PAPER Special Section on Enriched Multimedia

Nearest Neighbor Search with the Revised TLAESA

Dong WANG[†a)], ***Nonmember***, **Hiroyuki MITSUHARA**[†b)], ***Member***, ***and*** **Masami SHISHIBORI**[†c)], ***Nonmember***

SUMMARY It is significant to develop better search methods to handle the rapidly increasing volume of multimedia data. For NN (Nearest Neighbor) search in metric spaces, the TLAESA (Tree Linear Approximating and Eliminating Search Algorithm) is a state of art fast search method. In this paper a method is proposed to improve the TLAESA by revising the tree structure with an optimal number of selected global pivots in the higher levels as representatives and employing the best-first search strategy. Based on an improved version of the TLAESA that succeeds in using the best-first search strategy to greatly reduce the distance calculations, this method improves the drawback that calculating less at the price of the lower pruning rate of branches. The lower pruning rate further can lead to lower search efficiency, because the priority queue used in the adopted best-first search strategy stores the information of the visited but unpruned nodes, and need be frequently accessed and sorted. In order to enhance the pruning rate of branches, the improved method tries to make more selected global pivots locate in the higher levels of the search tree as representatives. As more real distances instead of lower bound estimations of the node-representatives are used for approximating the closet node and for "branch and bound", not only which nodes are close to the query object can be evaluated more effectively, but also the pruning rate of branches can be enhanced. Experiments show that for k-NN queries in Euclidean space, in a proper pivot selection strategy the proposed method can reach the same fewest distance calculations as the LAESA (Linear Approximating and Eliminating Search Algorithm) which saves more calculations than the TLAESA, and can achieve a higher search efficiency than the TLAESA.

key words: *TLAESA, Nearest Neighbor, global pivots, best-first*

1. Introduction

With the rapidly increasing volume of multimedia data, such as texts, images, videos, in the internet, it is urgent to design better access methods that can support retrievals like similarity queries which often refers to finding some nearest neighbors (NN) to a query object in the multimedia database. Especially, the cross-media retrieval and multimedia abstraction are included in the scope of EMM (Enriched Multimedia), which is the theme of this special section. On the cross-media retrieval system, the correlation between cross-media data must be calculated to be able to retrieve the cross-media, such as the retrieval between text and images. The similarity retrieval and indexing technique, which are proposed in this paper, are often used to calculate the correlation [1]. On the video abstraction and summarization systems, moreover, the content-based image and video retrieval must be used in order to extract key-frames and "highlight" sequences [2]. The similarity retrieval and indexing technique are very important phase on the content-based image and video retrieval. If more effective retrieval and indexing methods are applied, high-speed processing will be realized.

Generally it is efficient to index the extracted feature vectors for similarity queries to avoid a sequential scan. Macroscopically we can build the index by two methods, and the one is called the Spatial Access Method (SAM), while the other is called the Metric Access Method (MAM). They are both subject to the "curse of dimensionality" [3]. However, the MAMs can handle high dimensional data more efficiently, while the SAMs are efficient only to index low dimensional data [4]. Besides, the SAMs are not so flexible as the MAMs when the data is not drawn from a vector space but just some distance assignments between every pair of objects.

We just consider the more widely applied MAMs in metric spaces where the (dis-)similarities are measured by distances, and shorter distances mean higher similarities. A metric space is defined by its distance function $d : U \times U \rightarrow \Re$ which has to satisfy the following properties [5], where U is a universe of objects.

$\forall\ x, y, z \in U$,

(1) Non-negativity: $d(x, y) \geq 0$

(2) Reflexivity: $d(x, y) = 0 \Leftrightarrow x = y$

(3) Symmetry: $d(x, y) = d(y, x)$

(4) Triangular inequality: $d(x, y) + d(y, z) \geq d(x, z)$

The most important property is the "triangular inequality", as we can prune some objects by it to avoid a through linear search in the database. The Euclidean distance, the Manhattan distance, and the Edit distance which are widely used satisfy those properties, and the spaces defined by them are metric spaces. However, those properties are not so easy to satisfy. A distance function failing in satisfying any of the properties is called a non-metric distance function, such as [6] the Cosine Distance, the Dynamic Time Warping, etc.

The MAMs, or the metric space indexes, can be divided into two categories: the one is based on the pivot-technique, which mainly refers to the AESA (Approximating and Eliminating Search Algorithm) [7] family including the AESA, the Linear AESA (LAESA) [8] and the Tree LAESA (TLAESA) [9], and the other is based on the clustering-technique, and the space is broken up hierarchically and recursively, including almost

Manuscript received March 11, 2014.

Manuscript revised August 8, 2014.

[†]The authors are with the Dept. of Information Science and Intelligent System, the University of Tokushima, Tokushima-shi, 770–8506 Japan.

a) E-mail: wangdong.uni@gmail.com

b) E-mail: mituhara@is.tokushima-u.ac.jp

c) E-mail: bori@is.tokushima-u.ac.jp

DOI: 10.1587/transinf.2014MUP0002

all the metric index trees, such as the GH-Tree [10], the VP-Tree [11], the M-Tree [12], etc. One can say that a tree based metric index structure materializes a persistent representation of a hierarchical clustering of the data [13]. Generally, the pivot-technique is good at reducing calculations by triangular inequality, while the clustering-technique is good at pruning branches to accelerate the search. There are some metric indexes which intend to combine the two techniques. Actually any metric index tree can be extended to a pivot based index tree after being equipped with some pivots and a pre-calculated distance-matrix. The PM-Tree [14] can be a case in point. The TLAESA can also be viewed not strictly as this kind of data structure.

An improved version of the TLAESA (iTLAESA for short) [15] was proposed and succeeded in greatly reducing the distance calculations. However, it saves the calculations at the cost of the lower pruning rate of branches. Furthermore, in the adopted best-first search strategy the priority queue which stores the information of the visited but unpruned nodes need be frequently accessed and sorted due to the lower pruning rate. Consequently, the search efficiency is not so high. Thus, in this study an improved method is proposed in order to enhance the pruning rate. This method intends to make the selected global pivots in the higher levels of the search tree as possible. As more real distances instead of lower bound estimations of the node-representatives are used for "branch and bound", the pruning rate can be enhanced, and hence a higher search efficiency can be achieved.

The contents left are organized as follows. Section 2 is a review on the TLAESA with its general build and search process. In Sect. 3 the improved method iTLAESA and its existing problems are illustrated. Section 4 is on the improved schemes. Section 5 describes some adopted pivot selection strategies that can play a vital role in affecting the search efficiency. Section 6 displays some evaluation experiments, and finally Sect. 7 sums up the whole paper.

2. TLAESA

The TLAESA can be classified as a pivot-technique based metric search method. In order to avoid distance calculations by triangular inequality, the pivot-technique uses a subset of objects in the database as pivots and a distance matrix which stores the pre-calculated distances from all or a portion of the objects to each pivot.

2.1 TLAESA and Its Evolution

TLAESA belongs to the AESA family, and the AESA is being considered as the fastest NN search method in terms of distance computations so far in metric spaces [16]. As the algorithm name indicates, approximation and elimination are two key steps for the AESA, and the AESA employs the lower bound estimation of real distance both for approximating and eliminating. However, the AESA achieves excellent computational behavior at the expense of huge quadratic memory space requirements as it needs a huge distance matrix to store all the distances between any pair of objects.

There are some methods to improve the AESA by reducing the overhead or the preprocessing time to accelerate the search. The k-AESAU (k-AESA with Upper bounds) [17] taking advantage of the upper bound function was proposed to avoid the drawback that no active objects can ever be eliminated until exactly k distances have been calculated for k-NN queries, but it has no effect on reducing distance calculations. The ROAESA (Reduced Overhead AESA) [18] was designed to acquire a sublinear overhead by progressively selecting the candidates in a bound area, but it has the same distance calculations as the AESA. The iAESA [19] attempts to be faster by using pivot-permutation to modify the approximation, but to keep updating the permutations makes the overhead much higher instead. The PiAESA [20] intends to raise the lower bounds of the still alive objects by an optimal number of pivots before switching to the AESA usual behavior in order to discard them later more easily by triangular inequality, but it is not easy to determine the optimal number of pivots and probably the optimal number is the number of all the selected pivots for k-NN queries if k is much bigger.

The improved methods of the AESA developing in the direction of using less memory space are more prospective. The LAESA was earlier proposed to cut the space complexity down to linear bounds by selecting an optimal number of outliers as pivots other than taking any calculated objects as pivots. Although this algorithm increases the number of distance calculations, roughly as 1.5 times as those of the AESA [8], it is still very useful when the distance calculation is time-consuming. The TLAESA evolving directly from the LAESA can achieve sublinear overhead. It can be viewed as an MAM combing the pivot technique with the clustering technique, and the original consider is possibly to use a tree structure to prune some branches and filter out some leaf-objects before the LAESA process. Like the Monotonous Bisector-Tree [21] which is an improved variation of the binary GH-Tree, it reuses the representative of the parent node as the representative of the one child-node and selecting another new object as the representative of the other child-node. Obviously in this way, the distance calculations from the representatives to the query object can be avoided nearly half even if they are calculated. Unfortunately, in the depth-first search order it calculates more than the LAESA.

2.2 General Build and Search of TLAESA

The TLAESA builds two different data structures, the search tree and the distance matrix [15]. In the search tree, each leaf node contains a single object which is its representative and all the indexed objects finally act as the representatives in the leaf-nodes. The distance matrix stores the distances from every object to some selected outliers as pivots.

The search tree of the TLAESA is originally a binary tree, and it is built by splitting the space recursively. For any node containing more than one object, the TLAESA let its right child-node reuse the same representative, and selects the farthest object to that representative as the representative of the left child-node. For each remaining object, if it is closer to the representative of the right child-node than that of the left one, it is divided into the right child-node. Otherwise, it is divided into the left child-node.

The TLAESA traverses the search tree in the depth-first order which is a common traversing mechanism. Unlike the other conventional indexed trees, the TLAESA uses the lower bound estimation instead of the real distance from the node-representative to the query object for "branch and bound". As no real distance calculations take place in the non-leaf-nodes, all the distances, except the calculated distances from the pivots to the query object initially, are only calculated in the leaf-nodes. Compared with filtering out some objects in the leaf-nodes, the more important role that the selected pivots and the relative distance matrix play is evaluating the lower bound estimations of the representatives for further "branch and bound", as in any leaf-node at most one object namely the representative can be filtered out.

As for the lower bound estimation of the real distance, it is defined as

$$g_x = g(x, q) = \max_{p \in P} \{|d(x, p) - d(p, q)|\} \quad (1)$$

where x is an arbitrary indexed object, q is the query object, and P is a set of selected pivots. In addition, the distance $d(x, p)$ can be loaded from the built distance matrix, while $d(p, q)$ can be pre-calculated at the beginning of the search, so it is very easy to get g_x. If we want to get the lower bound estimation of the representative m_t of the node $t \in T$ (T is the search tree of the TLAESA), just let $x = m_t$ in Eq. (1).

We should notice that the original TLAESA does not follow the simplex depth-first strategy but combing root-left-right and root-right-left strategy [9]. In the depth-first strategy there is a question whether it is more efficient to first visit the left-child or it is more efficient to first visit the right-child. To answer that question, the TLAESA indicates that the child whose representative has a smaller lower bound estimation is first visited if it is not pruned.

3. iTLAESA

The iTLAESA, an improved version of the TLAESA, was proposed to mainly improve the search process to reduce the number of distance calculations.

3.1 Search Process

The iTLAESA converts the original depth-first search strategy to the best-first strategy, which is actually a width-first search strategy plus a heuristic rule. This rule is that the node which corresponds to a set should be traversed preferentially when it is the closest to the query object [15], as the closest node possibly contains the closest object. This can be viewed as an extension of the step approximation in the AESA from just approximating a point to approximating a node/set.

On how to define the approximate distance h_t from the node t to the query object q, the iTLAESA adopts

$$h_t = g_{m_t} - r_t \quad (2)$$

where r_t is the covering radius of the node t. Then the iTLAESA selects the node t with the minimum h_t to visit preferentially.

As the priority queue is a widely applied data structure based on heap for fast sorting, the iTLAESA uses it to store the information of any visited but unpruned node t that is expressed as a triple (t, g_{m_t}, r_t). In this study this priority queue is called the TPQ (Tree-node-information-Priority-Queue) for short.

We can say that the whole search process is transformed from the traverse of the search tree to the frequent accesses of the TPQ. If the TPQ is not empty, the element namely the triple (t, g_{m_t}, r_t) which can let h_t minimum is taken out so that the node to be visited next can be acquired. Furthermore, if the acquired node is not the leaf, for each of its unpruned child-node c, the triple (c, g_{m_c}, r_c) is added into the TPQ. Otherwise, no triples are added into the TPQ, and the distance from the representative which can not be pruned to the query need be calculated to update the nearest neighbors. Initially the triple of the root is added into the TPQ, and then the steps above are recursively repeated until the TPQ becomes empty.

Besides, the priority queue is also used to store the information of each candidate NN that is expressed as a pair (n_i, d_{n_i}), where n_i is the i-th candidate NN and d_{n_i} is its distance to the query object $d(n_i, q)$, for k-NN queries. In this study this priority queue is called the CPQ (Candidate-nearest-neighbor-information-Priority-Queue) for short. In addition, in this study the triples in the TPQ is automatically sorted in the decreasing order of h_t, and the top element of the TPQ can let h_t minimum; while the pairs in the CPQ is automatically sorted in the increasing order of d_{n_i} and the top element of the CPQ can provide the current search radius.

The search algorithm of the iTLAESA for k-NN queries with some modifications from the original iTLAESA is given in Fig. 1 plus Fig. 2. In the original iTLAESA, the lower bound of the representative of the child-node g_{m_c} is evaluated after the judgment

$$g_{m_t} < r_c + r_s \quad (3)$$

where c is the child-node of the non-leaf t, r_s is the dynamically decreased search radius. This may lead to the incorrect results. In this study, the judgment is modified to

$$g_{m_c} \leq r_c + r_s \quad (4)$$

and the easily calculated g_{m_c} is evaluated before the

Algorithm: k**-NN Query**

Input:
B: indexed database
$P \subset B$: set of pivots
T: search tree
$M \in \Re^{|B|\times|P|}$: distance matrix storing
distances from all indexed objects to pivots
q: query object
$k \in \mathbb{Z}^+$: number of NNs
Output:
N: CPQ
r_s: search radius

```
1: r_s = ∞
2: for each p ∈ P do
3:   ▷ first calculate the distances from the pivots to the query
4:   D[p] = d(p, q)
5:   if D[p] < r_s then
6:     N ← N ∪ (p, D[p])
7:     if |N| > k then
8:       remove top pair from N
9:     end if
10:    if |N| == k then
11:      (n_k, r_s) = top pair in N
12:    end if
13:  end if
14: end for
15: t = root of T
16: g_{m_t} = max_{p∈P}{|M[m_t, p] − D[p]|}
17: Q ← {(t, g_{m_t}, r_t)}                    ▷ Q is TPQ
18: while Q ≠ ∅ do
19:   (t, g_{m_t}, r_t) = top triple in Q
20:   remove top triple from Q
21:   SEARCH(t, g_{m_t}, q, k, r_s, D, M, P, Q, N)
22: end while
```

Fig. 1 k-NN query algorithm of the iTLAESA.

judgment.

3.2 Build Process

There are no problems that the iTLAESA adopts the search tree structure of the TLAESA built specially for the depth-first search strategy. However, for the best-first search it is unnecessary to build such a troublesome structure where the right child-node has to reuse the representative. Besides, in order to evaluate which nodes are close to the query more effectively, the iTLAESA uses one of the pivots (the first selected pivot) as the representative of the root-node, and apparently the lower bound estimation of the distance from the representative to the query actually becomes the real distance. For more details on building the search tree of the iTLAESA, you can refer to the original paper [15].

3.3 Existing Problems

Because of the best-first search strategy, the iTLAESA indeed can greatly reduce the distance calculations compared to the TLAESA. The experiments show that given the same pivots, the iTLAESA calculates almost the same number of distances as the LAESA. However, the pruning rate of branches defined as the ratio of the pruned

```
1: function SEARCH(t, g_{m_t}, q, k, r_s, D, M, P, Q, N)
2:   if t is leaf then
3:     if g_{m_t} ≤ r_s and m_t ∉ P then
4:                ▷ try to use the lower bound to filter out
5:       d = d(m_t, q)
6:       if d < r_s then
7:         N ← N ∪ (m_t, d)
8:         if |N| > k then
9:           remove top pair from N
10:        end if
11:        if |N| == k then
12:          (n_k, r_s) = top pair in N
13:        end if
14:      end if
15:    end if
16:  else
17:    for each child c of t do
18:      if m_c == m_t then  ▷ get the parent lower bound
19:        g_{m_c} = g_{m_t}
20:      else                          ▷ evaluate the lower bound
21:        g_{m_c} = max_{p∈P}{|M[m_c, p] − D[p]|}
22:      end if
23:      if g_{m_c} ≤ r_c + r_s then
24:                ▷ add the unpruned children into the TPQ
25:        Q ← Q ∪ {(c, g_{m_c}, r_c)}
26:      end if
27:    end for
28:  end if
29: end function
```

Fig. 2 Search a node of the iTLAESA.

branches to all the branches deteriorates by experiments. We can say that there exists a "trade-off" between the distance calculations and the pruning rate, and probably the iTLAESA saves the distances at the sacrifice of the pruning rate. To explain more specifically, probably the iTLAESA makes some distance calculations happening only in the leaf-nodes not be calculated in advance but be pruned with a shorter search radius later; however, the number of the unpruned branches could increase as the search radius could not be shrunk before those calculations.

The low pruning rate leads to the disadvantage that during the search the TPQ is frequently accessed as more triples need be added into the TPQ. Suppose that the elements in the TPQ need be sorted one time when one triple is added (the remove process should not be considered), and the times to add equal the times to sort. Although the priority queue is quite efficient, to sort its elements frequently is still time-consuming, particularly when its size is very large. The frequent sort eventually could make the search efficiency of the iTLAESA lower than that of the TLAESA when the distance function is not so complex, even if a large quantity of distance calculations can be reduced.

4. Proposed Method

It is desirable that more pivots are arranged as the representatives in the higher or upper level of the search tree, and then more real distances instead of lower bound estimations can be used to evaluate which nodes are

close to the query more effectively in the early stage [15]. Meanwhile, if more pivots are visited as the representatives of the nodes in the higher levels, the pruning rate of branches can be higher, as the process of "branch and bound" is more efficient to earlier use real distances than lower bound estimations. However, in the iTLAESA only one pivot namely the first selected pivot is used as the representative of the root, and the root as well as the successor nodes reusing the same representative is benefited. Therefore, we hope to improve the pruning rate of branches by making more pivots in the higher levels of the search tree as the representatives as possible.

4.1 Selecting Representatives as Pivots

One easy way is to directly select some representatives of the nodes in the higher levels getting rid of the repetitions as pivots. These selected pivots are called local node-pivots, compared to the global pivots like the outliers. Usually this method can affect the distance calculations. It is possible that by some selection strategies the selected global pivots are not so good as the node-pivots, and in this case the method selecting node-pivots can reduce the calculations given the same number of pivots. However, this method is invalid when we use a proper strategy to obtain a set of optimal or good global pivots. Of course, we can adjust the proportion of the node-pivots, and let the global pivots and the node-pivots work together to decrease the number of distance calculations. Then the method TLAESA_MP (Mixed Pivots) using a portion of global pivots and the rest portion of node-pivots comes into being. However, experiments showed that if a set of optimal or good global pivots are selected, given the same number of pivots, probably the more the global pivots take up, the fewer the distance calculations, and the higher the search efficiency. An extreme case is that the global pivots take up 100% and no node-pivots are used. Hence, the TLAESA_MP is not so worth to be recommended.

4.2 Revising Tree Structure

Another way is to change the building rule to reshape the search tree structure of the TLAESA. When we build the search tree after preparing a set of global pivots, the global pivot has the priority to be selected as the representative of the node if the node contains the remaining global pivots. In other words, if the join of the node-set and the set of the remaining global pivots is not empty, the object in the join is selected as the representative preferentially. Otherwise, the representative is selected according to the original building rule of the TLAESA.

In this way, the global pivots which could exist in lower levels can be risen artificially to the higher levels of the search tree as the representatives as possible. In consideration of reusing, eventually the levels in the first parent-nodes are risen. A case is depicted in Fig. 3, where $t_l \in T$ is a tree node, $p_k \in P \subset B$ is a global pivot, and

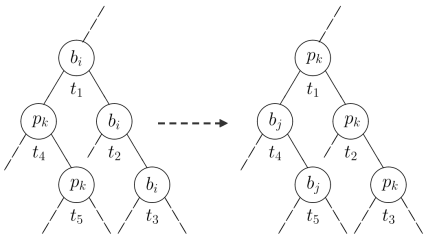

Fig. 3 Selecting the global pivot as the node-representative preferentially to let it into the higher level.

Algorithm: Build Search Tree

Input:
B: indexed database
$P \subset B$: set of selected pivots
$w \in \mathbb{Z} \cap [2, |B|)$: number of branches
Output:
$M \in \Re^{|B|\times|P|}$: distance matrix storing distances from all indexed objects to pivots
T: search tree

```
1: for each b ∈ B do                          ▷ first build the distance matrix
2:     for each p ∈ P do
3:         M[b, p] = d(b, p)
4:     end for
5: end for
6: let t be root
7: m_t = p ∈ P
8: P = P − {p}
9: S_t = B
10: T = ∅
11: BUILD(t, w, m_t, S_t, P, T)               ▷ build the search tree
```

Fig. 4 Build the search tree of the eTLAESA.

$b_i, b_j \in B - P$ are two indexed data (T is the search tree of the TLAESA, P is a set of selected pivots, and B is the indexed database), $i, j, k, l \in \mathbb{Z}^+$. Suppose that a global pivot has been selected for the representative of the parent-node of t_1, and the pivot p_k is the representative of the child-node of t_1, and then the level of p_k can be risen to that of t_1 if p_k is selected preferentially than b_i as the representative of t_1.

Experiments show that this is a simple but an effective method, and it can enhance the pruning rate of branches and greatly reduce the times to access and sort the TPQ. In this study this method is called the eTLAESA (enhanced TLAESA) that employs the same search process as the iTLAESA but improves the build process. Compared with the iTLAESA, the eTLAESA can not save more distance calculations but can keep the same number of calculations given the same pivots. In fact, given the same pivots the eTLAESA or the iTLAESA has reached the limited fewest distance calculations as the LAESA.

The building algorithm of the eTLAESA extended to a multi-way tree is given in Fig. 4 plus Fig. 5. The built structure of each node is (node-identity, number of branches, node-representative, covering-radius, pointer to children). As for the searching algorithm, it is the same as that of the iTLASEA (the modified edition above).

```
1: function BUILD(t, w, m_t, S_t, P, T)
2:     if |S_t| > 1 then
3:         t is not leaf
4:         c_p = (c_1, c_2, ..., c_w)          ▷ suppose t has w children
5:         i = 1
6:         m_{c_i} = m_t                       ▷ reuse the parent representative
7:         S_{c_i} = {m_{c_i}}
8:         S_t = S_t − {m_{c_i}}
9:         while i < w and S_t ≠ ∅ do
10:            i = i + 1
11:          ▷ select the representative in the join preferentially
12:            V = S_t ∩ P
13:            if V == ∅ then
14:                V = S_t
15:            end if
16:            m_{c_i} = arg max_{v∈V} Σ_{j=1}^{i−1} d(v, m_{c_j})
17:            S_{c_i} = {m_{c_i}}
18:            S_t = S_t − {m_{c_i}}
19:            P = P − {m_{c_i}}
20:        end while
21:        w = i                               ▷ renew the number of branches
22:        r_t = d(m_{c_1}, m_{c_2})
23:        if St ≠ ∅ then
24:    ▷ divide the remaining objects to the closest child-node
25:            for each s ∈ S_t do
26:                i = arg min_{1≤i≤w} d(s, m_{c_i})
27:                S_{c_i} = S_{c_i} ∪ {s}
28:            end for
29:        end if
30:        for i = 1 → w do ▷ recursively build each child-node
31:            BUILD(c_i, w, m_{c_i}, S_{c_i}, P, T)
32:        end for
33:    else
34:        t is leaf
35:        c_p = null                          ▷ leaf has no children
36:        w = 0
37:        r_t = 0
38:    end if
39: T ← T ∪ {(t, w, m_t, r_t, c_p)}
40: end function
```

Fig. 5 Recursively build each tree-node of the eTLAESA.

The TLAESA wants the representatives of the children to be far away from each other, and the global pivots, like the outliers which are a set of objects far away from each other and from the rest of the objects, don't violate this requirement even if they are selected as the representatives in the higher levels. In addition, if we select all the indexed data objects as the global pivots, the eTLAESA becomes the iTLAESA which can be called a degradation. However, we don't need too many pivots, because too many pivots could increase the distance calculations as the pivots themselves should be calculated with the query at first. Meanwhile, too many pivots could lead to the too large pre-calculated distance-matrix which is time-consuming to load and access. Therefore, a smaller number of pivots make the eTLAESA feasible. In fact, there exists an optimal number of pivots, but in this study the number of pivots to be selected is limited to 5% of the number of the whole indexed data.

As the iTLAESA, the eTLAESA adopts the best-first strategy for searching, and the node t with the minimum approximate distance h_t is first visited. Nevertheless, to evaluate h_t we can just use g_{m_t} irrespective of the covering radius r_t as

$$h_t = g_{m_t} \tag{5}$$

A heuristic factor θ of r_t can be inducted to unify the equations as

$$h_t = g_{m_t} - \theta \cdot r_t \tag{6}$$

where $0 \le \theta \le 1$. If $\theta = 0$, it denotes Eq. (5), and if $\theta = 1$, it denotes Eq. (2). Of course, θ can be other values like 0.5.

We can say that the best-first strategy can be specialized with the heuristic factor of the covering radius of the node. Experiments show that different values of heuristic factors can greatly affect the search performance. In this study, the best-first search strategy with the heuristic factor θ is defined as a strategy that uses Eq. (6) to evaluate the approximate distance of the node, and from the root-node first visits the node whose approximate distance is the closest to the query. Thus the original best-first search strategy used in the iTLAESA can be called the best-first search strategy with the heuristic factor 1.

5. Pivot Selection Strategies

The pivot-technique can be applied to fast search by filtering out objects without measuring their actual distance to the query to save distance calculations. However, the way that the pivots are selected can drastically affect the search performance of the algorithm. Better selected pivots can largely reduce the distance calculations, or given the same calculations a better selected set of pivots can require much less space for building the distance matrix than a random set.

Several main strategies for selecting pivots are described as follows. Of course, these strategies are used for selecting the global pivots not the local pivots. In this study these strategies are divided into two categories: selecting pivots based on the data distribution, and selecting pivots based on the evaluation criterion.

5.1 Selecting Based on Data Distribution

The classic pivot selection strategy in this category is the incremental outlier selection technique which is earliest applied in the LAESA. It selects objects as pivots located far away from the previously selected pivots incrementally. In other words, it let the selected pivots as separated as possible, because if the pivots are not so disperse, there exist overlapped pruning regions, and obviously two very close pivots give almost the same information for filtering out objects.

To implement the classic incremental outlier selection strategy, there are actually two ways to select the next pivot after a first random pivot p_1. The one can be called the Maximum of Sum of Distances (MSD) strategy [8], and the

other can be called the Maximum of Minimum Distances (MMD) strategy [22]. The next pivot is selected by the following equations:
MSD :

$$p_i = \arg\max_{s \in B - P_i} \sum_{j=1}^{i-1} d(s, p_j) \quad (7)$$

MMD :

$$p_i = \arg\max_{s \in B - P_i} \min_{j=1}^{i-1} d(s, p_j) \quad (8)$$

where B is the database to be indexed, and $P_i = \{p_1, p_2, \ldots, p_{i-1}\}$ is a set of selected pivots.

It is said that the MMD is more effective than the MSD [22], but in this study we found that it is not always so in different databases. By experiments, amazingly the MMD can not save more distance calculations than the MSD in the uniformly distributed databases.

5.2 Selecting Based on Evaluation Criterion

The strategy selecting pivots based on the data distribution actually selects the pivots from the aspect of avoiding overlapped pruning regions, and it let the selected pivots disperse. Generally good pivots are objects far away from each other and from the rest of the objects, but the objects far away from each other and from the rest of the objects are not always good pivots. We can imagine if all the pivots are the corner points, they are indeed very disperse, but we hardly discard any objects by triangular inequality as the search radius keeps very big before any other objects are calculated. Although there are improved strategies, such as the Sparse Spatial Selection (SSS) [23] and its dynamic extension Dynamic Pivot Selection (DPS) [24], which are proposed to dynamically select a set of pivots that are not very far away from each other neither very far from the rest objects, they can hardly obtain the fewest distance calculations compared to the MSD or the MMD [20].

To Select pivots directly from the aspect of triangular inequality could be more effective, as for fast search we need to filter out the objects as possible by triangular inequality. We should notice that the lower bound estimation deduced from triangular inequality is not bigger than the real distance, but we can select some better pivots to maximize the lower bound estimation to approximate the real distance. Basing on maximizing the lower bound estimations, a strategy selecting a set of "good pivots" [25] was proposed and performed better than outliers for similarity queries in the real-world databases by experiments. This strategy tries to select the objects which can maximize the average lower bound estimation of some randomly chosen data pairs as pivots. Since this strategy is based on an evaluation criterion in Eq. (9) involving the Average Lower Bound estimation, it is called ALB strategy in this study.

$$\mu_{P_i} = 1/A \sum_{j=1}^{A} \max_{p \in P_i} \{|d(a_j, p) - d(p, b_j)|\} \quad (9)$$

where a_j and b_j are from A pairs of randomly chosen data (a_j, b_j), and P_i is better than P'_i when $\mu_{P_i} > \mu_{P'_i}$.

There are also three ways to select pivots to implement this strategy [25]:
(1)Selection of K random groups;
(2)Incremental selection;
(3)Local optimum selection.
The incremental selection method described briefly as follows, is recommended as it is the most effective one.
A pivot p_1 is selected from a sample of K objects in the database B to be indexed, so that that pivot alone has the maximum μ_{P_1} value. Then, a second pivot p_2 is selected from another sample of K objects in B, so that p_1, p_2 has the maximum μ_{P_2} value, considering p_1 fixed. The third pivot p_3 is selected from another sample of K objects in B, so that p_1, p_2, p_3 has the maximum μ_{P_3} value, considering p_1 and p_2 fixed. The process is repeated until the required number of pivots is obtained.

It is better to have a high value of parameter A which denotes the number of evaluated data pairs, but it is not feasible to evaluate all the data pairs. Besides, a low value of parameter K is suggested. In this study, the value of parameter A is set to the number of the indexed data, and the parameter K is set to the suggested value 50.

6. Experiments

The experiments in this study are mainly to compare the proposed method eTLAESA with the state of art method TLAESA as well as with its improved method iTLAESA employing the best-first search strategy and the first selected global pivot as the root-representative. The more significant MAMs supporting k-NN similarity queries, which can be systematically implemented by range queries, are carried out to test their performances. In all the experiments, the Euclidean distance is used to measure (dis-)similarity. In addition, all the built indexed trees are binary trees, as the experiments showed that the multi-way trees were not so effective as the binary trees.

Two real-world databases as well as an artificial database are used in the experiments:
(1) Two real-world databases: the NasaD20 and the ColorsD112, are downloaded from http://www.sisap.org. The NasaD20 contains 40,150 20-dimensional vectors obtained from the NASA video and image archives. The ColorsD112 contains 112,682 112-dimensional vector obtained from the color images.
(2) An artificial uniformly distributed database: the UniformD20, is generated by the code from http://www.sisap.org. The data dimension is set to 20.

The default number of the indexed data in each database are fixed to 10,000. For each database, 1,000 data objects are randomly chosen from the unindexed data as the query objects, and finally the average results of the 1,000 queries are figured out. The values of the parameters like distance calculations for evaluating the search performance are all average values, but the word "average" is omitted.

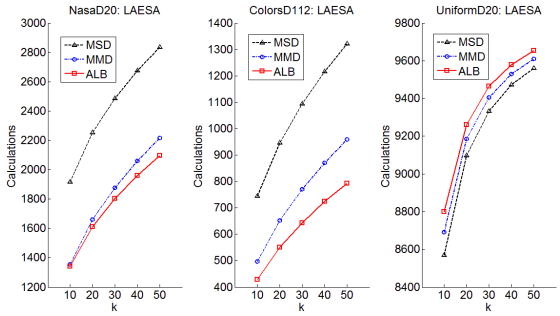

Fig. 6 Comparing calculations as k increases among different pivot selection strategies, in NasaD20 (left), ColorsD112 (middle), UniformD20 (right), by LAESA.

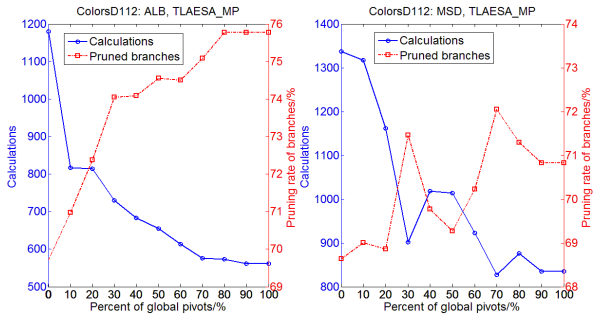

Fig. 7 Calculations & pruning rate of branches as percent of global pivots increases, in ColorsD112, in pivot selection strategy: ALB (left), MSD (right), by TLAESA_MP.

6.1 Comparing Pivot Selection Strategies

Good pivots play a vital role in reducing distance calculations and enhancing the search efficiency. We need a better or proper strategy to select a set of good global pivots for the successive study. In the following experiments, the three pivot selection strategies, the MSD and the MMD based on the data distribution, and the ALB based on the evaluation criterion are compared for the optimal. Because the TLAESA, the iTLAESA, and the proposed eTLAESA, can be viewed as the extensions of the LAESA with the tree structure to do pre-pruning, the LAESA is just used to evaluate the selected pivots.

After a certain number of pivots are selected by each pivot selection strategy, the k-NN queries are carried out with them. The value of k is from 10 to 50 at intervals of 10. The calculations with the variable k are shown in Fig. 6, where the pivot-number is fixed to 100, but even if the pivot-number is added to 200 or subtracted to 50, those curves still keep the same relative positions.

From Fig. 6 we can find that in the two real-world databases NasaD20 and ColorsD112, the MSD strategy calculates most while the ALB strategy calculates least whichever value k is, and it is true that the MMD strategy is more effective than the MSD strategy in terms of calculations. However, it is opposite in the artificial database UniformD20, and the MSD strategy calculates least. In addition, to our surprise, too many calculations took place in the database UniformD20 compared to the same dimensional database NasaD20. We can say it is harder to search in the high dimensional uniformly distributed data instead. Besides, we would ask the same question in the paper [25] whether it is valid to test pivot selection strategies in uniformly distributed spaces as usual, because the results in the uniformly distributed database are probably counter to those in the real-world databases, and the selected pivots for the uniformly distributed database can slightly cut down the distance calculations compared to the linear scan for k-NN queries if k is much bigger.

Since all the applications should be put into the real-world databases at last, the two real-world databases are directly selected for the following experiments, and without special illustrations the ALB strategy is adopted for selecting global pivots due to the fewest distance calculations in the real-world databases.

6.2 Testing TLAESA_MP

The TLAESA_MP intends to combine some global pivots with some representatives in the higher levels to improve the pruning rate, but the fact could be that if we adopt a proper strategy to acquire a set of optimal or good global pivots, given the same number of pivots, the distance calculations are fewer when more global pivots are used. This can be observed in the following experiments of the common 20-NN queries, where the total number of pivots is fixed to 100, and the percent of global pivots is increased from 0% to 100% at intervals of 10%.

The results on the distance calculations plus the pruning rate of branches with the variable percent of global pivots in the database ColorsD112 are depicted in Fig. 7 (left). Note that the search strategy is the best-first strategy with the heuristic factor 1, and the pivot selection strategy is the ALB. We can clearly find that as the percent of global pivots increases, the distance calculations trend to decrease while the pruning rate trends to increase. A few node-pivots can not make the calculations and the pruning rate change when the percent of global pivots reaches 90% or more.

If we use an improper pivot selection strategy to get a set of not so good global pivots, when we combine them with some node-pivots, indeed the distance calculations can be reduced a lot and meanwhile a high pruning rate can be acquired. We can take the same queries in the database ColorsD112 as an example. Note that the pivot selection strategy is changed to the MSD. In Fig. 7 (right), there are fewest distance calculations and a highest pruning rate when the percent of global pivots is 70%. Overall the ALB strategy can save more distance calculations than the MSD strategy at the same percent of global pivots, and then we regard the former as a proper pivot selection strategy. However, it remains a question that because of relativity a new improved pivot selection strategy could make the old one improper, and usually we don't know whether the

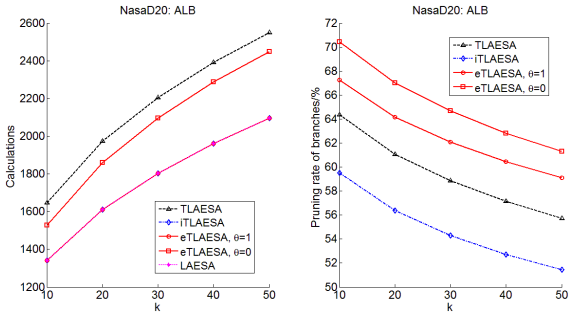

Fig. 8 Comparing calculations (left), pruning rate of branches (right) as k increases among different search methods, in NasaD20, in ALB.

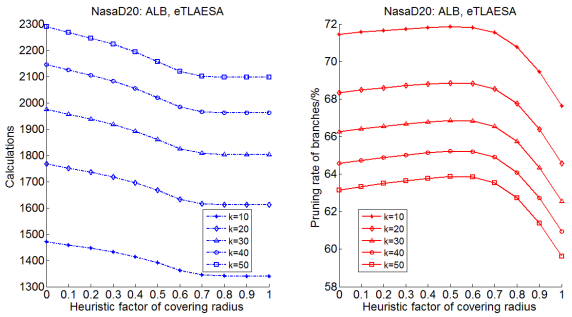

Fig. 9 Calculations (left), pruning rate of branches (right) as the heuristic factor of covering radius increases for different valus of k, in NasaD20, in ALB, by eTLAESA.

selected pivots are good pivots or how good they are for an unknown database.

6.3 Evaluating the Proposed eTLAESA

6.3.1 General k-NN Queries

In the experiments of the general k-NN queries, the value of k is from 10 to 50 at intervals of 10. For different search methods the selected global pivots are the same in the same database, and the number of the global pivots is fixed to 100.

The proposed method eTLAESA is compared with the TLAESA and the iTLAESA. As for the search strategies, the TLAESA uses the depth-first search strategy, the iTLAESA uses the original best-first search strategy, and the eTLAESA also uses the best-first search strategy but with two different heuristic factors of the covering radius which are 1 and 0. The search results of the calculations and pruning rate of branches in the database NasaD20 are shown in Fig. 8. The similar results can be obtained in the database ColorsD112. In the figures, "$\theta = 1$" denotes that the heuristic factor is 1, while "$\theta = 0$" denotes that the heuristic factor is 0.

We can find the curves of calculations of the LAESA, the iTLAESA, and the eTLAESA with the heuristic factor 1 completely overlap, which means that their calculations are the same. Probably the LAESA has the fewest calculations given the same pivots, and then the iTLAESA and the eTLAESA with the heuristic factor 1 can reach their limited minimum number of distance calculations compared to the LAESA. We can also find that the iTLAESA can save more calculations than the TLAESA, but its pruning rate is the lowest. The emergence of the eTLAESA improves this plight and succeeds in transcending the TLAESA on the pruning rate.

Compared to the eTLAESA with the heuristic factor 0, the eTLAESA with the heuristic factor 1 can reach the limited fewest calculations, but it can not gain the highest pruning rate of branches. It seems as if there exists a "trade-off" between the distance calculations and the pruning rate, but the following experiments shows that we can use a heuristic factor lower than 1 to make the pruning rate higher, and meanwhile to keep the limited fewest distance calculations unchanged.

6.3.2 Optimizing the Heuristic Factor

In the following experiments on the eTLAESA, some k-NN queries, where the value of k is from 10 to 50 at intervals of 10 and the number of the pivots is fixed to 100, are carried out when the heuristic factor is increased from 0 to 1 at intervals of 0.1. The following Fig. 9 shows the curves on the calculations and the pruning rate of branches as the heuristic factor increases in the database NasaD20.

We can find that there exists a breakthrough point of the heuristic factor for different k-NN queries, and when the heuristic factor surpasses that point, the number of the distance calculations almost keeps the same minimum till the heuristic factor is 1, while the pruning rate drops sharply after a peak. The similar results can be acquired in the other databases with different pivot selection strategies, but the breakthrough point may be different.

In this study the breakthrough point is selected as the optimal value of the heuristic factor. In order to make the distance calculations completely reach the minimum amount, the optimal value can be slightly bigger than the breakthrough point. Thus, it is possible that the eTLAESA can achieve a higher pruning rate while keeping the same fewest distance calculations as the LAESA even if the heuristic factor of the covering radius is not 1.

6.3.3 Special 20-NN Queries

Unlike the general k-NN queries above, in this section we focus on a specific k-NN query which is the widely applied 20-NN query to evaluate the performance of the proposed method eTLAESA compared to the TLAESA and the iTLAESA. The heuristic factors of the covering radius applied to the eTLAESA are two values, which are a fixed value 1 and an optimal value. The 20-NN queries are carried out in the two real-world databases as the number of the selected global pivots is gradually increased. Note that the max number of the global pivots is restricted to 5% of the indexed data, and the same pivot set is used for the same database.

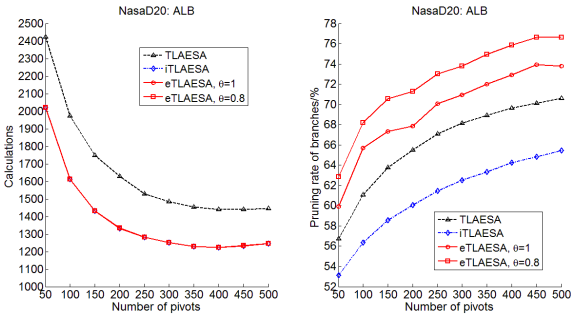

Fig. 10 Comparing calculations (left), pruning rate of branches (right) as number of pivots increases for 20-NN queries among different search methods, in NasaD20, in ALB.

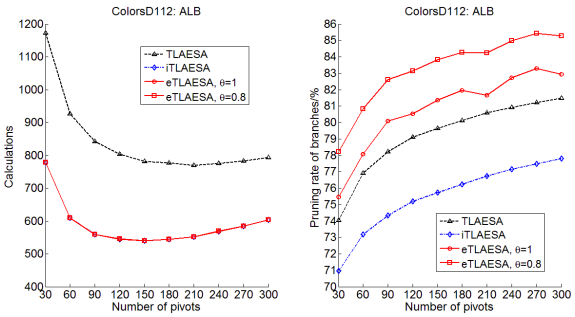

Fig. 11 Comparing calculations (left), pruning rate of branches (right) as number of pivots increases for 20-NN queries among different search methods, in ColorsD112, in ALB.

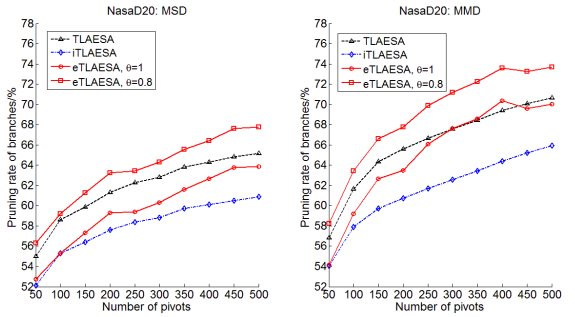

Fig. 12 Comparing pruning rate of branches as number of pivots increases for 20-NN queries among different search methods, in NasaD20, in MSD (left) and MMD (right).

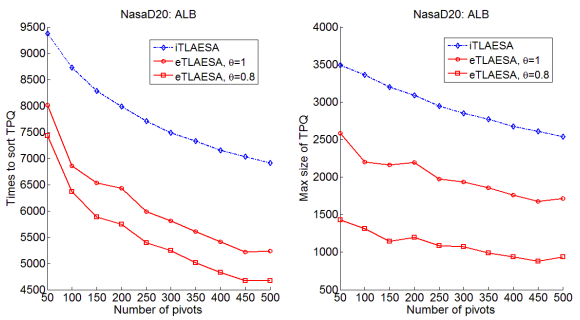

Fig. 13 Comparing times to sort TPQ (left), max size of TPQ (right) as number of pivots increases for 20-NN queries among different search methods, in NasaD20, in ALB.

The optimal heuristic factor is set to 0.8 both in the two real-world databases. Then the curves on the calculations and the pruning rates of branches are displayed in Fig. 10 and Fig. 11. We can draw the similar conclusions as the general k-NN queries above except that, the eTLAESA with the heuristic factor 0.8 not only has reached the limited fewest distance calculations due to the nearly overlapped curves on distance calculations, but also obtains the highest pruning rate.

Note that in the experiments above the pivot selection strategy is the ALB. If the pivot selection strategy is the MSD or the MMD, the results on the pruning rate of the search tree in the database NasaD20 are shown in Fig. 12. We can find that the iTLAESA indeed does not improve the pruning rate as its pruning rate is the lowest when the pivot selection strategy is the MSD or the MMD, although it can reach the fewest distance calculations. Besides, the proposed method eTLAESA with the heuristic factor 1 performs better than the iTLAESA, but it still can not or partly transcend the TLASEA, particularly when the pivot selection strategy is the MSD. However, we can adjust the heuristic factor to some extent, and when we let it be 0.8, the pruning rate becomes the highest (the calculations are almost the same) in either of the pivot selection strategies.

The times to access and sort the TPQ are counted on condition that the elements in the TPQ need be sorted one time when a new element is added. What's more, the max size of the TPQ is also computed according to the max number of the elements contained in the TPQ. The following experiments are carried out in the two real-world databases, but only the results in the database NasaD20 are displayed due to the similar results. In Fig. 13, we can find that the TPQ need be frequently sorted by the iTLAESA and its size is rather huge, but by the eTLAESA, particularly with the optimal heuristic factor 0.8, the times to sort the TPQ and also the max size of the TPQ decrease sharply. These results are in accordance with our expectation in that generally by the best-first search strategy, the lower the pruning rate of the search tree is, the less the number of the nodes are added into the TPQ, furthermore, the fewer the times to sort the TPQ are, and furthermore, the smaller the max size of the TPQ is.

As for how fast those methods can execute the 20-NN queries, the search CPU-time cost can tell us. The following Fig. 14 shows the curves on the CPU-time cost as the number of the pivots increases in the database NasaD20. By the way, the CPU-time cost is measured in the Ubuntu 11 operating system with a $4G\ Hz$ CPU. We can see clearly that the iTLAESA spends the most CPU-time. As long as there are enough pivots, about 250 or more, the eTLAESA with the heuristic factor 1 can save more CPU-time than the TLAESA, not to mention the eTLAESA with the optimal heuristic factor 0.8.

As the previous research, we should determine an optimal number of pivots for the further comparisons. It has

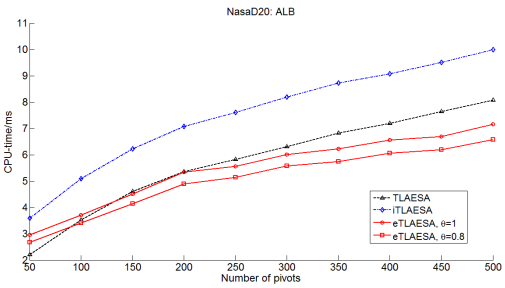

Fig. 14 Comparing CPU-time cost as number of pivots increases for 20-NN queries among different search methods, in NasaD20, in ALB.

been shown by experiments [8] that the optimal number of pivots does not depend on the number of indexed data when the data reach a certain scale, but depends on the intrinsic dimensionality of the space, and the number of nearest neighbors to be queried [26]. Of course, the distribution of the data and the pivot selection strategy also affect the optimal number of pivots. Although the previous researches provide some optimal number of pivots, they are mainly for the uniformly distributed database in lower dimensional spaces. As many factors affect the optimal number of pivots, the best way to determine that number in an unknown database is by experiments.

Actually we determine the optimal number of pivots only according to the minimum number of distance calculations. From the curves on the calculations with the increased number of pivots in Fig. 10 and Fig. 11, we determine roughly in Table 1 and Table 2 the optimal number of pivots in the two real-world databases for each index method. In fact, the optimal number of pivots for the eTLAESA is almost the same as that for the iTLAESA as their curves almost overlap, but may be different from that for the TLAESA. As an example, we can find that in the database ColorsD112, the optimal number of pivots for the eTLAESA is less than that for the TLAESA, which means that to achieve the same performance the eTLAESA or the iTLATSA requires fewer pivots and a smaller size of distance matrix.

Some evaluation parameters like the distance calculations, pruning rate of branches, for 20-NN queries are computed according to the optimal number of pivots in the same pivot selection strategy ALB, and they are also listed in the tables. Note that " × " means no such evaluation parameters.

From Table 1, in the database NasaD20, compared to the TLAESA, the iTLAESA reduces about 15.0% distance calculations, but it can not reduce the pruning rate and the CPU-time cost. Then compared to the iTLAESA, when the heuristic factor is 1, the eTLAESA keeps the same calculations, but it increases the pruning rate of search tree about 8.7%, and decreases the times to sort the TPQ about 24.3%, the max size of the TPQ about 34.2%, the CPU-time cost about 27.9%. Furthermore, when the heuristic factor is 0.8, the eTLAESA keeps almost the same calculations, but it increases the pruning rate of the search tree about 11.7%,

Table 1 Evaluation parameters in NasaD20.

	TL-AESA	iTL-AESA	eTLAESA	
			$\theta = 1$	$\theta = 0.8$
Pivot-number	400	400	400	400
Calculations	1440.2	1224.2	1224.2	1224.6
Pruned branches	69.7%	64.2%	72.9%	75.9%
Times to sort TPQ	×	7153.2	5416.4	4825.0
Max size of TPQ	×	2676.8	1762.2	937.5
CPU-time	7.20ms	9.08ms	6.55ms	6.06ms

Table 2 Evaluation parameters in ColorsD112.

	TL-AESA	iTL-AESA	eTLAESA	
			$\theta = 1$	$\theta = 0.8$
Pivot-number	210	150	150	150
Calculations	769.0	539.3	539.3	539.5
Pruned branches	80.6%	75.7%	81.4%	83.8%
Times to sort TPQ	×	4856.7	3725.2	3235.4
Max size of TPQ	×	1918.6	1299.6	722.0
CPU-time	3.51ms	3.72ms	2.62ms	2.44ms

and decreases the times to sort the TPQ about 32.6%, the max size of the TPQ about 65.0%, the CPU-time cost about 33.3%. At last, compared to the TLAESA, the eTLAESA with the heuristic factor 0.8 can save about 15.0% distance calculations, increase the pruning rate about 6.2%, and decrease the CPU-time cost about 15.9%.

We can similarly analyze the data in Table 2. In the database ColorsD112, when the heuristic factor is 0.8, compared to the iTLAESA, the eTLAESA increases the pruning rate about 8.1%, and decreases the times to sort the TPQ about 33.4%, the max size of the TPQ about 62.4%, the CPU-time cost about 34.4%. While compared to the TLAESA, the eTLAESA can save about 29.9% distance calculations, increase the pruning rate about 3.2%, and decrease the CPU-time cost about 30.5%.

7. Conclusions

We proposed a method called the eTLAESA to improve the state of art method TLAESA by arranging the selected global pivots in the higher levels of the search tree as representatives as possible. As more real distances instead of lower bound estimations of the node-representatives are used for approximating the closet node and "branch and bound" in the earlier stage, not only which nodes are close to the query can be evaluated more effectively, but also the pruning rate of branches can be enhanced, and further the times to access and sort the priority queue used for the best-first search strategy can be greatly reduced. Moreover, if the selected global pivots are not so good, to some extent the eTLAESA can adjust the heuristic factor of the covering radius of the nodes for an optimal value to make the pruning rate transcend the TLAESA.

The eTLAESA builds an optimal number or a smaller number of global pivots into the higher level of the search tree as possible and works well, but it degrades into the iTLAESA which is an improved version of the TLAESA if all the index objects are selected as the

pivots. Compared to the iTLAESA, the eTLAESA can not save more distance calculations as it reaches the limited fewest distance calculations given the same pivots, but it can greatly improve the pruning rate, the times to sort the priority queue and the search efficiency measured in CPU-time by experiments in different real-world databases. The experiments also show that for answering the common 20-NN queries, in the same proper pivot selection strategy, compared to the TLAESA, the eTLAESA can save more distance calculations and enhance the pruning rate to achieve a higher search efficiency with the same or fewer optimal pivots.

It remains a work in future to apply the eTLAESA to the other more complex metric spaces like the metric EMD (Earth Mover's Distance) and evaluate the performance.

Acknowledgments

This research was supported in part by a grant from the Grant-in-Aid for Scientific Research nos. 24501136 from the Ministry of Education, Science, and Culture, Japan.

References

[1] D. Ma, X. Zhai, and Y. Peng,, "Cross-media retrieval by cluster-based correlation analysis," 20th IEEE International Conference on Image Processing, pp.3986–3990, 2013.
[2] O. Marques and B. Furht, Content-Based Image and Video Retrieval, pp.35–46, Kluwer Academic Publishers, 2002.
[3] K.S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is nearest neighbor meaningful?," Proc. Internat. Conf. on Database Theory, Lecture Notes Comput. Sci., vol.1540, pp.217–235, 1999.
[4] C. Böhm, S. Berchtold, and D.A. Keim, "Searching in high dimensional spaces: index structures for improving the performance of multimedia databases," ACM Computing Surveys, vol.33, no.3, pp.322–373, 2001.
[5] T. Skopal, "Unified framework for fast exact and approximate search in dis-similarity spaces," ACM Trans. Database Systems, vol.V, no.N, pp.1–45, 2007.
[6] V. Sepulveda and B. Bustos, "CP-Index: using clustering and pivots for indexing non-metric spaces," Proc. SISAP'10, pp.75–82, Istanbul, Turkey, 2010.
[7] E. Vidal, "An algorithm for finding nearest neighbors in (approximately) constant average time," Pattern Recognit. Lett., vol.4, no.3, pp.145–157, 1986.
[8] 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 Recognit. Lett., vol.15, no.1, pp.9–17, 1994.
[9] L. Micó, J. Oncina, and R.C. Carrasco, "A fast branch & bound nearest neighbour classifier in metric spaces," Pattern Recognit. Lett., vol.17, no.7, pp.731–739, 1996.
[10] J.K. Uhlmann, "Satisfying general proximitysimilarity queries with metric trees," Inf. Process. Lett., vol.40, no.4, pp.175–179, 1991.
[11] P.N. Yianilos, "Data structures and algorithms for nearest neighbor search in general metric spaces," Annual ACM-SIAM Symposium on Discrete Algorithms, pp.311–321, Austin, USA, 1993.
[12] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: an efficient access method for similarity search in metric spaces," Proc. 23rd Internat. Conf. on Very Large Data Bases (VLDB), pp.426–435, Athens, Greece, 1997.
[13] E. Chavez, G. Navarro, R. Baeza-Yates, and J.L. Marroquin, "Proximity searching in metric spaces," ACM Computing Surveys, vol.33, no.3, pp.273–321, 2001.
[14] T. Skopal, J. Pokorný, and V. Snášel, "PM-tree: Pivoting metric tree for similarity search in multimedia databases," 8th East-European Conf. on Advances in Databases and Information Systems (ADBIS), pp.99–114, Budapest, Hungary, 2004.
[15] K. Tokoro, K. Yamaguchi, and S. Masuda, "Improvements of TLAESA nearest neighbour search algorithm and extension to approximation search," 29th Australasian Computer Science Conf. (ACSC), pp.77–83, Hobart, Tasmania, Australia, vol.48, 2006.
[16] K. Figueroa, E. Chávez, G. Navarro, and R. Paredes, "Speeding up spatial approximation search in metric spaces," J. Experimental Algorithmics, vol.14, no.6, pp.6–21, 2009.
[17] P. Aibar, A. Juan, and E. Vidal, "Extension to the Approximating and Eliminating Search Algorithm (AESA) for finding k-nearest-neighbours," Technical Report DSIC II/29/93, Dept. DSIC, Univ. Politécnica de Valencia, 1993.
[18] J.M. Vilar, "Reducing the overhead of the AESA metric-space nearest neighbour searching algorithm," Inf. Process. Lett., vol.56, no.8, pp.265–271, 1995.
[19] K. Figueroa, E. Chávez, G. Navarro, and R. Paredes, "On the least cost for proximity searching in metric spaces," WEA 2006, Lect. Notes Comput. Sci., vol.4007, pp.279–290, 2006.
[20] R. Socorro, L. Micó, and J. Oncina, "A fast pivot-based indexing algorithm for metric spaces," Pattern Recognit. Lett., vol.32, no.11, pp.1511–1516, 2011.
[21] E. Bugnion, S. Fei, T. Roes, P. Widmayer, and F. Widmer, "A spatial index for approximate multiple string matching," First South American Workshop on String Processing, pp.43–53, Belo Horizonte, Brazil, 1993.
[22] L. Micó and J. Oncina, "Comparison of fast nearest neighbour classifiers for handwritten character recognition," Pattern Recognit. Lett., vol.19, no.3–4, pp.351–356, 1998.
[23] N. Brisaboa, A. Farina, O. Pedreira, and N. Reyes, "Similarity search using sparse pivots for efficient multimedia information retrieval," Proc. 8th IEEE Internat. Symposium on Multimedia, IEEE Computer Society, pp.881–888, Washington, DC, USA, 2006.
[24] B. Bustos, O. Pedreira, and N. Brisaboa, "A dynamic pivot selection technique for similarity search," First Internat. Workshop on Similarity Search and Applications, IEEE Computer Society, Washington, DC, USA, pp.105–112, 2008.
[25] B. Bustos, G. Navarro, and E. Chávez, "Pivot selection techniques for proximity searching in metric spaces," Pattern Recognit. Lett., vol.24, no.14, pp.2357–2366, 2003.
[26] F. Moreno-Seco, L. Micó, and J. Oncina, "Extending LAESA fast nearest neighbour algorithm to find the k-nearest neighbors," Lect. Notes Comput. Sci., vol.2396, pp.718–724, 2002.

Dong Wang was born in 1982. He received the master degree in Mechanical Engineering from the Beijing Institute of Technology, China, in 2008. His Ph.D. study is on Multimedia Information Retrieval at the Dept. of Information Science and Intelligent System, the University of Tokushima, Japan. Now his main research interests are Multimedia Information Indexing, Contents Based Image Retrieval, and Metric Spaces.

Hiroyuki Mitsuhara was born in 1975. He received the B.E. and M.E. degrees from Kinki University in 1998 and 2000, and then he received the Ph.D. degree from The University of Tokushima in 2003. He is currently an associate professor at The University of Tokushima. His research interests include entertainment computing, human computer interaction, e-Learning, and Web-based learning. He received the IEEE Young Researcher Award 2011 from IEEE Education Society Japan Chapter.

Masami Shishibori was born in 1965. He received his BS Degree in 1991, his MS Degree in 1993 and PhD Degree in 1997, from Tokushima University, Japan. He is currently a Professor in the Department of Information Science and Intelligent Systems at Tokushima University, Japan. His research interests include multimedia processing, information retrieval, natural language processing.