Reliability Level List Based Direct Target Codeword Identification Algorithm for Binary BCH Codes

B.YAMUNA[1], T.R.PADMANABHAN [2]
[1]Department of ECE, [2] Department of IT
Amrita Vishwa Vidyapeetham. Amrita School of Engineering
Amrita Nagar, Coimbatore-641 112
INDIA
b_yamuna@cb.amrita.edu

Abstract: - In BCH coded schemes the reliability information available with the demodulated bits can be effectively used for soft decision decoding (SDD) to improve signal to noise ratio performance. Chase algorithms, their adaptations, and modifications available for SDD trade complexity for performance to different levels. A new iterative algorithm – Reliability Level List based Direct Target Codeword Identification Algorithm (DTCI) - is proposed in the paper; the algorithm yields the best that is possible with SDD. The concept of reliability level list (RLL) introduced in the paper is central to the application of the algorithm. At every stage of the iterative process followed, the algorithm uses the reliability information of the bits and identifies the next most likely candidate word to be examined. This ensures that the correct decoded codeword is identified through the shortest number of steps. Detailed simulation studies with different BCH codes amply bring out the effectiveness and superiority of the algorithm.

Key-Words: - BCH codes, Reliability based decoding, Soft decision decoding, Probability of error.

1 Introduction

Channel coding techniques which combat channel impairments are an integral part of digital wireless communication systems. The fact that Hard Decision Decoding (HDD) discards information regarding the reliability of the received bit has been its bane. If the number of errors exceeds $\lfloor d_{\min}/2 \rfloor$, the decoding is not unique. Approaches where we choose reliability information in the received bit sequence to go beyond HDD and identify codewords in a more meaningful manner comprehensively, constitute Soft Decision Decoding (SDD). Chase and Forney [1-2] are the forerunners of reliability based SDD algorithms which use reliability information along with the hard decision decoded sequence to refine the decoding process. Fossorier et al. proposed a soft decoding scheme based on processing of the most reliable positions [3]. Of these, Chase algorithms have been of great interest to many researchers; its adaptations are available for binary linear error-correcting block codes [4]. These refinements have basically been multidimensional in nature in the sense that they exploit available computing power to improve performance, allow transmission through feeble channels (channels with low SNR values), exploit channel capacity to the utmost subject to specific performance criteria, and so on.

The class of Chase algorithms decodes using the reliability values obtained from the demodulator and running subsequent trials of codeword estimation using suitable test patterns. The trials are run on a conventional (algebraic) decoding algorithm. These algorithms are all based on processing of the least reliable bits so that in each trial, different combinations of the least reliable received bits are processed and the decoder output is the candidate word with the best soft decision metric**.** These algorithms have a decoding radius of ($d_{\min}$ -1), $d_{\min}$ being the minimum Hamming distance. Variations of Chase algorithms essentially focus on reducing the number of trials and hence the complexity involved in decoding to the codeword.

Chana et al. have proposed an interesting SDD algorithm for binary cyclic codes which performs permutation decoding over a fixed number of least reliable positions of the received word [5]. A reliability based soft decision decoding algorithm for binary cyclic block codes that identifies the codeword with minimal effort has been proposed recently [6]. In this paper, the reliability based soft decision decoding algorithm has been extended for decoding binary BCH codes over the Additive

White Gaussian Noise (AWGN) channel with Binary Phase-Shift Keying (BPSK) signaling. Performance wise the proposed algorithm has an edge over the class of Chase algorithms.

In the proposed algorithm the reliability of information obtained from the demodulator is used in computing the error probability and hence estimating all possible patterns of ($d_{\min}$ -1) errors and even beyond. The error probability calculated from the reliability information is used to decide on the occurrence of single error, double errors, triple errors etc., and hence decode from amongst the 2^n possible words. The procedure is structured to yield the target codeword with minimal number of trials.

The rest of the paper is organized as follows: Section 2 discusses the preliminaries of soft decision decoding and the Chase algorithms, Section 3 presents the proposed Direct Target Codeword Identification (DTCI) algorithm, Section 4 discusses the simulation results and Section 5 forms the conclusion.

2 Preliminaries

For an (n,k) binary BCH code the message sequence of length k is encoded into an n-bit length codeword C using conventional encoding techniques. The code bits $C_i = C_1C_2 \ldots..C_n$ are fed to data modulator which converts the bit stream to the appropriate wave form (BPSK modulated) for transmission through the channel. This signal may be corrupted by the channel. Let the received signal be represented as $r = (r_0, r_1, r_2 \ldots\ldots r_{n-1})$. From the r_i values, a binary sequence Z_i is generated based on the hard-decision rule:

$$Z_i = \begin{cases} 0 & \text{for } r_i < 0 \\ 1 & \text{for } r_i \geq 0 \end{cases} \quad (1)$$

The magnitude of r_i can be used as a reliability measure of the hard-decision decoded bit and decoding decision regarding the error position could be made. The larger the $|r_i|$ the more reliable the hard decision regarding the bit value and less likely that the bit is in error.

The Chase algorithms use a test vector T which is added to the received word to form the 'distorted word' and seek the codeword by scanning around the distorted word within a radius of $\lfloor d_{\min}/2 \rfloor$ [2]. The three Chase algorithms differ in the number and pattern of test vectors T (and hence the number of trials) needed to decode to the closest codeword. Hence the scope of Chase algorithms is limited to a Hamming sphere of radius $d_{\min}$ -1 about the received word with the exception of Chase III which decodes some patterns of errors beyond $d_{\min}$ - 1.

- For Chase I algorithm the set T consists of all binary vectors of length n which contain exactly $\lfloor d_{\min}/2 \rfloor$ ones. (i.e., all possible patterns of errors up to $d_{\min}$ -1).
- For Chase II algorithm the set T consists of every combination of 1's, which are located in the $\lfloor d_{\min}/2 \rfloor$least reliable positions.
- For Chase III algorithm the set T consists of all binary vectors of length n which contain ones in the i least reliable positions and zeros elsewhere, where i = 0, 2, 4, . . . , $d_{\min}$ -1, if $d_{\min}$ is odd and i =0, 1, 3, 5, . . . , $d_{\min}$ - 1,if $d_{\min}$ is even.

The main drawback of Chase I algorithm is the large number of test patterns involved in the trials. ${}^nC_{\lfloor d_{\min}/2 \rfloor}$ numbers of test patterns which run through the entire length of n bits are invoked in the algorithm. The test pattern generation does not use the reliability information; the latter is used to derive an analog weight which decides the selection of the error pattern from amongst the set of error patterns obtained. Chase II algorithm generates distorted words with $2^{\lfloor d_{\min}/2 \rfloor}$ test patterns all of them focused on the least reliable end. Chase III algorithm uses $\lfloor d_{\min}/2 \rfloor$+1 test patterns for the search. But with these two – unlike with Chase I – only a restricted number of error patterns up to $d_{\min}$ - 1 are decoded.

The decoding limit of Chase algorithms for a specific case of (15,7) binary BCH code with $d_{\min}$ = 5 is given in Figure 1. From Figure 1 it is seen that in Chase I all the distorted words - ${}^{15}C_2$ in number- lie within a Hamming sphere of radius $\lfloor d_{\min}/2 \rfloor$= 2. These on decoding using conventional algebraic decoding method identify all the codewords within a Hamming sphere of radius $d_{\min}$ -1 around Z_i (Figure 1a), these being the candidate codewords. All possible 0 to 4 error patterns are included here.

It is seen that Chase II uses four test patterns ($2^{\lfloor d_{\min}/2 \rfloor}$) - all zero pattern and three patterns with all combinations of 1's in the least two $\lfloor d_{\min}/2 \rfloor$reliable positions. All the candidate codewords lie within Hamming spheres of radius $\lfloor d_{\min}/2 \rfloor$- here $\lfloor d_{\min}/2 \rfloor$ = 2 - around the corresponding distorted words - that is within the envelope of these 4 spheres as represented in Figure

1b. It is evident from the figure that all these candidate codewords lie within a Hamming sphere of radius d_{min}-1 and form a subset of those in Chase I.

It is pertinent to point out here that the simplification from Chase I to Chase II has left out some of the candidate codewords within the Hamming sphere of radius d_{min}-1 (Figure 1b). If one of the left out candidate words were to have a lower value of the metric, Chase II would have decoded wrongly. One can see from the figure that the search is restricted to patterns of 2 errors, all 3 error cases where one of the errors is in any one of the two least reliable bit positions, and all 4 error cases where two of the errors are in the two least reliable bit positions in the received word.

It is seen that Chase III uses three test patterns ($\lfloor d_{min}/2 \rfloor+1$) - all zero pattern, a pattern of two 1's and another of four 1's from the least reliable end. All the candidate codewords lie within Hamming spheres of radius $\lfloor d_{min}/2 \rfloor$- here $\lfloor d_{min}/2 \rfloor$= 2 - around the corresponding distorted words- that is within the envelope of these 3 spheres as shown in Figure 1c.

It is evident from the figure that all these candidate codewords do not lie within a Hamming sphere of radius d_{min}-1; in this respect Chase III stands apart from both Chase I and Chase II.

From the figure we find that the search is restricted to all patterns of 2 errors, all 4 error cases when two of the errors are in the two least reliable bit positions, all 6 error cases where four of the errors are in the four least reliable bit positions in the received word.

Though it corrects up to d_{min} -1 errors in any of the positions, Chase I algorithm is of limited interest due to the extensive search involved. Chase II and Chase III algorithms are simpler by one or two orders. The simplicity is achieved by a trade-off in the error pattern decoded as has been explained above. The same is true of Generalized Minimum Distance (GMD) [7] (Essentially Chase III is the same as GMD vis-à-vis binary codes) as well as a number of other modifications [8] and generalizations [9] for bounded distance decoding.

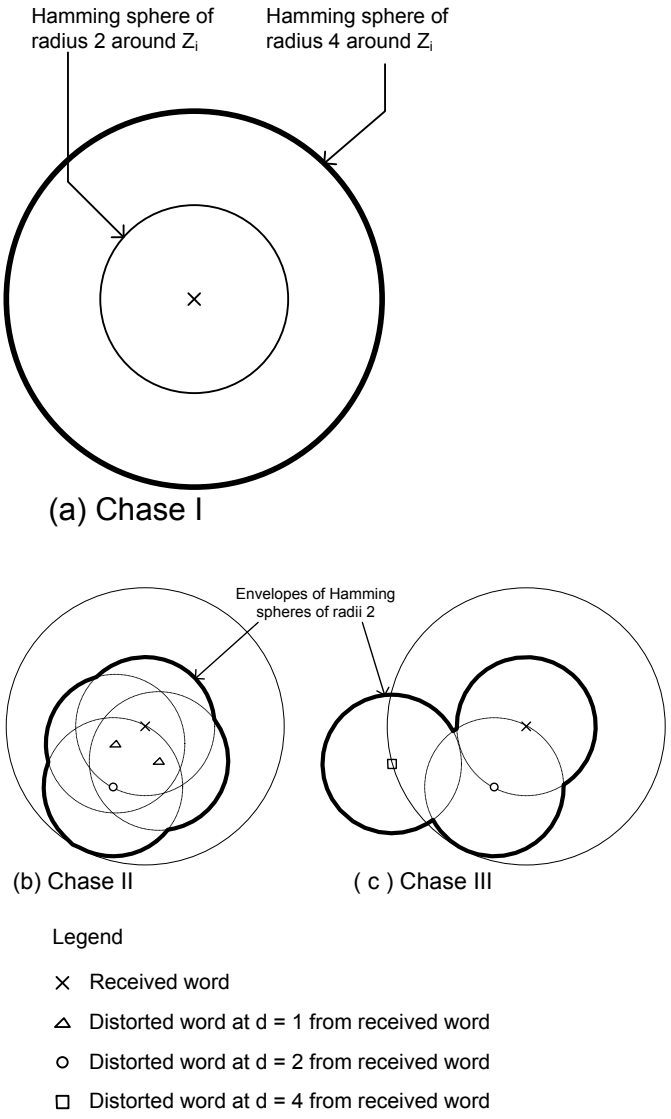

Figure 1 Decoding Bounds of Chase Decoding Algorithms: In each case the enclosures in thick lines represent the scan range.

While decoding, with these classes of algorithms the focus is on optimizing a metric in terms of bits at the least reliable end; others optimize a metric defined in terms of bits at the most reliable end [10] to identify the target codeword. By providing different trade-offs between error performance and complexity these algorithms play a significant role in soft decision decoding. However the fact remains that there is a clear need for an algorithm which can use the reliability information in a structured manner and with minimal effort identify the most reliable codeword. Such an algorithm is evolved in the sequel.

3 Direct target codeword identification algorithm

A new class of algorithm that uses soft information from the channel directly as an index for estimating the error positions is proposed here. The novelty of the algorithm lies in the following:

- All the patterns of errors that are guaranteed to be decoded by Chase algorithm I are successfully decoded by using an approach that eliminates the need for running repeated trials on conventional decoder. This approach is much simpler as brought out through the case studies.
- The algorithm decodes all patterns of errors up to $d_{\min}$ -1 which are beyond the decoding limits of Chase II and Chase III algorithms.
- There exists the possibility of Chase III erroneously declaring a less reliable candidate codeword as the target codeword which is avoided here. This has been brought out through an illustrative example.
- The algorithm also has an edge over the class of Chase algorithms by being able to decode beyond their upper bound of $d_{\min}$ - 1.

Every hard decision represented by (1) has a probability associated with it. For the AWGN channel it is given by

$$Q\left(\frac{m_i}{\sigma}\right) = \int_{m_i/\sigma}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx \qquad (2)$$

σ^2 being the variance of the noise and $| r_i | = m_i$ is the reliability value of bit *i*. In the proposed algorithm the received word is decoded by successive scanning done from the least reliable end. At every step the probability of error as given by (2) is used as the index for selecting the next candidate word that is to be examined.

The steps involved in the algorithm are explained through an illustrative example of a (15, 7) binary BCH code.

Example - I

304eh is the transmitted code word. With σ = 0.8, the received word obtained is 78cch. There are 4 errors at bit positions 1, 7, 11, 14. As before let $| r_i | = m_i$ be the reliability value of bit *i*. The bits are assigned integer reliability indices *k* from 1 to 15 in the ascending order of magnitude m_i as in Table 1.

Chase I decodes the codeword with an extensive search as explained earlier. Chase II fails to decode this 4 error pattern. The possible distorted words are 78cch, 780ch, 788ch, and 784ch. Algebraic decoding of these four distorted words results in decoding failure.

With Chase III the three possible distorted words are 78cch, 780ch and 781eh. Algebraic decoding of these four distorted words also results in decoding failure.

Table 1

Reliability magnitude and index for Example - I

i	m_i	*k*
0	1.107031	10
1	0.140967	3
2	1.151953	11
3	0.987512	8
4	0.405945	4
5	2.387561	15
6	0.095972	1
7	0.110425	2
8	2.065784	14
9	1.741907	13
10	1.014600	9
11	0.431921	6
12	1.326001	12
13	0.408484	5
14	0.432691	7

If $Q\left(\frac{m_i}{\sigma}\right) = \int_{m_i/\sigma}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx$ represents the probability that the i^{th} bit with *k* as the reliability index is received wrongly then let

$$q_i[k] = Q\left(\frac{m_i}{\sigma}\right) \qquad (3)$$

$$p_i[k] = 1 - Q\left(\frac{m_i}{\sigma}\right) \quad (4)$$

where $p_i[k]$ is the probability that the i^{th} bit with k as the reliability index is received correctly. The index i representing the bit position – being superfluous here – has been left out in the rest of the discussion. All the 2^{15} terms in the expansion of the product

$$\prod_{k=1}^{15}(p[k]+q[k])$$

together represent the probability space of the candidate word of 15 bits. These 2^{15} elements of this space have their own associated probability of being the target codeword. All the possible 2^7 codewords are included in this set of 2^{15} – each with its own probability.

The list formed with all the terms in the expansion of the above expression arranged in descending order of magnitude is called as '*Reliability Level List*' (RLL) here. For any given k value $q[k] \leq 0.5$ and hence $p[k] \geq 0.5$. Hence the term $\prod_{k=1}^{15} p[k]$ representing the probability that all bits are received without error has the largest magnitude and is at the top of RLL.

$$\prod_{k=1}^{15} p[k] = (1\text{-}0.00142)\ (1\text{-}0.00491)\ (1\text{-}0.01473)\ (1\text{-}0.04871)\ (1\text{-}0.07495)\ (1\text{-}0.08321)\ (1\text{-}0.10236)\ (1\text{-}0.10853)\ (1\text{-}0.29429)\ (1\text{-}0.29463)\ (1\text{-}0.30481)\ (1\text{-}0.30594)\ (1\text{-}0.43007)\ (1\text{-}0.44511)\ (1\text{-}\ 0.45226) = 0.026297$$

The corresponding word – being the most reliable of all the words – deserves examination first; the word here is the received word (78cch) itself. The examination can be carried out directly by dividing the word by the generator polynomial of the code. A zero remainder implies the word to be a codeword and the search stops here. This not being the case here the search has to be continued with the next entry in the RLL. $q[1]$ being the maximum amongst all the $q[k]$ values, the product

$$q[1]\prod_{k=2}^{15} p[k] = 217.12\times10^{-4}$$

has the second largest magnitude; it is the second entry in the RLL indicating that the word with the bit for which $k = 1$ – that is b_6 – being in error is the next one to be examined. The word is 788ch which is not a codeword. The third entry in the RLL is the product

$$q[2]\,p[1]\prod_{k=3}^{15} p[k] = 174.16\times10^{-4}$$

The corresponding word to be examined is 784ch taking b_7 (since $i = 7$ for $k = 2$) to be in error. Since this is not a codeword, the fourth entry in RLL is to be examined: It is decided by the larger of the two of the products:

1. $q[3]p[2]p[1]$ – the probability that bit for $k = 3$ (b_1)is in error.
2. $p[3]q[2]q[1]$ – the probability that bits for $k = 2$ and $k = 1$ (b_6 and b_7) are in error.

If $q[3]p[2]p[1] > p[3]q[2]q[1]$, this entry in RLL is $q[3]p[2]p[1]\prod_{k=4}^{15} p[k]$, else it is $p[3]q[2]q[1]\prod_{k=4}^{15} p[k]$.

The dual process of identifying the next most probable word to be examined – that is the next entry in the RLL – and dividing it by the generator polynomial to ascertain whether it is a codeword is continued until the codeword – the target codeword itself – is identified. This completes successful decoding and RLL formation need not be continued further. The progress of the decoding process is summarized in Table 2.

The salient features of the proposed algorithm can be summarized as follows:

1. Once a codeword has been identified further scan is not needed because subsequent codewords if identified will be less reliable than the earlier one.
2. The search has shown the bits b_7, b_1, b_{11} and b_{14} to be in error and the identified target codeword - 304eh - is the transmitted codeword itself.
3. The RLL entry at any stage can be represented as the product $f\prod_{k=1}^{15} p[k]$. The term $\prod_{k=1}^{15} p[k]$ in this product being a constant, one need to compute only the factor f at each stage to decide the next entry in RLL and the corresponding word to be examined; use of the factor f in place of the full probability value simplifies the computations [6].
4. In fact the magnitude of f associated with a word decides the position of the word in the RLL. For a candidate codeword the f value represents its metric. In turn from amongst a set of candidate codewords the one with the maximum value of f represents the target codeword. The values of f are also given in Table 2.
5. The candidate codewords identified by

applying Chase I algorithm are given in Table 3 along with their respective *f* values. Despite the extensive search, Chase I also zeros on to 304eh as the target codeword, its value of the metric being the smallest.

6. Though Chase I will successfully decode here, Chase II and Chase III fail.

Table 2

RLL Decoding Sequence index for Example - I

S.No	Bits in error	*K*	Error Probability	*f*
1	None		262.97×10^{-4}	1
2	6	1	217.12×10^{-4}	0.82568
3	7	2	174.16×10^{-4}	0.80216
4	1	3	131.42×10^{-4}	0.7546
5	6,7	1,2	99.17×10^{-4}	0.662329841
6	6,1	1,3	61.79×10^{-4}	0.623062337
7	7,1	2,3	37.40×10^{-4}	0.60531
8	6,7,1	1,2,3	18.69×10^{-4}	0.499795173
9	4	4	8.24×10^{-4}	0.4408
10	13	5	3.61×10^{-4}	0.43846
11	11	6	1.51×10^{-4}	0.4177
12	14	7	0.63×10^{-4}	0.41701
13	6, 4	1,4	0.22917×10^{-4}	0.363959547
14	6,13	1,5	0.08297×10^{-4}	0.362025835
15	7,4	2,4	0.029337024×10^{-4}	0.35359
16	7,13	2,5	0.010318×10^{-4}	0.351711
17	6,11	1,6	3558.57×10^{-10}	0.344884638
18	6,14	1,7	1225.289×10^{-10}	0.344320676
19	7,11	2,6	410.5×10^{-10}	0.335058
20	7,14	2,7	37.33×10^{-10}	0.33451
21	1,4	3,4	45.7×10^{-10}	0.33263
22	1,13	3,5	15.11×10^{-10}	0.33086
23	1,11	3,6	4.763×10^{-10}	0.31519
24	1,14	3,7	1.499×10^{-10}	0.31468
25	6,7,4	1,2,4	0.43765×10^{-10}	0.291953491
26	6,7,13	1,2,5	0.127095×10^{-10}	0.290402347
27	6,7,11	1,2,6	3516×10^{-15}	0.276652378
28	6,7,14	1,2,7	971.14×10^{-15}	0.276199991
29	6,1,4	1,3,4	266.71×10^{-15}	0.274643
30	6,1,13	1,3,5	72.861×10^{-15}	0.273184
31	7,1,4	2,3,7	19.441×10^{-15}	0.266819
32	7,1,13	2,3,5	5.1596×10^{-15}	0.265402
33	6,1,11	1,3,6	1.3428×10^{-15}	0.260249
34	6,1,14	1,3,7	0.34888×10^{-15}	0.259824
35	7,1,11	2,3,6	0.0882×10^{-15}	0.252835
36	7,1,14	2,3,7	0.02226×10^{-15}	0.252422
37	6,7,1,4	1,2,3,4	0.00490×10^{-15}	0.220308449

38	6,7,1,13	1,2,3,5	0.00107×10^{-15}	0.219137954
39	6,7,1,11	1,2,3,6	0.000224×10^{-15}	0.208762211
40	6,7,1,14	1,2,3,7	4.656×10^{-20}	0.208420839
41	4,13	4,5	0.8998×10^{-20}	0.19327
42	4,11	4,6	0.16567×10^{-20}	0.18412
43	4,14	4,7	0.03045×10^{-20}	0.18382
44	13,11	5,6	0.00558×10^{-20}	0.18314
45	13,14	5,7	0.00102×10^{-20}	0.18284
46	11,14	6,7	0.000178×10^{-20}	0.17419
47	6,4,13	1,4,5	$2.8329250 \times 10^{-25}$	0.15958
48	7,4,13	2,4,5	0.4392×10^{-25}	0.155034
49	6,4,11	1,4,6	0.06678×10^{-25}	0.15202
50	6,4,14	1,4.7	0.01014×10^{-25}	0.15178
51	6,13,11	1,5,6	0.001533×10^{-25}	0.151218
52	6,13,14	1,5,7	23.14×10^{-30}	0.150971
53	7,4,11	2,4,6	3.4176×10^{-30}	0.147693
54	7,4,14	2,4,7	0.50393×10^{-30}	0.147452
55	7,13,14	2,5,7	0.0739×10^{-30}	0.146663
56	1,4,13	3,4,5	0.01078×10^{-30}	0.14583
57	6,11,14	1,6,7	0.00155×10^{-30}	0.143819
58	7,11,14	2,6,7	$0.0002166 \times 10^{-30}$	0.139724
59	1,4,11	3,4,6	3.01×10^{-35}	0.138926
60	1,4,14	3,4,7	$0.4175 \times 10^{-35`}$	0.138698
61	1,13,11	3,5,6	$0.05770 \times 10^{-35`}$	0.138215
62	1,13,14	3,5,7	$0.00796 \times 10^{-35`}$	0.137989
63	1,11,14	3,6,7	$0.001947 \times 10^{-35`}$	0.131442
64	6,7,4,13	1,2,4,5	$0.0001340 \times 10^{-35`}$	0.128008449
65	6,7,4,11	1,2,4,6	1.6337×10^{-40}	0.121947506
66	6,7,4,14	1,2,4,7	0.19890×10^{-40}	0.121748095
67	3	8	0.024214×10^{-40}	0.12174
68	6,7,13,11	1,2,5,6	0.002937×10^{-40}	0.121298825
69	6,7,13,14	1,2,5,7	$0.0003557 \times 10^{-40}$	0.121100475
70	7,1,4,13	2,3,4,5	4.1611×10^{-45}	0.116988743
71	6,7,11,14	1,2,6,7	0.48014×10^{-45}	0.115387401
72	10	9	0.05475×10^{-45}	0.11403
73	7,1,4,11	2,3,4,6	0.006102×10^{-45}	0.111449561
74	7,1,4,14	2,3,4,7	0.000679×10^{-45}	0.111267316

75	7,1,13,11	2,3,5,6	7.5264 x 10^{-50}	0.11085643
76	7,1,13,14	2,3,5,7	0.8329 x 10^{-50}	0.11067516
77	7,1,11,14	2,3,6,7	0.08782 x10^{-50}	0.105435394

Direct Target Codeword Identification Algorithm (DTCI)

The procedure evolved through the illustrative example above can be cast as a structured algorithm; we call this the 'Direct Target Codeword Identification Algorithm (DTCI)'. The step by step procedure is as follows:

1. Let $\{r_i\}$, $\{m_i\}$, and $\{b_i\}$ represent the sets of received signal values, their magnitudes, and the corresponding bit values.
2. Compute the set $\{Q(m_i)\}$ where

$$Q\left(\frac{m_i}{\sigma}\right) = \int_{m_i/\sigma}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx$$

is the probability of the i^{th} bit being in error.

3. Arrange the set $\{Q\left(\frac{m_i}{\sigma}\right)\}$ in descending order of magnitude and assign integer values k to them such that k =1 for the bit with the largest value of $Q\left(\frac{m_i}{\sigma}\right)$, k =2, for the next and so on until $k = n$, for the bit with the smallest value of $Q\left(\frac{m_i}{\sigma}\right)$.

4. Let $\{q[k]\}$ be the rearranged set of numbers in step (3) above. Compute $s[k]$ as

$$s[k] = \frac{q[k]}{1 - q[k]}$$

and form the set $\{s[k]\}$.

5. The first three entries of the Reliability Level List (RLL) are 1, $s[1]$, and $s[2]$ respectively in that order. The corresponding candidate words to be examined are $\{b_i\}$, $\{b_i\}$ with $b_{i[1]}$ complemented, and $\{b_i\}$ with $b_{i[2]}$ complemented respectively. Examine each in the same order; if any of them is a codeword, it is the target codeword. This completes decoding. Note that the indices $i[1]$ and $i[2]$ here stand for the bit indices corresponding to k =1 and k =2.

6. If none of the above is the target codeword, the next entry in the RLL is to be decided; it is $s[3]$ if $s[3] > s[1]s[2]$ and the candidate word is $\{b_i\}$ with $b_{i[3]}$ complemented; else the RLL entry is $s[1]s[2]$ and the candidate word is $\{b_i\}$ with $b_{i[1]}$ and $b_{i[2]}$ complemented. Examine the candidate word; if it is a codeword, it is the target codeword and decoding is complete.

7. If decoding is not completed in step (6), the next entry in RLL is to be identified; If the previous entry was $s[1]s[2]$, this one is $s[3]$ and the candidate word is $\{b_i\}$ with $b_{i[3]}$ complemented; else it is the larger of $s[4]$ and $s[1]s[2]$. Correspondingly the candidate word is $\{b_i\}$ with $b_{i[4]}$ complemented or $\{b_i\}$ with $b_{i[1]}$ and $b_{i[2]}$ complemented as the case may be. Examine the candidate word; if it is a codeword, it is the target codeword and decoding is complete.

Proceed as above – always deciding the next entry in RLL, forming the candidate word by complementing the selected bits and checking whether it is a codeword – the target codeword – the one with the maximum value of f – the metric.

Table 3

Chase I – Decoding results for Example - I

Candidate code words	Hamming distance from 78cch	f	Analog metric value
7ac8h	2	0.001211	2.89386
58cfh	3	0.030033	1.656482
7d8ch	3	0.000465	3.176356
304eh	4	0.105435394	1.116004
7c5dh	4	0.003659601	2.638001
609ch	4	0.007784	2.259839
9cch	4	0.000046193	4.23296

A few illustrative examples are considered here to bring out the effectiveness of the algorithm.

Example - II

This is a hypothetical case of a (15, 7) code with 304eh as the transmitted code word and 317eh at a Hamming distance of 3 from it as the received word; the errors are in b_4, b_5, and b_8. The bit indices and the corresponding reliability indices assigned to each of them are given in Table 4.

Table 4

Reliability magnitude and index for Example –II

Bit index (*i*)	Reliability index (*k*)
14	15
13	14
12	13
11	12
10	4
9	10
8	7
7	9
6	8
5	6
4	5
3	11
2	3
1	2
0	1

Chase I decodes this properly but Chase II fails. Chase III uses the three *T* vectors- the received word, the received word with $b_{i[1]}$ and $b_{i[2]}$ complemented, and the received word with $b_{i[1]}$, $b_{i[2]}$, $b_{i[3]}$, and $b_{i[4]}$ complemented. The Hamming spheres around the corresponding distorted words – 317eh, 317dh, and 3579h are scanned to identify candidate codewords; the word 3579h is identified as the only candidate codeword and erroneously returned as the target codeword.

The *s* values for the first few least reliable bits are reproduced in Table 5; selected and relevant segments of the RLL are shown in Table 6. Applying the proposed algorithm, the word 304eh - is the first candidate codeword identified and it is returned as the target codeword. Incidentally the codeword 3579h with f = 0.883466 is not the target codeword, since its reliability is less than that of 304eh for which f = 0.90142773.

Table 5

s values for Example - II

S.No	Bit index *i*	*k*	*s* [*k*]
1	0	1	0.971
2	1	2	0.970
3	2	3	0.969
4	10	4	0.968
5	4	5	0.967
6	5	6	0.966
7	8	7	0.965

Table 6

Partial RLL for Example - II

S.No	Bits in error	*k*	*f*
1	0	1	0.971
2	1	2	0.970
3	2	3	0.969
.	.	.	.
.	.	.	.
.	.	.	.
61	10,4,8	4,5,7	0.90329404
62	2,5,8	3,6,7	0.90329211
63	**4,5,8**	**5,6,7**	**0.90142773**

Example - III

A (31, 16) binary BCH code with d_{min} = 7 and $t = \lfloor d_{min}/2 \rfloor$ = 3 is taken. The message ab30h has been encoded as 55986ad2h and transmitted over an AWGN channel with σ = 0.8. Details of an erroneously received word are reproduced in Table 7.

Table 7

Reliability magnitude, index and *s* values for Example - III

S.No	Bit index *i*	m_i	*k*	*s* [*k*]
1	19	0.317847	1	0.528046
2	16	0.332117	2	0.512938
3	1	0.362207	3	0.482241
4	24	0.378931	4	0.465968
5	8	0.386804	5	0.458513
6	6	0.402893	6	0.443567
7	11	0.408740	7	0.43825
8	22	0.453369	8	0.3995
9	12	0.464685	9	0.390217
10	7	0.518054	10	0.34884
11	13	0.519543	11	0.34777
12	25	0.556042	12	0.32187
13	20	0.564929	13	0.315898
14	9	0.637524	14	0.270205
15	21	0.689807	15	0.241097
16	4	1.099182	16	0.092563
17	28	1.111890	17	0.089638
18	14	1.140991	18	0.083313
19	15	1.399084	19	0.041834
20	2	1.430371	20	0.038301
21	5	1.452161	21	0.035997
22	3	1.541406	22	0.027751
23	17	1.593726	23	0.02373
24	18	1.632312	24	0.021091
25	27	1.661707	25	0.01926
26	23	1.760144	26	0.014092
27	30	1.765674	27	0.013842
28	26	1.808460	28	0.012035
29	0	1.884961	29	0.009323
30	10	1.991687	30	0.006435
31	29	2.580152	31	0.00063

Table 8

Partial RLL for Example -III

S.No	Bits in error	*k*	*f*
1	19	1	0.528046
2	16	2	0.512938
3	1	3	0.482241
.	.	.	.
.	.	.	.
.	.	.	.
333	19,16,4	1,2,16	0.025071
334	19,16,28	1,2,17	0.024279
335	**1,11,7,20**	**3,7,10,13**	**0.023289448**

Selected and relevant segments of the RLL are shown in Table 8. Applying the proposed algorithm, the word 55986ad2h – with f = 0.023289448 – is the first candidate codeword identified and it is returned as the target codeword. This is returned after 335 entries in the RLL have been checked.

Example - IV

Similar results were obtained with (127, 64) code, once again bringing out the efficacy of the proposed method.

A (127, 64) binary BCH code with d_{min} = 21 and $t = \lfloor d_{min}/2 \rfloor$ = 10 is taken. The message 0065432100123456h has been encoded as {32a190,80091a2b,78e22e73,47b6ae9ch} and transmitted over an AWGN channel with σ = 0.7.

Details of the bits in an erroneously received word are reproduced in Table 9.

The received word Z = {122a190,a0380e2b,78e22672,47f62a9ch}; the error pattern e = {1100000,20311400,801,408400h};

K values in error: 17,21,37,62,8,26,11,14,13,16,30,6,25.

Table 9

Reliability magnitude, index and *s* values for Example - IV

S.No	Bit index *i*	m_i	*K*	*s* [*k*]
1	10	0.18	17	0.6626511
2	15	0.21	21	0.6183553
3	22	0.52	37	0.296632
4	32	0.81	62	0.1410521
5	43	0.12	8	0.760423
6	74	0.27	26	0.53813
7	76	0.14	11	0.726341
8	80	0.15	14	0.709859
9	84	0.15	13	0.709859
10	85	0.17	16	0.6779692
11	93	0.37	30	0.42560007
12	116	0.08	6	0.833181
13	120	0.25	25	0.5637412

Chase II and Chase III algorithms fail to decode this received word and the proposed Direct Target Codeword Identification Algorithm identifies the target codeword with the *f* value as 0.000347999 which is the transmitted codeword.

4 Simulation Results

Extensive simulations were carried out with (15, 7), (31, 16), and (127, 64) BCH codes using the proposed algorithm. Plots of Block Error Rate with conventional hard decision decoding, RLL decoding and Chase-2 decoding for about 1000 transmissions for (15,7) and (31,16) BCH codes are given in Figure 2 and Figure 3 respectively. For RLL the reliability index *k* is used as a threshold for error correction. In Figure 2 and Figure 3, a threshold value of 4 and 6 (= d_{min}-1) are used. However the performance of RLL can be improved by increasing the threshold (the error correcting capability). The same is shown in Figure 4 as plots of BLER versus threshold values (>4) for (15,7) code for some representative SNR values.

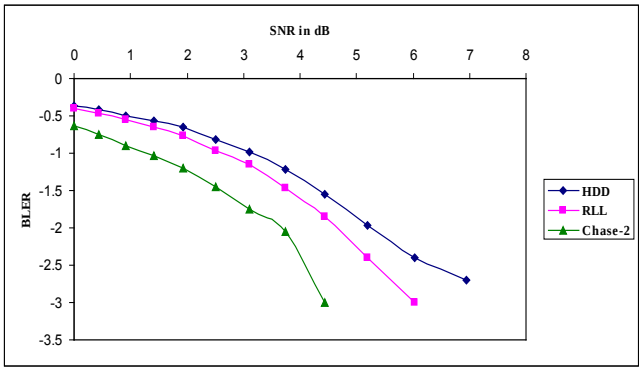

Figure 2. Comparision of Block Error Rate between RLL, HDD, Chase-2 for (15,7) BCH code

The numbers in the *y* axis represent the respective log value.

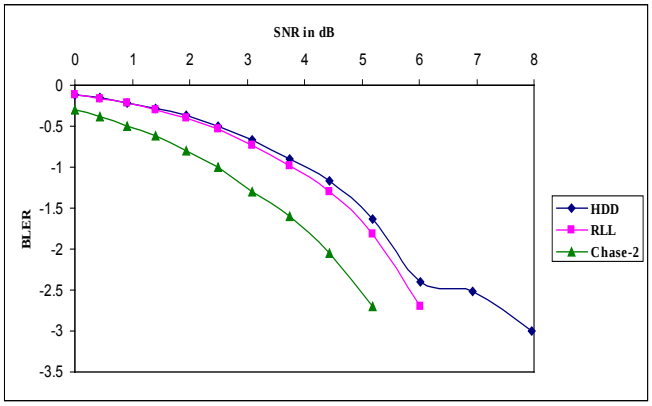

Figure 3. Comparision of Block Error Rate between RLL, HDD, Chase-2 for (31,16) BCH code

The numbers in the *y* axis represent the respective log value.

The implementation of Chase-2 calls for $2^{d\,min/2}$ [11] number of algebraic decodings for all values of SNR. With $2^{d\,min/2}$ algebraic decodings, the complexity of Chase-2 decoding is made orders higher by the Berlekamp Massey algorithm having multiplicative complexity O(t^2), and Chien search

requiring O(*tn*) multiplications [12].

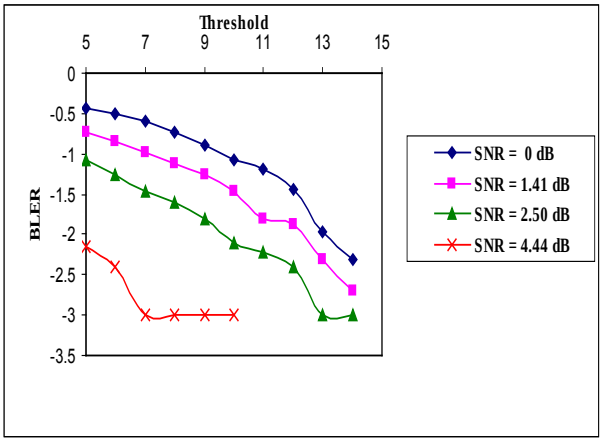

Figure 4. Plot of Block Error Rate versus Threshold value (>4) for (15,7) code

With RLL, the algebraic decoding is simplified to division by the generator polynomial. Since the evaluation of RLL entry differs with the noise level the RLL complexity varies with SNR. With this as the basis for complexity measure, the plot of average decoding complexity for (15,7) code for a representative threshold value of 4 is given in Figure 5. It can be noted that complexity here is substantially lower than that of implementation of algebraic decoding using say Berlekamp algorithm and Chien search [12]. A similar study has been carried out for different threshold values with other codes as well. The study shows that with increase in the threshold value for performance improvement, the complexity increases significantly at very low noise levels. Plots of average decoding complexity versus threshold (>4) for the (15,7) code at representative SNR values are shown in Figure 6.

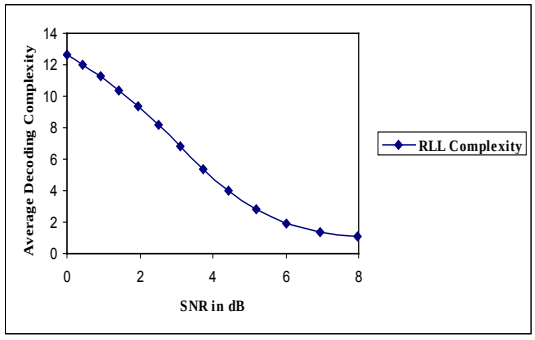

Figure 5. Complexity curve of (15,7) BCH code for a threshold value of 4

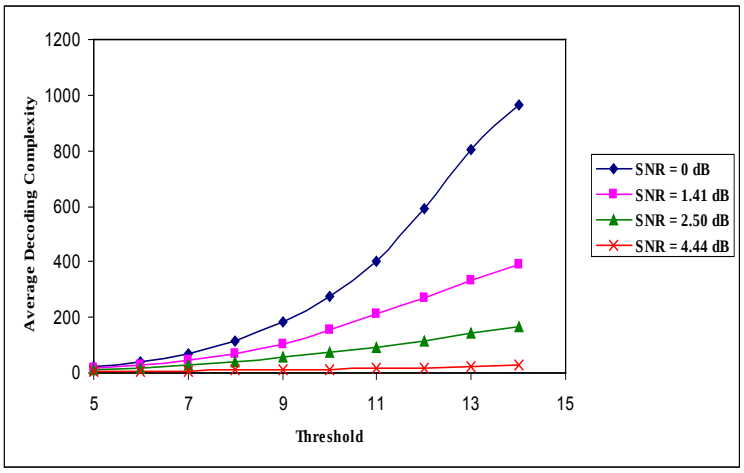

Figure 6. Average decoding complexity versus threshold (>4) for (15,7) BCH code

5. Conclusion

A new algorithm for soft decoding of binary codes is evolved in the paper. It uses soft information and achieves extended error correcting radius. The superiority of the algorithm lies in the number of errors and patterns of errors corrected even under low SNR conditions. The concept of the structured RLL – that has been introduced in the paper – is central to the proposed soft decoding algorithm; it enables identification of the *target codeword* with minimal search. The need for identifying a number of candidate codewords and selecting the most reliable one from amongst them – as with the class of Chase algorithms and their modified versions is obviated. Detailed simulations carried out bring out the effectiveness of the approach. Significantly the algorithm yields the best that is possible with SDD based approaches. The simulation results have been summarized in the paper and a few representative cases presented to illustrate the superiority of the approach.

All soft decision algorithms are equally effective at high SNR conditions and the decisive factor in the preference of one or another is the effectiveness under low SNR conditions. Communication schemes of recent interest which target low SNR channels can benefit from algorithms of the type presented here.

Non-binary BCH codes especially Reed Solomon Codes are in wide use either in stand alone or in concatenated schemes in various applications. After Guruswami and Sudan's novel algorithm [13] of list decoding of Reed Solomon codes many

significant developments [14] have come up. The seminal work of Kotter [15] has spurred interest in soft decision decoding strategies for decoding Reed Solomon Codes beyond the conventional decoding radius. In this scenario the extension of the concept of RLL and its use for soft decoding of non-binary codes can be of real potential.

References:

[1] G. D. Forney, Jr., Generalized Minimum Distance Decoding, *IEEE Transactions on Information Theory*, Vol. 12, 1966, pp. 125-131.

[2] David Chase, A Class of Algorithms for Decoding Block Codes With Channel Measurement Information, *IEEE Transactions on Information Theory*, Vol. 18, 1972, pp.170-182.

[3] Marc P. C. Fossorier, and Shu Lin, Soft-Decision Decoding of Linear Block Codes based on Ordered Statistics, *IEEE Transactions on Information Theory*, Vol. 41, 1995, pp. 1379-96.

[4] Jos.H.Weber, Low Complexity Chase-Like Bounded-Distance Decoding Algorithms *GLOBECOM*, 2003, pp.1608-1612.

[5] Idriss Chana, Hamid Allouch, and Mostafa Belkasmi, An Efficient New Soft Decision Decoding Algorithm for Binary Cyclic Codes, *International Conference on Multimedia Computing and Systems*, 2011.

[6] B.Yamuna, T.R.Padmanabhan, A Reliability Level List based SDD algorithm for binary Cyclic Block codes, *International Journal of Computers, Communication and Control*, Vol.7, No.2, 2012, pp. 388-395.

[7] Eran Fishler, Ofer Amrani, and Yair Be'ery, Geometrical and Performance Analysis of GMD and Chase Decoding Algorithms, *IEEE Transactions on Information Theory*, Vol. 45, 1999, pp.1406-1422.

[8] Jos H. Weber, Marc P.C. Fossorier, Limited-Trial Chase-Like Algorithms achieving Bounded-Distance Decoding, *IEEE Transactions on Information Theory*, Vol. 50, 2004, pp. 3318-3323.

[9] Marc P. C. Fossorier, and Shu Lin, Chase-Type and GMD Coset Decodings, *IEEE Transactions on Commuications*, Vol. 48, 2000, pp.345-350.

[10] WenyiJin and Marc.P.C.Fossorier, Reliability-Based Soft-Decision Decoding with Multiple Biase, *IEEE Transactions on Information Theory*, Vol. 53, 2007, pp. 105-120.

[11] Toshimitsu Kaneko, Toshihisa Nishijima,Hiroshige Inazumi and Shigeichi Hirasawa, An efficient Maximum- likelihood-Decoding algorithm for linear Block codes with algebraic decoder, *IEEE Transactions on Information Theory*, Vol. 40,No.2, 1994, pp. 320-327.

[12] Schipani D, Elia M, Rosenthal J, On the decoding complexity of cyclic codes up to BCH bound, *IEEE International Symposium on Information Theory Proceedings* (ISIT), 2011. pp. 835-839.

[13] Robert J. McEliece, The Guruswami-Sudan decoding algorithm for Reed-Solomon codes, *Interplanetary Network Progress Report* (NASA), 2003, pp. 42–135.

[14] Jing Jiang, Krishna R. Narayanan, Algebraic Soft-Decision Decoding of Reed–Solomon Codes Using Bit-Level Soft Information, *IEEE Transactions on Information Theory*, Vol. 54, pp, 2008, 3907-3928.

[15] Ralf Koetter, Alexander Vardy, Algebraic Soft-Decision Decoding of Reed–Solomon Codes, *IEEE Transactions on Information Theory*, Vol. 49, 2003, pp. 2809-2825.