Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2012, Article ID 503834, 12 pages
doi:10.1155/2012/503834

Research Article

Efficient Dual Domain Decoding of Linear Block Codes Using Genetic Algorithms

Ahmed Azouaoui, Mostafa Belkasmi, and Abderrazak Farchane

SIME Lab, ENSIAS, Mohammed V-Souissi University, Rabat, Morocco

Correspondence should be addressed to Ahmed Azouaoui, aazouaoui@gmail.com

Received 14 September 2011; Revised 21 December 2011; Accepted 12 January 2012

Academic Editor: Lisimachos P. Kondi

Copyright © 2012 Ahmed Azouaoui et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A computationally efficient algorithm for decoding block codes is developed using a genetic algorithm (GA). The proposed algorithm uses the dual code in contrast to the existing genetic decoders in the literature that use the code itself. Hence, this new approach reduces the complexity of decoding the codes of high rates. We simulated our algorithm in various transmission channels. The performance of this algorithm is investigated and compared with competitor decoding algorithms including Maini and Shakeel ones. The results show that the proposed algorithm gives large gains over the Chase-2 decoding algorithm and reach the performance of the OSD-3 for some quadratic residue (QR) codes. Further, we define a new crossover operator that exploits the domain specific information and compare it with uniform and two point crossover. The complexity of this algorithm is also discussed and compared to other algorithms.

1. Introduction

The current large development and deployment of wireless and digital communication encourage the research activities in the domain of error correcting codes. The later is 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 vectors. Decoding algorithms try to find the most likely transmitted codeword related to the received one as depicted in Figure 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 [1].

Soft-decision decoding is an *NP*-hard problem and was approached in different ways. Recently, artificial intelligence techniques were introduced to solve this problem. Among the related works are the decoding of linear block codes using algorithm A* [2], another one uses genetic algorithms for decoding linear block codes [3], and the third one uses neural networks to decode BCH codes [4].

Maini et al. [3] were the first, according to our knowledge, to introduce genetic algorithm in the soft decoding of linear block codes. After that, Cardoso and Arantes [5] came to work on the hard decoding of linear block codes using GA and Shakeel [6] worked on soft-decision decoding for block codes using a compact genetic algorithm. These decoders based on GA use the generator matrix of the code; this fact makes the decoding very complicated for codes of high rates.

Genetic algorithms are search algorithms that were inspired by the mechanism of natural selection where stronger individuals are likely the winners in a competing environment. They combine survival of the fittest among string structures with a structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search. In each generation, a new set of artificial creatures (chromosomes) is created using bits and pieces of the fittest of the old [7, 8].

The Dual Domain Decoding GA algorithm (DDGA) is a significant contribution to soft-decision decoding. In effect, a comparison with other decoders, that are currently the most successful algorithms for soft decision decoding, shows its efficiency. This new decoder can be applied to any binary linear block code, particularly for codes without algebraic

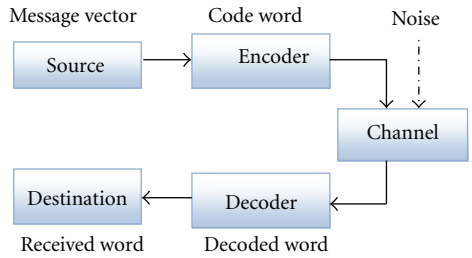


Figure 1: A simplified communication system model.

decoder. Unlike chase algorithm which needs an algebraic hard-decision decoder. Further, it uses the dual code and works with parity-check matrix. The later makes them less complicated for codes of high rates. In order to show the effectiveness of this decoder, we applied it for BCH and QR codes over two transmission channels.

The remainder of this paper is organized as follows: in Section 2, we introduce the genetic algorithms. Section 3 expresses soft-decision decoding as a combinatorial optimisation problem. In Section 4, DDGA, our genetic algorithm for decoding, is described. Section 5 reports the simulation results and discussions. Finally, Section 6 presents the conclusion and future trends.

2. Genetic Algorithm

Genetic algorithm is an artificial intelligence based methodology for solving problems. It is a non-mathematical, nondeterministic, but stochastic process or algorithm for solving optimization problems. The concept of genetic algorithm was introduced by John Holland [7] in 1975 with the aim of making computers do what nature does. He was concerned with algorithms that manipulate strings of binary digits to find solution to problem in such a way that it exhibits the characteristics of natural evolution, that is, developing an algorithm that is an abstract of natural evolution. The idea of genetic algorithm by Holland stemmed from the evolutionary theory.

GA is excellent for all tasks requiring optimization and highly effective in any situation where many inputs (variables) interact to produce a large number of possible outputs (solutions). Some example situations are as the following.

Optimization. Such as data fitting, clustering, trend spotting, path finding, and ordering.

Management. Distribution, scheduling, project management, courier routing, container packing, task assignment, and timetables.

Financial. Portfolio balancing, budgeting, forecasting, investment analysis, and payment scheduling.

Engineering. Structural design (e.g., beam sizes), electrical design (e.g., circuit boards), mechanical design (e.g., optimize weight, size cost), process control, and network design (e.g., computer networks).

The typical steps in the design of genetic algorithm are described below and illustrated in the Figure 2:

Step 1. Representation of the problem variable domain as a chromosome of a fixed length, the size of a chromosome population is chosen as well as the crossover probability.

Step 2. Definition of fitness function, which is used for measuring the quality of an individual chromosome in the problem domain. The fitness function establishes the basis for selecting chromosomes that will be mated during reproduction.

Step 3. Random generation of an initial population of chromosomes of a fixed size.

Step 4. Calculation of fitness function for each individual chromosome.

Step 5. Selection of pairs of chromosomes for mating from the current population. Parent chromosomes are selected with a probability related to their fitness. The highly fit chromosomes have a higher probability of being selected for mating.

Step 6. Application of the genetic operations (crossover and mutation) for creating pairs of offspring chromosomes.

Step 7. Placement of the created offspring chromosomes in the new population.

Step 8. Repetition of Step 5 through Step 7 until the size of the new chromosome population becomes equal to the size of the initial population.

Step 9. Replacement of the initial (previous) parent chromosome population with the new offspring population.

Step 10. Repeat Step 4 through Step 9 until the termination criterion is satisfied.

The termination criterion could be any of the following:

(1) attaining a known optimal or acceptable solution level;

(2) a maximum number of generations has been reached.

3. Soft Decision Decoding as an Optimisation Problem

The maximum-likelihood soft-decision decoding problem of linear block codes is an *NP*-hard problem and can be stated as follows.

Given the received vector r and the parity-check matrix H, and let $S = zH^T$ be the syndrome of z, where z is the hard decision of r, H^T is the transpose of matrix H, and $E(S)$ be

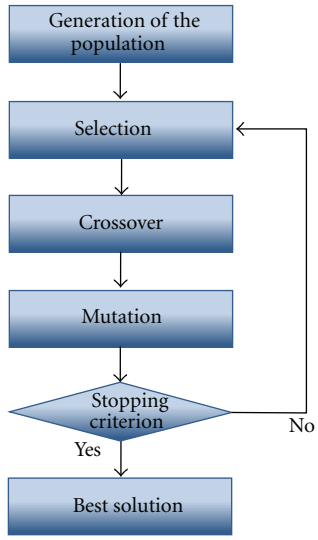


Figure 2: A simplified model of a genetic algorithm.

the set of all errors patterns whose syndrome is S, find the $E \in E(S)$ which minimises correlation discrepancy:

$$f_r(E) = \sum_{j=1}^{n} E_j \left| r_j \right|, \\ = \sum_{j=1, E_j=1}^{n} \left| r_j \right|, \tag{1}$$

where n is the code length. The optimisation problem (1) has n *error* variables, out of which only $(n\text{-}k)$ 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 $(n\text{-}k)$ variables.

Up till now, only few authors have tried to solve the soft decision decoding problem by viewing it as an optimisation problem and using artificial intelligence techniques.

4. DDGA Algorithm

Let C denote, a (n, k, d) binary linear block code of parity check matrix H, and let $(r_i)_{1 \le i \le n}$ be the received sequence over a communication channel with noise variance $\sigma^2 = N_0/2$, where N_0 is noise power spectral density.

Let N_i, N_e, and N_g denote, respectively, the population size, the number of elite members, and the number of generations.

Let p_c and p_m be the crossover and the mutation rates.

4.1. Decoding Algorithm. The decoding-based genetic algorithm is depicted on Figure 3. The steps of the decoder are as follows.

Step 1. Sorting the sequence r in such a way that $|r_i| > |r_{i+1}|$ for $1 \le i \le n$. 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 π the permutation related to this permutation $(r' = \pi(r))$. Apply the permutation π to H to get a new check matrix $H' = [AI_{n-K}]$ $(H' = \pi(H))$.

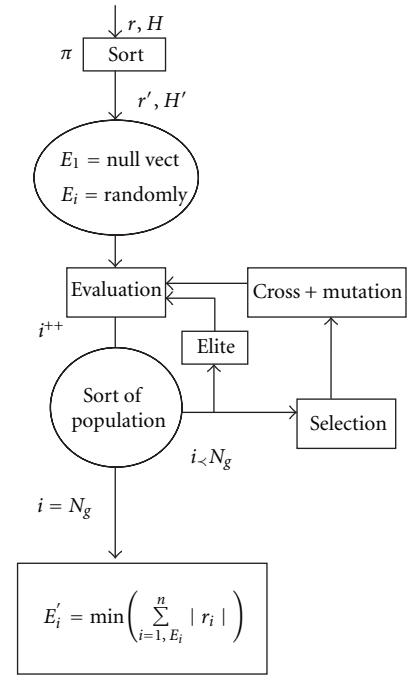


Figure 3: Basic structure of DDGA.

Step 2. Generate an initial population of N_i binary vectors of k bits (an individual represents the systematic part of an error candidate).

Substep 2.1. The first member, E_1, of this population is zero vector.

Substep 2.2. The other $N_i - 1$ members, $(E_j)_{2 \le j \le N_i}$, are uniformly random generated.

Step 3. For i from 1 to N_g.

Substep 3.1. Compute the fitness of each individual in the population. An individual is a set of k bits.

Let E' be an individual, z be the quantization of r', S be the syndrome of z such that $S = zH'^T$, S_1 be an $(n - k)$-tuple such that $S_1 = E'A^T$ where A is submatrix of H', and S_2 be an $(n - k)$-tuple such that $S_2 = S + S_1$.

We form the error pattern E such that $E = (E', E'')$, where E' is the chosen individual and $E'' = S_2$. Then, $(z + E)$ is a codeword.

The fitness function is the correlation discrepancy between the permuted received word and the estimated error such that

$$f_{r'}(E) = \sum_{j=1,\ E_j=1}^{n} \left| r'_j \right|. \tag{2}$$

Substep 3.2. The population is sorted in ascending order of member's fitness defined by (2).

Substep 3.3. The first (elite) N_e best members of this generation are inserted in the next one.

Substep 3.4. The other $N_i - N_e$ members of the next generation are generated as follows.

Subsubstep 3.4.1. Selection operation: a selection operation that uses the linear ranking method is applied in order to identify the best parents $(E'^{(1)}, E'^{(2)})$ on which the reproduction operators are applied. For each individual j $(1 \le j \le N_i)$, we choose the following linear ranking:

$$w_j = w_{\max} - \frac{2(j-1)(w_{\max}-1)}{N_i - 1}, \quad 1 \le j \le N_i, \tag{3}$$

where w_j is the jth member weight and $w_{\max} = 1.1$ is the maximum weight associated to the first member.

Subsubstep 3.4.2. Crossover operator: Create a new vector E'_j "child" of k bits. Let Rand be a uniformly random value between 0 and 1 generated at each occurrence. The crossover operator is defined as follows: if $\text{Rand}_1 \prec p_c$, then the ith bit of child $(E'_j)_{N_e+1 \le j \le Ni}$, $(1 \le i \le k)$ is given by

$$E'_{ji} = \begin{cases} E_i^{'(1)} & \text{if } E_i^{'(1)} = E_i^{'(2)} \\ \text{otherwise} \begin{cases} E_i^{'(1)} & \text{if } \text{Rand}_2 \prec p \\ E_i^{'(2)} & \text{otherwise,} \end{cases} \end{cases} \tag{4}$$

where

$$p = \begin{cases} \dfrac{1}{1+e^{-4r'_j/N_0}} & \text{if } E'^{(1)} = 0,\ E^{'(2)} = 1, \\ \dfrac{e^{-4r'_j/N_0}}{1+e^{-4r'_j/N_0}} & \text{if } E'^{(1)} = 1,\ E^{'(2)} = 0. \end{cases} \tag{5}$$

It is clear that if the ith bit of the parent is different, then for greater positive values r'_j, the function $1/(1+e)^{-4|r'_j|/N_0}$ converges to 1. Hence, the ith bit of child has a great probability to be equal to 0. Note that if $\text{rand}_1 \ge p_c$ (no crossover case):

$$E'_j = \begin{cases} E'^{(1)} & \text{if Rand} \prec 0.5, \\ E^{'(2)} & \text{otherwise.} \end{cases} \tag{6}$$

Subsubstep 3.4.3. Mutation operator: if the crossover operation realized, the bits E'_{ji} are muted with the mutation rate p_m:

$$E'_{ji} \longleftarrow 1 - E'_{ji} \quad \text{if } \text{Rand}_3 \prec p_m. \tag{7}$$

Step 4. The decoder decision is $V^* = \pi^{-1}(E_{\text{best}} + z)$, where E_{best} is the best member from the last generation.

Remark 1. In Step 1 of the DDGA, in order to have a light algorithm we apply the Gaussian eliminations on the $(n-k)$ independent columns corresponding to the least reliable positions, without the permutation π. This optimisation is not used in other similar works [3, 9, 10].

4.2. Complexity Analysis. Firstly, we show that our algorithm has a polynomial time complexity.

Let n be the code length, k be the code dimension which must be equal to the length of individuals in population, and N_i be the population size which must be equal to the total number of individuals in the population.

At any given stage, we maintain a few sets of $N_i \times k$ arrays, therefore the memory complexity of this algorithm is $O(N_i k)$.

In order to get the complexity of our algorithm, we will compute the complexity of each step.

Step 2 has time complexity of $O(k^2 n)$ [2]. This complexity depends on the random number generator in use, but the cost is negligible compared to that of Step 3.

Substeps 3.1 to 3.3 have a computational complexity of

$$O(k(n-k)N_i) + O(N_i \log N_i) + O(k). \tag{8}$$

Steps 3.4.1 to 4 have an advantage case complexity of

$$\begin{aligned} &O(1) + p_c[O(k) + O(k) + O(k)] + (1-p_c)[O(1) + O(k)] \\ &\quad = O(1) + p_c O(k) + (1-p_c)O(k). \end{aligned} \tag{9}$$

This reduces to $O(k)$, which is also the worst case complexity. Hence, any iteration of the genetic algorithm part of DDGA has a total time complexity of $O(k(n-k)N_i + N_i \log N_i)$ per generation.

Algorithm DDGA has a time complexity of $O(N_i N_g [k(n-k) + \log N_i])$.

Now, we compare our algorithm with four competitors. The Table 1 shows the complexity of the five algorithms. The Chase-2 algorithm increases exponentially with t, where t is the error correction capability of the code. Its complexity is 2^t time the complexity of the *hard-in hard-out* decoding [11]. Similarly, the complexity of the OSD algorithm of order m is exponentially in m [9]. Moreover, the decoding becomes more complex for codes with large code length.

For Maini [3] and DDGA algorithms, the complexity is polynomial in k, n, N_g, N_i, and $\log N_i$, making it less complex compared to other algorithms.

For Shakeel algorithm, the complexity is also polynomial in k, n and T_c, where T_c presents the average number of generations.

The three decoders based on genetic algorithms are almost the same complexity and have lower complexity than Chase-2 and OSD-m algorithm.

Table 1: Complexity of Chase-2, OSD-m, Shakeel, Maini, and DDGA algorithms.

Algorithm	Complexity
Chase-2	$O(2^t n^2 \log_2 n)$
OSD-m	$O(n^{m+1})$
Shakeel algorithm	$O(T_c kn)$
Maini algorithm	$O(N_i N_g [kn + \log N_i])$
DDGA	$O(N_i N_g [k(n-k) + \log N_i])$

Table 2: Default parameters.

Simulation parameter	Parameter value
p_c (crossover rate)	0.97
p_m (mutation rate)	0.03
N_g (generation number)	100
N_i (population size)	300
N_e (elite number)	1
Channel	AWGN
Modulation	BPSK
Minimum number of bit errors	200
Minimum number of bloks	1000
Default code	BCH (63,51,5)

5. Simulation Results and Discussions

In order to show the effectiveness of DDGA, we do intensive simulations.

Except the subsection J of the current section, the simulations where made with default parameters outlined in Table 2.

The performances are given in terms of BER (bit error rate) as a function of SNR (Signal to Noise Ratio E_b/N_0).

5.1. Evolution of BER with GA Generations. We illustrate the relation between bit error probability and the number of genetic algorithm generation for same SNRs.

Figure 4 indicates the evolution of bit error probability with genetic algorithm generations. This figure shows that the bit error probability decreases with increasing number of generations.

5.2. Effect of Elitism Operator. The Figure 5 compares the performances of DDGA with or without elitism for the BCH (63,51,5) code. These simulation results reveal a slight superiority of DDGA with elitism.

5.3. Effect of Code Length. The Figure 6 compares the DDGA performance for four codes with rate = 1/2. Except the small code, all other codes have performance almost equal and this is possibly due to the parameters of DDGA, in particular, the number of individuals chosen is fixed for all codes. Indeed, BCH code (31.16) is small and gives poor performance compared to the other three codes (BCH and QR) that are comparable in length and performance.

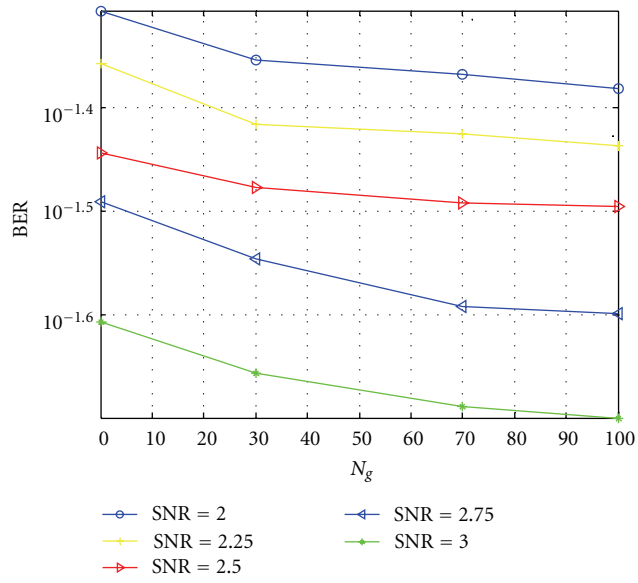


Figure 4: BER versus number of generations.

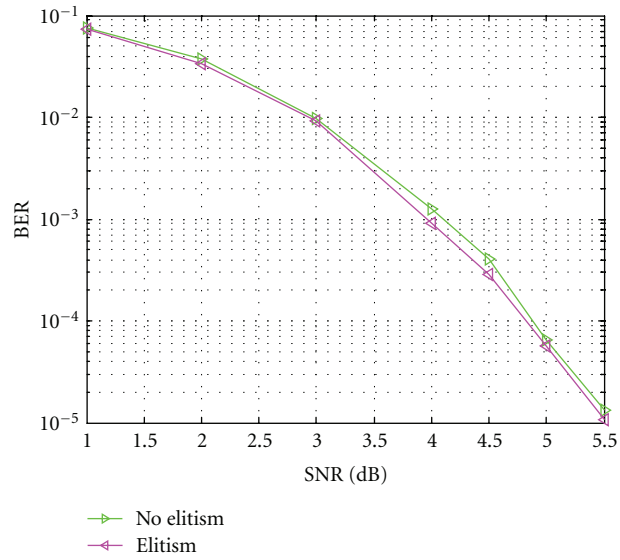


Figure 5: Effect of elitism operator on DDGA performances.

5.4. Effect of Crossover Rate on Performance. The Figure 7 shows the effect of crossover rate on the performance. From this figure, we remark that the increasing of the crossover rate from 0.01 to 0.97 improves the performance of DDGA for BCH (63,51,5) code by 1 dB at 10^{-4}.

5.5. Effect of Mutation Rate on Performances. The Figure 8 emphasizes the influence of the mutation rate on the performance of DDGA. Decreasing the mutation rate from 0.2 to 0.1, we can gain 1 dB at 10^{-4}. If we further decrease the mutation rate, the gain becomes negligible as shown in Figure 8. According to this result, we remark that 0.03 is the

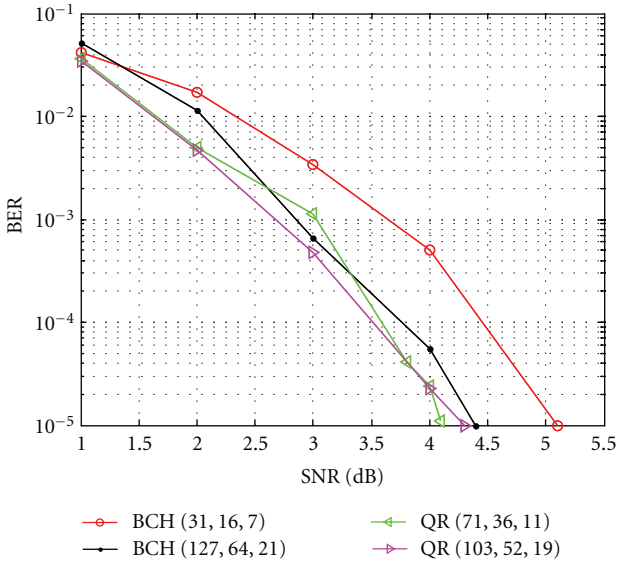


Figure 6: Effect of code length on DDGA performances.

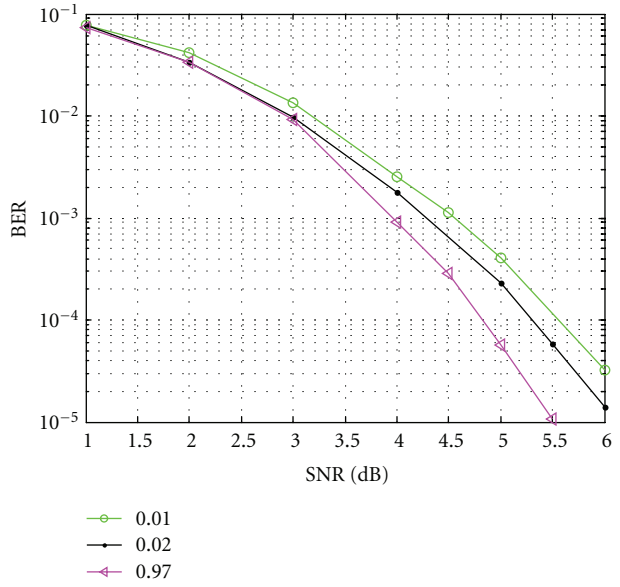


Figure 7: Effect of crossover rate on performances.

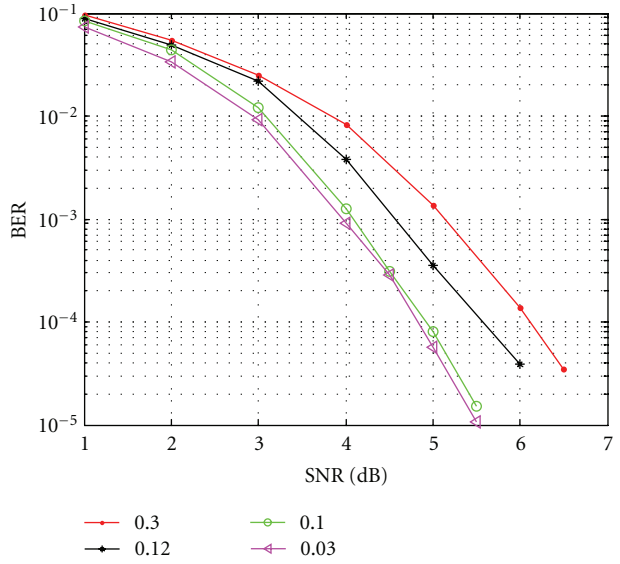


Figure 8: Effect of mutation rate on performances.

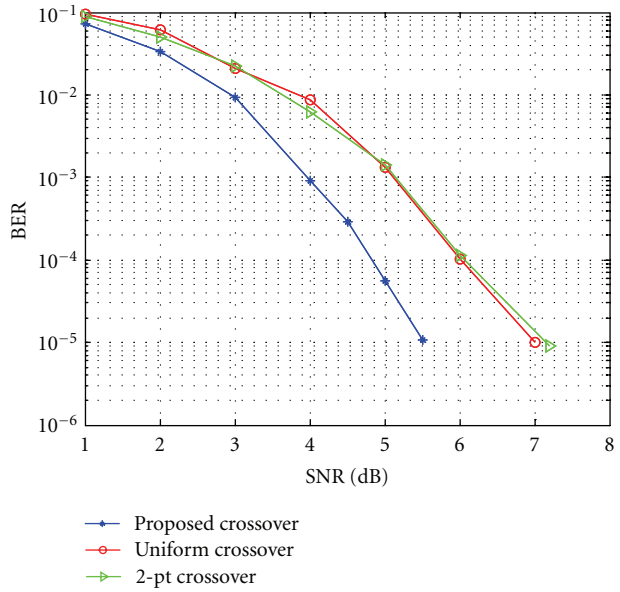


Figure 9: Comparison between different crossover operators in DDGA.

optimum value of the mutation rate. This confirms the result in literature.

5.6. Comparison between Various Crossover Methods in DDGA. In the Figure 9, we compare between results obtained using the proposed crossover (see Section 4), uniform crossover (UX), and two point crossover, (2-pt), in DDGA. Simulation results show that the proposed crossover is better than UX and 2-pt ones. The gain between the proposed crossover and the two other crossovers is 1.5 dB at 10^{-5}. Besides, the three crossover methods have the same complexity of order $O(k)$.

5.7. Comparison between Different Selection Methods in DDGA. In the Figure 10, we present a comparison between the results obtained using linear ranking selection, tournament selection, and random selection in DDGA. Simulation results show that the linear ranking is better than tournament and random selection.

5.8. Comparison of DDGA versus Other Decoding Algorithms. In this subsection, we compare the performance of DDGA with other decoders (Chase-2 decoding, OSD-1, OSD-3, and Maini decoding algorithms).

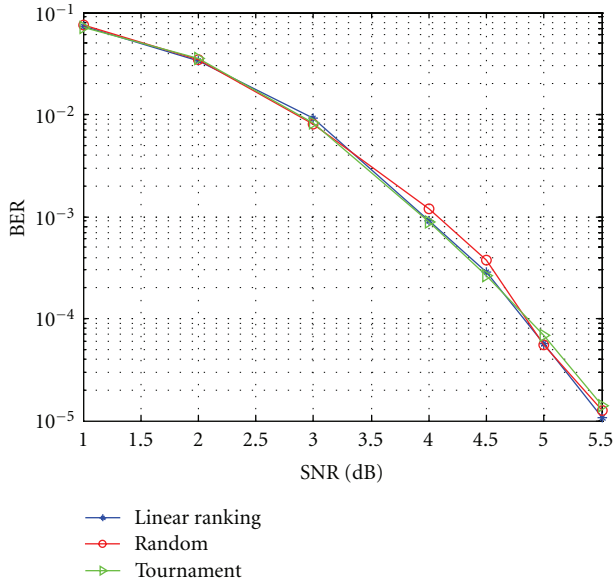


Figure 10: Comparison between different selection operators in DDGA.

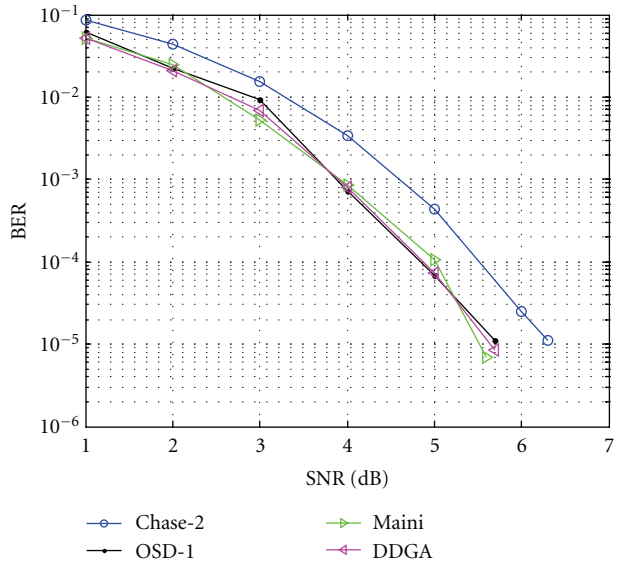


Figure 12: Performances of Chase-2, OSD-1, Maini, and DDGA algorithms for BCH (31,21,5) code.

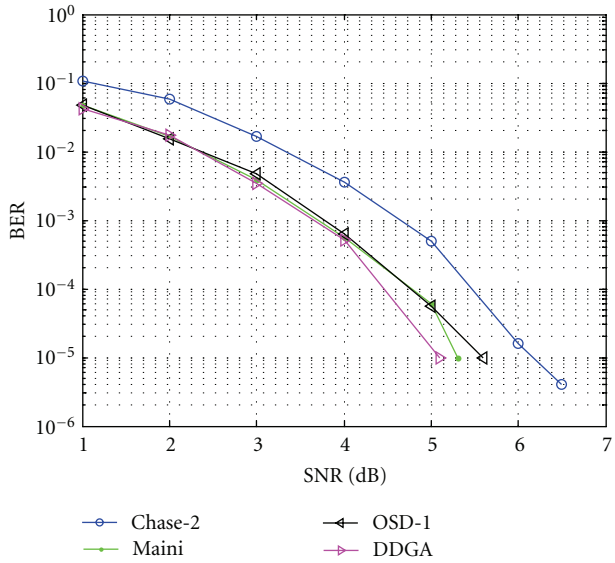


Figure 11: Performances of Chase-2, Maini, OSD-1, and DDGA algorithms for BCH (31,16,7) code.

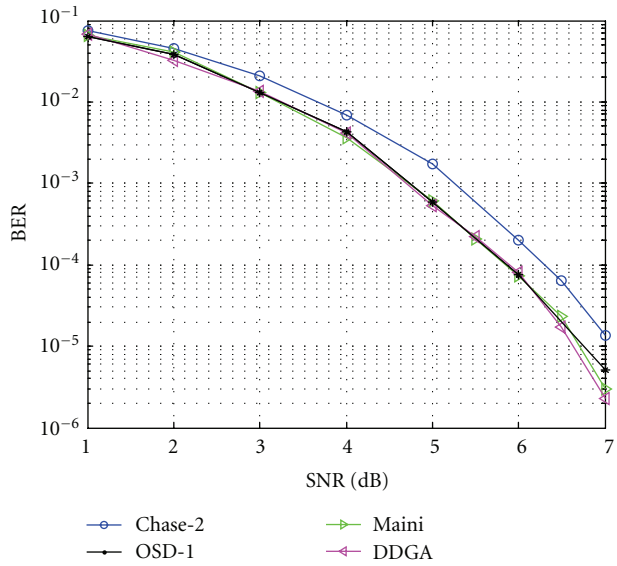


Figure 13: Performances of Chase-2, OSD-1, Maini, and DDGA algorithms for BCH (31,26,3) code.

The performance of DDGA is better than Chase-2 and OSD-1 algorithms as shown in Figure 11. According to this figure, we observed that DDGA is comparable to Maini algorithm.

The performance of DDGA, Chase, OSD-1 and Maini algorithms, for BCH (31,21,5) code, is shown in Figure 12. From the later, we remark that our algorithm is better than Chase-2 algorithm and comparable with Maini and OSD-1 algorithms for this code.

The Figure 13 presents the performance of different decoders for BCH (31,26,3) code. The behaviors of the four decoders for this code are similar as for the BCH (31,21,5) code.

For the BCH (63,45,7) code, DDGA outperforms Chase-2 by 1 dB at 10^{-5}. Nevertheless, the gain of DDGA from the others is negligible as shown in Figure 14.

The Figure 15 compares the performances of DDGA and others decoders for BCH (63,51,5) code. We notice the superiority of DDGA over Chase-2 algorithm and similarity to the others.

The performances of DDGA, Chase-2, OSD1, and Maini algorithms, for BCH (63,57,3) code, are shown in Figure 16.

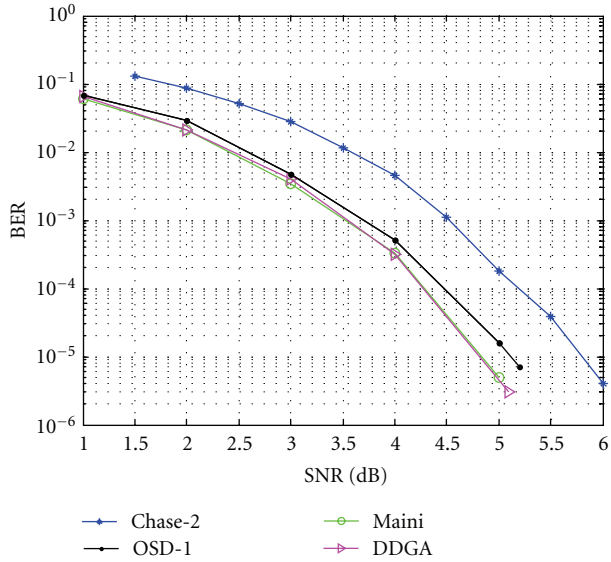


Figure 14: Performances of Chase-2, OSD-1, Maini, and DDGA algorithms for BCH (63,45,7) code.

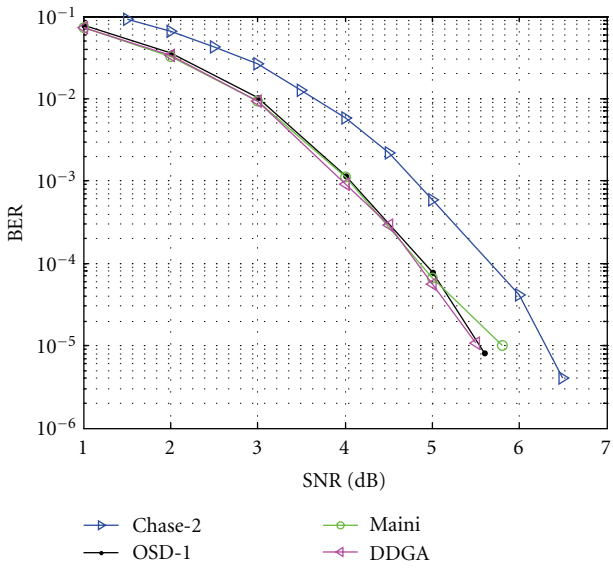


Figure 15: Performances of Chase-2, OSD-1, Maini, and DDGA algorithms for BCH (63,51,5) code.

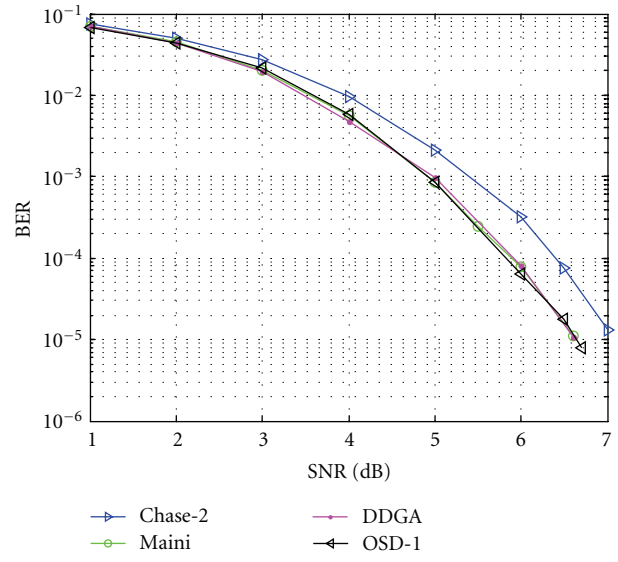


Figure 16: Performances of Chase-2, Maini, DDGA, and OSD-1 algorithms for BCH (63,57,3) code.

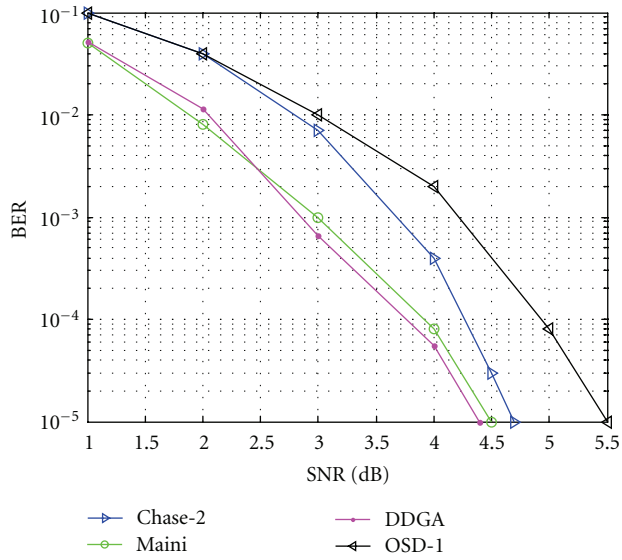


Figure 17: Performances of Chase-2, Maini, DDGA, and OSD-1 algorithms for BCH (127,64,21) code.

From the later, we observe that DDGA is better than Chase-2, and comparable to Maini and OSD-1 algorithms.

DDGA outperforms OSD-1 by 1 dB at 10^{-5}. However, the gain of DDGA from the others is negligible as shown in Figure 17.

The Figure 18 compares the performance of DDGA with other decoders for BCH (127,113,5) code. From this figure, we remark that DDGA is better than Chase-2 algorithm and comparable to the others.

The Figure 19 presents the performances of the decoders mentioned above, for the BCH (127,120,3) code. The behaviors of the four decoders are the same as for the BCH (127,113,3) code.

Figure 20 shows that DDGA performs the same as the OSD-3 for code QR (71,36,11). The complexity of DDGA is less than that of the OSD-3.

The performances of DDGA, Chase OSD-1, and Maini algorithms, for QR (103,52,19) code, is shown in Figure 21. From the later, we notice that our algorithm is better than OSD-1 algorithm and is comparable with Maini algorithm for this code.

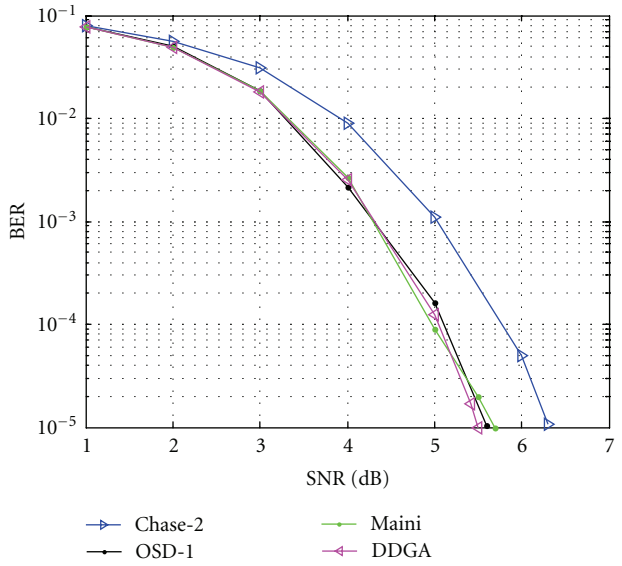


Figure 18: Performances of Chase-2, OSD-1, Maini, and DDGA algorithms for BCH (127,113,5) code.

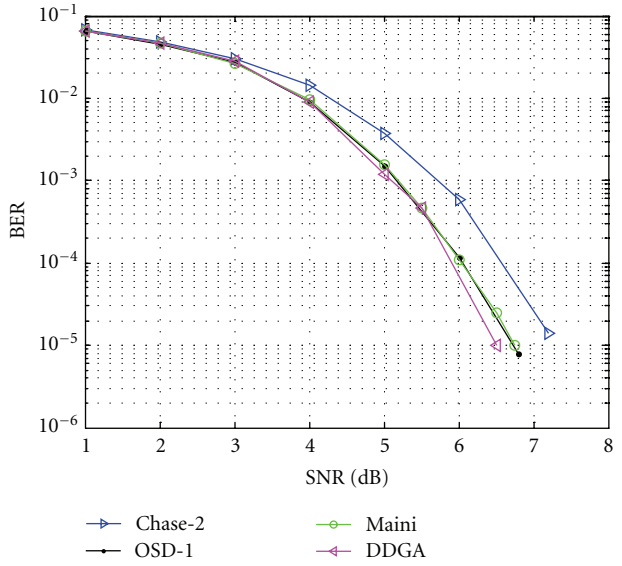


Figure 19: Performances of Chase-2, OSD-1, Maini, and DDGA algorithms for BCH (127,120,3) code.

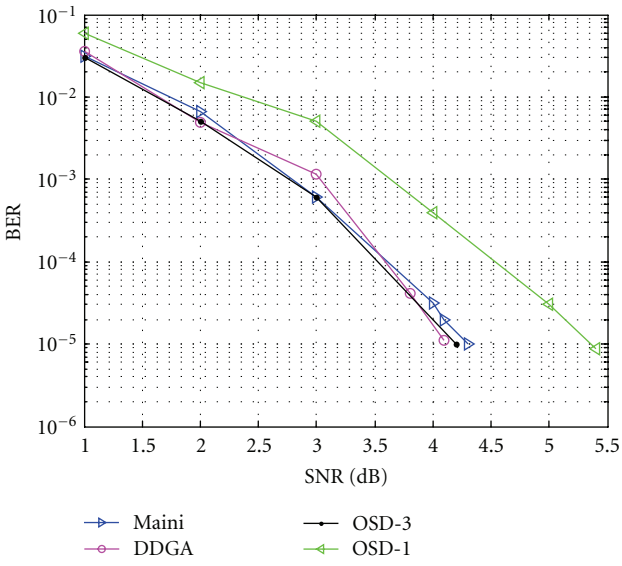


Figure 20: Performances of Maini, DDGA, OSD-3, and OSD-1 algorithms for QR (71,36,11) code.

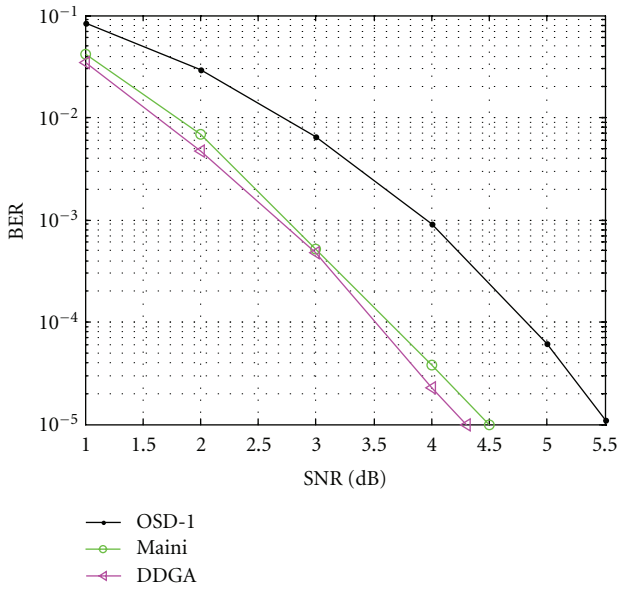


Figure 21: Performances of OSD-1, Maini, and DDGA for QR (103,52,19) code.

5.9. DDGA Performances on Rayleigh Channel. DDGA has been described for the AWGN channel. Soft-decision decoding is particularly desirable over channels with more severe impairments. Therefore, a wire channel corrupted by both AWGN and Rayleigh is used for the performance investigation of DDGA.

The fading channel is modelled as

$$r_i = a_i z_i + n_i, \tag{10}$$

where $i = 1, \ldots, N$; $\{a_j\}_1^N$ are statistically independent Gaussian random variables with zero mean and variance $N_0/2$; $\{a_j\}_1^N$ represent the statistically independent Rayleigh distributed fading coefficients. We assume ideal channel state information (CSI) of $\{a_j\}_1^N$ at the receiver.

The objective function (1) in Step 2 of DDGA is slightly changed for the channel model considered.

The resulting function is:

$$f(z) = \sum_{j=1}^{N} a_j z_j r'_j, \tag{11}$$

where $\{a_j\}_1^N$ is the CSI vector. The remaining steps of the algorithm are kept unchanged.

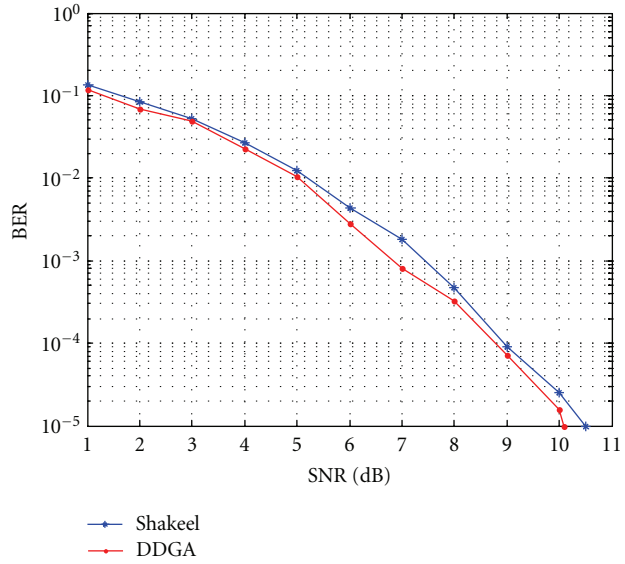


Figure 22: Performances of DDGA, and Shakeel algorithms for BCH (31,16,7) code over RC.

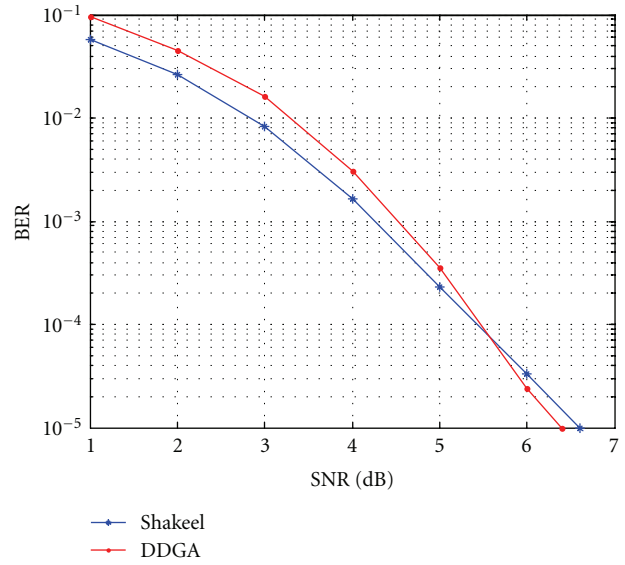


Figure 23: Performances of DDGA, and Shakeel algorithms for BCH(31,21,5) code over RC.

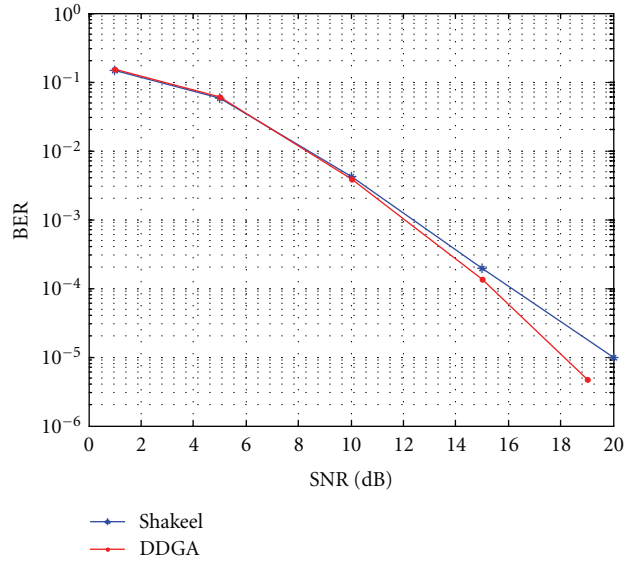


Figure 24: Performances of DDGA, and Shakeel algorithms for BCH (31,26,3) code over RC.

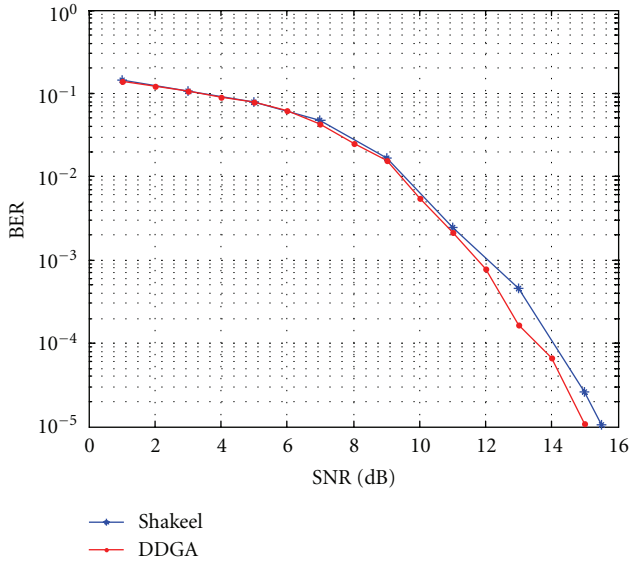


Figure 25: Performances of DDGA, and Shakeel algorithms for BCH (127,113,5) code on RC.

We provide a comparison of DDGA and Shakeel algorithm over Rayleigh channel (RC).

The performance of DDGA and Shakeel algorithms, for BCH (31,16,7) code, is shown in Figure 22. From the later, we observe that the two algorithms are similar. Again, the Figure 23 shows that the two decoders are similar. The Figure 24 compares the performances of DDGA and Shakeel algorithms for BCH (31,26,3) code. We remark that our decoder is slightly better than the Shakeel one. The performances of DDGA and Shakeel algorithms, for BCH (127,113) code, are shown in Figure 25. From the simulation result, we observe that the performances of the two decoders are similar. The Figure 26 illustrates the performances of DDGA and Shakeel algorithms for QR (71,31,11) code. According to this figure we remark that our decoder outperforms Shakeel decoder about 2.5 at 10^{-5}. The Figure 27 illustrates the performances of DDGA and Shakeel algorithms for BCH (63,51,5) code. Then, we notice that our decoder is better then the Shakeel one. The Figure 28 presents the performances of DDGA and Shakeel algorithms for QR (127,64,21) code. According to this figure, we remark that our decoder outperforms Shakeel decoder by 2 dB at 10^{-5}. The Figure 29 shows the performance of DDGA and Shakeel algorithm for BCH (63,45,7) code. We observe that our decoder is slightly better than the Shakeel one. The Figure 30 presents the performance of DDGA and Shakeel

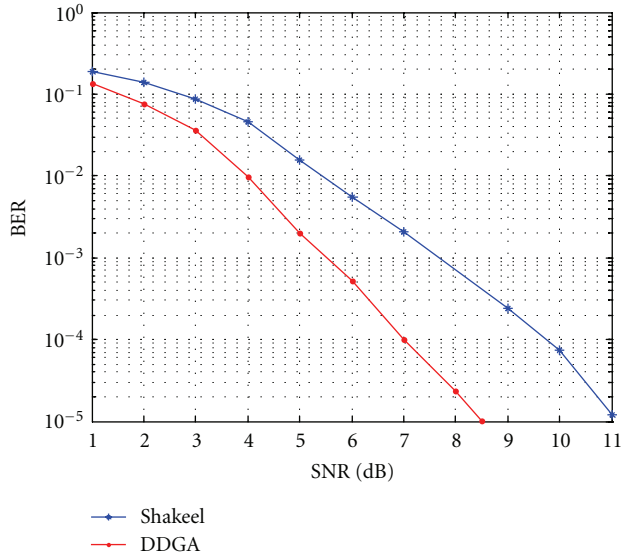


Figure 26: Performances of DDGA, and Shakeel algorithms for QR (71,36,11) code over RC.

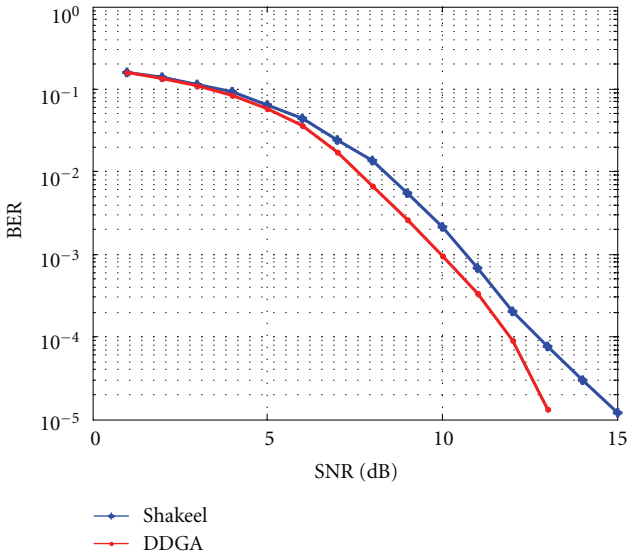


Figure 27: Performances of DDGA, and Shakeel algorithms for BCH (63,51,5) code over RC.

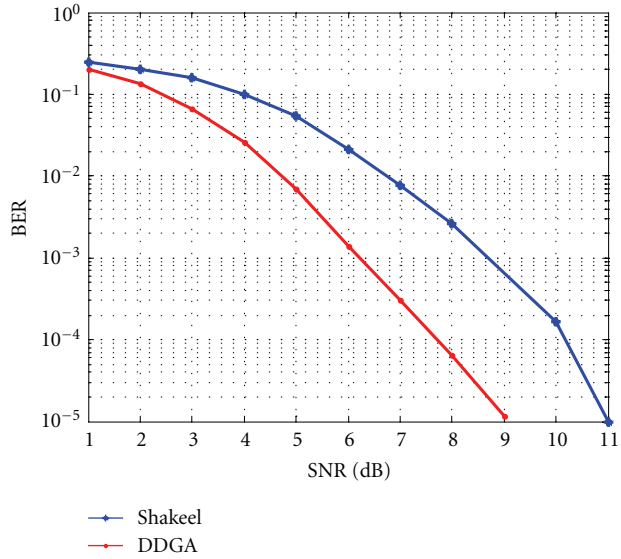


Figure 28: Performances of DDGA, and Shakeel algorithms for BCH (127,64,21) code over RC.

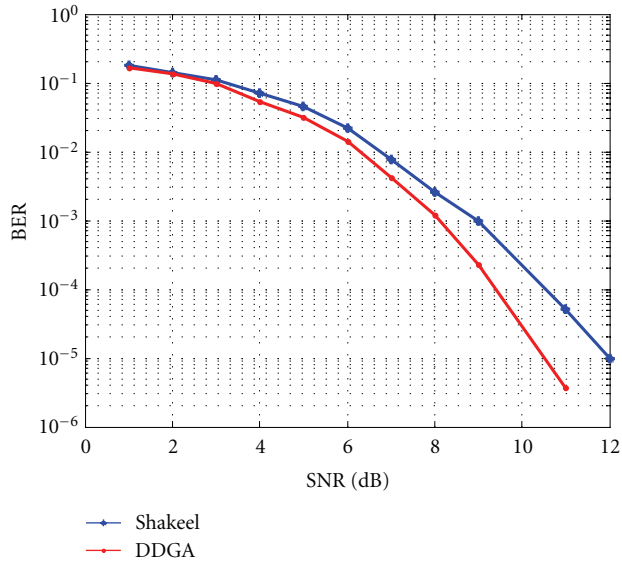


Figure 29: Performances of DDGA, and Shakeel algorithms for BCH (63,45,7) code over RC.

algorithms for QR (103,52,19) code. According to this figure, we remark that our decoder outperforms Shakeel decoder by 3 dB at 10^{-5}.

6. Conclusion

In this paper, we have proposed a new decoder based on GA for linear block codes. The simulations applied on some BCH and QR code, show that the proposed algorithm is an efficient soft-decision decoding algorithm. Emphasis was made on the effect of the genetic algorithm parameters and the code length on the decoder performance. A comparison was done in terms of bit error rate performance and complexity aspects of the decoder. The proposed algorithm has an advantage compared to competitor decoders developed by Maini and Shakeel.

The crossover operator developed in this paper can be generalized and successfully applied to other optimisation problems.

The obtained results will open new horizons for the artificial intelligence algorithms in the coding theory field.

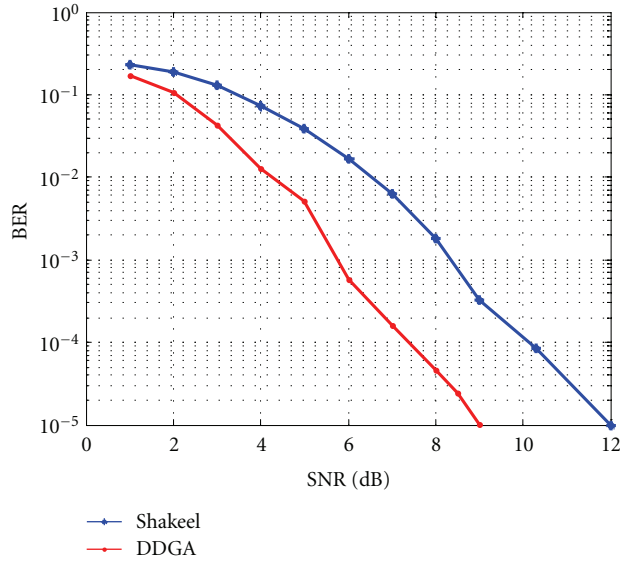


Figure 30: Performances of DDGA, and Shakeel algorithms for QR (103,52,19) code over RC.

References

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

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

[3] H. Maini, K. Mehrotra, C. Mohan, and S. Ranka, "Soft decision decoding of linear block codes using genetic algorithms," in *Proceedings of the IEEE International Symposium on Information Theory*, p. 397, Trondheim , Norway, 1994.

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

[5] F. A. Cardoso and D. S. Arantes, "Genetic decoding of linear block codes," in *Proceedings of the Congress on Evolutionary Computation*, pp. 2302–2309, Washington, DC, USA, 1999.

[6] I. Shakeel, "GA-based Soft-decision decoding of linear block codes," in *Proceedings of the International Confrrence on Telecommunications Congress on Evolutionary Computation*, pp. 13–17, Doha, Qatar, April 2010.

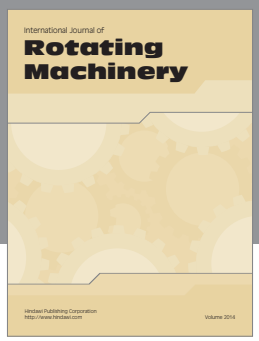
[7] J. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, Mich, USA, 1975.

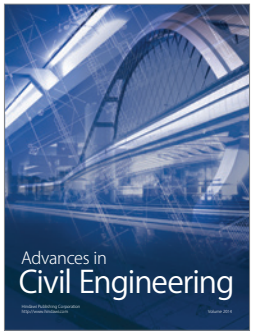
[8] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Mass, USA, 1989.

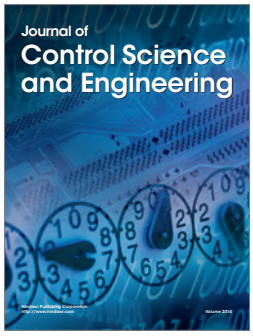
[9] M. P. C. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Transactions on Information Theory*, vol. 41, no. 5, pp. 1379–1396, 1995.

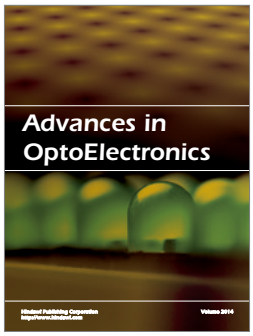
[10] M. P. C. Fossorier, S. Lin, and J. Snyders, "Reliability-based syndrome decoding of linear block codes," *IEEE Transactions on Information Theory*, vol. 44, no. 1, pp. 388–398, 1998.

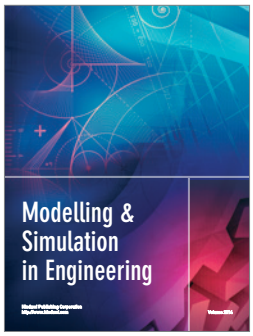
[11] D. Chase, "Class of algorithms for decoding blockcodes with channel measurement information," *IEEE Transactions on Information Theory*, vol. 18, no. 1, pp. 170–182, 1972.

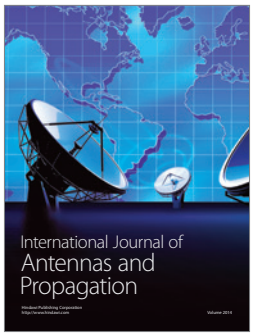


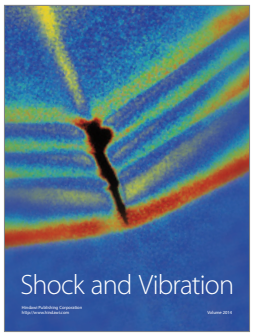


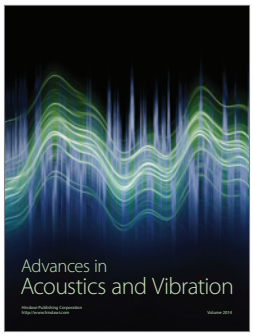












Hindawi

Submit your manuscripts at
http://www.hindawi.com