International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 / https://doi.org/10.14445/22315381/IJETT-V70I12P227
Volume 70 Issue 12, 289-295, December 2022

Original Article

A New Fast Iterative Decoder of Product Codes Based on Hash and Syndromes and Optimized by Genetic Algorithms

Hamza Faham[1], Seddiq El Kasmi Alaoui[2], Mohammed El Assad[3], Saïd Nouh[4], Idriss Chana[5], Mohamed Azzouazi[6]

[1,3,4,6] *LTIM Lab, Ben M'sick Sciences Faculty, Hassan II University of Casablanca, Casablanca, Morocco*
[2] *LIS Lab, Aïn Chock Sciences Faculty, Hassan II University of Casablanca, Casablanca, Morocco*
[5] *ISIC Research Team of ESTM, LMMI Lab of ENSAM, University of Moulay Ismail, Meknes, Morocco*

[1]*Corresponding Author : faham.hamza@gmail.com*

Received: 18 July 2022 Revised: 14 October 2022 Accepted: 18 December 2022 Published: 24 December 2022

Abstract *- Iterative decoding techniques have become very interesting, motivated by the encouraging results of the turbo codes. The Soft Decision Decoder based on Hash Techniques (SDHT) is a recent decoder of best performances and low temporal complexity. It is this second characteristic of the speed of SDHT which prompted us to use it here as a component decoder. In this paper, we adopt then SDHT as a soft input hard output (SIHO) decoding algorithm, about implement an iterative decoder for product codes at the base of Quadratic Residue (QR) and Bose Ray-Chaudhuri and Hocquenghem (BCH) codes. To compute the SDHT soft output, we exploit extrinsic information according to Soleymani et al. The iterative decoding is achieved via Pyndiah's connection layout. The major aim of using the SDHT decoder is to benefit from its low computational complexity. We have also used a genetic algorithm to optimize the confidence value Φ that yields good performance in terms of Bit-Error-Rate. Simulation results and the study of complexities show that the proposed iterative decoder exceeds some competitors in terms of performance and complexity.*

Keywords *- Error correcting codes, Genetic algorithms, Hash techniques, Product codes, Iterative decoder.*

1. Introduction

The information theory has introduced the essential components of any digital communication system, in which a discrete source of information produces information. The transmitter envisages communication with the receiver via a transmission medium in this model. This modelling is schematized in Figure 1.
By analysing this model, we can distinguish the following parts: source encoder/decoder, channel encoder/decoder, modulator/demodulator and the transmission medium. In this work, we are interested in the channel encoder/decoder part [1-4].

Berrou et al. [5] have created the turbo codes by concatenating convolutional codes and using the Bahl Cocke Jelinek Raviv (BCJR) decoder and the Viterbi soft output. Shortly after, Pyndiah et al. [6] proposed extending the iterative decoding for bloc codes employing concatenation of BCH codes, Chase II decoder and a soft output computed as described in [6]. Until our days, iterative decoding has captured the attention of diverse works [7-12]. Turbo codes are high-performance error-correcting codes used primarily in applications requiring reliable information transfer.

In this paper, we have designed a new iterative decoder called I-SDHT (Iterative-SDHT) while respecting three primary conditions: Firstly, we have constructed product codes. Secondly, we have selected an elementary decoding algorithm treated through a soft output. Finally, we have adopted a decoding scheme where the soft output is transformed into extrinsic information exchanged between the component decoders through an iterative process. Our iterative decoding adopts the SDHT decoder as a SIHO decoding algorithm with the intention of exploiting its low complexity.

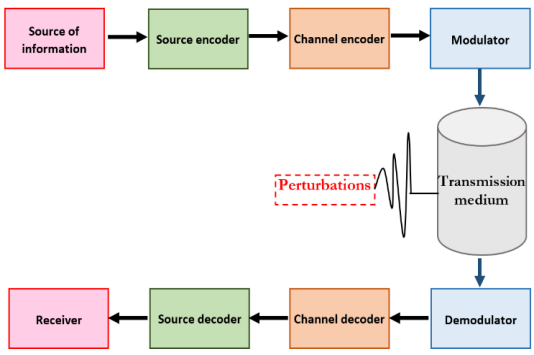

Fig. 1 Basic modeling of a digital communication system

To determine the SDHT soft output, we have used Soleymani irrelevant information [13]. Moreover, we have followed Pyndiah's connection layout omitting the well-known coefficients α and β. The product codes used in this paper are at the base of the cyclic QR and BCH codes.
This paper's remainder is arranged as follows: Section 2 examines the experimental setup. Section 3 shows the simulation results and comparisons. In the end, section 4 summarizes our contribution.

2. Method

2.1. Product Codes

We note the linear code by C (n, k, d). k is the code dimension, n is the length, and d is its Hamming minimum distance.

To define a product code, let us consider two codes, C_1 (n_1, k_1, d_1) and C_2(n_2, k_2, d_2). The product code C_p=C_1×C_2 is seen as a matrix of n_1 rows and n_2 columns, where the information constitutes a sub-matrix M of k_1 lines and k_2 columns, each of the k_1 lines is coded by C_1, and each of the n_1 columns is coded by C_2. The resulting metrics of C_p are $n_p = n_1 \times n_2$, $k_p = k_1 \times k_2$, and $d_p = d_1 \times d_2$.

The important profit of using product codes is their gain of minimum distance. In contrast, their main inconvenience is the code rate waste. In our work, C_1 and C_2 are identic. We have used product codes based on the cyclic QR and BCH codes.

2.2. Our Proposed Turbo Decoder

To design the I-SDHT decoder, we have followed three primary steps: Firstly, we have composed product codes. Secondly, we have selected an elementary decoder treated by a soft output. Thirdly, we have established a decoding scheme where the soft output is converted to extrinsic information exchanged between the component decoders via an iterative process. We have adopted the SDHT decoder [22] as a SIHO decoding algorithm to exploit its low complexity.

Table 1 lists the complexities of the following soft decoding algorithms: SDHT, Chana SIHO Decoder [15], Chase-HSDec[16], OSD-m (for an integer m) [23], Maini[18], DDGA[24] and Aut-DAG[20].

Table 1. The complexity of SDHT and some challengers

Algorithm	Complexity
SDHT	$O(\log(n).\log(n-k)+3n-2k+2nC)$
Chana SIHO Decoder	$O(2^t k.\log(n)((n+1).\log(n-k)))$
Chase-HSDec	$O(2^t(\log(n).\log(n-k)+3n-2k))$
OSD-m	$O(n^{m+1})$
Maini	$O(N_i.N_g(k.n+\log(N_i)))$
DDGA	$O(N_i.N_g(k.(n-k)+\log(N_i)))$
Aut-DAG	$O(N_i.N_g(\log(n).\log(n-k)))$

This table shows that the SDHT has a reduced complexity compared to its challengers.

C is the maximum length of hash table TH rows [22], t is the error correcting capability, m is the order of OSD decoding algorithm, N_i is the population size, and N_g is the number of generations.

Our iterative decoder employs Soleymani's irrelevant information to compute the SDHT soft output [13]. In addition, it adopts the iterative decoding scheme initiated by Pyndiah [6] with a little alteration for the connection between iterations by omitting the terms α and β. The diagram in fig.2 describes the process of our iterative decoding.

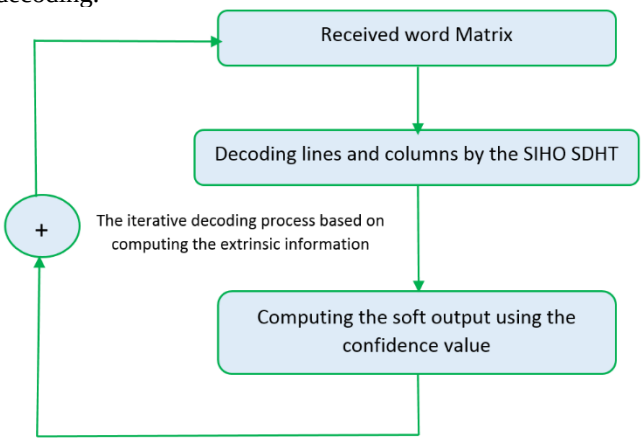

Fig. 2 The process of the proposed iterative decoder

The decoder of Chana et al. [15] uses a parameter called confidence value Φ to compute both the soft output and the extrinsic information. This confidence value is a function of the destructive Euclidean distance between the received word and the decided codeword:

$$Dist_{dest} = \sum_{j \in DES} (r_j - d_j)^2 \quad (1)$$

where $$DES = \{j | (r_j - d_j).d_j < 0\}$$

Chana et al. have used a statistical approach to find Φ. In order to optimize it, we propose in this work to benefit from the power of Genetic Algorithms to find a vector Φ that yields good performances of the proposed decoder.
The evolutionary approach of genetic algorithms is inspired by scientific methods used in Darwin's work on natural selection.

The proposed genetic algorithm consists of many steps repeated several times. The first step establishes a first random set of N possible solutions: each solution is an individual of DistMax genes (DistMax is an integer representing the maximum destructive Euclidean distance for which the confidence value is considered null).

For example, the following individual:

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*DistMax=10*
0.99	*0.99*	*0.99*	*0.87*	*0.83*	*0.79*	*0.7*	*0.6*	*0.4*	*0.0*

When the destructive Euclidean distance between the received word and the decided codeword is between 0 and 3, the confidence value is 0.99.

When the destructive Euclidean distance is between 5 and 6, the confidence value is 0.79.

In the second step, each individual is then evaluated by computing its fitness, which is equal to the Bit Error Rate obtained by this individual as the Φ parameter.

In the third step, the selection operator, simple crossover in one point and real mutation are applied. To mutate a gene (real value between 0 and 1), a random value is added or subtracted to this gene. In order to have individuals in decreased order, we have proposed to sort it. Each individual's genes are corrected to ensure they are between 0 and 1.

In each iteration, the population is sorted in increasing order of fitness. The considered optimal solution is then the first individual in the last generation.

Tables 2, 3,4 and 5 give the Φ parameter obtained by the proposed genetic algorithm:

Table 2. Confidence values for QR(23,12)[2]

$Dist_{dest}$	**Confidence value**
$Dist_{dest}$ <1	0.904532
1≤ $Dist_{dest}$ <2	0.888007
2≤ $Dist_{dest}$ <3	0.868403
3≤ $Dist_{dest}$ <4	0.845267
4≤ $Dist_{dest}$ <5	0.817719
5≤ $Dist_{dest}$ <6	0.780205
6≤ $Dist_{dest}$ <7	0.745906
7≤ $Dist_{dest}$ <8	0.690201
8≤ $Dist_{dest}$ <9	0.675685
9≤ $Dist_{dest}$ <10	0.672749
10≤ $Dist_{dest}$ <11	0.664256
11≤ $Dist_{dest}$ <12	0.655763
12≤ $Dist_{dest}$ <13	0.609927
13≤ $Dist_{dest}$ <14	0.561200
14≤ $Dist_{dest}$ <15	0.518566
15≤ $Dist_{dest}$ <16	0.486234
16≤ $Dist_{dest}$ <17	0.460333
17≤ $Dist_{dest}$ <18	0.420058
18≤ $Dist_{dest}$ <19	0.383881
19≤ $Dist_{dest}$ <20	0.351512
20≤ $Dist_{dest}$ <21	0.321926
21≤ $Dist_{dest}$ <22	0.293731
22≤ $Dist_{dest}$ <23	0.265536
$Dist_{dest}$ ≥23	0

Table 3. Confidence values for QR(31,16)[2]

$Dist_{dest}$	**Confidence value**
$Dist_{dest}$ <1	0.978637
1≤ $Dist_{dest}$ <2	0.969344
2≤ $Dist_{dest}$ <3	0.953093
3≤ $Dist_{dest}$ <4	0.938990
4≤ $Dist_{dest}$ <5	0.931186
5≤ $Dist_{dest}$ <6	0.926570
6≤ $Dist_{dest}$ <7	0.921953
7≤ $Dist_{dest}$ <8	0.881310
8≤ $Dist_{dest}$ <9	0.834348
9≤ $Dist_{dest}$ <10	0.784892
10≤ $Dist_{dest}$ <11	0.738190
11≤ $Dist_{dest}$ <12	0.702914
12≤ $Dist_{dest}$ <13	0.670982
13≤ $Dist_{dest}$ <14	0.640072
14≤ $Dist_{dest}$ <15	0.614168
15≤ $Dist_{dest}$ <16	0.595245
16≤ $Dist_{dest}$ <17	0.587986
17≤ $Dist_{dest}$ <18	0.582184
18≤ $Dist_{dest}$ <19	0.580844
19≤ $Dist_{dest}$ <20	0.573702
20≤ $Dist_{dest}$ <21	0.572114
21≤ $Dist_{dest}$ <22	0.552166
22≤ $Dist_{dest}$ <23	0.506442
23≤ $Dist_{dest}$ <24	0.475922
24≤ $Dist_{dest}$ <25	0.444044
25≤ $Dist_{dest}$ <26	0.409821
26≤ $Dist_{dest}$ <27	0.379812
27≤ $Dist_{dest}$ <28	0.321021
28≤ $Dist_{dest}$ <29	0.269545
29≤ $Dist_{dest}$ <30	0.221728
30≤ $Dist_{dest}$ <31	0.173910
$Dist_{dest}$ ≥31	0

Table 4. Confidence values for BCH(31,16)[2]

$Dist_{dest}$	**Confidence value**
$Dist_{dest}$ <1	0.926908
1≤ $Dist_{dest}$ <2	0.897159
2≤ $Dist_{dest}$ <3	0.869659
3≤ $Dist_{dest}$ <4	0.848996
4≤ $Dist_{dest}$ <5	0.832999
5≤ $Dist_{dest}$ <6	0.819357
6≤ $Dist_{dest}$ <7	0.802336
7≤ $Dist_{dest}$ <8	0.780271
8≤ $Dist_{dest}$ <9	0.756673
9≤ $Dist_{dest}$ <10	0.733994
10≤ $Dist_{dest}$ <11	0.708230
11≤ $Dist_{dest}$ <12	0.680154
12≤ $Dist_{dest}$ <13	0.652575
13≤ $Dist_{dest}$ <14	0.627747
14≤ $Dist_{dest}$ <15	0.606533
15≤ $Dist_{dest}$ <16	0.588735
16≤ $Dist_{dest}$ <17	0.573949
17≤ $Dist_{dest}$ <18	0.561844
18≤ $Dist_{dest}$ <19	0.552095

$19 \leq Dist_{dest} < 20$	0.543382
$20 \leq Dist_{dest} < 21$	0.528155
$21 \leq Dist_{dest} < 22$	0.514969
$22 \leq Dist_{dest} < 23$	0.505452
$23 \leq Dist_{dest} < 24$	0.490659
$24 \leq Dist_{dest} < 25$	0.468755
$25 \leq Dist_{dest} < 26$	0.440767
$26 \leq Dist_{dest} < 27$	0.381809
$27 \leq Dist_{dest} < 28$	0.319394
$28 \leq Dist_{dest} < 29$	0.255251
$29 \leq Dist_{dest} < 30$	0.191107
$Dist_{dest} \geq 30$	0

Table 5. Confidence values for BCH(31,21)²

$Dist_{dest}$	**Confidence**
$Dist_{dest} < 1$	0.914609
$1 \leq Dist_{dest} < 2$	0.820214
$2 \leq Dist_{dest} < 3$	0.755324
$3 \leq Dist_{dest} < 4$	0.692830
$4 \leq Dist_{dest} < 5$	0.633588
$5 \leq Dist_{dest} < 6$	0.566788
$6 \leq Dist_{dest} < 7$	0.519150
$7 \leq Dist_{dest} < 8$	0.488785
$8 \leq Dist_{dest} < 9$	0.486969
$9 \leq Dist_{dest} < 10$	0.477477
$10 \leq Dist_{dest}$	0.466169
$11 \leq Dist_{dest}$	0.423018
$12 \leq Dist_{dest}$	0.370255
$13 \leq Dist_{dest}$	0.325584
$14 \leq Dist_{dest}$	0.240772
$15 \leq Dist_{dest}$	0.135889
$16 \leq Dist_{dest}$	0.031007
$Dist_{dest} \geq 17$	0

3. Results and discussion

3.1. Performances of our Proposed Genetic Algorithm

To judge the efficiency of our suggested genetic algorithm that we have applied to compute the Φ parameter,

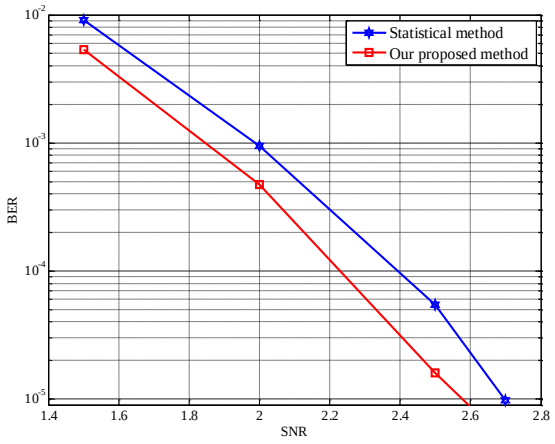

Fig. 3 Comparison between the statistical and the proposed method for QR (23, 12)²

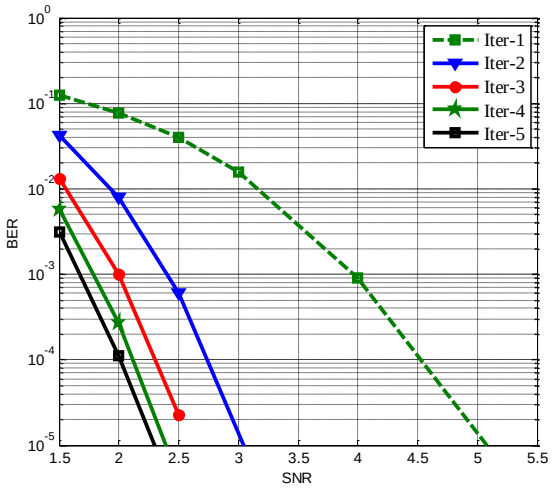

Fig. 4 Iterative effect for QR (31, 16, 7)²

Figure 3 compares the statistical method [15] and our proposed method for QR(23,12)² code after 5 iterations. It proves that our genetic algorithm yields interesting results and offers better performances.

3.2. Iterative Effect

All simulations in this paper are made by computing the bit error rate (BER) for several signal-to-noise ratios (SNR) in dB.

The propagation channel used is the additive white Gaussian noise (AWGN) channel with variance $\sigma^2 = \frac{N_0}{2E_s}$ where E_s is the energy per symbol and N_0 is the noise spectral density. The modulation employed is BPSK. In addition, we transmit at least 105 blocks in every simulation and stop when we get 500 residual errors.

An engine has a turbo effect; the more it runs, the better its performance. Consequently, the iterative effect has been introduced in technology, especially in communication, to reduce errors. Figure 4 reveals the performance of QR (31, 16, 7)². Thanks to this figure, we notice a remarkable coding gain moving from iteration 1 to 2.

Nevertheless, moving beyond iteration 4, the coding gain is slight. After that, it is inconsiderable. As a result, we conclude that the iterative effect is reached.

3.3. Performances study of our Iterative Decoder

To examine more our iterative decoder, we applied it to several product codes with different code dimensions.

Figure 5 shows the performance of our decoding algorithm for BCH (31, 16, 7)² and BCH (31, 21, 5)² codes. We conclude that our decoding process presents good results in terms of coding gain.

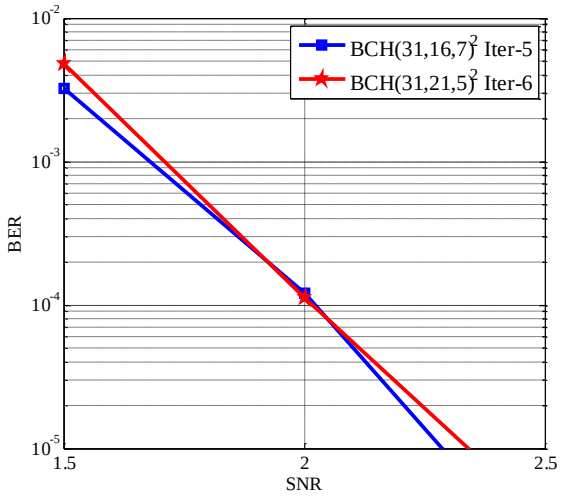

Fig. 5 Performances of our iterative decoder for some Product BCH codes

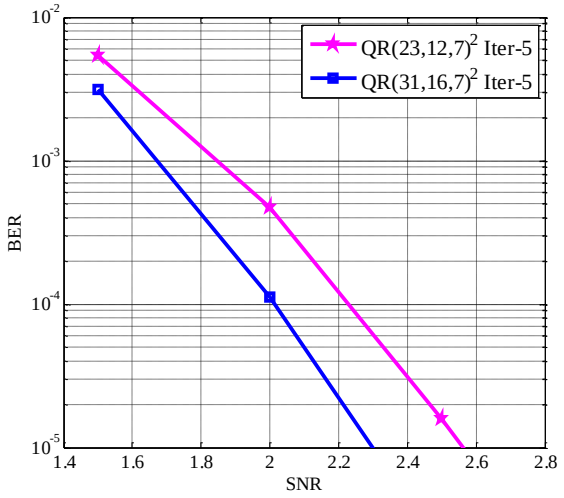

Fig. 6 Performances of our decoder for some Product QR codes

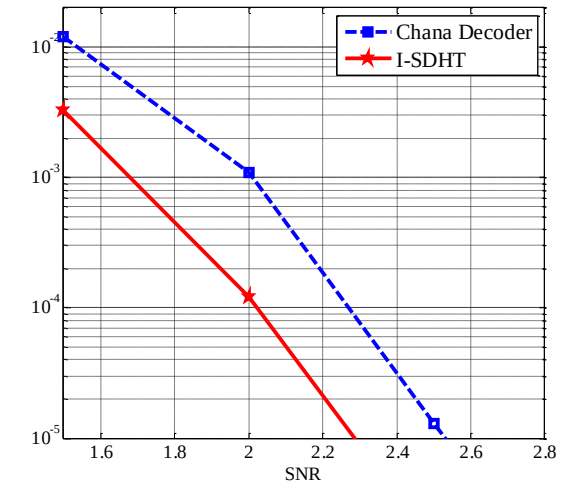

Fig. 7 Comparison of our decoder and Chana decoder for BCH (31, 16, 7)^2

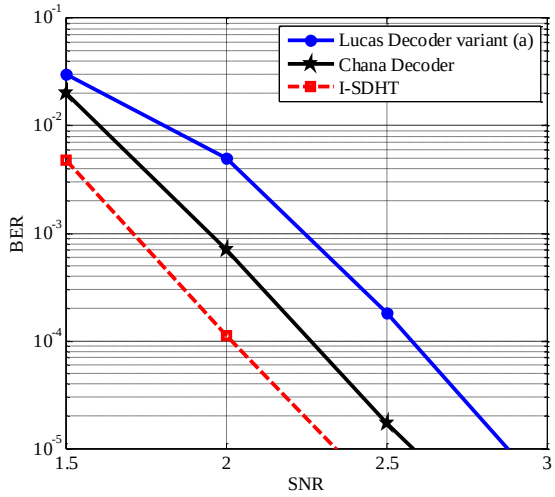

Fig. 8 Comparison of our decoder, Lucas decoder and Chana decoder for BCH (31, 21, 5)^2

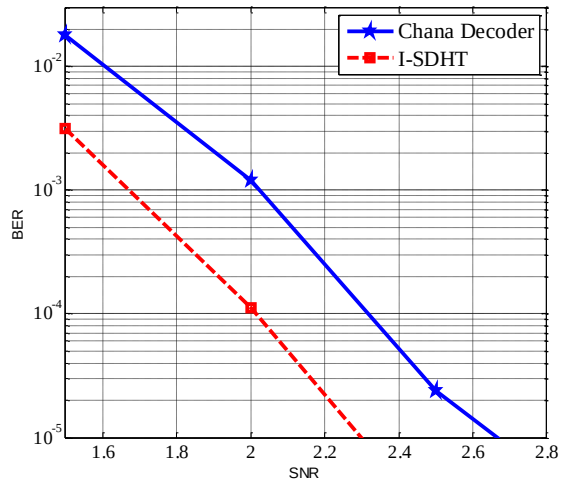

Fig. 9 Comparison of our decoder and Chana decoder for QR (31, 16, 7)^2

Figure 6 presents the performance of our iterative decoder for QR (23, 12, 7)^2 and QR (31, 16, 7)^2 codes.

We conclude again that our algorithm has interesting coding gains.

3.4. Comparison with other Decoders

In this subsection, we will compare the performance of our suggested iterative decoder with other decoders for some product codes.

Figure 7 shows a comparison between our decoding algorithm and the Chana decoder for BCH (31, 16, 7)^2.

The figure reveals that our decoding process exceeds its competitor for several SNRs.

Figure 8 introduces a comparison between our decoding algorithm and Lucas and Chana algorithm for BCH (31, 21, 5)^2 code.

This figure proves that our decoding scheme outperforms Lucas's decoder with approximately 0.6 dB at BER 10^{-5}.

Figure 9 shows a comparison between our decoder and Chana decoding scheme for QR (31, 16, 7)^2 code.

This figure reveals that our decoding algorithm outperforms the Chana algorithm with approximately 0.4 dB at BER 10^{-5}.

4. Conclusion

This paper introduces a speedy iterative decoder by adopting the SDHT algorithm for BCH and QR product codes. We have also employed a genetic algorithm to enhance the confidence value Φ that yields better performances in terms of Bit-Error-Rate. The results are very satisfying in terms of BER. The iterative effect is achieved by increasing the number of iterations. The Computer simulations show that the performances produced by our method are relatively better than some competitors for BCH and QR codes. The extension of this study can be done by applying this method to other forms of concatenations based on linear codes.

References

[1] Hamza Faham et al., "High Performance Decoding by Combination of the Hartmann Rudolph Decoder and Soft Decision Decoding by Hash Techniques," *Lecture Notes in Networks and Systems,* vol. 211, pp. 781–790, 2021. *Crossref,* https://doi.org/10.1007/978-3-030-73882-2_71

[2] Hamza Faham et al., "An Efficient Combination between Berlekamp-Massey and Hartmann Rudolph Algorithms to Decode BCH Codes," *Periodicals of Engineering and Natural Sciences,* vol. 6, no. 2, 2018. *Crossref,* http://dx.doi.org/10.21533/pen.v6i2.540

[3] Hamza Faham et al., "New Way to Enumerate Large Quadratic Residue Codes Based on Hash and Automorphism Group," *Lecture Notes in Networks and Systems,* vol. 357, pp. 545–556, 2022. *Crossref,* https://doi.org/10.1007/978-3-030-91738-8_50

[4] Hamza Faham et al., "High Speed Decoding by Collaboration between the Hartmann Rudolph and Information Set Decoding Algorithms," *Journal of Theoretical and Applied Information Technology,* vol. 100, no. 17, pp. 5377–5385, 2022.

[5] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error Correcting Coding and Decoding: Turbo Codes," *IEEE International Conference on Communications, ICC'93*, vol. 2, pp. 1064-1070, 1993. *Crossref,* http://dx.doi.org/10.1109/ICC.1993.397441

[6] R.M. Pyndiah, "Near Optimum Decoding of Product Codes: Block Turbo Codes," *IEEE Transactions on Communications,* vol. 46, no. 8, pp. 1003–1010, 1998. *Crossref,* http://dx.doi.org/10.1109/26.705396

[7] Yihan Jiang et al., "Turbo Autoencoder: Deep Learning Based Channel Codes for Point-To-Point Communication Channels," *Advances in Neural Information Processing Systems*, vol. 32, pp. 2758-2768, 2019.

[8] Natasa Zivic, and Obaid Ur-Rehman, "Iterative Self-Correction for Secured Images Using Turbo Codes and Soft Input Decryption," *Journal of Intelligent & Fuzzy Systems*, vol. 38, no. 2, pp. 1841-1854, 2020. *Crossref,* http://dx.doi.org/10.3233/JIFS-190234

[9] Lucian Trifina et al., "Upper Bounds on the Minimum Distance for Turbo Codes Using CPP Interleavers," *Telecommunication Systems*, vol. 76, no. 3, pp. 423-447, 2021. *Crossref,* https://doi.org/10.1007/s11235-020-00723-4

[10] Xiaowei Wu, Min Qiu, and Jinhong Yuan, "Partially Information Coupled Duo-Binary Turbo Codes," *In 2020 IEEE International Symposium on Information Theory (ISIT), IEEE,* pp. 461-466, 2020. *Crossref,* https://doi.org/10.1109/ISIT44484.2020.9174156

[11] Victoria Herranz, Diego Napp, and Carmen Perea, "1/n Turbo Codes from Linear System Point of View," *Magazine of the Royal Academy of Exact, Physical and Natural Sciences. Series A. Mathematics,* vol. 114, no. 3, pp. 1-16, 2020. *Crossref,* https://doi.org/10.1007/s13398-020-00850-2

[12] Yihan Jiang et al., "Feedback Turbo Autoencoder," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE,* pp. 8559-8563, 2020. *Crossref,* https://doi.org/10.1109/ICASSP40776.2020.9053254

[13] N. Le, A.R. Soleymani, and Y.R. Shayan, "Distance-Based-Decoding of Block Turbo Codes," *IEEE Communications Letters*, vol. 9, no. 11, pp. 1005-1008, 2005. *Crossref,* https://doi.org/10.1109/LCOMM.2005.11014

[14] M E Purushoththaman, and Bhavani Buthtkuri, "Effective Multiple Verification Process Ensuring Security and Data Accuracy in Cloud Environment Storage," *SSRG International Journal of Computer Science and Engineering,* vol. 6, no. 7, pp. 1-4, 2019. *Crossref,* https://doi.org/10.14445/23488387/IJCSE-V6I7P101

[15] Idriss Chana, Hamid Allouch, and Mostafa Belkasmi, "New Turbo Decoding of Product Codes Based on Cyclic Codes," *Journal of Telecommunications,* vol. 11, no. 2, pp. 39-48, 2011.

[16] M.S. El Kasmi Alaoui, S. Nouh, and A. Marzak, "A Low Complexity Soft Decision Decoder for Linear Block Codes," *Proceedings of the First International Conference on Intelligent Computing in Data Sciences,* vol. 127, pp. 287-292, 2018. *Crossref,* https://doi.org/10.1016/j.procs.2018.01.124

[17] Dasari Ramanna, and Ganesan V, "Design and Analysis of Improved Raptor Encoder-based Hybrid Recursive Systematic Convolutional Encoding Technique for 6G Networks," *SSRG International Journal of Electrical and Electronics Engineering*, vol. 9, no. 11, pp. 11-16, 2022. *Crossref*, https://doi.org/10.14445/23488379/IJEEE-V9I11P102

[18] H. Maini et al., "Soft Decision Decoding of Linear Block Codes Using Genetic Algorithms," *Proceedings of IEEE International Symposium on Information Theory*, p. 397, 1994. *Crossref*, https://doi.org/10.1109/ISIT.1994.394622

[19] Latifa Mostari, and Abdelmalik Taleb-Ahmed, "Serial Concatenation of Binary LDPC Codes with Iterative Decoding," *International Journal of Recent Engineering Science (IJRES)*, vol. 6, no. 6, pp. 7-10, 2019. *Crossref*, https://doi.org/10.14445/23497157/IJRES-V6I6P102

[20] Faissal El Bouanani et al., "Comparison of Chase Decoders, OSD and those based on Genetic Algorithms," *Proceedings of Colloque GRETSI*, pp. 1153-1156, 2007.

[21] Korada Kishore Kumar, and Konni Srinivasa Rao, "An Efficient users Authentication and Secure Data Transmission of Cluster based Wireless Sensor Network," *SSRG International Journal of Computer Science and Engineering*, vol. 5, no. 1, pp. 1-5, 2018. *Crossref*, https://doi.org/10.14445/23488387/IJCSE-V5I1P101

[22] Moulay Seddiq El Kasmi Alaoui, Said Nouh, and Abdelaziz Marzak, "High Speed Soft Decision Decoding of Linear Codes Based on Hash and Syndrome Decoding," *International Journal of Intelligent Engineering and Systems*, vol. 12, no. 1, pp. 94-103, 2019. *Crossref*, https://doi.org/10.22266/ijies2019.0228.10

[23] M.P.C. Fossorier, and S. Lin, "Soft Decision Decoding of Linear Block Codes Based on Ordered Statistics," *IEEE Transaction on Information Theory*, vol. 41, no. 5, pp. 1379–1396, 1995. *Crossref*, https://doi.org/10.1109/18.412683

[24] Ahmed Azouaoui, Mostafa Belkasmi, and Abderrazak Farchane, "Efficient Dual Domain Decoding of Linear Block Codes Using Genetic Algorithms," *Journal of Electrical and Computer Engineering*, vol. 2012, 2012. *Crossref*, https://doi.org/10.1155/2012/503834