Brock University

Department of Computer Science

A Comparative Study of Search Techniques Applied to the Minimum Distance Problem of BCH Codes

Joanna L. Wallis and Sheridan K. Houghten
Technical Report # CS-02-08
May 2002

To be presented at the International Conference on Artificial Intelligence and Soft Computing (ASC 2002), Banff, July 2002.

Brock University
Department of Computer Science
St. Catharines, Ontario
Canada L2S 3A1
www.cosc.brocku.ca

A COMPARATIVE STUDY OF SEARCH TECHNIQUES APPLIED TO THE MINIMUM DISTANCE PROBLEM OF BCH CODES

JOANNA L. WALLIS
Department of Mathematics
Simon Fraser University
Burnaby, BC, Canada V5A 1S6
jlwallis@sfu.ca

SHERIDAN K. HOUGHTEN
Department of Computer Science
Brock University
St. Catharines, ON, Canada L2S 3A1
houghten@cosc.brocku.ca

ABSTRACT

BCH codes have been shown to be excellent error-correcting codes among codes of short lengths. The error-correcting capability of a code is directly related to its minimum distance. It is computationally very difficult to determine the true minimum distance of BCH codes. We analyze and compare the behaviour of different heuristic search techniques when applied to the problem of finding the true minimum weight of BCH codes. A basic genetic algorithm significantly outperformed hill-climbing, tabu search and hybrid techniques.

KEY WORDS

Genetic Algorithms, Tabu Search, Hybrid Systems, Optimization, Error-Correcting Codes.

1 Introduction

In this paper we evaluate the use of different search techniques when applied to the problem of finding the true minimum weight of BCH codes. Our main goals with respect to this problem are to determine the suitability of such algorithms, and to ascertain which algorithms are most promising. Tabu search has been used with some success in solving this problem (see [1]); we compare this technique with several others. The search techniques we examined are hill-climbing, tabu search and a genetic algorithm, as well as two hybrids: a genetic algorithm with hill-climbing, and a genetic algorithm with tabu search. Such hybrid techniques have been shown to be useful for combinatorial problems [2,3].

In the interest of brevity, we shall discuss the basics of coding theory only to the extent that is necessary to understand the problem at hand. There exist several excellent books on error-correcting codes, including BCH codes; examples include [4] and [5]. More detail on this research in particular may be found in [6].

There are several characteristics that are considered desirable in an error-correcting code; among these are its ease of encoding and decoding, its rate, and the number of errors that it can correct. The rate of a code is k/n, where k is the size of its information set and n is its length. The information set of a code is k, the number of digits that represent the actual information being transmitted; the remaining $n-k$ digits provide redundancy information and enable us to recognize a *codeword*, or vector in the code. The number of errors a code can correct is denoted by t. There is a trade-off between the value of t and the rate of the code: to increase the number of errors, more redundancy must be added to the information set, thus decreasing the rate.

The *weight* of a codeword is the number of nonzero elements in the codeword. The minimum weight of a code is the weight of the codeword with the lowest non-zero weight. The *distance* between two codewords is the number of elements in which they differ. The *minimum distance* of a code is the lowest distance seen between any two codewords. A code is *linear* if for any two codewords in the code, their sum is also a codeword. In a linear code, minimum weight is equivalent to minimum distance. For a given code, the number of errors that can be corrected, t, is related to its minimum distance, d, by the equation $d = 2t+1$.

Discovered by R.C. Bose and D.K. Ray-Chaudhuri, and independently by A. Hocquenghem, BCH codes are considered to be one of the best known codes for small values of n (up to lengths of several thousands [4]) according to the criteria outlined above. BCH codes are a type of fixed length, cyclic code. Such codes are linear. Additionally, if $c=c_1c_2c_3...c_{n-1}$ is a codeword, then any cyclic shift of c is also a codeword (thus, for example, $c_2c_3....c_{n-1}c_1$ is a codeword).

For the purposes of this paper, the length of the BCH code will be restricted to $n = q^m-1$ where q is prime, and m is an integer. This type of BCH code is known as a primitive BCH code. Also, all codes considered in this research are binary codes, but generalizations to non-binary BCH codes can be made [6].

Associated with each BCH code is a designed distance and a minimum distance. The designed distance δ is the lower bound of the true minimum distance d, and is used

to construct the code. Formally, for any binary BCH code of length n and designed distance δ, we have $d \geq \delta$. A code with designed distance $\delta=2t+1$ can correct at least t errors. If, however, we have $d > \delta$, then the code can correct more errors than it was designed to correct.

There are several mathematical theorems that can prove that $\delta = d$ if certain conditions hold [4,7]. The problem of determining the necessary and sufficient conditions on n and δ required to ensure that $d = \delta$, for all n and δ, remains unsolved. The problem of determining the exact value of d when $d > \delta$ is an even more difficult problem and is also unsolved.

Since, to date, these problems cannot be solved mathematically, it becomes necessary to physically search the codewords of a code in order to find the codeword with the minimum weight. Unfortunately, as the size of n increases, the size of the search space becomes prohibitively large. The number of information bits in a code, k, defines the size of a search space. Note that k is also the number of basis vectors in the code and thus the size of the search space is 2^k. For codes as short as $n = 63$, the search space can be as large as 2^{57}. Thus, an exhaustive search is not feasible, but a heuristic search may provide valuable information and, in some cases, perhaps a solution. We evaluate several different heuristic search techniques when applied to the problem of determining the true minimum weight of a BCH code.

The search problem is not limited to finding the exact minimum distance of a code. Mathematically, the upper bound of any binary BCH code is given by $2\delta-1$ [4]. If the various search techniques cannot find d, they could at least improve the upper bound of the code.

The remainder of this paper is organized as follows. Section 2 briefly describes how we represent BCH codes. Section 3 describes the specific details of the implementation of the search techniques considered. Section 4 summarizes the experiment results. Finally, the conclusions and possible future directions of this research are outlined in section 5.

2 Representation of BCH Codes

The length and designed distance define a particular instance of a BCH code. Once these two attributes have been selected, the generation of the code is purely computational. BCH(n,δ,k) denotes a binary BCH code of length n, designed distance δ and information bits k. Since these codes are linear, they can be represented by a *generator matrix* – a $k \times n$ matrix that contains k linearly independent basis vectors. The codewords correspond to the linear combinations of the vectors in the generator matrix.

For each of the experiments performed for this research, we used Maple to construct the generator matrices of the BCH codes being considered. Since BCH codes are linear, every codeword can be expressed as a linear combination of its basis vectors [8]. Thus, following [1], a codeword $c \in$ BCH(n,δ,k) can be represented by the coefficients of the basis vectors b_i, i = 1,2,3,...,k. So a solution vector $s=\{s_i\}$ for i = 1,2,3,...,k and s_i = {0,1}, represents the codeword $\Sigma(s_i \times b_i)$ where multiplication and addition are bitwise.

For example, consider BCH(7,3,4). For this code we may select the basis vectors b_1=1000101, b_2=0100111, b_3=0010110, and b_4=0001011. The solution vector s = 1001 represents the codeword $c = 1\times(1000101) + 0\times(0100111) + 0\times(0010110) + 1\times(0001011) = 1000101+0001011 = 1001110$.

All five search algorithms use this indirect representation of the codewords, although it is labelled as a chromosome in the genetic algorithms, as is the custom, rather than as a vector. The weight, or fitness, of a codeword is calculated by converting a solution vector v into its corresponding codeword, and counting the number of nonzero elements.

3 Implementation

We study and compare the behaviour of several search techniques when applied to the minimum distance problem of BCH codes. For each of the search techniques considered, we used 20 test cases ranging from BCH(127,21,64) to BCH(511,183,58).

The hill-climbing algorithm and the tabu search both implement neighbourhood searches. In both of these techniques, a neighbourhood is defined as all vectors at a distance of 1 from the current vector. Hence, if s is the current solution, then $N(s) = \{s' \mid d(s,s') = 1\}$ where $d(a,b)$ is the distance between vectors a and b. For the hill-climbing algorithm $|N(s)| = k$, and for the tabu search $|N(s)| \leq k$. Both of these neighbourhoods are relatively small since $k < n$ for all BCH codes.

3.1 Hill-Climbing

We used two separate strategies for selecting the initial solution s. The first strategy was to set s to the vector $s = \{s_1=1, s_i=0 \ \forall \ 1 < i \leq k\}$ as is outlined in [1]. The second strategy was to randomly select initial solutions for s. This strategy was performed ten times taking the best results for each test case.

3.2 Tabu Search

We used three different strategies in the implementation of the tabu-search. The first strategy uses the fixed initial point $s = \{s_1=1, s_i=0 \ \forall \ 1 < i \leq k\}$. The

second strategy randomly selects initial solutions for s. This strategy was also performed ten times, taking the best results for each test case.

The final strategy uses intensification. Backtracking is used to enhance the basic tabu search in order to obtain better results [1]. After every p steps, the tabu search returns to the best solution so far, and continues the solution from that point. The tabu list is unchanged, so the search is forced into a new path since it is restricted from visiting any solution it has already been to. This strategy is implemented from the fixed initial solution $s = \{s_1=1, s_i=0 \ \forall \ 1 < i \leq k\}$, with p set to 3 and 10.

For all three strategies, the tabu list was implemented using recency-based memory. Solutions are entered into the list as they are visited. Each strategy was run for 100 steps, and the size of the tabu list was set to 100, so no cycling was permitted.

3.3 Genetic Algorithm

We implemented the genetic algorithm using tournament selection to randomly select 3 members of the population. Crossover is implemented using 2-point crossover. Two different points are selected on a chromosome, segmenting the chromosome into three parts. The centre part of the two parent chromosomes is then swapped, creating two new offspring.

Mutation is performed by selecting a mutation rate, m_r between 0 and 1. Each individual gene on the chromosome is tested for mutation, and has a probability of m_r of being mutated.

We ran the genetic algorithm with a population size of 1000 and with a population size of 10000. All other parameters remained constant and are as follows:

Parameter	Value
Probability of Crossover	80%
Probability of Mutation	10%
Probability of Replication	10%
Crossover Type	2-point
Mutation Rate	30%
Selection Type	Tournament
Tournament Size	3
Maximum Number of Generations	75

Each instance was run ten times, and the results were averaged over the ten runs.

3.4 Optimized Genetic Algorithm

We optimized the genetic algorithm using two different techniques: the hill-climbing search and the tabu search [9]. Both were implemented as hybrid techniques. The local search was applied at every generation. Before each new individual was placed into the new population, the optimization technique was performed on it.

In the genetic algorithm with hill-climbing, the hill-climbing algorithm was performed with no restrictions. That is, the climber was allowed to climb until it could go no further.

In the genetic algorithm with tabu search, the tabu search was allowed to perform ten steps. A new tabu list of length ten was created for each new chromosome, so the path followed by one chromosome had no effect on the path taken by another.

Once again, each technique was run ten times, and the results were averaged.

4 Experiment Results

The tabu search and the hill-climbing algorithms were compared first for their ability to find a solution, and second for the number of searches required to find that solution (see Table 1). In this table, Opt denotes the optimum result found during the run and M_{opt} denotes the number of steps taken to find the optimal solution. A step is a full search of the neighbourhood of s, and constitutes a search of approximately $M_{opt} \times k$ solutions. The shaded areas show the best solution for the test case.

For both criteria, the tabu search proved the more effective algorithm. In only one case, namely BCH(511,91,184), did the hill-climber outperform the basic tabu search, but it did not outperform either of the tabu searches with the added backtracking feature.

The basic tabu search found its solution in the least number of searched solutions on average. However, in most cases, it found the solution within five steps of the original solution, failing to take advantage of the full running time of the algorithm (100 steps). This may be due to the fact that the initial solution is already good, since this characteristic is not true of the basic tabu search from a random initial point.

The tabu searches with the added backtracking facility obtained the best results as far as solution found, but they took the longest in terms of solutions searched. This technique did not appear to have the same difficulty in finding improving solutions as the basic tabu search despite also starting at a good initial solution.

The above discussion, unless explicitly stated, did not include the results of either search from a random start point. The results of these runs proved unimpressive, in some cases even failing to improve on the upper bound. They are included in the table for completeness only.

Table 1: Hill-Climbing and Tabu Search – Number of Searches Required to Find a Solution

BCH(n,δ,k)	Upper Bound	Hill-Climbing				Tabu Search							
		Fixed Initial Point		Random Initial Point		Fixed Initial Point		Backtrack at 3		Backtrack at 10		Random Initial Point	
		Opt	M_{opt}	Opt	M_{opt}	Opt	M_{opt}	Opt	M_{opt}	Opt	M_{opt}	Opt	M_{opt}
(127,21,64)	41	28	772	44	775	24	2	22	59	22	35	22	60
(127,23,57)	45	28	865	47	865	23	2	23	2	23	2	27	36
(127,27,50)	53	32	982	51	987	31	5	28	9	31	5	28	64
(255,43,115)	85	57	435	109	436	55	3	50	71	55	3	76	34
(255,45,107)	89	64	468	105	468	62	3	58	41	59	22	77	22
(255,47,99)	93	64	501	93	504	61	4	59	42	61	4	77	48
(255,51,91)	101	72	548	108	546	69	3	64	14	64	32	77	11
(255,53,87)	105	70	569	105	572	66	7	65	80	65	82	74	24
(255,55,79)	109	74	627	106	627	64	6	69	65	64	6	80	25
(255,59,71)	117	79	698	98	699	79	3	73	21	69	74	82	17
(511,51,304)	101	90	170	225	167	85	5	83	62	85	5	178	16
(511,55,286)	109	96	176	226	179	92	4	87	5	92	4	182	15
(511,75,238)	149	118	214	226	215	112	2	111	45	112	2	183	19
(511,79,220)	157	123	230	222	230	117	2	116	56	117	2	187	18
(511,91,184)	181	135	273	224	274	140	3	132	36	139	24	180	24
(511,95,166)	189	152	303	224	307	140	7	147	24	140	7	187	27
(511,117,121)	233	163	413	227	413	163	5	163	5	163	5	192	26
(511,123,103)	245	179	487	216	486	179	4	175	6	175	15	199	15
(511,171,76)	341	195	654	227	656	184	15	180	29	187	72	199	33
(511,183,58)	365	207	849	224	851	199	8	191	96	192	47	200	34

Table 2 shows the optimal results for all six algorithms. As before, the shaded areas of this table show the best solution for the test case. The genetic algorithms obtain the best results, which is to be expected since they searched far more solutions. The basic genetic algorithm, with a population of 1 000, searches 75 000 solutions over 75 generations. The tabu search and hill-climbing algorithms search at most 20 000 solutions.

The genetic algorithms were also evaluated based on their fitness and on the number of solutions considered. Interestingly, the basic genetic algorithm with a population of 1 000 performed comparably well with the two optimization techniques, despite a significant disadvantage in the number of solutions searched. A second basic genetic algorithm with a population of 10 000 was added to the study in order to make up for this discrepancy. Surprisingly, this algorithm outperformed both optimization techniques.

Unlike their stand-alone counterparts, the genetic algorithm with hill-climbing outperformed the genetic algorithm with tabu search, although the results from both algorithms proved disappointing. In two test cases, namely BCH(511,51,304) and BCH(511,55,286), these two algorithms failed to improve even the upper bound of the BCH codes within the allotted 1 000 000 searches.

5 Conclusion

BCH codes have been shown to be excellent error-correcting codes among codes of short lengths. They are simple to encode and relatively simple to decode. Due to these qualities, there is much interest in the exact capabilities of these codes. Of greatest importance is the exact minimum distance of these codes. Much study has gone into mathematically and computationally solving the minimum distance problem for each BCH code.

Modelled after a paper by Bland and Baylis [1], a tabu search was implemented, and tested on 20 test cases, including the 10 used in the cited paper. Despite its poor performance as an optimization technique for a genetic algorithm in this research, the tabu search performed well as a stand-alone algorithm. Its performance was further enhanced by the simple intensification technique, outlined in [1]. This research was able to further improve the results given in [1] by setting the backtracking to 3.

Table 2: Optimum Results for all Algorithms

BCH(n,d,k)	Hill-Climbing F.I.P	Tabu F.I.P	Tabu Backtrack at 3	GA Population 1000	GA Population 10000	GA – Hill	GA - Tabu
(127,21,64)	28	24	22	21	21	21	21
(127,23,57)	28	23	23	23	23	23	23
(127,27,50)	32	31	28	27	27	27	27
(255,43,115)	57	55	50	50	50	51	52
(255,45,107)	64	62	58	56	51	55	56
(255,47,99)	64	61	59	59	55	59	59
(255,51,91)	72	69	64	62	59	61	63
(255,53,87)	70	66	65	63	57	62	65
(255,55,79)	74	64	69	65	60	67	66
(255,59,71)	79	79	73	70	66	67	72
(511,51,304)	90	85	83	94	79	94	91
(511,55,286)	96	92	87	100	84	96	97
(511,75,238)	118	112	111	117	105	117	113
(511,79,220)	123	117	116	122	111	119	121
(511,91,184)	135	140	132	137	128	133	134
(511,95,166)	152	140	147	145	137	140	143
(511,117,121)	163	163	163	162	152	159	160
(511,123,103)	179	179	175	169	164	168	170
(511,171,76)	195	184	180	181	176	181	181
(511,183,58)	207	199	191	189	185	190	189

Surprising results were seen with the basic genetic algorithms, which performed well above expectations, significantly outperforming the other search techniques. The optimization techniques however proved disappointing. Early convergence did not prove to be a problem with these techniques, as is often feared when using optimization techniques, but they also failed to result in any noticeable improvement in the generation average, relative to the number of searches performed. The disappointing results of the hybrids may actually have been predicted by the performance of their stand-alone counterparts when the initial solution was taken to be random. The local searches perform well when the initial solution is good. By performing this optimization on every member of every generation, potential benefits may have been diluted by wasted effort on solutions that were not good.

Given the successful results of the genetic algorithm, it would be interesting to further explore various potential optimizations of the genetic algorithm. A significant improvement is seen from a simple increase in population size; other parameter changes may also result in significant improvements.

Some analysis is needed in order to discover why the genetic algorithm optimization techniques failed to show any improvements. Applying the local search less often, or only to individuals with a high fitness, may improve the results seen in this research. Applying the local search less often may also allow the addition of an intensification procedure to the tabu search.

It may be useful to apply some of the search techniques outlined in this paper to other types of BCH codes, since only primitive BCH codes were considered.

6 Acknowledgements

This research was supported by the Natural Sciences and Engineering Research Council of Canada.

The authors gratefully acknowledge the suggestions and assistance given by Brian Ross of the Department of Computer Science, Brock University.

References

1. Bland, J.A. & Baylis, D.J. A Tabu Search Approach to the Minimum Distance of Error-Correcting Codes. *International Journal of Electronics*, *79*(6), 1995, 829-837.

2. Hart, W.E. & Belew, R.K., Optimization with Genetic Algorithm Hybrids that use local search, in

R.K. Belew & M. Mitchell (Eds.), *Adaptive Individuals in Evolving Populations: Models and Algorithms* (Reading, MA: Addison-Wesley, 1996), 483-496.

3. Magyar, G., Johnsson, M. & Nevalainen, O., An Adaptive Hybrid Genetic Algorithm for the Three-Matching Problem, *IEEE Transactions on Evolutionary Computation*, *4*(2), 2000, 135-146.

4. MacWilliams, F.J. & Sloane, N.J.A., *The Theory of Error-Correcting Codes* (New York: North-Holland, 1977).

5. Pless, V., *Introduction to the Theory of Error-Correcting Codes,* 2nd edition, (New York: John Wiley & Sons, 1989).

6. Wallis, J.L. *A comparative study of search techniques on the minimum distance of BCH codes* (St. Catharines, ON: Brock University Honours Thesis, 2001).

7. Pless, V.S., & Huffman, W.C. (Eds.), *Handbook of Coding Theory*, (Amsterdam: Elsevier, 1998).

8. Spence, L.E., Insel, A.J. & Friedberg, S.H., *Elementary Linear Algebra: A Matrix Approach* (New Jersey: Prentice Hall Inc, 2000).

9. Glover, F. & Laguana, M., *Tabu Search* (Boston: Kluwer Academic Publishers, 1997).