Syndrome Weight Decision based Genetic Algorithm Decoder for LDPC Codes

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

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

My Ahmad Faqihi
SIME Lab, ENSIAS,
Mohammed V-Souissi
University, Rabat, Morocco

ABSTRACT
Genetic algorithms are successfully used for decoding some classes of error correcting codes, and offer very good performances when solving large optimization problems .This article introduces a new Decoder based on Genetic Algorithm and the Syndrome Weight decision (GADSW) for decoding Low Density Parity Check (LDPC) codes. The performances of (GADSW) decoder are very good compared to sum-product decoder, which prove its efficiency.

General Terms
Error correcting codes

Keywords
Genetic Algorithm, syndrome weight, Sum-product decoder, LDPC code, Error correcting codes.

1. INTRODUCTION
Error correcting codes have been successfully implemented in wireless communication to offer error-free transmission with high spectral efficiency. Coding techniques create codewords by adding redundant information to the user information vectors. Decoding algorithms search the most likely transmitted codewords related to the received one as depicted in Figure 1.

Recently artificial intelligence techniques were introduced to solve this problem. Among the related works, the decoding of linear block codes using algorithm A* [1], genetic algorithms [2], [3] and neural networks [4].

There are two classes of error correcting codes: convolutional codes and block codes. The class of block codes contains two subclasses: nonlinear codes and linear codes. A Low-density parity-check (LDPC) codes are a class of linear block codes [5].

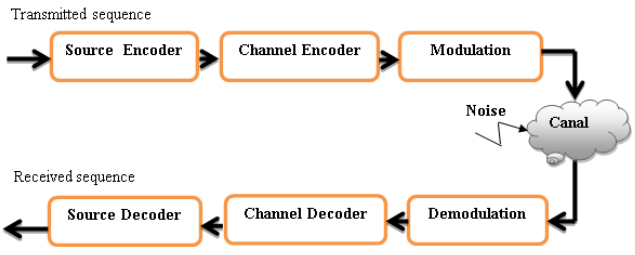

Figure 1: Simplified model of communication systems

LDPC codes were invented by Robert Gallager [6] in his PhD thesis. Soon after their invention, they were largely forgotten, and reinvented several times for the next 30 years. Their comeback is one of the most intriguing aspects of their history, since two different communities' reinvented codes similar to Gallager's LDPC codes at roughly the same time, but for entirely different reasons. The name comes from the characteristic of their parity-check matrix which contains only a few 1's in comparison to the amount of 0's [7]. Their main advantage is that they provide a performance which is very close to the capacity for a lot of different channels and linear time complex algorithms for decoding. Furthermore are they suited for implementations that make heavy use of parallelism.

LDPC codes have emerged as the best error correcting codes with close to the theoretical Shannon limit performance. When these codes are decoded using Gallager's iterative probabilistic decoding method, also known as the sum-product algorithm or belief propagation algorithm, their empirical BER performance are found to be excellent [8],[9],[10]. This is true when the length of the code vector is large enough.

The LDPC sum-product decoding algorithm [6],[9], makes an estimation of the A Posteriori Probability (APP) of each symbol as a function of the received symbol and the properties of the channel. In this sense, the decoding algorithm does require to know the signal-to-noise ratio in the channel.

In this paper, we introduce a new Genetic algorithm decoder using a decision based on the syndrome weight (GADSW) for LDPC codes. Simulations show that the GADSW decoder provides good performances compared to sum-product decoder for LDPC codes.

This paper is organized as follows. In section II, we introduce Genetic algorithm; Section III presents GADSW our decoder and analyzes their performances. Finally, Section IV presents the conclusion and future trends.

2. GENETIC ALGORITHM
Genetic algorithms are heuristic search algorithms premised on the natural selection and genetic [2],[3],[11]. It is a non-mathematical, non-deterministic, but stochastic process or algorithm for solving optimization problems. The concept of genetic algorithm was introduced by John Holland [4] in 1975, it is defined by:

- *Individual or chromosome:* a potential solution of the problem, it's a sequence of genes.
- *Population:* a set of points of the research space.
- *Environment:* the space of research.
- *Fitness function:* the function to maximise / minimise.
- *Encoding of chromosomes:* it depends on the treated problem, the famous known schemes of coding are: binary encoding, permutation encoding, value encoding and tree encoding.
- *Operators of evolution:*

Selection: it permits to select the best individuals to insert in the intermediate generation.

Crossover: For a pair of parents (p1, p2) it permits to create two children (ch1; ch2), with a crossover probability Pc .

Mutation: The genes of the individual are muted according to the mutation rate m_r and the mutation probability Pm. The typical steps in the design of genetic algorithm are described below and illustrated in the Figure 2:

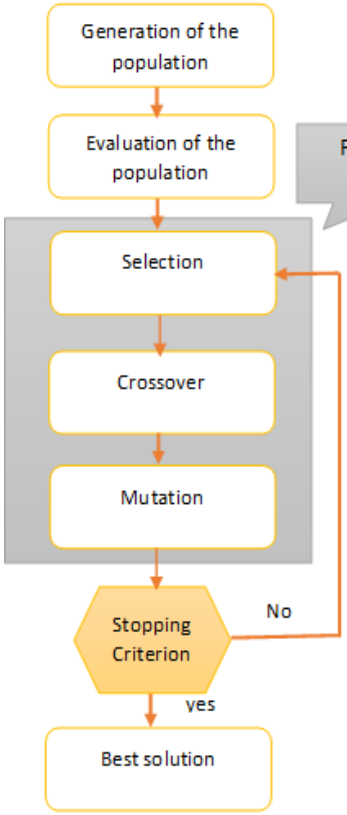

Figure 2. A model of genetic algorithm

3. GENETIC ALGORITHM DECODER BASED ON SYNDROME WEIGHT (GADSW)

This decoder can be implemented in three global steps. In the first step the decoder calculate the syndrome of the received vector, so if the syndrome is null, the decoder returns a decoded vector equal to the binary decision of received one, else the decoder turn GA step *P* times with an initial population randomly generated, each execution of GA return the best individuals as a candidate of the decision step, and lastly the decoder returns the decoded vector having the smallest values of the syndrome weight.

The decoding based Genetic Algorithm and syndrome weight decision is depicted in Figure 3.

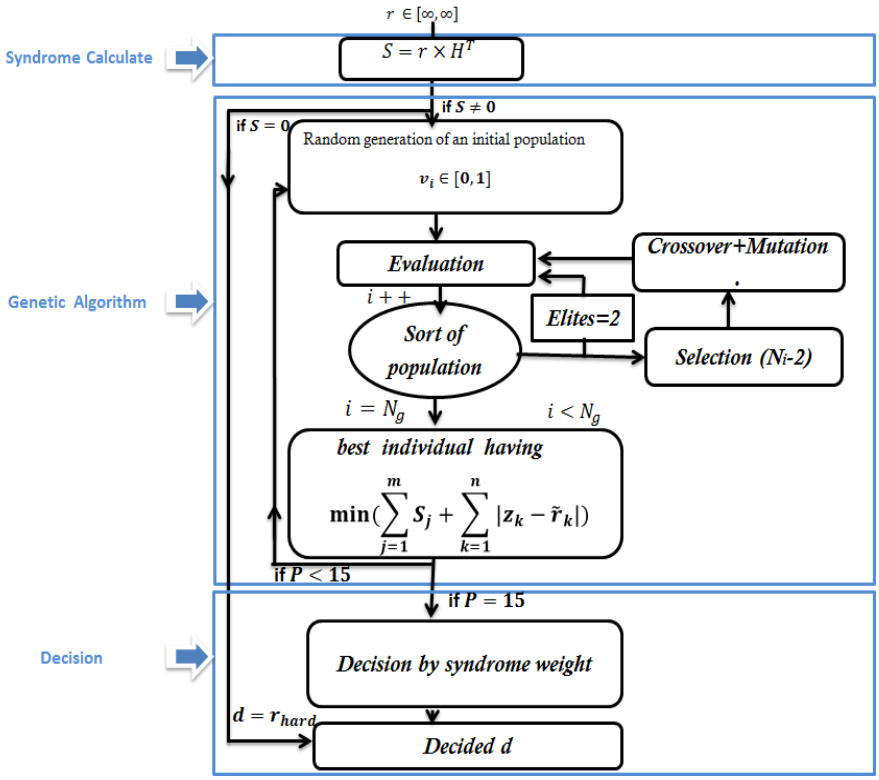

Figure 3: Basic structure of GADSW.

3.1 Method of decision based on Syndrome Weight

In this work we use a method of decision based on Syndrome Weight (SW). Since the decision made by GA (z) is not a codeword in all case, we have chosen the criteria of syndrome weight. For this, we do intensive simulations.

We have calculated the syndrome weight of all the words with weight error equal to q, where q∈{1,2, ..., 7}, and we have plotted the maximum and the minimum value of the syndrome weight (the length of vector is 60) according to the figure 4. After several simulations, we have observed a relation between the syndrome weight of the received vector and the distance of the codeword and that one as shown in Figure 4 and Figure 5.

The table 1 gives the statistic results related to the proposed method. This table presents the number of vectors having the minimum syndrome weight (NbMin), the number of vectors having the maximum syndrome weight (NbMax), and the number of vectors having the syndrome weight between the NbMax and the NbMin, also the mean of the SW.

The figure 4 presents the relation between the error weight and the maximum and minimum values of the syndrome weight. And figure 5 presents the relation between the error weight and the mean of the syndrome weight.

The two figures show that when weight of syndrome is close to zero, the less the distance between the codeword and the received vector is close to zero. So, the syndrome weight is close to zero, the closest the received vector is to the codeword sent.

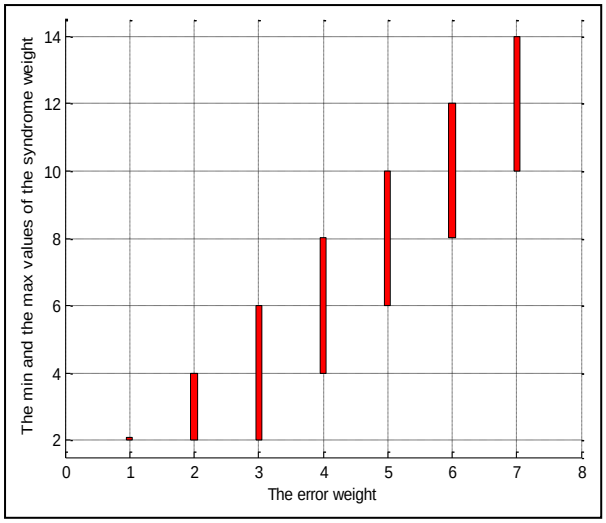

Figure 4: Relation between the error weight and the minimum and the maximum values of the syndrome weight

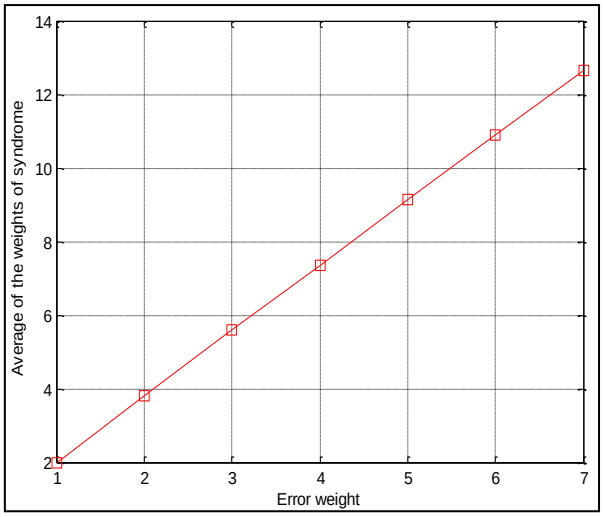

Figure 5: Error weight according to the average of the syndrome weight

Table 1: Statistic Result

Error weight	1	2	3	4	5	6	7
Total vectors	**60**	**117**	**172**	**225**	**276**	**325**	**372**
The number of vectors having the minimum SW	**60**	**12**	**2**	**8**	**20**	**38**	**62**
The number of vectors having the maximum SW	-	**105**	**138**	**161**	**176**	**183**	**182**
The number of vectors having the SW between Max and Min	**0**	**0**	**32**	**56**	**80**	**104**	**128**
The mean of SW	**2**	**3,79**	**5,58**	**7,36**	**9,13**	**10,89**	**12,64**

3.2 The proposed algorithm

Let **C** a low-density parity-check (LDPC) code, and let $(\mathbf{r_i})_{1\le i\le n}$ be the received sequence over a communication channel with an AWGN noise variance $\boldsymbol{\sigma = N_0/2}$, where $\boldsymbol{N_0}$ is noise power spectral density.

Let $\boldsymbol{N_i}$, $\boldsymbol{N_e}$, $\boldsymbol{N_g}$ and $\boldsymbol{P}$ denote, respectively, the population size, the number of elite members, the number of generations and the number of GA execution.

Let $\boldsymbol{p_c}$ and $\boldsymbol{p_m}$ be the crossover and the mutation rates.

Let **U=[0,1],** $\tilde{\boldsymbol{r}} \in \boldsymbol{U}$ is the received vector transformed into [0,1] interval using the hyperbolic tangent (3).

$$\tau: IR \rightarrow U \qquad (1)$$

$$\tau: r \rightarrow \tilde{r} \qquad (2)$$

$$\tilde{r}_i = \frac{1}{2} * (1 + tanh(r_i)) \qquad (3)$$

The decoding-based on Genetic Algorithm and decision by the syndrome weight is depicted on Figure 3.The steps of the decoder are as follows:

Step1: The decoder calculates de syndrome of de received vector (eq.4).

$$\boldsymbol{S = r^{hard} * H^T} \qquad (4)$$

where $\boldsymbol{r^{hard}}$ is the hard decision of $\boldsymbol{r}$ (eq.5), and $\boldsymbol{H^T}$ is the transpose of the matrix $\boldsymbol{H}$.

$$r_i^{hard} = \begin{cases} 1 & if \quad r_i > 0 \\ 0 & Otherwise \end{cases} \qquad (5)$$

If $\boldsymbol{S_i = 0} \quad \forall \boldsymbol{i} \in [\boldsymbol{0}, \boldsymbol{n-k}]$, then a valid code vector $\boldsymbol{d}$ is obtained by $\boldsymbol{d = r^{hard}}$. Otherwise the decoder applies the genetic algorithm **P** times as below:

Step2. Generate an initial random population containing $\boldsymbol{Ni}$ soft vectors $\boldsymbol{v_i}$ of $\boldsymbol{n}$ components ($\boldsymbol{v_i} \in [\boldsymbol{0}, \boldsymbol{1}]$).

Step3. Compute the fitness of each individual in the population:

The fitness function is the sum of the syndrome weight of de candidate, and the distance between the candidate vector and the received one (eq.6).

$$fitness = \sum_{j=1}^{m} S_j + \sum_{i=1}^{n} |z_i - \tilde{r}_i| \qquad (6)$$

Where

$$z_i = \begin{cases} 0 & if \quad \tilde{r}_i < v_i \\ 1 & Otherwise \end{cases} \qquad (7)$$

And

$$S = z * H^T \qquad (8)$$

z is the solution candidate of GA.

$\boldsymbol{m}$ and $\boldsymbol{n}$, denote, respectively, the number of rows of the parity check matrix $\boldsymbol{H}$, and the code vector length.

Step4. The population is sorted in ascending order of candidates' fitness value defined by (eq.6).

Step5. The best two candidates ($\boldsymbol{Ne = 2}$) of each generation are inserted in the next one.

Step6. The other $\boldsymbol{Ni - Ne}$ members of the next generation are generated as follows:

Substep6.1. *Selection operation:* a selection operation that uses the tournament selection method is applied in order to identify the best parents($\boldsymbol{v_1, v_2}$), on which the reproduction operators are applied.

Substep6.2.*Crossover operation:* Create new vectors ($\boldsymbol{v'_1, v'_2}$) "children", with a given probability rate $\boldsymbol{p_c = 0.95}$. We use Ring Crossover (RC) [12].

Substep6.3.Mutation operator: To complete the new generation, children are mutated by introducing random changes with a given probability rate $\boldsymbol{p_m = 0.01}$ to single parent.

The best member from the last generation for each GA run is returned as the candidate for the next step.

Step7. Decision Step:

A set of **P** decoded vectors **z**, is obtained applying GA algorithm step **P** times, where **P** is an arbitrary integer value heuristically optimized (partial solutions), this step generates the final solution (decoded vector $\boldsymbol{d}$).

Substep7.1. Calculate de syndrome weight for each vector provided by GA step as follows:

$$S'_i = \sum_{j=1}^{m} S_j \ , \qquad 1 < i < P$$

Substep7.2. Find the minimum value of syndrome weight:

$$M = \min_{1<i<P} (S'_i)$$

Substep7.3. Generate the final solution, i.e, a decoded vector $\boldsymbol{d}$ having the smallest value $\boldsymbol{M}$ of syndrome weight.

3.3 Simulation Results and Discussions

In order to prove the effectiveness of GADSW, we do intensive simulations.

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 Eb/N0).

Table 2: Default parameters of simulations

Simulation parameter	Parameter value
Pc (crossover rate)	0.95
Pm (mutation rate)	0.01
Ng (generation number)	25
Ni (population size)	500
Ne (elite number)	2
Channel	AWGN
Modulation	BPSK
Minimum number of bit errors	100
Minimum number of bloc	300
P(GA runs)	15
Default code	Regular LDPC(60,30)
Type of crossover	Ring Crossover (RC)
Type of selection	Tournament

Comparison between Various Crossover Methods in GADSW

In the Figure 6, we compare results obtained using the ring crossover, single point, two points and tree points crossover, in GADSW for regular LDPC(60,30).

Simulation results show that the ring crossover is better than all other ones. The gain between the RC and the tree other crossovers is 3.5 dB at 10^{-3}.

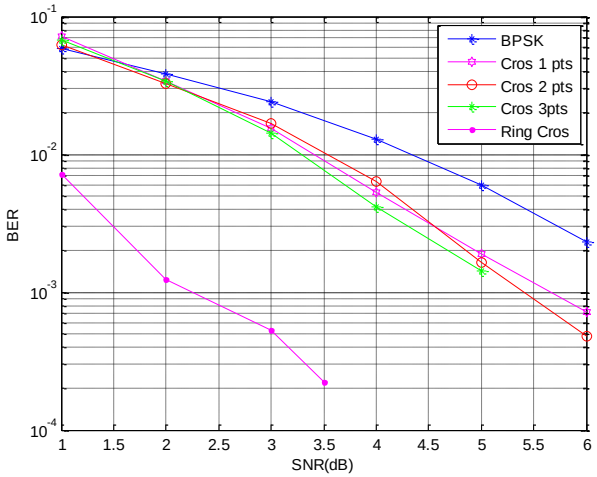

Figure 6: Comparison between different crossover operators in GADSW for regular LDPC(60,30).

Comparison between Different Selection Methods in GADSW.

In the Figure 7, we present a comparison between the results obtained using tournament, linear ranking, Roulette Wheel, Rank, Elitism and random selection in GADSW for regular LDPC (60, 30). Simulation results show that the tournament selection is better than all other selection.

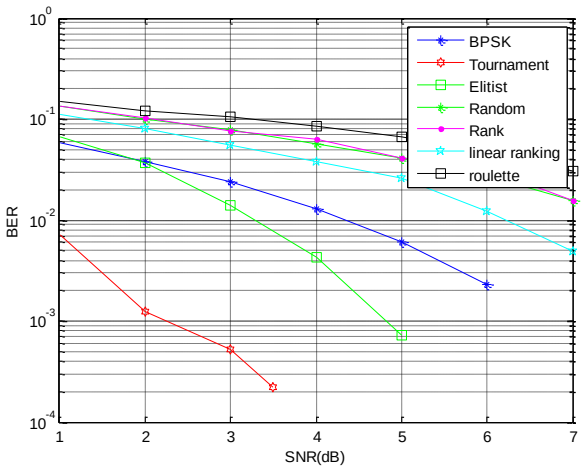

Figure 7: Comparison between different selection operators in GADSW for regular LDPC (60,30).

Comparison with Sum-Product Decoder

Our new decoder has been compared with the Sum-Product Decoder for regular LDPC(60,30), LDPC(75,45) and LDPC(96,48) codes. The results are given in Figure 8, figure 9 and Figure 10:

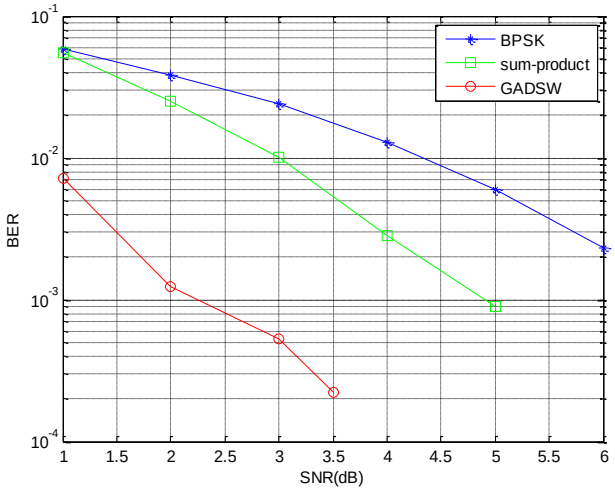

Figure 8: Performances of GADSW and sum-product decoder for regular LDPC (60, 30) code.

The figure 8 shows that the GADSW provides good performances compared to sum-product decoder for regular LDPC(60,30) code. The gain between the GADSW and sum-product decoder is 2.7 dB at10^{-3}.

Figure 9 compares the performances of GADSW with sum-product decoder for regular LDPC(75,45) code. We remark that the GADSW is better than sum-product decoder. The gain between the GADSW and sum-product decoder is 0.8 dB at 10^{-4}.

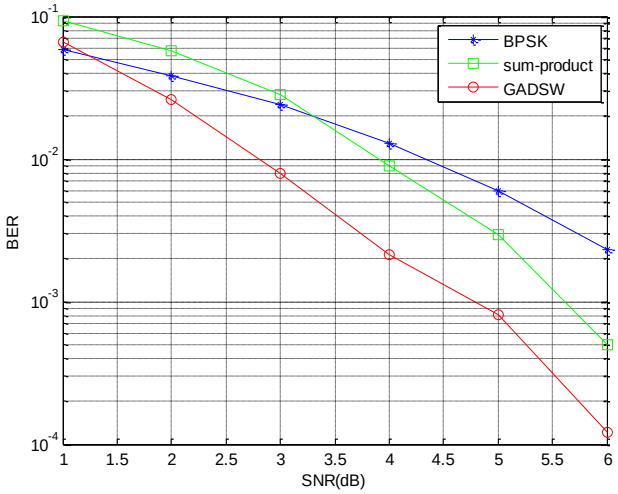

Figure 9: Performances of GADSW and sum-product decoder for regular LDPC (75, 45) code.

Figure 10 compares the performances of GADSW with sum-product decoder. We remark that the GADSW is better than sum-product decoder for regular LDPC(96,48) code. The gain between the GADSW and sum-product decoder is 0.5 dB at 10^{-3}.

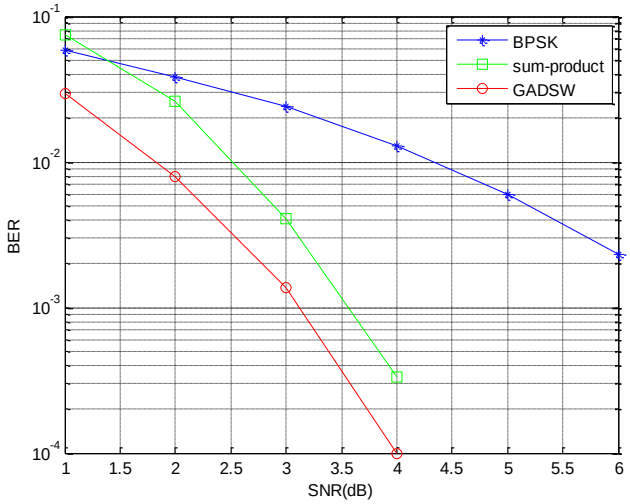

Figure 10: Performances of GADSW and sum-product decoder for regular LDPC (96, 48) code.

4. CONCLUSIONS

In this paper, we have proposed a new decoder based on GA and the decision by the syndrome weight for LDPC codes. The simulations applied on some LDPC code, show that the proposed algorithm is an efficient algorithm. The comparison between our GADSW and sum-product decoder shows that our decoder is better in terms of performances. The obtained results will open new horizons for the artificial intelligence algorithms in the coding theory field.

5. REFERENCES

[1] Han, Y. S., Hartmann, C. R. P., and Chen, C.-C. 1991 Efficient maximum likelihood softdecision decoding of linear block codes using algorithm A*, Technical Report SUCIS- 91-42, School of Computer and Information Science, Syracuse University, Syracuse, NY 13244.

[2] Maini, H.S. , Mehrotra, K. G, Mohan, C., Ranka, S. 1994 Genetic Algorithms for Soft Decision Decoding of Linear Block Codes, Journal of Evolutionary Computation, Vol.2, No.2, pp.145-164.

[3] Azouaoui, A. , Belkasmi, M., and Farchane, A. 2012 Efficient Dual Domain Decoding of Linear Block Codes Using Genetic Algorithms, Journal of Electrical and Computer Engineering, vol. 2012, Article ID 503834, 12 pages.

[4] Holland, J. 1975 Adaptation in Natural and Artificial Systems, University of Michigan Press.

[5] Khalifa, O.O., khan S., Zaid M., and Nawawi M. 2008 Performance Evaluation of Low Density Parity Check Codes, International Journal of Computer Science and Engineering, pp. 67-70, Spring.

[6] Gallager, R. G. 1963 Low-Density Parity-Check Codes. The MIT Press.

[7] MacKay, D.J.C. 1999 "Good error-correcting codes based on very sparse matrices" IEEE Trans. Inform.Theory, 45,399-431.

[8] Richardson, T., Shokrollahi, A. and Urbanke, R. 2001 Design of capacity approaching irregular Low-Density Parity-Check codes. IEEE Trans. Inform.Theory, 47,619-637.

[9] Richardson, T. , and Urbanke, R. 2001 The capacity of Low-Density Parity-Check codes under message-passing decding. IEEE Trans. Inform.Theory, 47,599-618.

[10] MacKay, D.J.C. and Neal, R.M. 1997 Near Shannon Limit Performance of Low-Density Parity-check Codes. IEE Elect. Lett., 33, 457-458.

[11] Janikow, C. Z. and Michalewicz, Z. 1991 An experimental comparison of binary and floating point representations in genetic algorithms. in Proc. 4th Int. Conf. Genetic Algorithms, pp. 31-36.

[12] Kaya, Y. Uyar, M. and Tekin, R. 2011 A Novel Crossover Operator for Genetic Algorithms: Ring Crossover. presented at CoRR.