# An Optimal Disk Allocation Strategy for Partial Match Queries on Non-Uniform Cartesian Product Files *

Sajal K. Das
Department of Computer Science
University of North Texas
Denton, TX 76203-1366
E-mail: das@cs.unt.edu

M. Cristina Pinotti
Ist. Elab. Infor.
National Council of Research
Pisa, ITALY
E-mail: pinotti@iei.pi.cnr.it

## Abstract

*The disk allocation problem addresses the issue of how to distribute a file on to several disks to maximize the concurrent disk accesses in response to a partial match query. In the past this problem has been studied for binary as well as for $p$-ary cartesian product files. In this paper, we propose a disk allocation strategy for non-uniform cartesian product files by a coding theoretic approach. Our strictly optimal disk allocation strategies are based on a large and flexible class of maximum distance separable (MDS) codes, namely the redundant residue codes.*

**Keywords** : Partial match queries, non-uniform cartesian product files, optimal disk allocation, pairwise prime residue codes.

## 1 Introduction

The demand for efficient retrieval methods in large databases has increased significantly in recent years. Important practical applications such as spatial databases, airline reservation systems, cartography, and Web access require voluminous amount of data (often of the order of terabytes) which are too large to store on a single secondary storage device like a disk. Moreover, distributing files on to multiple disks that are accessible concurrently, allows us to retrieve data in parallel, thus reducing the retrieval time.

This paper designs an optimal disk allocation strategy for solving parallel match queries on large, non-uniform, cartesian product files. Before proceeding further, let us first define the problem and the associated terminology.

### 1.1 Cartesian Product Files

An *$n$-attribute file* $F$ is a set of $n$-tuples, called *records*. Assuming that the $i$th attribute ranges in the interval $D_i$, for $1 \leq i \leq n$, a record is an element of the space $D = D_1 \times D_2 \times \ldots \times D_n$, and a file is a subset of $D$. A file is stored on a disk by partitioning the records into *buckets* (or *pages*), containing mutually disjoint sets of records. In order to equally distribute the records of the file $F$ over the buckets, each interval $D_i$ is divided into $m_i$ disjoint sets. Each bucket is denoted by a string $< x_1, x_2, \ldots, x_n >$, where $x_i \in [0, 1, \ldots, m_i - 1]$. From now on, we will refer to $m_i$ as the *domain* of the $i$th attribute. In this paper we will deal with an $n$-attribute file at the bucket level.

A *$p$-ary cartesian product file* is a file in which all the attribute domains assume identical value $p$, that is, $m_1 = m_2 = \ldots = m_n = p$. A *binary cartesian product file* is a special case for $p = 2$. A *non-uniform cartesian product file* is a file in which the domains assume different values. Selecting different ranges for the attribute domains can be very useful when the original intervals $D_i$ are quite different from each other. Non-uniform and $p$-ary cartesian product files consist of $M = \Pi_{i=1}^{n} m_i$ and $M = p^n$ buckets, respectively.

### 1.2 Disk Allocation Problem

An *$m$-disk allocation strategy* distributes the buckets of a cartesian product file on to $m$ disks. A *partial match query*, $q$, is defined as an $n$-tuple $q =< q_1, \ldots, q_n >$, where $q_i$ is either a value from the domain $m_i$ of the $i$th attribute (i.e., $q_i$ is specified) or is unspecified. A bucket *qualifies* for the partial match query $q$ if its associated string coincides with $q$ on the specified attributes.

The set of qualifying buckets for a query $q$ forms the *query response set* having the size $N(q) = \Pi_{j=1}^{k} m_{i_j}$, where $q_{i_1}, q_{i_2}, \ldots, q_{i_k}$ are the unspecified attributes of $q$. The *response time* of the query $q$ is defined as $R(q) = \max\{N_1(q), N_2(q), \ldots, N_m(q)\}$, where $N_i(q)$ is the number of qualifying buckets stored on disk $i$, for $1 \leq i \leq m$. Clearly, $N(q) = \sum_{i=1}^{m} N_i(q)$. Hence, the best possible response time is obtained by distributing the $N(q)$ qualifying buckets on to different disks as evenly as possible and performing the disk retrievals concurrently.

Thus, the *disk allocation problem* addresses the issue of how to distribute a multi-attribute file on several disks in order to minimize the response time for arbitrary queries. Intuitively, the buckets assigned to a given disk should be as different as possible so that only one (or very few) of

*This work is partially supported by Texas Advanced Technology Program grant TATP-003594031 and Texas Advanced Research Program grant TARP-003594013.

them may qualify for each partial match query. In particular, given an $m$-disk system, a disk allocation strategy is termed *strictly $q$-optimal* if the response time is given by $R(q) = \left\lceil \frac{N(q)}{m} \right\rceil$. Moreover, a disk allocation strategy is termed *strictly optimal* if it is strictly $q$-optimal *for every query $q$*.

### 1.3 Previous Work

In some of the earlier work related to the disk allocation, buckets were assigned to an $m$-disk system with the help of a (pseudo)random number generator, with $1/m$ as the probability of assigning a bucket to a particular disk. This method has no constraints either on the number of disks or the cardinality of the domains of the attributes. More sophisticated strategies have also been proposed for binary and $p$-ary cartesian product files [5, 8, 7, 1], most of which exhibit a better performance than this simple method.

Fang, Lee and Chang[6] interpreted the string associated with each bucket as an $n$-dimensional space point, and partitioned such points into two "similar" groups. The minimum spanning tree and the short spanning paths algorithms are the basis of their declustering algorithms. The method, as it is described, applies only to a 2-disk system.

Du and Sobolewski [5] proposed the *Disk Modulo* allocation method for an $m$-disk system, which assigns the bucket $< x_1, \ldots, x_n >$ to the disk unit $d_j$ such that $j = (\sum_{i=1}^{n} x_i) \bmod m$, where $0 \leq j \leq m-1$. Such a method is *strictly optimal* for the following cases: (i) partial match queries with only one unspecified attribute; (ii) partial match queries with at least one unspecified attribute $j$ whose domain $m_j$ satisfies $m_j \bmod m = 0$; (iii) all possible partial match queries when $m_i \bmod m = 0$ or $m_i = 1$ for all $1 \leq i \leq n$; and (iv) all possible partial match queries when $m = 2, 3$.

The idea of of finding an (ad-hoc) hash function [5], to be applied to the bucket strings, was followed by Kim and Pramanik [8] who proposed the *Field Exclusive-Or* allocation method. Given an $m$-disk system and letting $[+]$ denote the bitwise exclusive-or, a bucket $< x_1, \ldots, x_n >$ is allocated to the disk $d_j = [([+]_{i=1}^{n} x_i) \bmod m]$. This approach is shown to be strictly optimal for any partial match query (i) with two unspecified attributes; or (ii) with more than two unspecified attributes if there exists at least one unspecified field whose range is larger than $m$.

Based on linear binary error-correcting codes, Faloutsos and Metaxas [7] presented a heuristic for distributing binary cartesian product files with $n$-attributes onto a $2^d$-disk system. The basic idea is to group the buckets for each disk in such way that the associated binary strings form a linear error-correcting binary code of minimum distance $d$. The construction guarantees that buckets whose associated strings differ in less than $d$ digits will be assigned to different disks. Therefore, partial match queries with less than $d$ unspecified attributes exhibit a constant response time. Although this allocation strategy is strictly optimal, for arbitrary values of $n$ and $d$, there may exist no codes with such properties, and hence no disk allocation method.

Abdel-Ghaffar and El Abbadi [1] extended the results of [7] to $p$-ary cartesian product files, where $p > 2$, and proved the equivalence between strictly optimal allocation methods and *maximum distance separable* (MDS) codes. Based on the *Reed-Solomon Codes* which provides a large family of $p$-ary MDS codes, they proposed strictly optimal allocation methods for declustering $p$-ary cartesian product files with at most $p-1$ attributes among a number of disks $m$ equal to a power of $p$. The main drawback of this approach is that for the general case of $p$-ary cartesian product files with $n$ attributes, $n \geq p$, there exists only a few MDS codes. (A complete table of the known MDS codes can be found in [10].) However, the approach based on codes achieves the best performance among the proposed solutions to the disk allocation problem, especially when partial match queries with a large number of unspecified attributes are considered.

### 1.4 Our Contributions

Let us first elaborate on the motivation behind our work. Non-uniform cartesian product files are more realistic than $p$-ary cartesian product files. In fact, having non-uniform ranges for attribute domains may help in obtaining (optimal) buckets as argued below. Consider a 3-attribute file (the attributes assume only integer values) such that the first attribute ranges in $D_1 = [1..1000]$, the second in $D_2 = [1..21]$, and the third in $D_3 = [1..9]$. Also suppose that there are at most $1000 \times 21 \times 9$ records (one for each attribute value) in the file, and that a disk page (or bucket) contains 100 records.

Now let us examine the scenario when a $p$-ary cartesian product file is used.

- For $p = 4$, we obtain very large buckets, each containing at most $250 \times 6 \times 3$ records, which do not fit in one disk page.
- $p = 10$ leads to at most $100 \times 3 \times 1$ records for each bucket, which is still too large in size.
- $p = 20$ implies at most $50 \times 2 \times 1$ records in each bucket. Although a bucket perfectly fits in a page, yet the file is mapped into $20^3$ buckets, most of them being empty.

On the other hand, if we use a non-uniform file system with different partitions, we will have a better solution. For example, consider $m_1 = 20$, $m_2 = 11$ and $m_3 = 9$. Then each bucket has at most $(1000/20) \times (\lceil 21/11 \rceil) \times (\lceil 9/9 \rceil) =$

$50 \times 2 \times 1 = 100$ records, and we consider $20 \times 11 \times 9 = 1980$ buckets.

Motivated by the importance of the non-uniform cartesian product file, the main contribution of this paper is to bring the attention to a very large and flexible class of MDS (maximum distance separable) codes, called the *redundant residue codes*, that apply to the disk allocation problem for *non-uniform* cartesian product files. Such codes lead to strictly an optimal disk allocation strategy.

## 2 Terminology and MDS Codes

Let $m_1, m_2, \ldots, m_n$,be $n$ positive integer *radices*. Let the space $S = m_1 \times m_2 \times \ldots \times m_n$ be defined as the the set of all $n$-tuples $a_1, a_2, \ldots, a_k, a_{k+1}, \ldots, a_n$, where $a_i \in [0, 1, \ldots m_i - 1]$. The cardinality of this set is $M = \Pi_{i=1}^{n} m_i$. A *code* $C$ of radices $m_1, m_2, \ldots, m_n$, is a subset of $S$. Each $n$-tuple of $C$ is called a *codeword*. A binary (resp. $p$-ary, $p > 2$) code, is a subset of the space $S$ defined with all the radices equal to 2 (resp. $p$). The *Hamming weight* of a codeword is the number of its nonzero components. The *Hamming distance* between two codewords is the number of digits in which they differ. The *minimum distance* of the code $C$ is defined as the minimum Hamming distance among all pairs of distinct codewords in $C$. If the $n$-tuple $< 0, 0, \ldots, 0 >$ belongs to $C$, the minimum distance $d$ of the code $C$ is the weight of the (non-zero) codeword with minimum Hamming weight.

Now onwards, let $C = [n, \gamma, d]$ denote a code of length $n$, having size (or cardinality) $\gamma$ and minimum distance $d$. A set $\{i_1, i_2, \ldots, i_k\}$ of $k$ positions is said to be the *information set* of $C$ if for every choice of $k$ digits $a_{i_1}, a_{i_2}, \ldots a_{i_k}$, where $a_{i_j} \in [0, m_{i_j})$, there is a unique codeword $< x_1, x_2, \ldots, x_n >$ such that $x_{i_j} = a_{i_j}$ for $1 \leq j \leq k$. A code with an information set is called a *systematic* code of size $\gamma = \Pi_{j=1}^{k} m_{i_j}$.

Given a set of $n$ radices $m_1, \ldots, m_n$ and an integer $\gamma$, the main problem in coding theory is to determine the maximum value $d$ such that there exists a code $C = [n, \gamma, d]$. Namely, a code of minimum distance $d$, is capable of detecting at least all non-zero error $n$-tuples with weight at most $d - 1$, and also correcting all error $n$-tuples of weight $\leq \lfloor (d-1)/2 \rfloor$.

In this context, the following result is crucial.

**The Singleton bound** [10]: Given a code $C$ of radices $m_1, m_2, \ldots, m_n$ and of size $\gamma$ such that $\max \left\{\Pi_{i=1}^{k} m_{j_i}\right\} > \gamma \geq \max \left\{\Pi_{i=1}^{k-1} m_{j_i}\right\}$, the minimum distance $d$ of such a code $C$ yields $d \leq n - k + 1$.

A code $C$ of minimum distance $d = n-k+1$ (i.e., satisfying the equality of the Singleton bound) is said to be a *maximum distance separable* (MDS) code.

It can be shown that: (i) a code is an MDS code if and only if any $k$ digits are able to represent $\gamma$ different $n$-tuples, and (ii) every MDS code is a systematic code.

## 3 Residue Codes

Let the *residue representation* of an integer $X$ in the *Residue Number System* (RNS) of pairwise prime radices, or *moduli*, $m_1, \ldots, m_n$ be defined as the $n$-tuple $< x_1, x_2, \ldots, x_n >$ such that $x_i \equiv X \bmod m_i$. Let $S$ be the space defined on the radices $m_1, \ldots, m_n$. The Chinese Remainder Theorem [2] guarantees that each $n$-tuple of the vector space $S$ can be uniquely associated with the residue representations of an integer $X \in [0, M-1]$ where $M = \Pi_{i=1}^{n} m_i$.

For an arbitrary value of $k$, $1 \leq k \leq n$, the *redundant residue code* (or simply, *residue code*), $C$, is the subset of $S$ consisting of the residue representations of the integers in $[0, M_I - 1]$, where $M_I = \Pi_{i=1}^{k} m_i$. The following properties hold [2, 9]:

- Each codeword of $C$ has length $n$.
- $C$ has size $M_I$.
- The moduli $m_1, \ldots, m_k$ form the information set. These moduli are called *non-redundant* since they are necessary and sufficient to represent the integers in $[0, M_I)$. The remaining $n - k$ moduli form the set of the *redundant* moduli. Let $M_R = \Pi_{i=k+1}^{n} m_i = \frac{M}{M_I}$ be the product of the redundant moduli.
- The all-zero codeword (the residue representation of the integer 0) belongs to $C$.
- $C$ has minimum distance $d$ if and only if

$$\max \left\{\Pi_{i=1}^{d} m_{j_i}\right\} > M_R \geq \max \left\{\Pi_{i=1}^{d-1} m_{j_i}\right\}$$

for $1 \leq j_i \leq n$.

A sufficient condition to construct an MDS residue code is given below.

**Fact 1** *[2, 9] Let the radices $m_1, \ldots, m_n$ be sorted in increasing order, $m_1 < m_2 < \ldots < m_n$. A residue code $C$ is an MDS code of minimum distance $d$ and has $m_{n-d+1}, m_{n-d+2}, \ldots, m_n$ as redundant moduli where $1 \leq d \leq n-1$.*

The following proposition will be crucial in our new solution to the disk allocation problem.

**Proposition 1** *Given an RNS of radices $m_1, \ldots, m_n$, and the residue code $C = [n, M_I, d]$ whose codewords correspond to the residue representations of the integers in*

$[0, M_I)$. *The space* $S = m_1 \times m_2 \times \ldots m_n$ *can be partitioned into residue codes* $C_1, C_2, \ldots C_{\frac{M}{M_I}-1}$, *where the codewords of* $C_j = [n, M_I, d]$ *are the residue representations of the integers in* $[jM_I, (j+1)M_I - 1]$.

## 4 Code-Based Disk Allocation Strategy

We are ready to design an efficient strategy for allocating an $n$-attribute, non-uniform cartesian product file $F$ onto multiple disks. The $i$-th attribute of $F$ ranges in $m_i$. We assume that the attribute domains $m_1, \ldots, m_n$ are pairwise prime moduli and they are sorted in increasing order. Let $M_I$ and $M_R$ be the product of the leftmost $n-d+1$ and the rightmost $d-1$ moduli, respectively. Let the *seed code* $C = [n, M_I, d]$ be an MDS residue code corresponding to the residue representation of the integers $[0, \ldots, M_I - 1]$ in the RNS associated with the same radices $m_1, m_2, \ldots, m_n$. The proposed strategy is given below for an $M_R$-disk system.

**Procedure Decluster (**$F$**,** $C = [n, M_I, d]$**,** $K = M_R$**)**

- Based on the seed code $C$, partition $S$ into $M_R$ codes, according to Proposition 1;
- Assign each code to a distinct disk.

**Theorem 1** *Distributing an* $n$*-attribute file* $F$ *into an* $M_R$*-disk system by the Decluster procedure based on a seed code* $C = [n, M_I, d]$, *every partial match query* $q$ *with at most* $d-1$ *unspecified attributes yields a constant response time.*

**Proof.** Observe that all the strings associated with the buckets qualifying for the query $q$ with at most $d-1$ unspecified attributes, have Hamming distance at most $d-1$. On the other hand, by Proposition 1, the strings allocated to the same disk have at least Hamming distance $d$. Hence, at most one bucket qualifies for each of those queries in each disk. □

Therefore,

**Theorem 2** *The disk allocation strategy, obtained by applying the procedure Decluster to the file* $F$, *is strictly* $q$*-optimal for every query* $q$ *if: (i)* $q$ *has at most* $d-1$ *unspecified attributes, or (ii)* $q_{n-d+2}, \ldots, q_n$ *are among the* $u$ *unspecified attributes of* $q$, *where* $u \geq d-1$.

**Proof.** For the query of Type (i), the result follows immediately from Theorem 1.

Now, let $q =< q_1, \ldots, q_n >$ be a query of Type (ii). Assume that the attributes $q_{n-d+2}, \ldots, q_n$, as well as $q_{i_1}, q_{i_2}, \ldots, q_{i_{u-d+1}}$ are unspecified, where $i_1 > i_2 > \ldots > i_{u-d+1}$ and $u \geq d-1$. The query $q$ can be decomposed into a set $Q$ of subqueries with all the redundant $d-1$ attributes $q_{n-d+2}, \ldots, q_n$ unspecified and such that, for every choice of the $u-d+1$ digits $a_{i_1}, a_{i_2}, \ldots, a_{u-d+1}$, there exists a unique query in $Q$ satisfying $q_{i_j} = a_{i_j}$ where $1 \leq j \leq i_{u-d+1}$.

It is easy to see that each bucket qualifying for a subquery of $Q$ also qualifies for $q$. Similarly, each bucket qualifying for $q$ qualifies also for a unique subquery of $Q$. Therefore, in order to count how many buckets qualify for $q$ in each disk, let us count how many buckets qualify in each disk for each single subquery of $Q$.

Each subquery of $Q$ has exactly the $d-1$ redundant attributes unspecified. Therefore, the response set for each subquery of $Q$ is $\Pi_{n-d+2}^{n}(m_i) = M_R$. As observed in Theorem 1, at most one bucket must qualify in each disk and since there are $M_R$ disks in the system, exactly one bucket qualifies in each disk for each subquery of $Q$. Now $Q$ consists of $\Pi_{j=1}^{u-d+1} m_{i_j}$ subqueries, which is also the same as the number of buckets qualifying for $q$ in each disk. That is, $R(q) = \Pi_{j=1}^{u-d+1} m_{i_j}$.

Observing that $N(q) = M_R \left(\Pi_{j=1}^{u-d+1} m_{i_j}\right)$, the proposed solution is strictly optimal for all queries of Type (ii) since $R(q) = \left\lceil \frac{N(q)}{M_R} \right\rceil$. □

Since an MDS residue code of minimum distance $d$ can be obtained for every value of $d$, where $1 \leq d \leq n$, it holds:

**Corollary 1** *Let* $m_1, \ldots, m_n$ *be* $n$ *pairwise prime moduli, sorted in increasing order. For* $1 \leq d \leq n$, *using* $M_R = \Pi_{i=n-d+2}^{n} m_i$ *disks, there exists a strictly optimal disk allocation strategy for*

- *every query* $q$ *with less than* $d$ *unspecified attributes, or*
- *every query with* $d-1$ *attributes* $q_{n-d+2}, \ldots, q_n$ *among the unspecified ones.*

In conclusion, we call a disk allocation strategy $\beta$*-strictly optimal* if $\beta = \left\lceil \frac{R(q)}{R^*(q)} \right\rceil$ is the ration between the response time of the query $q$ and the response time of the strictly $q$-optimal allocation in a $D$-disk system, that is, $R^*(q) = \lceil \frac{N(q)}{D} \rceil$. In other words, a $\beta$-strictly optimal disk allocation strategy is $\beta$ times slower than a strictly optimal strategy.

The following result can be claimed for the general case:

**Theorem 3** *The disk allocation strategy, obtained by applying the procedure Decluster to the file* $F$, *guarantees that each query* $q$ *with* $u$ *unspecified attributes,* $q_{i_1}, \ldots, q_{i_u}$, *where* $i_1 > i_2 > \ldots > i_u$ *and* $d-1 \leq u \leq n$, *has response time at most max* $\left(1, \Pi_{j=d}^{u} m_{i_j}\right)$. *Equivalently, there*

*exists a* $\left\lceil \frac{max(1,\Pi_{j=d}^{u} m_{i_j})}{max(1,\Pi_{j=1}^{u} m_{i_j}/M_R)} \right\rceil$ *-strictly optimal disk allocation strategy.*

**Proof.** The query $q$ can be decomposed into a set $Q$ of $\Pi_{j=d}^{u} m_{i_j}$ subqueries such that for every choice of the $u - d + 1$ digits $a_{i_d}, a_{i_{d+1}}, \ldots, a_{i_u}$ there exists a subquery in $Q$ with $q_{i_d} = a_{i_d}, q_{i_{d+1}} = a_{i_{d+1}}, \ldots, q_{i_u} = a_{i_u}$ and the attributes $q_1, q_2, \ldots, q_{d-1}$ unspecified.

As mentioned before, each bucket qualifying for a subquery of $Q$ also qualifies for $q$, and vice versa.

Each subquery of $Q$ has exactly $d - 1$ unspecified attributes, and at most $M_R$ qualifying buckets. From Theorem 1, at most one bucket qualifies in each disk. Therefore, observing that $Q$ consists of $\Pi_{j=d}^{u} m_{i_j}$ subqueries, at most $\Pi_{j=d}^{u} m_{i_j}$ buckets qualify for $q$ in each disk. □

## 5 Conclusion

In this paper, we have discussed the disk allocation problem for non-uniform cartesian product files. Based on a large and flexible class of maximum distance separable codes, namely the *residue* codes, we have derived strictly optimal allocation strategies where the attribute domains are pairwise prime. We have extended these results to non-uniform product files with non-pairwise prime attribute domains [4], by defining new families of residue codes, called the *redundant non-pairwise prime residue codes*.

## References


[1] K.A.S. Abdel-Ghaffar, A. El Abbadi, "Optimal Disk Allocation for Partial Match Queries", *ACM Trans. on Database Systems*, Vol. 18, No.1, Mar. 1993, pp. 132-156.

[2] F. Barsi, P. Maestrini, "Error Correcting Properties of Redundant Residue Number Systems", *IEEE Trans. on Computers*, Vol. 22 , 1973, pp. 307-315.

[3] F. Barsi, P. Maestrini, "Error Codes in Residue Number Systems with Non-Pairwise-Prime Moduli", *Information and Control*, Vol. 46, No. 1, July 1980, pp. 16-25.

[4] S.K. Das, M.C. Pinotti, "The Disk Allocation Problem", *Technical Report*, IEI-CNR, Pisa, Italy, Sept. 1998.

[5] H.C. Du, J.S. Sobolewski, "Disk Allocation for Cartesian Product Files" *ACM Trans. on Database Systems*, Vol. 7, No. 1, Mar. 1982, pp. 82-101.

[6] M.F. Fang, R.C.T. Lee and C.C. Chang, "The Idea of De-Clustering and its Applications", in *Proceedings 12th Int'l Conf. VLDB*, Kyoto, Japan, Aug. 1986, pp. 181-188.

[7] C. Faloutsos, D. Metaxas, "Disk Allocation Methods using Error Correcting Codes", *IEEE Trans. on Computers*, Vol. 40, No. 8, Aug. 1991, pp. 907-913.

[8] M.H. Kim, S. Pramanik, "Optimal File Distribution for Partial Match retrieval" in *Proceedings of the ACM-SIGMOD Int'l Conference on Management of Data*, Chicago, 1988, pp. 173-182.

[9] H. Khrisna, K. Lin and J. Sun, "A Coding Theory Approach to Error Control in Redundant Residue Number Systems – Part I: Theory and Single Correction", *IEEE Trans. on Circuits and Systems, II*, Vol. 39, No. 1, Jan. 1992, pp. 8-17.

[10] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes, Parts I and II*, North-Holland, New York, 1977.