A Soft Decoding of Linear Block Codes by Genetic Algorithms

Ahmed Azouaoui and Dr. Mostafa Belkasmi

Abstract — In this paper, we present a hard-decision decoding technique based on Genetic Algorithms (HDGA), which is applicable to the general case where the only known structure is given by the generating matrix G. We also present a new soft-decision decoding based on HDGA and the Chase algorithm (SDGA). The simulations, applied on some binary Linear Block Codes, show that the genetic decoder has the same performance as the Berlekamp-Massey Algorithm (BMA) and SDGA is equivalent to soft-decision decoding using Chase algorithm and BMA (Chase-BMA).

Keywords — Genetic Algorithms (GA); Error correcting Codes; Chase Algorithm; Information Set; BM; Chase-BMA; HDGA; SDGA.

I. INTRODUCTION

The current large development and deployment of wireless and digital communication has a great effect on the research activities in the domain of error correcting codes. Codes are used to improve the reliability of data transmitted over communication channels susceptible to noise. Coding techniques create code words by adding redundant information to the user information vectors. Decoding algorithms try to find the most likely transmitted code word related to the received (see Fig 1)

Decoding algorithms are classified into two categories: Hard decision and soft decision algorithms. Hard decision algorithms work on a binary form of the received information. In contrast, soft decision algorithms work directly on the received symbols [12].

Soft-decision decoding is an NP-hard problem and was approached in different ways. Recently artificial intelligence techniques were introduced to solve this problem. These techniques show very good results. Among related works, one work A* algorithm to decode linear block codes [2], another one uses genetic algorithms for decoding linear block codes [3] and a third one uses Compact Genetic Algorithms to decode BCH codes [9]. Maini et al were the first, to our knowledge, to introduce Genetic algorithms [10] in the decoding of linear block codes. Other works were lead for the use of genetic algorithms in turbo codes issues. Luke Hebbes et al worked on the integration of genetic algorithms in a classical turbo codes decoder [13], and Nicolas Durand et al worked on the optimization of turbo decoding by optimizing the interleave with a genetic algorithm [14].

We have investigated the use of genetic algorithms in different ways. In [7], it was shown that a decoder based on the genetic algorithms applied to RS codes outperforms the Chase Decoding algorithms. In [5], a new iterative decoding of product block codes based on the genetic algorithms with very good performances was presented. The previous works deal with block codes decoding.

In this paper, we tackle with the decoding of linear block codes with genetic algorithms. A comparison in terms of performances of the Genetic decoder versus the BM decoder. The paper is structured as follows: The second section introduces the decoder based on genetic algorithms for linear block codes. The third section presents the Chase algorithms decoding. And before concluding this paper we present the experimentation results.
II. GENETIC ALGORITHM BASED DECODER

A. Genetic algorithm

The technique of GA [1] was used by Cardoso et al. [3] for decode linear codes. The purpose of the presented GA is to convert the decoding problem into search problem for a feasible solution around the received vector \( r = (r_0, r_1, ..., r_{n-1}) \). In the presented case, the population will always consists of individuals which are intrinsically associated with codewords. Besides, the evolutionary operations over the individuals will always result in new individuals which are associated with codewords. The evolution, therefore, will take place within the constrained search space. After a finite number of generations, the best individual (codeword) in the population will become a highly evolved solution to the problem.

A simple GA consists of three basic operations: crossover, mutation and natural selection. The last two crossover types exploit particular characteristics of the studied decoding problem, in such a way that the search is carried out over a region around the received vector where the closet codeword is more likely to be found. Furthermore, the mutation type exploits particular characteristics of the decoding problem, so that the convergence is accelerated [3].

A basic GA cycle is shown in Figure 2. Initially, a set of \( N \) solutions are randomly assigned to constitute the initial population. These solutions are encoded to be consistent with the chosen chromosome representation. Each GA cycle corresponds to a new generation.

The crossover operator provides \( 2N \) offsprings. These offsprings and parents are put together to form an intermediate population. This results in a population with \( 3N \) individuals that will be assessed so that a fitness value is assigned to each of them. The fitness is evaluated by calculating the hamming distance between a codeword represented by an individual and the received vector \( r \). After crossover and evaluation, the \( M \) fittest individuals are copied and saved, and the Mutation operator is then applied to this population of \( 3N \) individuals.

The resulting population is again evaluated, and the \( M \) fittest individuals are saved. From these \( 2M \) individuals only the \( M \) best ones are saved.

In order to reduce the population size back to \( N \) individuals, the Natural Selection operator is then applied to the population. The Natural Selection is an operator that tends to discard the worst individuals. However, such operation may also eliminate the fittest individuals. To avoid this, the well known elitist strategy, that replaces to the population the best \( M \) individuals, is used.

B. Chromosome Representation

One very important feature of a GA is the chromosome representation, that is, the way in which feasible solutions are encoded into chromosomes as a string of binary digits. In the decoding problem, for example, the \( k \) information bits of a systematic code could be used as bitstream chromosome of length \( k \). The \( (n-k) \) remaining digits would then be calculated that is associated with a codeword of length \( n \) is constituted of a bitstream chromosome of length \( k \). Although this approach seems quite reasonable at first, it in fact results in disappointing performance. As matter of fact, a GA implemented with this chromosome representation failed to converge for quite simple examples of received vectors, even for relatively short codes. This was a surprising result to us.
Therefore, due to the importance of this representation, other encoding schemes for "chromosoming" a codeword was investigated. The representation of one individual by two chromosomes was chosen, where one of them is necessarily an Information set (IS) of the code. The other chromosome, call it $X$, is a bitstream with $n$ bits that may not belong to the code $C$. The objective here is to guide the search by using as much of the problem structure as possible into the genetic algorithm. This significantly improves the algorithm performance.

In an $(n,k)$ linear block code, an information set is defined as set of $k$ symbols of a codeword that can be independently specified. The remaining $n-k$ elements are redundant parity-check symbols. Thus, if a received word has been corrupted by noise in some bits of the transmitted word, and if it is possible to find an error-free IS, so that all the wrong bits are parity-check bits, then the received vector can be successfully decoded.

In this work, an IS is represented by an $n$-dimensional vector in which $k$ bits are equal to 1 and the others are set to 0. The non-zero bits of the IS correspond to linearly independent columns of the matrix $G$. such an information set is represented by the symbol $p$.

Here, there is an important difference between individual and chromosome. The idea is to represent a codeword by an individual with two chromosomes: an IS chromosome $p_i$ and another binary n-tuple chromosome $x_i$. A codeword $c_i$ can then be obtained by using those bits of $x_i$ that correspond to the check equations of matrix $G$. Thus, even if $x_i$ is not a codeword, the individual represented by the pair $(x_i, p_i)$ is guaranteed to be a codeword of $C$. this encoding scheme allows for the definition of crossover and mutation over information sets, in such a way as to guarantee feasible solutions along the whole evolution process. That is all individuals produced by the operators correspond to valid codewords. For this reason, the individual $(x_i, p_i)$ with two chromosomes is used in this work to represent a codeword $c_i \in C$.

C. Initial Population

The initial population is obtained by random generation of $N$ pairs of vectors $x_i$ and $p_i$ to form $N$ individuals $(x_i, p_i)$. For example, if $N = 4$ the initial population is:

- Type1: $\{(x_0, p_0), (x_1, p_1), (x_2, p_2), (x_3, p_3)\}$
- Type2: $\{(r, p_0), (r, p_1), (r, p_2), (r, p_3)\}$ where the three first individuals are female and the last one is male
- Type 3: $\{(r, p_0), (r, p_1), (r, p_2), (r, p_3)\}$

D. Fitness Evaluation and Selection

Consider an individual (codeword) $c_i = (x_i, p_i)$. In this decoding problem, the objective function is the Hamming distance between a codeword $c_i$ and the received vector $r$, given by $D_H(c_i, r)$. The fitness function can be simply defined as

$$a(c_i) = \frac{1}{1 + D_H(c_i, r)}$$

As usual, the probability of selecting the $i$th individual $c_i$ of the population, will be defined as

$$p(c_i) = \frac{\sum_{j=0}^{N-1} a(c_j)}{\sum_{j=0}^{N-1} a(c_j)}.$$ (2)

The basis for natural selection lies in the above probability. If the selection probability of each individual of the population is known, then roulette can be set up to select an individual. This is done as follows. First, a random number $\zeta$ is generated in the interval $0 < \zeta < 1$.

If $\sum_{i=0}^{m} p(c_i) < \zeta \leq \sum_{i=0}^{m+1} p(c_i)$ (3)

Then the $m$th individual is selected. If a new selection is desired, then generate $\zeta$ at random so that $0 < \zeta < 1 - p(c_m)$, set $p'(c_m) \leftarrow p(c_m)$ and set $p(c_m) \leftarrow 0$. This must be done for each accomplished selection. Generally, the procedure consists in generating $\zeta$ so that

$$0 < \zeta < 1 - \sum_{j \in J} p'(c_j)$$

Where $J$ is a set of indexes representing the selected individual. Then, the individual $c_m$ must be selected so that Equation 3 is satisfied. Notice that an individual is selected only once.
E. Crossover

Let \( c_i = (x, p_i) \) and \( c_k = (y, p_k) \), where \( i,k \in \{0,1,...,N-1\} \), be individuals associated with \( n \)-dimensional binary vectors \( x, y, p_i \) and \( p_k \). A way to generate new individuals through crossover is by combining subparts of \( x \) and \( y \) to produce two offsprings that contain some parts of both parent’s genetic materials. Three of the four implemented crossover types use this model of recombination.

Another possible form of recombination uses individuals of the type \( c_i = (r, p_i) \) and \( c_k = (r, p_k) \), \( i,k \in \{0,1,...,N-1\} \), and implements the recombination using the IS chromosome instead of chromosome \( X \).

- Crossover type 1

This crossover combines subparts of \( x \) and \( y \). In this case, the GA is implemented to use the first population type. This recombination is carried out in the simplest way. Given two parents and a randomly assigned slice point, the portions of the two chromosomes beyond this point are exchanged to from other chromosomes. However, the two offsprings are only created when an IS chromosome is randomly generated for each chromosome \( X \) that was obtained through crossover.

- Crossover type 2

Crossover 2 is equivalent to crossover 1, except that crossover 2 uses uniform recombination instead of one-point crossover. In this case, the GA is also implemented to use the population type 1. The uniform crossover is a generalization of the one-point crossover in the sense that every bit is a potential crossover point. Bits are randomly generated to from a binary string indicates which parent will supply the offspring with the associated bit. If the random string contains a 1 at specific location, then the corresponding parent’s bits are exchanged in the process that forms two offsprings. If this random bit is “0”, then no exchange takes place.

- Crossover type 3

This crossover type also employs parent’s chromosome \( X \) to carry out the uniform recombination. However, the population model used here is 2, the chromosome \( y \) is always equal to the received vector \( r \), while the chromosome \( x \) may be any binary \( n \)-dimensional vector. Notice that an offsprings is generated only when its IS chromosome is randomly selected.

- Crossover type 4

The idea consists in intelligently combining the parents’ IS chromosome to obtain another IS vector, in such a way that it differs as much as possible form its parents’ IS. Because of this, this recombination model will be named aberrant crossover. The algorithm tries to find \( k \) information bits without errors in the corresponding chromosome \( X \). The idea here is if an optimal solution has not yet been found, it is because the used IS chromosome \( X \) point to information bits in chromosome \( X \) which are corrupted by noise. The algorithm is described as follows:

Choose from the intermediate population two individuals at random, for example, \( c_i = (r, p_i) \) and \( c_k = (r, p_k) \).

Pair the IS chromosome \( p_i \) with \( p_k \) to generate a crossover \( q_i \) as described below.

Set all \( n \) bits of \( q_i \) to zero.

For each \( l = 1,2,...,n \), do:

If \( p_i[l] = p_k[l] = 0 \) do:

\[ q_i[l] \leftarrow 1; \]

For each location of \( q_i \) that is equal to 1, get the corresponding columns of the generator matrix \( G \) to form a vectorial set. If this set is linearly dependent (LD), do:

\[ q_i[l] \leftarrow 0; \]

End of for.

Since the code is \( (n,k) \), if only \( m \) positions with \( m < k \) were set to 1 in the IS chromosome \( q_i \), then the remaining \( n - m \) positions equal to 0 must be used to find the remaining \( k - m \) information bits. Thus, set at random \( k - m \) 1s in the \( n - m \) remaining zero bits, in such a way that \( q_i \) represents a valid information set of the code.

Finally, one of the parents’ chromosomes \( X \) is inherited by the offspring. For example, assuming that \( y \) has been selected as chromosome, then the generated offspring is the \( f_i = (r', q_i) \) whose parents are \( c_i \) and \( c_k \).

F. Mutation

Mutation is an operator that randomly introduces variations into a given chromosome, so that all possible solutions in the search space can be reached.

- Mutation type 1

Given an individual, this operator alters every gene (bit) of chromosome \( X \) by inversion with probability \( P_m \).
• Mutation type 2

This operator alters an individual by bit inversion of chromosome \(X\). However, such an inversion takes place only in one bit and only when an improvement in the individual’s fitness is achieved. If it is not possible to improve the individual’s fitness, then no alteration is performed. The algorithm simply goes through every \(X\) chromosome’s gene to determine which of them must be changed in such a way as to improve individuals’ fitness. Again, the idea is to guide the search so that the convergence of the GA is accelerated.

III. SOFT DECODING BY GA AND CHASE

A. Chase algorithm

The Chase algorithm (Chase 1972) [6] is a suboptimal decoding procedure that uses a set or list of most likely error patterns. These error patterns are selected on the basis of the reliability of the received symbols. Each error pattern is added to the hard-decision received word and decoded using a hard-decision decoder. Each decoded code word is scored by computing is metric with respect to the received SD sequence. The code word with the best metric is selected as the most likely.

Our soft decoding based GA (SDGA) developed in the following section is based on the Chase algorithm.

B. Soft decoding by GA (SDGA)

Let \(C\) be a binary linear \((n, k, d)\) block code where \(d\) is a minimum distance, capable of correcting any combination of \(t = \left\lfloor \frac{d-1}{2} \right\rfloor \) or less random bit errors. Let \(\overline{r} = (r_1, r_2, \ldots, r_n)\) be the received word from the output of the channel, \(r_i = (-1)^{\nu_i} + w_i\), where \(w_i\) is a zero-mean Gaussian random variable with variance \(\frac{N_0}{2}\), \(i = 1, 2, \ldots, n\).

The sign bits of the received values represent the hard-decision

\[
\overline{z}_0 = (z_{0,1}, z_{0,2}, \ldots, z_{0,n}) \quad z_{0,j} = \text{sign}(r_j), 0 \leq j < n,
\]

Where

\[
\text{sign}(x) = \begin{cases} 0, & \text{if } x \geq 0; \\ 1, & \text{otherwise.} \end{cases}
\]

The reliabilities of the received channel values, for binary transmission over an \(AWGN\) channel, are the amplitudes\(|r_i|\). The received symbol reliabilities are ordered with a sorting algorithm (e.g., quick sort). The output of the algorithm is a list of indexes \(I_j, j = 1, 2, \ldots, n\), such that

\[
|r_{I_j}| \leq |r_{I_{j-1}}| \leq \cdots \leq |r_{I_n}|
\]

In the first round of decoding, the hard-decision received word \(\overline{z}_0\) is fed into a HDGA. Let \(\overline{v}_0\) denote the decoded code word, stored as the initial code word guess. Then the metric of \(\overline{v}_0\) with respect to the received word

\[
v_0 = \sum_{j=1}^{n} (-1)^{\nu_j}, \quad (3)
\]

is computed and its value is stored as the maximum.

Test the error patterns with at most \(\left\lfloor \frac{d}{2} \right\rfloor\) errors located within the bit positions having the \(\left\lfloor \frac{d}{2} \right\rfloor\) lowest reliabilities.

- For \(i = 1, 2, \ldots, 2^{t-1}\), an error pattern \(\overline{e}_i\) is added to the hard-decision received word: \(\overline{z}_i = \overline{e}_i \oplus \overline{z}_0\).
- The error patterns are generated among the \(t\) least reliable positions (LRPs), that is, Positions \(\{I_1, I_2, \ldots, I_t\}\) for which reliabilities (amplitudes) are the smallest.
  - Each test vector \(\overline{z}_i\) is input to a HDGA, producing a code word \(\overline{v}_i\), \(i = 1, 2, \ldots, 2^{t-1}\).
  - The metric is computed according to Equation (3) and if maximum, code word \(\overline{v}_i\) stored as the most likely.
IV. SIMULATION RESULTS

The simulations were made with the default parameters outlined in the Table 1 except where indicated.

For transmission we used an AWGN channel with a BPSK modulation.

A. HDGA versus BMA performances

The Fig. 4 and Fig. 5 below compares the performance of HDGA and BMA for the codes BCH (31,16,7) and BCH (31,21,5). We notice in general the equality in terms of performance between the two decoders.

B. SDGA versus Chase-BMA performances

The Fig. 6 and Fig. 7 compares the performance of SDGA and the Chase-BMA for the codes BCH(15,7,5) and RM(32,16,8). We notice also the equality in terms of performance between the two decoders.
V. CONCLUSION

In this paper, we have presented a genetic algorithm based decoder for binary linear block codes. This algorithm is in fact evolutionary versions of the well known information set decoding.

The simulations, applied on some binary Linear Block Codes, show that the genetic decoder has the same performance as BMA and SDGA is equivalent to soft-decision decoding using Chase-BMA. The algorithms can be generalized to other kinds of linear codes.

REFERENCES