A New Genetics-Aided Message Passing Decoding Algorithm for LDPC Codes

Jui-Hui Hung[1], Yi-De Lu, and Sau-Gee Chen[2]
Department of Electronics Engineering & Institute of Electronics
National Chiao Tung University
Hsinchu, Taiwan
[1]paholisi.nctu@gmail.com [2]sgchen@mail.nctu.edu.tw

***Abstract*—The popular LDPC decoding algorithms based on the message passing (MP) algorithm have high decoding performances. However, they are noticeably inferior to the maximum likelihood (ML) decoding algorithm. This work proposes a genetics-aided message passing (GA-MP) algorithm by applying a new genetic algorithm to MP algorithm. As a result, significantly performance improvement over MP algorithm can be achieved. Besides, compared with other genetic-aided decoding algorithms, the proposed algorithm has much better performances and much lower computational complexity. Simulations show that the decoding performance of GA-MP algorithm can achieve performances very close to the algorithm, while outperform MP algorithm. Besides, its performance will grow proportionally with the generation number without leveling off as observed in conventional MP algorithms, under high SNR condition.**

Keywords-channel coding; LDPC codes, genetic algorithm; ML decoding

I. INTRODUCTION

Low-density parity check (LDPC) code was introduced by Gallager [1], which can achieve decoding performance close to Shannon bound. Since the recent decade, LDPC code has been increasingly popular and adopted in many advanced communication systems such as IEEE 802.16e [2], 802.11n [3], 802.3an [4], and so on. LDPC code is often more favored than another near-Shannon limit correction code, that is, Turbo code, because it can achieve much higher throughputs with parallel operations.

The current most popular LDPC decoding algorithms are message-passing algorithm (MPA) family [5-10]. MPA family have very good correction performances, especially the sum-product algorithm (SPA) [5]. However, they are still noticeably inferior to the maximum-likelihood (ML) decoding algorithm [6]. Although ML decoding algorithm has the best performance, it is impractical for realizations. On the other hand, SPA is not suitable for hardware implementation either. As such, several approximation decoding algorithms with lower complexities were proposed, at the cost of performance degradations, for examples, the popular min-sum algorithm (MSA) [7] and normalized MSA (NMSA) [8]. There are also some modified SPA techniques [9][10] which approximate the performance of ML decoding algorithm. Nonetheless, they are either still too complicated or only get slight improvement.

Genetic algorithm (GA) is a kind of heuristic optimization methods first advocated by Holland [11]. GA simulates the natural evolution processes by a series of stochastic mutation and crossover processes, which expedites the search of the optimum solutions. GA has been widely utilized to solve many complicated problems in engineering areas. GA is often applied to existing sub-optimal algorithms and approach optimal solutions with much less effort than ML algorithm.

We will introduce the basic concept of GA in Section II. Next, Section III will introduces the proposed GA-MP algorithm which can achieves performances very close to ML decoding algorithm with much lower computational complexity. The proposed algorithm also outperforms the GA-assisted decoding techniques in [9] and [10], both in computational complexity and decoding performance. The simulation results are shown in Section IV, followed by conclusion.

II. BASIC CONCEPT OF GENETIC ALGORITHM

The basic concept of GA is as follows. When applied to channel decoding, first it encodes each candidate solution in a bit string, which is phrased "individual" or "chromosome". The evolution process of GA starts from an initial gene pool, which contains a certain number of randomly generated individuals, and a succession of generations follows. In each generation, one or some individuals will be stochastically selected from the gene pool according to their fitness computed by a pre-defined fitness function. These selected individuals will become the "ancestry" of this generation, and randomly hybridize (i.e., crossover) with possible mutations one by one to produce the "descendants". Note that the crossover or mutation steps are not definitely necessary for GA. Some GAs only apply one of them. Depending on strategies, the next generation gene pool will be constructed by these descendants and ancestries, or descendants only. The above steps will be iteratively executed until the termination criterion is satisfied, for example, maximal generation number is reached.

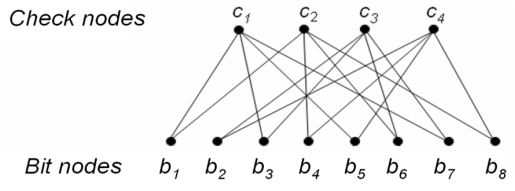

Figure 1. Tanner graph example of four check nodes and eight bit nodes.

978-1-4673-1881-5/12/$31.00 ©2012 IEEE

III. THE PROPOSED GA-MP ALGORITHM

Consider an $m \times n$ parity-check matrix H of a LDPC code, represented by Tanner graph as shown in the Fig. 1 example. It is composed of n bit nodes b_j, j=1 to n, and m check nodes c_i, i=1 to m. Those bit nodes and check nodes are connected by edges defined by nonzero entries of H.

In conventional LDPC iterative decoding algorithms, the magnitude of belief reliability (MBR) of a bit node value (i.e., the log-likelihood ratio (LLR) value before hard decision) is a basis for decoding. If the MBR magnitude of a bit node is relatively larger than others, the bit will be regarded as a more reliable one. However, a large MBR still may lead to an incorrect hard-decision result. Those incorrect results with high MBRs can make the whole LDPC decoding a failure, even under high iteration numbers. Hence, the problem is to find out those bit nodes without errors as much as possible. That is what this work intend to do, by proposing a genetics-aided decoding algorithm, as follows.

A. Overviewof the proposed GA-MP algorithm

At first, GA-MP uses the belief reliability sets in each iteration of an MP algorithm (MPA) as the individuals. Then, it establishes an initial gene pool for future evolution. For example, if the maximum iteration number of the MPA is N, then there will be N individuals in the initial gene pool. Besides, in each generation, a specified fitness function (to be detailed in Section III-B) is used to compute the fitness value of all individuals. GA-MP will select the individual with the best fitness value from the gene pool as the ancestry of this generation. The ancestry will mutate into N individuals according to belief-reliability mutation mechanism (as will be described in Section III-C). Then, it performs 1-iteration MP decoding on each individual and transmit the mutated information to whole individuals. Those new individuals will be the descendants of this generation and form a new gene pool.

Next, GA-MP will use the fitness computing function to compute all the fitness values of descendants and then pick an elite descendant with the best fitness value of all. Since we observe that an MPA often fails to decode correctly owing to some critical bit node channel values, an MPA's performance can be significantly improved if we can find out those critical bit nodes and modify their channel values. Here, we propose to detect whether the channel value of a bit node should be updated or not, based on the belief reliability of each bit node of the elite descendant. This procedure, called channel value update, will be detailed in Section III-D.

After updating channel values, those values will be processed by L-iteration MP decoding (where L is a constant set by the user). Its result will be the ancestry of the next generation. Besides, we also check whether the L-iteration decoding result is a valid codeword or not. If yes, record it for a later decision of the final decoding result, as will be described in Section III-F. The above procedures are repeated until termination criterion (defined in Section III-E) is satisfied. In the next subsections, we will introduce the fitness computing function and key mechanisms of GA-MP algorithm in detail.

B. Fitness computing function

First, GA-MP defines a fitness computing function to evaluate every candidate in the gene pool and only keep the best one in the pool. According to our analysis, we consider the following two factors in the fitness computing function.

1) Syndrome weight (S): The summation of all entries in a syndrome vector is called syndrome weight (S). Obviously, a hard-decision sequence vector with smaller syndrome weight is closer to a valid codeword.

2) Euclidean distance (D): Since the solution of ML decoding is coddeword with the minimum Euclidean distances from the received channel values, among all possible valid codewords, Euclidean distance has been unanimously adopted as a key factor in fitness computing function.

The fitness computing function is defined as ($a \times D + b \times S$), where a and b are constants. The values of a and b depend on code length and code structure, which can be decided by simulations. Note that a should be smaller than b, because S is always larger than D. For example, from simulations, we find $\{a,b\} = \{1,8\}$ is a good choice for (155, 64) Tanner LDPC code. Obviously, an individual with lower fitness value has higher possibility to become the correct solution after MP decoding. The reason why the fitness computing function does not take belief reliability into account is because it will likely lead the evolution results to converge to the MP decoding result and make the whole evolution meaningless.

C. Belief reliability mutation

Here, a mechanism of belief reliability mutation is defined to perform the mutation process. First, define the direct connected bit node set (DCBS) as the set of bit bodes connected to the same check node. For example, the Tanner graph in Fig.1 has four DCBSs which are set1=$\{b_1,b_3,b_5,b_7\}$, set 2=$\{b_1,b_4,b_6,b_8\}$, set3=$\{b_2,b_3,b_6,b_7\}$, and set4=$\{b_2,b_4,b_5,b_8\}$. Note that each check node has its own DCBS.

Besides, we denote a DCBS as DCBS-1 and DCBS-0, if the DCBS's syndrome is 1 and 0, respectively. That is, DCBS-1 has odd incorrect bit nodes, and DCBS-0 has even or zero incorrect bit nodes. According to our analysis, when the ancestry's syndrome weight is relatively small, the number of incorrect bit nodes of DCBS-0 and DCBS-1 will have large variance. Hence, different belief reliability mutation mechanisms are required for ancestries with different syndrome weights. Here, we define a syndrome-reliability threshold T_{SW} for classifying high syndrome weight (HSW) and low syndrome weight (LSW) ancestries. If the syndrome weight of the ancestry is larger than this threshold, the ancestry will be classified as the HSW ancestry and vice versa.

*1) **Mechanism A:** For the HSW ancestry, the mutation mechanism has three operations as follows*

a) Decide suspicious bit nodes for mutation

Since a bit node with low MBR is less reliable, if the sign bits of belief reliability and the channel value are different, the channel value of this bit node should be likely corrected by the employed LDPC decoding algorithm. Therefore, we define a

bit node as a suspicious bit node when the following two conditions are satisfied:

- The MBRs of the bit nodes in DCBS-1 are less than a specified suspicious-bit threshold T_{SB}.
- The sign bits of the bit nodes in DCBS-1 are contrary to those of channel values.

b) Pick the selected suspicious bit nodes (SSBNs) and assign them new belief reliabilities

We randomly pick up some of DCBS-1s among all ancestry DCBS-1s , and select one of the suspicious bit nodes from each selected DCBS-1. After that, we flip the sign bits of the selected suspicious bit nodes (SSBN) and assign a constant value as their MBRs. This constant value is called HSW reliability level RL_H.

c) Limit the belief reliability of bit nodes

According to MP decoding algorithms, if a DCBS contains a bit node whose sign bit is incorrect, this DCBS's check node updating messages are also incorrect even if all other bit nodes have correct signs and large magnitudes. It may induce the check node to update a wrong-sign message with a very high magnitude, and cause burst errors in the ensuing iterations. Hence, we must limit the MBR value to avoid this situation, i.e., if the MBRs of bit nodes are larger than HSW reliability bound RB_H, we will set those bit nodes' MBRs to RB_H.

Besides, all the mutated individuals will be processed by 1-iteration MP decoding so that the mutated information can be spread to whole individuals. However, it is also observed that the belief reliabilities of the 1-iteration MP decoding results are always with high magnitudes, and may cause burst errors in the following iterations. Hence, all the bit nodes should be multiplied by a reliability scaling factor S_R (<1), except suspicious bit nodes (because suspicious bit nodes have been assigned an appropriate constant value in Mechanism A-b).

*2) **Mechanism B:** For the LSW ancestry, the mutation mechanism has four operations as below*

a) Decide suspicious bit nodes for mutation

Since the number of DCBS-1 of LSW ancestry is fewer than that of HSW ancestry, we can't use the same mechanism of HSW ancestry to identify suspicious bit nodes. Otherwise, the suspicious bit nodes will be too few to provide enough diversity for evolution. Hence, for LSW ancestry, we relax the decision condition by treating all the bit nodes of DCBS-1 as suspicious bit nodes.

b) Pick the selected suspicious bit nodes (SSBNs) and assign their belief reliabilities

This step is the same as Mechanism A-b, but the constant value is called LSW reliability level RL_L here.

c) Limit the belief reliability of bit nodes

The procedure does the same thing as what described in Mechanism A-c, except that in this case, the belief reliability bound is defined as LSW reliability bound RB_L.

d) to Decide whether to flip or not the signs of bit nodes with highmagnitudes, according to flipping probability P_{SF}

We observe that there always exists many incorrect bit nodes with high MBRs in LSW ancestry. Therefore, if those bit nodes with MBRs greater than LSW reliability bound are selected, their signs will be flipped with sign-flipping probability P_{SF} (properly set by users). With this step, we will have chances to correct those incorrect bit nodes, which cannot be identified by MP decoding algorithms.

After belief reliability mutation, those new individuals will be the descendants and form a new gene pool. We will then pick up a descendant with the best (i.e., the smallest) fitness value from the new gene pool as an elite descendant. Based on the bit node MBRs in the elite descendant, the received channel values will be updated as explained below.

D. *Channel value update*

As mentioned in Section III-A, MP decoding could perform well, if the channel values of some critical bit nodes had been updated. Besides, we also observe when the syndrome weight of an ancestry retains the same value in several continual generations, the evolution of GA-MP algorithm may be terminated. We call this situation as evolution stalemate. In order to break the stalemate, we add an additional procedure to detect if the GA-MP falls into evolution stalemate, by detecting if the latest N_{SD} generations' syndrome weights of ancestry retains the same number. If yes, GA-MP enters the stalemate breaking procedure.

In general situation (without entering stalemate breaker), we define a threshold T_C for channel value update and check the belief reliability of each bit node in the elite descendant. The bit nodes with MBRs larger than T_C are regarded as trustful bit nodes, otherwise untrustful. For those trustful bit nodes, the channel values are replaced by the elite descendant's belief reliabilities; and for those incredible bit nodes, the channel values are not altered. When GA-MP enter stalemate breaker, it does the same procedure as the general situation, except for untrustful bit nodes. The belief reliabilities of those untrustful bit nodes will be multiplied by a stalemate-breaking scaling factor S_B and added with the channel value as the updated bit nodes' channel values. This step will reduce the effect of those untrustful bit nodes during MP decoding for the next generation and provide sufficient diversity for future evolution.

E. *Termination criterion*

There are two termination criterions for in the proposed GA-MP algorithm. The first one is when the maximum generation number N_G has been reached, and the second one is when the maximum number of valid codewords N_{VC} has been collected.

F. *Detailed flows of the Proposed GA-MP Algorithm*

The detailed procedures of GA-MP algorithm are summarized as follows. Note that the Optional Step for the purpose of reducing the computing effort, at the cost of lowering decoding performance.

Step 0) Initialization: Initialize the values of N, L, N_G, T_{SB}, T_{SW}, RL_H, RB_H, S_R, RB_L, RL_L, P_{SF}, N_{SD}, T_C, S_B, and N_{LC}.

Step1) Generate the initial gene pool: Set i=1 and j=0. Perform the adopted MP decoding algorithm by L_0 iterations, where L_0 is the maximum iteration number. Each belief reliability set of iterative decoding is collected in the initial gene pool.

Optional Step) Optional check of GA-MP algorithm: For high-throughput requirement, if the N-iteration MP decoded solution is valid, go to Step 11.

Step 2) Select the first ancestry: Compute the fitness of all individuals in the initial gene pool and pick the one with the lowest fitness from the pool as the first ancestry.

Step 3) Record the syndrome weight of ancestry and set i=i+1:

Step 4) Identify HSW or LSW ancestry: Set k=1. If the syndrome weight of ancestry is larger than T_S, go to Step 4A-1; otherwise, go to Step 4B-1.

Step 4A-1) Indicate suspicious bit node for HSW ancestry: Mark those bit nodes of DCBS-1 as suspicious bit nodes, if their MBRs are less than T_{SB} and have opposite signs to their corresponding channel values.

Step 4A-2) Select SSBN for HSW ancestry: Randomly chose several DCBS-1's and select one suspicious bit node (i.e., SSBN) from each chosen DCBS-1 with probability of 0.5. Set belief reliability of each SSBN to RL_H.

Step 4A-3) Confine the belief reliabilities of the bit nodes for HSW ancestry: If the bit nodes' MBRs are larger than RB_H, they are set to RB_H. Otherwise, they are multiplied by S_R. Set k=k+1. If k>N, go to Step 5; otherwise, go to Step 4A-1.

Step 4B-1) Set all the bit nodes in each DCBS-1 as its suspicious bit node for LSW ancestry.

Step 4B-2) Select SSBN for LSW ancestry: Randomly select several SSBNs from suspicious bit nodes of each DCBS-1 with probability of 0.5. Set belief reliability of each SSBN to RL_L.

Step 4B-3) Confine the belief reliabilities of bit nodes for LSW ancestry: If the bit nodes' MBRs are larger than RB_L, those MBRs are set to RB_L; otherwise, they are multiplied by S_R.

Step 4B-4) Flip the signs of bit nodes for LSW ancestry: Randomly flip the sign of the confined bit nodes with probability of P_{SF}. Set k=k+1, if k>N, go to Step 5; otherwise, go to Step 4B-1.

Step 5) Build a new gene pool: Process the N results of belief reliability mutation using 1-iteration MP decoding to form a new gene pool.

Step 6) Select elite descendant: Select a descendant with the smallest fitness value as an elite descendant of the gene pool.

Step 7) Update channel values: If the elite descendant's bit-node MBRs > T_C, update those bit nodes' channel values with the belief reliabilities of elite descendant.

Step 8) Detect the evolution stalemate: In case of i>N_{SD}, check whether the previous N_{SD} ancestry syndrome weights are identical. If not, go to Step 10; otherwise, go to Step 9.

Step 9) Update channel values with stalemate breaker: In stalemate breaker, the belief reliabilities of bit nodes un-updated in Step 7are additionally multiplied by S_B and added with their channel values as the updated channel values.

Step 10) Generate the next-generation ancestry: Perform L-iteration MP decoding on the updated channel values to generate the next-generation ancestry. Besides, the hard-decision codeword of L-iteration MP decoding results will be used to detect whether the codeword is valid or not. If yes, record this result in a defined decoding-result set, and set j=j+1.

Step 11) Termination criterion: If i<N_G or j<N_{VC}, go to Step 3.

Step 12) Decide the decoding result: Among all the collected valid codewords in the decoding-result set, the codeword with the minimum Euclidean distance from its channel values will be chosen as the final solution of GA-MP algorithm. If there is no any valid codeword collected in the decoding-result set, assign the hard-decision codeword of the initial-generation ancestry as the final solution.

Step 13) End of GA-MP algorithm: Output the decoded solution.

Due to page limitation, the decision strategies for those mentioned parameters are omitted here. In the next section, we will show the simulation results of the proposed algorithm.

IV. Simulation Results

For 802.16e (576,288) LDPC code, we set RL_H=3, T_{SB} =2, T_{SW} =10, S_R=0.5, P_{SF}=0.1, N_{SD}=3, S_B=0.5, TCU=RB_H=8.8, and RL_L=RB_L=4.2 for GA-MP algorithm. The decoding performance of GA-MP algorithm is shown in Fig. 2, where GA-SPA is the proposed algorithm, where SPA is the chosen MA. In the figure, we find the decoding performance increases as the maximum generation number N_G increases. It shows that the decoding performance of GA-MP algorithm is not confined by performance bound. We conjecture that it maybe is because N_G=2000 is not large enough to show the performance bound. On the other hand, when N_G is larger than 2000, the simulation time of our program will be too long to obtain simulation results. Nevertheless, the proposed algorithm is well applied to other code lengths..

In order to compare GA-MP algorithm with other algorithms, we use the maximum iterations as a measure of computation complexity. The maximum iteration number L_{total} of GA-MP for each received codeword can be shown to be:

$$L_{total}=L_0+(N+L)*N_G \quad (1)$$

Although generally genetic algorithms assume the same gene pool size for each generation, i.e., L_0=N, L_0 and N still can be set to different values for the sake of reducing iteration number.

Fig. 3 shows the performance comparison with existing MP algorithms, ML algorithm, and a genetic-assisted decoding algorithm [10] with the same maximum iteration number for a (155, 64) Tanner LDPC code, where GA-NMSA and GA-MSA

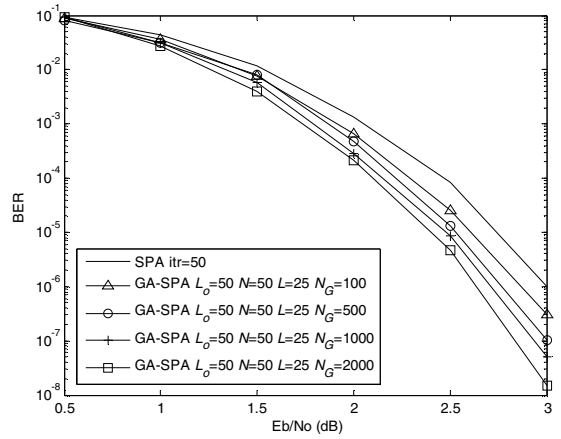

Figure 2. The bit-error rate (BER) performance of 802.16e (576,288) LDPC code with various maximum generation number of GA-SPA.

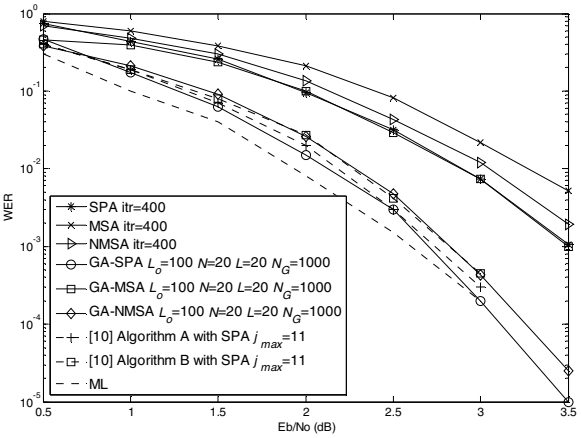

Figure 3. The word error rate (WER) performance of (155, 64) Tanner LDPC code.

means the proposed algorithm combining with NMSA and MSA, respectively. Their maximum iteration number for each received codeword is 100+ $(2^{12}$-2)*10=41040 based on the equation in [10]. For convenience, we set L_0=100, N=20, L=20 and N_G=1000 for the proposed GA-MP algorithm. Thus, the maximum iteration number for each received codeword is 100 + 1000 * (20+20) =40100 due to (1), which is about the same as that in [10]. Besides, for this code, we set RL_H=3, T_{SB} =2, T_{SW} =10, S_R=0.5, P_{SF}=0.1, N_{SD}=3, S_B=0.5, TCU=RB_H=7 and RL_L= RB_L=3.5 for GA-MP algorithm.

As the figure shows, GA-SPA algorithm has the best decoding performance and is very close to the performance of ML algorithm. The algorithms of [10] also perform well. However, their memory requirements will drastically grow with j_{max}. According to [10], since it needs to store at least $2^{j_{\max}}$ soft channel values, they are costly for the decoding of long LDPC codes with acceptable hardware resource. On the other hand, the proposed algorithm totally only needs to store three-folds soft channel values in the memory. Consequently, the proposed algorithm is much more competitive in realization than the algorithms of [10].

Due to page limitation, we omit the performance comparisons for various other LDPC codes. However, their behaviors are similar to the case of (155, 64) Tanner LDPC code, that is, the decoding performance improvements of the proposed algorithm over the existing algorithms is more significant for shorter codes than longer codes. Since the algorithm in [9] is much more complicated than [10], it is also omitted here.

V. CONCLUSION

This paper proposes a novel GA-MP algorithm which combines the concept of GA with MP algorithm. The decoding performance of GA-MP algorithm is superior to original MP algorithms by about 0.5 dB and is very close to the performance of ML decoding for (155, 64) Tanner LDPC code. Besides, the proposed GA-MP algorithm also works well with other lower-performance MP algorithms, such as MSA and NMSA. Compared with [10], the proposed algorithm has better decoding performance but with much less hardware cost.

ACKNOWLEDGMENT

This work was supported in part by the NSC, Taiwan, under grants NSC 100-2219-E-009 -016 and NSC 100-2220-E-009 -026.

REFERENCES

[1] R. G. Gallager, *Low-density parity-check codes*, Cambridge, MA: MIT Press, 1963.

[2] *Air Inference for fixed and mobile broadband wireless access systems*, IEEE Std.802.16e, 2005.

[3] *Information technology-telecommunications and information exchange between system --Local and metropolitan area networks*, IEEE Std. 802.11n, 2007.

[4] *Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, IEEE Std. 802.3an, 2006.

[5] D. J. C. Mackay, "Good error-correcting codes based on very sparse matrices,"IEEE Trans. Inform. Theory, vol. 45, pp. 399-431, Mar. 1999.

[6] C. Stark, S. Jonathan and W. Yige. "ML decoding via mixed-integer adaptive linear programming," IEEE Trans. Commun., pp. 1308-1317, Jul. 2007.

[7] X. Y. Hu, E. Eleftheriou, D. M. Arnold, and A. Dholakia, "Efficient implementation of the sum-product algorithm for decoding LDPC codes," IEEE GLOBECOM'01, vol. 02, pp. 1036-1036E, Nov. 2001.

[8] J.Chen and M. P.C. Fossorier, "Near optimum universal belief propagation based decoding of low-density parity check codes," IEEE Trans. on Commun., vol. 50, pp. 583-587, no.3 Mar. 2002.

[9] Z. Deng,L. Xingcheng and T.Man, "Modified BP decoding algorithms combined with GA for low-density parity check codes," International Conference on Computational Intelligence and Security, pp. 371-375, Dec. 2008.

[10] N. Varnica, P. C. Marc, and K. Aleksandar, "Augmented belief propagation decoding of low-density parity check codes," IEEE Trans. Commun., pp. 1308–1317, Jul. 2007.

[11] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence,* Ann Arbor, MI: University of Michigan Press, 1975.