System Evaluation of Disk Allocation Methods for Cartesian Product Files by using Error Correcting Codes

Shigeichi Hirasawa, Tomohiko Saito, Hiroshige Inazumi and Toshiyasu Matsushima

Abstract—We discuss disk allocation methods for Cartesian product files by introducing error correcting codes, and have clarified the performance of the methods by system evaluation models developed by using rate distortion theory. Let us assume $q^n$ Cartesian product files with $n$ attributes and $q$ actual values in each attribute, and store $q^n$ files into $G(\leq q^n)$ disks. For a partial match access request, we represent new disk allocation methods which able to access the disks in parallel as much as possible, where the partial match access request includes an indefinite case (don’t care: "*") in some attributes and the * requires to access the files for corresponding attribute to the all actual attribute values. In this paper, we propose to apply unequal error protection codes to the case where the probabilities of occurrence of the * in the attributes for a partial match access request are not the same. We show the disk allocation methods have desirable properties as $n$ becomes large.

Index Terms—disk allocation, Cartesian product files, error correcting codes, unequal error protection codes, system evaluation model, flexible, elastic, rate distortion theory, Chernoff bound

I. INTRODUCTION

In the latter decade of 1970’s, J. Pearl and A. Crozott have discussed on the trade-off between amount of memory and error in QA (Question Answering) systems based on rate-distortion theory [9]. They clarified the conditions such that we can reduce the amount of memory, if the small error rate can be tolerated. These conditions are called “flexible” or “elastic”. We have extended these conditions to be more useful by generalized trade-off evaluation model, called system evaluation model [4], [5] and have applied it to various information systems [6], [7]. By using this model, we can decide whether the system has “effective elastic” condition or not, where the “effectively elastic” implies the relatively effective property as the system size becomes large.

On the other hand, it is well known that the structure of linear error correcting codes can be used to experimental design [3], and disk allocation for files [1]. Let us assume the $q^n$ Cartesian product files with $n$ attributes and $q$ actual values in each attribute, and store $q^n$ files into $G(\leq q^n)$ disks. For a partial match access request (PMAR), we intend to access the disks in parallel as much as possible by using disk allocation methods, where the PMAR includes an indefinite case (don’t care: "*") in some attributes and the * requires to access the files for corresponding attribute to the all actual attribute values.

In this paper, first we apply the system evaluation model to the disk allocation methods, and discuss whether the disk allocation methods satisfy the (effective) elastic condition or not. Next, we extend them into the case where the probability of the occurrence of the * at $i$-th attribute $X_i$, $Pr(z_i = \star)$ is not uniform for $i$, where $z_i$ is the $i$-th actual attribute value. We propose new disk allocation methods based on unequal error protection (UEP) codes for PMAR, and we show that the proposed methods are superior compared to those based on equal error protection (EEP) codes [1] from the view-point of the flexible condition.

II. PRELIMINARIES

A. Outline of Rate-Distortion Theory

Rate-distortion theory discusses on information compression by the trade-off property between rate and distortion [2]. The rate-distortion function can be written by:

$$R = R(D),$$

where $R$ is the rate defined by $R = (1/n) \log M$, and $M$ is the number of code words, $n$, the code length, and $D$, the distortion. The $R = R(D)$ is usually a convex downward and non-increasing function of $D$. The function $R = R(D)$ suggests us that we can decrease the rate drastically with tolerating a slightly growth of the distortion by proper source encoding.

B. System Evaluation Model

Generally, the rate $R$ discussed in the previous subsection corresponds to the cost of a system, and distortion $D$, degradation of the performance of the system [7]. By extending

*We use “EEP codes” to show the conventional error correcting codes to contrast with UEP codes, although we have no such technical term.
the trade-off evaluation model, we have proposed the system evaluation model [5], where we have introduced a variable \( n \) as the system size.

Let the rate \( R \) be normalized by the maximum of \( R, R_{\text{max}} \), and the distortion \( D \) by the maximum of \( D, D_{\text{max}} \), then we have \( r = R/R_{\text{max}} \) and \( d = D/D_{\text{max}} \), and the normalized function introducing \( n \):

\[
r = r(d, n).
\]

For evaluation of the systems, we define the following properties to the (normalized) system evaluation function (2):

**Definition 1:**

1) **Flexible** [9]: The system A with \( r = r_A(d, n) \) is flexible compared to the system B with \( r = r_B(d, n) \), if \( r_A(d, n) < r_B(d, n) \) for arbitrary \( d (0 < d < 1) \), and \( n (n > 1) \).
2) **Elastic** [9]: The system with \( r = r(d, n) \) is elastic, if \( r = r(d, n) \) is a decreasing function of \( n \) for arbitrary \( d (0 < d < 1) \).
3) **Effective elastic** [5]: The system is effective elastic, if the system is elastic and \( r = r(d, n) \) is a convex downward function of \( n \).
4) **Trivial elastic** [9]: The system with \( r = r(d, n) \) is trivial elastic, if \( d = d(0, n) \) is a decreasing function of \( n \), where \( d = d(r, n) \) is the inverse function of \( r = r(d, n) \).
5) **Marginal elastic** [5]: The system with \( d = d(r, n) \) is marginal elastic, if \( d = d(0, n) \) is a convex downward function of \( n \).

4) Trivial elastic and 5) Marginal elastic are sometimes observed depending on the structure of systems (See [6]).

### III. Disk Allocation of Cartesian Product Files

#### A. Cartesian Product Files

Let a set of attributes be denoted by \( X_1, X_2, \ldots, X_n \), where an actual attribute value of \( X_i \) \((i = 1, 2, \ldots, n)\) is given by a domain \( Z_i = \{0, 1, 2, \ldots, q - 1\} \). Then the Cartesian product files are constructed by \( q^n \) buckets, and each bucket is specified by \( n \)-tuple \((z_1, z_2, \ldots, z_n)\), where \( z_i \in Z_i = \{0, 1, 2, \ldots, q - 1\} \).

#### B. Partial Match Access Request

A partial match access request (PMAR) \( Q \) to \( q \)-ary Cartesian product file is given by:

\[
Q = (X_1 = z_1, X_2 = z_2, \ldots, X_n = z_n),
\]

where \( z_i \in \{0, 1, 2, \ldots, q - 1\} \). The symbol * shows an indefinite value, hence * \( \in \{0, 1, 2, \ldots, q - 1\} \). This implies that the * at the attribute \( X_i \) of \( Q \) requires to access the files with all the actual attribute values in \( Z_i \). Thus all attributes are specified except for \( X_i \), then we call such access request, PMAR.

**Example 1:** (PMAR) Letting \( q = 2 \) and \( n = 4 \), an example of the Cartesian product files is shown in Table I. If the PMAQ is given by:

\[
Q = (0, 0, *, 1),
\]

then we must access the buckets of \((0, 0, 0, 1)\) and \((0, 0, 1, 1)\), since \( X_3 \) of \( Q \) requires don’t care (whether married or not as shown in Table I).

#### C. Disk Allocation Methods

Let us consider a disk allocation problem of the Cartesian product files. The problem is to partition the \( q^n \) buckets into \( G (G \leq q^n) \) disks so that we can access disks in parallel to simultaneously different buckets. This problem can be effectively solved by standard array appeared in coding theory.

**Example 2:** (Standard array) Letting \( q = 2, n = 6, G = 8, \) and \( q^n = 64 \), a construction method obtained by standard array is shown in Table II. We easily see that the following PMAR \( Q \):

\[
Q = (0, *, 1, *, 0, 0)
\]

can be accessed to the all required buckets of \((0, 0, 1, 0, 0, 0), (0, 0, 1, 0, 0, 0), (0, 1, 1, 0, 0, 0), \) and \((0, 1, 1, 0, 0, 0)\) in parallel at once by Table II.

Let a set of the buckets required by PMAQ \( Q \) be \( S(Q) \). If \( z_i = * \), and \( z_j = * \), \((i \neq j)\), we hope to access simultaneously \( q^2 \) disks in parallel so that we can decrease the access time. Obviously, the maximum value of \( G \), \( G_{\text{max}} = q^n \), and the minimum value of \( G \), \( G_{\text{min}} = 1 \). It is known that coding theory gives the following Lemma by using \( q \)-ary \((n, k, d)\) code \( C \), where \( n \) is the code length, \( k \), the number of information symbols, and \( d \), minimum distance.

**Lemma 1:** [1] Let the number of the * occurred in \( Q \) be \( w \) \((0 \leq w \leq n)\). If \( 0 \leq w < d \), then a disk allocation method based on a \( q \)-ary \((n, k, d)\) code is the optimum. \( \square \)

**Lemma 2:** [1] Let the number of the * occurred in \( Q \) be \( w \) \((0 \leq w \leq n)\). If \( 0 \leq w < d \), then a disk allocation method based on a \( q \)-ary \((n, k, d)\) code is the optimum. \( \square \)

**Lemma 1** states that the \( q \)-ary \((n, k, d)\) code can give the method for accessing the \( q^n \) buckets in parallel at once, if \( w < d \).

### IV. Evaluation of Disk Allocation Methods

#### A. Formulation of Disk Allocation Methods

Let the \( q^n \) bucket be stored in \( q^{n-k} \) disks \(^2\) by using an \((n, k, d)\) code \( C \). For given PMAQ \( Q \), we let the number of access time to disks for \( S(Q) \) be \( J \). While we let a set of bucket accessible by using code \( C \) with \( J = 1 \) be \( S(C) \). Then we give the evaluation loss (distortion measure) as the following definition.

**Definition 2:** The evaluation loss \( \rho \) is given by:

\[
\rho = \begin{cases} 
0 & (J = 1) \\
1 & (J \geq 2). 
\end{cases}
\]

\( ^2\)Note that the number of disks \( G \) equals to the number of coset of codes \( C \).
Note that Substitution of (10) into (7) gives \( S \).

From Lemma 1, we have

Let us consider a case where \( \Pr(z_i = *) \) is distinct of \( i \). Then in such case, the following unequal error protection (UEP) codes [8], [12], [11] can play an important role to construct the disk allocation methods. We illustrate a UEP code \( C_u \) in Fig. 1, where \( \Pr(z_i = *) = p_1 \) for \( i = 1, 2, \cdots, n_1 \),

\[
\Pr(z_j = *) = p_2 \quad \text{for} \quad j = n_1 + 1, n_1 + 2, \cdots, n_2, \quad \text{and} \quad n = n_1 + n_2. 
\]

A simplest case as shown in Fig. 1 is called the 2-split UEP code, where \( n_1 \) \( n_2 \), \( d_1 \) \( d_2 \), and \( p_1 \) \( p_2 \) are the code length, the minimum distance, and the probability of occurrence of the * in the 1st part (2nd part) of the UEP code \( C_u \), respectively, and \( M \) is the number of the code words, i.e. the rate \( R \) is given by \( R = (1/n) \log M \).

Lemma 2: [8] The \([n_1, n_2, M, (d_1, d_2)]\) UEP code can access the disks with \( J = 1 \) as follows:

1. When \( w_1 = 0 \), then \( J = 1 \) if \( w_2 < d_2 \).
2. When \( w_1 \geq 1 \), then \( J = 1 \) if \( w_1 + w_2 < d_1 \).

From Lemma 2, we have the following theorem.

**Theorem 1:** Suppose a set of buckets \( S(C_u) \) accessible to the disks with \( J = 1 \) using the code \( C_u \). Then the probability of the access time with \( J \geq 2 \) satisfies

\[
\Pr(J \geq 2) \leq \Pr(w_1 = 0) \Pr(w_2 \geq d_2) + \sum_{s=1}^{n_1} \Pr(w_1 = s) \Pr(w_2 \geq d_1 - s),
\]

where \( w_1 \) \( w_2 \) is the number of the * in the 1st part (2nd part) of the UEP code \( C_u \).

**Proof:** Using the inverse of the sufficient conditions stated in 1), and 2) in Lemma 2, we can easily derive (11).

Substitution of (11) into (7) gives \( \nu \).

**D. Calculation for Evaluation**

In the previous subsection, we derive the access performance \( \nu \), and the cost \( g \). Note that \( \nu \) and \( g \) can be given by the functions of parameters \( (n, k, d) \) for a code \( C \), and \([n_1, n_2, M, (d_1, d_2)]\) for a UEP code \( C_u \). We then simply denote the functions by \( \nu = \nu(d, n) \), and \( g = g(k, n) \) for the code \( C \), and by \( \nu = \nu(d_1, d_2, n_1, n_2) \), and \( g = g(M, n) \) for the code \( C_u \).

To compute the functions \( \nu \) and \( g \), we can use

- The LP upper bound [12]: \( M \leq f(d_1, d_2, n_1, n_2) \)
- The Gilbert lower bound [10]: \( d/n \leq H^{-1}(1 - R), n \to \infty \)
- Constructive codes such as BCH codes and RS codes.

\[\text{The LP upper bound gives an upper bound on } M \text{ (or } R) \text{ solving the existence area of a code } C \text{ and } C_u \text{ by linear programming for given code parameters. Hence the computed results are given by a table.}\]
where $R = k/n$, or $R = (1/n) \log M$ holds. We have no bound up to now corresponding to the Gilbert lower bound for codes $C_u$. On the other hand, the LP upper bound can apply to both codes $C$ and codes $C_u$. Note that the LP upper bound gives the lower bound on $g$, since $g = 1/M$, and similarly the Gilbert lower bound, the upper bound on $g$, since $g = q^{-k}$.

V. NUMERICAL RESULTS

A. Binomial Distribution (BD)

Assuming that the probability of occurrence of the * in a disk allocation method constructed by the code $C$ or the code $C_u$ is given by the binomial distribution (BD) function, we have the following results (See Appendix A):

1) Cases by (EEP) codes $C$: Fig. 2 shows a case of $p = 0.5$, where we have used the LP upper bound. The difference between the LP upper bound and the Gilbert lower bound is depicted in Fig. 3 for a case of $p = 0.3$, and $n = 10$.

2) Cases by (UEP) codes $C_u$: Fig. 4 shows a case of $p_1 = 0.5$, and $p_2 = 0.25$, where we have used the LP upper bound. The difference between codes $C$ and codes $C_u$ is illustrated in Fig. 5 for a case of $p_1 = 0.5$, and $p_2 = 0.25$. Fig. 6 shows $g$ as a function of $n$ using the LP upper bound by codes $C$ and by codes $C_u$.

B. Chernoff Bound (CB)

To compute the access performance $\nu$, we use the Chernoff bounding (CB) techniques which are known in information theory [2] to evaluate coding systems. We have the following results (See Appendix B) without assuming any probability distribution function of occurrence of the *:

V. NUMERICAL RESULTS

A. Binomial Distribution (BD)

Assuming that the probability of occurrence of the * in a disk allocation method constructed by the code $C$ or the code $C_u$ is given by the binomial distribution (BD) function, we have the following results (See Appendix A):

1) Cases by (EEP) codes $C$: Fig. 2 shows a case of $p = 0.5$, where we have used the LP upper bound. The difference between the LP upper bound and the Gilbert lower bound is depicted in Fig. 3 for a case of $p = 0.3$, and $n = 10$.

2) Cases by (UEP) codes $C_u$: Fig. 4 shows a case of $p_1 = 0.5$, and $p_2 = 0.25$, where we have used the LP upper bound. The difference between codes $C$ and codes $C_u$ is illustrated in Fig. 5 for a case of $p_1 = 0.5$, and $p_2 = 0.25$. Fig. 6 shows $g$ as a function of $n$ using the LP upper bound by codes $C$ and by codes $C_u$.

B. Chernoff Bound (CB)

To compute the access performance $\nu$, we use the Chernoff bounding (CB) techniques which are known in information theory [2] to evaluate coding systems. We have the following results (See Appendix B) without assuming any probability distribution function of occurrence of the *

V. NUMERICAL RESULTS

A. Binomial Distribution (BD)

Assuming that the probability of occurrence of the * in a disk allocation method constructed by the code $C$ or the code $C_u$ is given by the binomial distribution (BD) function, we have the following results (See Appendix A):

1) Cases by (EEP) codes $C$: Fig. 2 shows a case of $p = 0.5$, where we have used the LP upper bound. The difference between the LP upper bound and the Gilbert lower bound is depicted in Fig. 3 for a case of $p = 0.3$, and $n = 10$.

2) Cases by (UEP) codes $C_u$: Fig. 4 shows a case of $p_1 = 0.5$, and $p_2 = 0.25$, where we have used the LP upper bound. The difference between codes $C$ and codes $C_u$ is illustrated in Fig. 5 for a case of $p_1 = 0.5$, and $p_2 = 0.25$. Fig. 6 shows $g$ as a function of $n$ using the LP upper bound by codes $C$ and by codes $C_u$.

B. Chernoff Bound (CB)

To compute the access performance $\nu$, we use the Chernoff bounding (CB) techniques which are known in information theory [2] to evaluate coding systems. We have the following results (See Appendix B) without assuming any probability distribution function of occurrence of the *:

V. NUMERICAL RESULTS

A. Binomial Distribution (BD)

Assuming that the probability of occurrence of the * in a disk allocation method constructed by the code $C$ or the code $C_u$ is given by the binomial distribution (BD) function, we have the following results (See Appendix A):

1) Cases by (EEP) codes $C$: Fig. 2 shows a case of $p = 0.5$, where we have used the LP upper bound. The difference between the LP upper bound and the Gilbert lower bound is depicted in Fig. 3 for a case of $p = 0.3$, and $n = 10$.

2) Cases by (UEP) codes $C_u$: Fig. 4 shows a case of $p_1 = 0.5$, and $p_2 = 0.25$, where we have used the LP upper bound. The difference between codes $C$ and codes $C_u$ is illustrated in Fig. 5 for a case of $p_1 = 0.5$, and $p_2 = 0.25$. Fig. 6 shows $g$ as a function of $n$ using the LP upper bound by codes $C$ and by codes $C_u$.

B. Chernoff Bound (CB)

To compute the access performance $\nu$, we use the Chernoff bounding (CB) techniques which are known in information theory [2] to evaluate coding systems. We have the following results (See Appendix B) without assuming any probability distribution function of occurrence of the *:

V. NUMERICAL RESULTS

A. Binomial Distribution (BD)

Assuming that the probability of occurrence of the * in a disk allocation method constructed by the code $C$ or the code $C_u$ is given by the binomial distribution (BD) function, we have the following results (See Appendix A):

1) Cases by (EEP) codes $C$: Fig. 2 shows a case of $p = 0.5$, where we have used the LP upper bound. The difference between the LP upper bound and the Gilbert lower bound is depicted in Fig. 3 for a case of $p = 0.3$, and $n = 10$.

2) Cases by (UEP) codes $C_u$: Fig. 4 shows a case of $p_1 = 0.5$, and $p_2 = 0.25$, where we have used the LP upper bound. The difference between codes $C$ and codes $C_u$ is illustrated in Fig. 5 for a case of $p_1 = 0.5$, and $p_2 = 0.25$. Fig. 6 shows $g$ as a function of $n$ using the LP upper bound by codes $C$ and by codes $C_u$.
bound on effective elastic, or flexible, although upper bound when codes $C$ is better to construct a disk allocation method based on codes than those of the latter. This result implies that it compared to that by codes $C$, since the curves of the former are lower than those of the latter. This result implies that it is better to construct a disk allocation method based on codes $C_u$ than codes $C$ for given parameters $p_1$ and $p_2$.

3) Bounding techniques: We use the bounding results from coding theory given by the LP upper bound and the Gilbert bound. These two bounds guarantee the existence of disk allocation methods as shown by shaded area in Fig. 3. We also use Chernoff bounding techniques for calculating $\nu$ without assuming the probability distribution for the occurrence of the *. On the other hand, assuming the binomial distribution for it, the difference between them is depicted in Figs. 8 and 9 for codes $C$. We see that the Chernoff bound is a good upper bound on $g$ enough to show the properties such as elastic, effective elastic, or flexible, although $n$ is relatively small, and $p_1$ and $p_2$ are relatively large $^6$.

VI. CONCLUDING REMARKS

In our another paper [11], we have precisely analyzed the performance of disk allocation methods from view-point of the average access time and the number of disks. In this paper, our approach is slightly different compared to [11], since we are interested in an application of system evaluation model. As the results obtained by the system evaluation model of the disk allocation methods constructed by error correcting codes, we can conclude that the $g = g(\nu, n)$ is:

(i) an elastic function
(ii) an effective elastic function

$^6$It is known that the Chernoff bound is always valid, and is usually tight upper bound when $n$ becomes large and $p$, $p_1$, and $p_2$ are small [2].

and

(iii) if the occurrence of the * is not uniform, disk allocation methods constructed by UEP codes $C_u$ are flexible compared to those by codes $C$.

The (i) states that tolerating a small access performance degradation in $\nu$ introduces a drastic saving of the cost $g$, and this property can be effectively enhanced by the number of attribute $n$ becomes large as seen in (ii). We can also remark that the UEP codes $C_u$ are useful for the disk allocation methods if the probabilities of occurrences of the * are not uniform as stated in (iii).

We note that the LP upper bound can give the case of $L(>2)$-split UEP codes, hence we can discuss the disk allocation methods in general. Furthermore, although we have only discuss in this paper on cases for $q = 2$, cases for $q \geq 3$ are straight forward, since we already have $q$-ary codes $C$ and codes $C_u$, and their upper bounds, where $q(\geq 2)$ is a prime power.

To compute a function such as $g = F(\nu, n)$ using the Chernoff bounding techniques by analytically and to show the effective elastic directly will be rest as a further research.

ACKNOWLEDGMENT

One of the present authors, S. H. would like to thank Professor S. Oishi of Waseda University for giving him a chance to submit this paper. The authors also thank all the members of Matsusima Lab. and Goto Lab. of Waseda University for their fruitful comments.
APPENDIXES

A. Derivation by Binomial Distribution

By the \((n, k, d)\) code \(C\), we have from (10):

\[
\nu = \Pr(J \geq 2) \leq \Pr(w \geq d) = \sum_{i=d}^{n} \binom{n}{i} p^i (1-p)^{n-i},
\]

and

\[
g = q^{-k}.
\]

While by the \([(n_1, n_2), M, (d_1, d_2)]\) code \(C_u\), we have from Theorem 1, (11):

\[
\nu = \Pr(J \geq 2) \leq (1 - p_1)^{n_1} \sum_{i=d_1}^{n_2} \binom{n_2}{i} p_2^i (1-p_2)^{n_2-i} \leq \exp[-nE(\lambda, p)],
\]

where

\[
E(\lambda, p) = -\lambda \ln p - (1 - \lambda) \ln(1-p) - H(\lambda),
\]

\[
H(x) = -x \ln x - (1 - x) \ln(1-x),
\]

and

\[
\lambda = \frac{d}{n}, \quad \lambda > p.
\]

B. Derivation by Chernoff Bound [10]

By the \((n, k, d)\) code \(C\), we have from (10):

\[
\nu = \Pr(J \geq 2) \leq \Pr(w \geq d) \leq \exp[-nE(\lambda, p)],
\]

where

\[
\lambda_{a,1} = \frac{d_2}{n_2}, \quad 1 \geq \lambda_{a,1} > p_2,
\]

\[
\lambda_{b,1} = \frac{d_1 - s}{n_2}, \quad 1 \geq \lambda_{b,1} > p_2, \quad (s = 1, 2, \ldots, \alpha)
\]

\[
\lambda_{b,2} = \frac{s}{n_1}, \quad 1 \geq \lambda_{b,2} > p_1, \quad (s = \alpha + 1, \ldots, n_1)
\]

\[
\lambda_{b,3} = \frac{d_1 - s}{n_2}, \quad 1 \geq \lambda_{b,3} > p_2, \quad (s = \alpha + 1, \ldots, n_1)
\]

and \(\alpha = n_1 p_1\).

REFERENCES


