TABLE II
NEW OPTIMUM GOLOMB RULERS.

Marks	Ruler
17	0,8,31,34,40,61,77,99,118,119,132,143,147,182,192,194,199
18	0,11,24,28,49,63,68,86,118,127,133,134,160,163,194,206,214,216
19	0,4,13,15,42,56,59,77,93,116,126,138,146,174,214,221,240,245,246

IV. CONCLUSION

The parallel implementation of the Shift Algorithm has been successful in identifying three previously unpublished Optimum Golomb Rulers, and is currently searching for the 20 mark Golomb Ruler.

An important research result is the demonstration of reliable parallel computation for extended execution times. Fault tolerance and restartability issues are addressed and have been shown to be effective in assuring processing reliability for program runs in excess of a month. Finally, previously unpublished results were found and verified against independent findings from another researcher, O. Sibert.

The sequential and parallel codes for the evaluation of Golomb rulers are available via anonymous ftp from the directory `/usr/ftp/pub/golomb` of the site `macedonia.mhl.tuc.gr`.

ACKNOWLEDGMENT

The authors wish to thank the Duke University Department of Electrical Engineering for providing resources and support for this research. Computational resources were further provided from the use of the idle cycles of a workstation cluster funded under the NSF under Grant DUE-9351480. In addition, the authors would like to extend their appreciation to E. Angerer for her help in implementation of the parallel algorithm, to A. Singh for his help at the early stages of the parallel runs, to M. Garry for his corrections regarding the scope of the midpoint reduction technique, and to A. Emmanouelides and E. Sotiriades for their review of the computer codes prior to their distribution.

REFERENCES

[1] W. C. Babcock, "Intermodulation interference in radio systems," *Bell Syst. Tech. J.*, pp. 63–73, Jan. 1953.
[2] G. S. Bloom and S. W. Golomb, "Applications of numbered unidirected graphs," *Proc. IEEE*, vol. 65, pp. 562–570, Apr. 1977.
[3] A. K. Dewdney, "Computer recreations," *Scient. Amer.*, pp. 16–26, Dec. 1985.
[4] A. K. Dewdney, "Computer recreations," *Scient. Amer.*, pp. 14–21, Mar. 1986.
[5] E. M. Reingold, J. Nievergelt, and N. Deo, *Combinatorial Algorithms: Theory and Practice*. Englewood Cliffs, NJ: Prentice-Hall, 1977.
[6] Geist *et al.*, *PVM 3 User's Guide and Reference Manual*, May 1993.
[7] T. Kløve, "Bounds and construction for difference triangle sets," *IEEE Trans. Inform. Theory*, vol. 35, pp. 879–886, July 1989.
[8] G. D. Martin, "Optimal convoluting self-orthogonal codes with an application to digital radio," in *Proc. IEEE Int. Conf. on Communication*, 1985, pp. 1249–1253.
[9] A. T. Moffet, "Minimum-redundancy linear arrays," *IEEE Trans. Antennas Propogat.*, vol. AP-16, pp. 172–175, Mar. 1968.
[10] J. Robbins, R. Gagliardi, and H. Taylor, "Acquisition sequences in PPM communications," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 738–744, Sept. 1987.
[11] J. P. Robinson and A. J. Bernstein, "A class of binary recurrent codes with limited error propogation," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 106–113, Jan. 1967.
[12] J. B. Shearer, "Some new optimum Golomb rulers," *IEEE Trans. Inform. Theory*, vol. 36, pp. 183–184, Jan. 1990.
[13] W. O. Sibert, in a letter to H. Taylor and S. Golomb, Apr. 1993.

Greedy and Heuristic Algorithms for Codes and Colorings

Tuvi Etzion, *Member, IEEE*, and Patric R. J. Östergård, *Member, IEEE*

***Abstract*— Many of the fundamental coding problems can be represented as graph problems. These problems are often intrinsically difficult and unsolved even if the code length is relatively small. With the motivation to improve lower bounds on the sizes of constant weight codes and asymmetric codes, we suggest a few greedy algorithms and other heuristic methods, which result in new, record-breaking codes. Some of the heuristics used are based on tabu search and evolutionary algorithms. Tables of new codes are presented.**

***Index Terms*—Asymmetric code, coloring, constant weight code, evolutionary algorithm, tabu search.**

I. INTRODUCTION

Many coding problems belong to the following general class of problems. Given a finite space F, a discrete metric M, and a positive integer d, find the largest subset of F, C, such that the minimum distance of C in the metric M is d, i.e., for any two distinct elements $c_1, c_2 \in C$ the distance between c_1 and c_2 in the metric M is at least d. This coding problem can be formulated in several ways in terms of graph theory.

Manuscript received June 9, 1996; revised May 19, 1997. This work was supported in part by the fund of promotion of sponsored research at the Technion and by the Academy of Finland.

T. Etzion is with the Computer Science Department, Technion–Israel Institute of Technology, Haifa 32000, Israel.
P. R. J. Östergård is with the Department of Computer Science and Engineering, Helsinki University of Technology, P.O. Box 1100, 02015 HUT, Finland.

Publisher Item Identifier S 0018-9448(98)00024-8.

We generate the graph $G = (V, E)$, where the set of vertices V represent the elements of the space F. The edges can be generated in two different ways. Either two vertices $v_1, v_2 \in V$ are connected by an undirected edge $\{v_1, v_2\}$ if the distance between v_1 and v_2 in the metric M is *less than* d, or, in the second approach, they are connected if the distance between them is *at least* d. (Occasionally, a third graph representation, where vertices at distance 1 from each other are connected, is useful.) Clearly, the graphs obtained in the two approaches are complements of each other. A set of vertices is an *independent set* (IS) in G if its elements are pairwise nonadjacent. A set of vertices is a *clique* in G if its elements are pairwise adjacent. With the former approach we want to find an IS of maximum size, and with the latter approach a clique of maximum size. The difficulties of these problems are indicated by the fact that the general problem of finding an IS or clique of a given size is NP-complete [11].

Coloring problems constitute another class of problems that are computationally hard to solve. One such problem is the chromatic number problem: Given a graph $G = (V, E)$, find the smallest integer k, such that the vertices of the graph can be colored in k colors, in such a way that adjacent vertices are colored by distinct colors. This problem is known to be NP-hard [11]. Note that each set of vertices that are colored with the same color forms an IS.

In the literature, various approximation methods have been suggested for these problems. Unfortunately, they are often only of theoretical interest and do not perform well enough in practice. Practical performance is of utmost interest, since these problems have found several important applications in areas including *register allocation* [5] and *VLSI testing* [12], only to mention a few. Recently, stochastic heuristics, such as simulated annealing and tabu search, have been used to obtained good colorings, see [14] and [16]. Here we develop that approach further using tabu search and greedy methods combined with ideas from evolutionary algorithms.

In this correspondence we discuss two types of codes: constant weight codes with minimum Hamming distance 4, and codes with minimum asymmetric distance 2. We then get new lower bounds on $A(n, 4, w)$, the maximum cardinality of a code of length n, constant weight w, and minimum distance 4, and on $A_a(n, 2)$, the maximum cardinality of a code of length n and minimum asymmetric distance 2. For earlier results on these two functions see [4] and [6], [9], respectively. We shall here discuss heuristic approaches for IS problems and for coloring problems. These approaches result in good (in the sense that earlier results in the literature are improved) codes of the above mentioned types. Extensive computer searches were carried out during this research.

The rest of this correspondence is organized as follows. In Section II we review partitioning methods for constant weight codes and asymmetric codes. Such partitionings are our main motivation for the coloring problem. The version that we present is an improved version of the partitioning method for asymmetric codes given in [6]. In Section III we briefly discuss the main heuristics used, namely, tabu search, evolutionary algorithms, and greedy algorithms. Application of these heuristics to our problems is considered in Sections IV and V, where we also consider some tailored methods that can be applied to further improve the outcomes of the algorithms. Section IV is devoted to packing and Section V to coloring. In Section VI we present tables with new bounds on the size of constant weight codes with minimum Hamming distance 4, and on the size of codes with minimum asymmetric distance 2. The new codes and colorings are published separately in a report [8] and on a WWW page.

II. Definitions and the Partitioning Method

As indicated in the Introduction, research on the coloring problem was motivated by the partitioning method for constant weight codes and for asymmetric codes. In this section we review this method, and we also present an improved version of the partitioning method for asymmetric codes. First, we define two metrics.

Let $u = (u_1, u_2, \cdots, u_n)$ and $v = (v_1, v_2, \cdots, v_n)$ be two binary vectors of length n. Let $N(u, v)$ denote the number of coordinates where u is 0 and v is 1. The Hamming distance is defined by

$$d_h(u, v) = N(u, v) + N(v, u)$$

and the asymmetric distance by

$$d_a(u, v) = \max\{N(u, v), N(v, u)\}.$$

A. Constant Weight Codes

The partitioning method for constant weight codes was first presented in [19]; see also [4], the notations of which we use here. A *partition* $\Pi(n, w) = (X_1, \cdots, X_m)$ of the $\binom{n}{w}$ vectors of weight w is a collection of subcodes X_i with minimum Hamming distance 4. The vector $\pi(n, w) = (|X_1|, \cdots, |X_m|)$ is the *index vector* of the partition $\Pi(n, w)$, and

$$\pi(n, w) \cdot \pi(n, w) = \sum_{i=1}^{m} |X_i|^2$$

is its *norm*. We always assume $|X_1| \geq |X_2| \geq \cdots \geq |X_m|$. When there are several different partitions available for a given n and w we often denote them by $\Pi_i(n, w)$, $i \geq 1$, and the corresponding index vectors by $\pi_i(n, w)$, $i \geq 1$.

The *direct product* $\Pi(n_1, w_1) \times \Pi(n_2, w_2)$ of two partitions $\Pi(n_1, w_1) = (X_1, \cdots, X_{m_1})$, $\Pi(n_2, w_2) = (Y_1, \cdots, Y_{m_2})$ consists of the vectors

$$\{(u, v):\ u \in X_i,\ v \in Y_i,\ 1 \leq i \leq m\}$$

where $m = \min\{m_1, m_2\}$. This code has length $n_1 + n_2$, minimum Hamming distance 4, weight $w_1 + w_2$, and cardinality

$$\pi(n_1, w_1) \cdot \pi(n_2, w_2) = \sum_{i=1}^{m} |X_i||Y_i|.$$

The Construction: To obtain a code of length n, minimum Hamming distance 4, and weight w by the partitioning method we write $n = n_1 + n_2$, choose $\epsilon = 0$ or 1, and take the union of the direct products

$$\begin{gathered}\Pi(n_1, \epsilon) \times \Pi(n_2, w - \epsilon)\\ \Pi(n_1, \epsilon + 2) \times \Pi(n_2, w - \epsilon - 2)\\ \Pi(n_1, \epsilon + 4) \times \Pi(n_2, w - \epsilon - 4)\\ \vdots\end{gathered}$$

to get a code with the required properties and

$$\pi(n_1, \epsilon) \cdot \pi(n_2, w - \epsilon) + \pi(n_1, \epsilon + 2) \cdot \pi(n_2, w - \epsilon - 2) + \pi(n_1, \epsilon + 4) \cdot \pi(n_2, w - \epsilon - 4) + \cdots$$

codewords. For examples how the construction is applied to obtain specific codes the reader is referred to [4] and [19]. Codes obtained from this construction will be indicated later by pi. The values of n_1, n_2, ϵ are as follows:

- for type $p0$, $n_1 = \lfloor \frac{n}{2} \rfloor$, $n_2 = n - n_1$, $\epsilon = 0$,
- for type $p1$, $n_1 = \lfloor \frac{n}{2} \rfloor$, $n_2 = n - n_1$, $\epsilon = 1$,
- for type $p2$, $n_1 = \lfloor \frac{n}{2} \rfloor - 1$, $n_2 = n - n_1$, $\epsilon = 0$.

We next discuss the choice for a good partition. We say that one partition $\Pi_1(n_1, w_1)$ *dominates* another $\Pi_2(n_1, w_1)$ if

$$\pi_1(n_1, w_1) \cdot \pi(n_2, w_2) \geq \pi_2(n_1, w_1) \cdot \pi(n_2, w_2)$$

holds for all choices of n_2, w_2, and all possible index vectors $\pi(n_2, w_2)$. If a partition is dominated it never produces best known codes in the construction. The concept of domination induces a partial ordering on partitions with the same values of n and w. A partition $\Pi(n, w)$ is *optimal* if it dominates all other partitions or just *maximal* if it is not itself dominated by any other partition.

Many best known partitions and the lower bounds obtained from them are given in [4] and [7]. In [7] there is also a method for improving on the codes obtained by the partitioning method, by using certain types of partitions.

B. Asymmetric Codes

The definitions and the construction for asymmetric codes are similar to those for constant weight codes. These are reviewed for completeness. Again, we want to remind that the construction here is an improved version of the one in [6], but the proof of its correctness is analogous and is thus omitted.

A *partition* $\Pi(n) = (X_1, \cdots, X_m)$ of the 2^n vectors of length n is a collection of subcodes X_i with minimum asymmetric distance 2. The vector $\pi(n) = (|X_1|, \cdots, |X_m|)$ is the *index vector* of the partition $\Pi(n)$, and

$$\pi(n) \cdot \pi(n) = \sum_{i=1}^{m} |X_i|^2$$

is its *norm*. We always assume $|X_1| \geq |X_2| \geq \cdots \geq |X_m|$. When there are several different partitions available for a given n and w we often denote them by $\Pi_i(n)$, $i \geq 1$, and the corresponding index vectors by $\pi_i(n)$, $i \geq 1$. The *direct product* $\Pi(n_1) \times \Pi(n_2, w)$ of two partitions, one partition of asymmetric codes and the other of constant weight codes, $\Pi(n_1) = (X_1, \cdots, X_{m_1})$, $\Pi(n_2, w) = (Y_1, \cdots, Y_{m_2})$ consists of the vectors

$$\{(u, v): \ u \in X_i, \ v \in Y_i, \ 1 \leq i \leq m\}$$

where $m = \min\{m_1, m_2\}$. This code has length $n_1 + n_2$, minimum asymmetric distance 2, and cardinality

$$\pi(n_1) \cdot \pi(n_2, w) = \sum_{i=1}^{m} |X_i||Y_i|.$$

The Construction: To obtain a code of length n and minimum asymmetric distance 2 by the partitioning method we write $n = n_1 + n_2$, choose $\epsilon = 0$ or 1, and take the union of the direct products

$$\begin{gathered}\Pi_1(n_1) \times \Pi(n_2, \epsilon)\\ \Pi_2(n_1) \times \Pi(n_2, \epsilon + 2)\\ \Pi_3(n_1) \times \Pi(n_2, \epsilon + 4)\\ \vdots\end{gathered}$$

to get a code with the required properties and

$$\pi_1(n_1) \cdot \pi(n_2, \epsilon) + \pi_2(n_1) \cdot \pi(n_2, \epsilon + 2) + \pi_3(n_1) \cdot \pi(n_2, \epsilon + 4) + \cdots$$

codewords. For examples how the construction is applied to obtain specific codes the reader is referred to [6]. In all cases here where this construction gives the best known code, $\epsilon = 0$ is used.

The definitions for domination and for maximal and optimal partitions are analogous to those for constant weight codes.

III. Heuristics Used

Throughout the years, a wide variety of heuristics have been developed to attack problems that cannot be solved within reasonable time using deterministic algorithms. Such methods have also been applied successfully to many problems in coding theory; see [15] for a recent survey.

In this section we shall briefly discuss some general heuristics used in this work, namely, tabu search, evolutionary algorithms, and greedy algorithms. These can be applied when the problems have been defined in terms of combinatorial optimization. The function that is optimized is called the *cost function*.

Tabu search [13] is a so-called local search method. For each (feasible) solution of our optimization problem, we define a *neighborhood* consisting of solutions that in some sense differ only slightly from the original solution. The search is now a walk in the solution space, where the next solution is the best solution in the neighborhood of the previous solution. To avoid looping and getting stuck in local optima, reverses of recent moves are forbidden. (Attributes of) recent moves are stored in a *tabu list*, which gives the method its name.

For an in-depth treatment and for information on more advanced issues of the tabu search heuristic, see [13]. To be able to apply this heuristic to our problems, we must define the concepts of solution, cost function, and neighborhood. We shall have a closer look at this in the next section.

The tabu search algorithm is essentially sequential: only one current solution is kept in memory. *Evolutionary algorithms* [1], on the other hand, constitute a large family of parallel algorithms: the search goes from a set of solutions to another set of solutions (clearly this can also be done on a regular, nonparallel computer, so we do not have to run these programs on parallel computer systems).

In evolutionary algorithms several, say p, search processes are carried out simultaneously. These processes interact periodically. Different strategies exist for choosing the set of solutions from which the processes continue after an interaction. In our work, we only borrow some of the numerous features of evolutionary algorithms. When the search processes interact, a new set of solutions is chosen from a subset of the best among the current solutions (so this is a so-called *elitist* strategy [1, p. 97]).

A third concept that we shall mention is that of *greedy algorithms*, which are here used in combination with genetic algorithms. A greedy algorithm iteratively makes the best possible choice according to some criterion. Thus also the tabu search algorithm is in some sense greedy. A lot of different algorithms fall under the framework of greedy algorithms; in the following sections we shall give specific details about the greedy heuristics used in this work.

In addition to these general heuristics, many tailored heuristics have also been used in the search for codes and colorings. These will be discussed in the following sections. In many cases, the new codes were found by combining several of the methods to be presented.

IV. Searching for Codes

In the two types of codes discussed in this correspondence, we get the following graphs, in which we want to find IS's. For constant weight codes, the graph consists of $\binom{n}{w}$ vertices, where each vertex represents a binary word of length n and constant weight w. We consider codes with minimum distance 4, so two vertices are adjacent if the Hamming distance between their related words is 2. For asymmetric codes, on the other hand, a graph consists of 2^n vertices, where each vertex represents a binary word of length n. Codes with minimum asymmetric distance 2 are considered, so two vertices are adjacent if the asymmetric distance between their related words is 1. The search for good codes can now be divided into three parts:

1) Start from an initial set of codewords.

2) Use a heuristic to find a good code.
3) Use local improvement to further improve the final code.

Some approaches do not comprise the first and the last parts.

A. Choosing Initial Codewords

As for the choice of possible initial codewords, it is impossible to give general guidelines. Often, it is necessary to construct such an initial set if we are searching for a large code, as the heuristics otherwise do not converge properly. One method of choosing initial words is to take a subset of words of a code recently found. If we have a set of initial codewords, we may or may not allow these to be changed during the remainder of the search.

In the construction of asymmetric codes, it has turned out that best known constant weight codes can be used to get good sets of initial codewords. If we want to construct a code of length n and minimum asymmetric distance 2, we take a best known code (there may be several nonisomorphic such codes) of length $n+1$ with minimum Hamming distance 4 and constant weight w close to $n/2$. One of the coordinates of this code is deleted (depending on which coordinate is deleted, we may obtain nonisomorphic codes), and the resulting code with words of weight $w-1$ and w will have minimum asymmetric distance 2. The search for the rest of the words is also made easier by the fact that words of weight less than $w-1$ and greater than w can be considered separately.

Constant weight codes proving the following bounds were used as seeds in this search to get record-breaking asymmetric codes: $A(12,4,5) \geq 80$ (in this case, local improvement was needed at the end of the heuristic search), $A(13,4,6) \geq 166$, and $A(14,4,7) \geq 325$. These constant weight codes are from [4].

B. Heuristics: Greedy and Tabu Search

The *greedy search* approach is based on ideas presented in [4] and [10]. Given a graph G (constructed from the coding problem in consideration), we want to find a large IS. This is done in the following way. At each iteration, we choose one vertex of minimum degree in the current graph, add this vertex to the code C, and delete this vertex and all its adjacent vertices to obtain a new smaller graph. This iterative process is then repeated from the new graphs obtained until there are no vertices left.

Clearly, there are often many vertices of minimum degree in the intermediate graphs. In [4] the method of choosing the lexicographically least vertex is suggested (when the vertices are labeled with their corresponding codewords). In [10] a random feature is suggested, thereby also allowing for different solutions after repeated runs. In this work, we randomly choose one of the minimum-degree vertices, but we also use features of evolutionary algorithms to make the search more effective.

We carry out p (independent) search processes for codes and introduce an integer sequence $r_1 < r_2 < \cdots$ of what we call *cutoff points*. Cutoff point i is attained when the codes have cardinality r_i. At these cutoff points the search processes interact, after which the search is continued until cutoff point $i+1$, and so on. In the interactions, the new starting points (codes) for the search processes are chosen by considering the sizes of the remaining graphs for the p codes. The codes with the largest graphs "survive" in this competition. Different strategies can be used for this: either one (largest) graph or a set of large graphs determine all new starting points.

An indication of the quality of the greedy algorithm in searching for constant weight codes is given by the fact that we, in any implemented version of it, were able to find the (unique; see [4]) Steiner system proving $A(12,4,6) = 132$.

Tabu search was recently used in [3] to search for mixed binary/ternary codes. One of the approaches in that paper is based on the problem of constructing covering codes. We first consider the case when d is odd. Then, if we in a graph G (where vertices at Hamming distance 1 from each other are connected!) want to find a code with minimum distance d, where d is odd, we are looking for a set of vertices (codewords) such that the spheres consisting of vertices at distance at most $(d-1)/2$ from the codewords are not overlapping.

Using this idea, we can define the three concepts needed to apply a local search algorithm such as tabu search. A solution C is any set of M codewords for some fixed M. A neighbor is obtained by replacing one vertex (codeword) with one of its adjacent vertices. Finally, the cost is

$$c(C) = \sum_{x \in V} \max(0, f(x) - 1) \tag{1}$$

where $f(x)$ is the number of codewords at distance at most $(d-1)/2$ from x. Since the distance between any two words in a constant weight code is even, we can clearly use this approach by considering spheres of vertices at distance at most $(d-2)/2$ from the codewords. In general, however, more substantial modifications have to be done when d is even.

In the construction of asymmetric codes in this correspondence with minimum distance 2 (even!), we use an idea from [3]. Now, we also use the cost function given in (1), but $f(x) = g(x) + h(x)/K$, where $g(x) = 1$ if x is a codeword and $h(x)$ is the number of codewords at asymmetric distance 1 from x for some fixed $K \geq \lfloor n^2/2 \rfloor$.

C. Local Improvement

Local improvements of packings were also discussed in [4] and [10]. In both these papers, an exchange procedure is discussed: for small values of k, an exhaustive search is carried out where it is determined whether $k+1$ codewords can be added after any k codewords have been deleted. If such an operation is not possible for a given k, the code is said to be k-optimal. For large values of k, where such an exhaustive search is not feasible, a similar *heuristic* method is discussed in [4].

In the search we can also utilize a variant of this. Namely, for a given small value of k, we can try to replace a set of k codewords by a different set of k codewords. Of course, this jiggling does not improve the packing as it just leads to a different packing with the same parameters. However, through this operation we are able to get a new starting point for the rest of the search.

V. Searching for Colorings

Searching for good colorings is much more difficult than searching for codes. Namely, as a coloring is a partitioning of all words in the space F into codes, we want to find many good codes simultaneously in order to get a coloring with a big norm. Unfortunately, a direct approach to find such partitions is not feasible in general. We shall here first discuss a general approach based on evolutionary algorithms and the code-searching algorithms discussed in the previous section, and then have a look at tailored methods for finding (parts of) good partitions.

A. A Sequential Approach

In the approach to be discussed here, the codes of a partition are constructed one by one. That is, we start from a code with largest known cardinality, delete these words from the search space, and try to find as large a code as possible using only the remaining words. After this, we try to find a third code in the same way, and so on.

TABLE I
PARTITIONS FOR CONSTANT WEIGHT CODES

n	w	m	$\Pi(n,w)$	Norm
13	5	14	123, 123, 123, 121, 113, 109, 109, 106, 98, 89, 76, 57, 34, 6	135537
13	5	14	123, 123, 123, 122, 113, 109, 109, 106, 99, 88, 72, 55, 34, 11	135069
13	5	14	123, 123, 123, 121, 113, 108, 108, 106, 96, 93, 77, 56, 34, 6	135483
13	6	14	166, 166, 166, 157, 150, 144, 141, 134, 128, 118, 98, 78, 50, 19, 1	237248
13	6	14	166, 166, 165, 156, 150, 143, 140, 134, 130, 119, 104, 79, 47, 16, 1	237762
14	5	14	169, 169, 169, 169, 155, 153, 151, 148, 146, 139, 131, 113, 88, 66, 29, 7	289940
14	5	14	169, 169, 169, 169, 155, 153, 151, 148, 146, 138, 132, 113, 92, 59, 29, 10	289822
14	6	17	278, 275, 265, 257, 248, 231, 229, 220, 211, 201, 182, 158, 127, 82, 31, 7, 1	671763
14	6	16	278, 273, 265, 257, 250, 231, 229, 219, 211, 203, 184, 156, 127, 81, 35, 4	672203
14	7	17	325, 322, 307, 298, 281, 265, 253, 250, 240, 228, 198, 174, 135, 85, 50, 16, 5	875352

TABLE II
LOWER BOUNDS ON $A(n,4,w)$, $21 \le n \le 28$, $7 \le w \le 13$

$n \backslash w$	7	8	9	10	11	12	13
21	6156^a	10753^a	16897^a	20188	20188	16897	10753
22	8252^a	16430^a	25570	36381	39688	36381	25570
23	11638	23276	40786	57436	73794	73794	57436
24	15656^a	34914	59387^a	96496	116937^a	146552	116937
25	21106^{p0}	46872^{p1}	88748^{p0}	140605^b	196449^{p0}	228901^{p0}	228901
26	26920	65364^{p1}	128050	218905	315700^{p0}	398381^{p2}	425950^{p2}
27	35510	87709	186058^{p1}	330347^{p0}	510571^{p1}	675262^{p0}	778872^{p1}
28	44747	121403	260224^{p0}	502068	806303	1154541	1400118^{p0}

a	–	[7]
b	–	Partition from this paper with the method in [7]
pi	–	See Section II.A

Thus the algorithms developed in Section IV can be used with only minor straightforward modifications to construct these codes; some of the earlier ideas can also be carried over to this case to get tailored heuristics for making local improvements on partitions.

Actually, we can search for a new code among a slightly larger set of words. Namely, it is worth adding some of the codewords in the partition so far to the search space. These are words with the property that they can be interchanged with an unused word so that the minimum-distance property of the subcode is fulfilled. If such a word of a subcode is in the final new subcode, the aforementioned interchange of words is carried out. This idea can be further generalized by considering pairs of codewords.

This method is greedy in the sense that it sequentially tries to find as good codes as possible. To further improve it, we have used a feature from evolutionary algorithms as in searching for codes, but now on a higher level: in constructing codes, we are repeatedly adding words, in finding colorings, we are repeatedly adding codes. Now, the search processes interact after each addition of a new subcode to the partition, and the partition(s) with the greatest norm become(s) the next starting point.

B. The Permutation Method

In discussing searches for good codes, we remarked that a construction of an initial set of codewords before starting the heuristic search is occasionally justified. This idea can immediately be carried over to the coloring problem: good results may be obtained by constructing the first few *codes* using a different method. For constant weight codes, this is conveniently done using what we call the permutation method.

In the permutation method, we start from a best known constant weight code with given parameters. When a permutation is applied to the codewords of the original code C_1, clearly the new code C_2 has the same minimum distance. Thus if $C_1 \cap C_2 = \emptyset$, these two codes can be used as subsets of a partition.

TABLE III
Partitions for Asymmetric Codes

n	m	$\Pi(n)$	Norm
9	11	62, 62, 62, 62, 58, 54, 51, 44, 34, 19, 4	27726
9	11	62, 62, 62, 62, 56, 56, 52, 44, 30, 20, 6	27624
10	13	112, 110, 110, 109, 104, 98, 95, 87, 75, 61, 46, 14, 3	97306
11	14	198, 197, 194, 189, 182, 180, 177, 170, 158, 140, 117, 88, 46, 12	345380

To get k initial codes of a partition we now try to find $k-1$ partitions such that the codes obtained by applying these to the original code have as small intersections as possible. That is, if the codes obtained (including the original code C_1) are C_i, $1 \le i \le k$, we want $|\bigcup_{i=1}^{k} C_i|$ to be as large as possible. The size of this set is at most $k|C_1|$ with equality exactly when no two subcodes intersect. Before proceeding, words that occur in several subcodes must be removed from all but one subcode (which can be done in several ways).

C. Local Improvements

Apart from applying local improvements to improve one given code, methods can be developed to improve partial partitions (for example, those obtained by the permutation method). One way to do this is to first move some words from the large codes in the partition to the small codes when this can be done without violating the minimum-distance criterion. After this, any of the methods described earlier can be used to make local improvements to the subcodes from which words were taken. As a result, the total number of codewords used in the partial partition is increased.

Local improvements to improve one or several codes of a partial partition can be applied at any stage of the search.

VI. New Bounds

As mentioned in the Introduction, the motivation for our work was to try to find improvements on lower bounds for constant weight codes with minimum Hamming distance 4, and for codes with minimum asymmetric distance 2. The codes and partitions giving the bounds presented in the sequel are not listed in this correspondence. They can be found in a report published by the authors [8] and in electronic form from the following WWW page: `http://saturn.hut.fi/~pat/codelist.html`.

In Table I we present the index vectors for the new partitions for constant weight codes; these partitions can be used to get new lower bounds for other constant weight codes. The parameter m in the table stands for the number of codes in the partition.

The partitions in Table I together with the partitions in [4] and [7] can be used to obtain some of the lower bounds in [4] and in the updated table presented here. Note that some of the partitions in [4] are not needed any more. In Table II we present a portion of the table for best known lower bounds on $A(n,4,w)$. Entries that have been improved since [4] are marked. Five other sporadic improvements have recently been found: $A(16,4,5) \ge 315$, $A(19,4,5) \ge 692$, $A(19,4,6) \ge 1620$, $A(20,4,6) \ge 2304$ (all [18]), and $A(23,4,4) \ge 418$ [2].

In Table III we present the index vectors for the new partitions for asymmetric codes.

Finally, in Table IV, we present lower bounds on $A_a(n,2)$, thus updating parts of the tables in [6] and [9]. The values for $n \le 9$ are exact.

TABLE IV
Lower Bounds on $A_a(n,2)$, $4 \le n \le 25$

n	Lower bound	n	Lower bound
4	4^a	15	2288^a
5	6^a	16	4280^a
6	12^a	17	8308^a
7	18^a	18	15762^a
8	36^a	19	29334^c
9	62^a	20	56144^a
10	112^b	21	107648^c
11	198^b	22	201508^c
12	378^b	23	369870^c
13	699^b	24	678098^c
14	1273^b	25	1313130^c

a – See [6, 9]
b – This paper (heuristic search)
c – This paper (partitioning)

The partitions used in calculating the bounds in Table IV are from this correspondence and from [4] and [6]; we also use the well-known fact that there is always a partition $\Pi(n)$ with $n+1$ codes [17]. All the codes obtained by the partitioning method have used $\Pi(n_1) \times \Pi(n_2, w)$, where n_2 is even and $n_2 - 4 \le n_1 \le n_2 - 1$.

Acknowledgment

The authors wish to thank R. Bar-Yehuda for many helpful discussions on independent sets and colorings.

References

[1] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms.* New York: Oxford Univ. Press, 1996.

[2] S. Bitan and T. Etzion, "The last packing number of quadruples and cyclic SQS," *Des. Codes Cryptogr.,* vol. 3, pp. 283–313, 1993.

[3] A. E. Brouwer, H. O. Hämäläinen, P. R. J. Östergård, and N. J. A. Sloane, "Bounds on mixed binary/ternary codes," this issue, pp. 140–161.

[4] A. E. Brouwer, J. B. Shearer, N. J. A. Sloane, and W. D. Smith, "A new table of constant weight codes," *IEEE Trans. Inform. Theory,* vol. 36, pp. 1334–1380, 1990.

[5] F. C. Chow and J. L. Hennessy, "The priority-based coloring approach to register allocation," *ACM Trans. Program. Lang. Syst.*, vol. 12, pp. 501–536, 1990.

[6] T. Etzion, "New lower bounds for asymmetric and unidirectional codes," *IEEE Trans. Inform. Theory*, vol. 37, pp. 1696–1704, 1991; vol. 38, pp. 1183–1184, 1992.

[7] T. Etzion and S. Bitan, "On the chromatic number, colorings, and codes of the Johnson graph," *Discr. Appl. Math.*, vol. 70, pp. 163–175, 1996.

[8] T. Etzion and P. R. J. Östergård, "Greedy and heuristic algorithms for codes and colorings," Tech. Rep. CS 909, Comput. Sci. Dept., Technion, Haifa, Israel, 1997.

[9] G. Fang and H. C. A. van Tilborg, "Bounds and constructions of asymmetric or unidirectional error-correcting codes," *Appl. Algebra Eng. Commun. Comput.*, vol. 3, pp. 269–300, 1992.

[10] T. A. Feo, M. G. C. Resende, and S. H. Smith, "A greedy randomized adaptive search procedure for maximum independent set," *Oper. Res.*, vol. 42, pp. 860–878, 1994.

[11] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.

[12] M. R. Garey, D. S. Johnson, and H. C. So, "An application of graph coloring to printed circuit testing," *IEEE Trans. Circuits Syst.*, vol. CAS-23, pp. 591–599, 1976.

[13] F. Glover, "Tabu search—Part I," *ORSA J. Comput.*, vol. 1, pp. 190–206, 1989.

[14] A. Hertz and D. de Werra, "Using tabu search techniques for graph coloring," *Computing*, vol. 39, pp. 345–351, 1987.

[15] I. S. Honkala and P. R. J. Östergård, "Code design," in *Local Search in Combinatorial Optimization*, E. Aarts and J. K. Lenstra, Eds. New York: Wiley, 1997, pp. 441–456.

[16] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, "Optimization by simulated annealing: An experimental evaluation; Part II, graph coloring and number partitioning," *Oper. Res.*, vol. 39, pp. 378–406, 1991.

[17] R. J. McEliece and E. R. Rodemich, "The Constantin–Rao construction for binary asymmetric error-correcting codes," *Inform. Contr.*, vol. 44, pp. 187–196, 1980.

[18] K. J. Nurmela, M. K. Kaikkonen, and P. R. J. Östergård, "New constant weight codes from linear permutation groups," *IEEE Trans. Inform. Theory*, vol. 43, pp. 1623–1630, Sept. 1997.

[19] C. L. M. van Pul and T. Etzion, "New lower bounds for constant weight codes," *IEEE Trans. Inform. Theory*, vol. 35, pp. 1324–1329, 1989.

Reliability-Based Syndrome Decoding of Linear Block Codes

Marc P. C. Fossorier, *Member, IEEE*, Shu Lin, *Fellow, IEEE*, and Jakov Snyders, *Member, IEEE*

***Abstract*—In this correspondence, various aspects of reliability-based syndrome decoding of binary codes are investigated. First, it is shown that the least reliable basis (LRB) and the most reliable basis (MRB) are dual of each other. By exploiting this duality, an algorithm performing maximum-likelihood (ML) soft-decision syndrome decoding based on the LRB is presented. Contrarily to previous LRB-based ML syndrome decoding algorithms, this algorithm is more conveniently implementable for codes whose codimension is not small. New sufficient conditions for optimality are derived. These conditions exploit both the ordering associated with the LRB and the structure of the code considered. With respect to MRB-based sufficient conditions, they present the advantage of requiring no soft information and thus can be preprocessed and stored. Based on these conditions, low-complexity soft-decision syndrome decoding algorithms for particular classes of codes are proposed. Finally, a simple algorithm is analyzed. After the construction of the LRB, this algorithm computes the syndrome of smallest Hamming weight among $o(K^i)$ candidates, where K is the dimension of the code, for an order i of reprocessing. At practical bit-error rates, for codes of length $N \leq 128$, this algorithm always outperforms any algebraic decoding algorithm capable of correcting up to $t+1$ errors with an order of reprocessing of at most 2, where t is the error-correcting capability of the code considered.**

***Index Terms*—Block codes, generalized Hamming weights, soft-decision decoding, syndrome decoding.**

I. Introduction

Recently, several reliability-based maximum-likelihood decoding (MLD) algorithms for binary linear block codes have been proposed [1]–[11]. These algorithms first reorder the symbols, within each received block, according to their reliability measures. Thereafter, an initial decoded codeword is reprocessed in a particular structured strategy based on the reordering. Such strategy confines the search to a usually small class of candidates of high likelihood, whereby the decoding complexity is significantly reduced. Basically, two general different approaches exist, namely, the vector space associated with either the code considered C or with its dual space $C^{\perp}$ are processed. These two spaces are, respectively, referred to as the G-space and the H-space of the code.

In [7] and [11], an algebraic decoder generates the successive candidates processed by the algorithm. The vectors entered into the decoder are obtained by systematically adding to the (bit-by-bit hard detected version of the) received sequence error patterns whose supports are confined to positions where the detection is of relatively low reliability. These reprocessing methods yield codewords to be scored, hence the algorithms are of the G-space type. In a different kind of G-space algorithms [6], [8]–[10] an equivalent code is first determined by identifying the K *most reliable* independent positions

Manuscript received February 1, 1996; revised June 24, 1997. This work was supported by the National Science Foundation under Grant NCR-91-15400 and Grant NCR-94-15374, and in part by the Israel Science Foundation administered by the Israel Academy of Sciences and Humanities. The material in this correspondence was presented in part at the IEEE Symposium on Information Theory, Whistler, BC, September 1995.

M. P. C. Fossorier and S. Lin are with the Department of Electrical Engineering, University of Hawaii, Honolulu, HI 96822 USA.

J. Snyders is with the Department of Electrical Engineering-Systems, Tel-Aviv University, Tel-Aviv 69978, Israel.

Publisher Item Identifier S 0018-9448(98)00072-8.