Int. J. Communications, Network and System Sciences, 2015, 8, 358-366
Published Online September 2015 in SciRes. http://www.scirp.org/journal/ijcns
http://dx.doi.org/10.4236/ijcns.2015.89034

Serial Genetic Algorithm Decoder for Low Density Parity Check Codes

Hasna Chaibi

SIME Lab., ENSIAS, Mohammed V University, Rabat, Morocco
Email: has.chaibi@gmail.com

Received 19 August 2015; accepted 20 September 2015; published 23 September 2015

Abstract

Genetic algorithms are successfully used for decoding some classes of error correcting codes, and offer very good performances for solving large optimization problems. This article proposes a new decoder based on Serial Genetic Algorithm Decoder (SGAD) for decoding Low Density Parity Check (LDPC) codes. The results show that the proposed algorithm gives large gains over sum-product decoder, which proves its efficiency.

Keywords

Serial Genetic Algorithm, Sum-Product Decoder, Sigmoidal Function, LDPC Code, Error Correcting Codes

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 latter 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 codewords 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 a NP-hard problem and has been approached in different ways. Recently artificial intelligence techniques were introduced to solve this problem. Among the related works, the decoding of linear block codes uses algorithm A* [2], genetic algorithms [1] [3] and neural networks [4].

LDPC codes were invented by Gallager [5] in his PhD thesis. Soon after their invention, they were largely

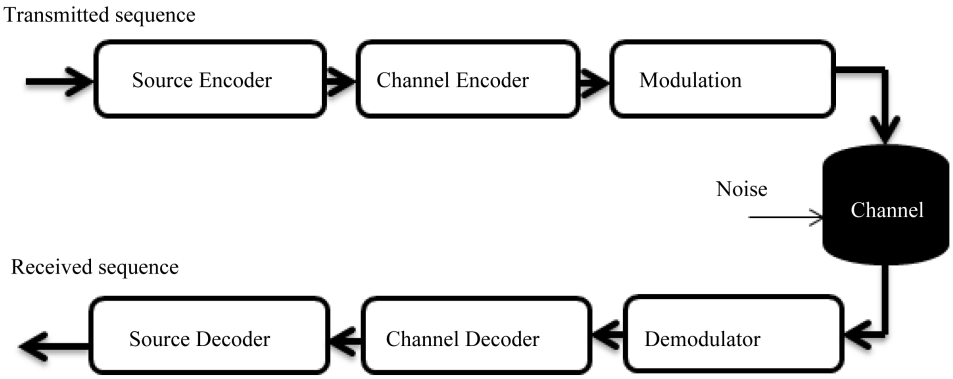

Figure 1. Communication system model.

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 are similar to Gallager's LDPC codes at roughly the same time, but for entirely different reasons.

Low Density Parity Check (LDPC) codes are a class of linear block codes. 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. 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, they are suited for implementations that make heavy use of parallelism [6].

LDPC codes have emerged as the best error correcting codes 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 is found to be excellent [7]-[9]. This is true when the length of the code vector is large enough.

The LDPC sum-product decoding algorithm [5] [10] 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 is required to know the signal-to-noise ratio in the channel.

In this article we introduce a new Serial Genetic Algorithm Decoder (SGAD) LDPC codes. In effect, a comparison with sum-product decoder, which is currently the most successful algorithm for LDPC, shows its efficiency, and gives the higher performances.

This paper is organized as follows. In Section II, we introduce Genetic algorithm; in Section III, we present SGAD (our decoder) and analyze the 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 [3] [11] [12]. It is a non-mathematical, non-deterministic, but stochastic process or algorithm for solving optimization problems. The concept of genetic algorithm was introduced by Holland [13] 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 **Figure 2**:

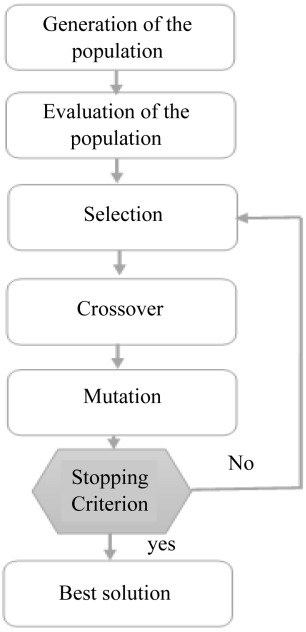

Figure 2. A simplified model of a genetic algorithm.

3. Serial Genetic Algorithm Decoder (SGAD)

This work is a serialization of the genetic algorithm, the decoder computes the syndrome of the received vector, so if the syndrome is null, the decoder returns the decoded vector equal to the binary decision of received one, else the decoder turn the first execution of GA with an initial population randomly generated (**PoP**), in the second execution of GA, the decoder inserts a subpopulation (**SubPoP**) (a some elites of the last generation for first GA execution) in the initial population (**PoP**). Then the decoder returns the best individuals as a decoded word. The pseudo code of SGAD is shown in **Figure 3**. The parameters are the population size (N_i); the subpopulation size (P), the number of generation (N_g).

3.1. The SGAD Decoder

Let C a Low Density Parity Check (LDPC) code, and let $(r_i)_{1\le i\le n}$ be the received sequence over a communication channel with noise variance $\sigma = N_0/2$, where N_0 is noise power spectral density.

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

Let P the size of the subpopulation.

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

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

$$\tau : IR \to U \tag{1}$$

$$\tau : r \to \tilde{r} \tag{2}$$

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

The decoding-based on serial genetic algorithm is depicted in **Figure 3**. The steps of the decoder based on SGA are as follows:

***Step* 1.** The decoder calculates de syndrome of de received vector (Equation (4)).

$$S = r^{hard} * H^T \tag{4}$$

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

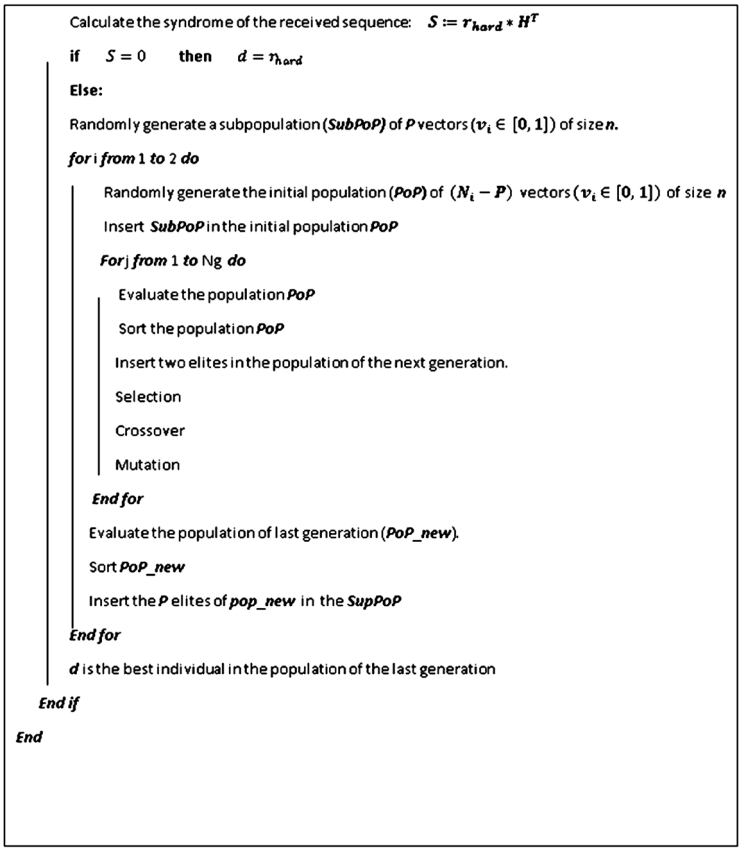

Figure 3. Pseudo code of SGAD.

$$r_i^{hard} = \begin{cases} 1 \text{ if } & r_i > 0 \\ 0 \text{ Otherwise} & \end{cases} \tag{5}$$

If $S_i = 0 \quad \forall i \in [0, n-k]$, then a valid code vector d is obtained by $d = r^{hard}$. Otherwise we apply the steps of GA:

***Step* 2.** Generate an initial random subpopulation (**SubPoP**) of P soft vectors v_i of n components ($v_i \in [0,1]$).

Step* 3**. For ***i from **1** to **2**.

***Substep* 3.1.** Generate an initial random population (**PoP**) of $(N_i - P)$ soft vectors v_i of n components ($v_i \in [0,1]$).

***Substep* 3.2.** Insert the subpopulation (**SubPoP**) in the initial population (**PoP**).

***Substep* 3.3.** Compute the fitness of each individual in the population:

The fitness function is the syndrome weight of de candidate, and the distance between the candidate vector and the received one (Equation (2)).

$$\text{fitness} = \sum_{j=1}^{m} S_j + \sum_{i=1}^{n} \left| z_i - \tilde{r}_i \right| \tag{6}$$

where

$$z_i = \begin{cases} 0 \text{ if } & \tilde{r}_i < v_i \\ 1 \text{ Otherwise} & \end{cases} \tag{7}$$

and

$$S = z * H^T \tag{8}$$

z is the solution provided by GA.

m and n, denote, respectively, the number of rows of the parity check matrix H, the code vector length.

***Substep* 3.4.** The population is sorted in ascending order of member's fitness defined by (6).

***Substep* 3.5.** The best two members ($Ne = 2$) of each generation are inserted in the next one.

***Substep* 3.6.** The other $Ni - Ne$ members of the next generation are generated as follows:

***Subsubstep* 3.6.1.** *Selection operation*: a selection operation that uses the tournament selection method is applied in order to identify the best parents (v_1, v_2), on which the reproduction operators are applied.

***Subsubstep* 3.6.2.** *Crossover operation*: Create new vectors (v_1', v_2') "children", with a given probability rate $p_c = 0.95$. We use Ring Crossover (RC) [14].

***Subsubstep* 3.6.3.** *Mutation operator*: To complete the new generation, children are mutated by introducing random changes with a given probability rate $p_m = 0.01$ to single parent.

The **P** best member from the last generation for the first execution of GA are inserted in the subpopulation (**SupPoP**).

***Step* 4.** The decoder decision is z the best member from the last generation after two GA runs.

3.2. Simulation Results and Discussions Related to SGAD

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

The simulations where made with default parameters outlined in **Table 1**. The performances are given in terms of BER (bit error rate) as a function of SNR (Signal to Noise Ratio Eb/N0).

3.2.1. Effect of the Subpopulation Size on the Performance

The number of size of the subpopulation (the number of the elites members of the last generation inserted in the initial population for second execution of GA) used in our decoder has an effect in the performances quality. **Figure 4** shows this effect.

Figure 4 shows that the performance improves by increasing the number of size of the subpopulation until 20, after, the performance decrease by increasing this number. So the number of size of the subpopulation must be chosen carefully to give good impact into the performances of our decoder.

3.2.2. Comparison with Sum-Product Decoder

Our new decoder has been compared with the Sum-Product Decoder for regular LDPC (63,37), LDPC (73,45) and

Table 1. Default parameters.

Simulation parameter	*Parameter value*
Pc (crossover rate)	0.97
Pm (mutation rate)	0.01
Ng (generation number)	30
Ni (population size)	300
Ne (elite number)	2
P (subpopulation size)	20
Channel	AWGN
Modulation	BPSK
Minimum number of bit errors	100
Minimum number of bloc	1000
Default code	LDPC (63,37)
Type of crossover	Ring Crossover (RC)
Type of selection	Tournament

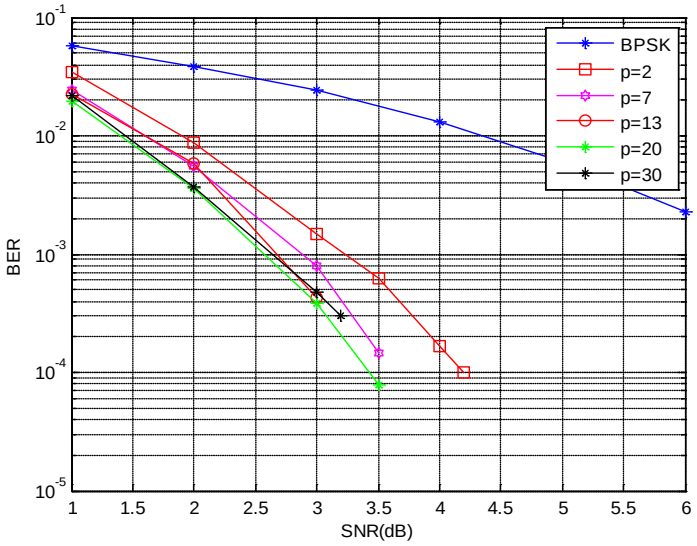

Figure 4. Performance of our decoder for different size of the subpopulation.

LDPC (96,48) codes. The results are given in **Figures 5-7**:

Figure 5 shows that the SGAD provides good performances compared to sum-product decoder for regular LDPC (63,37) code. The gain between the SGAD and sum-product decoder is 1.5 dB at 10^{-4}.

Figure 6 compares the performances of SGAD with sum-product decoder for regular LDPC (73,45) code. We remark that the SGAD is better than sum-product decoder. The gain between the SGAD and sum-product decoder is 1.5 dB at 10^{-4}.

Figure 7 compares the performances of SGAD with sum-product decoder. We remark that the SGAD is better than sum-product decoder for regular LDPC (96,48) code. The gain between the SGAD and sum-product decoder is 1 dB at 10^{-3}.

3.2.3. Study of Hyperbolic Tangent Sigmoidal Function

We have used the hyperbolic tangent function (Equation (3)) in our decoder based on serial genetic algorithm for transform the values of the received vector (r) in domain [0,1]. We have taken the value of the received word r in the interval [−4,4], and we have plotted the transformation using the hyperbolic tangent function (**Figure 8**).

From this figure we can say that the tangent hyperbolic function, applied component wise, is to map the components of the received vector $\boldsymbol{r}$ into [**0**,**1**] Also, shows that for one r lower than zero, when we increase the factor $\boldsymbol{\omega}$, $\tilde{\boldsymbol{r}}$ close to zero, and for one r is greater than 0, when we increase the factor $\boldsymbol{\omega}$, $\tilde{\boldsymbol{r}}$ close to one. we can conclude that, by reducing the value of the factor ω we increase the diversity of the received sequence component. For this, we have study the influence of the factor ω (Equation (9)) on the performances. **Figure 9** shows this influence on the performances.

$$\tilde{r}_i = \frac{1}{2} * \left(\tanh\left(\boldsymbol{\omega} * r_i\right) + 1\right), \quad r_i \in [-4, 4] \tag{9}$$

Figure 9 emphasizes the influence of values factor of the hyperbolic tangent function on the performance of SGAD. Decreasing this factor from 1 to 0.5, we can gain 1.5 dB at 10^{-4}.

4. Conclusion

In this paper, we have proposed a new decoder based on serial GA for LDPC codes. The simulations applied on some LDPC code show that the proposed algorithm is an efficient algorithm. The comparison between our SGAD and sum-product decoder shows that our decoder is better in terms of performances. We have shown that the performances must be improved by the factor of the sigmoidal function. The obtained results will open new

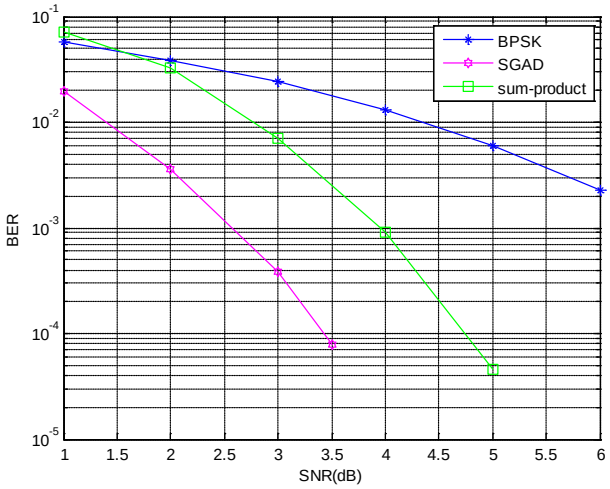

Figure 5. Performance of SGAD compared to sum-product decoder for a regular LDPC (63,37) code.

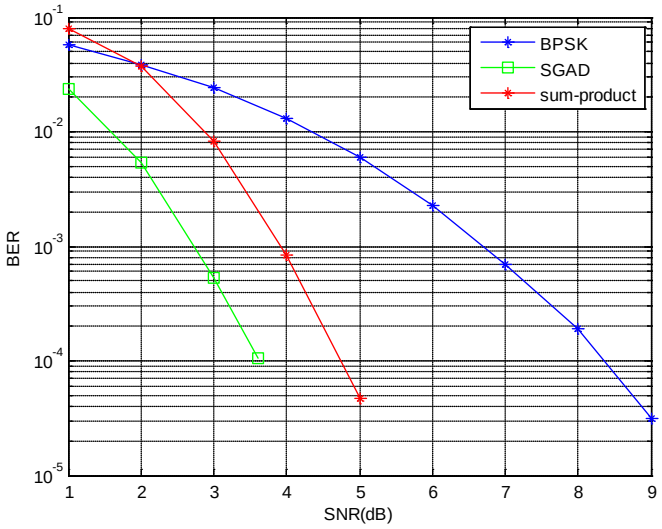

Figure 6. Performance of SGAD compared to sum-product decoder for a regular LDPC (73,45) code.

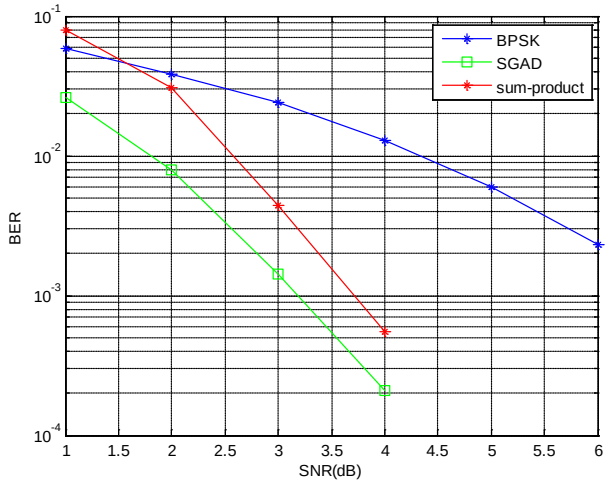

Figure 7. Performance of SGAD compared to sum-product decoder for a regular LDPC (96,48) code.

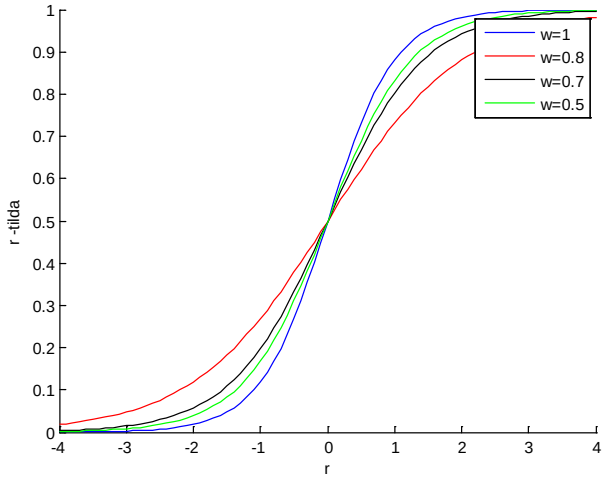

Figure 8. $\tilde{r}$ according to r, by varying the factor ω.

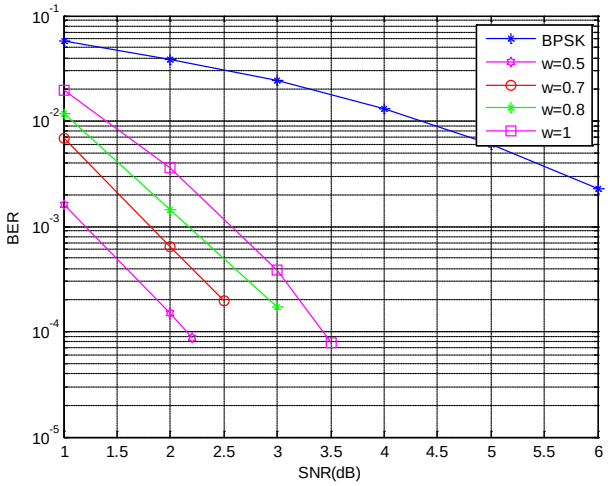

Figure 9. Performance of SDAG decoder varying the factor values ω in the hyperbolic tangent function for LDPC (63,37) code.

horizons for the artificial intelligence algorithms in the coding theory field.

References

[1] Maini, H.S., Mehrotra, K.G., Mohan, C. and Ranka, S. (1994) Genetic Algorithms for Soft Decision Decoding of Linear Block Codes. *Journal of Evolutionary Computation*, **2**, 145-164. http://dx.doi.org/10.1162/evco.1994.2.2.145

[2] Han, Y.S., Hartmann, C.R.P. and Chen, C.-C. (1991) Efficient Maximum Likelihood Soft Decision Decoding of Linear Block Codes Using Algorithm A*. Technical Report SUCIS-91-42, School of Computer and Information Science, Syracuse University, Syracuse, NY 13244.

[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*, **2012**, Article ID 503834, 12 p.

[4] Wu, J.-L., Tseng, Y.-H. and Huang, Y.-M. (2002) Neural Networks Decoders for Linear Block Codes. *International Journal of Computational Engineering Science*, **3**, 235-255. http://dx.doi.org/10.1142/S1465876302000629

[5] Gallager, R.G. (1963) Low-Density Parity-Check Codes. The MIT Press, Cambridge.

[6] 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*, **2**, 67-70.

[7] Richardson, T., Shokrollahi, A. and Urbanke, R. (2001) Design of Capacity Approaching Irregular Low-Density Parity-Check Codes. *IEEE Transactions on Information Theory*, **47**, 619-637. http://dx.doi.org/10.1109/18.910578

[8] Richardson, T. and Urbanke, R. (2001) The Capacity of Low-Density Parity-Check Codes under Message-Passing Decoding. *IEEE Transactions on Information Theory*, **47**, 599-618. http://dx.doi.org/10.1109/18.910577

[9] MacKay, D.J.C. (1999) Good Error-Correcting Codes Based on Very Sparse Matrices. *IEEE Transactions on Information Theory*, **45**, 399-431. http://dx.doi.org/10.1109/18.748992

[10] MacKay, D.J.C. and Neal, R.M. (1997) Near Shannon Limit Performance of Low-Density Parity-Check Codes. *Electronics Letters*, **33**, 457-458. http://dx.doi.org/10.1049/el:19970362

[11] Gordon, V.S. and Whitley, D. (1993) Serial and Parallel Genetic Algorithms as Function Optimizers. In: Forrest, S., Ed., *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, 177-183.

[12] Janikow, C.Z. and Michalewicz, Z. (1991) An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms. *Proc. 4th Int. Conf. Genetic Algorithms*, July 1991, 31-36.

[13] Holland, J. (1975) Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor.

[14] Kaya, Y., Uyar, M. and Tekin, R. (2011) A Novel Crossover Operator for Genetic Algorithms: Ring Crossover. Presented at CoRR.