IEICE TRANS. INF. & SYST., VOL.E101–D, NO.1 JANUARY 2018

PAPER

Pivot Generation Algorithm with a Complete Binary Tree for Efficient Exact Similarity Search

Yuki YAMAGISHI[†a)], *Nonmember*, Kazuo AOYAMA[††], Kazumi SAITO[†], *and* Tetsuo IKEDA[†], *Members*

SUMMARY This paper presents a pivot-set generation algorithm for accelerating exact similarity search in a large-scale data set. To deal with the large-scale data set, it is important to efficiently construct a search index offline as well as to perform fast exact similarity search online. Our proposed algorithm efficiently generates competent pivots with two novel techniques: hierarchical data partitioning and fast pivot optimization techniques. To make effective use of a small number of pivots, the former recursively partitions a data set into two subsets with the same size depending on the *rank order* from each of two assigned pivots, resulting in a complete binary tree. The latter calculates a defined objective function for pivot optimization with a low computational cost by skillfully operating data objects mapped into a *pivot space*. Since the generated pivots provide the tight lower bounds on distances between a query object and the data objects, an exact similarity search algorithm effectively avoids unnecessary distance calculations. We demonstrate that the search algorithm using the pivots generated by the proposed algorithm reduces distance calculations with an extremely high rate regarding a range query problem for real large-scale image data sets.

key words: *similarity search, pivot generation, complete binary tree*

1. Introduction

Similarity search algorithms for large-scale data sets have been developed on the growing demand [10], [22], [24]. Recently, for high-dimensional data sets, approximation algorithms represented by locality-sensitive hashing (LSH) family have been studied, which save the computational cost for search at the expense of obtaining exact solutions [2], [8], [14]. In contrast, exact similarity search algorithms, which find all objects exactly meeting a given query, have received interest for a long time in various application domains, for example, a simple one is to search a data set with a relatively low intrinsic dimensionality [22]. Furthermore, their acceleration is necessary for the accuracy evaluation of heuristic algorithms and the approximation algorithms since an exact search result is used as the ground truth.

We consider to accelerate an exact similarity search algorithm for large-scale and high-dimensional data sets by reducing a computational cost. In particular, we focus on the exact algorithm for a range query since it is the most basic type of queries. A general approach for the acceleration is to omit unnecessary distance calculations of data objects and a query object (query for short) in a search stage. To judge whether a distance calculation between a data object and a query is necessary or not beforehand, the lower bound on the distance is utilized, which is derived based on the *triangle inequality* in a metric space. The distance lower bound of an object and a query is calculated by using a triangle of the object, the query, and a distinguished object or fixed reference point called a *pivot* [6], [9], [18], [23]. Most of the accelerated algorithms employ a search index constructed by using pivots [6], [9], [23].

In those algorithms, a pivot selection is the most important procedure for a significant acceleration. To select better pivots, several objective functions for pivot optimization have been reported [1], [7], [17]. Of these, the incremental selection in [7] sequentially adopts an object as a pivot one by one, which maximizes the sum of the lower bounds on distances between objects. This objective function is derived straightforwardly if it is assumed that a query appears in a given space with the identical distribution to that of the data objects.

This objective function has been also employed by algorithms for Euclidean spaces as well as for the general metric spaces because of its validity [15], [16]. In the Euclidean space, the pivots generated by the algorithm in [15] outperformed those selected by the incremental selection in terms of the reduction of the number of the distance calculations. However, the pivot generation algorithm requires the high computational complexity of $O(N^2)$, where N denotes the number of objects in a given data set. This high complexity makes it difficult to apply the pivot generation algorithm to a large-scale data set.

To alleviate the above-mentioned drawback and significantly accelerate a similarity search algorithm for a large-scale data set, we propose a novel pivot-set generation algorithm. The proposed algorithm generates a set of pivots with the computational complexity of $O(N \log N)$ offline. Moreover, a search algorithm can effectively avoid unnecessary distance calculations online by using the tighter distance lower bounds based on the generated pivot set. The above-mentioned performance is due to newly introduced two techniques: hierarchical data partitioning and fast pivot optimization. The former recursively partitions a data set (or subset) into two subsets with the same size depending on the *rank order* from each pivot, resulting in a complete binary tree where each node has a pair of pivots. The latter is characterized by fast calculation of the objective function

Manuscript received March 2, 2017.
Manuscript revised July 18, 2017.
Manuscript publicized October 20, 2017.

[†]The authors are with University of Shizuoka, Shizuoka-shi, 422–8526 Japan.
[††]The author is with NTT Communication Science Laboratories, Kyoto-fu, 619–0237 Japan.
a) E-mail: yamagissy@gmail.com

DOI: 10.1587/transinf.2017EDP7077

Copyright © 2018 The Institute of Electronics, Information and Communication Engineers

Table 1 Notation

Symbol	Description and Definitions
N	**Number** of objects, $N = \|\mathcal{X}\|$
r	**Range** for range search
H	**Dimensionality** of feature vector of object
L	Maximum **level** of complete binary tree index
T	**Number** of PGM iterations for pivot generation
ϵ	**Parameter** for stopping PGM iterations
i	**Index** for tree level
j	**Index** for node in CBT
v_j	**Node** in CBT
$\mathcal{S}$	**Set** of objects in a metric space $\mathcal{M}$, $\{\mathcal{X}, \mathcal{Z}\} \subset \mathcal{S}$
$\mathcal{X}$	**Set** of given objects, $\mathcal{X} = \{x_1, x_2, \cdots, x_N\}$
$\mathcal{Z}$	**Set** of query objects for search, $\mathcal{Z} = \{z, \cdots\}$
$\mathcal{P}$	**Set** of pivots, $\mathcal{P} = \{p_1, p_2, \cdots, p_{\|\mathcal{P}\|}\}$
$\mathcal{X}_j$	**Set** of objects included in node v_j
$\mathcal{A}(x)$	**Set** of objects to which distances from a pivot are smaller than a distance to object x
$\mathcal{B}(x)$	**Set** of objects to which summations of distances from two pivots are smaller than a distance to x
$C(x)$	**Set** of objects to which differences of distances from two pivots are smaller than a distance to x
$d(x_s, x_t)$	**Function** that returns a distance between objects x_s and x_t
$D(x_s, x_t; \mathcal{P})$	**Function** that returns the lower bound on a distance between object x_s and x_t by pivot set $\mathcal{P}$
$F(\mathcal{P}; \mathcal{X})$	**Function** that returns the sum of lower-bound distances of objects in $\mathcal{X}$ by pivot set $\mathcal{P}$
$\rho(x)$	**Function** that returns a rank order of x, which is equal to $\|\mathcal{A}(x)\| + 1$
$J(\mathcal{X}, z, r; \mathcal{P})$	**Function** that returns a set of discarded objects in $\mathcal{X}$ for query z and range r by pivot set $\mathcal{P}$
$\overline{Cal}(r)$	**Function** that returns an average computational cost on search for range r

based on a skillful operation of objects mapped into a *pivot space* [10], [17]. We demonstrate that the proposed algorithm efficiently generates an effective pivot set through our experimental results on large-scale real image data sets.

The remainder of this paper is organized as follows. Section 2 briefly reviews related work. Section 3 describes pivot-based similarity search as preliminaries. Section 4 explains our proposed algorithm in detail. Section 5 shows the experimental settings and reports our results. The final section offers our conclusions. We summarize the notation in Table 1 for easy reference.

2. Related Work

This section reviews two related topics: pivot selection and generation algorithms and pivot-based data partition algorithms for index construction.

2.1 Pivot Selection and Generation Algorithms

An advantage of our proposed algorithm is a fast pivot optimization technique. The existing pivot selection and generation algorithms are here described as regards their objective functions and computational costs.

The objective functions have been designed depending on their usage and hypotheses [1], [9]. A hypothesis is that better pivots are far away from each other [6], [18]. A simple algorithm based on the hypothesis sequentially selects a pivot $\hat{p}$ from a set of data objects $\mathcal{X} = \{x_1, x_2, \cdots, x_N\}$ one by one with the same procedure as ϵ-net construction in [12], [13], as expressed by

$$\hat{p} = \mathop{\arg\max}_{x_i \in \mathcal{X} \setminus \mathcal{P}} \{\min_{p \in \mathcal{P}} \{d(x_i, p)\}\}, \quad (1)$$
$$\mathcal{P} = \mathcal{P} \cup \{\hat{p}\},$$

where $\mathcal{P}$ denotes the set of the selected pivots and $d(x_i, p)$ a distance between x_i and p. Another sequential algorithm selects a pivot $\hat{p}$ in the similar manner to that in Eq. (1) as follows [18].

$$\hat{p} = \mathop{\arg\max}_{x_i \in \mathcal{X} \setminus \mathcal{P}} \{ \textstyle\sum_{p \in \mathcal{P}} d(x_i, p) \}, \quad (2)$$

In contrast, there is a selection algorithm based on the hypothesis that a given query object appears with the identical distribution to that of the data objects. The algorithm incrementally selects such a pivot $\hat{p}$ that maximizes the sum of the lower bounds on distances between a pair of data objects, one of which is regarded as the query object [7], as shown in Eqs. (3)–(5). Maximizing $F(\mathcal{P}'; \mathcal{X} \setminus \mathcal{P}')$ means to make the average lower bound tighter. As in Eq. (5), the distance lower bound is derived by the *triangle inequality* applied to the triangle with the pivot and two objects.

$$\hat{p} = \mathop{\arg\max}_{p \in \mathcal{X} \setminus \mathcal{P}} \{ F(\mathcal{P}'; \mathcal{X} \setminus \mathcal{P}') \}, \quad (3)$$
$$\mathcal{P}' = \mathcal{P} \cup \{p\},$$
$$F(\mathcal{P}'; \mathcal{X} \setminus \mathcal{P}') = \textstyle\sum_{x_i, x_j \in \mathcal{X} \setminus \mathcal{P}'} D(x_i, x_j; \mathcal{P}'), \quad (4)$$
$$D(x_i, x_j; \mathcal{P}') = \max_{p' \in \mathcal{P}'} \left(|\, d(x_j, p') - d(x_i, p') \,| \right), \quad (5)$$

where $\mathcal{P} = \emptyset$ at the initial state and $j > i$ in Eq. (4). By the incremental selection of $\hat{p}$, $\mathcal{P}$ is updated like $\mathcal{P} \leftarrow \mathcal{P} \cup \{\hat{p}\}$. This algorithm is referred to as PSM (Pivot Selection by Maximizing the objective function) because it selects a pivot set $\mathcal{P}$ from a given object set $\mathcal{X}$, i.e., $\mathcal{P} \subset \mathcal{X}$. Note that it requires the computational complexity of $O(N^3)$ even for selecting only one pivot $\{\hat{p}\}$ such that

$$\hat{p} = \mathop{\arg\max}_{p \in \mathcal{X}} \{ F(\{p\}; \mathcal{X}) \}.$$

Since the objective function in Eq. (4) is valid for some applications, it has been utilized in a Euclidean space. Then pivots are not selected from data objects in $\mathcal{X}$ but are generated as points in a Euclidean space by maximizing the objective function $F(\mathcal{P}; \mathcal{X})$, which is expressed as

$$F(\mathcal{P}; \mathcal{X}) = \textstyle\sum_{x_i, x_j \in \mathcal{X}} D(x_i, x_j; \mathcal{P})$$
$$= \textstyle\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} D(x_i, x_j; \mathcal{P}), \quad (6)$$
$$D(x_i, x_j; \mathcal{P}) = \max_{p \in \mathcal{P}} (\,|\, d(x_j, p) - d(x_i, p) \,|\,). \quad (7)$$

Along this line of the research direction, iterative algorithms for the L_2 and for the L_1 Euclidean space have been proposed in [15] and [16], respectively. These algorithms are collectively referred to as PGM (Pivot Generation by Maximizing the objective function). The PGM algorithm first initializes a set of pivots as arbitrary points, and then iteratively updates these pivot so as to maximize the objective function $F(\mathcal{P}; \mathcal{X})$. However, the algorithm still requires a high computational cost for each iteration. This requirement makes it difficult to apply the algorithm to a large-scale data set. Thus a lower computational cost for the pivot generation is desired for the large-scale data set.

2.2 Pivot-Based Data Partitioning Algorithms

In order to exploit a small number of pivots efficiently, our proposed algorithm partitions a given data set into subsets. The data partitioning and clustering algorithms based on pivots are briefly reviewed in terms of search index construction.

In the most cases, data objects are partitioned by a *distance measure* from pivots. A simple algorithm for constructing the metric tree recursively employs two distinct schemes: generalized hyperplane partitioning and ball partitioning [23]. The former partitions the data set into two subsets depending on a distance of the data object to each of two pivots. The latter divides the objects into two subsets depending on whether each object exists inside or outside the ball whose center is a single pivot and radius is the median of distances from the pivot to all the objects. Other than the dual-partitioning leading to the binary tree, most of the algorithms partition the data set into multiple subsets based on distances from multiple pivots and construct a k-ary tree by recursive partitioning [6], [11], [20], where the k value is not always a constant.

Few algorithms partition the data set into subsets based on the *size* (the number of objects) of the partitioned subset. Of these, the algorithm for *list of clusters* (LC) constructs an unbalanced binary tree using a variable radius from a single pivot (cluster center), which is controlled so that the number of data objects inside the ball becomes a pre-defined fixed number [9].

3. Preliminaries: Pivot-Based Similarity Search

We employ similarity search performance as a practical indicator of a quality of pivot set $\mathcal{P}$. Here, we briefly describe an exact similarity search using a pivot set, in particular, a range query that is used for the evaluation in Sect. 5.3. Let $\mathcal{M}$ be a metric space consisting of an object set $\mathcal{S}$ and distance $d(x_i, x_j)$ for any pair of objects $x_i, x_j \in \mathcal{S}$. Given a set of objects $\mathcal{X} = \{x_1, x_2, \cdots, x_N\} \subset \mathcal{S}$, a query object $z \in \mathcal{S}$, and a range distance r, a range query is defined as a task of finding a set of all the objects x_i satisfying $d(x_i, z) \leq r$. As described in Sect. 1, in order to reduce a computational cost for calculating distances between the query and objects in the task, we exploit a pivot-based algorithm. More specifically, we focus on the algorithm that employs the following lower bound on a distance between an object x_i and a query z using a set of pivots $\mathcal{P} = \{p_1, p_2, \cdots, p_{|\mathcal{P}|}\}$:

$$D(x_i, z; \mathcal{P}) = \max_{p \in \mathcal{P}} |\, d(x_i, p) - d(z, p)\,| \leq d(x_i, z). \tag{8}$$

Note that this lower bound is easily derived from the basic properties of a metric space, i.e., non-negativity, identity, symmetry, and the triangle inequality. Based on Eq. (8), a set of the discarded objects $J(\mathcal{X}, z, r; \mathcal{P}) \subset \mathcal{X}$ is defined by

$$J(\mathcal{X}, z, r; \mathcal{P}) = \{x \in \mathcal{X} \mid D(x, z; \mathcal{P}) > r\}\ . \tag{9}$$

Obviously, we do not need to calculate the distance between the query and any object in the set $J(\mathcal{X}, z, r;\ \mathcal{P})$. More distance calculations are saved, lower a computational cost becomes. That is, a better pivot set $\mathcal{P}$ in terms of the quality makes $|J(\mathcal{X}, z, r;\ \mathcal{P})|$ larger.

4. Proposed Algorithm

Our proposed algorithm consists of two main parts: a pivot generation algorithm that is an accelerated PGM for two pivots, and a data partitioning algorithm that partitions a set (or subset) of objects into two subsets with almost the same size by using the obtained two pivots, resulting in a complete binary tree (CBT for short) for a search index. Hereinafter, we refer to our proposed algorithm as PGM-CBT. We begin with an outline of the proposed algorithm PGM-CBT and then detail the two main algorithms in Sects. 4.1 and 4.2.

Algorithm 1 shows our proposed algorithm that constructs a CBT by growing branches from a root to leaf nodes level by level, which is identified by parameter i. A node in the CBT contains object subset $\mathcal{X}_j$ and two pivots $\{p_{j0}, p_{j1}\}$.

M1: At line 3, create a tree consisting of only a root node v_1 whose member $\mathcal{X}_1$ is set up by $\mathcal{X}_1 \leftarrow \mathcal{X}$, and initialize a tree level parameter i by $i \leftarrow 1$ and a set of pivots by $\mathcal{P} \leftarrow \emptyset$.

M2: At lines 4–12, for each leaf node v_j, $2^{(i-1)} \leq j < 2^i$, containing $\mathcal{X}_j$ at level i of the current tree, perform the following two algorithms, and the procedure is repeated while $i < L$.

M2-1: At lines 6 and 7, generate a pair of pivots $\{p_{j0}, p_{j1}\}$ using $\mathcal{X}_j$ by the PGM algorithm with a new technique

Algorithm 1 Proposed algorithm (PGM-CBT)

```
 1: Input: $\mathcal{X}, L$
 2: Output: $\mathcal{P}$
 3: Initialize with $\mathcal{X}_1 \leftarrow \mathcal{X},\ \mathcal{P} \leftarrow \emptyset,\ i \leftarrow 1$.
 4: while $i < L$ do
 5:     for all $j$ such that $2^{(i-1)} \leq j < 2^i$ do
 6:         Execute pivot generation algorithm and obtain $\{p_{j0},\ p_{j1}\}$.
 7:         $\mathcal{P} \leftarrow \mathcal{P} \cup \{p_{j0}, p_{j1}\}$
 8:         Execute data partitioning algorithm and obtain $\{\mathcal{X}_{j0},\ \mathcal{X}_{j1}\}$.
 9:         Create child nodes with $\mathcal{X}_{2j} \leftarrow \mathcal{X}_{j0}, \mathcal{X}_{(2j+1)} \leftarrow \mathcal{X}_{j1}$.
10:     end for
11:     $i \leftarrow i + 1$
12: end while
13: return $\mathcal{P}$
```

to efficient calculate the objective function shown in Sect. 4.1. Then add these pivots as $\mathcal{P} \leftarrow \mathcal{P} \cup \{p_{j0}, p_{j1}\}$.

M2-2: At lines 8 and 9, partition $\mathcal{X}_j$ into $\mathcal{X}_{j0}$ and $\mathcal{X}_{j1}$, as shown in Sect. 4.2. Then produce the two child nodes v_{2j} and $v_{(2j+1)}$ at the next level, which contain $\mathcal{X}_{2j}$ and $\mathcal{X}_{(2j+1)}$, respectively.

M3: If $i = L$, output $\mathcal{P}$ and terminate at line 13.

4.1 Pivot Generation Algorithm

We present a novel pivot generation algorithm that produces a pair of pivots $\{p_{j0}, p_{j1}\}$ using a set of objects $\mathcal{X}_j$. We first overview a generic procedure of PGM as follows.

G1: Initialize $\mathcal{P}$ by two objects randomly selected from $\mathcal{X}$, and iterate the following three steps until convergence.
G1-1: Calculate distance $d(x, p)$ for each pair of p and x.
G1-2: Compute the objective function $F(\mathcal{P}; \mathcal{X})$.
G1-3: Update each pivot $p \in \mathcal{P}$ so as to maximize $F(\mathcal{P}; \mathcal{X})$.
G2: Output $\mathcal{P}$ and terminate.

In our experiments, we employed a convergence criterion defined by

$$F(\mathcal{P}'; \mathcal{X}) / F(\mathcal{P}; \mathcal{X}) < (1+\epsilon),$$

where $\mathcal{P}'$ means an updated set of the pivots from $\mathcal{P}$, and we used $\epsilon = 10^{-8}$. We omit the details of Step **G1-3** due to space limitations, which is the same as that in [15], [16].

We show the computational complexity of the pivot generation algorithm. In the case of an H-dimensional vector space with L_1 or L_2 metric, the complexity at both Steps **G1-1** and **G1-3** becomes $O(NH)$ as analyzed in [15], [16] although an actual computational cost depends on a metric space, where N stands for the number of given objects. The complexity at Step **G1-2** becomes $O(N^2)$ in general. As shown below, by using our proposed technique in the case of two pivots, we can reduce the complexity from $O(N^2)$ to $O(N \log N)$. Namely, we can obtain the computational complexity of the pivot generation algorithm as follows:

$$O(\,TN(H + \log N)\,),$$

where T means the number of iterations until the PGM algorithm converges. Thus, the pivot generation algorithm has reasonably good scalable properties that the computational complexity to N is quasi-linear, and linear to H. In our experiments, T was less than 100 in most trials as shown later.

We explain our proposed technique of calculating the objective function efficiently. First, we describe the case of one pivot p for $\mathcal{X}_j$. For simplicity, we assume that two different objects has a different distance from the pivot although this restriction is resolved later. Figure 1 shows an example for intuitively understanding the proposed technique when $\mathcal{X} = \{x_1, x_2, x_3, x_4\}$. The objective function value is the sum of the six entries in the distance lower bound matrix in Fig. 1 (a). We map the objects x_s in the original space into the object vectors $\mathbf{u}_s$ in the 1-dimensional pivot space where each object's position is a distance from the pivot p, i.e., $d(x_s, p)$ shown in Fig. 1 (b). Suppose $d(x_2, p) < d(x_4, p) < d(x_1, p) < d(x_3, p)$. We attach a rank order $\rho(\mathbf{u}_s)$ to each object $\mathbf{u}_s$ in descending order of $d(x_s, p)$, and then regard $\rho(x_s)$ as $\rho(\mathbf{u}_s)$. For instance, suppose that the rank order of $\mathbf{u}_1$ is 2, and when the absolute operations are eliminated in the entries in Fig. 1 (a), $-d(x_1, p)$ and $+d(x_1, p)$ appears once and twice. Using the rank orders of all objects, we obtain the distance lower bound matrix in Fig. 1 (c). Since $-d(x_s, p)$ and $+d(x_s, p)$ appear $\rho(x_s)-1$ times and $|\mathcal{X}|-\rho(x_s)$ times, respectively, the objective function value is expressed by

$$\begin{aligned}&\textstyle\sum_{s=1}^{4}(|\mathcal{X}| - 2\rho(x_s) + 1)\, d(x_s, p)\\ &\quad = d(x_1, p) - 3\,d(x_2, p) + 3\,d(x_3, p) - d(x_4, p)\,.\end{aligned}$$

To deal with a general case, we define $\mathcal{A}(x_s) \subset \mathcal{X}_j$ and $\overline{\mathcal{A}(x_s)}$ for each object x_s as

$$\mathcal{A}(x_s) = \{x_t \mid d(x_s, p) < d(x_t, p)\} \quad \text{and}$$
$$\overline{\mathcal{A}(x_s)} = \mathcal{X}_j \setminus (\mathcal{A}(x_s) \cup \{x_s\})\,.$$

We can eliminate the absolute operation as follows:

$$|d(x_t, p) - d(x_s, p)| = \begin{cases} d(x_t, p) - d(x_s, p) & x_t \in \mathcal{A}(x_s), \\ -d(x_t, p) + d(x_s, p) & x_t \in \overline{\mathcal{A}(x_s)} \end{cases}$$

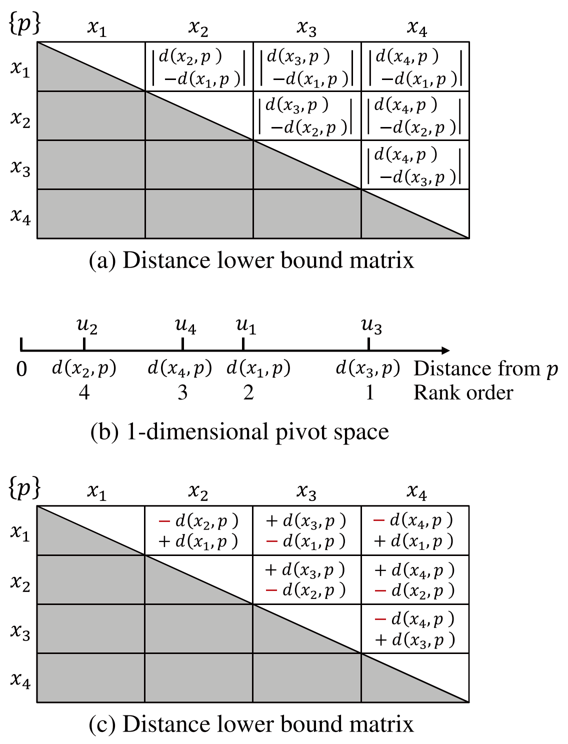


Fig. 1 Example of distance lower bound matrices when object set $\mathcal{X} = \{x_1, x_2, x_3, x_4\}$ ($N = 4$) and pivot p are used. (a) Distance lower bound matrix: The value at the s-th row and the t-th column is $|d(x_t, p) - d(x_s, p)|$. (b) 1-dimensional pivot space where $\mathbf{u}_s$ corresponds to x_s in the original space and its position is $d(x_s, p)$. Each object $\mathbf{u}_s$ is given a rank order $\rho(\mathbf{u}_s)$ in descending order of $d(x_s, p)$. (c) Distance lower bound matrix based on the rank order in the pivot space. $\rho(\mathbf{u}_s) - 1$ is the frequency with which $-d(x_s, p)$ appears in the entries.

This equation relates to the rank order of $\mathbf{u}_s$ and $\mathbf{u}_t$ in the 1-dimensional pivot space in the foregoing example. In the calculation of the objective function, the frequencies with which $-d(x_s, p)$ and $d(x_s, p)$ appear are $|\mathcal{A}(x_s)|$ times and $|\mathcal{X}_j| - 1 - |\mathcal{A}(x_s)|$ times, respectively. Based on the fact, the objective function $F(\{p\}; \mathcal{X}_j)$ is expressed by

$$F(\{p\}; \mathcal{X}_j) = \sum_{x_s \in \mathcal{X}_j} (|\mathcal{X}_j| - 1 - 2|\mathcal{A}(x_s)|)\, d(x_s, p)\,. \qquad (10)$$

Formally, for a given $x_t \in \mathcal{X}_j$, the numbers of objects such that $x_t \in \mathcal{A}(x_s)$ and $x_t \in \overline{\mathcal{A}(x_s)}$ can be obtained as

$$|\{x_s \mid x_t \in \mathcal{A}(x_s)\}| = |\{x_s \mid d(x_s, p) < d(x_t, p)\}| = |\overline{\mathcal{A}(x_t)}|$$

and

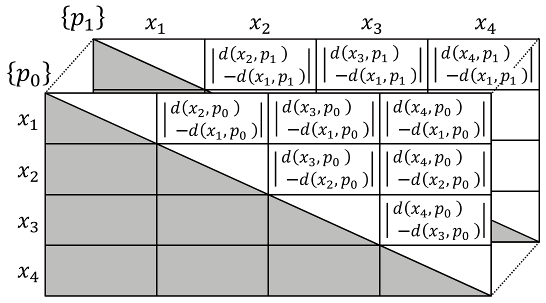
$$|\{x_s \mid x_t \in \overline{\mathcal{A}(x_s)}\}| = |\{x_s \mid d(x_t, p) < d(x_s, p)\}| = |\mathcal{A}(x_t)|.$$

Using these relationships, we can calculate the objective function as follows:

$$\begin{aligned}
F(\{p\}; \mathcal{X}_j) &= \frac{1}{2} \sum_{x_s \in \mathcal{X}_j} \sum_{x_t \in \mathcal{X}_j \setminus \{x_s\}} |d(x_t, p) - d(x_s, p)| \\
&= \frac{1}{2} \sum_{x_s \in \mathcal{X}_j} \sum_{x_t \in \mathcal{A}(x_s)} (\, d(x_t, p) - d(x_s, p)\,) \\
&\quad + \frac{1}{2} \sum_{x_s \in \mathcal{X}_j} \sum_{x_t \in \overline{\mathcal{A}(x_s)}} (\, d(x_s, p) - d(x_t, p)\,) \\
&= \frac{1}{2} \sum_{x_t \in \mathcal{X}_j} |\overline{\mathcal{A}(x_t)}| d(x_t, p) - \frac{1}{2} \sum_{x_s \in \mathcal{X}_j} |\mathcal{A}(x_s)| d(x_s, p) \\
&\quad + \frac{1}{2} \sum_{x_s \in \mathcal{X}_j} |\overline{\mathcal{A}(x_s)}| d(x_s, p) - \frac{1}{2} \sum_{x_t \in \mathcal{X}_j} |\mathcal{A}(x_t)| d(x_t, p) \\
&= \sum_{x_s \in \mathcal{X}_j} |\overline{\mathcal{A}(x_s)}| d(x_s, p) - \sum_{x_s \in \mathcal{X}_j} |\mathcal{A}(x_s)| d(x_s, p) \\
&= \sum_{x_s \in \mathcal{X}_j} \Big(|\mathcal{X}_j| - 1 - 2\,|\mathcal{A}(x_s)| \Big)\, d(x_s, p) \qquad (11) \\
&= \sum_{x_s \in \mathcal{X}_j} \Big(|\mathcal{X}_j| - 2\rho(x_s) + 1 \Big)\, d(x_s, p)\,, \qquad (12)
\end{aligned}$$

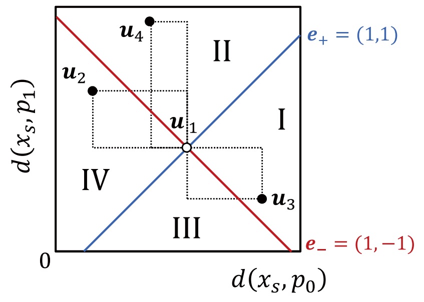
where $|\mathcal{A}(x_s)| = \rho(x_s) - 1$. Consider a situation that two different objects has the same distance from p. We resolve ties of the same distance arbitrarily since the values for the same distances are mutually canceled by arbitrary order.

We extend our technique obtained in the case of one pivot to the case of two pivots $\{p_{j0}, p_{j1}\}$. We begin with an example in Fig. 2, where $\mathcal{X} = \{x_1, x_2, x_3, x_4\}$ and $\{p_0, p_1\}$ are given. Unlike the one-pivot case, the distance lower bound matrix has double-layered structure as in Fig. 2 (a). We need two operations of (1) selection of a larger value in the corresponding entries in the $\{p_0\}$ layer (front) and the $\{p_1\}$ layer (back) in Fig. 2 (a) and (2) elimination of the absolute operation of the larger value. To perform the operations at one time, we map the objects x_s into the object vectors $\mathbf{u}_s$ in the 2-dimensional pivot space where the object's position is $(\, d(x_s, p_0), d(x_s, p_1)\,)$ shown in Fig. 2 (b). Consider

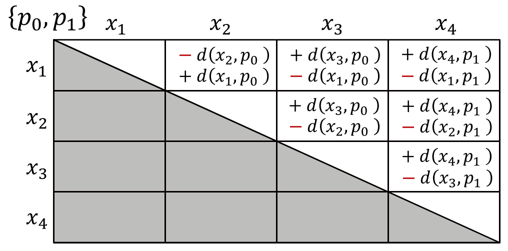


$\{p_1\}$
x_1 x_2 x_3 x_4
$\{p_0\}$
x_1 x_2 x_3 x_4
$|d(x_2,p_1) - d(x_1,p_1)|$
$|d(x_3,p_1) - d(x_1,p_1)|$
$|d(x_4,p_1) - d(x_1,p_1)|$
$|d(x_2,p_0) - d(x_1,p_0)|$
$|d(x_3,p_0) - d(x_1,p_0)|$
$|d(x_4,p_0) - d(x_1,p_0)|$
$|d(x_3,p_0) - d(x_2,p_0)|$
$|d(x_4,p_0) - d(x_2,p_0)|$
$|d(x_4,p_0) - d(x_3,p_0)|$

(a) Distance lower bound matrix



(b) 2-dimensional pivot space



$\{p_0, p_1\}$
x_1 x_2 x_3 x_4
x_1 x_2 x_3 x_4
$-d(x_2,p_0) + d(x_1,p_0)$
$+d(x_3,p_0) - d(x_1,p_0)$
$+d(x_4,p_1) - d(x_1,p_1)$
$+d(x_3,p_0) - d(x_2,p_0)$
$+d(x_4,p_1) - d(x_2,p_1)$
$+d(x_4,p_1) - d(x_3,p_1)$

(c) Distance lower bound matrix

Fig. 2 Example of distance lower bound matrices when object set $\mathcal{X} = \{x_1, x_2, x_3, x_4\}$ and two pivots of p_0 and p_1 are used. (a) Distance lower bound matrix with the double-layer structure: the front and the back layer are based on distances from p_0 and p_1, respectively. (b) 2-dimensional pivot space where $\mathbf{u}_s = (\, d(x_s, p_0), d(x_s, p_1)\,)$. (c) Distance lower bound matrix based on the object positions in the pivot space.

the first rows in Fig. 2 (a), that is, regard $\mathbf{u}_1$ as the origin in Fig. 2 (b). Furthermore, we introduce two vectors $\mathbf{e}_+ = (1, 1)$ and $\mathbf{e}_- = (1, -1)$ in the pivot space, and two lines through $\mathbf{u}_1$ and parallel to each of $\mathbf{e}_+$ and $\mathbf{e}_-$. The two lines divide the pivot space into the four regions denoted by I, II, III, and IV in Fig. 2 (b). From a geometric point of view, the absolute operation is eliminated by using $-d(x_1, p_0)$ and $+d(x_1, p_0)$ for a pair of $\mathbf{u}_t$ and $\mathbf{u}_1$ when $\mathbf{u}_t$ is in region I and IV, respectively. Similarly, $-d(x_1, p_1)$ and $+d(x_1, p_1)$ for a pair of $\mathbf{u}_t$ and $\mathbf{u}_1$ when $\mathbf{u}_t$ is in region II and III, respectively. In a naïve approach, we can make the distance lower bound matrix in Fig. 2 (c) by applying this procedure to all the objects, and calculate an objective function value by counting the frequencies of each distance in the entries.

Instead, we calculate inner-products of $\mathbf{u}_s$ and each of $\mathbf{e}_+$ and $\mathbf{e}_-$, and give to $\mathbf{u}_s$ two distinct rank orders of $\rho(\mathbf{u}_s; \mathbf{e}_+)$ and $\rho(\mathbf{u}_s; \mathbf{e}_-)$ in descending order of $\mathbf{u}_s \cdot \mathbf{e}_+$ and $\mathbf{u}_s \cdot \mathbf{e}_-$, respectively. For instance, $\rho(\mathbf{u}_1, \mathbf{e}_+) = 3$ and $\rho(\mathbf{u}_1, \mathbf{e}_-) = 2$. $\rho(\mathbf{u}_1, \mathbf{e}_+) - 1$ means the number of objects in region I and II while $\rho(\mathbf{u}_1, \mathbf{e}_-) - 1$ means that in region I and III. Note that we can obtain the object sets in those regions as well as the number of the objects because we give the rank order to each object. By set operations, we identify the number of the objects contained in each region and calculate the objective function value.

Next, we explain the general case with reference to the example. Similar to the case of the 1-dimensional pivot space based on $\mathcal{A}(x_s)$, we define $\mathcal{B}(x_s)$ and $C(x_s)$ as follows:

$$\mathcal{B}(x_s) = \{x_t \mid \mathbf{u}_s \cdot \mathbf{e}_+ < \mathbf{u}_t \cdot \mathbf{e}_+\}, \quad (13)$$

$$C(x_s) = \{x_t \mid \mathbf{u}_s \cdot \mathbf{e}_- < \mathbf{u}_t \cdot \mathbf{e}_-\}, \quad (14)$$

where x_s and x_t denote the same objects as $\mathbf{u}_s$ and $\mathbf{u}_t$. Note that the conditions of x_t contained in $\mathcal{B}(x_s)$ and $C(x_s)$ are also represented in the same order as follows.

$$d(x_s, p_{j0}) + d(x_s, p_{j1}) < d(x_t, p_{j0}) + d(x_t, p_{j1})$$
$$d(x_s, p_{j0}) - d(x_s, p_{j1}) < d(x_t, p_{j0}) - d(x_t, p_{j1})$$

Furthermore, $|\mathcal{B}(x_s)|$ and $|C(x_s)|$ are represented by using the rank orders $\rho(\mathbf{u}_s, \mathbf{e}_+)$ and $\rho(\mathbf{u}_s, \mathbf{e}_-)$ in descending order of the inner-products of $\mathbf{u}_s \cdot \mathbf{e}_+$ and $\mathbf{u}_s \cdot \mathbf{e}_-$ as follows.

$$|\mathcal{B}(x_s)| = \rho(\mathbf{u}_s, \mathbf{e}_+) - 1 = \rho_+(x_s) - 1 \quad (15)$$

$$|C(x_s)| = \rho(\mathbf{u}_s, \mathbf{e}_-) - 1 = \rho_-(x_s) - 1, \quad (16)$$

where $\rho_+(x_s)$ and $\rho_-(x_s)$ denote $\rho(\mathbf{u}_s, \mathbf{e}_+)$ and $\rho(\mathbf{u}_s, \mathbf{e}_-)$, respectively. $\rho_+(x_s)$ and $\rho_-(x_s)$ are introduced so as to save us from considering the pivot space. Actually, $\rho_+(x_s)$ and $\rho_-(x_s)$ are obtained by sorting $x_s \in \mathcal{X}_j$ in descending order of $d(x_s, p_{j0}) + d(x_s, p_{j1})$ and $d(x_s, p_{j0}) - d(x_s, p_{j1})$, respectively. Using $\mathcal{B}(x_s)$ and $C(x_s)$, we can eliminate the absolute operation as follows:

$$\max_{p \in \{p_{j0}, p_{j1}\}} |d(x_t, p) - d(x_s, p)| = \begin{cases} +d(x_t, p_{j0}) - d(x_s, p_{j0}) & x_t \in \mathcal{B}(x_s) \cap \overline{C(x_s)} \quad (\text{I}), \\ +d(x_t, p_{j1}) - d(x_s, p_{j1}) & x_t \in \mathcal{B}(x_s) \cap \overline{C(x_s)} \quad (\text{II}), \\ -d(x_t, p_{j1}) + d(x_s, p_{j1}) & x_t \in \overline{\mathcal{B}(x_s)} \cap C(x_s) \quad (\text{III}), \\ -d(x_t, p_{j0}) + d(x_s, p_{j0}) & x_t \in \overline{\mathcal{B}(x_s) \cup C(x_s)} \quad (\text{IV}). \end{cases}$$

In the calculation of $F(\{p_{j0}, p_{j1}\}; \mathcal{X}_j)$, $d(x_s, p_{j0})$ appears as subtractions $|\mathcal{B}(x_s) \cap C(x_s)|$ times and as additions $|\mathcal{X}_j| - 1 - |\mathcal{B}(x_s) \cup C(x_s)|$ times while $d(x_s, p_{j1})$ appears as subtractions $|\mathcal{B}(x_s) \cap \overline{C(x_s)}|$ times and as additions $|\overline{\mathcal{B}(x_s)} \cap C(x_s)|$ times. From this fact, the objective function $F(p_{j0}, p_{j1}; \mathcal{X}_j)$ is expressed as follows.

$$F(p_{j0}, p_{j1}; \mathcal{X}_j) = \frac{1}{2} \sum_{x_s \in \mathcal{X}_j} \{\alpha_0 \cdot d(x_s, p_{j0}) + \alpha_1 \cdot d(x_s, p_{j1})\} \quad (17)$$

$$\alpha_0 = \left(|\mathcal{X}_j| - 1 - |\mathcal{B}(x_s) \cup C(x_s)|\right) - |\mathcal{B}(x_s) \cap C(x_s)|$$

$$\alpha_1 = \left|\overline{\mathcal{B}(x_s)} \cap C(x_s)\right| - |\mathcal{B}(x_s) \cap \overline{C(x_s)}|$$

To simplify α_0 and α_1, we use the following relationship.

$$|\mathcal{B}(x_s) \cup C(x_s)| + |\mathcal{B}(x_s) \cap C(x_s)|$$
$$= |\mathcal{B}(x_s)| + |C(x_s)| = \rho_+(x_s) + \rho_-(x_s) - 2 \quad (18)$$

$$\left|\overline{\mathcal{B}(x_s)} \cap C(x_s)\right| - \left|\mathcal{B}(x_s) \cap \overline{C(x_s)}\right|$$
$$= |\mathcal{B}(x_s)| - |C(x_s)| = \rho_+(x_s) - \rho_-(x_s) \quad (19)$$

Based on Eqs. (18) and (19), the objective function in Eq. (17) is rewritten as

$$F(p_{j0}, p_{j1}; \mathcal{X}_j) = \frac{1}{2} \sum_{x_s \in \mathcal{X}_j} \{\alpha_0 \cdot d(x_s, p_{j0}) + \alpha_1 \cdot d(x_s, p_{j1})\} \quad (20)$$

$$\alpha_0 = |\mathcal{X}_j| + 1 - \rho_+(x_s) - \rho_-(x_s)$$

$$\alpha_1 = \rho_+(x_s) - \rho_-(x_s).$$

Thus, we can calculate the objective function in Eq. (6) or Eq. (20) with the computational complexity of $O(N \log N)$ by sorting the objects $\mathbf{u}_s$ in the 2-dimensional pivot space with respect to the inner-products of $(\mathbf{u}_s \cdot \mathbf{e}_+)$ and $(\mathbf{u}_s \cdot \mathbf{e}_-)$. We summarize our pivot generation algorithm in Algorithm 2.

Algorithm 2 Pivot generation

```
1: Input: $\mathcal{X}_j$
2: Output: $\{p_{j0}, p_{j1}\}$
3: Initialize $\{p_{j0}, p_{j1}\}$ with two objects randomly chosen from $\mathcal{X}_j$
4: Calculate $d(x_s, p_{j0})$ and $d(x_s, p_{j1})$ for all $x_s \in \mathcal{X}_j$
5: Sort $x_s$ in descending order of $d(x_s, p_{j0}) + d(x_s, p_{j1})$
       and give the rank order $\rho_+(x_s)$ to $x_s$
6: Sort $x_s$ in descending order of $d(x_s, p_{j0}) - d(x_s, p_{j1})$
       and give the rank order $\rho_-(x_s)$ to $x_s$
7: Calculate objective function $F(\{p_{j0}, p_{j1}\}, \mathcal{X}_j)$
       using $\rho_+(x_s)$ and $\rho_-(x_s)$ in Eq. (20)
8: Update pivots $\{p_{j0}, p_{j1}\}$ so as to maximize the objective function
       in the same manner in [15], [16]
9: return $\{p_{j0}, p_{j1}\}$
```

4.2 Data Partitioning Algorithm

We show our data partitioning algorithm in Algorithm 3. The algorithm produces a pair of subsets $\mathcal{X}_{2j}$ and $\mathcal{X}_{(2j+1)}$ by alternately assigning objects in $\mathcal{X}_j$ in ascending order of distance from p_{j0} and p_{j1} at lines 5–11 after the initialization at line 3. At line 5, it is judged whether the order of the object assignment is even or odd. If the assignment order is even, the object $\hat{x}_s$ in a tentative object set $\mathcal{Y}_j$ closest to p_{j0} is added to $\mathcal{X}_{2j}$ as shown at lines 6 and 7. Otherwise, the object in $\mathcal{Y}_j$ closest to p_{j1} is added to $\mathcal{X}_{(2j+1)}$ at lines 9 and 10. The object $\hat{x}_s$ is removed from $\mathcal{Y}_j$ as $\mathcal{Y}_j \setminus \{\hat{x}_s\}$. This assignment makes the numbers of objects in $\mathcal{X}_{2j}$ and $\mathcal{X}_{(2j+1)}$ almost equal. The data partitioning algorithm constructs a CBT by repeating the foregoing procedure until $\mathcal{Y}_j = \emptyset$, i.e., while $k < |\mathcal{X}_j|$ at line 4.

In terms of the computational complexity, sorting the objects according to each of $d(x_s, p_{j0})$ and $d(x_s, p_{j1})$ at each i level is dominant. Then the computational complexity is $O(N \log N)$ at each i level.

Now, we discuss the computational complexity at Step **M2** of PGM-CBT, which generates pivots at Step **M2-1** and partitions a set of objects at Step **M2-2** for each

Algorithm 3 Data partitioning

```
1: Input: $\mathcal{X}_j$, $\{p_{j0}, p_{j1}\}$, $d(x_s, p_{j0})$ and $d(x_s, p_{j1})$ for $x_s \in \mathcal{X}_j$
2: Output: $\mathcal{X}_{2j}$, $\mathcal{X}_{(2j+1)}$
3: Initialize with $\mathcal{X}_{2j} \leftarrow \emptyset$, $\mathcal{X}_{(2j+1)} \leftarrow \emptyset$, $\mathcal{Y}_j \leftarrow \mathcal{X}_j$, $k \leftarrow 0$.
4: while $k < |\mathcal{X}_j|$ do
5:     if $0 \equiv k \pmod 2$ then
6:         $\hat{x}_s \leftarrow \arg\min_{x_s \in \mathcal{Y}_j} \{d(x_s, p_{j0})\}$
7:         $\mathcal{X}_{2j} \leftarrow \mathcal{X}_{2j} \cup \{\hat{x}_s\}$
8:     else
9:         $\hat{x}_s \leftarrow \arg\min_{x_s \in \mathcal{Y}_j} \{d(x_s, p_{j1})\}$
10:        $\mathcal{X}_{(2j+1)} \leftarrow \mathcal{X}_{(2j+1)} \cup \{\hat{x}_s\}$
11:    end if
12:    $\mathcal{Y}_j \leftarrow \mathcal{Y}_j \setminus \{\hat{x}_s\}$
13:    $k \leftarrow k+1$
14: end while
15: return $\mathcal{X}_{2j}$, $\mathcal{X}_{(2j+1)}$
```

leaf node. Note that at the level i of the CBT, the number of nodes is 2^{i-1}, and the number of assigned objects at each node is $|\mathcal{X}_j| = \lfloor |\mathcal{X}| / 2^{i-1} \rfloor$ or $|\mathcal{X}_j| = \lceil |\mathcal{X}| / 2^{i-1} \rceil$. Since $2^{i-1}|\mathcal{X}_j| \sim |\mathcal{X}| = N$, the total computational complexity at Steps **G1-1** and **G1-3** is $O(TNH)$, and the complexity at Step **G1-2** is bounded by $O(TN \log N)$ shown in Sect. 4.1. In contrast, the computational complexity of the data partitioning algorithm is $O(N \log N)$ at each level of a CBT. By using these, we can obtain the computational complexity of the PGM-CBT algorithm as follows:

$$O(\, L\,T\,N\,(H + \log N)\,)\,. \tag{21}$$

Thus, the PGM-CBT algorithm has the reasonably good scalable properties that the computational complexity to N is quasi-linear while those to L and H are linear.

4.3 Discussion on Computational Complexity

We discuss the computational complexity of our proposed algorithm in comparison to those of the existing similarity search algorithms based on pivots, in the case that each object is expressed in an H-dimensional vector space. Since every $p \in \mathcal{X} \setminus \mathcal{P}$ is evaluated, the total complexity for computing the right-hand-side of Eq. (5) becomes $O(HN^3)$. Then the number of pivots, which is denoted as $M = |\mathcal{P}|$, is assumed to be much smaller than the number of objects, i.e., $M \ll N$. Thus, since this selection is repeated M times, the computational complexity of this straightforward algorithm is $O(MHN^3)$. On the other hand, the complexity for computing the Eq. (6) becomes $O(MHN^2)$. Thus, the computational complexity of the existing algorithms in [15], [16] is $O(MTHN^2)$ because these algorithms do not perform any data partitioning, where T denotes the number of iterations for obtaining the generalized pivots.

In contrast, the computational complexity of our proposed algorithm is $O(\, LTN(H + \log N)\,)$, where the level L of the CBT is expressed by

$$L = \log_2(M + 2) - 1.$$

This indicates that our proposed algorithm has a good scalable property with respect to both the numbers of objects and pivots, i.e., N and M. Here, this acceleration is brought by using the two techniques: one for hierarchically partitioning a data set into two subsets with almost equal sizes, and the other for efficiently calculating the objective function in the case of only two pivots. Finally, we should emphasize that it is quite difficult for existing pivot based algorithms to conduct experiments using large-scale data sets like one containing a million of objects.

5. Experiments

We show the experimental data sets followed by our results on both speed (or a computational cost) of the pivot generation and effectiveness of the generated pivots, which is evaluated with similarity search performance. In our experiments, all algorithms for pivot generation (or selection) and similarity search were executed on a computer system equipped with two Xeon E5-2697v2 2.7GHz CPUs and a 256GB main memory with a single thread within the memory capacity.

5.1 Data Sets

As a given data set for our experiments, the CoPhIR (Content-based Photo Image Retrieval) test-collection [4], [5] was employed. The CoPhIR contains the following five MPEG-7 global descriptors extracted from each of more than 100 million images: scalable color (SC for short), color structure (CS), color layout (CL), edge histogram (EH), and homogeneous texture (HT). Each of the descriptors is represented as a vector, which corresponds to a point in a Euclidean space, and a distinct distance measure for each descriptor is defined by the MPEG-7 group [19]. The numbers of the elements of descriptors, each of which represents the dimensionality; SC, CS, CL, EH, and HT are 64, 64, 12, 80, and 62, respectively.

We, in particular, utilized the four descriptors of SC, CS, EH and, HT descriptors. In terms of EH, instead of the original 80-dimensional vector, we used a 150-dimensional vector that includes elements related to global and semi-global edge histograms as described in [21]. For each descriptor, we prepared the five data sets with the different sizes (the number of the data vectors), which were randomly chosen from the given data set, 1×10^4 (referred to as 10K-size), 5×10^4 (50K-size), 1×10^5 (100K-size), 5×10^5 (500K-size), and 1×10^6 (1M-size). As the test set for the similarity search performance evaluation, we chose 1×10^3 query objects randomly in the same manner as the data sets so that the chosen query objects were not included in the data sets.

5.2 Speed of Pivot Generation

We compared our proposed algorithm PGM-CBT with two variants that were designed to highlight the effects of the two main parts of PGM-CBT, that is, the pivot generation and the data partitioning for constructing the complete binary tree (CBT) based on the rank order. One of the two variants is an algorithm of which part corresponding to our pivot generation is the pivot selection referred to as PSM (Pivot Selection by Maximizing the objective function) in Sect. 3, where a pivot is a valid data object. The remaining parts of this variant, which contain the data partitioning, are the same as our CBT. The other variant is an algorithm of which part of the data partitioning divides a set of objects into two subsets based on a distance from two pivots instead of the rank order. Namely, from a given pair of pivots $\{p_{j0}, p_{j1}\}$, a set of objects $\mathcal{X}_j$ is partitioned into a pair of subsets $\mathcal{X}_{2j}$ and $\mathcal{X}_{(2j+1)}$ as

$$\mathcal{X}_{2j} = \{x_s \in \mathcal{X}_j \,|\, d(x_s, p_{j0}) \le d(x_s, p_{j1})\},$$
$$\mathcal{X}_{(2j+1)} = \mathcal{X}_j \setminus \mathcal{X}_{2j} \,.$$

This partitioning algorithm makes two subsets with markedly different sizes if there is a large difference in the object densities around the pivots. Here, since it is not so effective to partition nodes containing small numbers of objects for the purpose of increasing the number of the discarded objects $|J(\mathcal{X}, z, r; \mathcal{P})|$, this algorithm first selects the node containing the largest number of objects from leaf nodes, and then successively produces an unbalanced tree by partitioning the selected node. Thus, this selection and partition strategy is likely to produce an unbalanced tree with a higher level. We refer to the former and the latter as PSM-CBT and PGM-DBT (DBT: Distance-based Binary Tree), respectively.

We evaluated the performance of the algorithms by the elapsed time, varying the number of pivots and the size of the data sets. In each condition of the number of pivots and the data set sizes, the five sets of pivots were generated by each of PGM-CBT and PGM-DBT with distinct initial pivots (data objects) under the convergence criterion for optimization in Sect. 4.1. Note that the number of pivots $|\mathcal{P}|$ directly relates to the level L of the CBT in PGM-CBT as $|\mathcal{P}| = \sum_{i=1}^{L} 2^{i-1} \times 2 = 2^{L+1} - 2$. We used the 10 CBT levels of $1, 2, \cdots, 10$. In contrast, we made the five sets of pivots for PSM-CBT by repeating the following procedure by five times. At each node of the CBT, we randomly chose 100 pairs of objects

$$\{p_{j0}^{[i]}, p_{j1}^{[i]}\} \;\; (i = 1, 2, \cdots, 100)$$

from the objects in the node $\mathcal{X}_j$ as pivot candidates. Note that in our preliminary experiments, the search performances of the results obtained by using more than 100 pairs of objects were almost comparable to those using 100 pairs, as also reported in previous work [7]. We next selected as the pivots the object pair $\{\hat{p}_{j0}, \hat{p}_{j1}\}$ expressed by

$$\{\hat{p}_{j0}, \hat{p}_{j1}\} = \underset{\{p_{j0}^{[i]}, p_{j1}^{[i]}\}}{\arg\max}\, F(\{p_{j0}^{[i]}, p_{j1}^{[i]}\}; \mathcal{X}_j) \,.$$

The average elapsed time was calculated through the five trials of the pivot-set generation.

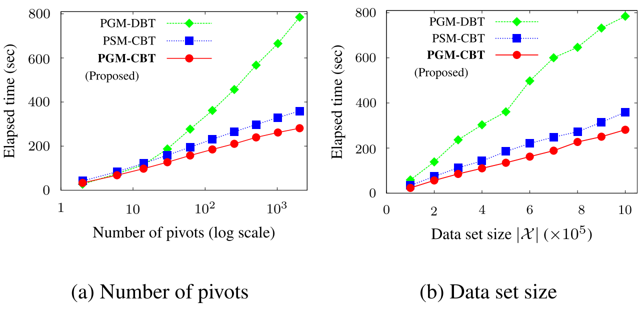


Fig. 3 Elapsed time required by the proposed PGM-CBT and the variants PSM-CBT and PGM-DBT. (a) With the number of pivots (depicted on the logarithmic scale) in the 1M-size SC data set, and (b) With size of the SC data sets when the 2046 pivots for the 10-level CBT were generated.

Figures 3 (a) and (b) show the average elapsed time required by the proposed PGM-CBT and the variants PSM-CBT and PGM-DBT. Although we show only the SC results, we observed the similar tendencies as those in Figs. 3 (a) and (b), in the other settings such as the other descriptors and the smaller data sets. Figure 3 (a) shows that PGM-CBT achieved the best performance among the algorithms when they made the identical number of pivots in the 1M-size SC data set. In particular, when the 2046 pivots corresponding to the 10 level of the CBT were generated, PGM-CBT operated almost 1.3 times faster than PST-CBT. By contrast, PGM-DBT spent more elapsed time since it required a lot of distance calculations between the pivots and many data objects contained at the nodes in the constructed unbalanced binary tree. Figure 3 (b) shows the scalability of the algorithms with data set size under the condition that the 2046 pivots were generated or selected, which correspond to the number of pivots used in the 10-level CBT. The average elapsed time of PGM-CBT is almost linear with respect to the data set size of the range from 10K-size to 1M-size. Finally, as mentioned earlier, the numbers of iterations until convergence required by the pivot generation algorithm described in Sect. 4.1 were much less than 100. For instance, in the cases of generating the first pairs of pivots for the 1M-size data sets, the average numbers of iterations over 5 trials were 12.0, 18.2, 14.6, and 17.6 seconds for CS, EH, HT, and SC in this order.

Thus the proposed PGM-CBT generated the identical number of the pivots faster than the others, and it was applicable to a large-scale data set due to its better scalability regarding the data size.

5.3 Effectiveness of Generated Pivots

Our proposed PGM-CBT and its variants of PSM-CBT and PGM-DBT generate or select pivots that maximize the objective function in Eq. (6), that is, maximize the sum of the lower bounds on distances of all pairs of objects in a given data set. We evaluated the practical effectiveness of the obtained pivots with similarity search performance. For comparison, we prepared other two variants designed focusing on the usage of pivots related to the data structure like our

CBT. One keeps our pivot generation algorithm applied to an object set in a current node, but divides a set into two subsets uniformly at random without using the relationship between an object and pivots, such that the difference of the size of the divided subsets is at most one. This variant uses the pivots only for a search stage. We refer the variant to as PGM-RCBT (RCBT: Random CBT). The other variant was introduced as a baseline algorithm without any data structure like trees. It employs a specified number of pivots chosen from objects in a given data set $\mathcal{X}$ uniformly at random. We refer this variant to as RPS (Random Pivot Selection algorithm). The number of the pivots was set at $(2^{L+1}-2)$ as many as that of PGM-CBT where $L = 1, 2, \cdots, 10$.

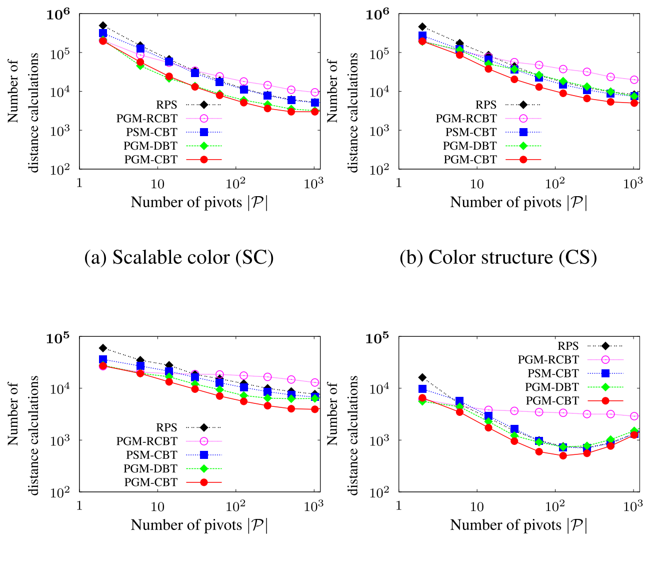
We evaluated the similarity search performance by the average number of required distance calculations $\overline{Cal}(r)$, which is expressed by

$$\overline{Cal}(r) = |\mathcal{X}| - \frac{1}{|\mathcal{Z}|}\sum_{z\in\mathcal{Z}} |\, J(\mathcal{X}, z, r; \mathcal{P})\,| + |\mathcal{P}|\,, \qquad (22)$$

where $\mathcal{Z}$ denotes a set of the query objects, r the range distance given for the range query problem. Note that the second and third terms in Eq. (22) mean the number of the unnecessary distance calculations between the query and the objects and the number of the distance calculations between the query and the pivots, respectively. We performed the exact similarity search in a *pivot space* as shown in [10], [17]. Owing to this scheme, we need no data structure for the evaluation. An object in the original space is mapped into a *pivot space*, i.e., a $|\mathcal{P}|$-dimensional Euclidean space as a vector each of whose elements is a distance from a pivot. In the *pivot space*, the discarded objects exist outside the hypercube of which center is the query and side length is $2r$, that is, $D(x, z; \mathcal{P}) > r$ in the original space. The range distance r was set at the average distance to the 10-th closest object to each of the 100 query objects z, i.e., $|\mathcal{Z}| = 100$. We determined the range distance r by our preliminary experiments, such as 950, 460, 260, and 190 for CS, HT, EH, and SC, respectively. In our experiments with the different r we also observed the similar characteristics to those in Fig. 4.

Figures 4 (a)–(d) show that the proposed PGM-CBT achieved the best performance for all the descriptors with different parameter settings. Note that the figures are illustrated in the logarithmic scale. For any data size, the number of the distance calculations of PGM-CBT had the minimum at a certain number of $|\mathcal{P}|$ around a range between 2^7 and 2^{10}, where recall the level L of the CBT is expressed by $L = \log_2(|\mathcal{P}| + 2) - 1$. Figure 4 (d) demonstrates this fact. This is due to the balance of the effects of the second and the third term in Eq. (22).

We shed light on our proposed PGM-CBT by comparison with the others. PGM-DBT provided the comparable performance, in particular, for SC descriptors unlike those for the other descriptors. This relates to the level of the leaf nodes. In case that $L = 10$ and $N = 10^6$, the average levels of PGM-DBT for CS, HT, EH, and **SC** were 101.96, 226.26, 468.00, and **40.30** while the level of PGM-CBT was



(a) Scalable color (SC)

(b) Color structure (CS)

(c) Edges histogram (EH)

(d) Homogeneous texture (HT)

Fig. 4 Number of distance calculations with the number of pivots on the logarithmic-logarithmic scale for the following descriptors: (a) Scalable color (SC), (b) Color structure (CS), (c) Edge histogram (EH), and (d) Homogeneous texture (HT). The results on the 1M-size data sets are shown for SC and CS while those on the 100K-size data sets for EH and HT.

exactly 10. These results indicate that the pivot generation and the data partitioning algorithm that lead to a more balanced binary tree substantially contributes to improve the search performance. Compared with the baseline algorithm RPS, PSM-CBT showed the better performance in the small $|\mathcal{P}|$ region for all the descriptors, but with the increase in $|\mathcal{P}|$, PSM-CBT lost the competitive edge. The performance difference of PGM-CBT and PSM-CBT to the basis of RPS suggests that the pivot generation rather than the selection plays the important role to achieve the better search performance. PGM-RCBT showed the quite poor performance especially in the large $|\mathcal{P}|$ region. Thus combining the pivot generation with the CBT is essential to high-performance exact similarity search.

6. Conclusion

We proposed the pivot-set generation algorithm called as PGM-CBT to improve exact similarity search performance for large-scale data sets. Compared with the other algorithms including variants of PGM-CBT and the baseline algorithm, PGM-CBT outperformed them in terms of both the speed of pivot generation and the effectiveness of generated pivots evaluated with similarity search performance in our experiments with the large-scale real image data sets. PGM-CBT efficiently generated a set of pivots with newly introduced techniques of the data partitioning with a complete binary tree (CBT) and the fast pivot generation algorithm that maximizes the specified objective function (PGM). The generated pivot set contributed to the acceleration of the exact similarity search by providing the tight lower bounds on distances between a query object and the data objects. Owing to the lower bounds, the exact similarity search algorithm effectively avoided unnecessary distance calculations.

Acknowledgments

This work was partly supported by JSPS Grant-in-Aid for Scientific Research (C) (No. 17K00159).

References

[1] G. Amato, A. Esuli, and F. Falchi, "A comparison of pivot selection techniques for permutation-based indexing," Information Systems, vol.52, pp.176–188, Aug.–Sept. 2015.

[2] A. Andoni, M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni, Nearest-neighbor methods in learning and vision: theory and practice, chapter 3: Locality-sensitive hashing using stable distributions, The MIT Press, Cambridge, Massachusetts, 2006.

[3] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A.Y. Wu, "An optimal algorithm for approximate nearest neighbor searching in fixed dimensions," Journal of the ACM, vol.45, no.6, pp.891–923, Nov. 1998.

[4] M. Batko, P. Kohoutkova, and D. Novak, "CoPhIR image collection under the microscope," In Proc. Int. Conf. Similarity Search and Applications, pp.47–54, IEEE, 2009.

[5] P. Bolettieri, A. Esuli, F. Falchi, C. Lucchese, R. Perego, T. Piccioli, and F. Rabitti, CoPhIR: a test collection for content-based image retrieval, CoRR, abs/0905.4627v2, 2009.

[6] S. Brin, Near neighbor search in large metric spaces, In Proc. Int. Conf. Very Large Data Bases, pp.574–584, Morgan Kaufmann, Sept. 1995.

[7] B. Bustos, G. Navarro, and E. Chávez, "Pivot selection techniques for proximity searching in metric spaces," Pattern Recognition Letters, vol.24, no.14, pp.2357–2366, Oct. 2003.

[8] M.S. Charikar, Similarity estimation techniques from rounding algorithms, In Proc. Annual ACM Symp. Theory of Computing, pp.380–388, New York, NY, ACM, 2002.

[9] E. Chávez and G. Navarro, "A compact space decomposition for effective metric indexing," Pattern Recognition Letters, vol.26, no.9, pp.1363–1376, July 2005.

[10] E. Chávez, G. Navarro, R. Baeza-Yates, and J.L. Marroquín, "Searching in metric spaces," ACM Computing Surveys, vol.33, no.3, pp.273–321, Sept. 2001.

[11] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An efficient access method for similarity search in metric spaces," In Proc. Int. Conf. Very Large Data Bases, pp.426–435. Morgan Kaufmann, Aug. 1997.

[12] K.L. Clarkson, "Building triangulations using ϵ-nets," In Proc. Annual ACM Sympo. Theory of Computing, pp.326–335, ACM, 2006.

[13] T.F. Gonzalez, "Clustering to minimize the maximum intercluster distance," Theoretical Computer Science, vol.38, pp.293–306, 1985.

[14] P. Indyk, Handbook of Discrete and Computational Geometry (second edition), chapter 39: Nearest neighbors in high-dimensional spaces, Chapman and Hall/CRC, 2004.

[15] M. Kimura, K. Saito, and N. Ueda, Pivot learning for efficient similarity search, In Proc. Int. Conf. Knowledge-Based and Intelligent Information & Engineering Systems, pp.227–234, Springer-Verlag, 2007.

[16] E. Kobayashi, T. Fushimi, K. Saito, and T. Ikeda, "Similarity search by generating pivots based on Manhattan distance," In Proc. Pacific Rim Int. Conf. Artificial Intelligence, vol.8862, pp.435–446, Springer International Publishing, 2014.

[17] R. Mao, W.L. Miranker, and D.P. Miranker, "Pivot selection: Dimension reduction for distance-based indexingm," Journal of Discrete Algorithms, vol.13, pp.32–46, May 2012.

[18] M.L. Micó, J. Oncina, and E. Vidal, "A new version of the nearest-neighbour approximating and eliminating search algorithm (AESA) with linear preprocessing time and memory requirements," Pattern Recognition Letters, vol.15, no.1, pp.9–17, Jan. 1994.

[19] MPEG-7, Multimedia content description interface – Part 3: Visual, ISO/IEC 15938-3:2002, 2002.

[20] D. Novak and M. Batko, "Metric index: An efficient and scalable solution for similarity search," In Proc. Int. Conf. Similarity Search and Applications, pp.65–73, IEEE, 2009.

[21] D.K. Park, Y.S. Jeon, and C.S. Won, "Efficient use of local edge histogram descriptor," In Proc. ACM Workshops on Multimedia, pp.51–54, New York, NY, ACM, 2000.

[22] H. Samet, Foundations of multidimensional and metric data structures, Morgan Kaufmann Publishers, San Francisco, CA, 2006.

[23] J.K. Uhlmann, "Satisfying general proximity/similarity queries with metric trees," Information Processing Letters, vol.40, no.4, pp.175–179, Nov. 1991.

[24] P. Zezula, G. Amato, V. Dohnal, and M. Batko, Similarity search: The metric space approach, Springer, New York, NY, 2006.

Yuki Yamagishi received the Ph.D. degree in arts and sciences from University of Shizuoka in 2017. In 2017, he joined the University of Shizuoka. He is currently a postdoctoral researcher at the School of Management and Information. His current research interests are machine learning and data mining.

Kazuo Aoyama received the B.E. degree in applied physics from Waseda University in 1986 and the M.E. degree from Tokyo Institute of Technology in 1988. In 1988, he joined NTT Corporation. His research interests include device modeling, LSI design methodology, computer architecture, data structure and algorithms, and machine learning.

Kazumi Saito received the B.S. degree in mathematics from Keio University in 1985 and the Ph.D. degree in engineering from University of Tokyo in 1998. In 1985, he joined the NTT Electrical Communication Laboratories. In 2007, he joined the University of Shizuoka. He is currently a professor at the School of Management and Information. His current research interests are machine learning and statistical analysis of complex networks.

Tetsuo Ikeda received the Master of Computer Science from University of Tokyo in 1981, and the Doctor of Engineering from University of Tokyo in 2001. In 1981, he joined the NTT Electrical Communication Laboratories. In 2002, he joined the Iwate Prefectural University. In 2007, he joined the University of Shizuoka. He is currently a professor at the School of Management and Information. His current research interests are data engineering, information retrieval, and GIS.