DESIGN AND GENERATION OF ERROR CORRECTION CODES FOR COMMUNICATION NETWORKS USING PARTICLE SWARM OPTIMIZATION (PSO)

Ankit Agarwal[1], K. Srinivas[2]
[1]Assistant Professor, Department of Computer Science & Engineering, Quantum School of Technology Roorkee
[2]Assistant Professor, Department of Electrical Engineering, Dayalbagh Educational Institute Agra

Abstract

In communication system, there is always a possibility that information can be corrupted during transmission. This corrupted information then either has to be retransmitted or the errors have to be detected and corrected. The retransmission for a corrupted message may be costly. Thus, the message is to be corrected by the receiver. Therefore, Error Correction Codes (ECC) is of immense importance in digital data communication networks. The problem of finding an error correcting code of n bits and M codewords that corrects a given maximum number of errors is NP-hard. For this reason the problem has to be solved by some Metaheuristic search technique such as PSO. In this paper, a special case of ECC, Constant Weight Error Correction codes have been generated using PSO which have an additional constraint of constant weight i.e. a constant number of 1's in every codeword. The algorithm presented here generate codes with the maximum number of codewords for a given length, constant weight and minimum Hamming distance. This algorithm is implemented in MATLAB. All the PSO operators expressly designed for the purpose of generating and maintaining feasibility of the codewords throughout the evolution for focused search in the feasible regions of the search space.

Keywords*:* ECC, Particle Swarm Optimization (PSO), Hamming distance.

I. INTRODUCTION

A major design criterion for all communication systems is to achieve error free transmission. But, during transmission data may get corrupted due to physical or logical faults which would bring the whole system down to destructive failures. So, Error Correcting Code (ECC) [5] which consists of designing a communication scheme for maximizing the reliability of information transmission through a noisy channel is required.

The transmission errors are classified based on the number of bits affected by these errors as Single bit Errors (Fig.1) or Burst Errors (Fig.2) [1]. The term single-bit error means that only 1 bit of a given data unit is changed either from 1 to 0 or from 0 to 1. This one bit changed cannot be ignored since one bit change can change the whole meaning of the data that is transmitted. These are least likely type of errors in serial data communications. Burst error means that two or more bits are changed when the transmitting data from the sender to the receiver the data units have changed from 0 to 1 or 1 to 0 because of the channel interference. Burst errors are likely to occur rather that the single bit error [1].

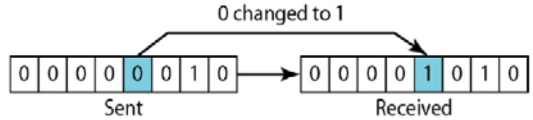

Figure 1 Single Bit Error

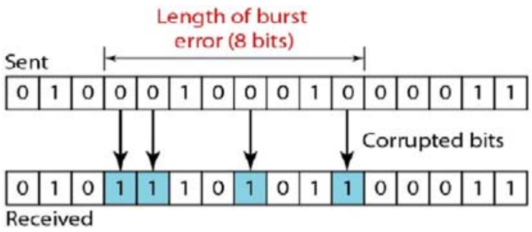

Figure 2 Burst Error

Various Error Detection & Correction (EDAC) methods can be used in communication systems such as, Parity codes [10], Checksums, Cyclic Redundancy codes [1], Hamming codes [2], Reed Solomon Codes [6], Residue Number Systems [8], Nordstorm-Robinson codes, Turbo codes, BCH codes and HVD [12]. In Error Correction Codes (ECC), a set of redundant bits is sent along with the original set of information bits. The receiver uses the redundant bits to determine if any errors have occurred during transmission.

One of the central concepts in coding for error control is the idea of the Hamming distance [2], which is the number of differences between the corresponding bits. A Hamming code can only correct a single error or detect a double error. The key is to split a burst error between several codewords, one error for each codeword. Then, instead of sending one codeword at a time, the codewords are arranged in a table and the bits in the table of a column are sent at a time. The minimum hamming distance (dmin) [2] is the smallest hamming distance between possible pairs. To guarantee the detection of up to ‘s’ errors, the minimum Hamming distance in a block code must be dmin = s+1.

The problem of finding an ECC with minimum hamming distance that corrects a given maximum number of errors is NP-hard [4]. For this reason some Metaheuristic search technique such as Particle Swarm Optimization [16] must be used. PSO is a heuristic global optimization method put forward originally by J. Kennedy and R. Eberhart in 1995. It is developed from swarm intelligence and is based on the research of birds and fishes flock movement behaviour. This algorithm is very simple, easily implemented and required fewer parameters as compared to other methods. PSO uses a population of individual called particles. It can take real number as particle. Each particle has its own position and velocity to move around the search space. Particles have memory and each particle keep track of its previous best position (pbest) [17] and corresponding fitness. It also has another value called global best (gbest) [17], which is the best value of all the particles pbest in the swarm. The basic concept of PSO lies in accelerating each particle toward its pbest and the gbest locations, with a random weighted acceleration at each time step. There are two key steps when applying PSO to optimization problems: the representation of the solution and the fitness function.

II. SYSTEM MODEL

In this paper, a special case of ECC have been generated using PSO which have an additional constraint of constant weight [13] i.e. a constant number of 1’s in every codeword. A constant-weight code is an error detection and correction code where all codewords share the same Hamming code. The ECC considered here is a set of binary n-tuples, C={x1, x2, x3,…, xn}. Each member is called a codeword. The Hamming distance dH(v,w) between two codewords is the number of bit positions where they differ. When the error correcting code is used in a communication system, up to dmin/2 transmission errors can be detected and corrected. Therefore a large minimum distance dmin is a desirable property of an error-correcting code. The Hamming weight of a codeword is the number of 1’s in it. We consider only constant-weight codes where all codewords have the same Hamming weight. A code can be formally represented by a three-tuple A(n, d, w), where ‘n’ is the length (number of bits) of each codeword, ‘d’ is the minimum Hamming distance between any pair of codewords and ‘w’ is constant codeword weight i.e. number of 1’s in each codeword. For generating ECC, a number of codewords are generated using PSO algorithm. Pseudo code for PSO:

1. Initialize each particle
2. For each particle:
   a. Calculate fitness value
   b. If the fitness value is better than the best fitness value (pbest) in history set current value as the new pbest
3. Choose the particle with the best fitness value of all the particles as the gbest
4. For each particle:
   a. Calculate particle velocity according equation (I)
   b. Update particle position according equation (II)

5. Repeat while maximum iterations or minimum error criteria is not attained

(I): v[] = v[] + c1 * rand() * (pbest[] – present[]) + c2* rand() * (gbest[] - present[])

(II): present[] = persent[] + v[]

where, v[] is the particle velocity

persent[] is the current particle (solution)

rand() is a random number between (0,1)

c1, c2 are learning factors

III. DESIGN

The PSO algorithm implemented here will work in following four steps:

1. Representation:

In the design of constant weight binary codes of size ‘S’, length ‘L’, weight ‘w’ and minimum distance ‘dmin’ or more, each possible solution string in the population is represented as binary string which is the concatenation of all the n codewords from the corresponding code. If the length of each codeword is CWL and if n codewords are required to be found then the solution string is represented as binary string of CWL * n elements. Each position in the list has values 0 or 1.

2. Population Initialization:

In the constant weight ECC every codeword should have constant weight i.e. fixed number of 1’s. This constraint has to be met otherwise the solution will be infeasible. So, this constraint has been taken care of in the code which generates only feasible codewords. The operators have also been designed in a way to maintain this feasibility of codewords.

3. Fitness Evaluation:

The fitness evaluation function for this purpose is required to evaluate the solution string and determine whether the solution string is the ideal solution to the problem or not. If not, then determine how good the solution is, so as to provide a direction and facilitate the search. The first part is achieved by checking the minimum Hamming distance between every pair of codewords. If the required minimum distance is there in every pair of codewords then this is the solution. Otherwise, the goodness of evaluated solution is determined as given below:

$$f = K - \sum_{\substack{x,y \in C \\ x \neq y}} \frac{1}{g^2 + 1} \quad g = \begin{cases} d_H(x,y) & \text{if } d_H(x,y) < d_{\min} \\ d_{\min} & \text{if } d_H(x,y) \geq d_{\min} \end{cases}$$

Where, K must prevent the cost function from taking negative values.

4. Selection:

The codewords of length n, which are unique, are selected by comparing different codewords and the string of length m, is generated using the codewords. The PSO is used for this generation and then the error is detected and then removed.

IV. RESULTS

The PSO algorithm code has been implemented in MATLAB and a number of ECCs with constant weight has been generated. For a fixed code size n, minimum distance dmin and constant weight w, the number of required codewords is fixed and then the program was run until the code is found or up to a maximum of 1000 iterations. The PSO is used for this generation and then the error is detected and removed. Using PSO algorithm, many good constant-weight codes are discovered whose parameters are listed in Table 1 and ECC generated are shown in Table 2, Table 3 and Table 4.

Table 1 Constant Weight Codes via PSO

Constant Weight ECCs	Length (n)	Distance (d)	Constant Weight (w)	Codeword Size
A(21,10,9)	21	10	9	22
A(22,10,9)	22	10	9	25
A(23,10,7)	23	10	7	18
A(23,10,8)	23	10	8	28
A(23,10,9)	23	10	9	27
A(23,10,10)	23	10	10	46
A(23,10,11)	23	10	11	47
A(24,10,8)	24	10	8	33
A(24,10,9)	24	10	9	27
A(24,10,11)	24	10	11	58

Table 2 Constant Weight Codes for A (23,10,7)

1.	00001101000000100010101
2.	00010000101000111000100
3.	00000101011010010001000
4.	10000000110111000010000
5.	00000010010000011110001
6.	10000011100000100100010
7.	00010001000101010000011
8.	11100000000010110000001
9.	01100001000100000111000
10.	01010110010100100000000
11.	11000100000001001100100
12.	01001000001110001000010
13.	01000000110000000001111
14.	10001010000100010001100
15.	00110010000010000010110
16.	00101010101001000000001
17.	10111001010000001000000
18.	00011000000011100101000

Table 3 Constant Weight Codes for A (23,10,8)

1.	011000000101010100010100
2.	110000110001011000010000
3.	000010011011111010000000
4.	000001010001101000001110
5.	100011100001000001100010
6.	100001001101000100011000
7.	001001011000000010100011
8.	000101001000001101101000
9.	001010110100010000001010
10.	010001100100000111000100
11.	000111110010001000000001
12.	010011010100100000110000
13.	000000111000000001011101
14.	110100101000100000001010
15.	100100000010110100100001
16.	110011001000001000000101
17.	101000010000011001100100
18.	010000000001100110100011
19.	100000011110001010000010
20.	001110001101000001000100
21.	010101000111010000000010
22.	011010000000100110101000
23.	100000000010010001110110
24.	101000100011000100000101
25.	000100010110000010101100
26.	100110000000000110010110
27.	001101100000011010010000
28.	001001000110100010010010

Table 4 Constant Weight Codes for A (24,10,8)

1.	000010101000000001100111
2.	001000000011001110000110
3.	000100000101001000001111
4.	001001101100001000001010
5.	000100110001011000100010
6.	000101011001000001010101
7.	001011011000100100000100
8.	000010001110011010100000
9.	001100101000010111000000
10.	001011000010000011011000
11.	010000000010010101110001
12.	000001010100010110000011
13.	001110000100100000010011
14.	010110010100010001001000
15.	00000100001111100000100l
16.	011000000100110100011000
17.	010001110010100000010010
18.	010100001100001100000101
19.	100000101111000000010001
20.	000000000101100011101010
21.	010111001001000010000010
22.	000000110110000100101100
23.	100100100000100010001101
24.	101000010100001001100001
25.	000000100100111001010100
26.	100101000110100101000000
27.	110000001101000001100010
28.	100010010001110010010000
29.	111110100010000000100000
30.	100010001001001101001000
31.	111000000000110100001010
32.	100001100000001110110000
33.	110001000101010000100100

V. CONCLUSION

Error Correction Code (ECC) is one of the important aspects in Data Communication Networks, dealing in detection and correction of errors. In this paper, constant weights ECCs are generated using a Metaheuristic search technique, PSO which is very useful in designing good codes. A constant weight ECC is formally represented by a three-tuple A(n, d, w), describing the length of each codeword (n), the minimum Hamming distance between any pair of codewords (d) and constant codeword weight (w). Expressly designed PSO operators ensure the feasibility of the codewords throughout the evolution and confine the search in the feasible regions of the search space. Proper tweaking of the control parameters and PSO operators resulted in obtaining very good results.

The importance of this problem calls for further investigation in terms of generation of more complex ECCs, better metaheuristic techniques and better operators which can be taken up in future work.

References:

[1] Afiqah Azahari, Raed Alsaqour, Md. Al-Hubaishi and Mueen Uddin, "Review of Error Detection of Data Link Layer in Computer Network", Middle-East Journal of Scientific Research, vol. 18(8), pp 1105-1110, 2013.

[2] R.W.Hamming, "Error Detecting and Error Correcting Codes", Bell System Technical Journal, vol. 29(2), pp 147-160, 1950.

[3] Vikas Gupta and Chanderkant Verma, "Error Detection and Correction: An Introduction", IJARCSSE, vol. 2(11), pp 212-218, 2012.

[4] Daniel E. McCarney, Sheridan Houghten and Brian J. Ross, "Evolutionary Approaches to the Generation of Optimal Error Correcting Codes", GECCO, 2012.

[5] Varinder Singh and Narinder Sharma, "A Review on Various Error Detection and Correction Methods Used in Communication", AIJRSTEM, vol. 9(3), pp 252-257, 2015.

[6] Priyanka Shrivastava and Uday Pratap Singh, "Error Detection and Correction Using Reed Solomon Codes", IJARCSSE, vol. 3(8), pp 965-969, 2013.

[7] M. T. Anwar, P. K. Lala, and P. Thenappan, "Decoder Design for a New Single Error Correcting/Double Error Detecting Code", International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering, vol. 1(4), pp 649-653, 2007.

[8] M. Roshanzadeh and S. Saqaeeyan, "Error Detection & Correction in Wireless Sensor Networks By Using Residue Number Systems", I. J. Computer Network and Information Security, vol. 2, pp 29-35, 2012.
[9] Minal M. Jaiswal and Ashish E. Bhande, "Review on Error Detection and Correction", IJERGS, vol. 3(2), pp 589-591, 2015.
[10] Brajesh Kumar Gupta and Rajeshwar Lal Dua, "Review Paper On Communication By Hamming Code Methodologies", International Journal of Electrical, Electronics and Computer Engineering, vol. 1(1), pp 52-54, 2012.
[11] Ankur Goyal and Vinita Nareda, "A Review Studies on Burst Error Detection and Correction in Big Data", IJSR, vol. 6(5), pp 943-947, 2017.
[12] Shaik Sumera Bhanu, G. Sreenivasulu and Syed Kareemsaheb, "A Review on Various Error Detection and Correction Using HVD Implementation", IJR, vol. 2(9), pp 222-226, 2015.
[13] Mao-Chao Lin, "Constant Weight Codes for Correcting Symmetric Errors and Detecting Unidirectional Errors", IEEE Transactions on Computers, vol. 42(11), pp 1294-1302, 1993.
[14] W. Kautz and B. Elspas, "Single-error-correcting codes for constant-weight data words", IEEE Transactions on Information Theory, vol. 11(1), pp 132-141, 1965.
[15] Sergio Verdii and Victor K. Wei, "Explicit Construction of Optimal Constant-Weight Codes for Identification via Channels", IEEE Transactions on Information Theory, vol. 39(1), pp 30-36, 1993.
[16] Qinghai Bai, "Analysis of Particle Swarm Optimization Algorithm", Computer and Information Science, vol. 3(1), pp 180-184, 2010.
[17] Nanda Dulal Janaa, Tapas Sia and Jaya Silb, "Particle Swarm Optimization with Adaptive Mutation in Local Best of Particles", IPCSIT, vol. 38, 2012.
[18] Arvinder Kaur and Divya Bhatt, "Particle Swarm Optimization with Cross-Over Operator for Prioritization in Regression Testing", International Journal of Computer Applications, vol. 27(10), pp 27-34, 2011.
[19] Dian Palupi Rini, Siti Mariyam Shamsuddin and Siti Sophiyati Yuhaniz, "Particle Swarm Optimization: Technique, System and Challenges", International Journal of Computer Applications, vol. 14(1), pp 19-27, 2011.
[20] Divya Kataria and Sudesh Jakhar, "Convolution Code Encoder Design Using Particle Swarm Optimization for Constraint Length 6", IJARSE, vol.3(9), pp 430-436, 2014.
[21] Sharandeep Singh, "A Review on Particle Swarm Optimization Algorithm", IJSER, vol. 5(4), pp 551-553, 2014.
[22] Shailendra S. Aote, M M Raghuwanshi and Latesh Malik, "A Brief Review on Particle Swarm Optimization: Limitations & Future Directions", IJCSE, vol. 2(5), pp 196-200, 2013.