New Distance Lower Bounds for Efficient Proximity Searching in Metric Spaces

Tao Ban and Youki Kadobayashi [*]

Abstract—**The prune rules derived from the triangle inequality has long been the most prevalent technique to avoid distance computations when fast nearest neighbor (NN) searching is sought in a general metric spaces. In this paper we introduce a group of new lower bounds to estimate the unknown distances in a metric space. The new bounds are proven to be tighter than the triangular inequality induced lower bound and have better pruning performance. The bounds take the assumption of positive semi-definite metric space models and are adaptable to most of the modern applications. In the simulations, the new lower bounds have led to significant improvement upon the search efficiency of AESA as well as the Linear AESA algorithm, which are known as the baseline of the known metric search algorithms.**

Keywords: Metric Search, fast search, nearest neighbor search, AESA

1 Introduction

Proximity searching consists in retrieving relevant information satisfying user formulated query conditions from a database. Since the evaluation of the similarity between queries and database items is computationally expensive, there have been many attempts to build indexing structures using as few distance computations as possible to answer the queries. In the framework of fast nearest neighbor (NN) searching in general metric spaces, the technique called Approximating and Eliminating Search Algorithm (AESA) [13, 14] is probably the fastest one in terms of distance computations required during NN search. The AESA searches for the distance from a query object to its NN prototype through the triangle inequality based lower bound function, both for selecting prototypes which are gradually closer to the query object (Approximation), and for pruning out prototypes whose lower bound estimates are no less than the smallest distance found from the query object to the already examined prototypes (Elimination). As a result, empirical tests have shown that the AESA performs NN search in (approximately) constant average time [13].

[*]The authors are with information Security Research Center, National Institute of Information and Communications Technology, Tokyo, 184-8795 Japan. Email: bantao@nict.go.jp, youki-k@is.aist-nara.ac.jp.

AESA has been for 20 years the algorithm that requires the least number of distance evaluations to answer proximity queries. In fact, all the development on metric indexes methods can be seen as attempts to simulate the performance of AESA using less memory [4]. There have been some algorithms aimed at reducing its preprocessing time or employed storage space. LAESA [8] chooses M elements from the dataset as potential pivots, and reduces the storage cost to $O(MN)$, where N is the number of prototypes. An improved version of LAESA is Tree LAESA [9] which achieves sublinear side computations at query time at the expense of doubling the number of distance computations on average. Reduced Overhead AESA [15] strictly calculates the same distances as AESA but reduces the query processing time. Recently, graph t-spanner indexes [10] were used to simulate AESA, obtaining almost the same number of distance calculations and using much less memory. In [5] a new technique called iAESA is introduced to choose the next pivot, which guesses better a close candidate and yields some reductions in the number of distance evaluations. In [6], the author suggests applying the technique of AESA to most of the existing metric search structures to efficiently reduce the number of distance computations during search.

All of the above mentioned methods—in fact most of the available metric search algorithms—are based on the distance lower bound derived from the triangular inequality (see Lemma 1 in Section 2.2). Although this lower bound is computationally cheap, the accuracy degenerates quickly as the dimension of the data increases. It is interesting to observe that, the triangular inequality is defined in a 1D embedding space: all prototypes are embedded in the space defined by the distance from the pivot. Hence we call this lower bound as the 1D lower bound hereafter. Assume that the set of prototypes can be embedded into a Euclidean space with a finite dimension c. It can be expected that more accurate lower bounds on inter-prototype distances can be obtained in a higher dimensional embedding space: The higher the dimension of the embedding space, the better the projected distance approaches the metric distance. When the dimension of the embedding space exceeds c, the projected distance will be identical to the corresponding metric distance. Following this idea, we propose two lower bounds to estimate the inter-prototype distances in the metric space. For easy conceivability, the new lower bounds are defined in the 2D and 3D embeddings of the metric space. With the same storage cost, the novel lower bounds are both experimentally proved to be tighter than the 1D lower bound. Especially, the 3D lower bound is theoretically proven to be tighter than the 1D counterpart. To evaluate the efficiency of the proposed distance lower bounds, we incorporate them in two search search algorithms. The first one is an extension from classical AESA and aims to estimate a new baseline for nearest neighbor searching in metric spaces. The second algorithm is a mimic of LAESA which requires much less storage than AESA with the cost of a few increased distance computations.

The rest of the paper is organized as follows. In Section 2, we briefly review the problem of metric search together with the classical AESA and LAESA algorithms. Section 3 introduces two novel distance lower bounds on inter-prototypes distances in general metric spaces. Section 4 specifies the detailed implementation of the search algorithms adopting these lower bounds. In the experiment section, numerical results on a series of simulations are reported. Section 6 concludes the paper. Because of space limitation, proofs of the lemmas and some detailed discussions are put in the appendix.

2 Related Works

Before specifying the AESA search strategies, we first review the properties of a metric space model and commonly used metric queries.

2.1 Similarity Search in Metric Spaces

Let $\mathbb{D}$ be the domain of prototypes, $d : \mathbb{D} \times \mathbb{D} \to \mathcal{R}$ a distance measure on $\mathbb{D}$, the tuple $\mathcal{M} = (\mathbb{D}, d)$ is called a *metric space*, if $\forall \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{z} \in \mathbb{D}$, the following conditions hold [2].

$$\begin{aligned} &d(\boldsymbol{u},\boldsymbol{v}) \geq 0 && \text{non}-\text{negativity} && (1)\\ &d(\boldsymbol{u},\boldsymbol{v}) = 0 \Leftrightarrow \boldsymbol{u}=\boldsymbol{v} && \text{identity} && (2)\\ &d(\boldsymbol{u},\boldsymbol{v}) = d(\boldsymbol{v},\boldsymbol{u}) && \text{symmetry} && (3)\\ &d(\boldsymbol{u},\boldsymbol{v}) + d(\boldsymbol{v},\boldsymbol{z}) \geq d(\boldsymbol{u},\boldsymbol{z}) && \text{triangular inequality} && (4)\end{aligned}$$

A metric query is generally defined by a query object $\boldsymbol{q}$ and a proximity condition. For a nearest-neighbor search, the algorithm retrieves the closest prototype to $\boldsymbol{q}$ as the result. The concept can be generalized to search for the k nearest neighbors, thus the result set of a kNN search $Q_k = (\boldsymbol{q}, k, \mathbb{S})$ is

$$\mathbb{K}(\boldsymbol{q}, k, \mathbb{S}) = \mathbb{R} : \{\mathbb{R} \subseteq \mathbb{S}, |\mathbb{R}| = k, \\ \forall \boldsymbol{u}_i \in \mathbb{R}, \boldsymbol{v} \in \mathbb{S} \setminus \mathbb{R}, d(\boldsymbol{q}, \boldsymbol{u}_i) \leq d(\boldsymbol{q}, \boldsymbol{v})\}. \quad (5)$$

For simplicity, we confine our discussion on the nearest neighbor query which is the most widely explored metric search type. All the discussions can be easily extended to a k-nearest neighbor algorithm by maintaining a list of the k candidates seen so far and using the largest distance among the k candidates for the elimination. Refer to [11] for detailed discussions on other kinds of queries.

2.2 AESA

Following [13, 14], the key to the use of AESA in performing nearest neighbor search is the following property directly derived from the triangular inequality in (4).

Lemma 1 *(1D lower and upper bounds) Let $\mathcal{M} = (\mathbb{D}, d)$ be a metric space, $\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{q} \in \mathbb{D}$. The following inequalities holds:*

$$d_{1\mathrm{D}}(\boldsymbol{q}, \boldsymbol{u}|\boldsymbol{p}) = |d(\boldsymbol{q}, \boldsymbol{p}) - d(\boldsymbol{u}, \boldsymbol{p})| \leq d(\boldsymbol{q}, \boldsymbol{u}), \quad (6)$$
$$D_{1\mathrm{D}}(\boldsymbol{q}, \boldsymbol{u}|\boldsymbol{p}) = |d(\boldsymbol{q}, \boldsymbol{p}) + d(\boldsymbol{p}, \boldsymbol{u})| \geq d(\boldsymbol{q}, \boldsymbol{u}). \quad (7)$$

Thus if $\mathbb{P}$ is the set of prototypes whose distances from $\boldsymbol{q}$ and $\boldsymbol{u}$ are known, the greatest lower bound $d_{1\mathrm{D}}(\boldsymbol{q}, \boldsymbol{u}|\mathbb{P})$ on $d(\boldsymbol{q}, \boldsymbol{u})$ for any prototype $\boldsymbol{u} \in \mathbb{D}$ is

$$d_{1\mathrm{D}}(\boldsymbol{q}, \boldsymbol{u}|\mathbb{P}) = \max_{\boldsymbol{p}_i \in \mathbb{P}} |d(\boldsymbol{q}, \boldsymbol{p}_i) - d(\boldsymbol{u}, \boldsymbol{p}_i)|. \quad (8)$$

AESA uses this lower bound during search to eliminate prototypes $\boldsymbol{u} \in \mathbb{S} \setminus \mathbb{P}$ whose lower bounds of distance from $\boldsymbol{q}$ are greater than the distance from $\boldsymbol{q}$ to the nearest neighbor candidate $\boldsymbol{u}_\mathrm{n}$. The algorithm first initializes $\mathbb{P}$ to the empty set, $\mathbb{U} = \mathbb{S} \setminus \mathbb{P}$, and $d_{1\mathrm{D}}(\boldsymbol{q}, \boldsymbol{u})$ to 0 for all $\boldsymbol{u} \in \mathbb{U}$. At each search step, the next pivot $\boldsymbol{p}$ is select as the one with the minimal lower bound from $\mathbb{U}$ and its distance from $\boldsymbol{q}$ is computed. The algorithm then updates the distance lower bounds for the remaining prototypes in $\mathbb{U}$ and eliminates the ones with the distance lower bounds greater than $d(\boldsymbol{q}, \boldsymbol{u}_\mathrm{n})$. The algorithm terminates once there is no more prototype left in $\mathbb{U}$. Note that the nearest neighbor candidate, $\boldsymbol{u}_\mathrm{n}$, is updated if necessary when distance computation is invoked.

2.3 LAESA

The main drawback of the AESA approach is the quadratic storage requirement and large preprocessing cost. LAESA alleviates this drawback by choosing a fixed number, M, of pivots whose distances from all other prototypes are computed and stored in advance. Thus, for a dataset consisted of N prototypes, the distance matrix contains $N \times M$ entries rather than $O(N^2)$ for AESA. The search algorithm of LAESA is very similar to that of AESA, especially when the distance from $\boldsymbol{q}$ to the selected pivots are preferentially computed.

In particular, let $\mathbb{P}$ be the set of selected pivots from $\mathbb{S}$, $\mathbb{U} = \mathbb{S} \setminus \mathbb{P}$ be the rest of the prototypes. To implement the approximating and eliminating strategy, LAESA first computes the distances between $\boldsymbol{q}$ and all pivots $\boldsymbol{p} \in \mathbb{P}$ and then estimates the distance lower bounds for all the prototypes in $\mathbb{U}$ applying (8). These lower bounds allow eliminating prototypes from $\mathbb{U}$ whose lower bound estimates from $\boldsymbol{q}$ are greater than the distance from $\boldsymbol{q}$ to the current nearest neighbor candidate. The remaining prototypes are sequentially compared with $\boldsymbol{q}$ in ascending order of the distance lower bound. Note that in the case of a distance computation the nearest neighbor candidate is updated if necessary.

3 Proposed Lower and Upper Bounds

It is interesting to ask that with a fixed set of pivots, can we obtain lower bound on the distance between two prototypes $\boldsymbol{q}$ and $\boldsymbol{u}$ tighter than that produced by (8). As suggested in [1], with a slightly more strict assumption of the metric space model, we can effectively tighten the lower bound. In the following, we first discuss a Euclidean case and then generalize the discussions to metric spaces.

3.1 Embedding of the Metric Space

In a multi-dimensional Euclidean space, a 2D embedding space can be decided by a triple of non-identical points $\boldsymbol{o}$, $\boldsymbol{p}$ and $\boldsymbol{u}$, as shown in Figure 1. Let $\overrightarrow{\boldsymbol{op}}$ be a coordinate axis, with $\boldsymbol{o}$ being the origin and $\boldsymbol{p}$ defining the positive direction. Let the projection of $\boldsymbol{u}$ on the axis be $\boldsymbol{u}'$. Then by the law of cosines, the projected distance between $\boldsymbol{o}$ and $\boldsymbol{u}$ along $\overrightarrow{\boldsymbol{op}}$ is:

$$\begin{aligned}\bar{d}(\boldsymbol{o},\boldsymbol{p};\boldsymbol{u}) &= d(\boldsymbol{o},\boldsymbol{u}') = \cos\theta d(\boldsymbol{o},\boldsymbol{u}) \\ &= \frac{1}{2d(\boldsymbol{o},\boldsymbol{p})}\big(d^2(\boldsymbol{o},\boldsymbol{u}) + d^2(\boldsymbol{o},\boldsymbol{p}) - d^2(\boldsymbol{u},\boldsymbol{p})\big). \qquad (9)\end{aligned}$$

We can see that the projected distance $\bar{d}(\boldsymbol{o},\boldsymbol{p};\boldsymbol{u})$ can be computed using only the inter-prototype distances, thus lower bound derived from (9) can be possibly applied to the metric space models. Then it is natural to ask that under what circumstance can a metric space model give rise to a configuration of points, $\{\boldsymbol{x}_i\}$, in a Euclidean space, so that the associated Euclidean distance $L_2(\boldsymbol{x}_i,\boldsymbol{x}_j) \equiv d(\boldsymbol{u}_i,\boldsymbol{u}_j)$ for all $\boldsymbol{u}_i,\boldsymbol{u}_j \in \mathbb{S}$. We have the following lemma to the answer of this question.

Lemma 2 *Let $\mathcal{M} = (\mathbb{D}, d)$ be a metric space, $\mathbb{S} = \{\boldsymbol{u}_i, i = 1,\cdots,N\} \subset \mathbb{D}$. Define the inter-prototype squared distance matrix as $\boldsymbol{B}^{N\times N}$, where $[\boldsymbol{B}]_{ij} = d^2(\boldsymbol{u}_i,\boldsymbol{u}_j)$, $i,j = 1,\cdots,N$ and the gram matrix $\boldsymbol{G}$ as*

$$\boldsymbol{G} = -\frac{1}{2}(\boldsymbol{I} - \frac{1}{N}\boldsymbol{1}\boldsymbol{1}^T)\boldsymbol{B}(\boldsymbol{I} - \frac{1}{N}\boldsymbol{1}\boldsymbol{1}^T), \qquad (10)$$

where $\boldsymbol{1} = \{1,1,\cdots,1\}$ is the n-ary all ones vector and $\boldsymbol{I}$ the identity matrix. If $\boldsymbol{G}$ is positive semi-definite, then there exists a configuration $\boldsymbol{x}_i$, $i = 1,\cdots,N$ in a Euclidean space with a dimension up to N which satisfies

$$L_2(\boldsymbol{x}_i,\boldsymbol{x}_j) \equiv d(\boldsymbol{u}_i,\boldsymbol{u}_j),\ (i,j = 1,\cdots,N). \qquad (11)$$

Proof of Lemma 2 and the way to find the configuration is discussed in the appendix. We call a metric space model which can always produce a positive semi-definite gram matrix as positive semi-definite. Fortunately, most of the commonly used metric space models are positive semi-definite so that the lower bounds derived from this property are applicable for most metric search problems. Note that a positive semi-definite metric space is in accordance with the positive semi-definite kernels [12] which is extensively studied recently in the field of machine learning. In fact, a positive semi-definite kernel always corresponds to a positive semi-definite metric space. Hence, the techniques explored here are readily adaptable to most of the modern applications.

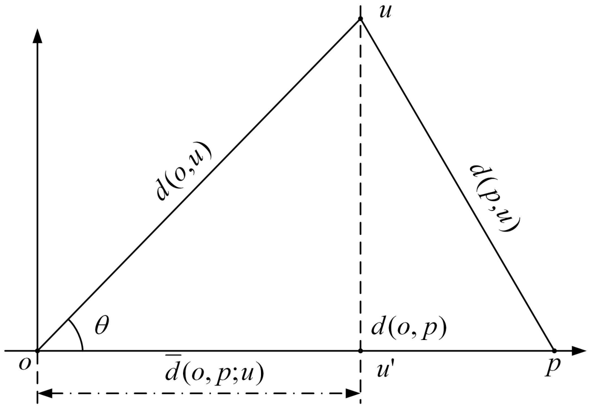

Figure 1: Distances in the 2D Euclidean embedding.

3.2 New Bounds on the Distances

By Lemma 2, we have the following lemma which defines a novel lower bound on the distance between two prototypes, given their distances to two pivots are known.

Lemma 3 *(2D lower bound) Let $\mathcal{M} = (\mathbb{D}, d)$ be a positive semi-definite metric space. $\boldsymbol{o},\boldsymbol{p},\boldsymbol{u} \in \mathbb{D}$, $\boldsymbol{o} \neq \boldsymbol{p}$. Then for any $\boldsymbol{q} \in \mathbb{D}$, the following inequality holds:*

$$d_{2\mathrm{D}}(\boldsymbol{p},\boldsymbol{u}|\boldsymbol{o},\boldsymbol{p}) = |\bar{d}(\boldsymbol{o},\boldsymbol{p};\boldsymbol{q}) - \bar{d}(\boldsymbol{o},\boldsymbol{p};\boldsymbol{u})| \leq d(\boldsymbol{q},\boldsymbol{u}). \qquad (12)$$

In Lemma 3, only the projected distance along $\overrightarrow{\boldsymbol{op}}$ is considered. The following lemma employs the distance perpendicular to $\overrightarrow{\boldsymbol{op}}$ for a tighter lower bound.

Lemma 4 *(3D lower and upper bounds) Let $\mathcal{M} = (\mathbb{D}, d)$ be a positive semi-definite metric space, $\boldsymbol{o},\boldsymbol{p},\boldsymbol{u},\boldsymbol{q} \in \mathbb{D}$, $\boldsymbol{o} \neq \boldsymbol{p}$. Let $\boldsymbol{u}'$ and $\boldsymbol{q}'$ be the projections of $\boldsymbol{u}$ and $\boldsymbol{q}$ on $\overrightarrow{\boldsymbol{op}}$ respectively. Then the following inequality holds:*

$$\begin{aligned}d(\boldsymbol{q},\boldsymbol{u}) &\geq d_{3\mathrm{D}}(\boldsymbol{p},\boldsymbol{u}|\boldsymbol{o},\boldsymbol{p}) \\ &= \sqrt{d^2(\boldsymbol{u},\boldsymbol{q}|\boldsymbol{o},\boldsymbol{p}) + (d(\boldsymbol{q},\boldsymbol{q}') - d(\boldsymbol{u},\boldsymbol{u}'))^2}. \qquad (13)\end{aligned}$$

Similarly $d(\boldsymbol{q},\boldsymbol{u})$ is upper bounded by

$$\begin{aligned}d(\boldsymbol{q},\boldsymbol{u}) &\leq D_{3\mathrm{D}}(\boldsymbol{p},\boldsymbol{u}|\boldsymbol{o},\boldsymbol{p}) \\ &= \sqrt{d^2(\boldsymbol{u},\boldsymbol{q}|\boldsymbol{o},\boldsymbol{p}) + (d(\boldsymbol{q},\boldsymbol{q}') + d(\boldsymbol{u},\boldsymbol{u}'))^2}. \qquad (14)\end{aligned}$$

Generally, when incorporated in a metric search algorithm, the tighter the distance lower bound the better the elimination performance of the algorithm to prune out unnecessary distance computations. It is obvious that the 3D lower bound is tighter its the 2D counterpart. For positive semi-definite metric spaces, given more than two nonidentical pivots, we can prove that the 3D lower bound is also tighter than the 1D lower bound defined in Lemma 1, as stated in the following lemma.

Lemma 5, *Let $\mathcal{M} = (\mathbb{D}, d)$ be a positive semi-definite metric space, $\boldsymbol{o}, \boldsymbol{p} \in \mathbb{D}, (\boldsymbol{o} \neq \boldsymbol{p})$ are pivots. The following inequalities hold:*

$$d_{3\mathrm{D}}(\boldsymbol{q}, \boldsymbol{u}|\boldsymbol{o}, \boldsymbol{p}) \geq \max\big(d_{1\mathrm{D}}(\boldsymbol{q}, \boldsymbol{u}|\boldsymbol{o}), d_{1\mathrm{D}}(\boldsymbol{q}, \boldsymbol{u}|\boldsymbol{p})\big), \quad (15)$$

$$D_{3\mathrm{D}}(\boldsymbol{q}, \boldsymbol{u}|\boldsymbol{o}, \boldsymbol{p}) \leq \max\big(D_{1\mathrm{D}}(\boldsymbol{q}, \boldsymbol{u}|\boldsymbol{o}), D_{1\mathrm{D}}(\boldsymbol{q}, \boldsymbol{u}|\boldsymbol{p})\big). \quad (16)$$

See the proof of the lemmas in the appendix A.

4 Searching Algorithms

In this section we specify the searching algorithms with the new lower bounds incorporated with the search strategy of AESA and LAESA. We term the new distance bound as *projection based distance bound*, so that the following algorithms are dubbed PAESA (the Projection-based AESA) and LPAESA.

4.1 PAESA

As specified in Table 1, PAESA basically follows the algorithm of AESA [13, 14]. The major difference between PAESA and AESA is the application of the new distance lower bound for approximation and elimination. In AESA, for each pivot $\boldsymbol{p}$ selected from the dataset, Lemma 1 is applied only once to update the associated lower bound $d_{1\mathrm{D}}(\boldsymbol{q}, \boldsymbol{u}_i)$ on $d(\boldsymbol{q}, \boldsymbol{u}_i)$. However, for PAESA, when a pivot is added to the pivot set, multiple lower bounds of $d(\boldsymbol{q}, \boldsymbol{u}_i)$ can be computed according to Lemma 3 or Lemma 4. Thus, with m pivots selected, the lower bound $d_{2\mathrm{D}}(\boldsymbol{q}, \boldsymbol{u}_i)$ for $d(\boldsymbol{q}, \boldsymbol{u}_i)$, or $d_{3\mathrm{D}}(\boldsymbol{q}, \boldsymbol{u}_i)$ for $d(\boldsymbol{q}, \boldsymbol{u}_i)$, is selected as the maximum from the $m(m+1)/2$ approximations. Hence the lower bound produced by PAESA will generally be much tighter than that of AESA and hence more distance computations can be saved. Note that, on line 13 of Table 1, in the case Lemma 3 is applied to get the lower bound, we term the algorithm as PAESA2D, otherwise it is dubbed PAESA3D if Lemma 4 is employed.

4.2 LPAESA

Like AESA, PAESA also requires a distance matrix storing $O(N^2)$ inter-prototype distances, which becomes impractical for large datasets. Following the idea of LPAESA, we can alleviate this drawback by choose a fixed number of M pivots, whose distances from all the prototypes are computed and stored beforehand.

```
0  Inputs: $\boldsymbol{q}$, $\mathbb{S} = \{\boldsymbol{s}_i | i = 1, \cdots, N\}$;
1  $\mathbb{P} \leftarrow \mathbb{P}$; // set of pivots
2  $\mathbb{U} \leftarrow \mathbb{S}$; // set of non-pivots
3  $D(\boldsymbol{u}_i) \leftarrow 0$, for $\boldsymbol{u}_i \in \mathbb{U}$; // lower bound
4  $d_\mathrm{n} \leftarrow \infty$; // distance to the nearest neighbor
5  while $\mathbb{U} \neq \emptyset$
6      $\boldsymbol{p} \leftarrow \arg\min_{\boldsymbol{u}_i \in \mathbb{U}} D(\boldsymbol{u}_i)$;
7      $\mathbb{U} \leftarrow \mathbb{U} \setminus \{\boldsymbol{p}\}$;
8      if $d(\boldsymbol{q}, \boldsymbol{p}) < d_\mathrm{n}$ then // distance evaluation
9          $d_\mathrm{n} \leftarrow d(\boldsymbol{q}, \boldsymbol{p})$;
10         $\boldsymbol{b} \leftarrow \boldsymbol{p}$; // update the nearest neighbor
11     for $\boldsymbol{u}_i \in \mathbb{U}$ do
12         $D_\mathrm{low}(\boldsymbol{u}_i) \leftarrow \max_{\boldsymbol{p}_j \in \mathbb{P}} appro(\boldsymbol{q}, \boldsymbol{u}_i, \boldsymbol{p}, \boldsymbol{p}_j)$;
13         $D(\boldsymbol{u}_i) \leftarrow \max(D(\boldsymbol{u}_i), D_\mathrm{low}(\boldsymbol{u}_i))$;
14         if $(D(\boldsymbol{u}_i) \geq d_\mathrm{n})$ then
15             $\mathbb{U} \leftarrow \mathbb{U} \setminus \{\boldsymbol{u}_i\}$; // elimination
16     $\mathbb{P} \leftarrow \mathbb{P} \cup \{\boldsymbol{p}\}$;
17 return $\boldsymbol{b}$; // Output: the nearest neighbor
```

Table 1: The PAESA metric search algorithm.

Table 2 specifies the search algorithm of LPAESA when a set of M pivots, $\mathbb{P}$, are previously selected from the dataset. During the search, the query object is first compared against the pivots and then the distances from the pivots are used to compute the lower bounds of the non-pivots. After that, the non-pivots are visited in ascending order of the lower bound until no prototype in the set can be nearest neighbor of the query object. Note that the distance to the current nearest neighbor candidate is updated when a distance computation is invoked. The LPAESA is termed as LPAESA2D or LPAESA3D according to the employed lower bound.

4.3 Pivot Selection

To speed up the search, LPAESA requires a distance matrix consisting the computed distances between pivots in the pivot set and all the prototypes in the dataset. Note that the number of selected pivots as M, the storage cost of LPAESA is $O(MN)$.

As pointed out in [3], the way pivots are selected affects the search performance of a metric search algorithm. Among the pivot selection heuristics studied in [3], the incremental selection strategy shows the best performance for real world metric spaces both in terms of approximating accuracy and computation cost. In the experiment, we employ an incremental selection algorithm, as specified in Table 3. to select the pivot set. By defining the distance from a prototype $\boldsymbol{u}$ to a set of prototypes $\mathbb{P}$ as the minimum from $\boldsymbol{u}$ to $\boldsymbol{p}_i \in \mathbb{P}$, the idea can be easily stated as: first initialize the pivot set with a randomly selected prototype, then sequentially select the next pivot as the most separated prototype from the pivot set.

```
0  Inputs: $\boldsymbol{q}$, $\mathbb{S} = \{\boldsymbol{s}_i | i = 1, \cdots, N - M\}$,
          $\mathbb{P} = \{\boldsymbol{p}_i | i = 1, \cdots, M\}$;
1  $\mathbb{U} \leftarrow \mathbb{S}$; // set of non-pivots
2  $d_{\mathrm{n}} \leftarrow \infty$; // distance to the nearest neighbor
3  for $\boldsymbol{p}_i \in \mathbb{P}$ do
4      if $d(\boldsymbol{q}, \boldsymbol{p}_i) < d_{\mathrm{n}}$ then // distance evaluation
5          $d_{\mathrm{n}} \leftarrow d(\boldsymbol{q}, \boldsymbol{p}_i)$;
6          $\boldsymbol{b} \leftarrow \boldsymbol{p}_i$; // update the nearest neighbor
7  for $\boldsymbol{u}_i \in \mathbb{U}$ do
8      $D_{\mathrm{low}}(\boldsymbol{u}_i) \leftarrow \max_{\boldsymbol{p}_j, \boldsymbol{p}_k \in \mathbb{P}} appro(\boldsymbol{q}, \boldsymbol{u}_i, \boldsymbol{p}_j, \boldsymbol{p}_k)$;
9      if $(D_{\mathrm{low}}(\boldsymbol{u}_i) \geq d_{\mathrm{n}})$ then
10         $\mathbb{U} \leftarrow \mathbb{U} \setminus \{\boldsymbol{u}_i\}$; // elimination
11 $\mathbb{U}_{\mathrm{s}} \leftarrow sort(\mathbb{U}, \{D_{\mathrm{low}}(\boldsymbol{u}_i)\})$; // ascending order
12 for $\boldsymbol{u}_i \in \mathbb{U}_{\mathrm{s}}$ do
13     if $(D_{\mathrm{low}}(\boldsymbol{u}_i) \geq d_{\mathrm{n}})$ then
14         break;
15     if $d(\boldsymbol{q}, \boldsymbol{u}_i) < d_{\mathrm{n}}$ then // distance evaluation
16         $d_{\mathrm{n}} \leftarrow d(\boldsymbol{q}, \boldsymbol{u}_i)$;
17         $\boldsymbol{b} \leftarrow \boldsymbol{u}_i$; // update the nearest neighbor
18 return $\boldsymbol{b}$; // Output: the nearest neighbor
```

Table 2: The LPAESA metric search algorithm.

```
0  Inputs: $\mathbb{S}$, $M$;
1  $\mathbb{P} \leftarrow \{\boldsymbol{p}_1 \in \mathbb{S}\}$; // random select the first pivot
2  for $i = 2$ to $M$ do
3      $d_i(\boldsymbol{u}_j, \mathbb{P}) = \min_{\boldsymbol{p}_k \in \mathbb{P}} d(\boldsymbol{u}_j, \boldsymbol{p}_k)$, for $\boldsymbol{u}_j \in \mathbb{S}$;
4      $\boldsymbol{p}_i \leftarrow \arg \max_{u_j \in \mathbb{S} - \mathbb{P}} d_i(\boldsymbol{u}_j, \mathbb{P})$;
5      $\mathbb{P} \leftarrow \mathbb{P} \cup \{\boldsymbol{p}_i\}$ ;
6  return $\mathbb{P}$; // return the pivot set
```

Table 3: Incremental pivot selection algorithm.

5 Experiments

In this section we present the experimental results of the proposed PAESA and LPAESA algorithms. Numerical results on a series of simulations are reported. The performance of the indexing structures is measured by the cutoff of distance calculations. Because AESA and LAESA, which provide the best effectivity in distance computation reduction, have long been the baseline of metric search algorithms, we only compare the proposed algorithms with these two algorithms. Other metric search algorithms—typically with less storage cost—will generally require more distance computations.

In the experiments, the indexed prototypes and the query objects are independently drawn from uniform distributions in c-dimensional unit hypercubes. The coordinates of the data points are never used directly and only the inter-prototype distances are employed in the algorithms. We set the number of prototypes in the datasets to 10,000, and the number of queries to 100. Reported results are averaged over 10 runs.

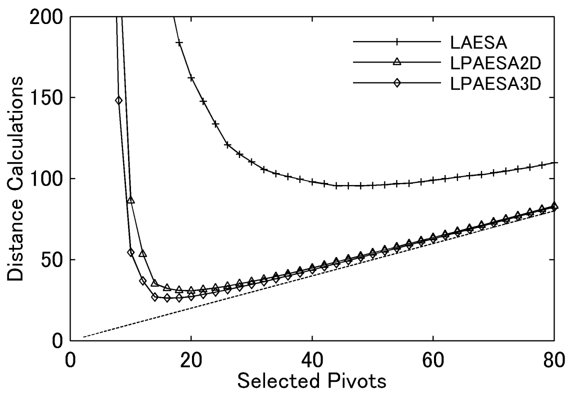

Figure 2: Search performance with varying pivot set size in 10D.

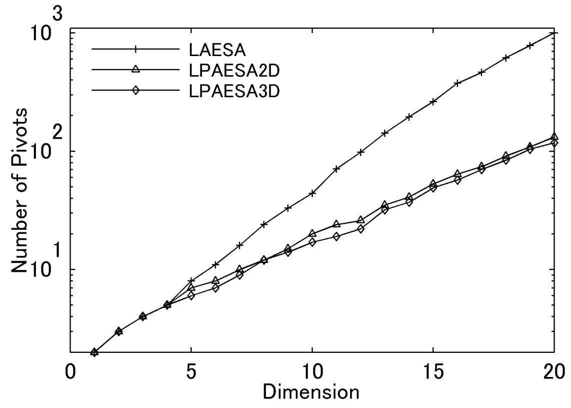

Figure 3: Pivot selection results for different dimensional datasets.

Before the performance evaluation of the algorithms, we first show the results of the pivot selection algorithms for LAESA, LPAESA2D and LPAESA3D. The curves of number distance calculations vs. pivot set size for a 10D dataset are shown in Figure 2. We can find evident saddle points in all the curves. That is, the number of distance calculations first drops because the approximation accuracy increases as pivots are added, however, after a certain threshold, the distance calculations go up since the computations against the pivots considerably increase. Experiments show that, for independently generated datasets with the same distribution, the optimal number of pivots is quite stable. We can also see that the curves for LPAESA2D and LPAESA3D are very close to the dashed line which stands for the number of pivots. This is because that the distance lower bounds given by LPAESA2D and LPAESA3D are very tight so that most of the non-pivot prototypes are pruned without distance computations. In Figure 3, the results of pivot selection for datasets up to dimension 20 are reported. It can be learn from the figure that, the number of pivots selected by LAESA is exponential in the dimension of the dataset. LPAESA2D and and LPAESA3D need comparable numbers of pivots for all the cases. For datasets with dimension over 5 LPAESAs select much fewer numbers of pivots than LAESA. In dimension 10, the three algorithms select 44, 15, and 12 pivots respectively, and in dimension 20, they select 1002, 132, and 118 pivots respectively.

The aim of the second experiment is to compare the performance of the referred metric search algorithms in different dimensions. The algorithms are tested on datasets up to dimension 20. The curves of distance calculations for the nearest neighbor queries are shown in Figure 4. From the curves we can see that the search difficulty increases exponentially in the dimension because of the so called curse of dimensionality. The performance comparison is clear enough: for datasets with lower dimension, since the selected pivots can produce very tight lower bounds, the pivot set based methods are more preferable than the full matrix based methods. When the search difficulty increase as the dimension goes up, accurate enough lower bound can only achieved by extensive usage of prestored distances. For dimension up to 8, we have the following performance order:

$$\begin{aligned} &LAESA \lessdot AESA \lessdot LPAESA2D \\ &\lessdot LPAESA3D \lessdot PAESA2D \lessdot PAESA3D, \end{aligned} \quad (17)$$

where '$\lessdot$' stands for an ascending order in terms of the saved distance computations. Since lemma 4 produces tighter lower bounds than lemma 3, PAESA3D needs slightly less distance computations than PAESA2D, and LPAESA3D needs less than LPAESA2D. However, the differences are not so prominent. If the computation cost to estimate the distance lower bounds is also taken into account—note that lemma 4 costs two or three times more than lemma 3—better choice can be made based on the nature of the application. Another thing to note is the effectiveness of the application of pivot sets. With much less storage cost than PAESAs, LPAESAs need no more than two times distance computations than PAESAs. For the most intensive case in dimension 20, LPAESAs requires about 1% of the space of PAESAs with about 50% more distance computations. So for large scale applications or for systems short on storage resources, LPAESAs are better choices.

In the third experiment we evaluate the algorithms against 10D datasets with sample size, N, varying from 1,000 to 1,3000. The required distance calculations are reported in Figure 5. It is interesting to find that for all these algorithms fewer objects in the dataset does not mean fewer number of distance computations required to retrieve the nearest neighbor. On the contrary, each of the algorithms maintains a stable number of distance computations as the sample size varies. Note that for

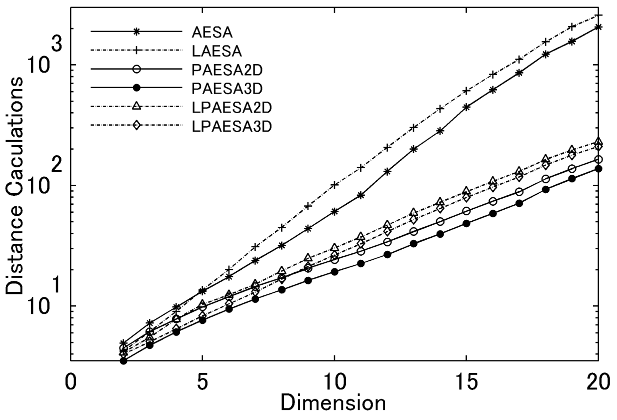

Figure 4: Experiments on varying dimension.

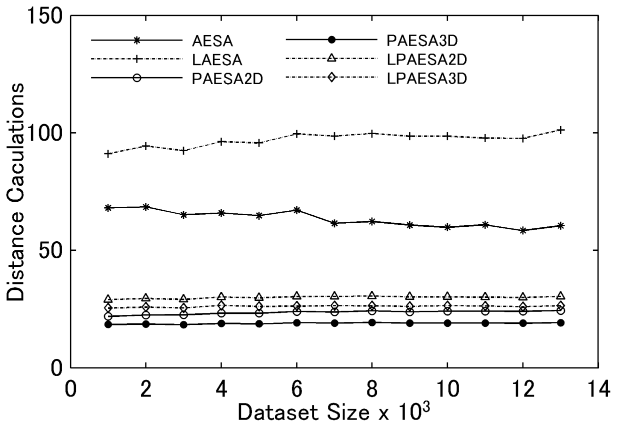

Figure 5: Experiments on varying sample size.

AESA and PAESAs, there is no influence from the selected number of pivots. This property is especially useful for large scale datasets. For all the cases, the performance order in (17) is preserved.

In the fourth experiment, we extend the algorithms to find the k nearest neighbors by introducing an ordered list of the k nearest prototypes to the query object and updating it when distance computation is invoked. The number of pivots are fixed for LAESA and LPAESAs. Figure 6 shows the curves of the distance calculations against k, which changes from 1 to 100. The experiments are done on a 10D dataset with 10,000 prototypes. We can learn from the figure that as k increases, the search difficulty increases quickly. For each of the algorithms, the number of distance calculations is in linear relation to the retrieved number of nearest neighbors. We have to note that this linearity in increased distance computations is only applicable for small k values. Since the number of pivots are fixed for LAESA and LPAESAs in the experiments, the results are in favor of PAESA and AESA. Better results can be produced by LAESA

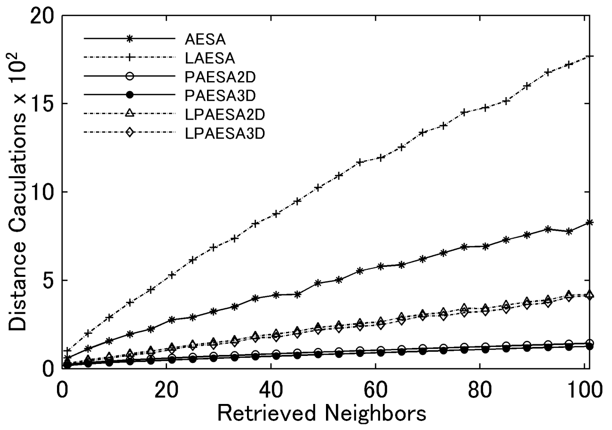

Figure 6: Experiments on varying sample size.

and LPAESAs if more pivots are selected for large k values. A more adaptive strategy could be: suppose a large enough pivot set is selected beforehand, selectively use a proportion of the pivot set according to the user defined k parameter.

From the experiments with different dimensions, sample sizes, and k parameters, we find the following rules. PAESAs have better search efficiency than AESA in all cases in terms of the reduction of distance computations. Similarly, LPAESAs always need fewer distance computations than LAESA. Finally, 3D lower bound imposed search algorithms always invoke fewer distance computations than their 2D lower bound imposed counterparts.

6 Conclusion

In this paper, we have proposed two novel lower bounds for distances in the metric space. The new lower bounds are derived from the geometrical properties in the embedding space and is applicable to positive semi-definite metric space models. When the proposed distance lower bounds are incorporated with the AESA search algorithm, experiments show that when the full distance matrix is employed, the search efficiency of the proposed PAEASAs is much better than AESA especially for high dimensional datasets. For better storage efficiency, we have also applied the new lower bounds following the idea of LAESA. In the experiments, the proposed LPAESAs show not only better performance than LAESA in reduction of distance computations but also lower storage cost in terms of the size of the inter-pivot distance matrix.

For most of the experiments, we have discovered a performance order as shown in (17). Hence we can have the conclusion that, for applications where the metric distance computations are the most essential computational cost, we suggest the application of PAESA. When storage cost are considered, LPAESAs are better choices. Whether apply the 3D lower bound or 2D lower bound depends on the comparative cost of side computations and distance evaluations.

References

[1] Ban, T. and Kadobayashi, Y., New prune rules for similarity search, *in* The 11th IASTED International Conference on Artificial Intelligence and Soft Computing, Palma de Mallorca, Spain, 2007.

[2] Brin S., Near neighbor search in large metric spaces, *in* The 21th International Conference on Very Large Data Bases, pp. 574–584, 1995.

[3] Bustos, B., Navarro, G., and Chávez, E., Pivot selection techniques for proximity searching in metric spaces, *Pattern Recognition Letters,* 24:2357–2366, 2003.

[4] Chávez, E., Navarro, G., and Marroquín, J. L., Searching in Metric Spaces, *ACM Computing Surveys,* 33(3), 273–321, 2001.

[5] Figueroa, K., Chávez, E., Navarro, G., and Paredes, G., On the least cost for proximity searching in metric spaces, *in* Workshop on Experimental and Efficient Algorithms, pp. 279–290, 2006.

[6] Fredriksson, K., Engineering efficient metric indexes, *Pattern Recognition Letters,* 28(1), 75–84, 2007.

[7] Mardia, K. V., Kent, J. T., and Bibby, J. M., *Multivariate Analysis*, London: Academic Press, 1979.

[8] Micó, M. L., Oncina, J., and Vidal, E., A new version of the nearest-neighbor approximating and eliminating search algorithm (AESA) with linear preprocessing time and memory requirements, *Pattern Recognition Letters,* 15(1), 9–17, 1994.

[9] Micó, M. L., Oncina, J., and Carrasco, R. C., A fast branch & bound nearest neighbour classifier in metric spaces, *Pattern Recognition Letters,* 17(7), 731–739, 1996.

[10] Navarro, G., Paredes, R., and Chávez, E., *t*-spanners as a data structure for metric space searching, LNCS 2476, pp. 298–309, 2002.

[11] Samet, H., *Foundations of Multidimensional and Metric Data Structures,* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[12] Shawe-Taylor, J, and Cristianini, N., *Kernel Methods for Pattern Analysis*, Cambridge University Press, Cambridge, England, 2004.

[13] Vidal, E., An algorithm for finding nearest neighbors in (approximately) constant average time, *Pattern Recognition Letters* 4:145–157, 1986.

[14] Vidal, E., New formulation and improvements of the nearest-neighbor approximating and eliminating search algorithm (AESA), *Pattern Recognition Letters* 15(1), 1–7, 1994.

[15] Vilar, J., Reducing the overhead of the AESA metric-space nearest neighbor searching algorithm, *Information Processing Letters* 56:256–271, 1995.

A Proof of the Lemmas

Proof of Lemma 2: Let the coordinates of N points in a c dimensional Euclidean space be given by $\boldsymbol{x}_i\,(i = 1, \cdots, n)$, where $\boldsymbol{x}_i = (x_{i1}, \cdots, x_{ic})^T$. Then the Euclidean distance between the ith and the j th points is given by

$$d_{ij}^2 = (\boldsymbol{x}_i - \boldsymbol{x}_j)^T(\boldsymbol{x}_i - \boldsymbol{x}_j) = \boldsymbol{x}_i^T\boldsymbol{x}_i + \boldsymbol{x}_j^T\boldsymbol{x}_j - 2\boldsymbol{x}_i^T\boldsymbol{x}_j. \quad (18)$$

Let the gram matrix be $\boldsymbol{G} = \{g_{ij}\}$, where

$$g_{ij} = \boldsymbol{x}_i^T\boldsymbol{x}_j. \quad (19)$$

Here we show how to find $\boldsymbol{G}$ from d_{ij}.

Firstly, to overcome the indeterminacy of the solution resulting from translation, we require the centroid of the configuration of points be placed at the origin. That is

$$\sum_{i=1}^{N} x_{il} = 0,\ l = 1, \cdots, c. \quad (20)$$

From (18) and (20), we have

$$\begin{aligned} \frac{1}{N}\sum_{i=1}^{N} d_{ij}^2 &= \boldsymbol{x}_j^T\boldsymbol{x}_j + \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i^T\boldsymbol{x}_i, \\ \frac{1}{N}\sum_{j=1}^{N} d_{ij}^2 &= \boldsymbol{x}_i^T\boldsymbol{x}_i + \frac{1}{N}\sum_{j=1}^{N}\boldsymbol{x}_j^T\boldsymbol{x}_j, \\ \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}^2 &= \frac{2}{N}\sum_{i=1}^{N}\boldsymbol{x}_i^T\boldsymbol{x}_i. \end{aligned} \quad (21)$$

Substituting (21) to (18) gives

$$g_{ij} = -\frac{1}{2}\Bigg(d_{ij}^2 - \frac{1}{N}\sum_{i=1}^{N} d_{ij}^2 - \frac{1}{N}\sum_{j=1}^{N} d_{ij}^2 + \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}^2\Bigg). \quad (22)$$

Define matrix $\boldsymbol{A}$ as $a_{ij} = \frac{-1}{2}d_{ij}^2$, then the gram matrix $\boldsymbol{G}$ can be computed from

$$\boldsymbol{G} = \boldsymbol{HAH}, \quad (23)$$

where $\boldsymbol{H}$ is the centering matrix,

$$\boldsymbol{H} = \boldsymbol{I} - \frac{1}{N}\boldsymbol{1}\boldsymbol{1}^T. \quad (24)$$

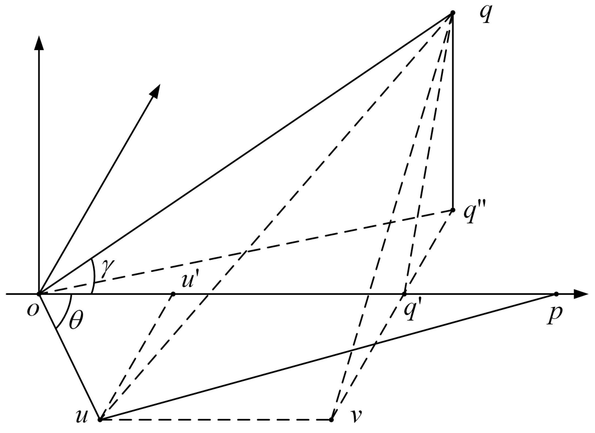

Figure 7: Distances in the 3D Euclidean embedding.

Following [7], Suppose dissimilarities $\delta_{ij} = \delta(\boldsymbol{u}_i, \boldsymbol{u}_j)$, where $\delta(\cdot,\cdot)$ is a metric distance function, are used instead of d_{ij} to define matrix $\boldsymbol{A}$, which is then centered to produce the gram matrix $\boldsymbol{G}$. Note that from the definition of metric space model and (23), both $\boldsymbol{A}$ and $\boldsymbol{G}$ are real valued symmetric matrices. Then it is guaranteed that $\boldsymbol{G}$ can be diagonalized whilst its eigenvalues λ_i $(i = 1, \cdots, c)$ and their associated eigenvectors $\boldsymbol{v}_i$ are real. Thus we have

$$\boldsymbol{G} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T, \quad (25)$$

where $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \cdots, \lambda_c), \boldsymbol{V} = \boldsymbol{v}_1, \cdots, \boldsymbol{v}_c$.

Equation (25) can also be rewritten as

$$\boldsymbol{G} = \boldsymbol{X}\boldsymbol{X}^T \quad (26)$$

where $\boldsymbol{X} = [\boldsymbol{x}_i]^T$, $\boldsymbol{x}_i = \lambda^{\frac{1}{2}}\boldsymbol{v}_i$. If $\boldsymbol{G}$ is positive semi-definite, then $\boldsymbol{x}_i$ are real valued vectors.

Now the distance between the ith and jth points of the configuration is given by $(\boldsymbol{x}_i - \boldsymbol{x}_j)^T(\boldsymbol{x}_i - \boldsymbol{x}_j)$, and hence

$$\begin{aligned} (\boldsymbol{x}_i - \boldsymbol{x}_j)^T(\boldsymbol{x}_i - \boldsymbol{x}_j) &= \boldsymbol{x}_i^T\boldsymbol{x}_i + \boldsymbol{x}_j^T\boldsymbol{x}_j - 2\boldsymbol{x}_i^T\boldsymbol{x}_j \\ &= g_{ii} + g_{jj} - g_{ij} \\ &= a_{ii} + a_{jj} - 2a_{ij} \\ &= -2a_{ij} = d_{ij}^2, \end{aligned} \quad (27)$$

by substituting for g_{ij} using (22). Hence the distance between the $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in the Euclidean space is equal to the dissimilarity $\delta(\boldsymbol{u}_i, \boldsymbol{u}_j)$.■

Proof of Lemma 3: Lemma 3 follows directly from the basic properties of the Euclidean space. Because $\overrightarrow{\boldsymbol{q'u'}}$ is the projection of $\overrightarrow{\boldsymbol{qu}}$ along $\overrightarrow{\boldsymbol{op}}$, the equation holds if and only if $\overrightarrow{\boldsymbol{qu}}$ parallels $\overrightarrow{\boldsymbol{op}}$.■ *Proof of Lemma 4:* In Figure 7, the prototypes are shown in the 3D space defined by $\boldsymbol{o}$, $\boldsymbol{p}$, $\boldsymbol{u}$, and $\boldsymbol{q}$. The projection of $\boldsymbol{q}$ onto axis $\overrightarrow{\boldsymbol{op}}$ is marked as $\boldsymbol{q}'$ and its projection onto the plane decided by $\boldsymbol{o}, \boldsymbol{p}, \boldsymbol{u}$

is marked as $\boldsymbol{q}''$. Line $\overrightarrow{\boldsymbol{uv}}$ is parallel to $\overrightarrow{\boldsymbol{op}}$ and cuts the extension line of $\overrightarrow{\boldsymbol{q}''\boldsymbol{q}'}$ at point $\boldsymbol{v}$. Then we have

$$\bar{d}(\boldsymbol{o},\boldsymbol{p};\boldsymbol{u}) - \bar{d}(\boldsymbol{o},\boldsymbol{p};\boldsymbol{q}) = d(\boldsymbol{u}',\boldsymbol{q}') = d(\boldsymbol{u},\boldsymbol{v}). \tag{28}$$

Since $\triangle \boldsymbol{uvp}$ is a right-angled triangle, by the Pythagorean theorem, we have

$$d(\boldsymbol{q},\boldsymbol{u}) = \sqrt{d^2(\boldsymbol{u},\boldsymbol{v}) + d^2(\boldsymbol{q},\boldsymbol{v})} \tag{29}$$

In $\triangle \boldsymbol{qq'v}$, from the triangular inequality,

$$d(\boldsymbol{q},\boldsymbol{v}) \geq |d(\boldsymbol{q},\boldsymbol{q}' - d(\boldsymbol{q}',\boldsymbol{v})|. \tag{30}$$

Substituting (28) and (30) to (29) and replacing $d(\boldsymbol{q}',\boldsymbol{v})$ with $d(\boldsymbol{u},\boldsymbol{u}')$ gives (13). Similarly, we have (14).■

Proof of Lemma 5: Here, we use two consequent prototypes to denote the distance between them. Then, following the denotations in Lemma 4 and Figure 7, we have

$$\begin{aligned} d^2_{3\mathrm{D}}(\boldsymbol{q},\boldsymbol{u}|\boldsymbol{o},\boldsymbol{p}) &= (\boldsymbol{u'o} - \boldsymbol{q'o})^2 + (\boldsymbol{uu'} - \boldsymbol{qq'})^2 \\ &= \boldsymbol{u'o}^2 - 2\boldsymbol{u'o}\cdot\boldsymbol{q'o}^2 + \boldsymbol{q'o}^2 \\ &\quad + \boldsymbol{uu'}^2 - 2\boldsymbol{uu'}\cdot\boldsymbol{qq'} + \boldsymbol{qq'}^2 \\ &= \boldsymbol{u'o}^2 + \boldsymbol{uu'}^2 + \boldsymbol{q'o}^2 + \boldsymbol{qq'}^2 \\ &\quad -2(\cos\theta\cos\gamma + \sin\theta\sin\gamma)\boldsymbol{uo}\cdot\boldsymbol{qo} \\ &= \boldsymbol{uo}^2 + \boldsymbol{qo}^2 - 2\cos(\theta-\gamma)\boldsymbol{uo}\cdot\boldsymbol{qo} \\ &\geq d^2_{1\mathrm{D}}(\boldsymbol{q},\boldsymbol{u}|\boldsymbol{o}). \end{aligned} \tag{31}$$

In the fourth line of the deduction, we have made use of the product-to-sum trigonometric identities. Similarly, we have

$$d^2_{3\mathrm{D}}(\boldsymbol{q},\boldsymbol{u}|\boldsymbol{o},\boldsymbol{p}) \geq d^2_{1\mathrm{D}}(\boldsymbol{q},\boldsymbol{u}|\boldsymbol{p}). \tag{32}$$

(15) follows from (31) and (32). Similarly we have (16). ■

B Analysis of Computational Cost

For applications with computationally intensive metric distances, side computation other than the distance computation are simply ignored for easy evaluation of various metric search methods. However, as suggested in [15], the search performance of AESA like algorithms can be still improved by carefully designed search procedures. Although how to minimize the side computations is beyond the scope of this paper, in this subsection, we give some discussions on the implementation details of the evaluation of the lower bounds.

As we have mentioned, to apply Lemma 1, we have to know $d(\boldsymbol{u},\boldsymbol{p})$ and $d(\boldsymbol{q},\boldsymbol{p})$, with $d(\boldsymbol{u},\boldsymbol{p})$ stored within the indexing structure and $d(\boldsymbol{q},\boldsymbol{p})$ computed during the search. Evaluation of the 1D lower bound is costless: 1 addition and 1 $fabs()$ function call are needed. At each search step, $\big(|\mathbb{P}||\mathbb{U}|\big)$ lower bounds are evaluated, where $|\mathbb{P}|$ is the number of selected pivots and $|\mathbb{U}|$ the number of remaining prototypes.

Applying Lemma 3 generally needs more side computations. From (??) we can see that $d^2_{2\mathrm{D}}(\boldsymbol{p},\boldsymbol{u}|\boldsymbol{o},\boldsymbol{p})$ can be computed from $d(\boldsymbol{u},\boldsymbol{o})$, $d(\boldsymbol{u},\boldsymbol{p})$, $d(\boldsymbol{q},\boldsymbol{o})$, $d(\boldsymbol{q},\boldsymbol{p})$, and $d(\boldsymbol{p},\boldsymbol{o})$. For computational efficiency, we evaluate $d^2_{2\mathrm{D}}(\boldsymbol{p},\boldsymbol{u}|\boldsymbol{o},\boldsymbol{p})$ instead of $d_{2\mathrm{D}}(\boldsymbol{p},\boldsymbol{u}|\boldsymbol{o},\boldsymbol{p})$. Accordingly, the squared inter-prototype distances are stored in the distance matrix. We compute $d^2_{2\mathrm{D}}(\boldsymbol{p},\boldsymbol{u}|\boldsymbol{o},\boldsymbol{p})$ in two steps:

$$t = d^2(\boldsymbol{u},\boldsymbol{o}) - d^2(\boldsymbol{u},\boldsymbol{p}) - d^2(\boldsymbol{q},\boldsymbol{o}) + d^2(\boldsymbol{q},\boldsymbol{p}), \tag{33}$$

$$d^2_{2\mathrm{D}}(\boldsymbol{p},\boldsymbol{u}|\boldsymbol{o},\boldsymbol{p}) = \frac{0.25}{d^2(\boldsymbol{o},\boldsymbol{p})}t\cdot t. \tag{34}$$

Hence, to evaluate the 2D lower bound, 3 additions, 2 multiplications, 1 division, and 1 assignments are invoked.

Things become a little complicated to compute the squared 3D lower bound. From (31) we have

$$\begin{aligned} d^2_{3\mathrm{D}}(\boldsymbol{q},\boldsymbol{u}|\boldsymbol{o},\boldsymbol{p}) = \boldsymbol{uo}^2 + \boldsymbol{qo}^2 - \overbrace{2\boldsymbol{uo}\cdot\boldsymbol{qo}\cos\theta\cos\gamma}^{T_1} \\ - \underbrace{2\boldsymbol{uo}\cdot\boldsymbol{qo}\sin\theta\sin\gamma}_{T_2}. \end{aligned} \tag{35}$$

From the law of cosines, we have

$$\cos\theta = \frac{\boldsymbol{uo}^2 + \boldsymbol{op}^2 - \boldsymbol{up}^2}{2\boldsymbol{uo}\cdot\boldsymbol{op}}, \tag{36}$$

$$\cos\gamma = \frac{\boldsymbol{qo}^2 + \boldsymbol{op}^2 - \boldsymbol{qp}^2}{2\boldsymbol{qo}\cdot\boldsymbol{op}}. \tag{37}$$

For fast evaluation, define two temporary variables

$$t_1 = \boldsymbol{uo}^2 + \boldsymbol{op}^2 - \boldsymbol{up}^2, \tag{38}$$

$$t_2 = \boldsymbol{qo}^2 + \boldsymbol{op}^2 - \boldsymbol{qp}^2. \tag{39}$$

So the term T_1 in (35) can be computed as

$$T_1 = \frac{0.5}{\boldsymbol{op}^2}t_1\cdot t_2, \tag{40}$$

and term T_2 can be computed as

$$\begin{aligned} T_2 &= 2\boldsymbol{uo}\cdot\boldsymbol{qo}\sqrt{(1-\cos^2\theta)(1-\cos^2\gamma)} \\ &= 2\boldsymbol{uo}\cdot\boldsymbol{qo}\sqrt{(1-\frac{t_1^2}{4\boldsymbol{uo}^2\cdot\boldsymbol{op}^2})(1-\frac{t_2^2}{4\boldsymbol{qo}^2\cdot\boldsymbol{op}^2})} \\ &= \frac{0.5}{\boldsymbol{op}^2}\sqrt{(4\boldsymbol{uo}^2\cdot\boldsymbol{op}^2 - t_1^2)(4\boldsymbol{qo}^2\cdot\boldsymbol{op}^2 - t_2^2)}. \end{aligned} \tag{41}$$

Substituting (40) and (41) to (35) gives

$$\begin{aligned} d^2_{3\mathrm{D}}(\boldsymbol{q},\boldsymbol{u}|\boldsymbol{o},\boldsymbol{p}) &= \boldsymbol{uo}^2 + \boldsymbol{qo}^2 \\ -\frac{0.5}{\boldsymbol{op}^2}\big(t_1t_2 + &\big((4\boldsymbol{uo}^2\cdot\boldsymbol{op}^2 - t_1^2)(4\boldsymbol{qo}^2\cdot\boldsymbol{op}^2 - t_2^2)\big)^{\frac{1}{2}}\big). \end{aligned} \tag{42}$$

Hence, to evaluate the 3D lower bound, 9 additions, 9 multiplications, 1 division, 2 assignments, and 1 $sqrt()$ function call will be invoked.

At each search step, PAESA and LPAESA need to evaluate $\big(\frac{1}{2}|\mathbb{P}|(|\mathbb{P}|+1)|\mathbb{U}|\big)$ lower bounds.