IEICE TRANS. INF. & SYST., VOL.E93–D, NO.6 JUNE 2010

PAPER Special Section on Info-Plosion

Margin-Based Pivot Selection for Similarity Search Indexes

Hisashi KURASAWA[†a)], **Daiji FUKAGAWA**[†b)], **Atsuhiro TAKASU**[††c)], ***and*** **Jun ADACHI**[††d)], ***Members***

SUMMARY When developing an index for a similarity search in metric spaces, how to divide the space for effective search pruning is a fundamental issue. We present Maximal Metric Margin Partitioning (MMMP), a partitioning scheme for similarity search indexes. MMMP divides the data based on its distribution pattern, especially for the boundaries of clusters. A partitioning boundary created by MMMP is likely to be located in a sparse area between clusters. Moreover, the partitioning boundary is at maximum distances from the two cluster edges. We also present an indexing scheme, named the MMMP-Index, which uses MMMP and pivot filtering. The MMMP-Index can prune many objects that are not relevant to a query, and it reduces the query execution cost. Our experimental results show that MMMP effectively indexes clustered data and reduces the search cost. For clustered data in a vector space, the MMMP-Index reduces the computational cost to less than two thirds that of comparable schemes.
key words: *similarity search, indexing, metric space*

1. Introduction

The purpose of our research is to reduce the query execution cost of a similarity search in metric spaces. Although a large number of studies have been made on indexing techniques for similarity searches, only a few exploit the data distribution in a metric space to divide the search space. Our goal is to develop a new partitioning scheme based on the data distribution that can prune the space more effectively.

A similarity search efficiently finds objects that are similar to a query from a large dataset [1]. Similarity searches based on a metric space can be applied to all types of data whose distances obey metric space postulates such as the triangle inequality. Therefore, metric space indexes are very useful for applications that deal with huge amounts of vectors, strings, graphs, tags, and so on.

Indexing schemes for metric spaces are mainly categorized into *Pivot Partitioning* and *Pivot Filtering*. A *Pivot* is a reference object in an index. Pivot partitioning divides the search space into regions by using a pivot to prune some of the regions during the search. Pivot filtering stores computed distances between a pivot and objects and filters out some of these objects during the search. Pivot partitioning discard the objects included in the regions irrelevant to a query, and it results in a smaller index because it does not need to store the distance to an object. Therefore, many indexing schemes start by dividing the dataset into regions with pivot partitioning and then use pivot filtering on each region.

The performance of pivot partitioning depends on the pivot selection. Pivot selection methods proposed in early researches were based on heuristics. They asserted that good pivots should be outliners of the space, because the distances from a pivot to each object vary and the objects are easily classified by the pivot [2]. They used simple statistical features such as the mean and the variance for selecting pivots. However, their choices of pivots are only a little better than random selection. Recently proposed selection mechanisms exploit data distribution (e.g., [3], [4]). These methods set cluster centers as the pivots and divide the space on the basis of the distances from the pivots. Although they can effectively classify dense regions, they only work well on particular distribution patterns. A cluster may be separated into multiple regions by a pivot, because their partitioning boundaries are based on cluster centers rather than cluster shapes.

We propose a novel method for pivot partitioning called Maximal Metric Margin Partitioning (MMMP), which can be adapted to the shapes of data clusters. First, MMMP constructs hierarchical clusters with arbitrary shapes by using Density-based Clustering [5]. Then, at each branch b in the hierarchical structure, it detects the space boundary that divides the objects in the cluster corresponding to b with the maximal margin. During this division, MMMP searches for a pivot object whose partitioning boundary is between clusters and also maximizes the distances to the cluster edges. Our main contribution is that we have developed a pivot selection for metric spaces that is based on maximal margins and is effective for clustered data. To the best knowledge of the authors, this is the first study to exploit the maximal margins for the pivot selection.

In this paper, we explain how MMMP divides data for a similarity search index. We also present an indexing scheme called MMMP-Index, that uses MMMP and pivot filtering. We evaluated the indexing performance in comparison with existing schemes. While our previous paper [6] showed the partitioning performance of MMMP, this paper extends the experimental evaluation and assesses the index

Manuscript received August 29, 2009.
Manuscript revised December 8, 2009.
[†]The authors are with the Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, 101–8430 Japan.
[††]The authors are with National Institute of Informatics, Tokyo, 101–8430 Japan.
a) E-mail: kurasawa@nii.ac.jp
b) E-mail: daiji@nii.ac.jp
c) E-mail: takasu@nii.ac.jp
d) E-mail: adachi@nii.ac.jp
DOI: 10.1587/transinf.E93.D.1422

performance.

The rest of the paper is organized as follows. In Sect. 2, we introduce the background to this study and related work. Sections 3 and 4 describe MMMP and its indexing scheme. Section 5 discusses the experimental results. We conclude in Sect. 6.

2. Background and Related Work

We first describe the notations and definitions used in this paper with examples of distance measures. We also show indexing schemes for similarity searches.

2.1 Metric Space and Distance Measure

Our similarity search index deals with all types of data whose distances obey metric space postulates.

For a domain of objects D and a distance function $d : D \times D \mapsto \mathbb{R}$ that satisfies:

- $\forall x, y \in D, d(x, y) \geq 0$ (non-negativity)
- $\forall x, y \in D, d(x, y) = d(y, x)$ (symmetry)
- $\forall x, y \in D, x = y \leftrightarrow d(x, y) = 0$ (identity)
- $\forall x, y, z \in D, d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality)

we denote a metric space as $M = (D, d)$ [1].

There are various distance functions for various data types. For example, the Minkowski distance is defined on n-dimensional vectors. Jaccard's coefficient and the Hausdorff distance [7] quantify distances on sets. The quadratic form distance is adapted to correlated dimensional vectors, such as color histograms. The edit distance [8] is applied to strings, such as bibliographic records.

Several similarity queries have been proposed, including the Range query, Nearest neighbor query, and Similarity join. This paper mainly deals with Range queries.

2.2 Indexing Scheme

A similarity search index can reduce the query execution cost, such as the page access cost and the computational cost. As mentioned above, many indexing schemes use pivot partitioning and pivot filtering. In this section, we focus on pivot selection in pivot partitioning.

Before introducing the various pivot selection techniques, we explain how to divide the space with pivots. Partitions are classified into two types from the viewpoint of the number of required pivots. *Ball partitioning* uses only one pivot and divides the space into two subspaces according to the distance from each object to the pivot [2], [9]. Hence, its partitioning boundary is spherical. When the space is partitioned into more than two spherical areas, it is called *multi-way ball partitioning* or *excluded middle partitioning* [10]–[12]. The *generalized hyper-plane* [9], which is also classified into this group, uses two pivots, and divides the space by the hyperplane equidistant from the two pivots.

Next, we list the pivot selection techniques. A better pivot partitioning requires fewer pivots and prunes more objects during a search. The early studies selected pivots that were far away from the other objects and other pivots [2], [13]. They aimed at getting various distances from the pivot to each object and discarding useless pivots. Some also tried to exclude dense regions [12], [14]. These methods used simple statistical features such as the mean [12], [15], the variance [2], [14], the sum [16], and the ratio [17].

Recently, researchers have utilized more elaborate features based on the data distribution. For example, iDistance [3] clusters the search space and identifies the cluster centers. Then, it divides the space by using Voronoi partition, a kind of generalized hyper-plane, by using the cluster centers as pivots. This helps to reduce the number of regions accessed during search. In other words, it reduces the page access cost. Recent methods not only improve the indexing performance [4], but also deal with parallel distributed queries [18]–[20].

The List of Clusters (LC) [21] is another way of exploiting the data distribution. It divides the space into compact regions, and it recursively excludes the indexed objects from the space. The radius of partitioning is determined depending on the number of objects inside the partition. In short, LC adapts the radius in proportion to the data density. However, it needs more pivots because it uses radiuses smaller than other schemes. We will mention the details of it in Sect. 2.3

We think pivot partitioning can further reduce the search cost by utilizing the data distribution of the objective dataset. In the existing approaches, skewed data clusters may be separated into multiple regions by pivots, because their partitioning boundaries are based on cluster centers rather than cluster shapes. This causes an increase in query execution cost. Therefore, we developed a new partitioning scheme based on the shapes of the data clusters.

2.3 List of Clusters (LC)

LC consists of many compact regions. It repeatedly selects a pivot and its partitioning distance to define a region. It starts with the object set S in the database and selects a pivot p_1 and its partitioning distance r_{p_1}. The pivot p_1 and the distance r_{p_1} define a region

$$I_1 = \{o \in E, d(p_1, o) \leq r_{p_1}\} . \quad (1)$$

Then, it selects the next pivot and its partitioning distance for the remaining object set $S - I_1$ and repeats the same process until the remaining object set is empty. As a result, a list of tuples (p_i, r_{p_i}, I_i) is generated. Five pivot selection schemes and two partitioning distance selection schemes were considered in [21]. The best performance was achieved when it selects the pivot such that it maximizes the sum of distances to previous pivots and it sets the partitioning distance on the basis of the number of objects inside the partition.

For a range query (q, r_q), LC computes the distance from q to the pivot p_i. If the inequality

$$d(q, p_i) - r_q \leq r_p \,, \tag{2}$$

holds, it exhaustively searches I_i. If the inequality

$$d(q, p_i) + r_q \leq r_p \,, \tag{3}$$

holds, it continues the search process for the rest of the list.

3. MMMP

For a metric space $M = (D, d)$, suppose a pivot p divides a set S of objects in D into two regions:

$$S_1 = \{o \in S \mid d(o, p) \leq r_p\} \,,$$
$$S_2 = \{o \in S \mid d(o, p) > r_p\} \,,$$

where r_p is the partitioning distance for the pivot p. When searching for objects within r_q from a query object q in terms of the metric space M, if the inequality

$$d(q, p) + r_q \leq r_p \,, \tag{4}$$

holds, it is sufficient to check for objects in S_1. Similarly, if the inequality

$$d(q, p) - r_q > r_p \,, \tag{5}$$

holds true, it is sufficient to check objects within S_2. Let C_1 and C_2 be the largest clusters in S_1 and S_2, respectively. If the margin between the regions C_1 and C_2 is large, either the inequality of (4) or (5) is more likely to hold; i.e., we can prune region S_2 or S_1. On the other hand, if the boundaries between C_1 and C_2 are close to each other, we need to check the objects within both S_1 and S_2 for a query around the boundaries, even when the centers of C_1 and C_2 are far from each other. This leads us to a pivot selection method based on a large margin criterion.

To find pivots that divide the space into regions with a large margin, we first make hierarchical clusters that recursively divide the space into two regions. Currently, we use OPTICS [5], which is a density-based clustering method. To handle a large dataset, we use OPTICS on a randomly sampled dataset. Then, for each division, we find a pivot that separates the clusters with a large margin. Figure 1 depicts an example of partitioning by MMMP.

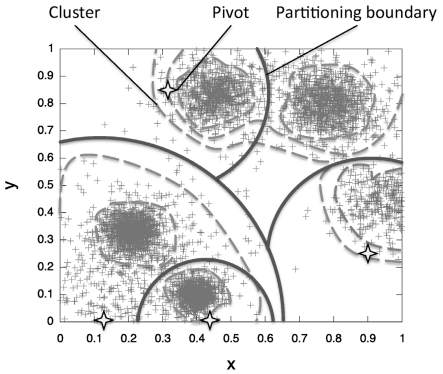


Fig. 1 Partitioning by MMMP.

3.1 Overview of OPTICS

OPTICS constructs hierarchical clusters with arbitrary shapes. It defines *the reachability distance* from the object o_1 to the object o_2 as the longer distance of the k-nearest neighbor distance of o_1 and the distance between o_1 and o_2. For a set D of objects, it first makes an ordered list q of objects by performing the following steps:

1. randomly select an object o from D and prepare an ordered list q consisting of o,
2. repeat the following steps until D becomes empty:
   a. choose an object p in D for which the reachability distance from its nearest object in q is the shortest,
   b. move p to the end of q.

As a result, objects in a same cluster tend to form a consecutive portion in the ordered list. Usually, a pair of adjacent objects in the list with a large reachability distance belongs to different clusters. OPTICS utilizes this feature to detect the boundary between clusters, Fig. 2 shows an example of the ordered list with the rechability distance on the vertical axis. The peaks in the graph correspond to boundaries between clusters. See [5] for details.

When the center area of a cluster is denser than the edge area, OPTICS tends to choose an edge object in the cluster first, reach the center area with the shortest path, and then move from the center area to the edge area. Therefore, we can extract objects around the edge of a cluster in the following way. For a cluster C detected by OPTICS, let q_C denote the sub ordered list corresponding to C. Suppose $\mathrm{Order}(o, C)$ denotes the position of an object $o \in C$ in q_C. We define the *edge object set* of C by

$$\mathrm{Edge}(n, C) = \left\{ o \;\middle|\; \mathrm{Order}(o, C) \leq \frac{n}{100}|C| \right\} \tag{6}$$

where n is a parameter. The edge object set represents the set of the objects whose reachability distances are ranked within the top $n\%$ in the cluster.

We chose OPTICS because

- it doesn't require the number of clusters,

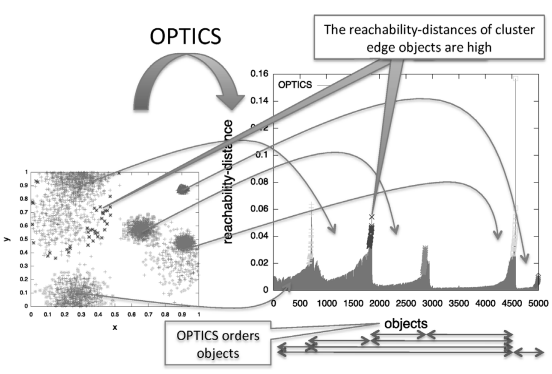


Fig. 2 Clustering by OPTICS.

- it constructs hierarchical clusters,
- it can extract arbitrarily-shaped clusters without calculating the center object in each cluster, and
- it can extract objects around the edge of the cluster efficiently.

The most important reason for using Density-based Clustering is to extract arbitrarily-shaped clusters while using the distances. Thus other schemes such as k-means and BIRCH are not appropriate for MMMP.

The clustering cost is $O(n^2)$ in the original paper [5]. Although the clustering cost can be reduced somewhat [22], it is still high. Therefore, we cluster random samples of the data instead of whole data.

3.2 Pivot Selection and Partitioning

MMMP aims to achieve effective search pruning. We focus on the partitioning boundaries and the data distribution patterns to achieve this requirement. We maximize the distances between the partitioning boundary and its closest objects, because the objects can be clearly classified. We divide the space on the basis of the OPTICS's clustering results because it effectively separates small dense regions. Therefore, we try to find partitioning boundaries that are between pairs of clusters that are at maximum distances from the cluster edges. However, we cannot directly detect the partitioning boundaries in a metric space, because the partitioning boundaries are created by pivots. Thus, MMMP basically searches for a pivot object that divides the clusters obtained by OPTICS with the largest margin.

From the clustering results given by OPTICS, we construct hierarchical clusters represented by a binary tree. Hierarchical clusters are represented with a tree structure whose nodes correspond to subspaces. Although a node can have more than two children, many nodes have just two children in the cluster hierarchy generated by OPTICS. Therefore, we shall first describe the pivot selection method for the node having two children. For a node v and an object p, let $C_{p,\mathrm{Near}}$ (resp. $C_{p,\mathrm{Far}}$) denote a v's child cluster that is near to (resp. far from) p. The cluster labels are judged on the basis of the distance from p to one object selected randomly from the objects in each cluster. Then, MMMP chooses the following object as a pivot

$$\mathrm{pivot} = \underset{p \in S}{\mathrm{argmax}} \Big(\min_{o_f \in C_{p,\mathrm{Far}}} d(p, o_f) - \max_{o_n \in C_{p,\mathrm{Near}}} d(p, o_n) \Big). \quad (7)$$

The first term on the right side of this formula represents the distance to the nearest object in $C_{p,\mathrm{Far}}$, whereas the second term is the distance to the farthest point in $C_{p,\mathrm{Near}}$. Therefore, the right side of Eq. (7) represents a kind of margin.

When a node has more than two children, MMMP merges the children into two clusters and obtained a candidate pivot by Eq. (7). Then, it chooses the candidate having the largest margin as a pivot.

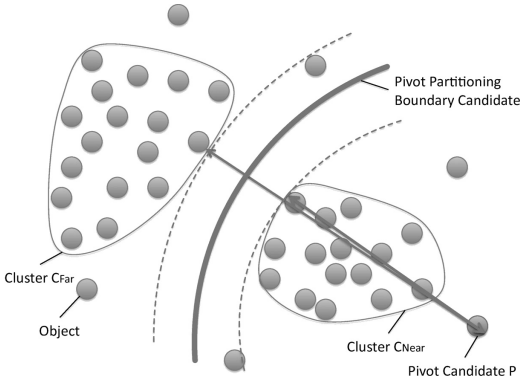


Fig. 3 Pivot selection in the MMMP.

The reasons why only one object in each cluster is used for distinguishing between $C_{p,\mathrm{Near}}$ and $C_{p,\mathrm{Far}}$ are as follows. One reason is to reduce the computational cost for this operation. The other reason is that the pivot candidates evaluated with the incorrect label clusters have no influence on the pivot selection. These pivot candidates are not relevant to the best pivot in most cases, and would finally be removed at Eq. (7). Of course, when there is no relevant pivot candidate in the data, this distinction does not work well and the evaluation of a pivot may have a minus value.

The partitioning distance r_p of the pivot p is defined as

$$r_p = \mathit{Distance}(p) = \frac{1}{2}\Big(\min_{o_f \in C_{p,\mathrm{Far}}} d(p, o_f) + \max_{o_n \in C_{p,\mathrm{Near}}} d(p, o_n) \Big). \quad (8)$$

Figure 3 is an overview of the pivot selection in MMMP.

4. MMMP-Index

4.1 Index Structure

Our indexing scheme, named MMMP-Index, uses MMMP and pivot filtering. As we described in Sect. 3, MMMP is designed to prune regions irrelevant to the query in the clustered data space. However, MMMP does not work well when the number of clusters in the data space is too small or the size of any one cluster is too large. The number of objects managed by a pivot affects the performance of the index. Therefore, we use the MMMP-Index to divide a large cluster into small enough regions for improving the performance by pivots as is done in LC [21]. Figure 4 shows the structure of MMMP-Index.

The index is implemented with two B+-trees. One B+-tree manages two kinds of pivots; the pivots selected by MMMP and those chosen by compact partitioning. The pivots form a tree structure. Each pivot has its own ID and the IDs of its leaf pivots. Smaller IDs are assigned to earlier selected pivots. The pivots' keys on the B+-Tree are also based on their IDs. The other B+-tree stores objects. The object key is measured by $d(o, p) + ID_p \times c$, where p is

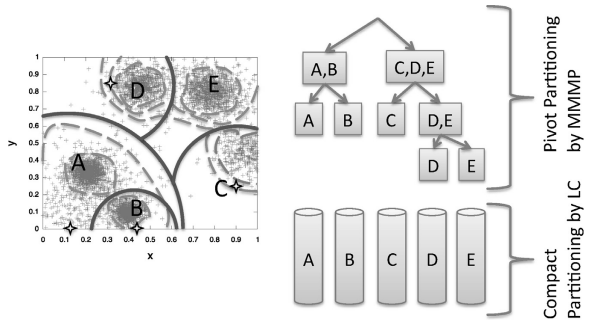


Fig. 4 MMMP-index.

the pivot that manages the object and c is a parameter that is sufficiently larger than the distance between objects. With these key definitions, when we search for objects whose distances from the pivot are within a certain range, we can sequentially access the disk and can reduce the page access cost.

In the implementation of MMMP-Index, we use Edge$(n, C_{p,\mathrm{Far}})$ (resp. Edge$(n, C_{p,\mathrm{Near}})$) defined by Eq. (6) instead of $C_{p,\mathrm{Far}}$ (resp. $C_{p,\mathrm{Near}}$) in Eqs. (7) and (8) for reducing the computational cost of the pivot evaluation. Note that the edge object set defined by Eq. (6) is a subset of its cluster. When the parameter n in Eq. (6) is 100, the edge object set is equivalent to the cluster. The rest of this section shows the index construction method and the range search method in the MMMP-Index.

4.2 Index Construction

4.2.1 Indexing

The MMMP-Index is constructed by performing the following steps:

1. Randomly select k objects from the data.
2. Discover hierarchical clusters from the objects by using OPTICS.
3. For each branch from the root of the cluster branches:
    a. Find the edge objects of each cluster in the branch whose reachability-distances are ranked within the top n% in the cluster.
    b. Select a pivot and its partitioning distance by using MMMP.
4. Create a tree structure with the pivot information.
5. For each object in the data, search for the region that the object belongs to.
6. For each region, execute compact partitioning by [21].
7. Construct two B+-Trees to manage objects and pivots, respectively.

4.2.2 Object Insertion

The MMMP-Index inserts an object by performing the following steps:

1. Search for the leaf region that the object belongs to in the tree structure of the index.
2. Search for the pivot that the object belongs to in the pivot list made by LC, and add the object [21].

4.2.3 Object Deletion

The MMMP-Index deletes an object by performing the following steps:

1. Search for an appropriate region for the object in the tree structure of the index.
2. Search for an appropriate pivot in the pivot list of the region, and delete the object [21].

4.3 Range Search

The MMMP-Index searches for objects whose distances to a query object q are within the distance r_q by performing the following steps:

1. Traverse the tree structure of the index and discard irrelevant regions by using the distance from the MMMP pivot to q and the triangle inequality (Eqs. (4) and (5)).
2. For each remaining region:
    a. Discard irrelevant pivots from the pivot list by using the distance from the pivot to q and the triangle inequality [21].
    b. For each remaining pivot:
        i. Extract the objects which may be relevant to the range query from the B+-Tree by using the distance from the pivot to q and the triangle inequality.
        ii. Measure the distance between each object and q. If the distance is within r_q, add the object to the result set.

4.4 Index Construction Cost

The MMMP-Index is constructed by

(a) the hierarchical clustering,
(b) selecting a pivot for each partition of a cluster in the hierarchy,
(c) and ball partitioning of the leaf clusters.

Step (a) requires from $O(n \log n)$ to $O(n^2)$ calculations by using OPTICS [5]. Step (b) requires $O(n^2)$, and step (c) requires from $O(m \log m)$ to $O(m^2)$, where m is the size of the maximum leaf cluster [21]. Generally m is much less than n, so steps (a) and (b) are the dominant parts of the indexing construction. On the other hand, LC requires from $O(n \log n)$ to $O(n^2)$ [21]. Hence, the computational complexity for constructing the MMMP-Index is the same as the worst case of LC. In our experiments, the indexing time for MMMP-Index was almost the same as that for LC.

Table 1 Data sets.

Data	Distance	No. of data	No. of queries	Data source	Dimension	Standard deviation range	No. of clusters
Vectors	Euclid distance	100,000	1,000	synthetic (see Sect. 5.1)	2	(0,0.10)	20
					8	Uniform	-
					8	(0,0.10)	10
					8	(0,0.10)	20
					8	(0,0.10)	30
					8	(0,0.20)	20
					16	(0,0.10)	20
					30	(0,0.10)	20
					8	(0,0.10) + noise	20
Julia Set	Euclid distance	100,000	1,000	synthetic (see Sect. 5.1)	2	-	-
Corel Image Features	Euclid distance	67,040	1,000	UCI KDD Archive	32	-	-
Photo tag sets	Jaccard's coefficient	50,000	1,000	Flickr (query:"tokyo")	-	-	-
English words	Edit distance	205,941	1,000	WordNet	-	-	-

5. Performance Evaluation

5.1 Data Set

We used three types of data to evaluate our scheme: vectors, sets, and strings. The datasets consisted of synthetic vectors generated by us, vectors of a Julia set, the real vector dataset named Corel Image Features, which were downloaded from the UCI KDD Archive [23], photo tag sets which were retrieved with the queries "tokyo" from flickr [24], and English words, which were downloaded from WordNet [25].

We generated 2, 8, 16, and 30-dimensional uniform and clustered vectors for the evaluation. The cluster centers of the clustered vectors were randomly selected. The centers numbered 10, 20, and 30. The number of objects for each cluster was randomly chosen. The objects in the cluster were based on the normal distribution. The standard deviations were randomly set from 0 to 0.10 and from 0 to 0.20. Furthermore, we generated synthetic vectors with noise objects. We mixed the clustered vectors with uniform vectors as noise. By referring to the previous study [21], we chose the size of the data to be 100,000. Queries were randomly chosen from the same distribution as the data set. When the chosen query happened to be the same point in the data set, we discarded it.

The Julia set is a fractal figure [26]. Let us consider a sequence of complex numbers defined by the recurrent form $z_{n+1} = {z_n}^2 + c$ where c is a complex constant. Then, the quadric Julia set is the set of the initial complex numbers z_0 in a sequence that does not diverge. We set c to be $0.40 - 0.35i$. Figure 12 (a) shows its shape.

As for the Corel image collection, we used the Corel Image Features [23] included in the collection as they were. They consist of 32 subspaces that divide up the HSV color space (32 colors: 8 ranges of H and 4 ranges of S). The value of each dimension indicated the density of each color in the entire image.

The photo tag set had 3,343 kinds of tags. The average number of tags per photo was 7.42. All the photos had the tag "tokyo".

As for the English words, we used all entry words included in the WordNet. We did not adjust the lengths of the words and uses the strings as the they were.

Table 1 lists the details of the datasets.

5.2 Outline of Experiments

We conducted experiments to evaluate the MMMP-Index together with MMMP. We compared the MMMP-Index with iDistance [3], D-Index [12], and LC [21].

As in the related work [1], the indexes' performances were evaluated in terms of the page access and computational costs. The page access cost of a region r_i Pagecost(r_i) is estimated by

$$\text{Pagecost}(r_i) = \lfloor N_{r_i} / P_{ave} \rfloor, \tag{9}$$

where N_{r_i} is the number of accessed objects in r_i, and P_{ave} is the average number of objects stored in a page. The page access cost of a query q is the sum of the costs of the accessed regions. On the other hand, we measured the computational cost by the total number of distance calculations during the search. Actually, the computational time depends not only on the number of distance calculations but also on the cost of each calculation. In addition, we show the search time of some data sets for reference, although the time depends on the machine resource. We conducted the experiment on a Linux PC equipped with an Intel(R) Quad Core Xeon(TM) X5492 3.40 GHz CPU and 64 GB Memory. We implemented the MMMP-Index in C and compiled it with GCC 4.2.

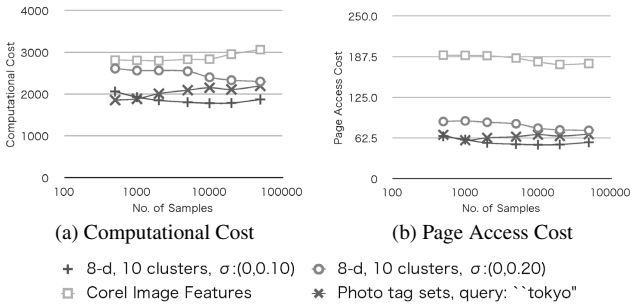
The performance of a similarity search index depends on the distribution of the data. Thus, this experiment used datasets containing various objects described in Sect. 5.1. According to the previous studies [3], the page size needed for estimating the page access cost is 4,096 bytes. Since the computational cost of iDistance to find the cluster centers in the data uses the computationally heavy edit distance or Jaccard's Coefficient, we didn't measure the performance of iDistance. Each result was the average over 1,000 queries of its dataset.

5.3 Effect of Sample Size

We first evaluated how the sample size of MMMP affects the index performance using four kinds of datasets, i.e.,

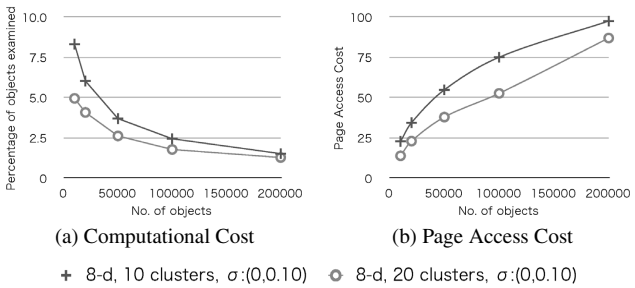
- synthetic vector datasets in an 8-dimensional space with 20 clusters of vectors with standard deviations of from 0 to 0.1 and from 0 to 0.2,
- real vectors, and
- the tag set.

We constructed indexes using 500 to 50,000 samples. The radius of a query in the experiment was set to the distance to the 20-nearest neighbor objects. The edge objects of each cluster were the objects whose reachability-distances were ranked within the top 5% in the cluster. Figure 5 plots the computational and page access costs with respect to sample size. The result indicates that the index performances are not so much affected by the number of samples. Presumably, this is because the vector data has clear clusters. We decided that the clustering in MMMP would be computed using 20,000 random samples from the dataset for the index performance evaluations.



(a) Computational Cost (b) Page Access Cost

Fig. 5 Samples for clustering.

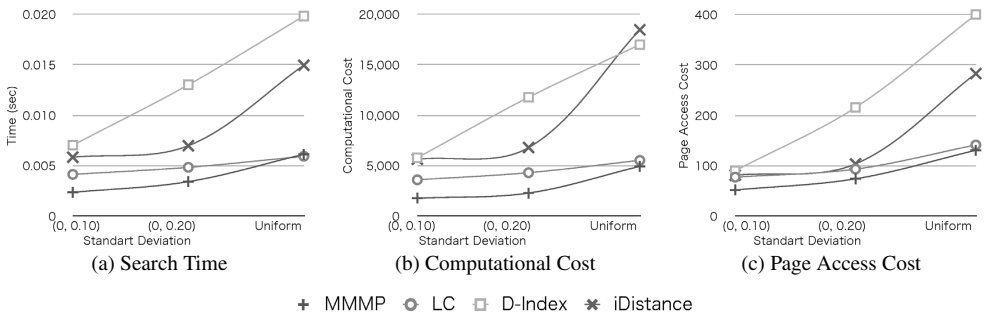


(a) Computational Cost (b) Page Access Cost

Fig. 6 Search cost w.r.t. no. of objects.

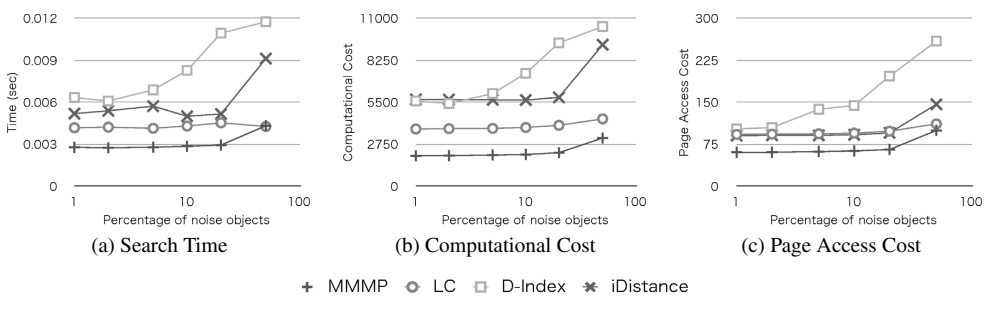
5.4 Synthetic Data

We compared the MMMP-Index with other indexes using synthetic data generated with a wide range of parameters. Figures 6, 7, 8, and 9 show the index performance evaluation results of the synthetic vectors. The radius of a query in the experiment was set to the distance to the 20-nearest neighbor objects. The numbers of objects were fixed to 100,000 in Figs. 7, 8, and 9 whereas the number varied from 10,000 to 200,000 in Fig. 6. We used various variances, the dimensions, numbers of clusters, and the number of objects, as shown in Table 1. It is clear that the MMMP-Index is superior to the other schemes for indexing clustered data. For example, Fig. 7 for 8-dimensional vectors shows that the computational cost of the MMMP-Index is less than two thirds that of the other schemes. The page access cost of the



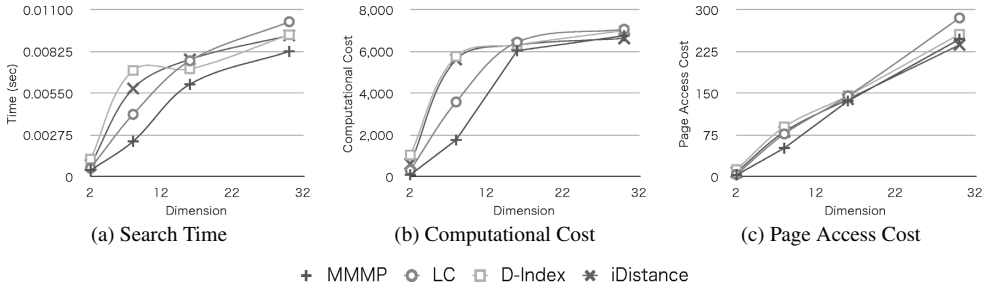
(a) Search Time (b) Computational Cost (c) Page Access Cost

Fig. 7 Search cost w.r.t. variance (8-d, 20 clusters and uniform).



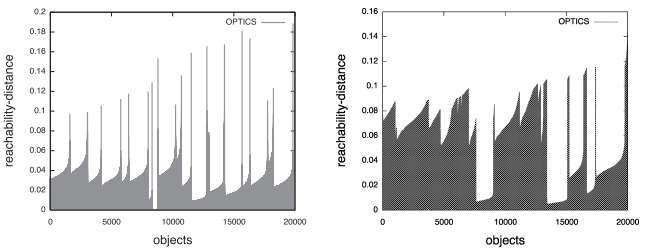
(a) Search Time (b) Computational Cost (c) Page Access Cost

Fig. 8 Search cost w.r.t. noise (8-d, 20 clusters, σ:(0,0.20)).



(a) Search Time (b) Computational Cost (c) Page Access Cost

Fig. 9 Search cost w.r.t. dimension (20 clusters, σ:(0,0.20)).



(a) 8-d, 20 clusters, σ:(0,0.10) (b) 8-d, 20 clusters, σ:(0,0.20)

Fig. 10 Clustering result.

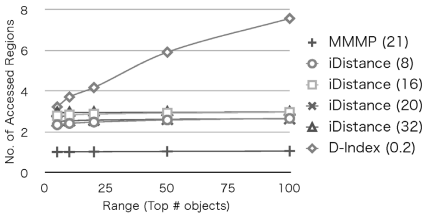


Fig. 11 Partitioning performance (8-d, 20 clusters, σ:(0,0.10)).

MMMP-Index is also less than the other schemes. These results prove that MMMP is useful for clustered vectors. MMMP achieves effective partitioning by exploiting knowledge about the distribution of the clustered data. The results in Fig. 6 indicate that the computational cost of the MMMP-Index for the clustered data with 20 clusters is almost half that of the data with 10 clusters.

Moreover, the results in Fig. 7 indicate the search cost of the MMMP-Index is lower for the clustered data with lower standard deviations. Figure 10 depicts the results of OPTICS clustering. Deeper valleys in the reachability plots mean denser clusters. Higher reachability distance points between valleys mean that clusters are far from each other. From the clustering results, we can clearly recognize that the clustered data with lower standard deviations in Fig. 10 (a) has 18 clusters that is almost same the number of clusters as its parameter. We can also see that the clusters in the data with lower standard deviations are denser and more isolated than the other data. It is more difficult to recognize clusters when the data have higher standard deviations. Thus, we think the difference is because the data with lower standard deviations has a sparser density space and MMMP could easily create a good partitioning boundary.

We also examined the index performance when the dataset included noise objects. Figure 8 shows the results. It is clear that MMMP-Index outperforms the compared methods for wide range of noise ratios. These results indicate that the clustering technique of MMMP is stable against noise.

To evaluate the effectiveness of MMMP separately, we counted the number of accessed regions during query processing. Figure 11 shows the number of the accessed regions while searching for the 8-dimension clustered vector. We compared MMMP with D-Index [12] and iDistance [3]. D-Index selects a pivot based on the mean distances between the pivot and objects, and recursively divides the space by using Excluded Middle Partitioning. iDistance divides the space with a Voronoi partition based on k-means clustering and sets the cluster centers as the pivots. The vertical axes represent the number of accessed regions. The horizontal axis is the query ranges. The number for each line in the legend represents the parameter for indexing. The numbers for MMMP and iDistance are the numbers of clusters, and the numbers for the D-Index are the partitioning distances of the exclusion sets. The results show that the MMMP minimizes the number of accessed regions while searching. Their number was approximately one from the MMMP results. This result indicates that the partitioning based on the cluster edges is better than partitioning based on cluster centers as in iDistance.

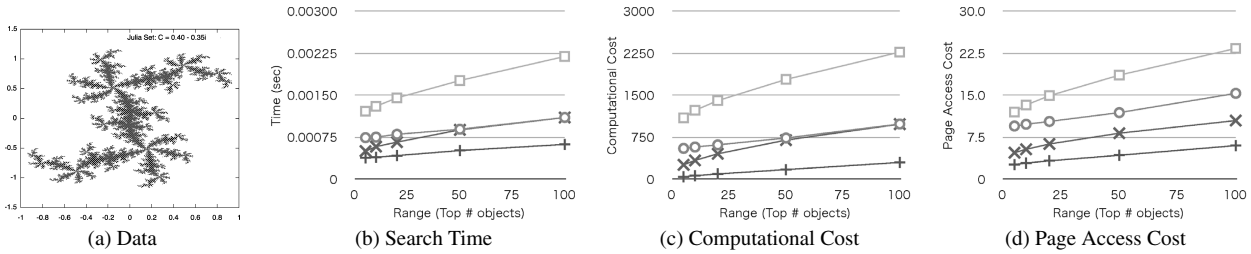
Figure 6 shows the index performance with respect to the number of objects. As in the related work [27], the computational cost was evaluated by the percentage of objects examined. The vertical axis of (a) represents the percentage of objects examined within certain distances from the pivots. The vertical axis of (b) is the page cost. The horizontal axis is the number of objects. From these results, we can see that the search cost of the MMMP-Index is lower for datasets with more clusters, although its cost scales up almost linearly as the size of dataset increases. We interpret this behavior as follows: the MMMP-Index correctly accesses the only region which is relevant to the query (as shown in Figs. 10 and 11); however, the cost of checking the objects in the region varies almost linearly because the objects are indexed by LC. For much larger datasets, we will have to use a more efficient indexing scheme.

The above synthetic vectors have ball-shaped clusters, as they are generated on the basis of a normal distribution. Ball-shaped clusters are easy to classify by pivot partitioning because a pivot makes a spherical boundary. Thus, for

evaluating the index performance for clusters whose shape is not spherical, we compared the MMMP-Index with other indexes by using the Julia set. We selected it because it has hierarchical clusters of various sizes that are not ball-shaped and it can be easily generated. Figure 12 shows the index performances of these methods on the Julia set. It is clear that the MMMP-Index is better than the other methods. Although iDistance sets the cluster centers as pivots, MMMP can select a pivot from all the objects on the basis of its partitioning boundary and render a good partition for search pruning regardless of the clusters' shapes.

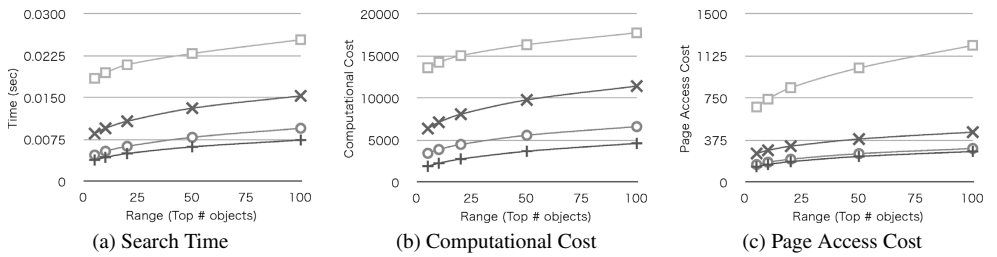
5.5 Real Data

We compared the MMMP-Index with other methods on three real data sets. The datasets have different distance functions. In this experiment, we compared the index performance with respect to the query radius. We set the radiuses to the distances to 5-nearest through 100-nearest neighbors. Figures 13, 14, and 15 show the index per-



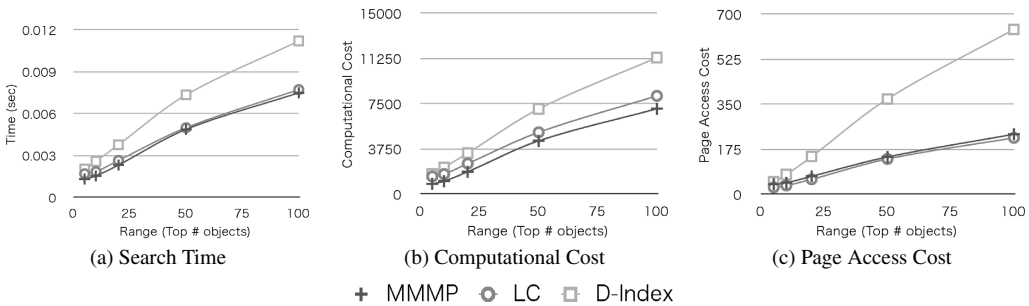
(a) Data (b) Search Time (c) Computational Cost (d) Page Access Cost

Fig. 12 Julia set.



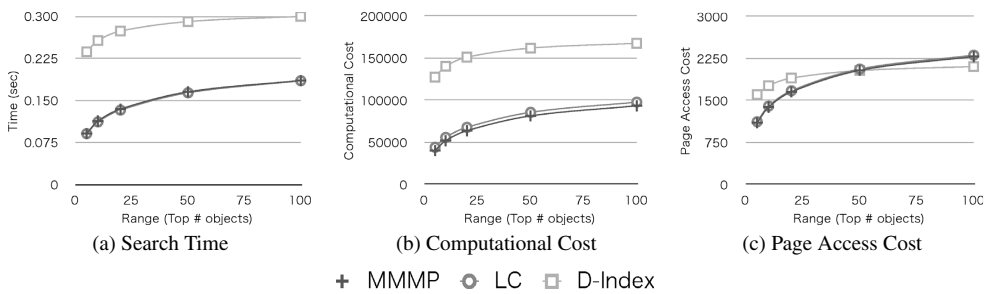
(a) Search Time (b) Computational Cost (c) Page Access Cost

Fig. 13 Corel image features.



(a) Search Time (b) Computational Cost (c) Page Access Cost

Fig. 14 Photo tag sets, query: "tokyo".



(a) Search Time (b) Computational Cost (c) Page Access Cost

Fig. 15 English words.

formance for the Corel image features, the photo set, and the English words, respectively. MMMP-Index outperforms other methods for the Corel image features and the photo tag sets (Figs. 13 and 14). This proves that MMMP-Index is not only effective for synthetic vectors but also the data whose distances obey metric space postulates.

The MMMP-Index and LC perform about the same for English words, because the datasets have few clusters. This means that the MMMP does not work well on them. Since MMMP requires enough clusters for selecting pivots, it would select few pivots for pivot partitioning and the index would mainly depend on pivot filtering in those datasets.

6. Conclusion

We developed MMMP as a means of pivot partitioning for the purpose of pruning in a similarity search index. MMMP divides data on the basis of the shapes of clusters extracted by using OPTICS. During partitioning, MMMP searches for an object to be the best pivot whose partitioning boundary is between clusters and maximizes the distances from the cluster edges. We also created an index, named the MMMP-Index. MMMP is most effective when the indexed data is clustered.

We think that MMMP has three problems. One problem is the pivot selection cost. The current MMMP reduces the relevance evaluation of an object p to the pivot by using edge objects of the clusters. However, the pivot candidate is not filtered from the data. We should improve the pivot selection. Another problem is the performance for a few clusters worth of data and for high dimensional data. The distances between objects in such data are not various. The MMMP cannot extract clusters or margins between clusters from them, and don't work well. The other problem is the pivot refinement. It is hard to refine the pivots generated by the MMMP after the index has been constructed. We should make the MMMP-Index adapt to not only a static data but also changing data. We are currently working on improving the indexing schemes to resolve these problems.

References

[1] P. Zezula, G. Amato, V. Dohnal, and M. Batko, Similarity Search: The Metric Space Approach (Advances in Database Systems), Springer-Verlag, New York, 2005.
[2] P.N. Yianilos, "Data structures and algorithms for nearest neighbor search in general metric spaces," SODA, 1993.
[3] H.V. Jagadish, B.C. Ooi, K.L. Tran, C. Yu, and R. Zhang, "idistance: An adaptive b+-tree based indexing method for nearest neighbor earch," ACM Trans. Database Syst., vol.30, no.2, pp.364–397, 2003.
[4] Y. Zhuang, Y. Zhuang, Q. Li, L. Chen, and Y. Yu, "Indexing high-dimensional data in dual distance spaces: A symmetrical encoding approach," EDBT, 2008.
[5] M. Ankerst, M.M. Breunig, H. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," SIGMOD, 1999.
[6] H. Kurasawa, D. Fukagawa, A. Takasu, and J. Adachi, "Maximal metric margin partitioning for similarity search indexes," CIKM, 2009.
[7] D.P. Huttenlocher, G.A. Klanderman, and W.J. Rucklidge, "Comparing images using the Hausdorff distance," IEEE Trans. Pattern Anal. Mach. Intell., vol.15, no.9, pp.850–863, 1993.
[8] V.I. Levenshtein, "Binary codes capable of correcting spurious insertions and deletions of ones," Problems of Information Transmissions, vol.1, no.1, pp.8–17, 1965.
[9] J.K. Uhlmann, "Satisfying general proximity/similarity queries with metric trees," Inf. Process. Lett., vol.40, no.4, pp.175–179, 1991.
[10] R.A. Baeza-Yates, W. Cunto, U. Manber, and S. Wu, "Proximity matching using fixed-queries trees," CPM, 1994.
[11] P.N. Yianilos, "Excluded middle vantage point forests for nearest neighbor search," ALENEX, 1999.
[12] V. Dohnal, C. Gennaro, P. Savino, and P. Zezula, "D-index: Distance searching index for metric data sets," Multimedia Tools and Applications, vol.21, no.1, pp.9–33, 2003.
[13] T. Bozkaya and Z.M. Ozsoyoglu, "Indexing large metric spaces for similarity search queries," ACM Trans. Database Syst., vol.24, no.3, pp.361–404, 1999.
[14] J. Venkateswaran, D. Lachwani, T. Kahveci, and C. Jermaine, "Reference-based indexing of sequence databases," VLDB, 2006.
[15] B. Bustos, G. Navarro, and E. Chevez, "Pivot selection techniques for proximity searching in metric spaces," Pattern Recognit. Lett., vol.24, no.14, pp.2357–2366, 2003.
[16] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An efficient access method for similarity search in metric spaces," VLDB, 1997.
[17] O. Pedreira and N.R. Brisaboa, "Spatial selection of sparse pivots for similarity search in metric spaces," SOFSEM, 2007.
[18] D. Novak and P. Zezula, "M-chord: A scalable distributed similarity search structure," INFOSCALE, 2006.
[19] M. Batko, D. Novak, F. Falchi, and P. Zezula, "On scalability of the similarity search in the world of peers," INFOSCALE, 2006.
[20] C. Doulkeridis, A. Vlachou, Y. Kotidis, and M. Vazirgiannis, "Peer-to-peer similarity search in metric spaces," VLDB, 2007.
[21] E. Chevez and G. Navarro, "A compact space decomposition for effective metric indexing," Pattern Recognit. Lett., vol.24, no.9, pp.1363–1376, 2005.
[22] S. Brecheisen, H. Kriegel, and M. Pfeifle, "Multi-step density-based clustering," Knowledge and Information Systems, vol.9, no.3, pp.284–308, 2006.
[23] "Uci kdd archive, http://kdd.ics.uci.edu/"
[24] "flickr, http://www.flickr.com/"
[25] "Wordnet, http://wordnet.princeton.edu/"
[26] L. Carleson and T.W. Gamelin, Complex Dynamics, Springer-Verlag, New York, 1993.
[27] G. Navarro, "Searching in metric spaces by spatial approximation," VLDB Journal, vol.11, no.1, pp.28–46, 2002.

Hisashi Kurasawa received the B.E. and M.E. in Information Science and Technology from the University of Tokyo, Tokyo, Japan in 2006 and 2008. He is currently a Ph.D. candidate in the Graduate School of Information Science and Technology at the University of Tokyo. His research interests are distributed information retrieval systems and context-aware computing. He is a student member of IPSJ and DBSJ.

Daiji Fukagawa received the B.E. degree in Computer Science in 2001, M.S. degree in Informatics in 2003, and Ph.D. degree in Informatics in 2006, all from Kyoto University. Since 2006, he has been working for the National Institute of Informatics as a Project Researcher. His research interests include the theory of combinatorial optimization for trees and graphs. He is a member of IPSJ and DBSJ.

Atsuhiro Takasu received B.E., M.E. and Dr. Eng. from the University of Tokyo in 1984, 1986 and 1989, respectively. He is a professor at the National Institute of Informatics, Japan. His research interests are database systems and machine learning. He is a member of ACM, IEEE, IPSJ, DBSJ and JSAI.

Jun Adachi is a Professor in the Digital Content and Media Sciences Research Division, National Institute of Informatics (NII), Japan. He is also the Director of the Cyber Science Infrastructure Development Department of NII. His professional career has largely been spent in research and development of scholarly information systems, such as NACSIS-CAT and NII-ELS. He is also an adjunct professor of the Graduate School of Information Science and Technology (Department of Information and Communication Engineering), University of Tokyo. His research interests are information retrieval, text mining, digital library systems, and distributed information systems. Adachi received his B.E., M.E. and Doctor of Engineering in Electrical Engineering from the University of Tokyo in 1976, 1978, and 1981, respectively. He is a member of IPSJ, IEEE, and ACM.