# Algorithms for Mining Association Rules for Binary Segmentations of Huge Categorical Databases

Yasuhiko Morimoto Takeshi Fukuda Hirofumi Matsuzawa
Takeshi Tokuyama Kunikazu Yoda

IBM Tokyo Research Laboratory
1623-14, Shimo-tsuruma, Yamato, Kanagawa 242-0001, Japan

{morimoto,fukudat,matuzawa,ttoku,yoda}@trl.ibm.co.jp

## Abstract

We consider the problem of finding association rules that make nearly optimal binary segmentations of huge categorical databases. The optimality of segmentation is defined by an objective function suitable for the user's objective. An objective function is usually defined in terms of the distribution of a given target attribute. Our goal is to find association rules that split databases into two subsets, optimizing the value of an objective function.

The problem is intractable for general objective functions, because letting $N$ be the number of records of a given database, there are $2^N$ possible binary segmentations, and we may have to exhaustively examine all of them. However, when the objective function is convex, there are feasible algorithms for finding nearly optimal binary segmentations, and we prove that typical criteria, such as "entropy (mutual information)," "$\chi^2$ (correlation)," and "gini index (mean squared error)," are actually convex.

We propose practical algorithms that use computational geometry techniques to handle cases where a target attribute is not binary, in which conventional approaches cannot be used directly.

**Proceedings of the 24th VLDB Conference**
**New York, USA, 1998**

## 1 Introduction

In recent years, data mining has made it possible to discover unexpected and valuable rules by analyzing huge databases. Efficient algorithms for finding association rules [AIS93, AS94, HF95, PS91, FMMT96c, FMMT96b] and classification and regression trees that use these rules as branching tests [BFOS84, Qui86, Qui93, MFMT97] have been widely studied in both database and machine learning literature.

In this paper, we focus on finding association rules that can be used for classifiers such as decision trees. Association rules, which make binary segmentations for classifying a target attribute, themselves provide us with valuable information for understanding the target attribute, and can therefore be used for various analytical purposes.

### Optimal Binary Segmentation Problem

In this paper, we consider only databases, which we call *categorical databases*, whose attributes are all categorical. When we want to deal with a database with numeric attributes, we include a preprocess to discretize numeric attributes into categorical ones.

Let $R$ be a database relation. We treat one attribute of the relation as special, and call it a *target attribute*. Other attributes of the relation are called *conditional attributes*. Let $A$ be the target attribute, $\mathrm{dom}(A) = \{a_1, a_2, \cdots, a_k\}$ be the domain of $A$, and $k$ be the *target domain size* (the number of distinct values). Let $x_i(S)$ denote the number of records in $S \subseteq R$ for which the value of the target attribute $A$ is $a_i$.

Table 1: mRNA Sequence

| N1 | N2 | N3 | ⋯ | Amino Acid |
|---|---|---|---|---|
| U | U | U | ⋯ | Phe |
| C | A | C | ⋯ | His |
| G | G | C | ⋯ | Gly |
| G | G | A | ⋯ | Gly |
| A | C | A | ⋯ | Thr |
| G | A | A | ⋯ | Glu |
| U | U | C | ⋯ | Phe |
| ⋯ | ⋯ | ⋯ | ⋯ | ⋯ |

Let $C_1, C_2, \cdots, C_M$ be the conditional attributes. We treat these attributes as a single attribute $C$ whose domain is the Cartesian product of their domains $(\text{dom}(C_1) \times \text{dom}(C_2) \times \cdots \times \text{dom}(C_M))$. If $C_i$ $(i = 1, ..., M)$ has $n_i$ distinct values, the *conditional domain size* of $C$ is $n = \prod_i n_i$ for $1 \leq i \leq M$.

To make a binary segmentation of $R$, we use a *test* on the conditional attribute $C$. $T \subseteq \text{dom}(C)$ is a test that divides the database relation $R$ into two segments $S = \{t \in R \mid t[C] \in T\}$ and $\bar{S} = \{t \in R \mid t[C] \notin T\}$. We say "$T$ splits $R$ into $(S; \bar{S})$". Our ideal goal is to find, among all possible tests, a test $T$ that maximizes (or minimizes) an objective criterion $f(S; \bar{S})$.

The objective function should be an information-theoretic function such as the "gini index," "entropy," or "$\chi^2$," which can be calculated from the distribution of values of the target attribute.

We call the following rule "the optimal association rule for a binary segmentation"

IF ($T$ is satisfied)
THEN (Segment1) ELSE (Segment2),

where Segment2 is the complement of Segment1.

We remark that, in the data mining literature [AIS93, AS94, PCY95, FMMT96c, FMMT96b], association rules that are collected on the basis of "support" and "confidence" values are also popular. However, we focus on association rules that optimize an information-theoretic function, since they are easier to use as parts of decision systems such as decision trees [BFOS84, Qui86, Qui93].

## A Naive Approach

An important problem in data mining is how to find the optimal association rule for binary segmentations of a database. If the conditional domain size is large, this problem is highly challenging.

Table 1 shows an example of a categorical database. The table shows the first three substances in an "mRNA" sequence and its corresponding amino acid.

Finding the optimal rule for classifying the amino acid gives us a clue to understand the relationship between RNA patterns and amino acids. In this example, we can finally find rules such as "`if N1=U and N2=U then Phe`," "`if N1=G and N2=G then Gly`," and so forth. (Note that these rules are just examples. The relationship between RNA patterns and amino acids has already been investigated in the biology literature.) The "mRNA" sequence is known to consist of four types of substance, denoted "U," "C," "A," and "G." Even if we only consider the first three substances, there are $4^3$ permutations. Therefore, we have to examine $2^{64-1}$ possible binary segmentations in this case to find the optimal rule with respect to the types of amino acid.

In general, if the conditional domain size is $n$, there are $2^{n-1} - 1$ possible binary segmentations. Hence, a naive exhaustive search for the optimal binary segmentation requires $O(2^n)$ time, which is not practical.

## Related Work

For the case where the target domain size is two, i.e., $k = 2$, we can order the $n$ values so that the optimal splitting is one "cut" of the ordered sequence, if we can assume convexity of the objective criterion [BFOS84] (and all of the mentioned criteria have such a property). Consequently, we have an $O(n \log n)$ algorithm. However, this algorithm is not applicable to cases in which the target domain size is greater than two.

For huge categorical databases, in which the conditional domain size is large and $k > 2$ , there is no practical existing algorithm that can find the optimal association rule. Despite the difficulty, there are some heuristics for handling the problem [BFOS84, MP91, Cen92, Qui93] that are used in practice for constructing decision trees.

A heuristics called "two-ing" [BFOS84] divides the target domain into two classes (called superclasses), and applies the $O(n \log n)$ algorithm for $k = 2$ to create the optimal subdivision for each of the $2^{k-1}$ possible divisions into superclasses, and finds the best one among them. This runs in $O(2^{k-1} n \log n)$ time, which is efficient for a small $k$.

Another heuristic called "value groups" [Qui93] greedily merges two conditional values from the conditional domain to reduce the conditional domain size to $n - 1$, so that the objective function is maximized (or minimized). It repeats this greedy merging process until $n = 2$ and finally returns the final two groups. This $O(n^3)$ heuristic can be used even if $k$ is large.

The above heuristics are known to be practical for constructing decision trees. However, neither of them has a guarantee on the optimality of the result. From the data mining point of view, our interest is not binary segmentation itself but unexpected and important rules that significantly affect the value of a target attribute. In this sense, we want to find more, hopefully the most, significant rules. Therefore, it is important to generate the optimal or near-optimal association rules of the form "itemset $\vee$ itemset $\vee \cdots$" that affect a target attribute.

**Main Results**

In this paper, we propose two algorithms named the Random Algorithm and the Probing Algorithm, that we designed by using computational geometry techniques. The algorithms have the following features:

- Both of the algorithms can feasibly compute nearly optimal tests for cases in which $k$ is a small constant. The domain size $n$ is allowed to be huge.
- The Random Algorithm finds nearly optimal tests, using random sampling. The time complexity becomes $O(s^{k-2}n)$, where $s(\ll n)$ is the sample size. Moreover, this algorithm can run with a small working space even if the target domain size becomes larger.
- We can use the same geometrical interpretation of all possible tests as we presented in [FMMT96a, MFMT97], and translate those tests into points in a $k$-dimensional space. Then, we consider how to find a point on convex hull of all points in the $k$-dimensional space. We used the guided branch-and-bound search algorithm for finding the optimal region in [FMMT96a, MFMT97]. The total cost of the incremental convex hull maintenance and probing of the convex hull in the algorithm is $O((n+m)|P|)$, if we have $m$ points and $|P|$ facets on the convex hull. Although $m$ and $P$ can be asymptotically as large as $n^{k-1}$ and $m^{\lfloor \frac{k}{2} \rfloor}$ respectively, in a pathological input, they are known to be much smaller in a normal input. Even so, the convex hull searching algorithm itself cannot be used within a working space of the size available in ordinary workstations, if $k > 4$. If we consider the binary segmentation problem for huge categorical databases, $k$ often becomes larger and the number of possible tests may become huge. Therefore, we need significant improvement with respect to time and working space to make the algorithm feasible for the problem.

  The Probing Algorithm improves the hull maintenance so that it works within a limited working space, and we improve the searching strategies so that we can find nearly optimal tests in an earlier step of the algorithm. We can report the current best solution found so far at every incremental step of the algorithm, so that the solution gradually converges to the optimal. In cases where a quick response is required, the Probing Algorithm can return the best test found in the required time, and is suitable for online applications.

We experimentally compare the performance of these algorithms.

## 2 Optimization Problem of Binary Segmentation

### 2.1 Stamp Point

For a segment $S$, let $x_i(S)$ be the number of data records in $S$ for which the value of the target attribute is $a_i$. Thus, each segment $S$ of the relation $R$ can be mapped to a point $\mathbf{x}(S) = (x_1(S), x_2(S), \cdots, x_k(S))$ in the $k$-dimensional Euclidean space, which is referred to as a *stamp point* of $S$. A stamp point represents the distribution of the target attribute of interest, and our objective functions of binary segmentations, $f(S;\bar{S})$, are defined in terms of $\mathbf{x}(S)$ or $\mathbf{x}(\bar{S})$. If a test $T$ splits $R$ into $(S;\bar{S})$, we also refer to the stamp point of $S$ as the stamp point of $T$ (i.e., $\mathbf{x}(T) = \mathbf{x}(S)$).

### 2.2 Criteria for Segmentation

The significance of discovered rules depends on the user's objective, and hence there is no universal criterion for measuring the significance of rules. A useful segmentation should divide data into segments whose target distribution is more skewed than that of the data as a whole. Therefore, we often use criteria such as the "gini index," "entropy gain," and "$\chi^2$," which indicate the extent to which the divided data distributions differ from the original data distribution.

In this subsection, we describe objective criteria commonly used for evaluating data segmentation and their corresponding functions, which are defined in terms of a stamp point of a binary segmentation.

**Gini Index (Mean Squared Error)**

The implication of this criterion is "how much the mean squared error of target values is decreased." The optimal segmentation according to this criterion minimizes the mean squared error. It is defined as follows, where $p_i(S) = x_i(S)/|S|$.

$$\begin{aligned} Gini(\mathbf{x}(S)) &= Gini(S;\bar{S}) \\ &= \left(1 - \textstyle\sum_{i=1}^{k} p_i(R)^2\right) \\ &\quad - \tfrac{|S|}{|R|}\left(1 - \textstyle\sum_{i=1}^{k} p_i(S)^2\right) \\ &\quad - \tfrac{|\bar{S}|}{|R|}\left(1 - \textstyle\sum_{i=1}^{k} p_i(\bar{S})^2\right) \end{aligned}$$

**Entropy (Mutual Information)**

The entropy gain function compares the mutual information gained by a rule.

$$\begin{aligned} Ent(\mathbf{x}(S)) &= Ent(S;\bar{S}) \\ &= -\textstyle\sum_{i=1}^{k} p_i(R)\log p_i(R) \\ &\quad +\tfrac{|S|}{|R|}\textstyle\sum_{i=1}^{k} p_i(S)\log p_i(S) \\ &\quad +\tfrac{|\bar{S}|}{|R|}\textstyle\sum_{i=1}^{k} p_i(\bar{S})\log p_i(\bar{S}) \end{aligned}$$

**$\chi^2$ (Correlation)**

A segmentation with a high $\chi^2$ value is significant, since the statistical hypothesis that "$S$ and $\bar{S}$ are not different from $R$" is strongly denied.

$$Chi(\mathbf{x}(S)) \;=\; Chi(S;\bar{S})$$

$$= \sum_{i=1}^{k} \frac{|S|(p_i(S)-p_i(R))^2 + |\bar{S}|(p_i(\bar{S})-p_i(R))^2}{p_i(R)}$$

**Example 2.1** To illustrate our optimal rule, we consider binary segmentations of the following data using the "gini index." In these data, a target attribute has three values, say $a_1$, $a_2$, and $a_3$, and each value has 40, 30, and 30 records, respectively.

| $S$ | $a_1$ | $a_2$ | $a_3$ |
|---|---|---|---|
| 100 | 40 | 30 | 30 |

The gini index, $Gini(S)$, of a certain dataset $S$ with respect to a target attribute is defined as

$$Gini(S) = 1 - \sum_j p_j^2,$$

where $p_j$ is the relative frequency of the $j$-th target value in $S$. In this example, the gini index value is

$$1-(\frac{40}{100})^2-(\frac{30}{100})^2-(\frac{30}{100})^2 = 0.66.$$

Let us consider a binary segmentation that divides $S$ into $S_1$ and $S_2$, containing $n_1$ and $n_2$ records, respectively. The gini index value of the binary segmentation is defined as

$$\begin{aligned} Gini(S_1;S_2) &= Gini(S) \\ &\quad -\tfrac{n_1}{n_1+n_2}Gini(S_1) - \tfrac{n_2}{n_1+n_2}Gini(S_2). \end{aligned}$$

If we assume the following segmentation for the above example, the stamp point for the test is a point (40, 10, 10) in three-dimensional space, and the gini index value of the segmentation is 0.16.

| $S_1$ | $a_1$ | $a_2$ | $a_3$ |
|---|---|---|---|
| 60 | 40 | 10 | 10 |

| $S_2$ | $a_1$ | $a_2$ | $a_3$ |
|---|---|---|---|
| 40 | 0 | 20 | 20 |

Let us consider another segmentation:

| $S_1$ | $a_1$ | $a_2$ | $a_3$ |
|---|---|---|---|
| 60 | 20 | 20 | 20 |

| $S_2$ | $a_1$ | $a_2$ | $a_3$ |
|---|---|---|---|
| 40 | 20 | 10 | 10 |

In this case, the value of the gini function of the stamp point (20, 20, 20) is only 0.01.

The optimal association rule for binary segmentation with respect to the gini index is the one that splits data into two segments, say "$S_1$" and "$S_2$," such that $Gini(S_1;S_2)$ is maximized among all possible segmentations.

## 2.3 Convexity of the objective functions

If we consider objective functions that are defined in terms of a stamp point in $k$-dimensional Euclidean space, the following property can be used in searching for nearly optimal points and their corresponding tests.

**Definition 2.1** $f(\mathbf{x})$ is convex if

$$max\{f(\mathbf{x_1}), f(\mathbf{x_2})\} \geq f((1-\gamma)\mathbf{x_1} + \gamma\mathbf{x_2})$$

for $0 \leq \gamma \leq 1$ and arbitrary points $\mathbf{x_1}$ and $\mathbf{x_2}$ in the domain of $f$.

**Lemma 2.1** The gini index, entropy gain, and $\chi^2$ are all convex functions on $\mathbf{x}$.

**Proof:** For any vector $\mathbf{\Delta} \neq 0$, the second derivative of the functions along with $\mathbf{\Delta}$ is non-negative.

We have shown this for $Ent(\mathbf{x})$ in [MFMT97]. So let us first focus on proving inequality for $Chi(S;\bar{S})$. Let $\mathbf{s} = (s_1, ...s_k)$ be the set of all records to be split, and let $||\mathbf{x}|| = \sum_{i=1}^{k} x_i$. Let $p_i(R) = s_i/||\mathbf{s}||$ and $p_i(S) = x_i/||\mathbf{x}||$; then $Chi(S;\bar{S})$ is transformed as follows:

$$Chi(S;\bar{S}) \;=\; C(\mathbf{x}(S)) + C(\mathbf{x}(R) - \mathbf{x}(S)),$$

where

$$C(\mathbf{x}(S)) = C(\mathbf{x}) = \sum_{i=1}^{k} \frac{(x_i - s_i\frac{||\mathbf{x}||}{||\mathbf{s}||})^2}{s_i\frac{||\mathbf{x}||}{||\mathbf{s}||}}.$$

Let $\mathbf{\Delta} = (\delta_1, \delta_2, ..., \delta_k)$, $Y = (\mathbf{\Delta}, \mathbf{x}) = \sum_{i=1}^{k} \delta_i x_i$.

$$\begin{aligned} C'(\mathbf{x}) &= \tfrac{dC(\mathbf{x})}{dY} \\ &= \textstyle\sum_{i=1}^{k} \tfrac{1}{\delta_i}\tfrac{\partial C(\mathbf{x})}{\partial x_i} \\ &= \tfrac{||\mathbf{s}||}{||\mathbf{x}||}\textstyle\sum_{i=1}^{k} \tfrac{2x_i}{\delta_i s_i} - \tfrac{||\mathbf{t}||||\mathbf{s}||}{||\mathbf{x}||^2}\textstyle\sum_{i=1}^{k} \tfrac{x_i^2}{s_i} - ||\mathbf{t}|| \end{aligned}$$

and

$$\begin{aligned} C''(\mathbf{x}) &= \frac{d^2C(\mathbf{x})}{dY^2} \\ &= \frac{2\|\mathbf{s}\|}{\|\mathbf{x}\|}\sum_{i=1}^{k}\frac{1}{s_i}\left(\frac{1}{\delta_i} - x_i\frac{\|\mathbf{t}\|}{\|\mathbf{x}\|}\right)^2 \geq 0, \end{aligned}$$

where $\mathbf{t} = (\delta_1^{-1}, ..., \delta_k^{-1})$. Therefore $Chi''(S;\bar{S}) \geq 0$.

Next, for $Gini(S;\bar{S})$, let $p_i(S) = x_i/\|\mathbf{x}\|$. It is transformed as follows:

$$Gini(S;\bar{S}) = Constant$$

$$+\frac{1}{|R|}G(\mathbf{x}(S)) + \frac{1}{|R|}G(\mathbf{x}(R) - \mathbf{x}(S)),$$

where

$$G(\mathbf{x}(S)) = G(\mathbf{x}) = \sum_{i=1}^{k}\frac{x_i^2}{\|\mathbf{x}\|}.$$

So,

$$\begin{aligned} G'(\mathbf{x}) &= \frac{dG(\mathbf{x})}{dY} \\ &= \sum_{i=1}^{k}\frac{1}{\delta_i}\frac{\partial G(\mathbf{x})}{\partial x_i} \\ &= \frac{1}{\|\mathbf{x}\|}\sum_{i=1}^{k}\frac{2x_i}{\delta_i} - \frac{\|\mathbf{t}\|}{\|\mathbf{x}\|^2}\sum_{i=1}^{k}x_i^2, \end{aligned}$$

and

$$G''(\mathbf{x}) = \frac{2}{\|\mathbf{x}\|}\sum_{i=1}^{k}\left(\frac{\|\mathbf{t}\|x_i}{\|\mathbf{x}\|} - \frac{1}{\delta_i}\right)^2 \geq 0.$$

Therefore, $Gini''(S;\bar{S}) \geq 0$. □

Thanks to this property, the optimal point that gives the maximum value of an objective criterion must be on the convex hull of all stamp points in the $k$-dimensional space. The problem now becomes how to find the optimal points on the convex hull.

In [FMMT96a, MFMT97], we used the guided branch-and-bound search for finding the optimal two-dimensional numeric association rule. The proposed algorithm efficiently finds *the optimal* point and its corresponding rule. However, $k$ may become larger, so that $k > 4$, when we consider the problem for huge categorical databases like that in the "mRNA" sequence example. The space limitation does not permit use of the algorithm on ordinary workstations; we therefore present two feasible algorithms for finding *nearly optimal rules* for huge categorical databases, which present such a challenging problem in data mining.

## 3 Algorithms

### 3.1 Preparation

Changing $T$, we will frequently compute stamp points $\mathbf{x}(T)$. If we scan the database relation $R$ to find each stamp point, it will take $O(|R|)$ every time. To speed up this process, we preprocess the relation as follows.

Among all possible tests, we call tests that consist of only one element *finest* tests and denote a finest test $T_{\text{finest}}$. We can construct an arbitrary test for the categorical conditional attribute by making a union of the finest tests.

We compute a stamp point $\mathbf{x}(T_{\text{finest}})$ for each finest test beforehand. To find the stamp point of $T$, we simply need to sum up the points $\mathbf{x}(T) = \sum_{T_{\text{finest}} \in T}\mathbf{x}(T_{\text{finest}})$, which will take $O(|T|)$ time, i.e., $O(n)$. We also compute the stamp point of the entire relation $\mathbf{x}(R)$, since the stamp point of $\bar{S} = R \backslash S$ (complement of $S$) can be easily computed from $\mathbf{x}(\bar{S}) = \mathbf{x}(R) - \mathbf{x}(S)$.

**Example 3.1** Suppose we are given a relation $R$ with categorical attributes $A$ and $C$. Let $A$ be a target attribute, and $C$ be a conditional attribute. The following SQL query will count the number of tuples for each distinct value of $A$ and value of $C$ to generate the stamp points of finest tests:

```
SELECT A, C, COUNT(*) FROM R
GROUP BY A, C.
```

| $A$ | $C$ | count(*) | |
|---|---|---|---|
| $a_1$ | $c_1$ | 26 | $= x_1(\{c_1\})$ |
| $a_1$ | $c_5$ | 15 | $= x_1(\{c_5\})$ |
| $a_2$ | $c_2$ | 31 | $= x_2(\{c_2\})$ |
| ⋮ | ⋮ | ⋮ | ⋮ |
| $a_k$ | $c_n$ | 17 | $= x_k(\{c_n\})$ |

### 3.2 Random Algorithm

To simplify explanation of the Random Algorithm, we assume that the target domain size $k$ is 3. Let us consider the following data whose conditional domain size $n$ is 4:

| | $c_1$ | $c_2$ | $c_3$ | $c_4$ |
|---|---|---|---|---|
| $x_1(A = a_1)$ | 20 | 20 | 20 | 30 |
| $x_2(A = a_2)$ | 20 | 10 | 20 | 10 |
| $x_3(A = a_3)$ | 40 | 10 | 0 | 40 |

For each stamp point $\mathbf{x}(\{c_i\}) = (x_1, x_2, x_3)$ of the finest tests in three-dimensional space, we consider a projected point in two-dimensional space $\mathbf{y} = (y_1 = x_1/(x_1 + x_2 + x_3),\ y_2 = x_2/(x_1 + x_2 + x_3))$. Figure 1 illustrates the two-dimensional space. A straight line $L$ in this space splits the projected points $\mathbf{y}$ into two groups. Let us consider a test defined as the union of the finest tests that corresponds to the points in one of the two groups.

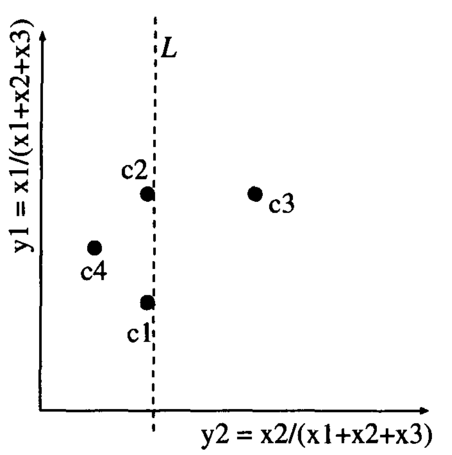


Figure 1: A segmentation in projected space

**Theorem 3.1** Tests defined by a straight line in the projected two-dimensional space correspond to stamp points that lie on the convex hull of all the stamp points in the original three-dimensional space. Moreover, the optimal test is one of those tests.

**Proof:** A straight line $L : ay_1 + by_2 = c$ corresponds to a plane $S : (a-c)x_1 + (b-c)x_2 - cx_3 = 0$ in the original three-dimensional space, which contains the origin $(0,0,0)$. $S$ splits the finest tests into two subsets. One subset contains finest tests whose inner products with the normal vector of $S$ are non-negative, while the other subset contains negative finest tests. Let $X$ be the set of all stamp points of tests. Summing up the finest tests with non-negative products generates a tangential point of the convex hull of $X$ with a plane whose normal vector is the same as $S$'s normal vector, i.e., $\Theta = (a-c, b-c, -c)$. Therefore, any line $L$ has a corresponding point on the convex hull of $X$. Conversely, any point on the convex hull of $X$ has a tangential plane $S$, and hence we can have a corresponding line $L$ in the projected space.

A broken line $L$ in the Figure 1, which is $0 \cdot y_1 + 1 \cdot y_2 = 0.25$, corresponds to a plane, say $S$, $-0.25x_1 + 0.75x_2 - 0.25x_3 = 0$ in the original three-dimensional space. The plane $S$ splits finest tests into two groups, $\{c_1, c_2, c_3\}$ that are non-negative and $\{c_4\}$ that is negative. (See also table in Section 3.3.1.) Stamp points, $(60, 50, 50)$ and $(30, 10, 40)$, that correspond to these groups are tangential points of the convex hull of all stamp points with a plane whose normal vector is $\Theta = (-0.25, 0.75, -0.25)$.

Theorem 2.1 proves that one of the points on the convex hull gives the optimal test. □

Therefore, we can concentrate on enumerating all the pairs of the projected points in order to find the optimal test.

**Algorithm 3.1**

```
(1) Project all points of finest tests to the two-dimensional space $\mathbf{x} \mapsto \mathbf{y}$
(2) For each pair of projected points $(\mathbf{y}_a, \mathbf{y}_b)$,
    (a) Draw a line $L = \mathbf{y}_a\mathbf{y}_b$ corresponding to $\mathbf{x}_a$ and $\mathbf{x}_b$.
    (b) Initialize a stamp point of a test, $\mathbf{p} = 0$
    (c) For each projected point $\mathbf{y}_i$, let $\mathbf{p} = \mathbf{p} + \mathbf{x}_i$ if $\mathbf{x}_i \mapsto \mathbf{y}_i$ is in the upper halfplane associated with $L$.
    (d) Evaluate $\mathbf{p}$ by means of an objective function.
```

Let $n$ be a conditional domain size of $C$. Since one splitting is defined by two projected points of finest tests, and we have $n$ points in this 2D space, there are $O(n^2)$ different pairs, and it will take $O(n)$ time to obtain the coordinates of a stamp point of a test. Therefore, the time complexity of Algorithm 3.1 is $O(n^3)$. There is a way to improve this complexity to $O(n^2)$ by using a sophisticated computational geometry algorithm [AT94].

We can extend this approach to cases where the stamp points are in space with more than three dimensions. Let $k$ be the target domain size, i.e., the dimension of stamp points. We project the stamp points of finest tests in $k$-dimensional space to points $\mathbf{y} = (x_1/||\mathbf{x}||, x_2/||\mathbf{x}||, \ldots, x_{k-1}/||\mathbf{x}||)$ in $(k-1)$-dimensional space. The optimal segmentation will then be one of the splittings with a $(k-2)$-dimensional plane in the $(k-1)$-dimensional space. Since $k-1$ points define one splitting and corresponding test, there are $O(n^{k-1})$ possible tests, and it takes $O(n)$ time to compute a stamp point of a test. Hence the overall time complexity is $O(n^k)$, which is too costly for large $k$'s.

To reduce the complexity, (1) we take $s$-sized random sample from $n$ finest tests, (2) project those sample stamp points to $(k-1)$-dimensional space, and (3) apply the same algorithm on the sample. Thus, the time complexity of this randomized algorithm becomes to $O(s^{k-1}n)$, and can be further improved to $O(s^{k-2}n)$.

## Performance Guarantee for the Random Algorithm

As we have shown above, the optimal solution is given as a subdivision of stamp points by a hyperplane cut in $(k-1)$-dimensional space. From the PAC learning theory, such a subdivision can be closely approximated by using a small number of samples. Let $Y$ be the set of points in $(k-1)$-dimensional space and $Z$ be a random sample from $Y$. We say that $Z$ is an $\epsilon$-net for a region family if $|V \cap Y|/|Y| \leq \epsilon$ holds for every region $V$ of the family satisfying $V \cap Z = \emptyset$.

Suppose that the optimal subdivision $S_{opt}$ is given by a hyperplane $H_{opt}$. Let us consider the family of wedges bounded by $H_{opt}$ and another hyperplane. This family of wedges defines at most $O(n^k)$ different subdivisions of $n$ points (roughly speaking, the Vapnik-Chervonenkis dimension is $k$), and hence it is known [HW87, BEHW89] that a random sample of size

$$s(\epsilon) = \epsilon^{-1} \max\{6k \log(16k\epsilon^{-1}), 4\log(2\delta^{-1})\}$$

is an $\epsilon$-net with a probability of at least $1 - \delta$. Note that $s(\epsilon)$ is independent of $n = |Y|$.

Let us take a sample that is an $\epsilon$-net for our wedges, and let $S_{sample}$ be the segmentation obtained by a sample maximizing the objective function (e.g., gini) $F$. There exists a subdivision $S'_{sample}$ obtained by the sample such that the edge bounded by $H_{opt}$ and $H'_{sample}$ (the hyperplane associated with $S'_{sample}$) contains no sample point. Since our sample is an $\epsilon$-net, the set difference between the segmentations $S'_{sample}$ and $S_{opt}$ contains at most $\epsilon n$ points. By definition, $F(\mathbf{x}(S_{sample})) \geq F(\mathbf{x}(S'_{sample}))$, and hence $F(\mathbf{x}(S_{sample}))$ has a guaranteed performance. For example, we can show that

$$Gini(\mathbf{x}(S_{opt})) - Gini(\mathbf{x}(S_{sample})) \leq 2\epsilon + \alpha\epsilon^2$$

where $\alpha = |R|(|S_{opt}|^{-1} + |\bar{S}_{opt}|^{-1})$. Since we do not want to find a subdivision with a very large $\alpha$, this is a good approximation if $\epsilon$ is small. We can use the theory on $\epsilon$-approximation for $k$-labeled space given by Hasegawa et al [HII95] to avoid introducing $\alpha$ into the analysis. Theoretically, if we want to make $\epsilon = 0.01$, $s(0.01) = 600k \log(1600k)$, which is very large. However, the theoretical bound is very pessimistic, and a much smaller sample is sufficient, as we will show by experiment later.

## 3.3 Probing Algorithm

### 3.3.1 Hand Probing for Categorical Tests

The Probing Algorithm also searches for stamp points (tests) that lie on a convex hull. A method of computing a point on a convex hull and its corresponding test without knowing the coordinates of the point is called *hand probing* in the field of computational geometry [DEY86]. Geometrically, hand probing in a $k$-dimensional space means computing a tangential point of a $(k-1)$-dimensional hyperplane and the convex hull of all stamp points.

A $(k-1)$-dimensional hyperplane in a $k$-dimensional space can be characterized by using a normal vector $\Theta = (\theta_1, \theta_2, \ldots, \theta_k)$ as $(\Theta, \mathbf{x}) = a$, where $(\Theta, \mathbf{x}) = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_k x_k$ and $a$ is a constant. A hyperplane with the $\Theta$ touches to the convex hull of all stamp points. We can compute the tangential point $\mathbf{p}$ of the hyperplane and the convex hull by maximizing (or minimizing) the inner product $(\Theta, \mathbf{p})$.

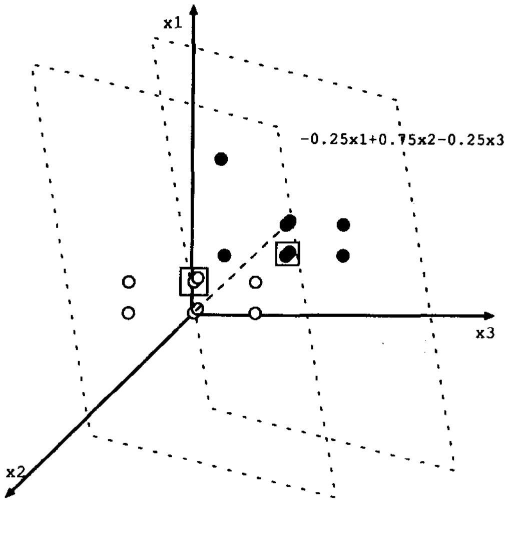


Figure 2: Hand Probing in 3D Space

Let us consider once again the example used in Section 3.2. The tangential point of a hyperplane with a normal vector $\Theta = (-0.25, 0.75, -0.25)$ maximizes (or minimizes) the inner product $(\Theta, \mathbf{x}) = -0.25x_1 + 0.75x_2 - 0.25x_3$. We compute the inner product $(\Theta, \mathbf{x})$ for each value of $c_1, \cdots, c_4$.

| | $c_1$ | $c_2$ | $c_3$ | $c_4$ |
|---|---|---|---|---|
| $(\Theta, \mathbf{x})$ | 0 | 0 | 10 | -10 |

The test corresponding to the hand probing is the union of those terms whose inner product is non-negative (resp. negative). The coordinates of the test (tangential point) can be obtained by summing up the coordinates of those terms. In this example, $C = c_1 \vee c_2 \vee c_3$ (resp. $C = c_4$) is the test, and the tangential point is $(60, 50, 50)$ (resp. $(30, 10, 40)$). Figure 2 illustrates the hand probing.

The time complexity of the hand probing used to compute the stamp point that maximizes $(\Theta, \mathbf{p})$ is $O(n)$, where $n$ is the conditional domain size.

### 3.3.2 Convex Hull Searching

We can compute stamp points on the convex hull by hand probing. Now, let us consider how to find nearly optimal points by hand probing.

First of all, we compute $k$ different tests and $2k$ corresponding stamp points by hand probing using several vectors. Empirically, a vector $\Theta_{init}$ satisfying $(\Theta_{init}, \mathbf{x}(R)) = 0$ finds a decent point with respect to a convex criterion. (The line containing the origin and $\mathbf{x}(R)$ is the central line of the convex hull, and a hyperplane whose normal vector is $\Theta_{init}$ is parallel to the line.) Therefore, we include such vectors in the initial set of vectors. We can make an initial convex polygon, which consists of $2^k$ facets using the initial $2k$ points, inside the convex hull of all stamp points. (Strictly speaking, we may not be able to find $k$ independent points in a $k$-dimensional space by any hand probing. In this case, we have to projects all points into a $(k-1)$-dimensional space.)

The guided branch-and-bound search method [FMMT96a, MFMT97] efficiently finds the optimal point on the hull from the initial convex polygon inside the hull by repeating the following polygon expansion procedure, when $k$ is small enough to allow sufficient working space.

**Algorithm 3.2**

```
        for each facet in set of valid facets
             on convex polygon
        do {
(1)          Compute a normal vector $\Theta$ of the facet.
(2)          Compute a new point $\mathbf{x}$
                  by hand probing with $\Theta$.
             if(new point $\mathbf{x}$ is found outside the facet)
             then{
                  if(the new point exceeds the best)
(3)                    then Update the best point.
(4)               Construct new facets by using $\mathbf{x}$.
(5)               Input those new facets
                       to the set of valid facets.
             } else {
(6)               Invalidate the facet.
             }
        }
```

Throughout the above recursions, we use the "beneath-beyond" method [PS85] introduced by Kallay, to maintain the convexity of the polygon.

If we maintain all the tangent planes of each hand probing, we can compute the value of the upper bound of each facet if the objective criterion is convex. In the guided branch-and-bound search, we use the upper bound value to prune facets and also to maintain a priority queue of facets to be expanded. The priority queue only maintains facets that have not been pruned and we call them *valid* facets. The authors applied this strategy to find two-dimensional numeric association rules for which the cost of each hand probing is expensive [MFMT97] and the target domain size is small such that $k = 2, 3, 4$. However, the cost, with respect to both time and working space, of maintaining facets with the priority queue is not affordable for the problem of huge categorical databases in which the cost of

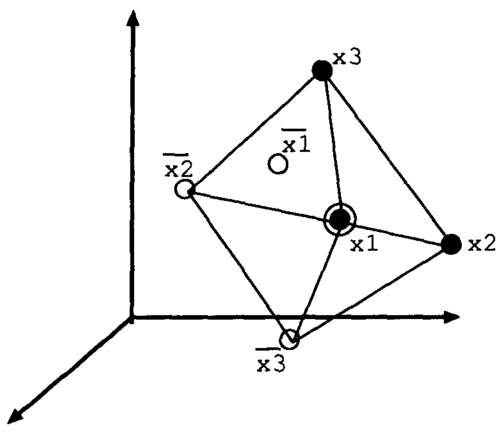


Figure 3: Initial Facets in 3D Space

each hand probing is low and the target domain size may not be small. Therefore, we need to eliminate a significant quantity of facets and use another heuristic for ordering them.

In the Probing Algorithm, we predefine the size of the working space that is used to maintain facets with a queue, and the algorithm maintains only facets that can be stored by the limited working space. For the initial convex polygon, we maintain only the facets that contain the best of $k$ points. Figure 3 shows an example of an initial set of facets in a three-dimensional space. If $\mathbf{x_1}$ is the best point, four facets, $\overline{\mathbf{x_1 x_2 x_3}}$, $\overline{\mathbf{x_1}\overline{\mathbf{x_2}}\mathbf{x_3}}$, $\overline{\mathbf{x_1 x_2}\overline{\mathbf{x_3}}}$, and $\overline{\mathbf{x_1}\overline{\mathbf{x_2 x_3}}}$, are selected as an initial set. If the working space does not allow these initial facets to be maintained, we randomly choose facets to be maintained.

After obtaining an initial set of facets, the Probing Algorithm maintains a facet queue with a skip list structure. The skip list is maintained by using the following heuristics.

- Facets that contain the best point so far are examined earlier.
- Facets that contain points that have nearly the best value so far are also examined earlier.

  We empirically find that stamp points whose value of an objective criterion exceed the best value tend to lie near the best point or near the points that have nearly the best value. The above heuristics contribute to find (nearly) optimal result in an earlier step of the algorithm.

- If the working space available in which to add new facets is running out, only facets that satisfy the above conditions are added.

The Probing Algorithm can run with much smaller working space compared with the guided branch-and-bound search by giving up the maintenance of the tangent planes of each hand probing. However, for the problem of larger target domain size, we need some limitations of the working space to make the Probing Algorithm feasible. Therefore, we prefer facets that lie near the best point or near the points that have nearly the best value, where stamp points that exceed the current best value tend to lie.

- If the distance between a facet and a new point that has been hand probed by the normal vector of the facet is smaller than a certain threshold, the facet is pruned away.

  For the problem of large target domain size, we sometimes suffer from overflow and underflow error caused by floating point operations. To prevent the problem, we should avoid too fine expansion of the convex polygon using the threshold value. This heuristics also contribute to speed up the searching.

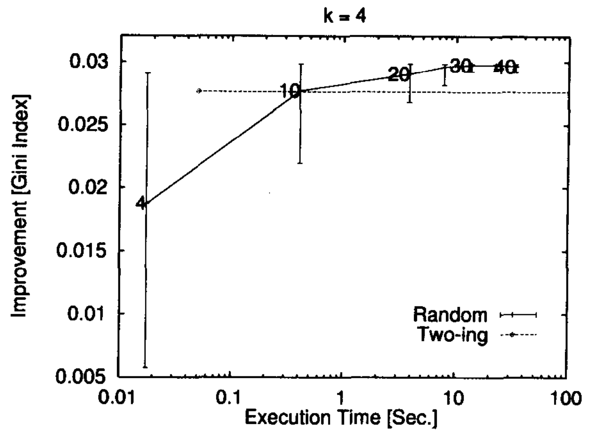
Though the running time of the Probing Algorithm is still substantial, it can find nearly optimal points in a period much shorter than the expected running time, and can return the intermediate (but optimal or nearly optimal) test interactively.

Both of the Random and the Probing Algorithm search for points on the convex hull of all stamp points. The Random Algorithm searches points on the hull using sample points corrected at random from stamp points of finest tests, while the Probing Algorithm maintains facets and points that have been found so far and makes the most of those for finding new points.

## 4 Experiments

We implemented the proposed algorithms and performed several experiments to evaluate their performance. All experiments were performed on an IBM RS/6000 workstation consisting of a POWER2 processor running at 66 MHz with 2 MB of L2 cache and 256 MB of real memory.

In this experiment, we generated synthetic data with $n = 1000$ (conditional domain size) and various values of $k$ (target domain size) to simulate a huge categorical database. Each type of dataset has 10,000 tuples and two categorical attributes, $C$ and $A$. The conditional attribute $C$ takes $c_1, \ldots, c_n$ distinct values. The target attribute $A$ takes $a_1, \ldots, a_k$ distinct values. For each tuple of the synthetic data, we randomly and independently assign a value of $C$ and a value of $A$. For the purposes of comparison, we use the "two-ing" method, in addition to our methods, to compute a binary segmentation.



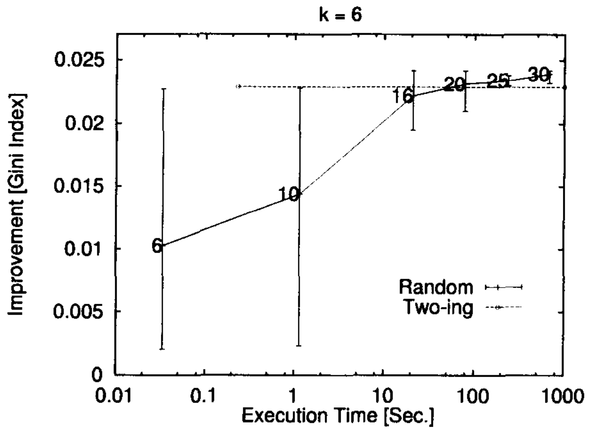


Figure 4: Performance of the Random Algorithm (n = 1000)

### Random Algorithm

To support our claim that the Random Algorithm can find a nearly optimal test by using a small sample, we performed the following experiments.

Figure 4 shows relationship between the execution time for a single run and the improvement in the gini index gained by a test obtained by the Random Algorithm for various sample sizes. The conditional domain size $n$ is 1,000, and the target domain size $k$ is 4 and 6. The numbers in the graph represent the sample sizes. Each error-bar (vertical line) indicates the range between the best and worst result of 32 runs for each sample size. The results of the 32 runs distribute in the range, and each point on the range shows the average value of these results.

The "two-ing" method deterministically computes

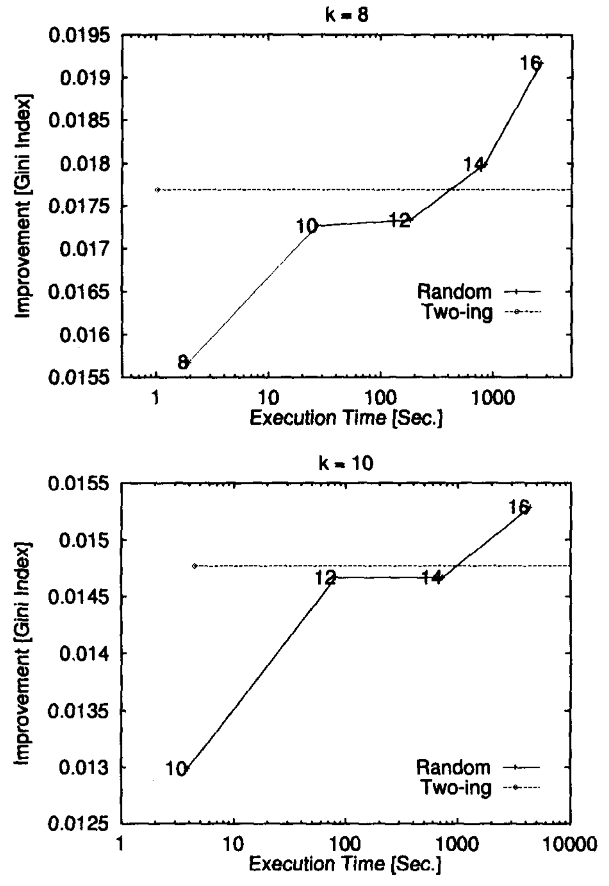


Figure 5: Performance of the Random Algorithm (n = 1000)

a test for each problem. Each diamond mark in the figures indicates the time taken to compute the test and the improvement in the gini index by the "two-ing" method. We draw a horizontal broken line for each diamond mark to compare our algorithms with the "two-ing" method easily. Though the "two-ing" gives the optimal test of a certain superclass obtained by grouping $k$ classes, the heuristic is known to find a relatively good approximated test.

From this experiment, we can see that the Random Algorithm generates a result of satisfactory quality within a practical time when the sample size is around 20. And notice that the best result of the 32 runs is better than the result of the "two-ing" method even if we use a small sample size. The CPU time taken for a single run is almost proportional to $\binom{s}{k-1}$.

When $k$ becomes larger, we have to use a small sample size $s$ so that the algorithm can terminate in a practical amount of time. However, a small sample often gives low-quality results. To overcome this problem, we run this algorithm with a small sample a number of times, and take the best result. Figure 5 shows the relationship between the execution time for 32 runs and the best improvement in the gini index of 32 runs when $k = 8$ and 10.

The Random Algorithm achieved better or comparable results with small sample by 32 runs. Those multiple trials can be executed independently. Therefore, we can expect more better results if we can use parallel environment.

## Probing Algorithm

To examine the Probing Algorithm, we performed several experiments using the same synthetic data that was used for the Random Algorithm.

Figures 6 show the extent to which the gini index value is improved by the Probing Algorithm, along with the time taken in seconds. Thanks to the heuristics for maintaining the skip list, we can observe that when the Probing Algorithm find a better result, it often find more better result within a short period of time. As a result, the Probing Algorithm can find nearly optimal tests much earlier than the time of its termination. Such nearly optimal tests, which are much more better than the result of the "two-ing" method's, are satisfactory for most of applications. So the Probing Algorithm returns the intermediate results in practical time.

One defect of the Probing Algorithm is its required working space when $k$ becomes large. However, it can find satisfactory results by the time when its working space becomes large. In the experiment for $k = 10$, working space of the Probing Algorithm is less than 64 MB when the best result in the graph was obtained. In these experiments, we limited the working space of the Probing Algorithm to 130 MB.

The results of the experiments show that both the Random Algorithm and the Probing Algorithm find a better test than the "two-ing" method in a practical time and feasible.

We consider the binary segmentation problem for cases in which the target domain size, $k$, is a small constant. However, $k$ may become large. Both the Random and the Probing Algorithm will suffer from large $k$. In such cases, we have to group $k$ values into smaller distinct values.

## 5 Concluding Remarks

We propose two geometric algorithms for finding association rules that make nearly optimal binary segmentations of a huge categorical database. We can use the "gini index," "entropy," or "$\chi^2$" as an objective criterion, which indicates the extent to which the values of

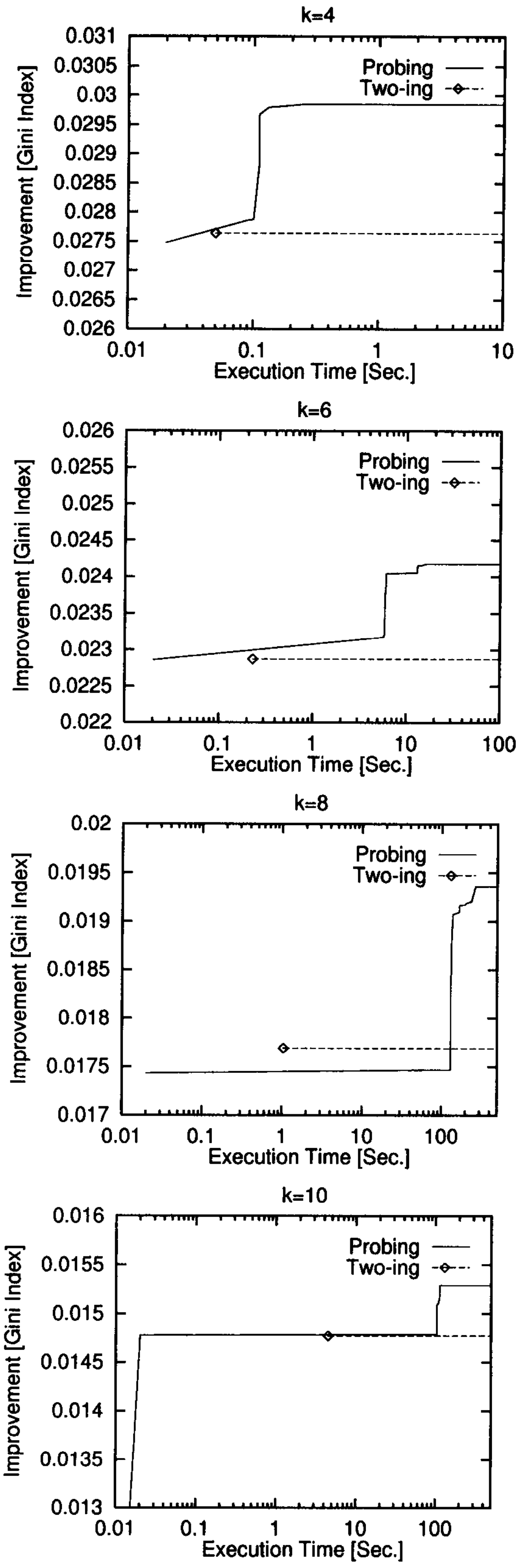


Figure 6: Performance of the Probing Algorithm (n = 1000)

a target attribute are skewed from the original distribution. Rules that indicate nearly optimal values of these criteria provide us with a clue to understanding the target attribute.

Though complexities are $O(s^{k-2}n)$ for the Random Algorithm and $O((n+m)|P|)$ for the Probing Algorithm, we make them practical by randomization and strategic facet maintenance. Diverse experiments confirmed that the algorithms could find nearly optimal tests within a practical computation time. The quality of the results obtained for various samples by the Random Algorithm differs dramatically if we use a smaller sample size. Therefore, multiple trials are needed to obtain better rules. Those trials are executed independently and in parallel; thus, the Random Algorithm is suitable for a parallel environment. On the other hand, the quality of the Probing Algorithm becomes stable after a certain amount of execution time. However, it requires a large working space. Since the available memory size is becoming larger and larger, the applicability of the Probing Algorithm seems to be promising.

Let us consider once again the "RNA" example, and assume that we do not know which subset of the RNA sequence most strongly affects a target character of interest. We can project several subsets of the sequence as categorical databases and examine the impact of each subset by using the presented algorithms. Then, we can find a subset of the sequence that significantly affects the target character, and carry out a further investigation of the RNA sequence, using the subset. In the further investigation, the presented algorithms can also be used as powerful mining tools.

**Acknowledgements**

This research is partially supported by the Advanced Software Enrichment Project of the Information-Technology Promotion Agency, Japan.

## References

[AIS93] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 207–216, May 1993.

[AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th VLDB Conference*, pages 487–499, 1994.

[AT94] T. Asano and T. Tokuyama. Topological walk revisited. In *Proceedings of the 6th CCCG*, 1994.

[BEHW89] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM*, 36:929–965, 1989.

[BFOS84] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth, 1984.

[Cen92] NASA Ames Research Center. *Introduction to IND Version 2.1.* GA23-2475-02, 1992.

[DEY86] D. Dobkin, H. Edelsbrunner, and C. Yap. Probing convex polytopes. In *Proc. 18th ACM Symposium on Theory of Computing*, pages 387–392, 1986.

[FMMT96a] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Constructing efficient decision trees by using optimized association rules. In *Proceedings of the 22nd VLDB Conference*, pages 146–155, 1996.

[FMMT96b] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Data mining using two-dimensional optimized association rules: Scheme, algorithms, and visualization. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 13–23, June 1996.

[FMMT96c] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Mining optimized association rules for numeric attributes. In *Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 182–191, June 1996.

[HF95] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *Proceedings of the 21st VLDB Conference*, pages 420–431, 1995.

[HII95] S. Hasegawa, H. Imai, and M. Ishiguro. $\epsilon$-approximations of k-label spaces. *Theoretical Computer Science*, 137:145–157, 1995.

[HW87] D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. *Discrete and Computational Geometry*, 2:127–151, 1987.

[MFMT97] Y. Morimoto, T. Fukuda, S. Morishita, and T. Tokuyama. Implementation and evaluation of decision trees with range and region splitting. *Constraint*, 2(3/4):163–189, December 1997.

[MP91] P. M. Murphy and M. J. Pazzani. Id2-of-3: Constructive induction of m-of-n concepts for discriminators in decision trees. In *Proceedings of the 8th Intl. Wrokshop on Machine Learning*, pages 183–187, 1991.

[PCY95] J. S. Park, M. Chen, and P. S. Yu. An effective hash-based algorithm for mining association rules. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 175–186, May 1995.

[PS85] F. P. Preparata and M. I. Shamos. *Computational Geometry, An Introduction.* Springer-Verlag, 1985.

[PS91] G. Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In *Knowledge Discovery in Databases*, pages 229–248, 1991.

[Qui86] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[Qui93] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.