On Tighter Inequalities for Efficient Similarity Search in Metric Spaces

Tao Ban and Youki Kadobayashi *

Abstract—**Similarity search consists of the efficient retrieval of relevant information satisfying user formulated query conditions from a database with prebuilt indexing structures. Since the evaluation of the distance functions between queries and indexed objects is often computationally expensive, there have been many attempts to build indexing structures that use as few distance computations as possible to answer queries. Among these methods, for 20 years the Approximating and Eliminating Search Algorithm (AESA) has been the baseline in terms of the required distance computations. By storing a pre-computed inter-object distance matrix, AESA is able to extensively apply the triangle-inequality based pruning rules to avoid unnecessary distance computations.**

In this paper, to further improve the performance of AESA, we introduce a novel group of pruning rules that are proven to be tighter than the triangle-inequality based rules and hence can further reduce the number of distance computations during the search. The new pruning rules require the assumption of positive semi-definite metric space models and can be used in most modern applications. With some slight modification, they can be easily extended to search algorithms in general metric spaces. In the simulations, when incorporated with the proposed pruning rules, AESA showed a significant improvement in distance-computation reduction. For low dimensional problems, applying the new pruning rules cut the distance computations in half, and for high dimensional problems, the reduction was sometimes more than 90%. The pruning rules were also applied to LAESA, a variant of AESA which imposes a linear storage requirement. For this algorithm, they not only helped to save more distance computations, but considerably reduced the storage requirement as well.

Keywords: Similarity search, nearest neighbor search, metric space model, positive semi-definite metric

1 Introduction

We are faced today with a rapidly increasing amount of published information. It would be hard to survive this information explosion without the search engines that are readily accessible on the web, PC, and various database servers. In these systems, similarity search techniques are always involved when retrieving relevant information from a large amount of data with limited computational resources. In addition, in the fields of data mining, computer vision, and pattern recognition, similarity search has also been playing an important role as a prominent data processing step.

The goal for a similarity search application is often one of the following [18], [25]: *Point Query*: Finding objects having particular feature values; *Range Query*: Finding objects whose feature values fall within a given range in relation to some query object; *Nearest Neighbor Query*: Finding the closest object to the query object within an object set; *Spatial Join Query*: Finding pairs of objects from the same set or different sets that are sufficiently similar to each other. To achieve these goals in spite of specific computation, storage, and especially time limitations, many similarity search algorithms have been proposed. One popular approach is mapping data objects into feature vectors and then conducting a search in the feature space. This method is computationally efficient because the geometrical properties of the feature space can help to speed up the search. However, it also introduces the undesirable element of indirection into the process. Moreover, for sophisticated applications—for example, when the indexed data consists of strings and trees—finding the vectorial presentation of the data is itself an open question. Another, and more elegant, approach is to define a distance function directly between objects and then build indexing structures based on the properties of the distance function. Researches have shown that if the distance satisfies some basic conditions, including non-negativity, identity, symmetry, and triangle inequality, efficient algorithms can be formulated for fast similarity search [5], [18], [12], [25]. Such a function, together with the data domain, are generally known as a metric space. A metric space model is considered a more general methodology for similarity search, since in some data domains, defining a distance function between objects can be accomplished more intuitively than mapping objects to feature vectors. Examples might include unstructured data such as multimedia objects, text documents, protein sequences, images, etc.

*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.

In the framework of fast similarity search in metric spaces, the widely adopted criterion used to evaluate the performance of search algorithms is the number of distance computations invoked during the search. This is because the evaluation of complicated metric distance functions is usually much more computationally intensive than side operations. In this sense, the technique called the Approximating and Eliminating Search Algorithm (AESA) [21], [22] is probably the fastest metric search algorithm. During a search, AESA extensively applies the triangle-inequality to approximate the distances from the query object to the indexed objects and eliminates objects that do not satisfy the query condition. It is reported to be able to answer nearest-neighbor queries with an approximately constant number of distance computations. However, the bottleneck of AESA is its quadratic storage requirement. To make use of the triangle-inequality, AESA requires that an $N \times N$ inter-object distance matrix be pre-computed and stored in the main memory, where N is the number of indexed objects. For 20 years, AESA has been the baseline method for metric search algorithms, which aim to answer similarity queries with as few distance computations as possible. In fact, all of the development on metric index methods can be seen as attempts to simulate the performance of AESA using less memory [5]. There have been many studies with the goal of reducing its preprocessing time or employed storage space. The LAESA [15] chooses M elements from the dataset as potential pivots, and reduces the quadratic storage cost of AESA to $O(MN)$. An improved version of LAESA is the Tree LAESA [16], which achieves sub-linear side computations at query time at the expense of doubling the average number of distance computations. The Reduced Overhead AESA [23] strictly calculates the same number of distances as AESA, but reduces the query processing time. Recently, graph t-spanner indexes [17] were used to simulate AESA, obtaining almost the same number of distance calculations with much less memory. In [9] a new technique called iAESA was introduced to choose the next pivot, which guesses a better close candidate and yields some reductions in the number of distance computations. To improve the search performance of the algorithms, in [10], the author suggests applying the AESA technique to most of the existing metric search structures to efficiently reduce the number of distance computations during the search.

All of the above mentioned methods—in fact most of the available metric search algorithms—try to avoid distance computations by applying the triangular-inequality. Although the triangle-inequality is computationally cheap, its pruning performance degenerates quickly as the dimensions of the data increase. It is interesting to observe that the triangular inequality is defined in a 1D embedding space: all objects are embedded in the space defined by the distance from the pivot—a reference object. All information perpendicular to this dimension is ignored. Assume that the indexed objects can be embedded into a Euclidean space with a finite dimension c. It can be expected that a more accurate approximation of inter-object metric 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 proposed two groups of inequalities to estimate the inter-object distances in the metric space. For easy conceivability, the new approximations were defined in the 2D and 3D embeddings of the metric space. With the same storage cost, the proposed inequalities were experimentally proven to be tighter than the triangle-inequality. In particular, the 3D inequalities were theoretically proven to be tighter than their 1D counterparts. To evaluate the efficiency of the proposed inequalities, we incorporated them into two indexing structures. The first one was AESA and we aimed to estimate a new baseline for similarity search in metric spaces. The second algorithm was the LAESA algorithm, which requires much less storage than AESA with the cost of a few additional distance computations. In [1], the proposed lower bounds were applied to the GNAT indexing structure [2], which is one of the most popular metric trees, with preferable performance reported.

The rest of this paper is organized as follows. In Section 2, we briefly review the problem of metric search, together with the classical AESA and LAESA algorithms and other popular metric indexing structures. Section 3 specifies the proposed inequalities to approximate the inter-object distances in metric spaces. Section 4 specifies the detailed implementation of the search algorithms adopting these inequalities. Some discussions on the implementation and side computations are given in Section 5. Section 6 reports the numerical results of a series of simulations. Section 7 concludes the paper.

2 Related Works

Before specifying the AESA search strategies, we will 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 objects, $d : \mathbb{D} \times \mathbb{D} \rightarrow \mathcal{R}$ 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].

$$d(\boldsymbol{u}, \boldsymbol{v}) \geq 0 \qquad \text{non} - \text{negativity} \qquad (1)$$

$$d(\boldsymbol{u}, \boldsymbol{v}) = 0 \Leftrightarrow \boldsymbol{u} = \boldsymbol{v} \qquad \text{identity} \qquad (2)$$

$$d(\boldsymbol{u}, \boldsymbol{v}) = d(\boldsymbol{v}, \boldsymbol{u}) \qquad \text{symmetry} \qquad (3)$$

$$d(\boldsymbol{u}, \boldsymbol{v}) + d(\boldsymbol{v}, \boldsymbol{z}) \geq d(\boldsymbol{u}, \boldsymbol{z}) \qquad \text{triangular inequality} \qquad (4)$$

A metric query is generally defined by a query object $\boldsymbol{q}$ and a proximity condition. There are two types of queries that are widely used: the range query and nearest neighbor query. The range query can be specified as: Given a metric space $\mathcal{M} = (\mathbb{D}, d)$, a finite set $\mathbb{U} \subset \mathbb{D}$, a query $\boldsymbol{q} \in \mathbb{D}$, and a range $r \in \mathcal{R}$, the result set for a query $Q = (\boldsymbol{q}, r, \mathbb{U})$ is the set

$$\mathcal{Q}_r(\boldsymbol{q}, r, \mathbb{U}) = \{\boldsymbol{u}_i | d(\boldsymbol{q}, \boldsymbol{u}_i) \leq r, \boldsymbol{u}_i \in \mathbb{U}\}. \quad (5)$$

Note that the point query is a special case of a range query with $r = 0$. For a nearest neighbor (NN) search, the algorithm retrieves the closest object to $\boldsymbol{q}$ as the result. The concept can be generalized as searching for k nearest neighbors. Thus the result set of a kNN search $Q = (\boldsymbol{q}, k, \mathbb{U})$ is

$$\mathcal{Q}_k(\boldsymbol{q}, k, \mathbb{U}) = \mathbb{Q} : \{\mathbb{Q} \subseteq \mathbb{U}, |\mathbb{Q}| = k, \\ \forall \boldsymbol{u}_i \in \mathbb{Q}, \boldsymbol{v} \in \mathbb{U} \setminus \mathbb{Q}, d(\boldsymbol{q}, \boldsymbol{u}_i) \leq d(\boldsymbol{q}, \boldsymbol{v})\}. \quad (6)$$

For simplicity, we confine our discussion mainly to the nearest neighbor query, which is the most widely explored metric search type. All of these discussions can be easily extended to a k-nearest neighbor search algorithm by maintaining a list of the k candidates seen so far and using the largest distance among the k candidates to eliminate the far away objects. The experiment section also reports the results of performance evaluations on k-nearest neighbor searches. A discussion of other kinds of queries, including the reverse nearest neighbor query, similarity join, and a combination of the enumerated search types, can be found in [18], [25].

2.2 AESA

Following [21], [22], the key to the use of AESA in performing a nearest neighbor search is the following property directly derived from the triangular inequality in (4).

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

$$d_{1\mathrm{D}}(\boldsymbol{q}, \boldsymbol{u}|\boldsymbol{p}) = |d(\boldsymbol{q}, \boldsymbol{p}) - d(\boldsymbol{u}, \boldsymbol{p})| \leq d(\boldsymbol{q}, \boldsymbol{u}), \quad (7)$$
$$D_{1\mathrm{D}}(\boldsymbol{q}, \boldsymbol{u}|\boldsymbol{p}) = |d(\boldsymbol{q}, \boldsymbol{p}) + d(\boldsymbol{p}, \boldsymbol{u})| \geq d(\boldsymbol{q}, \boldsymbol{u}). \quad (8)$$

If we let $\mathbb{P}$ be the set of pivots 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 object $\boldsymbol{u} \in \mathbb{D}$ is

$$d_{1\mathrm{D}}(\boldsymbol{q}, \boldsymbol{u}|\mathbb{P}) = \max_{\boldsymbol{p}_i \in \mathbb{P}} |d(\boldsymbol{q}, \boldsymbol{p}_i) - d(\boldsymbol{u}, \boldsymbol{p}_i)|. \quad (9)$$

Similarly, the least upper bound is

$$D_{1\mathrm{D}}(\boldsymbol{q}, \boldsymbol{u}|\mathbb{P}) = \min_{\boldsymbol{p}_i \in \mathbb{P}} |d(\boldsymbol{q}, \boldsymbol{p}_i) + d(\boldsymbol{u}, \boldsymbol{p}_i)|. \quad (10)$$

As the inequalities help to prune some of the objects from further consideration, they are generally called pruning rules [11]. To apply the pruning rules, the distances from the object $\boldsymbol{p}$ to the indexed object $\boldsymbol{u}$ and the query object $\boldsymbol{q}$ need to be known beforehand. Hence, object $\boldsymbol{p}$ is called a pivot.

At the preprocessing step, AESA computes all $O(N^2)$ inter-object distances for the N objects in the indexed object set $\mathbb{X}$ and stores them in a matrix. At query time, according to (9), the distance matrix is used to provide distance lower bounds to objects whose distances have not yet been computed, based on the object distances already computed. More specifically, AESA first initializes $\mathbb{P}$ to the empty set, $\mathbb{U} = \mathbb{U}_0 \setminus \mathbb{P}$, 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 selected as the one with the minimal lower bound from $\mathbb{U}$ and its distance from $\boldsymbol{q}$ is computed. The algorithm then updates the lower bounds for the remaining objects in $\mathbb{U}$ and eliminates the ones with distance lower bounds greater than the distance from $\boldsymbol{q}$ to the nearest neighbor candidate $\boldsymbol{v}$. AESA terminates once there are no more objects left in $\mathbb{U}$. The nearest neighbor candidate, $\boldsymbol{v}$, is updated if necessary when the distance computation is invoked.

2.3 LAESA

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

In particular, let $\mathbb{P}$ be the set of selected pivots from $\mathbb{U}_0$, and let $\mathbb{U} = \mathbb{U}_0 \setminus \mathbb{P}$ be the rest of the objects. To implement the approximating and eliminating strategy, LAESA first computes the distances between $\boldsymbol{q}$ and all of the pivots $\boldsymbol{p} \in \mathbb{P}$. It then estimates the distance lower bounds for all the objects in $\mathbb{U}$ by applying (9). This allows objects to be eliminated 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 objects are sequentially compared with $\boldsymbol{q}$ in ascending order of their distance lower bounds. In the case of a distance computation, the nearest neighbor candidate is updated if necessary.

It is well known that pivot selection can drastically affect the performance of LAESA. Between two sets of pivots with the same size, the better chosen pivots can yield greater eliminating ability and thus largely reduce the search time. In [15], the pivot selection algorithm attempts to choose pivots that are as far away from each other as possible. In this way, objects from corners of the space are chosen as pivots. Better eliminating ability is

achieved because the distance densities of corner objects are much flatter than for objects at the center.

2.4 Other Metric Search Algorithms

In general, any successful metric search should pay attention to two key points: the indexing structure used to organize the indexed objects and techniques to avoid unnecessary distance computations.

Besides the mentioned AESA class of algorithms, which seek to reduce the distance computations as much as possible, there have been many proposals for advanced indexing structures that try to hit a balance between the distance computations, side computations, and storage cost. A few of these are the metric tree approaches, such as the Vantage Point tree (VPT) [24], Burhard-Keller tree (BKT) [3], Generalized Hyperplane tree (GHT) [20], Geometric Near-neighbor Access tree (GNAT) [2] and Metric tree (M-tree) [7], and similarity hashing methods, such as the D-index [8] and its decedents. In these algorithms, objects are organized into locally adjacent groups so that the pruning rules can be applied to multiple objects at one time. In this way, side computations can be greatly reduced.

3 Proposed Lower and Upper Bounds

We can analyze the design of metric search algorithms from another point of view. For a metric space model, nothing beyond the inter-object distance can be acquired. In this sense, the distance matrix possesses all the information that we can get from the metric space model. Note that AESA tries to take advantage of all the information from the distance matrix to reduce the distance computations, whereas other algorithms, e.g. the metric tree structures, only make use of some sketched information about the matrix. This may be the reason why it is hard for other indexing structures to beat AESA when the evaluation criterion is the reduction of distance computations.

On the other hand, it might be possible to improve the search efficiency with respect to the pruning techniques. It is interesting to ask whether, with a fixed set of pivots, we can obtain a lower bound on the distance between two objects $\boldsymbol{q}$ and $\boldsymbol{u}$ that is tighter than that produced by (9). 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 discussion to metric spaces.

3.1 Embedding of the Metric Space

In a multi-dimensional Euclidean space, a 2D embedding space can be defined by a triple of non-identical points $\boldsymbol{o}$, $\boldsymbol{p}$ and $\boldsymbol{u}$, as shown in Fig. 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:

$$\begin{aligned}\bar{d}(\boldsymbol{o},\boldsymbol{p};\boldsymbol{u}) &= d(\boldsymbol{o},\boldsymbol{u}') = \cos\theta d(\boldsymbol{o},\boldsymbol{u}) \\ &= \frac{1}{2d(\boldsymbol{o},\boldsymbol{p})}\big(d^2(\boldsymbol{o},\boldsymbol{u}) + d^2(\boldsymbol{o},\boldsymbol{p}) - d^2(\boldsymbol{u},\boldsymbol{p})\big). \end{aligned} \quad (11)$$

We can see that the projected distance $\overline{d}(\boldsymbol{o},\boldsymbol{p};\boldsymbol{u})$ can be computed using only the inter-object distances, thus it might be possible to apply the lower bound derived from (11) to metric space models. Then it is natural to wonder about what circumstance might allow a metric space model to give rise to a configuration of points, $\{\boldsymbol{x}_i\}$, in a Euclidean space, where the associated Euclidean distance $L_2(\boldsymbol{x}_i,\boldsymbol{x}_j) \equiv d(\boldsymbol{u}_i,\boldsymbol{u}_j)$ for all $\boldsymbol{u}_i,\boldsymbol{u}_j \in \mathbb{U}$. We have the following lemma to the answer to this question. However, note that since the projection onto the 2D embedding space requires the assumption of a Euclidean space model, the above discussion cannot be generalized to all metric space problems if no stronger assumption is taken.

Lemma 2 *Let $\mathcal{M} = (\mathbb{D}, d)$ be a metric space, $\mathbb{U} = \{\boldsymbol{u}_i, i = 1,\cdots,N\} \subset \mathbb{D}$. Define the inter-object 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*

$$\boldsymbol{G} = -\frac{1}{2}(\boldsymbol{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T)\boldsymbol{B}(\boldsymbol{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T), \quad (12)$$

where $\mathbf{1} = \{1,1,\cdots,1\}$ 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 that satisfies

$$L_2(\boldsymbol{x}_i,\boldsymbol{x}_j) \equiv d(\boldsymbol{u}_i,\boldsymbol{u}_j),\ (i,j = 1,\cdots,N). \quad (13)$$

Proof: 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 jth points is given by

$$d_{ij}^2 = (\boldsymbol{x}_i - \boldsymbol{x}_j)^T(\boldsymbol{x}_i - \boldsymbol{x}_j) = \boldsymbol{x}_i^T\boldsymbol{x}_i + \boldsymbol{x}_j^T\boldsymbol{x}_j - 2\boldsymbol{x}_i^T\boldsymbol{x}_j. \quad (14)$$

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

$$g_{ij} = \boldsymbol{x}_i^T\boldsymbol{x}_j. \quad (15)$$

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

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

$$\sum_{i=1}^{N} x_{il} = 0,\ l = 1,\cdots,c. \quad (16)$$

From (14) and (16), we have

$$\frac{1}{N}\sum_{i=1}^{N} d_{ij}^2 = \boldsymbol{x}_j^T\boldsymbol{x}_j + \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i^T\boldsymbol{x}_i,$$
$$\frac{1}{N}\sum_{j=1}^{N} d_{ij}^2 = \boldsymbol{x}_i^T\boldsymbol{x}_i + \frac{1}{N}\sum_{j=1}^{N}\boldsymbol{x}_j^T\boldsymbol{x}_j,$$
$$\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}^2 = \frac{2}{N}\sum_{i=1}^{N}\boldsymbol{x}_i^T\boldsymbol{x}_i. \quad (17)$$

Substituting (17) to (14) gives

$$g_{ij} = -\frac{1}{2}\Big(d_{ij}^2 - \frac{1}{N}\sum_{i=1}^{N} d_{ij}^2 - \frac{1}{N}\sum_{j=1}^{N} d_{ij}^2 + \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}^2\Big). \quad (18)$$

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

$$\boldsymbol{G} = \boldsymbol{H}\boldsymbol{A}\boldsymbol{H}, \quad (19)$$

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

$$\boldsymbol{H} = \boldsymbol{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T. \quad (20)$$

Following [14], suppose that the 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 the metric space model and (19), both* $\boldsymbol{A}$ *and* $\boldsymbol{G}$ *are real valued symmetric matrices. Then it is guaranteed that* $\boldsymbol{G}$ *can be diagonalized with its eigenvalues* λ_i $(i = 1, \cdots, c)$ *and their associated eigenvectors* $\boldsymbol{v}_i$ *are real. Thus we have*

$$\boldsymbol{G} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T, \quad (21)$$

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

Equation (21) can also be rewritten as

$$\boldsymbol{G} = \boldsymbol{X}\boldsymbol{X}^T \quad (22)$$

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 i*th and* j*th points of the configuration is given by* $(\boldsymbol{x}_i - \boldsymbol{x}_j)^T(\boldsymbol{x}_i - \boldsymbol{x}_j)$*, and hence*

$$\begin{aligned}(\boldsymbol{x}_i - \boldsymbol{x}_j)^T(\boldsymbol{x}_i - \boldsymbol{x}_j) &= \boldsymbol{x}_i^T\boldsymbol{x}_i + \boldsymbol{x}_j^T\boldsymbol{x}_j - 2\boldsymbol{x}_i^T\boldsymbol{x}_j \\ &= g_{ii} + g_{jj} - g_{ij} \\ &= a_{ii} + a_{jj} - 2a_{ij} \\ &= -2a_{ij} = d_{ij}^2, \end{aligned} \quad (23)$$

by substituting for g_{ij} *using (18). Hence the distance between* $\boldsymbol{x}_i$ *and* $\boldsymbol{x}_j$ *in the Euclidean space equals to the dissimilarity* $\delta(\boldsymbol{u}_i, \boldsymbol{u}_j)$.■

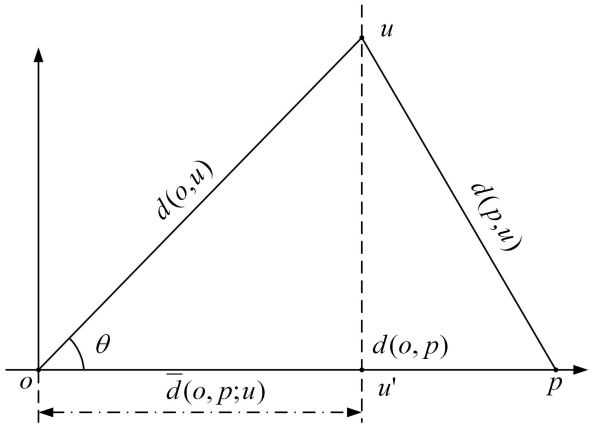

Figure 1: Distances in the 2D Euclidean embedding.

As can be learned from the proof, the configuration of points in the embedding space can be found by a sequence of basic matrix operations. We call a metric space model that can always produce a positive semi-definite gram matrix 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 to most metric search problems. Note that a positive semi-definite metric space is in accordance with positive semi-definite kernels [19], which have recently been extensively studied 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 modern applications.

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 objects when 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:*

$$d_{2\mathrm{D}}(\boldsymbol{p}, \boldsymbol{u}|\boldsymbol{o}, \boldsymbol{p}) = |\bar{d}(\boldsymbol{o}, \boldsymbol{p}; \boldsymbol{q}) - \bar{d}(\boldsymbol{o}, \boldsymbol{p}; \boldsymbol{u})| \leq d(\boldsymbol{q}, \boldsymbol{u}). \quad (24)$$

Proof: Lemma 3 follows directly from the basic properties of the Euclidean space. Because $\overrightarrow{\boldsymbol{q}'\boldsymbol{u}'}$ *is the projection of* $\overrightarrow{\boldsymbol{q}\boldsymbol{u}}$ *along* $\overrightarrow{\boldsymbol{o}\boldsymbol{p}}$*, the equation holds if and only if* $\overrightarrow{\boldsymbol{q}\boldsymbol{u}}$ *parallels* $\overrightarrow{\boldsymbol{o}\boldsymbol{p}}$.■

In Lemma 3, only the projected distance along $\overrightarrow{\boldsymbol{o}\boldsymbol{p}}$ is considered. The following lemma employs the distance perpendicular to $\overrightarrow{\boldsymbol{o}\boldsymbol{p}}$ 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*

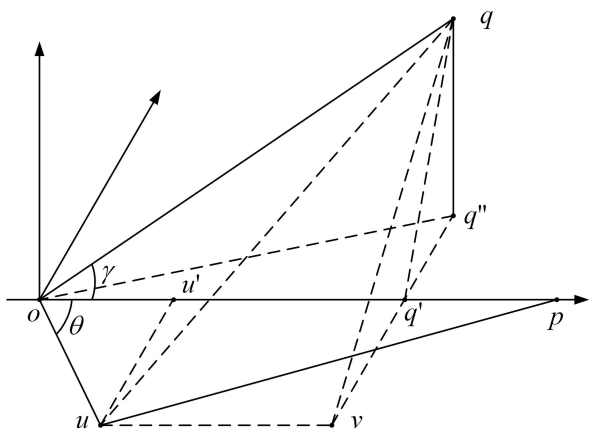

Figure 2: Distances in the 3D Euclidean embedding.

$\overrightarrow{\boldsymbol{op}}$, *respectively. Then the following inequality holds:*

$$\begin{aligned} d(\boldsymbol{q},\boldsymbol{u}) &\geq d_{3\mathrm{D}}(\boldsymbol{p},\boldsymbol{u}|\boldsymbol{o},\boldsymbol{p}) \\ &= \sqrt{d^2(\boldsymbol{u},\boldsymbol{q}|\boldsymbol{o},\boldsymbol{p}) + (d(\boldsymbol{q},\boldsymbol{q}') - d(\boldsymbol{u},\boldsymbol{u}'))^2}. \end{aligned} \quad (25)$$

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

$$\begin{aligned} d(\boldsymbol{q},\boldsymbol{u}) &\leq D_{3\mathrm{D}}(\boldsymbol{p},\boldsymbol{u}|\boldsymbol{o},\boldsymbol{p}) \\ &= \sqrt{d^2(\boldsymbol{u},\boldsymbol{q}|\boldsymbol{o},\boldsymbol{p}) + (d(\boldsymbol{q},\boldsymbol{q}') + d(\boldsymbol{u},\boldsymbol{u}'))^2}. \end{aligned} \quad (26)$$

Proof: In Fig. 2, the objects 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*

$$\bar{d}(\boldsymbol{o},\boldsymbol{p};\boldsymbol{u}) - \bar{d}(\boldsymbol{o},\boldsymbol{p};\boldsymbol{q}) = d(\boldsymbol{u}',\boldsymbol{q}') = d(\boldsymbol{u},\boldsymbol{v}). \quad (27)$$

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

$$d(\boldsymbol{q},\boldsymbol{u}) = \sqrt{d^2(\boldsymbol{u},\boldsymbol{v}) + d^2(\boldsymbol{q},\boldsymbol{v})} \quad (28)$$

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

$$d(\boldsymbol{q},\boldsymbol{v}) \geq |d(\boldsymbol{q},\boldsymbol{q}' - d(\boldsymbol{q}',\boldsymbol{v})|. \quad (29)$$

Substituting (27) and (29) to (28) and replacing $d(\boldsymbol{q}',\boldsymbol{v})$ *with* $d(\boldsymbol{u},\boldsymbol{u}')$ *gives (25). Similarly, we have (26).*■

Generally, when incorporated in a metric search algorithm, the tighter the distance lower bound, the better the pruning performance of the algorithm to avoid unnecessary distance computations. It is obvious that the 3D lower bound is tighter than its 2D counterpart. For positive semi-definite metric spaces, given more than two non-identical 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, with* $\boldsymbol{o},\boldsymbol{p} \in \mathbb{D}, (\boldsymbol{o} \neq \boldsymbol{p})$ *as pivots. The following inequalities hold:*

$$d_{3\mathrm{D}}(\boldsymbol{q},\boldsymbol{u}|\boldsymbol{o},\boldsymbol{p}) \geq \max\big(d_{1\mathrm{D}}(\boldsymbol{q},\boldsymbol{u}|\boldsymbol{o}), d_{1\mathrm{D}}(\boldsymbol{q},\boldsymbol{u}|\boldsymbol{p})\big), \quad (30)$$

$$D_{3\mathrm{D}}(\boldsymbol{q},\boldsymbol{u}|\boldsymbol{o},\boldsymbol{p}) \leq \min\big(D_{1\mathrm{D}}(\boldsymbol{q},\boldsymbol{u}|\boldsymbol{o}), D_{1\mathrm{D}}(\boldsymbol{q},\boldsymbol{u}|\boldsymbol{p})\big). \quad (31)$$

Proof: Here, we use two consequent objects to denote the distance between them. Then, following the denotations in Lemma 4 and Fig. 2, we have

$$\begin{aligned} d^2_{3\mathrm{D}}(\boldsymbol{q},\boldsymbol{u}|\boldsymbol{o},\boldsymbol{p}) &= (\boldsymbol{u}'\boldsymbol{o} - \boldsymbol{q}'\boldsymbol{o})^2 + (\boldsymbol{u}\boldsymbol{u}' - \boldsymbol{q}\boldsymbol{q}')^2 \\ &= \boldsymbol{u}'\boldsymbol{o}^2 - 2\boldsymbol{u}'\boldsymbol{o}\cdot\boldsymbol{q}'\boldsymbol{o}^2 + \boldsymbol{q}'\boldsymbol{o}^2 \\ &\quad +\boldsymbol{u}\boldsymbol{u}'^2 - 2\boldsymbol{u}\boldsymbol{u}'\cdot\boldsymbol{q}\boldsymbol{q}' + \boldsymbol{q}\boldsymbol{q}'^2 \\ &= \boldsymbol{u}'\boldsymbol{o}^2 + \boldsymbol{u}\boldsymbol{u}'^2 + \boldsymbol{q}'\boldsymbol{o}^2 + \boldsymbol{q}\boldsymbol{q}'^2 \\ &\quad -2(\cos\theta\cos\gamma + \sin\theta\sin\gamma)\boldsymbol{u}\boldsymbol{o}\cdot\boldsymbol{q}\boldsymbol{o} \\ &= \boldsymbol{u}\boldsymbol{o}^2 + \boldsymbol{q}\boldsymbol{o}^2 - 2\cos(\theta-\gamma)\boldsymbol{u}\boldsymbol{o}\cdot\boldsymbol{q}\boldsymbol{o} \\ &\geq d^2_{1\mathrm{D}}(\boldsymbol{q},\boldsymbol{u}|\boldsymbol{o}). \end{aligned} \quad (32)$$

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

$$d^2_{3\mathrm{D}}(\boldsymbol{q},\boldsymbol{u}|\boldsymbol{o},\boldsymbol{p}) \geq d^2_{1\mathrm{D}}(\boldsymbol{q},\boldsymbol{u}|\boldsymbol{p}). \quad (33)$$

(30) follows from (32) and (33). Following the same reasoning we have (31). ■

3.3 Extending to Generic Metric Spaces

To incorporate the proposed inequalities into search algorithms in generic metric spaces, consideration should be given to the case where a metric distance leads to a matrix $\boldsymbol{B}$ that is not positive semi-definite. According to [6], the solution for this case is adding a constant to the metric distances in $\boldsymbol{B}$, except for the self-distances $(\boldsymbol{u}_i,\boldsymbol{u}_i)$. The new distance functions $\{d'_{ij}\}$ in the form of $d'_{ij} = d_{ij} + C(\delta^{ij}$, where C is a constant and δ^{ij} the Kronecker delta, makes $\boldsymbol{B}$ positive semi-definite. This has been referred to as the additive constant problem for many years. The smallest value of C that can turn $\boldsymbol{B}$ into a positive semi-definite is $-2\lambda_n$, where λ_n is the smallest eigenvalue of $\boldsymbol{B}$ [13].

4 Search Algorithms

In this section, we specify the search algorithms with the new lower bounds incorporated with the search strategy of AESA and LAESA. We call the new distance bounds a *projection based distance bounds*, and dub the following algorithms PAESA (the Projection-based AESA) and LPAESA.

4.1 PAESA

As specified in Table 1, PAESA basically follows the algorithm of AESA [21], [22]. The major difference between PAESA and AESA is the application of the new

Table 1: The PAESA metric search algorithm.

```
0  Inputs: q, U_0 = {u_i | i = 1, ..., N};
1  P ← ∅; // set of pivots
2  U ← U_0; // set of non-pivots
3  D(u_i) ← 0, for u_i ∈ U; // lower bound
4  d_n ← ∞; // distance to the nearest neighbor
5  while U ≠ ∅
6      p ← arg min_{u_i ∈ U} D(u_i);
7      U ← U \ {p};
8      if d(q, p) < d_n then // distance evaluation
9          d_n ← d(q, p);
10         b ← p; // update the nearest neighbor
11     for u_i ∈ U do // approximation
12         D_low(u_i) ← max_{p_j ∈ P} appro(q, u_i, p, p_j);
13         D(u_i) ← max(D(u_i), D_low(u_i));
14         if D(u_i) ≥ d_n then
15             U ← U \ {u_i}; // elimination
16     P ← P ∪ {p};
17 return b; // Output: the nearest neighbor
```

Table 2: The LPAESA metric search algorithm.

```
0  Inputs: q, U_0 = {u_i | i = 1, ..., N − M},
           P = {p_i | i = 1, ..., M};
1  U ← U_0 \ P; // set of non-pivots
2  d_n ← ∞; // distance to the nearest neighbor
3  for p_i ∈ P do
4      if d(q, p_i) < d_n then // distance evaluation
5          d_n ← d(q, p_i);
6          b ← p_i; // update the nearest neighbor
7  for u_i ∈ U do //approximation
8      D_low(u_i) ← max_{p_j, p_k ∈ P} appro(q, u_i, p_j, p_k);
9      if (D_low(u_i) ≥ d_n) then
10         U ← U \ {u_i}; // elimination
11 U_s ← sort(U, {D_low(u_i)}); // ascending order
12 for u_i ∈ U_s do
13     if (D_low(u_i) ≥ d_n) then
14         break;
15     if d(q, u_i) < d_n then // distance evaluation
16         d_n ← d(q, u_i);
17         b ← u_i; // update the nearest neighbor
18 return b; // Output: the nearest neighbor
```

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 for $d(\boldsymbol{q}, \boldsymbol{u}_i)$ can be computed according to Lemma 3 or Lemma 4. Thus, with M pivots selected, the lower bounds $d_{2\mathrm{D}}(\boldsymbol{q}, \boldsymbol{u}_i)$ or $d_{3\mathrm{D}}(\boldsymbol{q}, \boldsymbol{u}_i)$ for $d(\boldsymbol{q}, \boldsymbol{u}_i)$, are 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. Line 12 of Table 1 applies an inequality to estimate the lower bound of the distance between $\boldsymbol{q}$ and $\boldsymbol{u}_i$. In the case where Lemma 3 is applied to get the lower bound, we term the algorithm 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-object 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 objects are computed and stored beforehand.

Table 2 specifies the search algorithm for 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 object in the set can be the nearest neighbor of the query object. Note that the distance to the current nearest neighbor candidate is updated if necessary when a distance computation is invoked. LPAESA is called LPAESA2D or LPAESA3D according to the employed lower bound.

4.3 Pivot Selection

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

As pointed out in [4], the way pivots are selected affects the search performance of a metric search algorithm. Among the pivot selection heuristics studied in [4], 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 employed an incremental selection algorithm, as specified in Table 3, to select the pivot set. By defining the distance from an object $\boldsymbol{u}$ to a set of objects $\mathbb{P}$ as the minimum from $\boldsymbol{u}$ to $\boldsymbol{p}_i \in \mathbb{P}$, the idea can be easily stated as follows. First initialize the pivot set with a randomly selected object, then sequentially select the next pivot as the most separated object from the pivot set.

5 Analysis of Computing Cost

For applications with computationally intensive metric distances, side computations, other than the distance

Table 3: Incremental pivot selection algorithm.

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

computations, are simply ignored for easy evaluation of the various metric search methods. However, as suggested in [23], the search performance of AESA like algorithms can still be improved by carefully designed search procedures. Although how to minimize the side computations is beyond the scope of this paper, in this subsection, we provide a discussion of the implementation details of the evaluation of the lower bounds.

As already 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. The evaluation of the 1D lower bound is costless: one addition and one $fabs()$ function call are all that 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 objects.

Applying Lemma 3 generally requires more side computations. From (24) 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-object 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:

$$t = d^2(\boldsymbol{u}, \boldsymbol{o}) - d^2(\boldsymbol{u}, \boldsymbol{p}) - d^2(\boldsymbol{q}, \boldsymbol{o}) + d^2(\boldsymbol{q}, \boldsymbol{p}), \tag{34}$$

$$d^2_{2\mathrm{D}}(\boldsymbol{p}, \boldsymbol{u}|\boldsymbol{o}, \boldsymbol{p}) = \frac{0.25}{d^2(\boldsymbol{o}, \boldsymbol{p})} t \cdot t. \tag{35}$$

Hence, to evaluate the 2D lower bound, three additions, two multiplications, one division, and one assignment are invoked.

Things become a little more complicated when computing the squared 3D lower bound. From (32) we have

$$d^2_{3\mathrm{D}}(\boldsymbol{q}, \boldsymbol{u}|\boldsymbol{o}, \boldsymbol{p}) = \boldsymbol{uo}^2 + \boldsymbol{qo}^2 - \overbrace{2\boldsymbol{uo} \cdot \boldsymbol{qo} \cos\theta \cos\gamma}^{T_1} - \underbrace{2\boldsymbol{uo} \cdot \boldsymbol{qo} \sin\theta \sin\gamma}_{T_2}. \tag{36}$$

From the law of cosines, we have

$$\cos\theta = \frac{\boldsymbol{uo}^2 + \boldsymbol{op}^2 - \boldsymbol{up}^2}{2\boldsymbol{uo} \cdot \boldsymbol{op}}, \tag{37}$$

$$\cos\gamma = \frac{\boldsymbol{qo}^2 + \boldsymbol{op}^2 - \boldsymbol{qp}^2}{2\boldsymbol{qo} \cdot \boldsymbol{op}}. \tag{38}$$

For fast evaluation, define two temporary variables

$$t_1 = \boldsymbol{uo}^2 + \boldsymbol{op}^2 - \boldsymbol{up}^2, \tag{39}$$

$$t_2 = \boldsymbol{qo}^2 + \boldsymbol{op}^2 - \boldsymbol{qp}^2. \tag{40}$$

So the term T_1 in (36) can be computed as

$$T_1 = \frac{0.5}{\boldsymbol{op}^2} t_1 \cdot t_2, \tag{41}$$

and term T_2 can be computed as

$$\begin{aligned} T_2 &= 2\boldsymbol{uo} \cdot \boldsymbol{qo} \sqrt{(1 - \cos^2\theta)(1 - \cos^2\gamma)} \\ &= 2\boldsymbol{uo} \cdot \boldsymbol{qo} \sqrt{(1 - \frac{t_1^2}{4\boldsymbol{uo}^2 \cdot \boldsymbol{op}^2})(1 - \frac{t_2^2}{4\boldsymbol{qo}^2 \cdot \boldsymbol{op}^2})} \\ &= \frac{0.5}{\boldsymbol{op}^2} \sqrt{(4\boldsymbol{uo}^2 \cdot \boldsymbol{op}^2 - t_1^2)(4\boldsymbol{qo}^2 \cdot \boldsymbol{op}^2 - t_2^2)}. \end{aligned} \tag{42}$$

Substituting (41) and (42) to (36) gives

$$d^2_{3\mathrm{D}}(\boldsymbol{q}, \boldsymbol{u}|\boldsymbol{o}, \boldsymbol{p}) = \boldsymbol{uo}^2 + \boldsymbol{qo}^2 - \frac{0.5}{\boldsymbol{op}^2} \big(t_1 t_2 + \big((4\boldsymbol{uo}^2 \cdot \boldsymbol{op}^2 - t_1^2)(4\boldsymbol{qo}^2 \cdot \boldsymbol{op}^2 - t_2^2)\big)^{\frac{1}{2}}\big). \tag{43}$$

Hence, to evaluate the 3D lower bound, nine additions, nine multiplications, one division, two assignments, and one $sqrt()$ 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.

6 Experiments

In this section we present the experimental results of the proposed PAESA and LPAESA algorithms. The numerical results of a series of simulations are reported. The performance of the indexing structures was measured by the reduction in the 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 objects and the query objects were independently drawn from uniform distributions in c-dimensional unit hypercubes. The coordinates of the data points were never used directly and only the inter-object distances were employed in the algorithms. We set the number of objects in the datasets to 10,000, and the number of queries to 100. The reported results were averaged over 10 runs.

Before discussing the performance evaluation of the algorithms, we first show the results of the pivot selection algorithms for LAESA, LPAESA2D, and LPAESA3D. Fig.

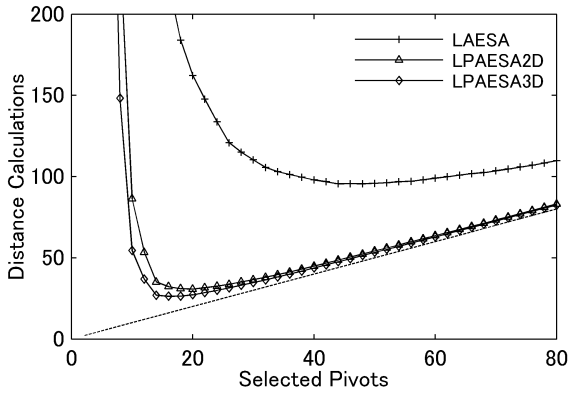

Figure 3: On varying pivot set size in 10D.

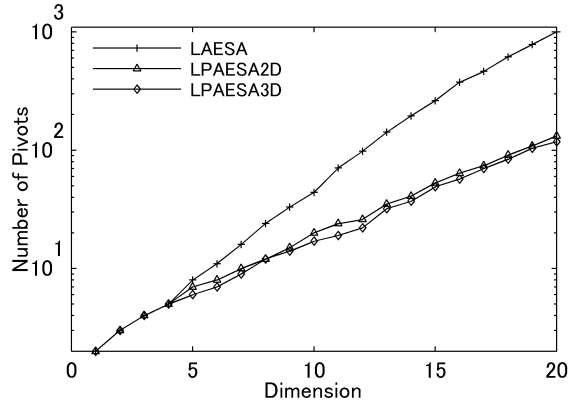

Figure 4: Pivot selection results for different dimensions.

3 shows the curves for the number distance calculations vs. pivot set size for a 10D dataset. Evident saddle points can be found 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 increase considerably. Experiments show that, for independently generated datasets with the same distribution, the optimal number of pivots is quite stable. It can also be seen that the curves for LPAESA2D and LPAESA3D are very close to the dashed line, which stands for the number of pivots. This is because the distance lower bounds given by LPAESA2D and LPAESA3D are very tight, so that most of the non-pivot objects are pruned without distance computations. Fig. 4 reports the results of pivot selection for datasets with up to 20 dimensions. The fig. shows that the number of pivots selected by LAESA was exponential in the dimension of the dataset. LPAESA2D and LPAESA3D needed comparable numbers of pivots for all the cases. For datasets with over 5 dimensions, the LPAESAs selected considerably fewer pivots than LAESA. For 10 dimensions, the three algorithms select 44, 15, and 12 Pivots, respectively, and for 20 dimensions, they selected 1002, 132, and 118 pivots, respectively.

The aim of the second experiment was to compare the performance of the referenced metric search algorithms in different dimensions. The algorithms were tested on datasets with up to 20 dimensions. Fig. 5 shows the distance calculations for the nearest neighbor queries. It can be seen that the search difficulty increases exponentially with the number of dimensions because of the so called curse of dimensionality. The performance comparison is clear enough: for datasets with a lower number of dimensions, since the selected pivots can produce very tight lower bounds, the pivot set based methods are more preferable than the full matrix based methods, especially when the storage cost is considered. When the search difficulty increases as the number of dimensions goes up, sufficiently accurate lower bounds can only be achieved by the extensive use of pre-stored distances. For up to 8 dimensions, we have the following performance order:

$$LAESA \lessdot AESA \lessdot LPAESA2D \lessdot LPAESA3D \lessdot PAESA2D \lessdot PAESA3D, \quad (44)$$

where '$\lessdot$' stands for an ascending order in terms of the saved distance computations. Since lemma 4 produced tighter lower bounds than lemma 3, PAESA3D needed slightly fewer distance computations than PAESA2D, and LPAESA3D needed fewer than LPAESA2D. However, the differences were not very 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—a 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 needed no more than twice the distance computations of PAESAs. For the most intensive case in 20 dimensions, LPAESAs required 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 evaluated the algorithms on 10D datasets with the sample size varying from 1,000 to 13,000. Fig. 6 shows the required distance calculations. It was interesting to find that for all these algorithms fewer objects in the dataset did not mean fewer distance computations were required to retrieve the nearest neighbor. On the contrary, each of the algorithms maintained a stable number of distance computations as the sample size varied. Note that for AESA and the PAESAs, there was 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 (44) was preserved.

In the fourth experiment, we extended the algorithms to find the k nearest neighbors by introducing an ordered list

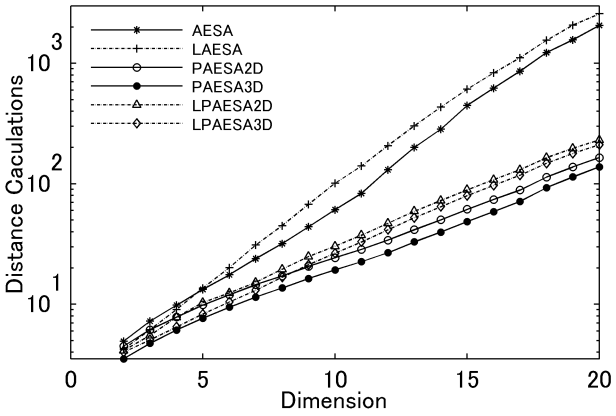

Figure 5: Experiments on varying dimension.

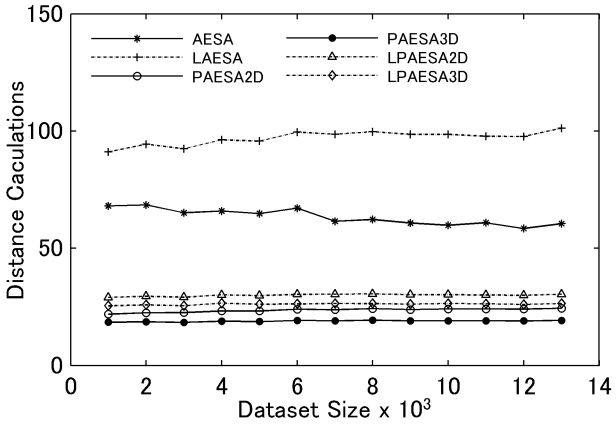

Figure 6: Experiments on varying sample size.

of the k nearest objects to the query object and updating it when the distance computation was invoked. The number of pivots were fixed for LAESA and the LPAESAs. Fig. 7 shows the curves of the distance calculations against k, which changed from 1 to 100. The experiments were done on a 10D dataset with 10,000 objects. We can learn from the figure that as k increased, the search difficulty increased quickly. For each of the algorithms, the number of distance calculations was 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 were fixed for LAESA and LPAESAs in the experiments, the results were in favor of PAESA and AESA. Better results could be produced by LAESA and LPAESAs if more pivots were selected for large k values. A more adaptive strategy could be selecting a large enough pivot set beforehand, and then selectively using a portion of the pivot set according to the user defined k parameter.

From the experiments with different dimensions, sample

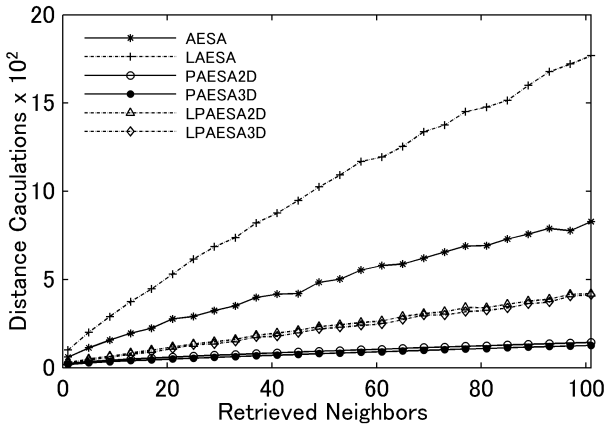

Figure 7: Experiments on varying sample size.

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.

7 Conclusion

In this paper, we proposed two groups of inequalities for approximating the distances in a metric space. The inequalities were derived from the geometrical properties of the embedding space and are applicable to positive semidefinite metric space models. They are also adaptable to generic metric spaces given some appropriate modification is made to the distances. The proposed distance lower bounds were incorporated with the AESA search algorithm. Experiments showed that when the full distance matrix was employed, the search efficiency of the proposed PAESAs was much better than AESA, especially for high dimensional datasets. For better storage efficiency, we also applied the new lower bounds following the idea of LAESA. In these experiments, the proposed LPAESAs not only showed better performance than LAESA in reducing distance computations, but also had lower storage cost in terms of the size of the inter-pivot distance matrix. When the dimensions exceeded 10, the two types of PAESAs both needed fewer than half of AESA's distance computations to retrieve the nearest neighbor. In a 20-dimensional space, the PAESAs only needed about 10% of the distance computations of AESA. The performance comparison between the LPAESAs and LAESA in terms of the distance computations was similar to that between AESA and PAESA.

For most of the experiments, we discovered the performance order shown in (44). Hence, we can reach the conclusion that, for applications where the metric dis-

tance computations are the most essential computational cost, the application of PAESA is the best choice. When the storage cost is considered, the LPAESAs are better choices. Whether to apply the 3D lower bound or 2D lower bound depends on the comparative cost of the side computations and distance evaluations.

References

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

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

[3] W.A. Burkhard and R.M. Keller, "Some approaches to best-match file searching," *Communications of the ACM,* 16(4):230–236, ACM Press, 1973.

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

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

[6] F. Cailliez, "The analytical solution of the additive constant problem," *Psychometrika,* 48:305–308, 1983.

[7] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: an efficient access method for similarity search in metric spaces," in *Proceedings of the 23rd International Conference on Very Large Data Bases*, pp. 426–435, 1997.

[8] V. Dohnal, C. Gennaro, P. Savino, and P. Zezula, "D-Index: distance searching index for metric data sets," *Multimedia Tools and Applications,* 21(1):9–33, 2003.

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

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

[11] L. Fukunaga and P.M. Narendra, "A branch and bound algorithm for computing k-nearest neighbors," *IEEE Transactions on Computers,* 24(7):750–753, 1975.

[12] H.R. Gisli, and H. Samet, "Index-driven similarity search in metric spaces," *ACM Transactions on Database Systems,* 28:517–580, 2003.

[13] J.C. Lingoes, "Some boundary conditions for a monotone analysis of symmetric matrices," *Psychometrika,* 36:406–407, 1971.

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

[15] M.L. Micó, J. Oncina, and E. Vidal, "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.

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

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

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

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

[20] J.K. Uhlmann, "Satisfying general proximity/similarity queries with metric trees," *Information Processing Letters,* 40(4):175–179, 1991.

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

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

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

[24] P.N. Yianilos, "Data structures and algorithms for nearest neighbor search in general metric spaces," *ACM-SIAM Symposium on Discrete Algorithms,* pp. 311–321, 1993.

[25] P. Zezula, G. Amato, V. Dohnal, and M. Batko, *Similarity Search—The Metric Space Approach,* Springer, 2006.