itself. To find the optimum solution, the standard *GAs* has a high processing overhead for generations and manipulations of populations. It also needs to store all the population's individuals in memory, and the lack of simplest way to initialize different operators and stopping criterions make it impractical to solve real-time optimization problems. Recently, a number of algorithms, called Probabilistic Model Building Genetic Algorithms (*PMBGAs*); or Estimation of Distribution Algorithms (*EDAs*) have been developed to replace the *GA*'s population and crossover operator with a probabilistic representation and generation method [4] [11] [19] [17] [13]. The *EDAs* are divided into three categories based on the interdependencies between the variables of solutions [15]: the no dependencies model, the bivariate dependencies model and the multiple dependencies model.
The compact Genetic Algorithm (*cGA*) [13], is one of the simplest algorithms of the first category, it generates offspring population according to an estimated probabilistic model of parent population instead of using traditional genetic operations. It only needs a small amount of memory which makes it useful for memory-constrained applications. Shakeel first introduced this algorithm for soft-decision decoding using the code itself [18]: his decoder uses the generator matrix of the code; this fact makes the decoding very complicated for high rate codes. We later used this algorithm for dual domain soft-decision decoding of linear codes [3], this decoder is less complex than Shakeel decoder for codes with a high rate ($n-k<<n$), where $n$ is the code length and $k$ its dimension.
In this work, we propose two new dual domain soft decision decoders that investigates tournament with larger size in the compact Genetic Algorithm using mutation and higher selection pressure, and then, study their performance, compare them with others known decoders and study their complexity. This paper is organized as follows: section 2 gives a description of the standard compact Genetic Algorithm, presents the tournament with larger size and how we can introduce it in the standard compact Genetic Algorithm. Section 3 develops the new Dual Domain Soft-Decision Decoders based on larger tournament size using mutation and higher selection pressure. Performance analysis is presented in section 4, and the complexity one is presented in the section 5. The paper concludes with a summary of the results in Section 6.

## 2. cGA with larger tournament size

### 2.1 The compact Genetic Algorithm

Harik et al presented in 1998 [13] the Compact Genetic Algorithm which is one of the simplest EDAs. Instead of dealing with natural operators of GAs, this algorithm uses a probability vector (p). During the processes of optimization, binary solutions are randomly generated from p at each generation, after that, they compete based on their fitness values. p is then updated based on these solutions until reaching the convergence state known when all positions of p became binary (0 or 1). This state of convergence represents the final solution of the optimization problem. The cGA represents the population as a probability vector over a set of solutions and operationally mimics the order-one behavior of the simple GA with the uniform crossover. The cGA has an advantage of using a small amount of memory, and then, may be quite useful in memory-constrained applications. Algorithm 1 describes the pseudo-code of the cGA.

**Algorithm 1:** Pseudo code of the compact Genetic Algorithm

```
Require $l$ the length of $p$;
Require $1/\lambda$: The updating step size.
1. Initialize the probability vector $p$ which length is $l$
for $m$ := 1 to $l$ do $p[m]$ := 0:5
end for
2. Generate two individuals from $p$
$a$ := generate(p);
$b$ := generate(p);
3. Let them compete
winner; loser := compete(a; b);
4. Update $p$ towards the better one
for $m$ := 1 to $l$ do
  if $winner[m] \neq loser[m]$ then
    if $winner[m] == 1$ then $p[m] := p[m] + 1=1/\lambda$
    else $p[m] := p[m] - 1=1/\lambda$
    end if
  end if
end for
5. Check if the vector $p$ has converged
for $m$ := 1 to $l$ do
  if $p[m] > 0$ and $p[m] < 1$ then return to step 2;
  end if
end for
6. The probability vector p represents the final solution.
```

### 2.2 Larger tournament size using mutation

#### 2.2.1 Mutation operator

Mutation is an operator referring in nature to an "accident" that changes DNA and adds a new allele which does not exist before. In GAs, its concept is quite simple: a gene is chosen randomly and its allele is changed. In the case of binary strings, which we are interested on in this work, mutation can be made by complementing the chosen bit. Alone, the mutation does not generally advances the search for a solution, but it insures the evolution of population and avoids the development of a uniform one.

### 2.2.2 Using mutation to generate individuals

This version of *cGA* differs from the standard version at the level of competition's vectors generation. Instead of using only two vectors *a* and *b*, this version uses a number of vectors defined according to a parameter *s*. We generate randomly two binary vectors *a* and *b* from *p*, we mutate *b* in such way of generating (*s-1*) other vectors. We select the best according to the objective function between *b* and (*s-1*) resultant vectors of the mutation, then we let it compete with the vector *a*. This version of the compact Genetic Algorithm is going to replace stages 2, 3, and 4 of the standard version by the stages given in Algorithm 2.

**Algorithm 2:** Modification of the cGA to implements s vectors of competition using mutation.

```
1. Generate two individuals from p
a := generate(p);
b := generate(p);
2. Mutate b to generate (s - 1) binary vectors
3. Determine the vector with the best value of fitness
among b and (s - 1) other vectors and to name it N;
4. Compete(a;N )
winner; loser := compete(a;N );
5. Update p towards the winner.
```

## 2.3 Larger tournament size using randomly generated vectors with Higher Selection Pressure

The selection operator helps to give more copies to better individuals. But it does not always help to keep the better genes, because the evaluation is done in the context of the individual altogether. Selection pressure controls the selection of individuals from one population to the next. It gives individuals of higher quality, higher probability of being used to create the next generation, and so the algorithm will focus on promising regions in the search space. In other words, higher selection pressure may play the role of memory to store best candidates' solution seen during the search progress. It can be insured with the cGA by introducing elitism [1], or by using tournament selection with increased size [13].
Harik et al introduced the simulation of higher selection pressure by increasing the tournament size in [13], instead of mimicking the pair-wise tournament selection in the standard version of cGA. Tournament with size s is implemented with the cGA by:
Generating s individuals from the probability vector and find out the best one;
Let the best individual compete with the others s-1 individuals, updating the probability vector along the way.
In order to simulate higher selection pressure, steps 2 to 4 of the cGA's pseudo-code (Algorithm 1) have to be replaced by the ones shown in Algorithm 3.
Elitism strategy is also a method to implement higher selection pressure. It is one of the operational characteristics of Genetic Algorithms (GA) that provides a reduction of genetic drift by insuring that the best candidate solution is allowed to contribute to the next generation. That is, the best chromosome is transferred to the next generation without alteration. This one gets the full chance to compete with the other candidates of the next generation. C. A. Wook and R. S. Ramakrishna introduced two elitism-based cGAs: the persistent-elitist cGA and the non persistent-elitist one [1].

**Algorithm 3:** Modification of the *cGA* using *s* vectors of competition

```
Require: l the lenght of p
Require: 1/λ, The updating step size
1. Generate s individuals from the probability vector p, and
store them in S :
for i := 1 to s do S[i] := generate(p);
end for
2. Rearrange S so that S[1] is the fittest individual ;
3. Let S[1] compete with the others individuals :
for j := 1 to s do
    winner; loser := compete(S[1]; S [j ]);
    update p (step 4 of the cGA)
end for
```

## 3. Decoding algorithms using larger tournament size

The two versions of *cGA* decoders presented in this work try to solve the dual domain soft-decision decoding of linear error correcting codes. Dual domain Soft-decision decoding is firstly modeled as an optimization problem and then a description of the two decoding algorithms is given.

### 3.1 Dual domain Soft-decision decoding as an optimization problem

The maximum-likelihood soft-decision decoding problem of linear block codes is an NP-hard problem [7], and can be stated as follows: Given the received vector *r* and the parity-check matrix *H*, let $S = w*H^T$ be the syndrome of *w*, where $w$ is the hard decision of $r$, $H^T$ is the transpose of matrix $H$, and $E(S)$ be the set of all errors patterns whose syndrome is $S$, find the $E \in E(S)$ which minimizes correlation discrepancy:

$$\boldsymbol{F_r(E)} = \sum_{j=1}^{n}(\boldsymbol{E_j}|\boldsymbol{r_j}|) = \sum_{j=1, E_j=1}^{n}(|\boldsymbol{r_j}|) \quad (1)$$

Where $n$ is the code length. The optimization problem (1) has $n$ error variables, out of which only ($n$-$k$) are independent, where $k$ is the code dimension. Using the algebraic structure of the code, the remaining $k$ variables can be expressed as a function of these ($n$-$k$) variables.
An individual is a set of $k$ bits. Let $E'$ be an individual, $S1$ be a ($n$-$k$)-tuple such that $S1 = E' \times A$ where $A$ is a sub matrix of $H'$, and $S2$ be a ($n$ - $k$)-tuple such that $S2 = S + S1$. We form the $E$ error pattern such that $E = (E', E'')$ where $E'$ is the chosen individual and $E'' = S2$. Then $w + E$ is a code word. The fitness function is the correlation discrepancy between the permuted received word and the estimated error ($E$) such indicated in (1).
Until now, just few numbers of researches treat the soft-decision decoding as an optimization problem because it is identified as a NP-hard problem [7], and also because optimization techniques generally solve a large problems once when the soft-decision decoding occurs repeatedly in communication systems.

### 3.2 Description of decoders

Let $C$ denotes, binary linear block code of length $n$, dimension $k$ and minimum Hamming distance $d$, defined over the Galois field of order 2 ($GF$ (2)). Let $H$ be its parity check matrix, and ($r_i$), $1 \leq i \leq n$ be the received sequence over a communication channel with noise variance $\sigma^2 = N_0/2$, where $N_0$ is noise power spectral density. We assume that codewords are modulated by a *BPSK* modulator and transmitted over an *AWGN* channel. Once the message received, the demodulator makes hard-decisions $wi$, $i = 1, ..., n$ :

$$w_i = \begin{cases} 1, if \ \ r_i \geq 0 \\ 0, if \ \ r_i < 0 \end{cases} \quad (2)$$

The receiver then calculates the syndrome $w \times H^T$ and accepts $w$ as the most likely transmitted codeword if the syndrome $w \times H^T = 0$. If the syndrome $w \times H^T \neq 0$, the soft-decision decoding process begins as described in algorithms 4 and 5 : the first one implements higher selection pressure with randomly generated vectors *cGAD-HSP*, and the second one uses mutation to have tournament with larger size *cGAD-M*.

**Algorithm 4:** cGAD-HSP

**Require:** k, The length of p

**Require:** $1/\lambda$, The updating step size

**Step 1:** Sorting the values of sequence r in decreasing order of reliability $|ri|$. Further, permute the coordinates of $r$ to ensure that the last ($n$ - $k$) positions of $r$ are the least reliable linearly independent positions. Call this vector $r'$ and let $\pi$ be the permutation related to this vector ($r' = \pi(r)$). Apply the permutation $\pi$ to $H$ to get a new check matrix $H' = \pi (H) = [AI_{n-k}]$.

**Step 2:** Map $r'$ onto probability vector $p$; $p \in R^k$. The probability vector $p$ defines the starting point for the genetic Search over the $k$-dimensional vector space $F^k{}_2$. It is expected that this search would terminate (converge) at a vertex of the $k$-dimensional hypercube. An obvious starting point is the center point of the search space, i.e. $p = (0:5)^k$. However, the search time and complexity can be greatly reduced if the search is initiated from a point close to the solution vector.
The following steps describe a method that uses soft information of the received vector to determine a starting point close to the optimum solution :

1. $z$ is the vector formed by the $k$ first components of $r'$;
2. Normalize $z$ using $\frac{|z|}{max\ |z|}$ so that $0 \leq z \leq 1$ ;
3. Since we are looking for the error pattern, we map the normalized values of $z$ to the probability vector $p$ using the transformation :

$$p_i = 1 - z_i \ ; \ i = 1; 2; \ldots ; k \quad (3)$$

**Step 3:** Generate $s$ binary random vectors and store them in $S$; $S[j] \in F^k{}_2$ following probabilities:

$$P\ (S[j] = 1) = p; \qquad P\ (S[j\,] = 0) = 1 - p \quad (4)$$

An uniform random number generator $U\ (0;\ 1)$ can be used to generate these vectors as follows:

$$S[i][j] = \begin{cases} 1, if\ U(0,1) \leq p_i \\ 0, otherwise \end{cases} \quad (5)$$

**Step 4:** Rearrange $S$ so that $S[1]$ is the fittest individual. This step evaluates the fitness values of each individual $S[j]$ using the objective function (1), and then rearrange $S$ so that $S[1]$ will be the fittest one.

**Step 5:** Let $S[1]$ compete with the $(s - 1)$ others individuals updating the probability vector along the way. Clearly, the best individual wins all the competitions. The update strategy is done following rules:

```
 If $S[1][i + 1] = 1$ and $S[j][i + 1] = 0$,
   then $p_{i+1} = p_i + 1/\lambda$
 If $S[1][i + 1] = 0$ and $S[j][i + 1] = 1$,
   then $p_{i+1} = p_i - 1/\lambda$
 If $S[1][i + 1] = S[j][i + 1]$,
   then $p_{i+1} = p_i$
 The updating step size $1/\lambda$ is a user defined parameter which is directly related to the performance of the decoder [3].
Step 6: Convergence attained? Here the algorithm checks if all $p_i$ are equal to 0 or 1. If not, it goes to step 3, but if yes, the converged p gives the final solution for the objective function (1).
Step 7: Encode $p$ and apply inverse permutation of $\pi$.
```

**Algorithm 5:** cGAD-M

```
Require: k, The length of p
Require: $1/\lambda$, The updating step size
Step 1: The same as Step 1 in algorithm 4
Step 2: The same as Step 2 in algorithm 4
Step 3: Produce a pair of binary random vectors $a$ and $b$. The pair of vectors is produced with the following probability:
        $P(a_i ; b_i = 1) = p_i ;$
        $P(a_i ; b_i = 0) = 1 - p_i$        (6)
An uniform random number generator $U(0; 1)$ can be used to generate these vectors as function (5) where $a_i$ and $b_i$ will be calculated the same as $S[i][j]$. Once $a$ and $b$ created, we mute b to create $(s - 1)$ binary random vectors. This group of vectors belongs to $F^k_2$.
Step 4: Estimate the objective function $F_r$ (1). This stage estimates the fitness values of the vector $b$ and $(s - 1)$ generated vectors by using the objective function (1). Once the vector with the best value of fitness is determined, we note it $N$ and we let it compete with $a$. The vector with the best fitness (between $a$ and $N$) is identified as the winner whereas the vector with the worst one is identified as the loser.
Step 5: Update of the probability vector $p$. The vector of probability is updated towards the winner using the previous rules (step5 of algorithm 4). The size of updating step $1/\lambda$ is a parameter defined by the user. It directly affects the performances of the decoder.
Step 6: Convergence attained? Here the algorithm checks if all $p_i$ are equal to 0 or 1. If not, it goes to step 3, but if yes, the converged p gives the final solution for the objective function (1).
Step 7: Encode $p$ and apply inverse permutation $\pi$.
```

## 4. Performance analysis

In order to show the effectiveness of the new decoders in Dual Domain Soft-Decision Decoding, we carry out intensive simulations. For transmission we used an *AWGN* channel with a *BPSK* modulation. The simulations were made with default simulation's parameters outlined in Table 1 and the default cGA parameters in Table 2. The performance is given in terms of bit error rate (*BER*) as a function of Signal to Noise Ratio (*SNR*).

Table 1: Simulations Parameters

| ***Simulation parameter*** | ***Value*** |
|---|---|
| Channel | AWGN |
| Modulation | BPSK |
| Minimum number of transmitted blocks | 1000 |
| Minimum number of residual bit errors | 200 |
| Default code | BCH(63,45,7) |

Table 2: cGA Parameters

| ***cGA parameter*** | ***Value*** |
|---|---|
| λ | 500 |
| s | 13 |
| Mutation rate (pm) | 0.04 |

### 4.1 The gain of larger tournament size in Dual Domain Soft-Decision Decoding

In order to show that tournament with larger size improves the decoding performance, in term of performance, we chose binary Golay[1] (23, 12) code and Figure 2 shows the results. From this Figure we can see that the *cGAD* with higher selection pressure using randomly generated individuals (*cGA-HSP*) presents best performances than the one using mutation (*cGA-M*). At *BER* value equal to $5.10^{-5}$ *cGA-HSP* offers a gain of 0.6 dB, when *cGA-M* gives the same performance as the standard *cGAD* presented by the green curve (*s*=2).

[1]**Golay codes :** A family of perfect linear error-correcting block codes.

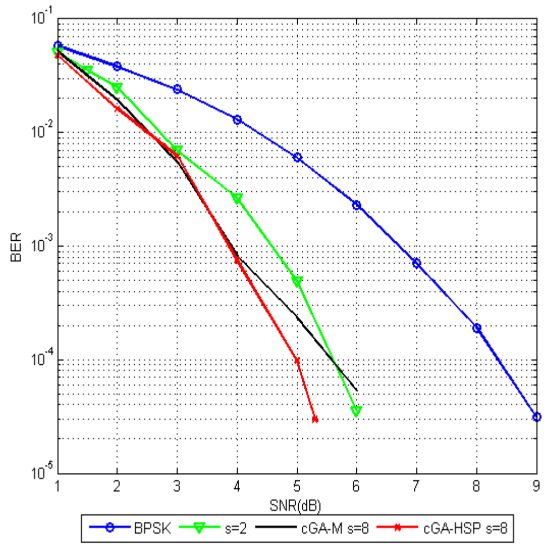

Fig. 2 The gain of larger tournament size on performance for Golay (23,12) code.

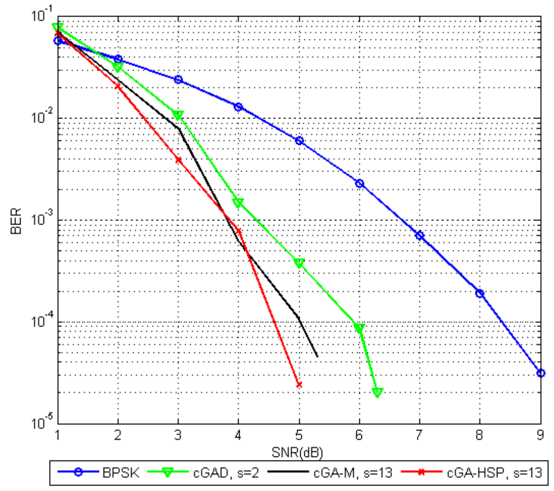

Fig. 3 The gain of larger tournament size on performance for BCH(63,45,7) code.

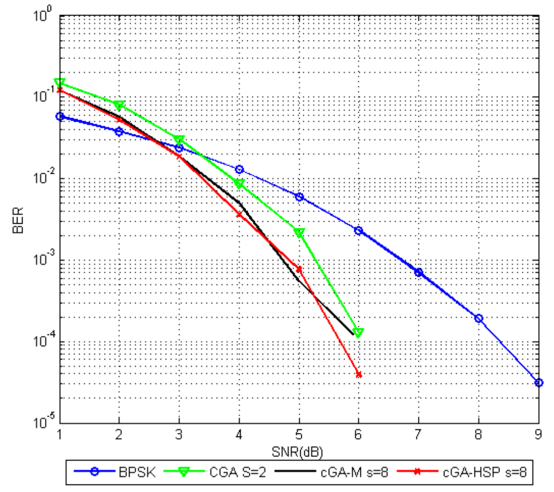

Fig. 4 The gain of larger tournament size on performance for BCH(127,64,21) code.

We compare afterwards the decoder based on standard compact Genetic Algorithm that uses only two competition's vectors (*s*=2), and the proposed decoders.
For BCH[1] (63,45,7), As we can see in the Figure 3, the new decoders outperform the standard one (a gain of 0.9 dB at *BER* value $4,5.10^{-5}$ for *cGA-M* and a gain of 1.2 dB at $2.10^{-5}$ for *cGA-HSP*) which proves the efficiency of the proposed Decoders.
The simulations of the BCH(127,64,21) code show that both decoders *cGA-HSP* and *cGA-M* are slightly better than the standard *cGAD* as illustrated in Figure 4.
We do the same thing for the code QR[2] (71,36) and as expected, both versions improve the decoding performance. The Figure 5 shows that *cGA-HSP* gives a gain of 1 dB at $4.10^{-5}$, where *cGA-M* gives a gain of about 1.3 dB at $7.10^{-5}$.
For the binary form of the code RS[3] (15,7,9), the *cGA-M* has a gain of 0.4 dB in comparison with *cGAD*, where *cGA-HSP* outperforms *cGAD* by a gain of 0.8 dB at *BER* value of $6.10^{-5}$, the listed results are shown in Figure 6.

[1] **BCH codes :** Abbreviation of " Bose-Chaudhuri-Hochquenghem " code. A family of cyclic error correcting codes.
[2] **QR codes :** Abbreviation of " Quadratic Residue" codes. A type of cyclic codes.
[3] **RS codes:** An Abbreviation of "Reed Solomon" codes. A family of codes that belongs to the class of non- binary cyclic error-correcting codes.

## 4.2 The effect of tournament size

After showing the impact of selection with increased size on the improvement of the *cGA* decoder's performance, we illustrate in Figures 7 and 8 the effect of tournament size (*s*).
As shown in Figure 7, the size of tournament has a considerable effect on the *cGA-HSP* decoder's performance. Applied to the BCH(63,45,7) code, a tournament of size *s* = 3 gives approximately the same performance as the standard *cGAD*. The performances get better along the augmentation of *s* value until *s* = 13: with

tournament of size 13 the gain is about 1.2 dB at $2.10^{-5}$. Simulating tournament of size equal to $s$ = 20 reduces them. The degradation caused by a bigger tournament size can be interpreted by a premature convergence toward local optimum. So we should use the tournament size carefully so as to get the better performances of a given code.

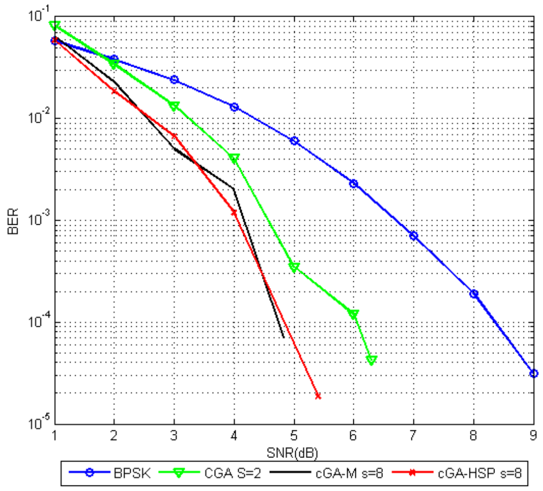

Fig. 5 The gain of larger tournament size on performance for QR(71,36) code.

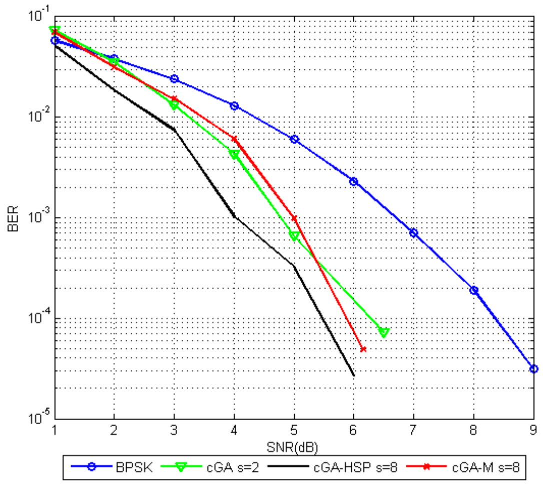

Fig. 6 The gain of larger tournament size on performance for the binary form of RS (15,7,9) code.

Figure 8 shows that also for the *cGA-M* decoder; the tournament size has an effect in improving the decoding performance like the *cGA-HSP*. When increasing the tournament size to s = 3, we reach approximately a gain of 0.7dB at SNR=$4.10^{-5}$. Comparative performance is obtained when tournament size is set to $s$ = 5 and $s$ = 8. The gain in *BER* is elevated to 1dB when $s$ = 13.

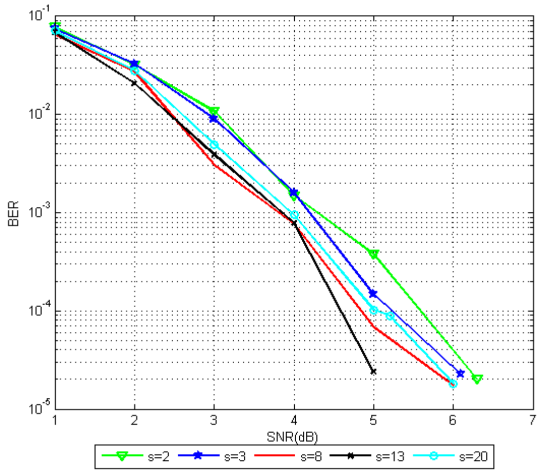

Fig. 7 The effect of tournament size with cGA-HSP on BCH code (63,45,7).

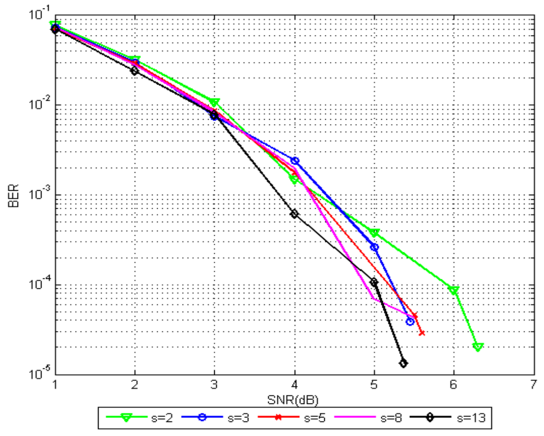

Fig. 8 The effect of tournament size with cGA-M on BCH(63,45,7) code.

## 4.3 The effect of the updating step size

The updating step size *1/λ* is a user defined parameter which is directly related to the performance of the decoder. This part shows the impact of increasing or decreasing this parameter. Figure 9 gives an illustration about the effect of decreasing the *1/λ* parameter applied to BCH(63,45,7) code. We can notice that the performance of our decoder get better when decreasing the size until reaching the value *1/λ* = 0.002, after that such a stabilization is remarked even if the size of 1/λ is decreased.
Figure 10 shows also that with $s$ = 20, the performances get better when decreasing the updating step size.

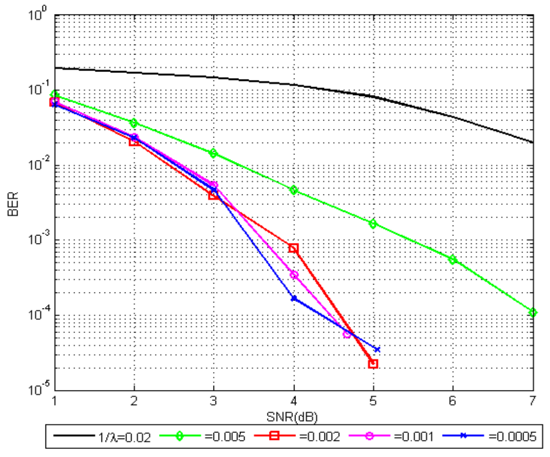

Fig. 9 The effect of the updating step size for cGA-HSP on BCH (63,45,7) code for s = 13.

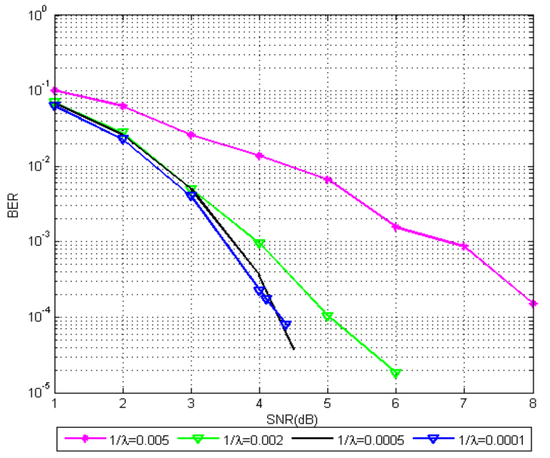

Fig. 10 The effect of the updating step size for cGA-HSP on BCH (63,45,7) code for s = 20.

The comparison of the updating step size decrease applied for tournament size $s$ = 13 and $s$ = 20 is made in Figure 11. It shows that for $s$ = 13, the decrease of the parameter $1/\lambda$ does not impact the performance after the value $1/\lambda$ =0.002. In contrast, for tournament size $s$ = 20 the same value of $1/\lambda$ improves the performances: for SNR= $4.10^{-5}$ a gain of 0.5dB is noticed between $s$ = 13 and s = 20 for $1/\lambda$= 0.0005. From this Result we can say that the updating step size has also a big effect, in addition to tournament size, on the improvement of the decoder performance. When the tournament size get bigger, the $1/\lambda$ parameter should be the smallest possible to have the best performance of the decoder.

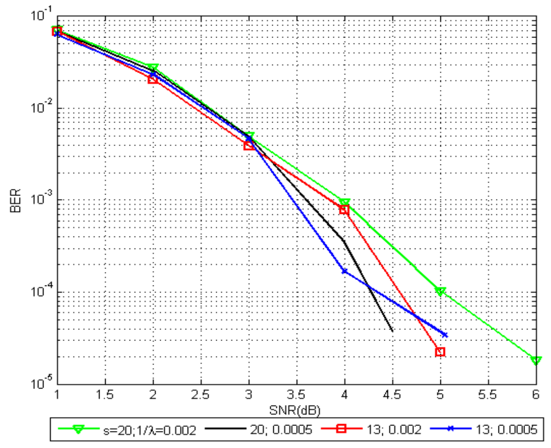

Fig. 11 The effect of the updating step size on the performance's improvement to BCH(63,45,7) code for the cGAD-HSP decoder.

## 4.4 The effect of mutation rate

In this part, we study the effect of the mutation rate into decoding performance of *cGA-M*, for a fixed tournament selection and updating step size ($s$ = 13, $\lambda$ =500). As shown in the Figure 12, when increasing the mutation rate from $pm$ = 0.001 to $pm$ = 0.01, we reach a gain of 0.8 dB at $3.10^{-5}$. After that, even if we increase to $pm$ = 0.04 we don't realize any gain.

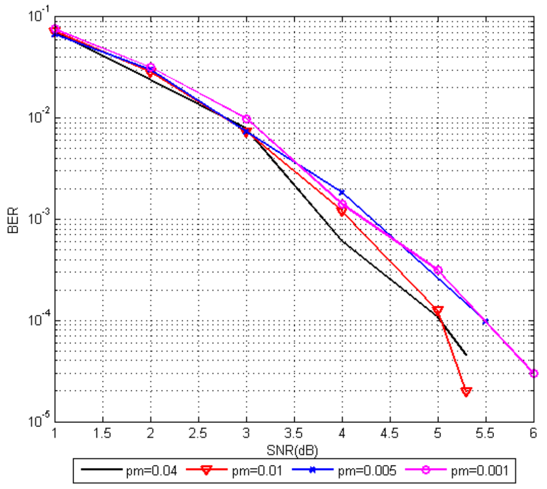

Fig. 12 The effect of mutation rate in cGA-M on BCH(63,45,7) code for s=13.

## 4.5 Comparison of the new decoders with other works

After showing the improvement of decoding performance using larger tournament size, and studying the effect of

tournament size, the updating step size and the mutation rate, we compare now our new decoders with previous ones such as *Chase-2* algorithm [9], *OSD-1* [12], and *DDGA* [2] decoders in term of performance.
Figure 13 shows the result of the performance's comparison for the code BCH(63,45,7). As we can see there, the standard *cGAD* presents the worst performance. *cGA-M* is better than *Chase-2* decoder, and the standard *cGAD*. *cGA-HSP* is better than *Chase-2*, and *OSD-1* decoders, its performance is equivalent than *DDGA* decoder. These results show that this new decoders effectively improve the decoding performance and so, the presented algorithms can be efficient for other optimization problems.

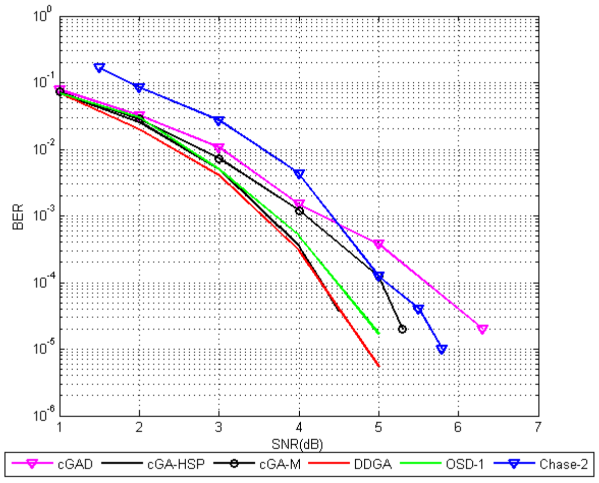

Fig. 13 Comparison of cGA-HSP and cGA-M with other decoders for BCH(63,45,7) code.

## 5. Complexity analysis

After showing that the new decoder based on Higher Selection Pressure (*cGA-HSP*) gives the best performances in comparison with the standard *cGA* based decoder (*cGAD*) and the mutation based one (*cGA-M*), we will compare the three decoders in term of their complexities. For this purpose, we will start by counting the number of iterations required by each decoder to reach the convergence then we will calculate the analytic complexity for them.

### 5.1 The average number of iterations to reach convergence

Table 3 shows the necessary average number of iterations to reach convergence for the three decoders: *cGAD*, *cGA-HSP* and *cGA-M*. As we can see there, *cGAD* and *cGA-M* has an average number of iterations necessary to convergence ($N_i$) around 20000, where *cGA-HSP* needs just around 2500 iterations. Having into account that *s*=13, if we try to calculate the average number of updates occurred during every decoding process (steps 5 of algorithms 4 and 5), we find that *cGAD* and *cGA-M* do around 20000 ones since any of these algorithms do just one update per iteration. *cGA-HSP* updates the probability vector (*s* - 1) times at each iteration, so the average number of updates for *cGA-HSP* is (12*2500) around 30000 updates. So we can conclude from this analysis that *cGA-HSP* is little more complex in term of the number of updates than *cGAD* and *cGA-M*.

### 5.2 The analytic complexity of the new decoders

Table 4 gives the analytic complexity of the three decoders: *cGA*, *cGA-M*, and *cGA-HSP*. It shows that the proposed decoders have the same complexity if we neglect the tournament size *s*, given that *s* is very small in comparison with the average number of iterations to reach convergence ($N_i$).

Table 3: The average number of iterations to reach convergence for the case of BCH(63,45,7).

| ***SNR*** | ***Average Number Of Iterations*** | | |
|---|---|---|---|
| | ***cGAD*** | ***cGAD-HSP*** | ***cGAD-M*** |
| 1 | 21166 | 2677*12 | 22144 |
| 3 | 19953 | 2576*12 | 17778 |
| 5 | 20401 | 2696*12 | 17230 |

Table 4: The analytic complexity of the new decoders

| ***Decoder*** | ***Complexiy*** |
|---|---|
| cGAD | $O(N_i.k(n-k))$ |
| cGA-HSP | $O(N_i.s.k(n-k))$ |
| cGA-M | $O(N_i.s.k(n-k))$ |

## 6. Conclusion

In this paper, we presented two new dual domain soft decision decoders that use *cGA* with larger tournament size: the first algorithm investigates tournament selection with larger size using mutation, and the second employs higher selection pressure with randomly generated individuals. A study of the new decoders has been done and showed that the introduction of mutation and higher selection pressure improves the performance of the decoding. The obtained results are compared to known previous works and show the superiority of the proposed decoders over other existing one. Also a complexity study has been done where it shows that the algorithms used for decoding are not very complex in comparison with the standard compact Genetic Algorithm based decoder (*cGAD*).

## References

[1] Chang Wook Ahn and R. S. Ramakrishna, “Elitism-based compact genetic algorithms,” IEEE Transactions on Evolutionary Computation, vol. 7, no. 4, pp. 367–385, Aug. 2003.

[2] Azouaoui, M. Belkasmi, and A. Farchane, “Efficient Dual Domain Decoding of Linear Block Codes Using Genetic Algorithms,” Journal of Electrical and Computer Engineering, vol. 2012, pp. 1–12, 2012.

[3] Azouaoui, A., Berkani and M. Belkasmi, “An Efficient Soft Decoder of Block Codes Based on Compact Genetic Algorithm”, International Journal of Computer Science Issues, Volume 9, Issue 5 September, 2012.

[4] S. Baluja, “Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning”, Carnegie Mellon Univ., Pittsburgh, PA. Tech. Rep. CMU-CS-94-163, 1994.

[5] Berkani, A. Azouaoui, and M. Belkasmi, “Soft-decision decoding by a compact genetic algorithm using higher selection pressure,” 2015 International Conference on Wireless Networks and Mobile Communications (WINCOM), Oct. 2015.

[6] Berkani, A. Azouaoui, M. Belkasmi, and B. Aylaj, “Compact Genetic Algorithms with larger tournament size for soft-decision decoding,” 2015 15th International Conference on Intelligent Systems Design and Applications (ISDA), Dec. 2015.

[7] E. Berlekamp, R. McEliece, and H. van Tilborg, “On the inherent intractability of certain coding problems (Corresp.),” IEEE Transactions on Information Theory, vol. 24, no. 3, pp. 384–386, May 1978.

[8] F. A. C. M. Cardoso and D. S. Arantes, “Genetic decoding of linear block codes,” Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406).

[9] D. Chase, “Class of algorithms for decoding block codes with channel measurement information,” IEEE Transactions on Information Theory, vol. 18, no. 1, pp. 170–182, Jan. 1972.

[10] G. C. Clark and J. B. Cain, “Error-Correction Coding for Digital Communications,” 1981.

[11] J. S. De Bonet, C. Isbell and P. Viola, “MIMIC: Finding optima by estimating probability densities”, Advances in Neural Information Processing Systems, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds. Cambridge, MA: MIT Press, vol. 9, 1997, 424.

[12] M. P. C. Fossorier and Shu Lin, “Soft-decision decoding of linear block codes based on ordered statistics,” IEEE Transactions on Information Theory, vol. 41, no. 5, pp. 1379–1396, 1995.

[13] G. R. Harik, F. G. Lobo, and D. E. Goldberg, “The compact genetic algorithm,” IEEE Transactions on Evolutionary Computation, vol. 3, no. 4, pp. 287–297, 1999.

[14] J. Holland, “Adaptation in Natural and Artificial Systems”, The University of Michigan Press, Ann Arbor, Mich, USA, 1975.

[15] P. Larrañaga and J. A. Lozano, Eds., “Estimation of Distribution Algorithms,” Genetic Algorithms and Evolutionary Computation, 2002.

[16] H. Maini, K. Mehrotra, C. Mohan, and S. Ranka, “Soft decision decoding of linear block codes using genetic algorithms,” Proceedings of 1994 IEEE International Symposium on Information Theory.

[17] H. Mühlenbein and G. Paaß, “From recombination of genes to the estimation of distributions I. Binary parameters,” Lecture Notes in Computer Science, pp. 178–187, 1996.

[18] Shakeel, “GA-based soft-decision decoding of block codes,” 2010 17th International Conference on Telecommunications, 2010.

[19] G. Syswerda, “Simulated Crossover in Genetic Algorithms,” Foundations of Genetic Algorithms, pp. 239–255, 1993.

**Ahlam Berkani** received her engineer diploma in Telecommunications and networks from ENSAO (Ecole Nationale des Sciences Appliquees d’Oujda), Morocco in 2010. Actually, she is preparing her PhD since 2011 in Computer Science and Engineering at Department of Computer Science ENSIAS (Ecole Nationale Supérieure d’Informatique et d’Analyse des Systmes), Rabat, Morocco. She got best paper awards in International workshop on codes, cryptography and communication systems (WCCCS) November 2013, Meknes, Morocco. She had 10 publications in national and international conferences and journals. Her areas of interest are Information and Coding theory, and Artificial Intelligence.

**Ahmed Azouaoui** first received his license in Computer Science and Engineering in June-2001 and Master in Computer Science and telecommunication from University of Mohammed V - Agdal, Rabat, Morocco in 2003. He received his PHD in Computer Science in 2014 and Engineering at Department of Computer Science ENSIAS (Ecole Nationale Supérieure d’Informatique et d’Analyse des Systèmes), Rabat, Morocco. Currently, he is An Assistant professor at Faculty of sciences El Jadida, University Chouaib Dokkaly His areas of interest are Information, Coding Theory and Artificial Intelligence.

**Pr. Mostafa Belkasmi** is a professor at ENSIAS (Ecole Nationale Supérieure d’Informatique et d’Analyse des Systèmes, Rabat); head of Telecom and Embedded Systems Team at SIME Lab. He had PhD at Toulouse University in 1991(France). His current research interests include mobile and wireless communications, interconnections for 3G and 4G, and Information and Coding Theory.

**Bouchaib Aylaj** received his PDH in computer science in 2015 in faculty of science El Jadida, Chouaib Doukkaly University. His areas of interest are Information, Coding Theory and Artificial Intelligence.