ABSTRACT

DESIGN AND ANALYSIS OF ALGORITHMS FOR SIMILARITY SEARCH BASED ON INTRINSIC DIMENSION

by
Xiguo Ma

One of the most fundamental operations employed in data mining tasks such as classification, cluster analysis, and anomaly detection, is that of similarity search. It has been used in numerous fields of application such as multimedia, information retrieval, recommender systems and pattern recognition. Specifically, a similarity query aims to retrieve from the database the most similar objects to a query object, where the underlying similarity measure is usually expressed as a distance function.

The cost of processing similarity queries has been typically assessed in terms of the representational dimension of the data involved, that is, the number of features used to represent individual data objects. It is generally the case that high representational dimension would result in a significant increase in the processing cost of similarity queries. This relation is often attributed to an amalgamation of phenomena, collectively referred to as the *curse of dimensionality*. However, the observed effects of dimensionality in practice may not be as severe as expected. This has led to the development of models quantifying the complexity of data in terms of some measure of the *intrinsic dimensionality*.

The *generalized expansion dimension* (GED) is one of such models, which estimates the intrinsic dimension in the vicinity of a query point q through the observation of the ranks and distances of pairs of neighbors with respect to q. This dissertation is mainly concerned with the design and analysis of search algorithms, based on the GED model. In particular, three variants of similarity search problem are considered, including adaptive similarity search, flexible aggregate similarity search, and subspace similarity search. The good practical performance of the proposed algorithms demonstrates the effectiveness of dimensionality-driven design of search algorithms.

DESIGN AND ANALYSIS OF ALGORITHMS FOR SIMILARITY SEARCH BASED ON INTRINSIC DIMENSION

by
Xiguo Ma

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Science

Department of Computer Science

January 2015

APPROVAL PAGE

DESIGN AND ANALYSIS OF ALGORITHMS FOR SIMILARITY SEARCH BASED ON INTRINSIC DIMENSION

Xiguo Ma

Dr. Vincent Oria, Dissertation Co-advisor Associate Professor, Department of Computer Science, NJIT	Date
Dr. Michael E. Houle, Dissertation Co-advisor Visiting Professor, National Institute of Informatics, Japan	Date
Dr. Alexandros V. Gerbessiotis, Committee Member Associate Professor, Department of Computer Science, NJIT	Date
Dr. Dimitri Theodoratos, Committee Member Associate Professor, Department of Computer Science, NJIT	Date
Dr. Yi Chen, Committee Member Associate Professor, School of Management, NJIT	Date
Dr. Philip R. Korn, Committee Member Researcher, Google	Date

BIOGRAPHICAL SKETCH

Author:	Xiguo Ma
Degree:	Doctor of Philosophy
Date:	January 2015

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2015
- Bachelor of Engineering in Software Engineering,
  Beihang University, Beijing, China, 2007

Major: Computer Science

Presentations and Publications:

Houle, M. E., Ma, X., Oria, V., and Sun, J. Efficient similarity search in axis-aligned subspaces. In preparation for *Information Systems Journal*.

Houle, M. E., Ma, X., and Oria, V. Effective and efficient algorithms for flexible aggregate similarity search in high dimensional spaces. Submitted to *Transactions on Knowledge and Data Engineering*.

Houle, M. E., Ma, X., Oria, V., and Sun, J. (2014). Efficient algorithms for similarity search in axis-aligned subspaces. In *International Conference on Similarity Search and Applications*, pages 1–12. (Best Paper Award).

Houle, M. E., Ma, X., Oria, V., and Sun, J. (2014). Improving the quality of K-NN graphs through vector sparsification: application to image databases. *International Journal on Multimedia Information Retrieval*, 3(4):259–274.

Houle, M. E., Ma, X., Oria, V., and Sun, J. (2014). Improving the quality of K-NN graphs for image databases through vector sparsification. In *International Conference on Multimedia Retrieval*, pages 89–96.

Houle, M. E., Ma, X., Nett, M., and Oria, V. (2012). Dimensional testing for multi-step similarity search. In *International Conference on Data Mining*, pages 299–308.

To My Beloved Parents.

ACKNOWLEDGMENT

I would like to express my deepest gratitude to my co-advisors, Dr. Vincent Oria and Dr. Michael E. Houle, not only for their countless helpful comments and advice, but also for their excellent caring and patience. I am also thankful for having Dr. Alexandros V. Gerbessiotis, Dr. Dimitri Theodoratos, Dr. Yi Chen and Dr. Philip R. Korn as members of the committee of my doctoral dissertation. I am very grateful for their time and commitment, as well as the comments provided.

I am also thankful to the Department of Computer Science at New Jersey Institute of Technology for its great financial support during my PhD study. I would also like to thank the National Institute of Informatics at Japan for providing me the research internship opportunities.

I furthermore want to express my gratitude to my friends and coworkers for their help and support: Cem Aksoy, Shuo Chen, Ananya Dass, Yulin Huang, Bing Li, Michael Nett, Sheetal N. Rajgure, Jichao Sun, Niya Su, Wei Wang, Ying Xu, Xiangqian Yu, and Shuangyi Zhang. I am especially thankful to my girlfriend Shan Gao for her endless love and support. I am very lucky to have her in my life.

Last but not least, I am deeply grateful to my parents Xiuzhen Ma and Wenlan Zhao for making me who I am, and my brothers and sister Yizhou Ma, Suping Ma and Xibao Ma for their endless support. Without them, I would have never been able to finish this journey.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter Page

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

Figure Page

LIST OF FIGURES
(Continued)

CHAPTER 1

INTRODUCTION

One of the most fundamental operations employed in data mining tasks such as classification, cluster analysis, and anomaly detection, is that of similarity search. Similarity search is the basis of k-nearest-neighbor (k-NN) classification, which often produces the lowest error rates in practice, particularly when the number of classes is large [Han and Kamber, 2006]. For clustering, many of the most popular strategies require the determination of neighbor sets based at a large proportion of the data set objects [Han and Kamber, 2006]. Content-based filtering methods for recommender systems [Sarwar et al., 2000] and anomaly detection methods [Chandola et al., 2009] commonly make use of k-NN techniques, either through the direct use of k-NN search, or by means of k-NN cluster analysis. A popular density-based measure, the Local Outlier Factor (LOF), relies heavily on k-NN computation to determine the relative density of the data in the vicinity of the test point [Breunig et al., 2000]. Due to its importance and generality, the similarity search problem has been the subject of intensive research for many decades.

For a given data domain $\mathbb{U}$, a similarity query typically aims to retrieve those objects from a dataset that are most similar to a given query object. To support similarity queries, data objects are often represented by D-dimensional feature vectors with the measure of similarity usually defined as a distance function $d : \mathbb{U} \times \mathbb{U} \mapsto \mathbb{R}_0^+$. A large distance value indicates the dissimilarity of two objects, while a distance value of 0 indicates perfect similarity.

Similarity queries are of two main types: range queries and k-NN queries. Given a query object $q \in \mathbb{U}$, a dataset $S \subseteq \mathbb{U}$, and a distance function $d : \mathbb{U} \times \mathbb{U} \mapsto \mathbb{R}_0^+$, range queries and k-NN queries can be characterized as follows:

- range queries report the set $\{v \in S \mid d(q,v) \leq \epsilon\}$ for some real value $\epsilon \geq 0$.

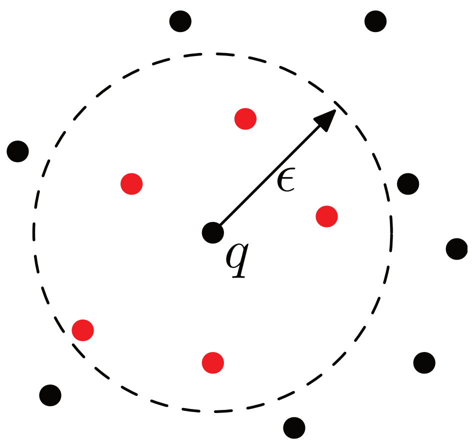
- k-NN queries report the set $K \subseteq S$ of size k objects satisfying $d(q,u) \leq d(q,v)$ for all $u \in K$ and $v \in S \setminus K$.

An illustration of the two types of similarity queries is shown in Figure 1.1.

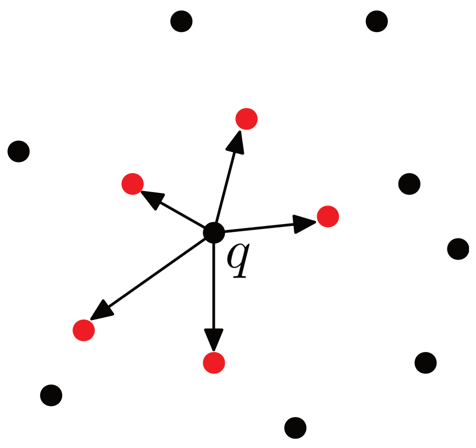
Most solutions for the similarity search problem seek to reduce the search space directly by means of an index structure. A very wide variety of data structures have been developed over the last few decades [Samet, 2006], such as multidimensional indexes and distance-based indexes. Multidimensional indexes make use of explicit knowledge of the data representation for their organizations. Examples include R-Trees [Guttman, 1984], R^+-Trees [Sellis et al., 1987], R^*-Trees [Beckmann et al., 1990], X-Trees [Berchtold et al., 1996], SS-Trees [White and Jain, 1996], SR-Trees [Katayama and Satoh, 1997] and A-Trees [Sakurai et al., 2002]. Distance-based indexes do not place any assumptions on the representation of data objects and rely solely on pairwise distance values between objects to organize the data. Examples include VP-Trees (*vantage point trees*) [Yianilos, 1993], M-Trees (*metric trees*) [Ciaccia et al., 1997], GNATs (*geometric near-neighbor access trees*) [Brin, 1995], GH-Trees (*generalized hyperplane trees*) [Uhlmann, 1991], and Cover Trees [Beygelzimer et al., 2006].

Effective solutions for the similarity search problem should possess great scalability, especially for modern applications. Nowadays, all forms of data such as text, images, market data, biological data, and scientific data are being accumulated in large data repositories at an amazingly high rate. Together with this data explosion, the demand for effective and scalable search methods continues to grow.

In general, the performance of search methods is closely related to the representational dimension of the data involved, that is, the number of features used to represent individual data objects. Numerous observations such as the ones presented in [Chávez et al., 2001; Volnyansky and Pestov, 2009; Pestov, 2000] show that as the representational dimension increases, the search performance would degrade quickly. This trend is widely



(a) range query



(b) k-NN query with $k = 5$

Figure 1.1 An illustration of range query and k-NN query.

believed to be caused by an amalgamation of phenomena usually referred to as the *curse of dimensionality*.

One of the frequently observed phenomenon with high dimensional data is the so-called *concentration of measure*: the measure of a space concentrates at the border of any sufficiently large region [Gromov and Milman, 1983]. As a result, the expansion of a region may suddenly capture nearly all measure of the space. Another frequently observed phenomenon is the dimensionality effect on similarity distances. In [Beyer et al., 1999], Beyer et al. found that pairwise distances between data objects would concentrate around their mean value as the dimensionality rises. Specifically, they observed that the proportional difference between the minimum and maximum distance, $(d_{\max} - d_{\min})/d_{\max}$, diminishes with the increase of dimensionality.

Nevertheless, the observed effects of dimensionality in practice may not be as severe as expected. For example, Beyer et al. showed that data following a mixture of (well-separated) distributions is often less affected by the curse of dimensionality [Beyer et al., 1999]. In addition, Houle et al. illustrated that increasing the number of features may improve the performance instead of causing performance degradation, as long as the added features have a sufficiently high signal-to-noise ratio [Houle et al., 2010; Bernecker et al., 2011]. Such observations have led to the development of alternative models quantifying the complexity of data in terms of some measure of the *intrinsic dimensionality*.

In the literature, many measures of the intrinsic dimensionality of data have been developed. Topological methods try to estimate the basis dimension of the tangent space of the data manifold based on local samples [Fukunaga and Olsen, 1971; Bruske and Sommer, 1998; Pettis et al., 1979; Verveer and Duin, 1995]. Projection methods search for a subspace to project the data with minimum error; the dimension of the resulting subspace can then be used as an estimate of the intrinsic dimension. Examples of projection methods include *principal component analysis* (PCA) [Fukunaga, 1990], manifold learning [Schlkopf et al., 1998; Roweis and Saul, 2000; Venna and Kaski, 2006] and other non-linear extensions [Karhunen and Joutsensalo, 1994]. Fractal approaches aim to provide an estimate of the intrinsic dimension in terms of the space-filling capacity of the data [Faloutsos and Kamel, 1994; Camastra and Vinciarelli, 2002; Gupta et al., 2003].

The *expansion dimension* [Karger and Ruhl, 2002] and the *generalized expansion dimension* (GED) [Houle et al., 2012a] allow the estimation of intrinsic dimension to be made through the observation of the rate at which the number of captured data objects grows as the considered range of distances expands. The *continuous intrinsic dimensionality* (CID) [Houle, 2013] models the intrinsic dimension in terms of the distribution of neighborhood measure.

This dissertation is mainly concerned with the design and analysis of algorithms for several variants of the similarity search problem, based on the GED model. In particular, three variants are considered, including adaptive similarity search, flexible aggregate similarity search, and subspace similarity search.

Adaptive Similarity Search. Adaptive similarity search aims to find the most similar objects to a query object q from the database S with respect to an adaptive similarity measure — one that can be determined by the user at query time. Adaptive similarity search can be useful in application areas where the underlying similarity measure may change very often, such as feature selection [Kohavi and John, 1997], subspace clustering [Kriegel et al., 2012], content-based image retrieval [Seidl and Kriegel, 1997] and document clustering [Kim and Lee, 2002].

In general, traditional similarity search methods are not suitable for handling adaptive similarity queries, since they require that the similarity measures used to rank query results be fixed. Changes in the underlying similarity measure may require the reconstruction of the search index, which is not practical during query time. In the literature, so-called 'multi-step' search strategy has been proposed for the adaptive similarity search problem that can accommodate changes in the underlying similarity measure without the need to rebuild the index [Korn et al., 1996]. The basic idea of the strategy is that a query result computed using a fixed 'lower-bounding' distance function can be adapted to answer the same query with respect to a user-supplied 'target' distance function.

Existing multi-step search algorithms [Korn et al., 1996; Seidl and Kriegel, 1998; Kriegel et al., 2007] all produce exact query results; however, they may be prohibitively expensive. Motivated by this observation, we design an approximate multi-step search algorithm by adopting an early termination condition based on tests of intrinsic dimensionality. GED-related conditions under which the algorithm is guaranteed to produce correct query results can be derived. Compared to state-of-the-art approaches, our algorithm is able to achieve better performance, which demonstrates the effectiveness of dimensionality-driven design of search algorithms.

Flexible Aggregate Similarity Search. The flexible aggregate similarity search (FANN) problem is a generalization of the aggregate similarity search problem (also referred to as aggregate nearest neighbor (AggNN) search). Given a group of query objects Q, the goal of AggNN is to retrieve the k objects from the database S that are most similar to Q, where the underlying similarity measure is defined as an aggregation (usually *sum* or *max*) of distances between the retrieved objects and every query object in Q.

AggNN can find many applications in different areas, such as spatial databases [Papadias et al., 2004; Papadias et al., 2005b; Razente et al., 2008a; Li et al., 2011a], multimedia databases [Razente et al., 2008b], and clustering [Razente et al., 2008a; Papadias et al., 2004]. In general, however, AggNN tends to favor only those objects that are similar to all query objects in Q, which in practice may greatly limit its performance when the objects of Q are greatly dispersed [Li et al., 2011b]. For such scenarios, FANN has been proposed [Li et al., 2011b], in which the restrictiveness of AggNN is eased by calculating aggregate distances only over subsets of Q.

Existing solutions for the AggNN problem cannot be effectively applied for FANN. Straightforward adaptation of these methods would require the checking of an exponential number of subsets of Q. Two exact solutions, R-tree and List, were proposed in [Li et al., 2011b] for the FANN problem. In general, Algorithm R-tree can only be applied in low-dimensional settings (up to 10 dimensions); while Algorithm List may incur high computational costs. In an attempt to lower the computational cost of processing FANN queries, two approximation methods, ASUM and AMAX, were also proposed by Li et al. [Li et al., 2011b]. ASUM and AMAX can be very efficient; however, they may fail to return query results with good quality.

We propose approximation algorithms for the FANN problem in high dimensional spaces, that possess great efficiency while returning query results with great quality. The idea is to simultaneously explore the search space from multiple locations, while applying two pruning strategies. The first strategy utilizes a distance bound based on an auxiliary distance relative to the aggregate distance. The second strategy applies a test of the intrinsic dimensionality (in the vicinity of Q) to determine whether early termination is possible. As for the multi-step search algorithm mentioned above, we are able to derive GED-related conditions under which the two algorithms are guaranteed to provide correct answers. The good practical performance obtained by our proposed algorithms for the FANN problem again shows the important role of dimensionality in the design of search methods.

Subspace Similarity Search. Subspace similarity search aims to retrieve from the database S the most similar objects to a query object q with the similarity distance function being defined over a subset of features (an axis-aligned projective subspace). In particular, each query may specify an arbitrary subspace with an arbitrary number of features. The efficient support of subspace similarity search could benefit application areas where the feature set under consideration changes from operation to operation, such as content-based image retrieval, subspace clustering [Kriegel et al., 2012] and feature selection [Kohavi and John, 1997].

Almost all existing similarity search methods are designed for fixed spaces. Traditional methods for fixed spaces usually cannot be effectively applied for the subspace search problem, since it would be prohibitively expensive to explicitly preprocess the data for every possible subspace. Of all the methods for similarity search appearing in the research literature, only very few have been specifically formulated for the subspace search problem [Kriegel et al., 2006; Bernecker et al., 2010b; Bernecker et al., 2010a]. These methods, however, may suffer greatly in terms of the computational cost.

In our work on subspace similarity search, we aim to design solutions that possess great efficiency. Subspace similarity search can be viewed as a special case of adaptive similarity search, since different subspaces specified by the user at query time would define different similarity measures. Therefore, to solve the subspace similarity search problem, we may apply the multi-step search algorithms designed for adaptive similarity search. The main concern here is the determination at query time of a lower-bounding distance function suitable for the indicated subspace. We consider all the 1-dimensional distance functions associated with each individual features as potential lower-bounding distance functions, and for a specified subspace, we select the ‘best’ one among possible choices to facilitate the search. The experimental results demonstrate that our proposed algorithms are able to achieve very competitive performance relative to state-of-the-art solutions.

The remainder of this dissertation is organized as follows. In Chapter 2, existing similarity search methods and the GED model are discussed. The works on adaptive similarity search, flexible aggregate similarity search and subspace similarity search are presented in Chapter 3, 3.4 and 4.6, respectively. Finally, the dissertation is concluded in Chapter 6.

CHAPTER 2

RELATED WORK

In this chapter, the work related to this dissertation is discussed. Section 2.1 surveys the research literature on similarity search methods. Section 2.2 introduces one of the intrinsic dimension models, the *generalized expansion dimension* [Houle et al., 2012a], which serves as the basis of design and analysis of our search algorithms.

2.1 Overview of Similarity Search Methods

In this section, we present an overview of existing similarity search methods, including traditional similarity search methods, adaptive similarity search methods, aggregate similarity search methods, and subspace similarity search methods.

2.1.1 Traditional Similarity Search Methods

In general, traditional similarity search methods can be categorized into four classes: multidimensional indexing methods, distance-based indexing methods, dimension reduction methods, and hashing-based indexing methods. Though the class of hashing-based methods can be discussed in the context of dimension reduction techniques, it is often treated as a separate class due to its popularity. We next discuss these four classes one by one.

Multidimensional Indexing Methods. Most multidimensional indexing methods can be categorized as spatial access methods, which are the most widely known index structures for similarity search. The basic idea underlying spatial access methods is to support efficient selection of objects based on spatial properties. Specifically, they organize the data into a hierarchy of groups by imposing explicitly represented geometric shapes, such as hyper-planes, spheres, and rectangles (see Figure 2.1). Examples of spatial access methods include R-Trees [Guttman, 1984], R$^+$-Trees [Sellis et al., 1987],

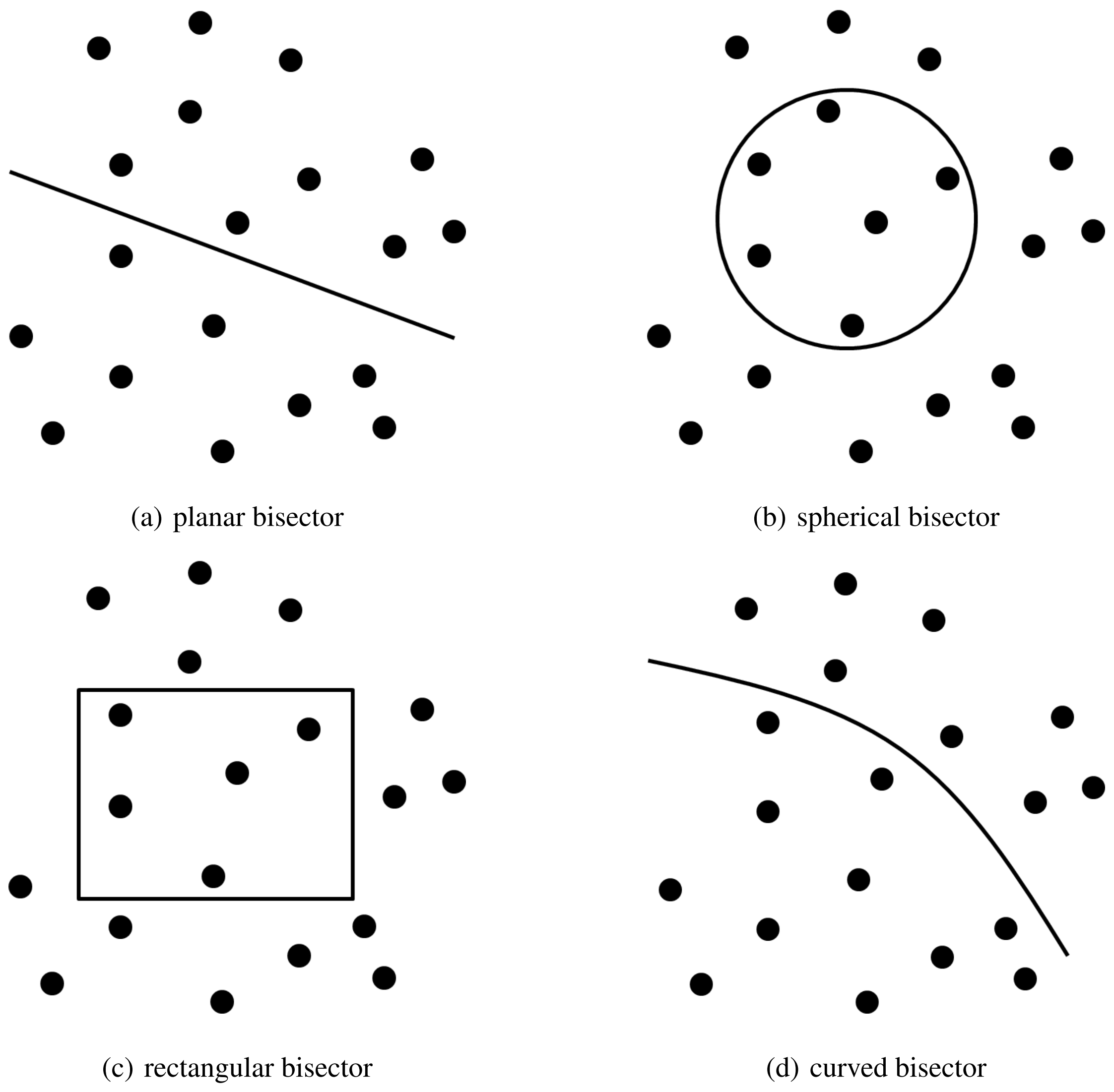

(a) planar bisector (b) spherical bisector (c) rectangular bisector (d) curved bisector

Figure 2.1 An illustration of different geometric shapes that are used as bisectors in the organization of data.

R*-Trees [Beckmann et al., 1990], X-Trees [Berchtold et al., 1996], SS-Trees [White and Jain, 1996], SR-Trees [Katayama and Satoh, 1997] and A-Trees [Sakurai et al., 2002].

The R-Tree can be viewed as a precursor of other spatial access methods. In the R-Tree, all the data objects are stored in the leaf nodes, while an entry in a non-leaf node stores two things: a pointer to a node at the next lower level, and the minimum bounding rectangle of all the objects in the subtree being pointed at. These minimum bounding rectangles are then utilized during query processing so as to prune subtrees that are irrelevant to the query.

Spatial access methods perform very well in low-dimensional settings; however, they may completely fail in high-dimensional spaces due to certain undesirable properties of the data organization process. For example, in high-dimensional spaces, an overwhelmingly large proportion of the measure contained within the hyper-rectangles utilized in the organization of R-Trees is located in the proximity of their boundaries. As a result, a randomly located query point is very likely to be closer to a large number of hyper-rectangles than to its nearest neighbor, and therefore, the search needs to be done by accessing a large number of memory pages associated with individual hyper-rectangles. Besides, as the dimensionality increases, the proportion of measure contained in overlapping regions of hyper-rectangles rises, making it difficult for R-Trees to exclude search paths. These issues greatly impair the ability of R-Trees or other spatial access methods in handling high-dimensional queries. Evidence shows that as the representational dimension approaches 20, the performance of such search structures degrades to that of a sequential scan of the entire dataset or even worse [Beyer et al., 1999].

The key advantage of sequential scan over spatial access methods is that data can be scanned sequentially from disk instead of at random. Random disk accesses are typically more expensive than sequential accesses due to the disk seek overhead. This has led to the investigation of methods based on the idea of sequential scan. The vector approximation file (VA-file) [Weber et al., 1998] is one of such methods. The key idea is to obtain a compressed approximation of the data. At query time, the compressed approximation is scanned in its entirety for pruning the search within the original dataset. Empirical studies show that the VA-file is able to render better performance than the R^*-Tree, X-Tree, and the sequential scan method on datasets with dimensions ranging as high as 45 [Weber et al., 1998].

The design of the VA-file relies on the assumption that the data features are independent. To address the situation where the assumption does not hold, Ferhatosmanoglu et al. [Ferhatosmanoglu et al., 2000] developed the VA$^+$-file and showed that the VA$^+$-file can outperform the VA-file on a number of real datasets.

Distance-Based Indexing Methods. One of the major short-comings of the previously introduced multidimensional indexing methods in terms of their generality is that they assume that data objects can be represented as finite-dimensional features. This assumption might not hold in many applications [Ciaccia et al., 1997; Agrawal et al., 1993; Hellerstein et al., 1995; Faloutsos et al., 1994]. In order to handle similarity queries in such cases, distance-based indexing methods have been developed, which do not place any assumptions on the representation of data objects and perform the search purely based on pairwise distances.

Typical of distance-based indexing structures are metric indexes, which require the underlying distance measure to be a distance metric. Let $d : \mathbb{U} \times \mathbb{U} \mapsto \mathbb{R}$ be a real-valued function operating on some data domain $\mathbb{U}$. The function d is a metric if and only if it satisfies the following conditions:

- For any $u, v \in \mathbb{U}$, $d(u,v) \geq 0$.
- For any $u, v \in \mathbb{U}$, $d(u,v) = 0$ if and only if $u = v$.
- The function d is symmetric, that is, $d(u,v) = d(v,u)$ for any $u, v \in \mathbb{U}$.
- The function d satisfies the triangle inequality $d(u,v) \leq d(u,w) + d(w,v)$ for any $u, v, w \in \mathbb{U}$.

Of the distance metric conditions, the triangle inequality is the most important one for metric indexes in pruning the search space when processing queries. Suppose we have a query object q and two other objects u and v, we can bound the distance $d(q,u)$ from both below and above using $d(q,v)$ and $d(v,u)$ based on the triangle inequality. Specifically, we have

$$|d(q,v) - d(v,u)| \leq d(q,u) \leq d(q,v) + d(v,u),$$

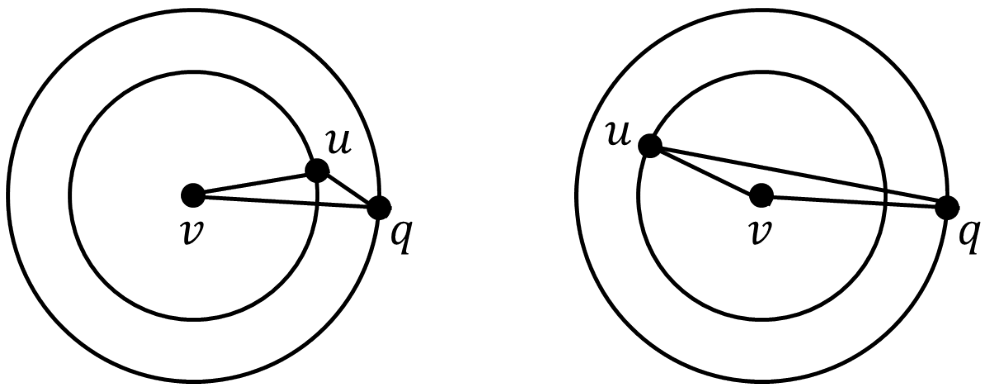


Figure 2.2 An illustration of distance bounds on $d(q, u)$. The left figure illustrates the situation where the lower bound $|d(q, v) - d(v, u)|$ is nearly attained. Similarly, the right figure illustrates the situation where the upper bound $d(q, v) + d(v, u)$ is nearly attained.

as illustrated in Figure 2.2. These bounds can then be used for pruning the search space. For example, when processing k-NN queries, if the lower bound on $d(q, u)$ is no less than the candidate k-NN distance, then object u can be safely eliminated from consideration as a candidate for the query result.

Popular examples of metric indexes include VP-Trees (*vantage point trees*) [Yianilos, 1993], M-Trees (*metric trees*) [Ciaccia et al., 1997], GNATs (*geometric near-neighbor access trees*) [Brin, 1995], GH-Trees (*generalized hyperplane trees*) [Uhlmann, 1991], and Cover Trees [Beygelzimer et al., 2006]. The M-Tree can be considered as a conceptual precursor of other metric indexes. In the M-Tree, all the data objects are stored in the leaf nodes, while an entry in a non-leaf node stores a pointer to a node at the next lower level, and a *routing object* together with some precomputed distance values, such as the distance to its parent and the distance to the farthest object in its corresponding subtree. These precomputed distance values are then utilized during query processing so as to exclude irrelevant subtrees from consideration.

A major drawback for metric indexes is that they make heavy use of the triangle inequality for pruning the search space. As discussed in Chapter 1, pairwise distances between data objects would concentrate around their mean value as the dimensionality rises. Consequently, the triangle inequality may cease to provide an effective means for pruning the search space while processing high-dimensional queries [Zezula et al., 2005]. This had led to the investigation of search methods without utilizing the triangle inequality for pruning, such as the *spatial approximation sample hierarchy* (SASH) [Houle and Sakuma, 2005] and the *rank cover tree* (RCT) [Houle and Nett, 2013; Houle and Nett, 2014].

The RCT is built upon the idea of the SASH. A SASH is a multi-level structure recursively constructed by building a SASH on a large random data sample, and then connecting the remaining objects to several of their approximate nearest neighbors from within the samples. The search is done by first locating approximate neighbors within the samples, and then using the pre-established connections to discover neighbors within the remaining objects.

Both the SASH and the RCT are able to process similarity queries very efficiently, even for extremely high-dimensional datasets (in the hundreds of thousands or even millions of dimensions). Nevertheless, such efficiency is achieved at the expense of query accuracy, though not much. In other words, both the SASH and the RCT are designed to answer approximate similarity queries, which are applicable for situations where obtaining exact results is not critical.

Dimension Reduction Methods. Trying to avoid the undesirable effects of high dimensionality, researchers have designed search methods based on the idea of explicit dimensionality reduction. For example, if a dataset originally embedded in a high-dimensional space can be somehow represented by only one dimension, then the dataset could be indexed by structures like the B-Tree, thereby enabling it to be searched easily. Of course, this comes at the expense of query accuracy unless the original dataset is used to refine the query result.

In general, dimension reduction methods rely on some transformation techniques that are able to reduce the dimensionality of a dataset without losing much information. Examples of such techniques include *singular value decomposition* (SVD) [Golub and

Van Loan, 1996], *principal component analysis* (PCA) [Fukunaga, 1990], and *discrete Fourier transform* (DFT) [Gershenfeld, 1999]. Even though these techniques have been applied for similarity search [Kanth et al., 1999; Wu et al., 1996; Agrawal et al., 1993], they are often being considered unattractive due to their high computational complexity.

The iDistance algorithm (for exact queries) [Jagadish et al., 2005] and the MedScore algorithm (for approximate queries) [Andoni et al., 2008], on the other hand, provide computationally inexpensive dimensionality reduction-based indexing. The idea underlying the iDistance method is to first select a number of reference objects, and then map data objects to one-dimensional values based on their distances to the nearest reference objects. These one-dimensional values are then indexed using a B^+-Tree to facilitate the search. The iDistance algorithm is susceptible to the concentration effect on distances, since the distances from data objects to their nearest reference objects will also concentrate around some mean value, making it difficult to prune the search space.

In the MedScore method, the original high-dimensional feature space is transformed into a low-dimensional feature space by means of random projections. For each dimension in the new feature space, a sorted list is then created. The search is done by aggregating precomputed sorted lists to find the objects with the minimum median scores. The well-known *threshold algorithm* [Fagin et al., 2001] is used here for the aggregation. The MedScore method may not work well in high-dimensional spaces, since the number of sorted lists that have to be created tends to be large in order to guarantee the query results with good quality. Consequently, the aggregation process may be expensive.

Hashing-Based Indexing Methods. The key concept for hashing-based indexing methods is the so-called *locality sensitive hashing* [Indyk and Motwani, 1998]. The basic idea is to hash data objects so that similar ones are mapped to the same buckets with high probability, and dissimilar ones are mapped to different buckets with high probability. Let d be a distance function operating on some data domain $\mathbb{U}$. Let $H = \{h : \mathbb{U} \mapsto X\}$ be a family of hash functions which map data objects from $\mathbb{U}$ to buckets $x \in X$. The family of hash functions H is as (r_1, r_2, p_1, p_2)-sensitive for d, if any function h drawn uniformly at random from H satisfies the following two conditions for any $u, v \in \mathbb{U}$ [Indyk and Motwani, 1998]:

- if $d(u,v) \leq r_1$, then $\Pr[h(u) = h(v)] \geq p_1$,
- if $d(u,v) \geq r_2$, then $\Pr[h(u) = h(v)] \leq p_2$.

A (r_1, r_2, p_1, p_2)-sensitive family is interesting when $r_1 < r_2$ and $p_1 > p_2$. Basically, one needs to design locality sensitive hash functions case by case. Solutions for some popular domains are available, including Euclidean spaces [Andoni and Indyk, 2006], binary vector spaces with Hamming distance [Indyk and Motwani, 1998], and vector spaces using vector angle distance [Charikar, 2002].

With an appropriate family of locality sensitive hash functions H, a search method can be designed as follows. First, a new family G of compounded hash functions is defined, such that for any $h_1, \cdots, h_m \in H$, a function $g \in G$ exists with

$$g(u) = (h_1(u), \cdots, h_m(u)).$$

Then, data objects are hashed into each of the l hash tables, each corresponding to a different randomly chosen hash function $g \in G$. Given a query object q, similar objects can be found by hashing q using the l hash functions and by retrieving the data objects that are hashed into the same buckets as q. In [Indyk and Motwani, 1998], the authors showed that if m and l are chosen sufficiently large, then one can obtain said approximation with constant probability. Search methods based on locality sensitive hashing are also susceptible to the concentration effect on distances. The gap $|r_2 - r_1|$ is a crucial factor for hashing-based methods to be effective. However, in high-dimensional spaces, a large number of hash functions is needed in order to amplify the distance gap, resulting in a high computational cost.

2.1.2 Adaptive Similarity Search Methods

Traditional similarity search methods typically require that the distance functions used to rank query results be fixed. In other words, they cannot handle adaptive similarity queries, which aim to find the most similar objects to a query object q from the database S with respect to an adaptive similarity measure — one that can be determined by the user at query time. In data mining applications such as subspace clustering or feature selection, changes to the underlying feature set can require the reconstruction of search indices to support fundamental data mining tasks. For such situations, adaptive search approaches have been proposed that can accommodate changes in the underlying similarity measure without the need to rebuild the index [Seidl and Kriegel, 1998].

The efficient handling of queries with respect to adaptive similarity measures can accelerate wrapper methods for feature selection [Kohavi and John, 1997], by accommodating small changes in the feature set (hopefully) without the need to rebuild the search indices associated with the feature evaluation process (such as k-NN classification or clustering). Subspace clustering [Kriegel et al., 2012] is another area that can potentially benefit from the availability of adaptive similarity indexing, as cluster formation is generally assessed with respect to a subset of the features. The effectiveness of using adaptive (or user-defined) similarity measures have already been demonstrated for problems with data mining applications, including content-based image retrieval [Seidl and Kriegel, 1997] and document clustering [Kim and Lee, 2002].

To handle adaptive queries, so-called 'multi-step' search strategy has been devised, by which a query result computed using a fixed 'lower-bounding' distance function can be adapted to answer the same query with respect to a user-supplied 'target' distance function. Formally, d' is a lower-bounding distance function for the target d if $d'(u,v) \leq d(u,v)$ for any two objects $u,v \in \mathbb{U}$, where $\mathbb{U}$ is a data domain upon which both d' and d are defined. Such lower-bounding relationships arise naturally, for example, when projecting high-dimensional feature vectors to low-dimensional feature vectors in Euclidean spaces.

In other cases, d' can be a lower-bounding approximation that is more efficient to compute than d. Lower-bounding distances that have appeared in the research literature include:

- The lower bound for the max-morphological distance of 2D shapes [Korn et al., 1996].
- The 3D-averaging lower bound [Rubner et al., 1998] and the L_p-norm-based lower bound [Assent et al., 2006] for the earth mover's distance.
- The lower bounds for the dynamic time-warping distance for time-series data [Vlachos et al., 2006].
- The L_2-based lower bound for the class of quadratic form distance functions [Hafner et al., 1995].

The class of quadratic form distance functions $d_M^2(u,v) = (u-v)^T M\,(u-v)$ allows the users to express their individual preferences by providing a weight matrix M. Note that, due to the complexity of defining a weight matrix manually, some relevance feedback mechanism usually needs to be incorporated with the use of quadratic form distance functions in order to automatically learn the matrix coefficients that best fit the users' preferences [Ishikawa et al., 1998].

In multi-step search, there are two main stages: filtering and refinement. In the filtering stage, from the data set S, a candidate set is generated using d'. In the refinement stage, the candidate set is refined using d to get the query result. The first multi-step k-NN search algorithm was proposed by Korn et al. [Korn et al., 1996]. In the filtering stage, the algorithm first obtains from the data set S the k-nearest neighbors of q with respect to d', and then computes their distances to q with respect to the target distance d. From among the target distance values, the maximum value d_{max} is determined. The candidate set is then obtained from S by performing a range query with respect to d', where d_{max} is used as the range limit. In the refinement stage, the algorithm simply evaluates the target distances from q to all the candidates to get the query result. The algorithm guarantees a correct query result; however, the candidate set produced may be prohibitively large.

Seidl and Kriegel [Seidl and Kriegel, 1998] later proposed a more efficient multi-step k-NN search algorithm. In contrast to the first multi-step algorithm, the algorithm proposed by Seidl and Kriegel performs rounds of filtering and refinement on a candidate set that grows incrementally. The algorithm scans the neighborhood list of the query object with respect to d' to retrieve candidates for the query result, and stops when the candidate k-NN distance (target distance) is no larger than the lower-bounding distance currently maintained by the scan. Seidl and Kriegel showed that their algorithm is optimal, in the sense that it produces the minimum number of candidates needed in order to guarantee a correct query result, based on the information available given d'. In other words, with even one such candidate missing, the query result may not be guaranteed correct.

Recently, the work of Seidl and Kriegel [Seidl and Kriegel, 1998] was extended by Kriegel et al. [Kriegel et al., 2007]. In this extension, an upper-bounding distance function was incorporated to reduce the number of evaluations of target distances. However, even if the upper-bounding distance function is a perfect approximation of the target distance function, at most k evaluations can be avoided. Considering the fact that the number of candidates needed to answer a query is typically much larger than the target neighborhood size k, the advantages are usually negligible.

2.1.3 Aggregate Similarity Search Methods

The aim of classical similarity queries is to retrieve from the database a set of objects most similar to a specified query object, based on a *single* ranking criterion that is usually expressed in terms of a similarity function. Recently, a novel type of similarity queries, aggregate similarity queries, has been studied, in which *multiple* ranking criteria are involved and the final rankings of objects are obtained by combining the individual rankings according to some monotone aggregation function (for example, *sum* or *max*). This subsection is devoted to the discussion of aggregate similarity queries. Two other general types of queries, top-k queries and skyline queries, are also discussed, since they also involve the aggregation of multiple ranking criteria.

Aggregate Similarity Queries. Given a group of query objects Q, aggregate similarity queries (also referred to as aggregate nearest neighbor (AggNN) queries) aim to retrieve the k objects from the database S that are most highly ranked with respect to Q, where the ranking criterion (similarity measure) is defined as an aggregation (usually *sum* or *max*) of distances between the retrieved objects and every query object in Q.

As a scenario illustrating the need for aggregate similarity search, we could imagine a spatial database consisting of a set of potential locations to construct a retail outlet, within which real estate developers can pose queries on collections of groups of potential customers at different locations. In order to maximize profit, the developers may wish to determine potential sites for the construction of new outlets that would minimize the total (or maximum) distance to target sets of customers. If there are more factors to be considered, such as the land price, then the k most suitable sites may be retrieved from the database as candidates.

Aggregate similarity search techniques could also benefit applications that make use of relevance feedback, a form of query refinement in which the user is given the opportunity to select several objects from a previous query result to serve as the basis for a subsequent query. In [Razente et al., 2008b], Razente et al. proposed aggregate similarity queries as a relevance feedback mechanism for content-based image retrieval. Moreover, as pointed out in [Razente et al., 2008a; Papadias et al., 2004], aggregate similarity queries can be utilized in clustering and outlier detection. For example, the quality of a solution for clustering or outlier detection can be evaluated by the total (or maximum) distance between the points and their nearest cluster centroid [Papadias et al., 2004].

Due to its importance and generality, AggNN has received a considerable amount of attention in the literature. Aggregate similarity search was first studied by Papadias et al. [Papadias et al., 2004] for the *sum* variant of the AggNN problem in Euclidean space. The state-of-the-art approach was proposed in [Papadias et al., 2005b], which is designed based on the R-tree index [Guttman, 1984]. The typical branch-and-bound methodology is adopted for pruning the search space. For general metric spaces, a similar approach was developed by Razente et al. [Razente et al., 2008a] using distance-based indexes such as the M-tree [Ciaccia et al., 1997]. These methods do not scale well due to the curse of dimensionality.

To improve the efficiency of the search, several approximation methods were also proposed [Li et al., 2011a; Papadias et al., 2005b]. In [Li et al., 2011a], Li et al. proposed an approximation method for the *max* variant of the AggNN problem in Euclidean space. The idea is to first find the center of the minimum enclosing ball (MEB) of group query Q, and then simply return the k-NN of this center as the result. The center of the MEB of Q is the point which minimizes the maximum distance to any point in Q. Li et al. [Li et al., 2011a] showed that this simple method returns a $\sqrt{2}$-approximate query result. Several other approximation methods were proposed in [Papadias et al., 2005b], but with no provable approximation ratios.

In general, however, AggNN methods tend to favor only those objects that are similar to all query objects in Q, which in practice may greatly limit its performance when the objects of Q are greatly dispersed [Li et al., 2011b]. In our real-estate development scenario, instead of attempting to attract all targeted customers, a developer may be content with attracting a large proportion of the potential customers, by determining locations minimizing the total (or maximum) distances to at least this proportion of potential customers. For such scenarios, Li et al. [Li et al., 2011b] proposed a generalized problem, called flexible aggregate similarity search (FANN), in which the restrictiveness of AggNN is eased by calculating aggregate distances only over subsets of Q. More precisely, FANN aims to retrieve the k objects which are most similar to a subset of group query Q with size $\phi|Q|$, for some target proportion $0 < \phi \leq 1$. When $\phi = 1$, FANN is equivalent to AggNN.

Compared to AggNN, FANN is not only better suited for finding semantically meaningful results, but it also allows the user to formulate the group query Q with more flexibility. As pointed out in [Li et al., 2011b], FANN query results may also be more diverse than AggNN results, since every object in the FANN result set may be related to a distinct subset of Q. For these reasons, FANN is a more difficult problem than AggNN, and existing solutions for the AggNN problem cannot be effectively applied for FANN. Straightforward adaptation of these methods would require the checking of an exponential number of subsets of Q.

Two exact solutions, R-tree and List, were proposed in [Li et al., 2011b] for the FANN problem in L_p spaces and general metric spaces, respectively. Algorithm R-tree extends the solution for AggNN due to Papadias et al. [Papadias et al., 2005b] through an adaptation of the pruning bound. Algorithm R-tree can only be applied in low-dimensional settings (up to 10 dimensions). The design of the other exact method, List, is based on the well-known *Threshold Algorithm* (TA) [Fagin et al., 2001]. Algorithm List retrieves candidates for the query result from the neighborhoods of every object of group query Q in a round robin fashion, while maintaining a lower bound for the best possible ϕ-aggregate distance of all the unseen objects. The algorithm terminates when at least k of the objects visited have ϕ-aggregate distances no greater than the lower bound. Although the algorithm is capable of handling higher-dimensional data, the execution cost may be prohibitively expensive.

In an attempt to lower the computational cost of processing FANN queries, two approximation methods, ASUM and AMAX, were also proposed by Li et al. [Li et al., 2011b], the former for the *sum* variant of FANN in general metric spaces, and the other for the *max* variant in L_p spaces. With the ASUM heuristic, a candidate set is generated by taking the union of the k-NN sets for every query object in Q. The k best objects within the candidate set is then returned as the result. The AMAX heuristic is very similar to ASUM, the only difference being that AMAX first finds the center of the minimum enclosing ball (MEB) of the query set neighborhood $N_Q(q, \phi m)$, based at each query object $q \in Q$; AMAX then treats these centers as query objects. When $\phi = 1$, AMAX is equivalent to the algorithm proposed in [Li et al., 2011a] for the *max* variant of AggNN. Li et al. [Li et al., 2011b] showed that ASUM returns a 3-approximate query result, and AMAX returns a $(1+2\sqrt{2})$-approximate query result. They also proposed faster variants of ASUM and AMAX in which certain operations are performed only for a reduced, randomly sampled subset of the group query set Q. An analysis is provided that shows that sampling variants achieve the original approximation ratio with a probability of success that depends on the size of the sample.

Top-k Queries. Assuming that each object in the database is associated with m scores obtained by applying different ranking criteria (the smaller the better), a top-k query returns the k objects with the smallest combined scores according to some (typically monotone) aggregation function, such as *min*, *max* and *sum*. For example, imagine a database of hotel records associated with two partial ranking scores, one on price, and the other on service (see Figure 2.3). A user may issue a query with $k = 1$ and aggregation function being *sum*, trying to find the hotel with the best combination of price and service. For this query, hotel h_8 would be returned to the user. Note that aggregate similarity queries discussed previously can be viewed as a special case of top-k queries.

Top-k queries can be useful in a variety of application areas such as similarity search on multimedia data [Chaudhuri et al., 2004; Nepal and Ramakrishna, 1999; Fagin, 1999; Güntzer et al., 2000; Natsev et al., 2001], ranked retrieval on digital libraries and on the Web [Anh et al., 2001; Long and Suel, 2003; Theobald et al., 2004; Bawa et al., 2003; Carmel et al., 2001; Persin et al., 1996], and collaborative recommendation on e-commerce product catalogs [Yu et al., 2003; Bruno et al., 2002; Güntzer et al., 2001; Chang and Hwang, 2002].

Existing methods for answering top-k queries can be classified into two categories: methods with the assumption that the m partial scores can be precomputed (available for preprocessing) and methods without such assumption. Within the first category, Chang et al. [Chang et al., 2000] proposed ONION, which answers top-k queries with respect to linear aggregation functions. In [Hristidis and Papakonstantinou, 2004], Hristidis and

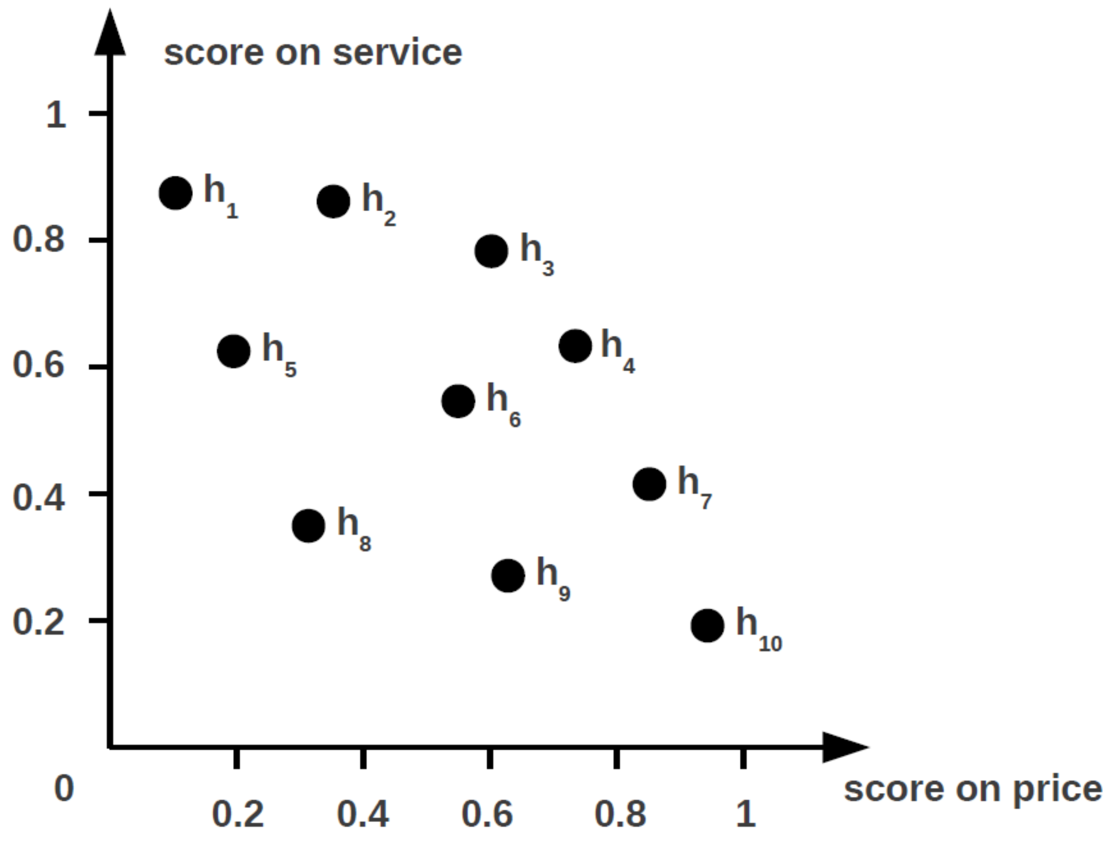


Figure 2.3 A database with hotel records.

Papakonstantinou proposed PREFER, which is able to answer top-k queries with a broader class of aggregation functions, but requires high maintenance cost. In [Yi et al., 2003], a similar approach was developed with lower maintenance cost. Tsaparas et al. [Tsaparas et al., 2003] proposed a method supporting arbitrary aggregation functions. However, this method is limited to two dimensions ($m = 2$). In [Tao et al., 2007], a method based on the R-tree [Guttman, 1984] was developed, which is able to handle all the monotone aggregation functions. Methods from the second category include Fagin's Algorithm (FA) [Fagin, 1999], the Threshold Algorithm (TA) [Fagin et al., 2001] and several variants of TA [Bruno et al., 2002; Güntzer et al., 2001; Fagin et al., 2001; Fagin et al., 2003].

Among the ample work on top-k query processing, Algorithm TA stands out as a very efficient and highly versatile method. For each ranking criteria, we conceptually build a ranked list, which lists all the objects in ascending order of their scores under that ranking criteria. Algorithm TA retrieves candidates for the query result by scanning these ranked lists from top to bottom in a round robin fashion, while maintaining a lower bound for the best possible aggregate score of all the unseen objects. The lower bound is obtained by aggregating the partial scores last retrieved from the ranked lists. The algorithm terminates when at least k of the objects visited have aggregate scores no greater than the lower bound. Algorithm TA works for all the monotone aggregation functions.

Skyline Queries. Previously discussed top-k queries assume the existence of an aggregation function for combining the partial ranking scores. However, in some applications, it might not be straightforward to define an aggregation function. For example, when query results are to be returned for more than one user, it would be more meaningful to return all interesting results, rather than a strict ordering based on some specified aggregation function. Skyline queries are introduced for such purposes.

A skyline query returns the objects that are not dominated by any other objects. An object dominates another object if it is as good or better with respect to all ranking criteria and better with respect to at least one ranking criterion. For example, the skyline in Figure 2.3 includes h_1, h_5, h_8, h_9 and h_{10}. A nice property of the skyline is that the top-1 object obtained according to any monotone aggregation function must be one of the skyline objects.

Existing methods for processing skyline queries can be classified into two categories: non-index based approaches and index based approaches. Methods from the first category do not assume any index on the underlying dataset. They retrieve the skyline by scanning the dataset. Examples from this category include Block Nested Loop (BNL) [Borzsony et al., 2001], Divide-and-Conquer (D&C) [Borzsony et al., 2001], Sort-Filter-Skyline (SFS) [Chomicki et al., 2003], Linear Elimination Sort for Skyline (LESS) [Godfrey et al., 2005], Sort and Limit Skyline algorithm (SaLSa) [Bartolini et al., 2008], and Object-based Space Partitioning (OSP) [Zhang et al., 2009]. The state-of-the-art approach for non-indexed data is OSP, which recursively divides the search space into separate partitions with respect to a reference skyline object, and progressively retrieves the skyline objects.

Methods from the second category utilize an appropriate index structure to facilitate skyline computation. Examples include Bitmap [Tan et al., 2001], Index [Tan et al., 2001], Nearest Neighbor (NN) [Kossmann et al., 2002], Branch and Bound Skyline (BBS) [Papadias et al., 2005a], and ZSearch [Lee et al., 2007]. The state-of-the-art approach within the second category is BBS, which utilizes the R-tree [Guttman, 1984] to compute the skyline.

2.1.4 Subspace Similarity Search Methods

In this subsection, we give a discussion of methods designed for the subspace similarity search problem. Subspace similarity search aims to retrieve from the database the most similar objects to a query object with the similarity distance function being defined over a subset of features (an axis-aligned projective subspace). In particular, each query may specify an arbitrary subspace with an arbitrary number of features. That is, the number of possible choices of subspace in a D-dimensional full space is $2^{\mathrm{D}} - 1$.

As with traditional similarity search on fixed spaces, subspace similarity search may also have an impact in application areas where the feature set under consideration changes from operation to operation. Such changes could be due to a modification of query preferences (as in content-based image retrieval), or to the determination of the local structure at different locations within data (as in subspace clustering), or to a systematic exploration of feature subspaces (as in feature selection).

In content-based image retrieval, images are often represented by feature vectors extracted based on color, shape, and texture descriptors. In an exploration of the data set, a query involving one combination of features (such as color) may be followed by a query on a different combination (such as shape). In subspace clustering [Kriegel et al., 2012], the formation of an individual cluster is generally assessed with respect to a subset of features that most closely describe the concept associated with the cluster. Since verification of a cluster requires the identification of a feature subset together with an object subset, the effectiveness of the overall clustering process may depend on the efficient processing of subspace similarity queries. Wrapper methods for feature selection [Kohavi and John, 1997] require an evaluation process, such as k-nearest neighbor (k-NN) classification, for the identification of effective combinations of features. Exploration of feature subspaces can be extremely time-consuming when the neighborhoods are determined using exhaustive search, due to the exponential number of potential combinations involved. To accelerate the process, the efficient support of subspace similarity search is needed.

Almost all existing similarity search methods, including the popular *locality sensitive hashing* [Indyk and Motwani, 1998], require that the similarity measure and associated vector space both be specified before any preprocessing occurs. Traditional methods for fixed spaces cannot be effectively applied for the subspace search problem: the subspaces to be searched are typically not known until query time, but even if they were known in advance, preprocessing every possible query subspace would be prohibitively expensive. Of all the methods for similarity search appearing in the research literature, only very few have been specifically formulated for the subspace search problem [Kriegel et al., 2006; Bernecker et al., 2010b; Bernecker et al., 2010a; Lian and Chen, 2008].

In [Kriegel et al., 2006], the Partial VA-file (PVA) was proposed, which adapts the vector approximation file (VA-file) [Weber et al., 1998] to support subspace queries. The VA-file, designed for fixed-space similarity search, stores a compressed approximation of the data as a single file; at query time, the compressed approximation is scanned in its entirety, and uses the information for pruning the search within the original dataset. PVA, on the other hand, stores an approximation of data on each dimension separately, and processes the search using only those 1-dimensional VA-files that correspond to dimensions involved in the query. PVA may suffer greatly in terms of the computational cost, due to its use of sequential scan.

In [Bernecker et al., 2010b], the Dimension-Merge Index (DMI) was developed, which combines multiple 1-dimensional index structures to answer subspace queries. DMI builds an index for each dimension separately (of any desired type), and utilizes those indexes with respect to the query dimensions to perform the search. The final query result is obtained by aggregation across neighborhoods associated with each of the query dimensions. Such aggregation may become prohibitively expensive as the number of query dimensions increases.

In [Bernecker et al., 2010a], the Projected R-Tree (PT) was proposed as a redefinition of the classical search structure R-tree [Guttman, 1984] for subspace similarity search. Instead of integrating results of queries on 1-dimensional indices, PT utilizes a single index built on the full feature space (an R-tree) to answer queries with respect to subspaces. A best-first search heuristic is employed, subject to the restriction that only the query dimensions are considered for distance computations. Due to the limits on the performance of R-trees for spaces of even moderate dimensionality, PT is expected to be prohibitively expensive as the number of query dimensions grows.

Another approach to the subspace search problem was proposed in [Lian and Chen, 2008], for range queries. The algorithm first selects several data objects as *pivots*, and then assigns 1-dimensional scores to each data object with respect to the chosen *pivots*. Finally, these scores are utilized at query time to reduce the search space through the application of the triangle inequality. As discussed previously, the triangle inequality may cease to provide an effective means for pruning the search space while processing high-dimensional queries, due to the concentration effect on distances.

2.2 Generalized Expansion Dimension

In this section, we introduce the model of *generalized expansion dimension* (GED) [Houle et al., 2012a], a framework for estimating the intrinsic dimension of data.

2.2.1 Expansion Dimension

In [Karger and Ruhl, 2002], Karger and Ruhl introduced a measure of intrinsic dimension as a means of analyzing the performance of a local search strategy for handling nearest neighbor queries. The complexity of their method depends heavily on the rate at which the number of visited elements grows as the search expands. Accordingly, they limited their attention to sets which satisfied the following *smooth-growth property*. Let $\mathbb{B}_S(q,r) = \{v \in S \mid d(q,v) \le r\}$ be the set of elements of $S \subseteq \mathbb{U}$ contained in the closed ball of radius r centered at $q \in \mathbb{U}$, where $\mathbb{U}$ is a data domain. S is said to have (b,α)-expansion if for all $q \in \mathbb{U}$ and $r > 0$,

$$|\mathbb{B}_S(q,r)| \ge b \Longrightarrow |\mathbb{B}_S(q,2r)| \le \alpha \cdot |\mathbb{B}_S(q,r)|.$$

The *expansion rate* of S is the minimum value of α such that the above condition holds, subject to the choice of minimum ball size b (in their analysis, Karger and Ruhl chose $b = \mathbf{O}(\log |S|)$).

One can consider the value $\log_2 \alpha$ to be a measure of the intrinsic dimension, by observing that for the Euclidean distance metric in $\mathbb{R}^c$, doubling the radius of a sphere increases its volume by a factor of 2^c. When sampling $\mathbb{R}^c$ by a uniformly distributed point set, the expanded sphere would contain proportionally as many points. Accordingly, the value of $\log_2 \alpha$ is often referred to as the *expansion dimension* of the set. It should be noted that this doubling property is true of many other distance measures as well, including the L_p norms. For other norms, the dependence on c may be different; one such example is the vector angle distance metric (equivalent to the cosine similarity measure), where the volume increases by a factor of 2^{c-1}. The definition of expansion dimension can be extended to handle such distance metrics [Houle et al., 2012a].

2.2.2 Generalized Expansion Dimension: Definition

The original definition of expansion rate was motivated to support analysis that depend on the use of the triangle inequality. In these contexts, it is useful to consider the effect of a doubling of the radius of the bounding balls. In general, however, the dimensionality of the space is revealed by any two balls of differing radii. Let B_1 and B_2 be two balls with the same center q and unequal radii $r_1 \neq r_2$. If V_1 and V_2 are the respective volumes of B_1 and B_2, then the dimension of the set is given by

$$\mathrm{rDim} = \frac{\log V_2 - \log V_1}{\log r_2 - \log r_1}.$$

Accordingly, by estimating the spherical volumes V_1 and V_2 by the numbers of data points they contain (call them k_1 and k_2), we can estimate the dimensionality of the data set in the vicinity of B_1 and B_2 by:

$$\mathrm{GED}(B_1, B_2) = \frac{\log k_2 - \log k_1}{\log r_2 - \log r_1}.$$

The balls B_1 and B_2 constitute two lower-dimensional measurements — distance-rank pairs — that together allow an assessment of the higher-dimensional structure — the intrinsic dimension — of the data points in the vicinity of the balls. $\mathrm{GED}(B_1, B_2)$ is referred to as a *dimensional test* value from the perspective of B_1 and B_2. Changing B_1 or B_2 would likely produce a change in the result of the dimensional test; however, aggregating the results of such tests over a range of ball radii or sizes can produce a useful estimate of the overall intrinsic dimension in the vicinity of a query point.

We next present the formal definition of the generalized expansion dimension (GED). Given a dataset $S \subseteq \mathbb{U}$ and a point $q \in \mathbb{U}$, let $\delta_S(q,k)$ be the k-th smallest distance from q to the points in S. Given a dataset $S \subseteq \mathbb{U}$, a point $q \in \mathbb{U}$, and a radius r, let $B_S^{\leq}(q,r)$ denote the closed ball centered at q with radius r containing all the points $v \in S$ satisfying $d(q,v) \leq r$, and let $B_S^{<}(q,r)$ denote the open ball centered at q with radius r containing all the points $v \in S$ satisfying $d(q,v) < r$. Let $|B_S^{\leq}(q,r)|$ and $|B_S^{<}(q,r)|$ denote the numbers of points of S contained in the two balls $B_S^{\leq}(q,r)$ and $B_S^{<}(q,r)$, respectively. Given two closed balls $B_S^{\leq}(q,r_1)$ and $B_S^{\leq}(q,r_2)$ with radii $0 < r_1 < r_2$ and the numbers of points of S contained in them satisfying $0 < |B_S^{\leq}(q,r_1)| < |B_S^{\leq}(q,r_2)|$, the GED is defined as in Figure 2.4.

Also defined in Figure 2.4 are the *inner ball set*, the *outer ball set* and the *maximum expansion dimension* (MED), all relative to a point $q \in \mathbb{U}$ and a neighborhood size $k \geq 2$. Note that the $\mathrm{MED}(q, k)$ has no definite value when the inner ball set $\mathfrak{B}_{\mathrm{in}}(q, k)$ is empty. In other words, in order for $\mathrm{MED}(q, k)$ to have a definite value, the ball $B_S^{\leq}(q, \delta_S(q, j))$ must have radius satisfying $0 < \delta_S(q, j) < \delta_S(q, k)$ for at least one choice of j in the range $[1, k-1]$. The *maximum restricted expansion dimension* (MRED) is also defined in Figure 2.4, in which only one outer ball, $B_S^{\leq}(q, \delta_S(q, k))$, is considered for the computation of maximum GED value.

Very recently, Houle proposed the *continuous intrinsic dimensionality* (CID) [Houle, 2013], which models the intrinsic dimensionality in terms of the distribution of neighborhood measure. We next briefly discuss the framework of CID and the connection between GED and CID. Given a data domain for which some distance measure is defined, let us consider the distribution of distances within the domain with respect to some fixed point of reference. This distribution can be modeled in terms of a random variable $\mathbf{X}$ with support $[0, \infty)$. $\mathbf{X}$ is said to have probability density $f_{\mathbf{X}}$, where $f_{\mathbf{X}}$ is a non-negative Lebesgue-integrable function, if

$$\Pr[a \leq \mathbf{X} \leq b] = \int_a^b f_{\mathbf{X}}(x)\, dx.$$

The corresponding cumulative density function $F_{\mathbf{X}}$ is defined as

$$F_{\mathbf{X}}(x) = \Pr[\mathbf{X} \leq x] = \int_0^x f_{\mathbf{X}}(u)\, du.$$

Accordingly, when $\mathbf{X}$ is absolutely continuous at x, $F_{\mathbf{X}}$ is differentiable at x and its first-order derivative is $f_{\mathbf{X}}(x)$. Let $\mathbf{X}$ be an absolutely continuous random distance variable. For any distance threshold x such that $F_{\mathbf{X}}(x) > 0$, the continuous intrinsic dimension of $\mathbf{X}$ at distance x is given by

$$\mathrm{ID}_{\mathbf{X}}(x) = \lim_{\epsilon \to 0^+} \frac{\log F_{\mathbf{X}}((1+\epsilon)x) - \log F_{\mathbf{X}}(x)}{\log(1+\epsilon)},$$

Generalized expansion dimension:

$$\mathrm{GED}(B_S^{\leq}(q,r_1), B_S^{\leq}(q,r_2)) = \frac{\log|B_S^{\leq}(q,r_2)| - \log|B_S^{\leq}(q,r_1)|}{\log r_2 - \log r_1}.$$

Inner ball set:

$$\mathfrak{B}_{\mathrm{in}}(q,k) = \{B_S^{\leq}(q,\delta_S(q,j)) \mid 1 \leq j \leq k-1, \delta_S(q,j) > 0\} \setminus \{B_S^{\leq}(q,\delta_S(q,k))\}.$$

Outer ball set:

$$\mathfrak{B}_{\mathrm{out}}(q,k) = \{B_S^{\leq}(q,\delta_S(q,j)) \mid k \leq j \leq |S|\}.$$

Maximum expansion dimension:

$$\mathrm{MED}(q,k) = \max\{\mathrm{GED}(B_{\mathrm{in}}, B_{\mathrm{out}}) \mid B_{\mathrm{in}} \in \mathfrak{B}_{\mathrm{in}}(q,k), B_{\mathrm{out}} \in \mathfrak{B}_{\mathrm{out}}(q,k)\}.$$

Maximum restricted expansion dimension:

$$\mathrm{MRED}(q,k) = \max\{\mathrm{GED}(B_{\mathrm{in}}, B_S^{\leq}(q,\delta_S(q,k))) \mid B_{\mathrm{in}} \in \mathfrak{B}_{\mathrm{in}}(q,k)\}.$$

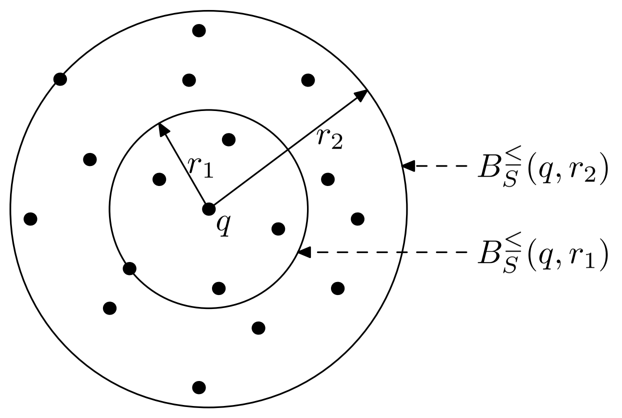


Figure 2.4 Formal definitions of generalized expansion dimension and maximum expansion dimension (from [Houle et al., 2012a]).

wherever the limit exists.

In [Houle, 2013], Houle showed several interesting findings about the CID, of which the most important one is that CID is equivalent to a formulation of indiscriminability of a distance measure. The connection between GED and CID is that GED can be used to estimate CID. The above definition of CID can be rewritten in terms of the expected number of points within distances $(1+\epsilon)x$ and x:

$$\mathrm{ID}_{\mathbf{X}}(x) = \lim_{\epsilon\to 0^+} \frac{\log\left(|S|\cdot F_{\mathbf{X}}((1+\epsilon)x)\right) - \log\left(|S|\cdot F_{\mathbf{X}}(x)\right)}{\log\left((1+\epsilon)x\right) - \log x}.$$

If we estimate the expected values $|S|\cdot F_{\mathbf{X}}((1+\epsilon)x)$ and $|S|\cdot F_{\mathbf{X}}(x)$ by the observed values (call them k_2 and k_1), and ignore the limit by assuming that k_2 and k_1 are small relative to $|S|$, we are then actually performing a GED test of neighborhoods of radii $(1+\epsilon)x$ and x. In this way, we are able to use GED to estimate CID. Besides GED, there are several other estimators of CID [Amsaleg et al., 2014], including the maximum likelihood estimator (MLE), the method of moment estimator (MOM) and the probability-weighted moment estimator (PWM). Compared to other estimators, GED may be less stable, but it can be evaluated much faster. Therefore, it is more suitable to integrate GED into the search than integrating other estimators, which may cause much overhead.

In the past, measures of intrinsic dimension (such as the expansion dimension) have been used strictly for the analysis of similarity search methods. In the following chapters, we will present several works demonstrating that tests of generalized expansion dimension can be used to dynamically guide the decisions made by search algorithms.

CHAPTER 3

DIMENSIONAL TESTING FOR ADAPTIVE SIMILARITY SEARCH

In this chapter, we present the work on adaptive similarity search, which aims to find the most similar objects to a query object q from the database S with respect to an adaptive similarity measure — one that can be determined by the user at query time. In the research literature, the multi-step search strategy has been devised for handling adaptive similarity queries, by which a query result computed using a fixed 'lower-bounding' distance function can be adapted to answer the same query with respect to a user-supplied 'target' distance function. Formally, d' is a lower-bounding distance function for the target d if $d'(u,v) \leq d(u,v)$ for any two objects $u,v \in \mathbb{U}$, where $\mathbb{U}$ is a data domain upon which both d' and d are defined.

In multi-step search, there are two main stages: filtering and refinement. In the filtering stage, from the data set S, a candidate set is generated using d'. In the refinement stage, the candidate set is refined using d to get the query result. Seidl and Kriegel [Seidl and Kriegel, 1998] developed a multi-step k-NN search algorithm which is known to filter the minimum number of candidates needed in order to guarantee a correct query result after refinement. Nevertheless, as we will show later, the Seidl-Kriegel (SK) algorithm may still produce a very large candidate set.

Motivated by this observation, we propose a heuristic multi-step k-NN search algorithm, that utilizes a measure of the intrinsic dimension of the data, the *generalized expansion dimension* [Houle et al., 2012a] (see Section 2.2 in Chapter 2), as the basis of an early termination condition. In contrast with previous multi-step algorithms, we consider a generalization of the lower-bounding relationship between d' and d, in which $\lambda \cdot d'(u,v) \leq d(u,v)$ for any two objects $u,v \in \mathbb{U}$, given some lower-bounding ratio $\lambda \geq 1$. The motivation for introducing λ is to allow as tight as possible a fit between the lower-bounding distance and the target distance.

The main contributions are:

- A novel multi-step algorithm for approximate similarity search, in which tests of generalized expansion dimension are used to guide early termination decisions.
- A theoretical analysis of our approach that shows conditions under which an exact result can be guaranteed.
- An experimental comparison with previous multi-step algorithms, showing that our approach achieves significant reductions in both execution time cost and the number of filter candidates considered, while losing very little in the accuracy of the query result.

The remainder of this chapter is organized as follows. Section 3.1 discusses the Seidl and Kriegel's algorithm in detail, which is our main competitor in this work. Section 3.2 presents our proposed algorithm, as well as its analysis. Experimental results are presented in Section 3.3. The discussion is concluded in Section 3.4.

3.1 Seidl and Kriegel's Algorithm

In the remainder of this chapter, we will refer to the algorithm proposed by Seidl and Kriegel [Seidl and Kriegel, 1998] as SK. The description of Algorithm SK is presented in Figure 3.1. Algorithm SK performs rounds of filtering and refinement on a candidate set that grows incrementally. The function *getnext* allows for incremental generation of the neighbors of q with respect to d'. For most search indexes, this can be done with simple bookkeeping of the search state, as indicated in [Li et al., 2011b]. With the bookkeeping, the search for the $(j+1)$-th nearest neighbor can be resumed from the saved state after the termination of the search for the j-th nearest neighbor.

At every iteration of the main loop of Algorithm SK, the candidate set is the current neighborhood of q with respect to d'. From the candidate set, the k-nearest neighbors of q with respect to d are stored as a tentative query result (the set of object-distance pairs P), and the k-th smallest target distance is maintained (d_{max}). The algorithm terminates when the value of d_{max} is no greater than the largest lower-bounding distance that has been seen so far (β'), or when all the objects in the data set S have been fetched as candidates

```
Algorithm SK (query q, neighborhood size k)
1:  Assume there is an index $I'$ created with respect to $d'$, together with a method getnext
    that uses $I'$ to iterate through the nearest neighbor list of $q$.
2:  Let $P$ be a set of pairs with the form $(v, \beta)$, where $v$ is an object and $\beta = d(q, v)$.
3:  $P \leftarrow \emptyset$.
4:  $d_{\max} \leftarrow \infty$.
5:  repeat
6:      $(v, \beta') \leftarrow I'$.getnext.    // $\beta' = d'(q, v)$
7:      if $\beta' \geq d_{\max}$ then
8:          Return P.
9:      end if
10:     Compute the target distance value $d(q, v)$.
11:     $P \leftarrow P \cup \{(v, d(q, v))\}$.
12:     if $|P| < k$ then
13:         Continue to Line 6.
14:     else if $|P| > k$ then
15:         Delete the pair with the largest distance $\beta$ from $P$. Ties are broken arbitrarily.
16:     end if
17:     Update the value of $d_{\max}$ to be the largest distance value $\beta$ in P.
18: until no more objects to be retrieved from $I'$.
19: Return P.
```

Figure 3.1 The multi-step algorithm proposed by Seidl and Kriegel [Seidl and Kriegel, 1998].

and refined. In [Seidl and Kriegel, 1998], Seidl and Kriegel showed that at termination, the value of $d_{\max}$ will have been decreased to the exact k-th smallest distance from q to the objects in S, ensuring the optimality of the algorithm.

Though SK produces only those candidates which are necessary to obtain an exact query result, it may still produce an unsatisfactorily large candidate set if the lower-bounding distance function d' used for filtering is not a good approximation of the exact distance function d. Clearly, the larger the number of candidates after the filtering step, the more expensive the total cost of evaluating exact distances.

In Figure 3.2, we illustrate how Algorithm SK handles a typical 10-NN query on a real data set, the Amsterdam Library of Object Images (ALOI) data set described in

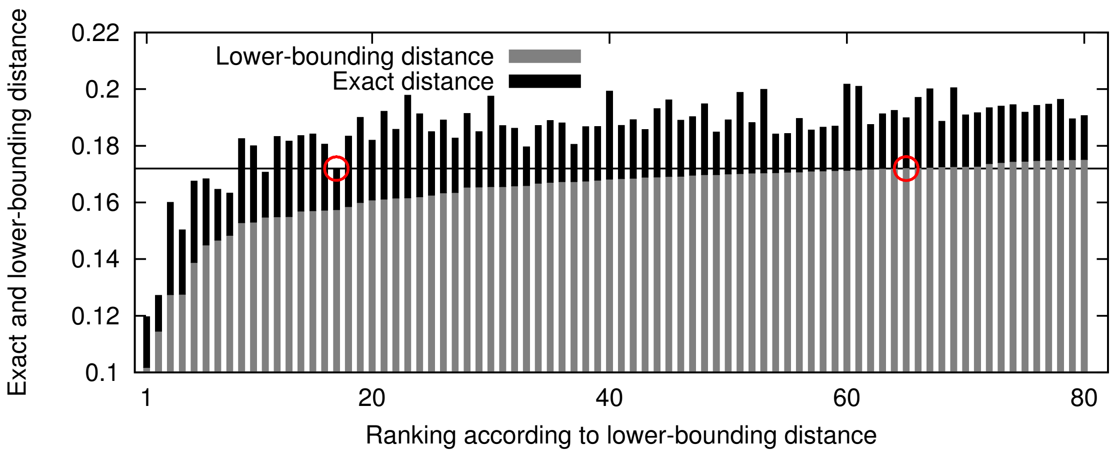


Figure 3.2 An example of a typical 10-NN query on the data set ALOI. The 10-th smallest exact distance is marked by the horizontal line. In this example, Algorithm SK produces 64 candidates, although the correct neighbor set is available once the 17th candidate has been produced (as marked by red circles).

Section 3.3. For the example, we projected the original 641-dimensional feature space to a 20-dimensional feature space, and used the Euclidean distance on these respective spaces as the target and lower-bounding distance measures. The 10-th smallest target distance is 0.172, which is marked by the horizontal line in the figure. To find a lower-bounding distance no less than 0.172 (thereby reaching one of the two termination conditions), Algorithm SK must produce and refine 64 candidates. All these candidates must be verified in order to guarantee that a correct query result is returned. At the iteration when the last candidate is verified, the lower-bounding distance is 0.1719; verification of the last candidate is therefore necessary to determine whether its target distance to q falls in the range $[0.1719, 0.172)$. For this particular example, the correct query result is available after only 17 candidates have been produced. Furthermore, 90% accuracy in the query result can still be achieved even after only 11 candidates have been produced. Clearly, if one is willing to accept a query result that is not necessarily exact, the potential exists for improving the performance of SK by reducing the number of candidates through early termination.

3.2 Algorithm Design and Analysis

We now turn our attention to the main contribution of this chapter, Algorithm MAET (Multi-step Algorithm with Early Termination). Let us first introduce some needed notation. For a query $q \in \mathbb{U}$ and a data set $X \subseteq \mathbb{U}$, let $\nu_X(q,k)$ be the k-th nearest neighbor of q in X with respect to d. Ties in distance values are broken arbitrarily but consistently. Let $\delta_X(q,k) = d(q, \nu_X(q,k))$ be the distance from q to its k-th nearest neighbor. Let $N_X(q,k)$ be the set of k-nearest neighbors of q over X, with respect to d. Ties are broken arbitrarily and consistently. Let $R_X^{\leq}(q,r)$ be the closed range set of objects in X having distances to q (with respect to d) no greater than r, and $R_X^{<}(q,r)$ be the open range set of objects in X having distances to q (with respect to d) strictly less than r.

3.2.1 Algorithm MAET

Algorithm MAET is described in Figure 3.3. The algorithm accepts a query $q \in \mathbb{U}$, a target query result size $k > 0$, a lower-bounding ratio $\lambda \geq 1$, and a termination parameter $t > 0$ as its inputs. Throughout the search process, the algorithm maintains a tentative query result set P. At each iteration, the algorithm obtains a candidate object from the underlying index structure I', computes its exact distance to q, and stores in the result set P those k objects associated with the smallest distances encountered so far. After at least k candidates have been retrieved and stored in P, two closed balls centered at q are determined. The inner ball contains at least $k_1 > 0$ points of S and has $r_1 > 0$ as its radius; the outer ball contains at least $k > 0$ points of S and has $r > 0$ as its radius. Provided that the two balls are distinct ($k_1 \neq k$), a dimensional test is performed. If the test indicates that a termination condition has been reached (as determined by termination parameter t), the algorithm halts and returns P. In the next subsection, we will show that when $k_1 > 0$ and $k_1 \neq k$ hold, $k_1 < k$ and $r_1 < r$ must hold, and the inner ball contains exactly k_1 points of S.

An illustration of MAET, with lower-bounding ratio $\lambda = 1$, is shown in Figure 3.4. All the objects of S are sitting in the area of $x \geq 0$ and $y \geq 0$. For a point $v \in S$, its target distance to q, $d(q,v)$, is represented by its Euclidean distance to the origin; its

```
*Algorithm* **MAET** $($*query* $q$, *neighborhood size* $k$, *lower-bounding ratio* $\lambda$, *termination parameter* $t)$
1: Assume there is an index $I'$ created with respect to $d'$, together with a method *getnext* that uses $I'$ to iterate through the nearest neighbor list of $q$.
2: Let $P$ be a set of pairs with the form $(v, \beta)$, where $v$ is an object and $\beta = d(q,v)$. Let $P_v$ denote the object set associated with $P$.
3: $P \leftarrow \emptyset$.
4: **repeat**
5:    $(v, \beta') \leftarrow I'$.*getnext*.    // $\beta' = d'(q,v)$
6:    Compute the target distance value from $q$ to $v$, $d(q,v)$.
7:    $P \leftarrow P \cup \{(v, d(q,v))\}$.
8:    **if** $|P| < k$ **then**
9:       Continue to Line 5.
10:   **else if** $|P| > k$ **then**
11:      Delete the pair with the largest distance $\beta$ from $P$. Ties are broken arbitrarily.
12:   **end if**
13:   $k_1 \leftarrow |R^{<}_{P_v}(q, \lambda \cdot \beta')|$.
14:   **if** $k_1 = k$ **then**
15:      Return $P$.
16:   **else if** $k_1 > 0$ **then**
17:      $r \leftarrow \delta_{P_v}(q,k)$.
18:      $r_1 \leftarrow \delta_{P_v}(q,k_1)$.
19:      **if** $r_1 > 0$ and $k_1 \cdot (r/r_1)^t < k+1$ **then**
20:         Return $P$.
21:      **end if**
22:   **end if**
23: **until** no more objects to be retrieved from $I'$.
24: Return $P$.
```

Figure 3.3 The description of Algorithm MAET.

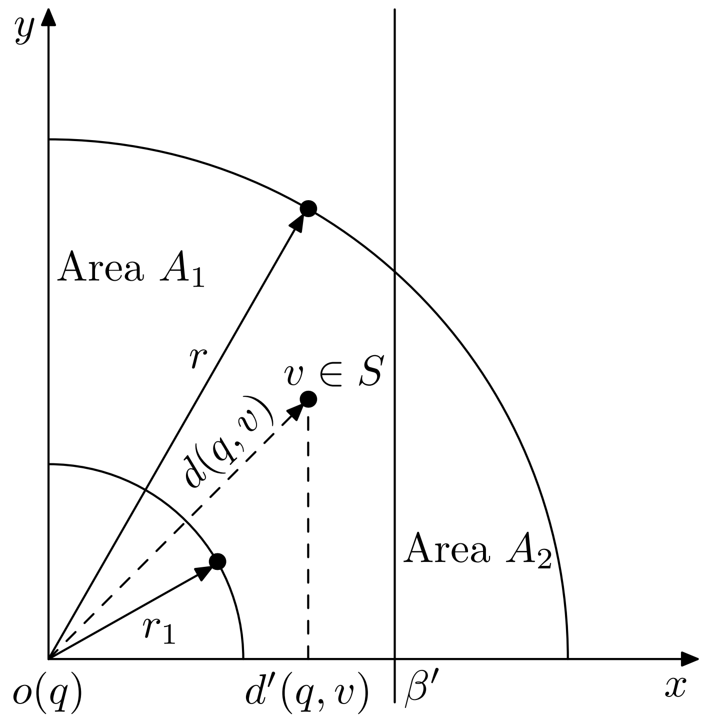


Figure 3.4 Illustration of MAET, with lower-bounding ratio $\lambda = 1$.

lower-bounding distance to q, $d'(q,v)$, is represented by its x coordinate. Accordingly, its y coordinate equals the square root of the difference between its squared target distance and squared lower-bounding distance, $\sqrt{(d(q,v))^2 - (d'(q,v))^2}$. The two balls are represented by two sectors as shown in the figure. The outer ball is divided into two regions, A_1 and A_2, by the vertical line $x = \beta'$. All the candidates that have been obtained at the current iteration are in the area of $0 \leq x \leq \beta'$ and $y \geq 0$, and the k objects currently stored in P are in the area A_1. It is easy to see from the figure that if area A_2 contains no object of S, then we have found the correct query result, the k-nearest neighbors of q over S with respect to d. Our goal is to reach this conclusion via dimensional testing while producing as few candidates as possible.

Let us return to the example presented in Figure 3.2. If we provide this data set to MAET (as S), together with $\lambda = 1$ and $t = \text{MRED}(q, k+1)$ (see Figure 2.4 on Page 32 for the definition of MRED), MAET would be able to return the correct query result while producing only 17 candidates. Specifically, after the first 10 candidates have been obtained via *getnext*, MAET will find an inner ball with radius $r_1 = 0.15$ containing $k_1 = 3$ points of S. With $t = \mathrm{MRED}(q, k+1)$, which is approximately 8.75 for this example, the dimensional test in Line 19, with outer ball having radius $r = 0.172$, will succeed immediately after the point with 10-th smallest target distance to q is discovered. Note that the inner ball does not change throughout the search process in this example.

The choice of t is generally made heuristically — larger choices can be expected to lead to a larger candidate set, which in turn would yield higher accuracies at the expense of computational cost. However, the analysis presented in the next subsection shows that if $\mathrm{MRED}(q, k+1)$ has a definite value, then the correctness of the method can be guaranteed whenever $t \geq \mathrm{MRED}(q, k+1)$; if not, then the correctness of the method can be guaranteed regardless of the value of t.

3.2.2 Analysis of MAET

We shall now give a formal theoretical analysis of MAET that establishes conditions by which its correctness can be guaranteed. At each iteration of the main loop, after the execution of line 13, let $K_1 = R^{<}_{P_v}(q, \lambda\beta')$ denote the current set of objects in P_v (the object set associated with the tentative query result set P) with target distances to q strictly less than $\lambda\beta'$. The size of K_1 is k_1 (line 13).

Lemma 1 *Let $\mathbb{U}$ be a data domain upon which distance functions d and d' are defined, with $\lambda d'(u,v) \leq d(u,v)$ for all $u, v \in \mathbb{U}$, where $\lambda \geq 1$. Let $q \in \mathbb{U}$, $k > 0$, λ and $t > 0$ be the inputs provided to a call to Algorithm MAET. At each iteration of the main loop, after the execution of line 13, if $k_1 = k$, then $K_1 = N_S(q,k)$; if $k_1 < k$, then $K_1 = R^{<}_{S}(q, \lambda\beta')$.*

Proof: Let T' denote the set of candidates we have obtained so far. First, we note that P_v, the set of objects currently stored in P, is $N_{T'}(q,k)$ (from lines 5-12), and $K_1 = R^{<}_{P_v}(q, \lambda\beta')$ with size k_1. If $k_1 = k$, obviously $K_1 = N_{T'}(q,k)$. If $k_1 < k$, then any object $v \in T' \setminus P_v$ must have $d(q,v) \geq \delta_{P_v}(q,k) \geq \lambda\beta'$, and therefore $K_1 = R^{<}_{T'}(q, \lambda\beta')$. Also, in both cases, we note that the candidates are retrieved from the underlying index incrementally, which ensures that the candidates are retrieved in ascending order of their lower-bounding distance to q. Then, any object $v \in S \setminus T'$ must have $d(q,v) \geq \lambda d'(q,v) \geq \lambda\beta'$. Therefore, if $k_1 = k$, then $K_1 = N_S(q,k)$ must be true; if $k_1 < k$, then $K_1 = R_S^{<}(q,\lambda\beta')$ must be true. □

Theorem 1 *Let* $\mathbb{U}$ *be a data domain upon which distance functions* d *and* d' *are defined, with* $\lambda d'(u,v) \leq d(u,v)$ *for all* $u,v \in \mathbb{U}$ *where* $\lambda \geq 1$. *Let* $q \in \mathbb{U}$, $k > 0$, λ *and* $t > 0$ *be the inputs provided to a call to Algorithm MAET. If MRED*$(q,k+1)$ *has a definite value, then the algorithm returns the correct query result whenever* $t \geq$ *MRED*$(q,k+1)$. *Otherwise, the algorithm returns the correct query result regardless of the value of* t.

Proof: First, we note that the algorithm must terminate, as the iterative retrieval of candidates would eventually result in the termination condition of the loop being reached (Line 23). In this case, all the target distance values from q to every object in S would have been evaluated, and only the k objects with the smallest distances would have been stored in P (Lines 5-12). Obviously, the algorithm would return the correct result for this case.

Otherwise, the algorithm would terminate with one of the following two conditions being reached (Line 14 or 19): $k_1 = k$ or

$$k_1 \left(\frac{r}{r_1}\right)^t < k+1. \tag{3.1}$$

For the case when $k_1 = k$, $K_1 = N_S(q,k)$ must be true (from Lemma 1). By the definitions of K_1 and P_v, we know that all the objects in K_1 are currently stored in P. Also, we have $|P| = k$. Therefore, the algorithm returns the correct result for this case also.

For the last case when Inequality (3.1) holds, $r_1 > 0$, $k_1 > 0$ and $k_1 \neq k$ must be true (Lines 19 and 16). Together with $k_1 = |R_{P_v}^{<}(q,\lambda\beta')| \leq |P_v| = k$, we obtain $k_1 < k$ for this case. Since $|R_{P_v}^{<}(q,\lambda\beta')| = k_1 < k$, we have $r_1 = \delta_{P_v}(q,k_1) < \lambda\beta'$ and $r = \delta_{P_v}(q,k) \geq \lambda\beta'$. Consequently, $r_1 < r$ must be true for this case. From Lemma 1, we know that $K_1 = R_S^{<}(q,\lambda\beta')$. Together with $K_1 = R_{P_v}^{<}(q,\lambda\beta')$ and $|K_1| = k_1$, we can derive

$$r_1 = \delta_{P_v}(q,k_1) = \delta_{K_1}(q,k_1) = \delta_S(q,k_1) < \delta_S(q,k_1+1).$$

Therefore, the closed ball $B_S^{\leq}(q, \delta_S(q, k_1))$ has r_1 as its radius and contains exactly k_1 points of S. We also know that the closed ball $B_S^{\leq}(q, \delta_S(q, k+1))$ has $\delta_S(q, k+1)$ as its radius and contains $|B_S^{\leq}(q, \delta_S(q, k+1))|$ points of S. Since $k_1 < k$, we obtain $r_1 < \delta_S(q, k_1+1) \leq \delta_S(q, k+1)$. Thus, $\text{MRED}(q, k+1)$ has a definite value for this case. Assume $t \geq \text{MRED}(q, k+1)$, we can derive

$$k_1 \left(\frac{\delta_S(q, k+1)}{r_1} \right)^t \geq |B_S^{\leq}(q, \delta_S(q, k+1))| \geq k+1.$$

Together with Inequality (3.1), $r < \delta_S(q, k+1)$ must hold for this case. Also, we have $|P| = k$ and $r = \delta_{P_v}(q, k)$, so all the k objects in P have target distances to q less than $\delta_S(q, k+1)$. Therefore, we can conclude that the algorithm returns the correct result for the last case. □

3.2.3 Sampling of Potential Queries

We have shown that for a particular query $q \in \mathbb{U}$, if $\text{MRED}(q, k+1)$ has a definite value, then MAET is guaranteed to produce a correct k-NN query result whenever termination parameter $t \geq \text{MRED}(q, k+1)$; otherwise, MAET returns the correct query result regardless of the value of t. In this subsection, we further show how that t can be chosen through sampling of potential queries for MAET, in order to correctly answer a desired proportion of potential queries with high probability. We will first describe the method for determining such a value of t, and then justify the method by providing a technical lemma. We call this method *SamplingGED*.

Let $\mathfrak{D}(\mathbb{U})$ be a list of $|\mathbb{U}|$ real values. For each point $q \in \mathbb{U}$, a value $a = \text{MRED}(q, k+1)$ exists in list $\mathfrak{D}(\mathbb{U})$. If $\text{MRED}(q, k+1)$ has no definite value, $a = -1$. The method *SamplingGED* can be stated as follows:

1. Sample a set $\mathbb{U}_1$ uniformly at random without replacement from $\mathbb{U}$.
2. Compute list $\mathfrak{D}(\mathbb{U}_1)$.
3. Choose termination parameter $t > 0$ to be the smallest value for which at least a desired proportion $0 < \eta_1 \leq 1$ of the list entries $\mathfrak{D}(\mathbb{U}_1)$ are no greater than t.

Correspondingly, we let η denote the proportion of values in $\mathfrak{D}(\mathbb{U})$ which are no greater than t. According to Theorem 1, if we provide this choice of t to MAET, the proportion of queries in $\mathbb{U}$ (and $\mathbb{U}_1$) for which MAET is guaranteed to answer correctly is η (and η_1 respectively). Now, we shall establish the relationship between the true success rate η and the sample success rate η_1.

Lemma 2 *Let $\mathbb{N}$ be a list of n real numbers and let $\mathbb{N}_1$ be a list of n_1 elements sampled uniformly at random from list $\mathbb{N}$ without replacement. Given a threshold $t \in \mathbb{R}$, let m and m_1 refer to the number of elements in $\mathbb{N}$ and $\mathbb{N}_1$, respectively, that are no greater than t. Take η and η_1 to refer to the proportion of those elements within $\mathbb{N}$ and $\mathbb{N}_1$, respectively. For any real number $\phi \geq 0$, we have $\mathsf{Pr}[|\eta_1 - \eta| \geq \phi] \leq 2e^{-2\phi^2 n_1}$.*

Proof: Given a threshold $t \in \mathbb{R}$, the number m_1 of sampled items a in list $\mathbb{N}_1$ satisfying $a \leq t$ follows the hypergeometric distribution $\mathscr{H}(m_1 | n, n_1, m)$ with expectation $\mathsf{E}[m_1] = n_1 \frac{m}{n}$. In [Chvátal, 1979], Chvátal showed that the random variable m_1 satisfies both $\mathsf{Pr}[m_1 \geq \mathsf{E}[m_1] + \phi n_1] \leq e^{-2\phi^2 n_1}$ and $\mathsf{Pr}[m_1 \leq \mathsf{E}[m_1] - \phi n_1] \leq e^{-2\phi^2 n_1}$. Both inequalities can be combined to provide the following convenient statement:

$$\begin{aligned}
&\mathsf{Pr}[|\eta_1 - \eta| \geq \phi] \\
&= \mathsf{Pr}\left[\left|\frac{m_1}{n_1} - \frac{m}{n}\right| \geq \phi\right] \\
&= \mathsf{Pr}\left[\left|\frac{m_1}{n_1} - \frac{\mathsf{E}[m_1]}{n_1}\right| \geq \phi\right] \\
&= \mathsf{Pr}[|m_1 - \mathsf{E}[m_1]| \geq \phi n_1] \leq 2e^{-2\phi^2 n_1}
\end{aligned}$$

□

When taking $\mathbb{N} = \mathfrak{D}(\mathbb{U})$ and $\mathbb{N}_1 = \mathfrak{D}(\mathbb{U}_1)$, Lemma 2 states that the probability of the unknown success rate η deviating from the sample success rate η_1 by more than $\phi \geq 0$ is at most $2e^{-2\phi^2 |\mathbb{U}_1|}$. That is, the probability of the true success rate being significantly less than the sample success rate vanishes as the sample size grows. Therefore, for a reasonable

```
Algorithm MAET+ (query $q$, neighborhood size $k$, termination parameter $t$)
 3: $P \leftarrow \emptyset$,  $\lambda_e \leftarrow \infty$.    // $\lambda_e$ is the estimate of $\lambda$.
 7: $P \leftarrow P \cup \{(v, d(q,v))\}$,  $\lambda_e \leftarrow \min\{\lambda_e, d(q,v)/\beta'\}$.
13: $k_1 \leftarrow |R^{\leq}_{P_v}(q, \lambda_e \cdot \beta')|$.
```

Figure 3.5 The description of Algorithm MAET+. Only those steps for which changes have been made are shown; all other steps are identical to those of Algorithm MAET (see Figure 3.3).

value of $\phi \geq 0$, we are able to choose a global value of t (through method *SamplingGED*) for MAET to correctly answer a certain proportion $(\eta_1 - \phi)$ of potential queries with high probability.

As an illustrative example, let us suppose that in Step 1 of *SamplingGED* we sample a set $\mathbb{U}_1$ with size $|\mathbb{U}_1| = 1000$, and then determine a value of t in Step 3 by choosing a desired success rate $\eta_1 = 95\%$ for this sample. Then the probability of the unknown success rate η being no less than 90% is at least 98.6% ($\phi = 0.05$). That is, with probability at least 98.6%, MAET is able to correctly answer at least 90% of potential queries with this choice of t.

3.2.4 Dynamic Estimation of Lower-Bounding Ratio

In practice, it might be difficult for the user to determine a value of the lower-bounding ratio λ with which MAET can perform well. If the ratio is too small, a larger candidate set can be expected; if it is too large, the accuracy of the query result might be low. In this subsection, we extend Algorithm MAET to be able to dynamically estimate a good value of λ. The extension, called MAET+, is described in Figure 3.5. The only change made in the extension is that observed distance values are used to dynamically estimate the lower-bounding ratio λ. More precisely, after each candidate v and lower-bounding distance $\beta' = d'(q,v)$ are retrieved from the underlying index, the ratio of the target distance $\beta = d(q,v)$ to β' is computed, and the smallest ratio encountered so far is stored in λ_e as the estimate of λ. The remainder of the steps are the same as in MAET. Note that for simplicity and clarity, we have decided not to explicitly include all of the algorithmic details for cases where early termination is never performed (such as when $\beta' = 0$). As we will show in Section 3.3, the performance of MAET+ is competitive with that of MAET.

3.3 Experimental Results

In this section we present the results of our experimentation with MAET and its variants. All algorithms were implemented in C++. The experiments were run on a desktop computer with a quad-core 2.4GHz CPU and 8GB of memory.

3.3.1 Experimental Framework

Before presenting our experimental results, let us first describe the data sets, methodology, and underlying indexes used in our experiments.

Data Sets. Three publicly-available data sets were considered for our experiments, two with large representational dimension, and one containing a large number of objects. The Amsterdam Library of Object Images (ALOI) [Geusebroek et al., 2005] consists of 110250 images of 1000 small objects taken under different conditions, such as differing viewpoints and illumination directions. The images are represented by 641-dimensional feature vectors based on color and texture histograms (for a detailed description of how the vectors were produced, see [Boujemaa et al., 2001]). The MNIST data set [LeCun et al., 1998] consists of 70000 images of handwritten digits from 500 different writers, with each image represented by 784 gray-scale texture values. The Forest Cover Type set (FCT) [Bache and Lichman, 2013] consists of 581012 data points, with each representing a 30×30 square meter area of forest. Each point is represented by 54 attributes, associated with elevation, aspect, slope and other geographical characteristics.

Methodology. From each data set considered, we randomly selected 1000 objects to serve as queries. Three values of the query result size were considered: $k = 10, 30$ and 100. Since we observed comparable results for these three values of k, we chose to report only the results for $k = 100$. Three evaluation parameters were measured: average accuracy, average number of candidates, and average running time. The number of candidates equals the number of target distance evaluations needed by the algorithms to return the query result. The average running time is shown as a proportion of the time needed for a sequential search of the entire data set. For one query q, the accuracy of its k-NN query result is defined as

$$\frac{|\{v \in Y \mid d(q,v) \leq \delta_S(q,k)\}|}{k},$$

where Y denotes the k-NN query result of q ($|Y| = k$). In order to generate lower-bounding distance functions d', the Karhunen-Loève Transform (KLT), also known as Principal Component Analysis (PCA), was used to project the original feature space of a data set to new feature spaces with reduced dimensions. The number of reduced dimensions for the new feature space will be denoted as D'. Unless stated otherwise, Euclidean distance was used to compute both target distances and lower-bounding distances. The values of t were chosen in $\{1, 2, 4, 8, 16, 32\}$, so as to cover as wide a range of accuracies as possible.

Underlying Indexes. For our experimentation, two underlying indexes were considered for producing candidates on the fly, the X-tree (for exact queries) and SASH (for faster approximate queries). We integrated the method presented in [Hjaltason and Samet, 1995] with the X-tree to produce candidates incrementally. We chose the X-tree for the reason that it can produce nearest neighbors incrementally, which ensures a fair comparison of MAET+ and SK. Moreover, as Seidl and Kriegel stated in [Seidl and Kriegel, 1998], the number of candidates, which can be used as an indicator of the total cost, does not depend on the underlying index structure.

The SASH is by nature an approximate index structure. Since it is known to be efficient in handling queries with respect to high-dimensional distance functions, it serves as an example of how MAET can further benefit from an efficient underlying index structure. When the SASH was used, candidates were produced by performing successive k-NN queries with the value of k doubled at each iteration, until a sufficient number of candidates were produced.

3.3.2 MAET vs MAET+

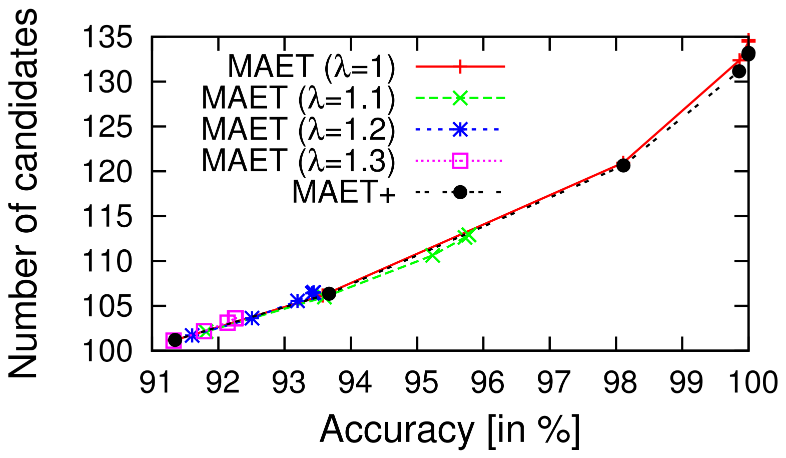
For the first set of experiments, we compared the performance of dynamic estimation of the lower-bounding ratio λ (MAET+) against that of fixed choices of λ (MAET). All three data sets were used here. The number of reduced dimensions D' was chosen to be 5, 30 and 30 for FCT, ALOI and MNIST, respectively. The X-tree was used as an appropriate underlying index for this set of experiments.

The results are shown in Figure 3.6. The CPU time, which follows almost exactly the same trend as the number of candidates in this example, is not shown in this figure. From the results, we can find that MAET+ is consistently competitive with MAET, which is provided with several fixed choices of λ. For MAET, the choice of λ that leads to a given accuracy value is different across the various data sets. For example, the best values of λ to achieve 95% average accuracy are 1.1, 1.2 and 1.3 for FCT, ALOI and MNIST, respectively. For this reason, for all the remaining experiments except those appearing in Section 3.3.5, we omit the experimental results of MAET, and instead compare only MAET+ with SK.

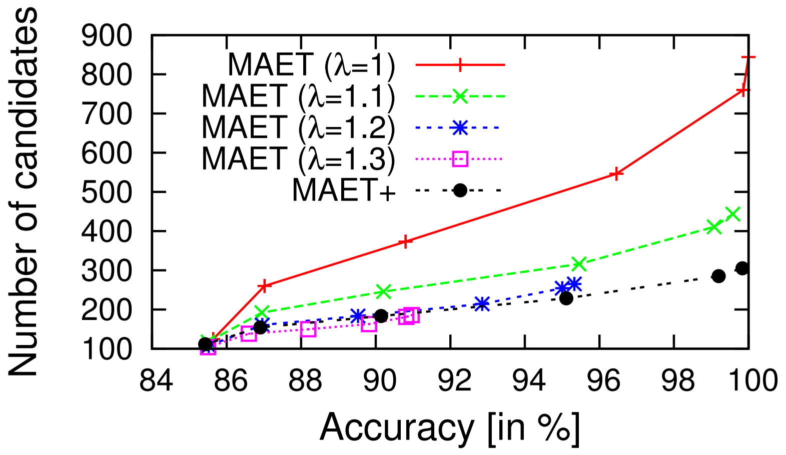
3.3.3 MAET+ vs SK

Two sets of experiments were conducted for comparing the performance of MAET+ with that of SK. In the first set, we compared their performances in dealing with one target distance function d with respect to different lower-bounding distance functions d'. In the second set, we compared their performances in handling adaptive target distance functions with respect to one lower-bounding distance function. The X-tree was used for the comparison.

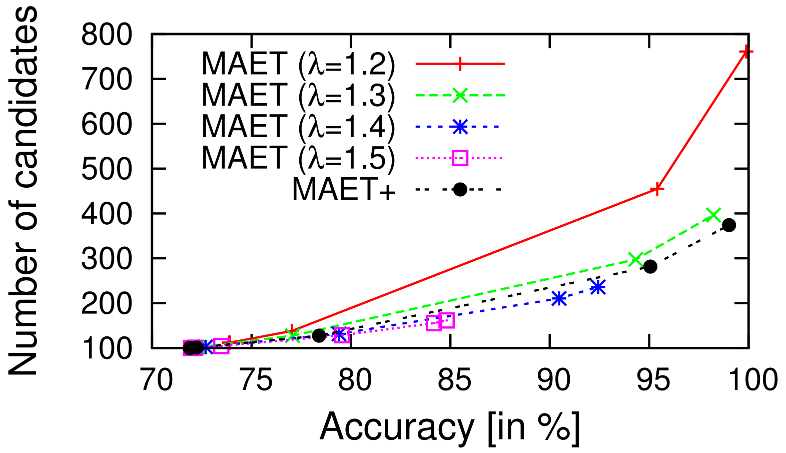
Single Target Distance Function. In this set of experiments, MAET+ and SK were compared for one choice of target distance function, but with several choices of lower-



(a) FCT ($D' = 5$)

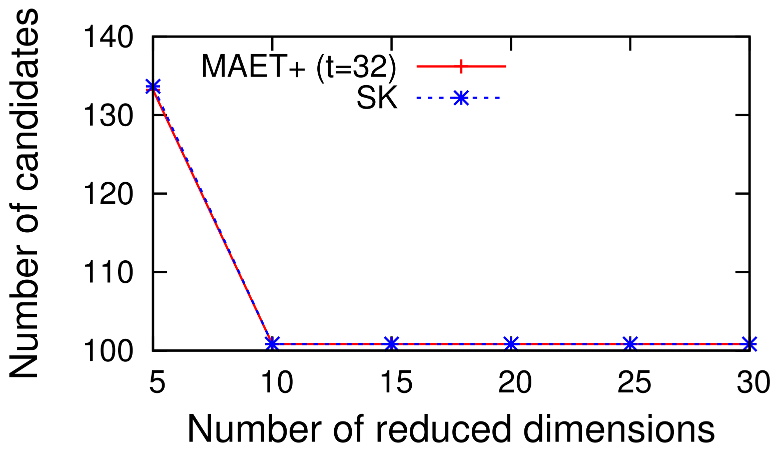


(b) ALOI ($D' = 30$)



(c) MNIST ($D' = 30$)

Figure 3.6 MAET vs MAET+ on FCT, ALOI and MNIST.



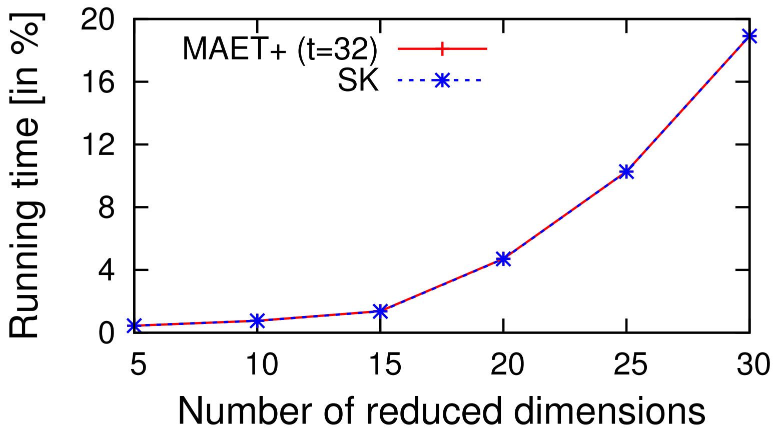


Figure 3.7 MAET+ vs SK on FCT (single d). The accuracy of MAET+ with $t = 32$ is more than 99%.

bounding distance functions. All three data sets were used for the comparison. For each data set, 6 values for the number of reduced dimension D' were chosen, each corresponding to a lower-bounding distance function for which an X-tree was created. The results for FCT, ALOI and MNIST are displayed in Figure 3.7, 3.8 and 3.9, respectively.

From the results for all three data sets, we can observe that with the increase of the number of reduced dimensions, the number of candidates monotonically decreases for both MAET+ and SK, whereas the running time (shown as a percentage of the sequential search time) does not, or may even monotonically increase. This is due to the effect of dimensionality on the performance of X-tree indexing.

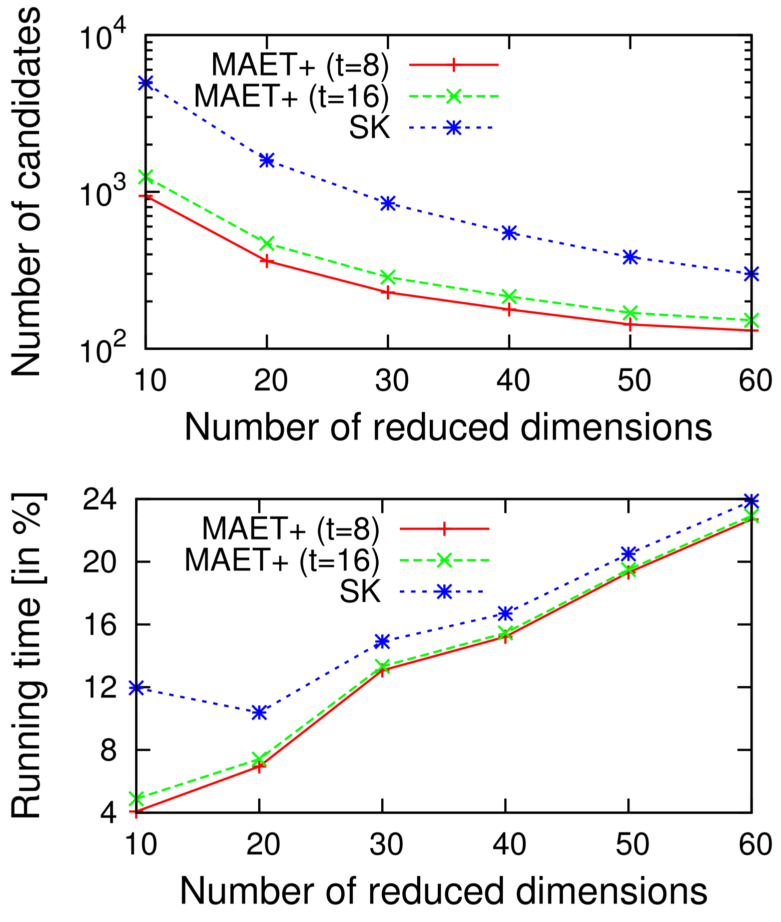


Figure 3.8 MAET+ vs SK on ALOI (single d). The accuracies of MAET+ are approximately 93% and 99% for $t = 8$ and 16, respectively.

For the FCT data set, the performances of MAET+ and SK are similar. However, for the other two data sets, MAET+ shows improvement over SK while sacrificing little in accuracy, with respect to all choices of lower-bounding distance functions. The improvement is especially evident when the number of reduced dimensions is small — that is, when the lower-bounding distance function is not a very good approximation of the target distance function. As an example, for the experiment on MNIST with $D' = 10$, the number of candidates and the running time are reduced by approximately 93% and 86% respectively, with an approximate loss of accuracy of only 1%.

Comparing the results across the three data sets for a common value of D' (for example, $D' = 10$), we find that MAET+ obtains the largest improvement over SK on

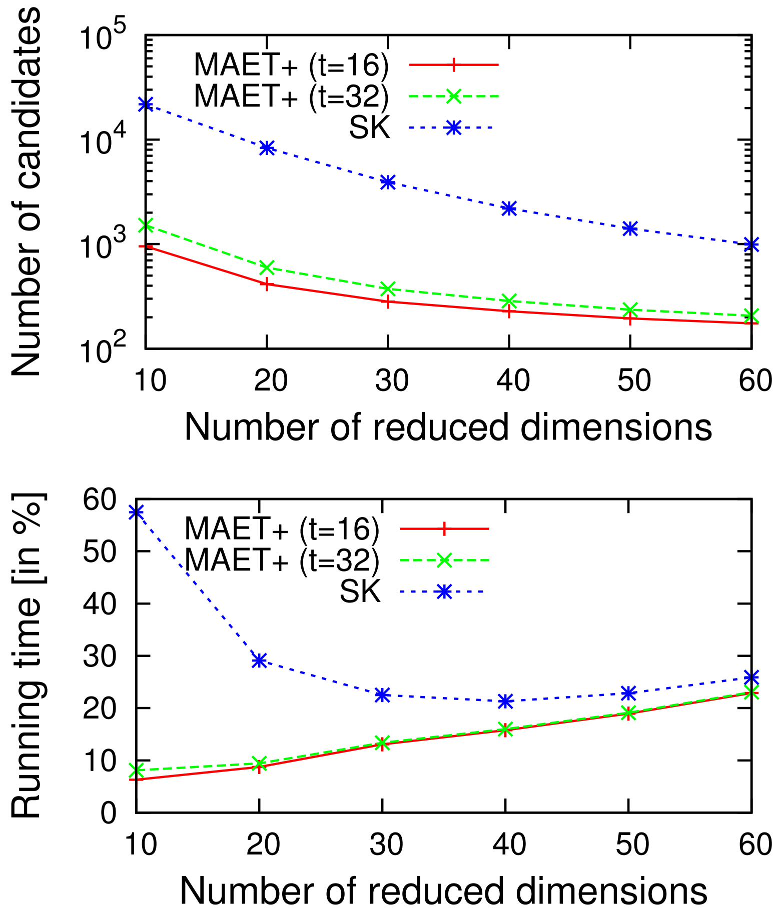


Figure 3.9 MAET+ vs SK on MNIST (single d). The accuracies of MAET+ are approximately 95% and 99% for $t = 16$ and 32, respectively.

MNIST; however, on FCT, no improvement is obtained. This can be explained by differences in the approximation qualities of d' to d across these data sets. Compared to SK, our method MAET+ is able to make further predictions of future distances, due to both tests of dimensionality and the dynamic estimation of lower-bounding ratio. Therefore, more improvement of MAET+ over SK can be expected as the approximation quality of d' to d worsens.

Adaptive Target Distance Functions. In this set of experiments, we compared the performances of MAET+ and SK regarding their ability to handle adaptive target distance functions. Here, all three data sets were used. Three target distance functions were considered: Euclidean distance (L_2), weighted Euclidean distance (WL_2) and Manhattan

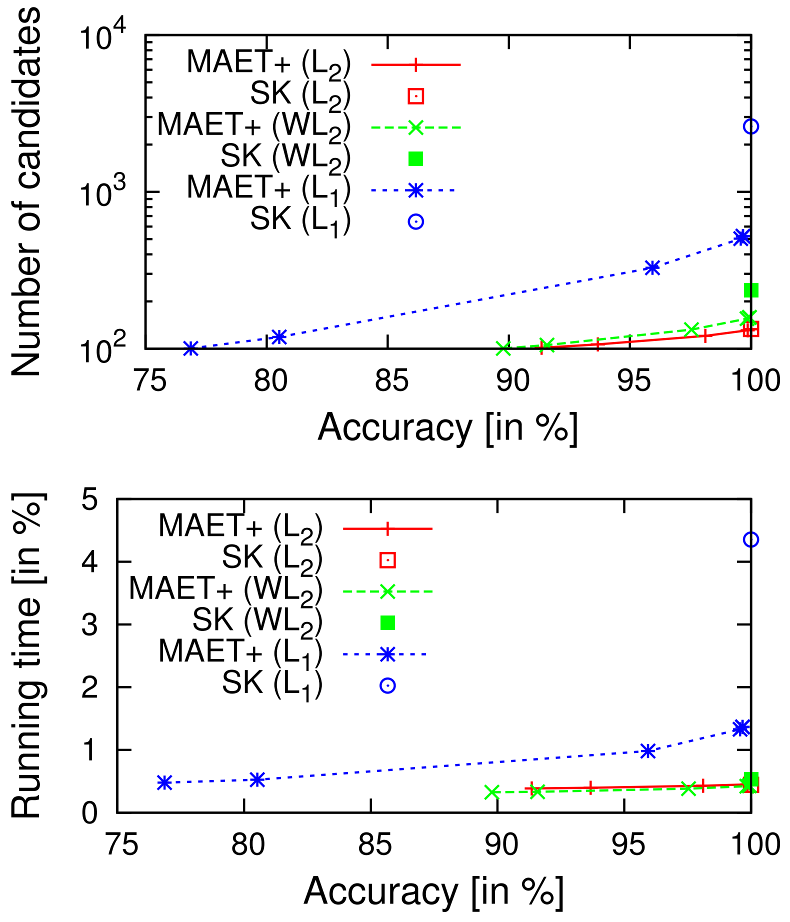


Figure 3.10 MAET+ vs SK on FCT (adaptive d). $D' = 5$.

distance (L_1). Euclidean distance was used for d'. In order to ensure the lower-bounding property, every weight used in WL_2 was randomly selected uniformly from the range $[1,2]$. The number of reduced dimensions D' was chosen to be 5 for FCT, 20 for ALOI, and 40 for MNIST, the values for which SK achieved its smallest execution times using the L_2 distance. The results are shown in Figure 3.10, 3.11 and 3.12 for FCT, ALOI and MNIST, respectively. For the sake of readability, the running time of SK for the L_1 distance function is not displayed in Figure 3.11 or 3.12, since the execution time cost of SK was approximately 1.91 and 1.75 times larger than that of sequential search, respectively.

From the results for ALOI and MNIST, we can find that MAET+ outperforms SK while still maintaining very high accuracy for all three target distance functions considered. For example, while with ALOI, Algorithm MAET+ suffers a loss in accuracy

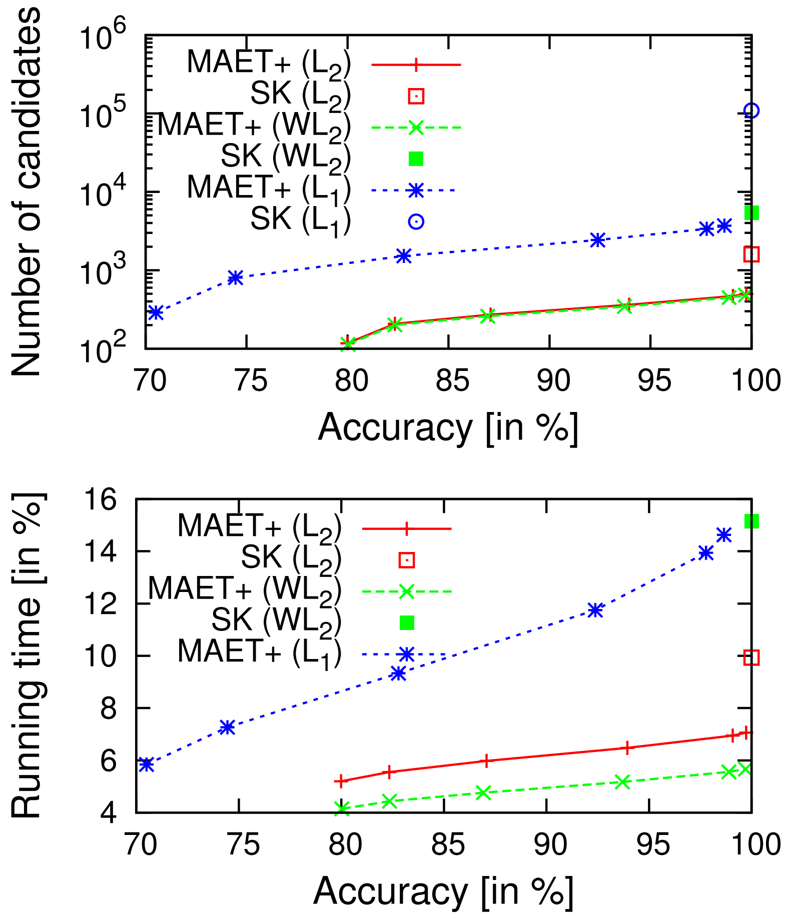


Figure 3.11 MAET+ vs SK on ALOI (adaptive d). $D' = 20$.

of approximately 1%, the numbers of candidates are reduced by approximately 68%, 91% and 96% for L_2, WL_2 and L_1, respectively. Correspondingly, the execution times are reduced by approximately 29%, 63% and 92%, respectively. The improvement is not so evident in the results for FCT. Specifically, with a loss in accuracy of approximately 1%, MAET+ examines almost the same number of candidates as SK for L_2, with comparable execution times. For WL_2 and L_1 respectively, the improvement is approximately 33% and 80% in the number of candidates, and 20% and 68% in the running time.

We note that when handling queries with respect to L_1 on ALOI and MNIST, Algorithm SK completely fails, in that it must refine almost every object in the data set to get the correct query result. In contrast, MAET+ still performs well with respect to both the total cost and the accuracy.

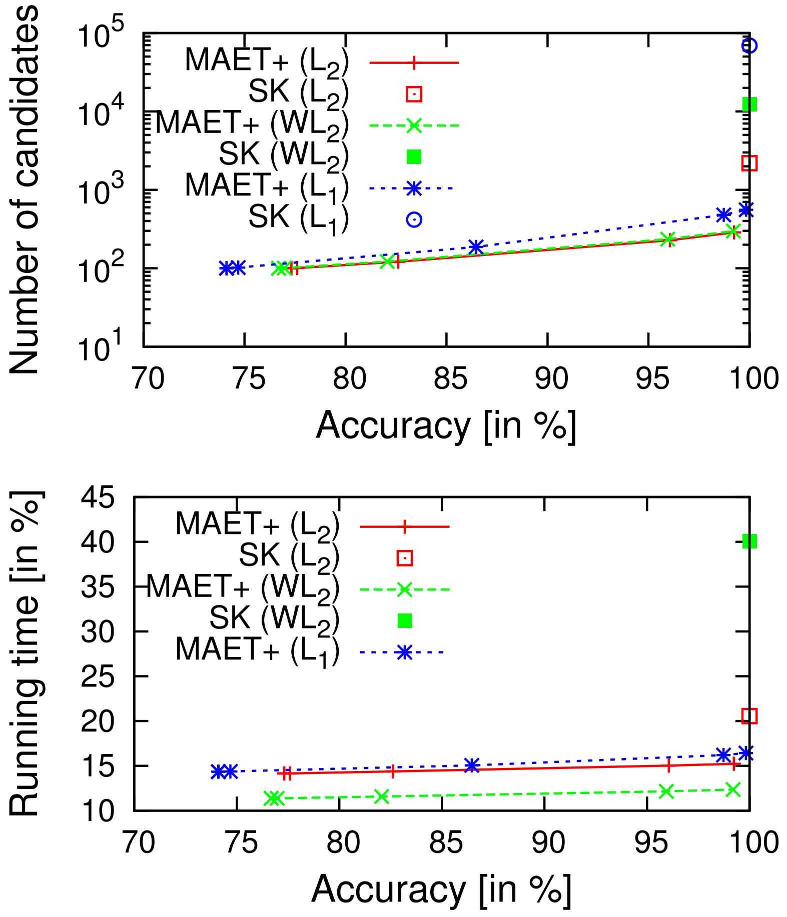
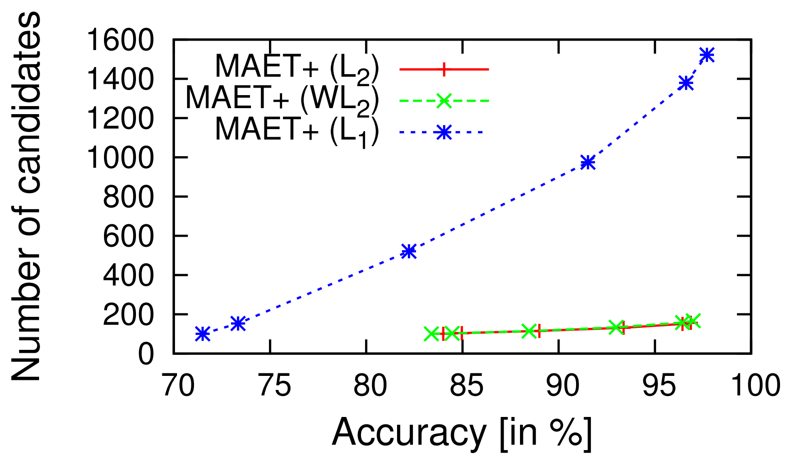


Figure 3.12 MAET+ vs SK on MNIST (adaptive d). $D' = 40$.

3.3.4 Further Evaluation of MAET+

For this set of experiments, we evaluated the performance of MAET+ using the SASH as the underlying index. Specifically, we tested the performance of MAET+ in handling adaptive target distance functions with respect to one lower-bounding distance function. Two data sets were used for the evaluation, ALOI and MNIST. As FCT is a sparse data set with only 54 dimensions in total, the X-tree performs very efficiently for this set, eliminating the need to use a fast approximate similarity search index such as the SASH; for this reason, FCT was omitted from this experiment. Three target distance functions were considered: Euclidean distance (L_2), weighted Euclidean distance (WL_2) and Manhattan distance (L_1). For both ALOI and MNIST, the number of reduced dimensions D' was chosen to be 60. Euclidean distance was used for computing lower-bounding distances.



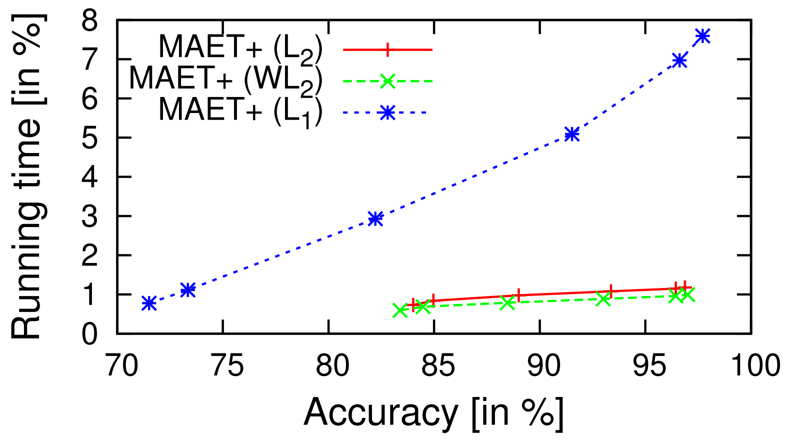
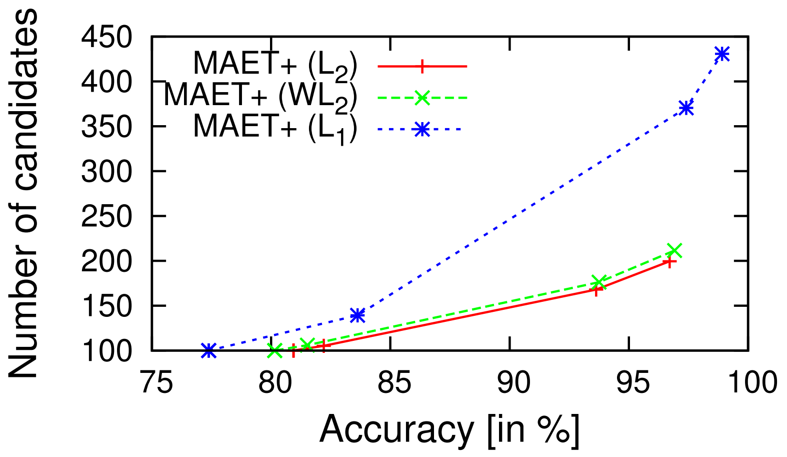


Figure 3.13 The results of MAET+ on ALOI ($D' = 60$). SASH was used as the underlying index structure, with approximately 93% average accuracy.

Again, every weight used in WL_2 was randomly selected from the range $[1,2]$ to ensure the lower-bounding property. The results are displayed in Figure 3.13 for ALOI, and Figure 3.14 for MNIST.

From the results for both ALOI and MNIST, we observe that significant speedups (of roughly two order of magnitudes) are achieved by MAET+ as compared to sequential search, all while maintaining very high accuracy, for the L_2 and WL_2 distances. For the L_1 distance, MAET+ obtains a speedup of more than one order of magnitude for both ALOI and MNIST. Moreover, for all three target distance functions, the number of candidates never exceeds 1.4% and 0.7% of the size of ALOI and MNIST, respectively.



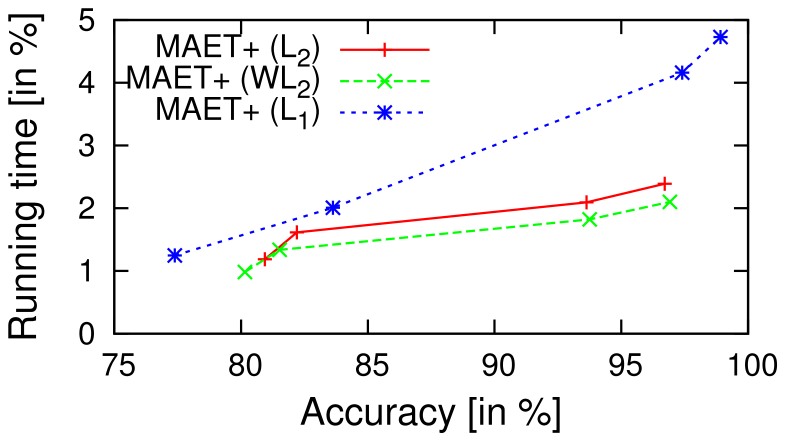


Figure 3.14 The results of MAET+ on MNIST ($D' = 60$). SASH was used as the underlying index structure, with approximately 94% average accuracy.

3.3.5 Theoretical Bounds and Practical Performance

The theoretical analysis in Section 3.2.3 indicates that through method *SamplingGED*, we are able to choose a value of t for MAET to correctly answer a desired proportion of potential queries with high probability. In this subsection, we will show that the performance of our method in practice is considerably better even than what the theoretical bounds indicate, through a set of experiments on all three available data sets.

In our experiment, we considered all data objects as potential queries. The value of k was chosen to be 100. In order to obtain a lower-bounding distance function, KLT was used to project the original feature space of each data set to a lower-dimensional feature space. For all three data sets, the number of reduced dimensions of the new feature space

Table 3.1 Performances of MAET with Respect to Different Choices of t (Each Corresponds to a Value of η_1)

	FCT		ALOI		MNIST	
η_1	θ	#(Cand)	θ	#(Cand)	θ	#(Cand)
0.1	0.99	100.1	0.45	1604	0.99	11686
0.2	0.99	100.2	0.67	2438	1.00	16460
0.3	0.99	100.2	0.84	3176	1.00	19574
0.4	0.99	100.2	0.95	3762	1.00	21489
0.5	0.99	100.3	0.99	4231	1.00	22314
0.6	0.99	100.3	0.99	4541	1.00	22644
0.7	0.99	100.4	0.99	4703	1.00	22734
0.8	0.99	100.5	1.00	4756	1.00	22763
0.9	0.99	100.6	1.00	4762	1.00	22768
1.0	1.00	100.8	1.00	4762	1.00	22769

was chosen to be 10. Euclidean distance was used for both the target distance function and the lower-bounding distance function. In order to avoid the introduction of error by overestimating the lower-bounding ratio λ, we set $\lambda = 1$. By applying *SamplingGED* on each data set, we picked 10 values for t, corresponding to 10 desired proportions η_1 at intervals of 0.1, from 0.1 to 1. The sample size was chosen to be 1000. The X-tree was used as an underlying exact index structure. For this set of experiments, we measured the proportion of queries for which MAET returns the correct query result; this proportion is denoted by θ in the results displayed in Table 3.1. We also measured the average number of candidates as an indicator of the total cost needed by MAET to return the query results (see Table 3.1).

From the results, we can see that the proportion of queries with correct results are much better than what the theoretical bounds would indicate, across the different choices

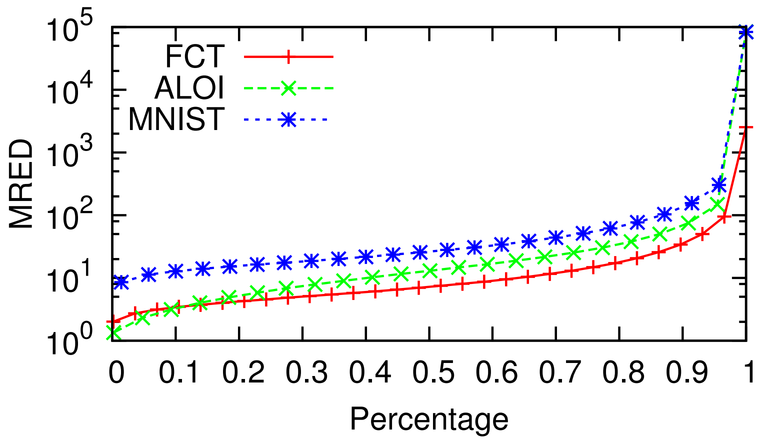


Figure 3.15 Accumulated distributions of MRED for FCT, ALOI and MNIST.

of t. For example, for the choice of t corresponding to the value of $\eta_1 = 40\%$, over all three data sets, MAET returns the correct query result for more than 90% of the queries for all three data sets; however, the theoretical bound is much more conservative, indicating that MAET would (with high probability) correctly answer only at least 35% of all the queries. This discrepancy between theoretical and practical performance shows that while the theoretical bound can serve as a guide when setting the termination parameter t, in practice we should choose values that are substantially lower (less conservative) than what is indicated by the bound.

Another observation which draws our attention is that the performance of MAET varies greatly with respect to the three data sets. Specifically, in order to answer all queries correctly, MAET must produce 100.8, 4756 and 16460 candidates on average for FCT, ALOI and MNIST, respectively. These numbers of candidates account for approximately 0.017%, 4.3% and 23.5% of the respective data set sizes. Such variance in the performances of MAET for FCT, ALOI and MNIST can be explained through the differences in their MRED values, which are displayed in Figure 3.15. Essentially, with an increase in overall MRED value, smaller proportional increases in distance lead to greater changes in neighborhood sizes. Consequently, it is more difficult to find a lower-bounding distance function that serves as a good approximation of the target distance function, since a small error introduced by approximation could significantly change the original ordering of the neighborhood items. In our example, the number of reduced dimensions for computing lower-bounding distances was chosen to be 10 for all three data sets. Clearly, the resulting approximation of the lower-bounding distance function is sufficient for FCT; however, for ALOI and MNIST, a larger number of reduced dimensions should be chosen in order to achieve better performance.

3.4 Conclusion

We have presented a novel multi-step k-NN search algorithm for approximate similarity search, MAET, which utilizes tests of local intrinsic dimensionality to guide the search decisions. Theoretical analysis shows that for a particular query q, if the expansion dimension $\mathrm{MRED}(q, k+1)$ has a definite value, then the correctness of MAET can be guaranteed whenever its termination parameter $t \geq \mathrm{MRED}(q, k+1)$; otherwise, its correctness can be guaranteed regardless of the value of t. We have further shown that given a success rate target, sampling can be used to determine a global value of the termination parameter t so that MAET correctly answers the desired proportion of potential queries with high probability. A variant of MAET has also been developed, MAET+, which utilizes history information of distance values to dynamically estimate the value of the lower-bounding ratio λ, instead of treating it as an input parameter.

Experimental results show that MAET+ is consistently competitive with MAET. Compared to SK, our approach MAET+ is able to obtain significant improvements for both the number of candidates and the running time, while losing very little in the accuracy of the query results. The experimental results also demonstrate that MAET+ is able to offer practical speedups with respect to handling high-dimensional distance functions and adaptive distance functions.

CHAPTER 4

EFFECTIVE AND EFFICIENT ALGORITHMS FOR FLEXIBLE AGGREGATE SIMILARITY SEARCH IN HIGH DIMENSIONAL SPACES

In this chapter, we present the work on flexible aggregate similarity search (FANN), which is a generalization of aggregate similarity search (AggNN). Given a group of query objects Q, AggNN aims to retrieve the k objects from the database S that are most highly ranked with respect to Q, where the underlying similarity measure is defined as an aggregation (usually *sum* or *max*) of distances between the retrieved objects and every query object in Q. FANN generalizes AggNN by calculating aggregate distances only over subsets of Q. More precisely, FANN aims to retrieve the k objects which are most similar to a subset of group query Q with size $\phi|Q|$, for some target proportion $0 < \phi \leq 1$. When $\phi = 1$, FANN is equivalent to AggNN.

Two exact solutions, R-tree and List, were proposed in [Li et al., 2011b] for the FANN problem. In general, Algorithm R-tree can only be applied in low-dimensional settings (up to 10 dimensions), while Algorithm List may incur high computational costs [Li et al., 2011b]. Two approximation methods, ASUM and AMAX, were also proposed by Li et al. [Li et al., 2011b], the former for the *sum* variant of FANN, and the other for the *max* variant. The authors showed that ASUM and AMAX are guaranteed to produce the query results within a constant approximation factor of the true (aggregate) k-NN distance.

Compared to distance approximation ratios, the recall rate (or accuracy) is generally a more meaningful measure of the quality of similarity query results. This is especially true in high dimensional settings, due to the *curse of dimensionality*. One way in which the curse manifests itself is a concentration of distance values around the mean object-to-object distance. For this reason, a query result with a small distance approximation ratio may not necessarily achieve a high recall.

Motivated by this observation, we propose the FADT (Flexible Aggregation through Dimensional Testing) family of approximation algorithms for the FANN problem in high dimensional spaces, that possess great efficiency while returning query results with great quality. We design our algorithms by adopting a multi-step search strategy [Seidl and Kriegel, 1998] (see Section 3.1 in Chapter 3), together with tests of a measure of intrinsic dimension, the *generalized expansion dimension* [Houle et al., 2012a] (see Section 2.2 in Chapter 2), for possible early termination [Houle et al., 2012b] (see Section 3.2 in Chapter 3). The main contributions are:

- Two approximation algorithms for the FANN problem, one for the *sum* variant and the other for the *max* variant. For both variants, several enhancements are also presented.
- A theoretical analysis of our methods, showing conditions under which an exact result can be guaranteed. We also show conditions under which approximate result can be guaranteed for a variable distance approximation ratio.
- An extensive experimental evaluation, showing that our algorithms are able to produce query results with good accuracy, while at the same time being very efficient in comparison with List, ASUM, and AMAX. Specifically, compared to List, our algorithms are typically more than one order of magnitude faster, with a loss of accuracy on the order of 10%. While ASUM and AMAX are also quite efficient, they often fail to produce query results with high accuracy.

The remainder of this chapter is organized as follows. In Section 4.1, a formal definition of the FANN problem is presented. Section 4.2 discusses Algorithm List, ASUM and AMAX in detail. The two approximation algorithms, together with their theoretical analysis, are presented in Sections 4.3 and 4.4. Section 4.5 presents experimental results that contrast the performance of our methods with that of existing methods. The discussion is concluded in Section 4.6.

4.1 Problem Description

Before formally presenting the FANN problem, let us first introduce some needed notation. Let $\mathbb{U}$ be a data domain with a distance metric $d(u, v)$ defined for any two objects $u, v \in \mathbb{U}$. Let $S \subseteq \mathbb{U}$ denote the set of objects in the database. For any object set $X \subseteq \mathbb{U}$ and any object $v \in \mathbb{U}$, let $N_X(v,k)$ denote the set of k nearest neighbors of v within X with respect to distance metric d. The AggNN aggregate distance measure $D(X,v)$ is defined as

$$D(X,v) = g(d(x_1,v),\ldots,d(x_{|X|},v)),$$

where g is an aggregation function (*sum* or *max*), and $x_i \in X$ for $i = 1,\ldots,|X|$.

FANN Problem. Given a group of query objects $Q \subseteq \mathbb{U}$ with size $|Q| = m > 1$, a distance metric d, an aggregation function g, a target proportion $0 < \phi \leq 1$, and a target neighborhood size k, a FANN query returns the k most similar objects to Q from the database S, where the similarity distance $D_\phi(Q,v)$ is defined as

$$D_\phi(Q,v) = D(N_Q(v,\lceil\phi m\rceil),v).$$

The similarity distance $D_\phi(Q,v)$ denotes the aggregate distance from v to all members of its $\lceil\phi m\rceil$ neighborhood in Q. We call this similarity distance the 'ϕ-aggregate distance'. Since $D_1(Q,v) = D(Q,v)$, the AggNN problem can be viewed as a special case of the FANN problem whenever $\phi = 1$.

Note that the commonly-used aggregation function *min* is usually not considered for this problem, since its usefulness is rather limited in most application contexts: the returned results may relate strongly to only one of the objects of the group query Q, and not at all to the other members of Q. Moreover, the FANN problem would admit a relatively trivial solution if *min* were to be used as the aggregation function. One could easily determine the exact query result by retrieving from the database only the k nearest neighbors of each query object in Q, for a total of only km candidates.

For convenience, in the remainder of the chapter, we will assume that the value of ϕm is an integer. As was done in previous studies on AggNN and FANN, we will assume that the size of the group query Q (denoted by m) is small relative to the total database size $|S|$, with values up to hundreds. For simplicity, in this work we will only focus on the L_p space, which applies to most of the previous applications of AggNN and FANN. For previous studies on AggNN and FANN in general metric spaces, we refer the reader to Section 2.1.3 in Chapter 2.

4.2 Algorithm List, ASUM and AMAX

Before discussing Algorithm List, ASUM and AMAX, let us first introduce some additional notation. For any object set $X \subseteq \mathbb{U}$ and any group query $Q \subseteq \mathbb{U}$, let $N_X(Q,k)$ denote the set of k nearest neighbors of Q within X with respect to ϕ-aggregate distance D_ϕ. Ties are broken arbitrarily but consistently. Note that when $k = 0$, the neighborhood set $N_X(Q,k)$ would be empty. Let $\nu_X(Q,k)$ be the k-th nearest neighbor of Q in X with respect to D_ϕ. Ties in distance values are broken arbitrarily but consistently. Let $\delta_X(Q,k) = D_\phi(Q, \nu_X(Q,k))$ be the ϕ-aggregate distance from Q to its k-th nearest neighbor.

Algorithm List is described in Figure 4.1, whose design is based on the well-known *Threshold Algorithm* (TA) [Fagin et al., 2001] for list aggregation. Algorithm List retrieves candidates for the query result from the neighborhood of every object of group query Q in a round robin fashion, while maintaining a lower bound (γ) for the best possible ϕ-aggregate distance of all the unseen objects. The algorithm terminates when at least k of the objects visited have ϕ-aggregate distances no greater than the lower bound γ. Note that the algorithm requires that the neighborhood of every object of Q be generated incrementally. For most search indexes, this can be done with simple bookkeeping of the search state, as indicated in [Li et al., 2011b]. With the bookkeeping, the search for the $(j+1)$-th nearest neighbor can be resumed from the saved state after the termination of the search for the j-th nearest neighbor. Although the algorithm is capable of handling higher-dimensional data, the execution cost may be prohibitively expensive, as we shall see in Section 4.5.

The two approximation methods, ASUM and AMAX, are very simple. With the ASUM heuristic (see Figure 4.2), a candidate set is generated by taking the union of the k-NN sets for every query object in Q. The k best objects within the candidate set is then

```
Algorithm List (database $S$, group query $Q$, target proportion $\phi$, aggregation function $g$, neighborhood size $k$)
 1: Assume the existence of an index $I$ created with respect to $d$, together with a method
    getnext that uses $I$ to iterate through the nearest neighbor list of a target object.
 2: Let $P$ be an object set. $P \leftarrow \emptyset$.
 3: Initialize $\gamma_i \leftarrow 0$ for $i = 1, \ldots, m$.    // $m = |Q|$
 4: while TRUE do
 5:   for $i = 1 \rightarrow m$ do
 6:     $(v_i, \beta_i) \leftarrow I.getnext(q_i)$.    // $q_i \in Q,\ \beta_i = d(q_i, v_i)$
 7:     $\gamma_i \leftarrow \beta_i$.
 8:     $P \leftarrow P \cup \{v_i\}$.
 9:     if $|P| < k$ then
10:       Continue to Line 5.
11:     else if $|P| = |S|$ then
12:       return $N_P(Q, k)$.
13:     end if
        // $\gamma$ maintains the best possible $\phi$-aggregate distance of all the unseen objects
14:     $\gamma \leftarrow g($smallest $\phi m$ values from $\gamma_1, \cdots, \gamma_m)$.
15:     if $\delta_P(Q, k) \leq \gamma$ then
16:       return $N_P(Q, k)$.
17:     end if
18:   end for
19: end while
```

Figure 4.1 The description of Algorithm List.

```
Algorithm ASUM (database $S$, group query $Q$, target proportion $\phi$, aggregation function sum, neighborhood size $k$)
1: Let $P$ be an object set. $P \leftarrow \emptyset$.
2: for each $q \in Q$ do
3:   $P \leftarrow P \cup N_S(q, k)$.
4: end for
5: return $N_P(Q, k)$.
```

Figure 4.2 The description of Algorithm ASUM.

```
Algorithm AMAX (database $S$, group query $Q$, target proportion $\phi$, aggregation function max, neighborhood size $k$)
1: Let P be an object set. $P \leftarrow \emptyset$.
2: for each $q \in Q$ do
3:     Find the center C of minimum enclosing ball of $N_Q(q, \phi m)$.    // $m = |Q|$
4:     $P \leftarrow P \cup N_S(C, k)$.
5: end for
6: return $N_P(Q, k)$.
```

Figure 4.3 The description of Algorithm AMAX.

returned as the result. The AMAX heuristic (see Figure 4.3) is very similar to ASUM, the only difference being that AMAX first finds the center of the minimum enclosing ball (MEB) of the query set neighborhood $N_Q(q, \phi m)$, based at each query object $q \in Q$; AMAX then treats these centers as query objects. The center of the MEB of the query set neighborhood is the point which minimizes the maximum distance to any point in the query set neighborhood. Note that the center is not required to be one of the points of this neighborhood. When $\phi = 1$, AMAX is equivalent to the algorithm proposed in [Li et al., 2011a] for the *max* variant of AggNN. Li et al. [Li et al., 2011b] showed that ASUM returns a 3-approximate query result, and AMAX returns a $(1+2\sqrt{2})$-approximate query result.

They also proposed faster variants of ASUM and AMAX in which certain operations are performed only for a reduced, randomly sampled subset of the group query set Q. More precisely, in the variant of ASUM, Line 3 is performed only for a subset of Q, and in the variant of AMAX, Lines 3-4 are performed only for a subset of Q. An analysis is provided that shows that sampling variants achieve the original approximation ratio with a probability of success that depends on the size of the sample. Nevertheless, as we will show in Section 4.5, ASUM and AMAX cannot guarantee good practical performance in terms of the query result accuracy.

4.3 The *Sum* Variant of FANN

We now present our solution to the *sum* variant of the FANN flexible aggregate similarity search problem. Note that for the FANN problem, the aggregation functions *sum* and *avg* are equivalent. For this reason, and for the ease of description of our algorithm, we will formulate our solution in terms of the *avg* aggregation function.

Let us first introduce some additional notation. For any object set $X \subseteq \mathbb{U}$ and any group query $Q \subseteq \mathbb{U}$, let $R^{\leq}_{X}(Q, r)$ be the closed range set of objects $v \in X$ with distance to Q satisfying $D_{\phi}(Q, v) \leq r$, and let $R^{<}_{X}(Q, r)$ be the open range set of objects $v \in X$ with distance to Q satisfying $D_{\phi}(Q, v) < r$. For any object $u \in \mathbb{U}$, let $R^{\leq}_{X}(u, r)$ be the closed range set of objects $v \in X$ with distance to u satisfying $d(u, v) \leq r$, and let $R^{<}_{X}(u, r)$ be the open range set of objects $v \in X$ with distance to u satisfying $d(u, v) < r$. Let $\nu_{X}(u, k)$ be the k-th nearest neighbor of u in X with respect to d. Ties in distance values are broken arbitrarily but consistently. Let $\delta_{X}(u, k) = d(u, \nu_{X}(u, k))$ be the distance from u to its k-th nearest neighbor.

4.3.1 Algorithm FADT_AVG

The algorithm, FADT_AVG, is described in Figure 4.4. Whereas traditional search techniques usually progress through the exploration of a neighborhood of a single point in the domain (the query point), our aggregate similarity search method simultaneously explores the neighborhoods of m 'start points' M_i, each of which are formed by aggregating a subset of the group query Q. Iterating through the neighborhoods of the start points in round robin fashion, two pruning strategies are applied to reduce the search space. The first strategy utilizes a distance bound based on an auxiliary distance relative to the ϕ-aggregate distance. The second strategy applies a test of the intrinsic dimensionality (in the vicinity of Q) to determine whether early termination is possible.

For each query point $q_i \in Q$, the algorithm constructs one start point M_i. The point M_i is constructed as the geometric median of the neighborhood $N_Q(q_i, \phi m)$ of q_i within group query Q. The geometric median M_i is the point which minimizes the average of

Algorithm **FADT_AVG** (*database* S, *group query* Q, *target proportion* ϕ, *aggregation function* avg, *neighborhood size* k, *lower-bounding ratios* $\lambda_1, \ldots, \lambda_m$, *termination parameter* t)

```
1: Assume the existence of an index $I$ created with respect to $d$, together with a method
   *getnext* that uses $I$ to iterate through the nearest neighbor list of a target object.
2: Let $P$ be an object set. $P \leftarrow \emptyset$.
3: For each $q_i \in Q$, find the geometric median $M_i$ of $N_Q(q_i, \phi m)$.    // $m = |Q|$
4: Let $\alpha_i$ be the average of the largest $\phi m$ distances from $M_i$ to all the query points in
   $Q$, for $i = 1, \ldots, m$.
5: Let $q^*$ and $r^*$ be the center and radius of $MEB(Q)$, respectively.
6: Initialize $\gamma_i \leftarrow -\infty$ for $i = 1, \ldots, m$.
7: Initialize $k_{in} \leftarrow 0$ and $k_{out} \leftarrow k$.
8: **while** TRUE **do**
9:    **for** $i = 1 \rightarrow m$ **do**
10:      $(v_i, \beta_i) \leftarrow I.getnext(M_i)$.    // $\beta_i = d(M_i, v_i)$
11:      $\gamma_i \leftarrow \lambda_i \cdot \beta_i - \alpha_i$.    // lower distance bound
12:      $P \leftarrow P \cup \{v_i\}$.
13:      **if** $|P| < k$ **then**
14:         Continue to Line 9.
15:      **else if** $|P| = |S|$ **then**
16:         **return** $N_P(Q, k)$.
17:      **end if**
         // prune using distance bound
18:      $\gamma \leftarrow \max\{\gamma_1, \ldots, \gamma_m\}$.
19:      $k' \leftarrow |R_P^{\leq}(Q, \gamma)|$.
20:      **if** $k' \geq k$ **then**
21:         **return** $N_P(Q, k)$.
22:      **else**
         // prune using dimensional test
23:         $k_{in} \leftarrow |R_P^{\leq}(q^*, \gamma - r^*)|$.
24:         **if** $k_{in} > 0$ **then**
25:            $r_{in} \leftarrow \delta_P(q^*, k_{in})$.
26:            $r_{out} \leftarrow \delta_P(Q, k) + r^*$.
27:            $k_{out} \leftarrow |R_P^{\leq}(q^*, r_{out})|$.
28:            **if** $r_{in} > 0$ and $k_{in} \cdot (r_{out}/r_{in})^t < k_{out} + 1$ **then**
29:               **return** $N_P(Q, k)$.
30:            **end if**
31:         **end if**
32:      **end if**
33:   **end for**
34: **end while**
```

Figure 4.4 The description of Algorithm FADT_AVG.

its distances to all points in the full query set neighborhood $N_Q(q_i, \phi m)$. Note that M_i is not required to be one of the points of this neighborhood. Algorithm FADT_AVG then searches for query result candidates by sequentially scanning the neighborhood of every M_i (with respect to d) in a round robin fashion. Throughout the search process, the algorithm maintains an object set P, which stores all objects that have been visited so far. The algorithm also maintains m lower bounds $(\gamma_1, \ldots, \gamma_m)$ for the ϕ-aggregate distances of unseen objects to Q, with each γ_i relating to the neighborhood of the geometric median M_i, for $1 \leq i \leq m$. We let γ denote the maximum of these m lower bounds.

The algorithm terminates when one of the following three conditions hold:

- all objects of S have been visited (Line 15); or
- the exact multi-step search termination condition is fulfilled, in which at least k visited objects have ϕ-aggregate distances to Q that are no greater than the maximum lower bound γ (Line 20); or
- a termination condition relating to the intrinsic dimensionality in the vicinity of Q is fulfilled (Line 28).

The test of intrinsic dimensionality at Line 28 (the 'dimensional test') is based on a user-supplied estimate, in the form of a termination parameter $t > 0$. The test is performed on two closed balls centered at q^*, the center of minimum enclosing ball $\mathrm{MEB}(Q)$ of Q. The inner ball has r_{in} as its radius and contains k_{in} points of S, all of which are members of the aggregate query result; the outer ball has r_{out} as its radius and contains at least k_{out} points of S, where $k_{\text{out}} \geq k$. The aim of the dimensional test is to verify that no unvisited points could possibly be included in the outer ball, and therefore that the correct query result has been found.

An illustration of FADT_AVG is shown in Figure 4.5 for the group query $Q = \{q_1, q_2, q_3, q_4\}$ and the target proportion $\phi = 0.5$. In this example, the geometric medians M_1 and M_2 coincide, as do M_3 and M_4. The two shaded ball regions centered at $M_1 = M_2$ and $M_3 = M_4$ represent the area that has been searched in the space. If the two error areas (as indicated in the figure) contain no objects of S, all objects of the query result are

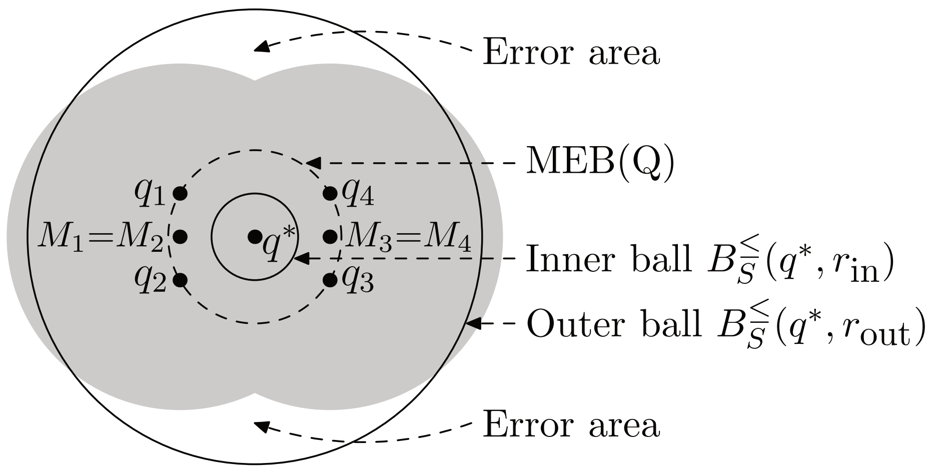

Error area
MEB(Q)
q_1
q_4
$M_1{=}M_2$
q^*
$M_3{=}M_4$
q_2
q_3
Inner ball $B^{\leq}_{S}(q^*, r_{\text{in}})$
Outer ball $B^{\leq}_{S}(q^*, r_{\text{out}})$
Error area

Figure 4.5 Illustration of FADT_AVG, with the group query $Q = \{q_1, q_2, q_3, q_4\}$ and the target proportion $\phi = 0.5$.

guaranteed to have been found. Our goal is to reach this conclusion via dimensional testing while visiting as few points as possible. To achieve this goal, the termination parameter t must satisfy certain conditions with respect to MED (see Figure 2.4 on Page 32 for the definition of MED), as we will see in the next subsection. These conditions will require that the aggregate distance be lower bounded by the pruning distance.

Lower-Bounding Relationship. The assumption that d is a distance metric implies that for every geometric median M_i ($1 \leq i \leq m$), every $q_j \in Q$ ($1 \leq j \leq m$) and any object $v \in \mathbb{U}$, the following triangle inequality holds:

$$d(M_i, v) \leq d(q_j, v) + d(M_i, q_j).$$

Together with the definition of ϕ-aggregate distance D_ϕ for the *avg* variant, we can derive that:

$$\phi m \cdot d(M_i, v) \leq \phi m \cdot D_\phi(Q, v) + \sum\nolimits_{q \in N_{Q}(v, \phi m)} d(M_i, q).$$

Let α_i be the average of the largest ϕm distances from M_i to all the points in Q (as defined in Line 4 of the description of FADT_AVG), we can derive the following lower-bounding relationship:

$$d(M_i, v) \leq D_\phi(Q, v) + \alpha_i.$$

Together with the dimensional test, Algorithm FADT_AVG utilizes this lower-bounding relationship to filter out candidate query result objects. The distance $d(M_i, v)$ from the candidate to geometric median M_i can be regarded as an approximation of $D_\phi(Q, v) + \alpha_i$. Clearly, the tighter the approximation, the greater the likelihood that FADT_AVG would perform well. In order to make the approximation as tight as possible, we introduce a lower-bounding ratio $\lambda_i \geq 1$ such that

$$\lambda_i d(M_i, v) \leq D_\phi(Q, v) + \alpha_i. \tag{4.1}$$

For each individual object $v \in \mathbb{U}$, consider the maximum possible value of the lower-bounding ratio $\lambda_i \geq 1$ for which the inequality holds. Ideally, λ_i should be set to the smallest such maximum so that the inequality holds for every object v.

Our algorithm requires the computation of the geometric median and the MEB. Unfortunately, for the computation of the geometric median, no exact method is known. However, the Weiszfeld algorithm [Chandrasekaran and Tamir, 1989] can be used to efficiently find the geometric median to any desired precision. For computing the MEB, both exact [de Berg et al., 1997] and approximate [Kumar et al., 2003] methods exist.

4.3.2 Analysis of FADT_AVG

We shall now give a formal theoretical analysis of FADT_AVG that proves the following:

- If $\mathrm{MED}(q^*, k+1)$ has no definite value, FADT_AVG returns the correct query result regardless of the value of t.
- Otherwise, if $t \geq \mathrm{MED}(q^*, k+1)$, FADT_AVG returns the correct query result.
- If $0 < t < \mathrm{MED}(q^*, k+1)$, FADT_AVG returns a $\sqrt[t]{\wp}$-approximate query result with $\wp = k_{\mathrm{out}} + 1$ at termination.

In the proofs that follow, we may assume without loss of generality that items sharing a common distance to Q under D_ϕ are ranked by Algorithm FADT_AVG consistently with the order of discovery of the points of S. Under this assumption, neighborhoods of the form $N_S(Q,k)$ are uniquely determined for any choice of $1 \le k \le |S|$. We begin by stating and proving several technical lemmas.

The first lemma justifies the correctness of our pruning distance bound. Specifically, it shows that γ is the smallest possible ϕ-aggregate distance for any object of S that is not already in the range set $R_P^{\leq}(Q,\gamma)$, where P is the set of all previously-visited objects.

Lemma 3 *At each iteration of the for loop of FADT_AVG, after the execution of Line 19, for any object* $v \in S \setminus R_P^{\leq}(Q,\gamma)$, *we have* $D_\phi(Q,v) \ge \gamma$.

Proof: For every geometric median M_i ($1 \le i \le m$), we note that objects $v \in S$ are retrieved from the underlying index in ascending order of $d(M_i,v)$. Then, any object $v \in S \setminus P$ must have

$$\begin{aligned} D_\phi(Q,v) &\ge \lambda_i d(M_i,v) - \alpha_i && \text{(from Inequality (4.1))} \\ &\ge \gamma_i && \text{(from Line 11).} \end{aligned}$$

Further, for any object $v \in S \setminus P$, we can derive $D_\phi(Q,v) \ge \max\{\gamma_i \mid 1 \le i \le m\} = \gamma$ (Line 18). Also, for any object $v \in P \setminus R_P^{\leq}(Q,\gamma)$, we know that $D_\phi(Q,v) > \gamma$. Therefore, we can conclude that for any object $v \in S \setminus R_P^{\leq}(Q,\gamma)$, we have $D_\phi(Q,v) \ge \gamma$. □

Following the first lemma, the second lemma further shows that the range set $R_P^{\leq}(Q,\gamma)$ is exactly the k'-NN set of Q, where k' denotes the size of the range set.

Lemma 4 *At each iteration of the for loop of FADT_AVG, after the execution of Line 19, we have* $R_P^{\leq}(Q,\gamma) = N_S(Q,k')$.

Proof: From Lemma 3, we know that $D_\phi(Q,v) \ge \gamma$ for any object $v \in S \setminus R_P^{\leq}(Q,\gamma)$. Then, no such object v may have distance $D_\phi(Q,v)$ smaller than that of any object in $R_P^{\leq}(Q,\gamma)$. This implies that $R_P^{\leq}(Q,\gamma) = N_S(Q,k')$, where $k' = |R_P^{\leq}(Q,\gamma)|$ (Line 19). □

The next lemma reveals the relationship between the ϕ-aggregate distance to Q and the distance to q^*, for any object in the data domain. Recall that q^* is the center of both $\mathrm{MEB}(Q)$ and the two closed balls determined in our algorithm. The relationship will be used in the validation of the two closed balls, as well as the dimensional test itself.

Lemma 5 *For any object* $v \in \mathbb{U}$, $D_\phi(Q,v) - r^* \leq d(q^*,v) \leq D_\phi(Q,v) + r^*$.

Proof: For every $q_i \in Q$ ($1 \leq i \leq m$) and any object $v \in \mathbb{U}$, we have the following inequality (triangle inequality):

$$d(q_i,v) - d(q^*,q_i) \leq d(q^*,v) \leq d(q_i,v) + d(q^*,q_i).$$

Since r^* is the radius of $\mathrm{MEB}(Q)$, $r^* \geq d(q^*,q_i)$ must hold. Thus, we have $d(q_i,v) - r^* \leq d(q^*,v) \leq d(q_i,v) + r^*$. Together with the definition of ϕ-aggregate distance D_ϕ, we can easily derive $D_\phi(Q,v) - r^* \leq d(q^*,v) \leq D_\phi(Q,v) + r^*$. □

The last lemma implies that all items contained in the inner ball are members of the aggregate query result.

Lemma 6 *At each iteration of the for loop of FADT_AVG, after the execution of Line 25,* $r_{in} = \delta_P(q^*,k_{in}) = \delta_S(q^*,k_{in})$, *and* $|B_S^{\leq}(q^*,r_{in})| = k_{in} < k$.

Proof: We can derive the following inequality for any object $v \in S \setminus R_P^{\leq}(Q,\gamma)$:

$$\begin{array}{rcll} d(q^*,v) & \geq & D_\phi(Q,v) - r^* & \text{(from Lemma 5)} \\ & \geq & \gamma - r^* & \text{(from Lemma 3).} \end{array}$$

Then no object $v \in S \setminus R_P^{\leq}(Q,\gamma)$ may have distance satisfying $d(q^*,v) < \gamma - r^*$. Letting $\Gamma = R_P^{\leq}(Q,\gamma)$, we have $R_\Gamma^{<}(q^*,\gamma - r^*) \supseteq R_S^{<}(q^*,\gamma - r^*)$. Since $\Gamma \subseteq P \subseteq S$, we obtain

$$R_\Gamma^{<}(q^*,\gamma - r^*) = R_P^{<}(q^*,\gamma - r^*) = R_S^{<}(q^*,\gamma - r^*).$$

From Line 25 we have $r_{in} = \delta_P(q^*,k_{in})$, and from Line 23 we have $k_{in} = |R_P^{<}(q^*,\gamma - r^*)|$. Then, $r_{in} = \delta_P(q^*,k_{in}) < \gamma - r^*$ must hold. Also, any object $v \in S \setminus P$ has distance satisfying

$d(q^*,v) \geq \gamma - r^*$ (since $R_P^{\leq}(q^*,\gamma - r^*) = R_S^{\leq}(q^*,\gamma - r^*)$). Therefore, $r_{in} = \delta_P(q^*,k_{in}) = \delta_S(q^*,k_{in})$.

Now, we shall prove that $|B_S^{\leq}(q^*,r_{in})| = k_{in} < k$. Since $k_{in} = |R_P^{\leq}(q^*,\gamma - r^*)| = |R_S^{\leq}(q^*,\gamma - r^*)|$, then $\delta_S(q^*,k_{in}) < \gamma - r^* \leq \delta_S(q^*,k_{in}+1)$ must hold. From this we obtain $r_{in} = \delta_S(q^*,k_{in}) < \delta_S(q^*,k_{in}+1)$, which implies that $|B_S^{\leq}(q^*,r_{in})| = k_{in}$. With $R_P^{\leq}(q^*,\gamma - r^*) = R_\Gamma^{\leq}(q^*,\gamma - r^*)$, we can show that

$$k_{in} = |R_P^{\leq}(q^*,\gamma - r^*)| = |R_\Gamma^{\leq}(q^*,\gamma - r^*)| \leq |\Gamma| = k' \text{ (Line 19)}.$$

Since $k' < k$ (Line 22), $|B_S^{\leq}(q^*,r_{in})| = k_{in} < k$ must hold. $\square$

We now state and prove the main result of the analysis, which justifies the two pruning strategies of our algorithm: the use of the pruning distance bound, and the dimensional test.

Theorem 2 *If MED*$(q^*,k+1)$ *has no definite value, then FADT_AVG returns the correct query result regardless of the value of* t. *Otherwise, if* $t \geq$ *MED*$(q^*,k+1)$, *FADT_AVG returns the correct query result; if* $0 < t <$ *MED*$(q^*,k+1)$, *FADT_AVG returns a* $\sqrt[t]{\wp}$-*approximate query result with* $\wp = k_{out}+1$ *at termination.*

Proof: First, we note that the algorithm must terminate, as the iterative retrieval of objects would eventually result in the termination condition at Line 15 being reached. In this case, all the objects in S would have been retrieved from the underlying index. Obviously, the algorithm would return the correct result for this case (Line 16).

Otherwise, the algorithm would terminate with one of the following two conditions being reached (Line 20 or 28): $k' \geq k$ or

$$k_{in}\left(\frac{r_{out}}{r_{in}}\right)^t < k_{out}+1. \tag{4.2}$$

For the case when $k' \geq k$, $R^{\leq}_{P}(Q,\gamma) = N_S(Q,k')$ must be true (from Lemma 4). Then, we can easily see that P contains $N_S(Q,k)$. Therefore, the algorithm returns the correct result for this case also.

For the last case when Inequality (4.2) holds, $k_{in} > 0$ and $r_{in} > 0$ must be true (Lines 24 and 28). From Lemma 6, we know that $r_{in} = \delta_P(q^*, k_{in}) = \delta_S(q^*, k_{in})$, and $|B^{\leq}_{S}(q^*, r_{in})| = k_{in} < k$. Then, $r_{in} < \delta_S(q^*, k) \leq \delta_S(q^*, k+1)$ must hold. Therefore, the closed ball $B^{\leq}_{S}(q^*, r_{in})$ belongs to the inner ball set $\mathfrak{B}_{in}(q^*, k+1)$, and thus $\mathrm{MED}(q^*, k+1)$ has a definite value for this case.

Now, let us consider the closed ball $B^{\leq}_{S}(q^*, \tilde{r}_{out})$ with $\tilde{r}_{out} = \delta_S(q^*, k_{out}+1)$. From Lines 10-17 we have $k \leq |P| < |S|$, and from Line 27 we have $k_{out} = |R^{\leq}_{P}(q^*, r_{out})|$. We observe that

$$k_{out} = |R^{\leq}_{P}(q^*, r_{out})| \leq |P| < |S|. \tag{4.3}$$

Also, for any object $v \in N_P(Q,k)$, we have

$$\begin{aligned} d(q^*, v) &\leq D_\phi(Q, v) + r^* && \text{(from Lemma 5)} \\ &\leq \delta_P(Q,k) + r^* \\ &= r_{out} && \text{(from Line 26).} \end{aligned}$$

Thus, $k_{out} = |R^{\leq}_{P}(q^*, r_{out})| \geq k$. Together with Inequality (4.3), we have

$$k \leq k_{out} < |S|. \tag{4.4}$$

Therefore, the closed ball $B^{\leq}_{S}(q^*, \tilde{r}_{out})$, where $\tilde{r}_{out} = \delta_S(q^*, k_{out}+1)$, belongs to the outer ball set $\mathfrak{B}_{out}(q^*, k+1)$.

Assume that $t \geq \mathrm{MED}(q^*, k+1)$. Letting $\tilde{k}_{out} = |B^{\leq}_{S}(q^*, \tilde{r}_{out})|$, we can derive

$$t \geq \frac{\log \tilde{k}_{out} - \log k_{in}}{\log \tilde{r}_{out} - \log r_{in}}$$

and

$$k_{in}\left(\frac{\tilde{r}_{out}}{r_{in}}\right)^t \geq \tilde{k}_{out} \geq k_{out}+1.$$

Applying Inequality (4.2) and solving for r_{out}, we obtain

$$r_{out} < \tilde{r}_{out} = \delta_S(q^*, k_{out}+1). \tag{4.5}$$

We also know that $|R_P^{\leq}(q^*, r_{out})| = k_{out}$ (Line 27). Together, these two conditions imply that any object $v \in S$ with distance satisfying $d(q^*, v) < \delta_S(q^*, k_{out}+1) = \tilde{r}_{out}$ must be included in P. In other words, for any object $v \in S \setminus P$, we have $d(q^*, v) \geq \tilde{r}_{out}$. Therefore, for any object $v \in S \setminus P$, we can derive

$$\begin{aligned} D_\phi(Q, v) &\geq d(q^*, v) - r^* && \text{(from Lemma 5)} \\ &\geq \tilde{r}_{out} - r^* \\ &> r_{out} - r^* && \text{(from Inequality (4.5))} \\ &= \delta_P(Q, k) && \text{(from Line 26).} \end{aligned}$$

Consequently, no object $v \in S \setminus P$ may have distance satisfying $D_\phi(Q, v) \leq \delta_P(Q, k)$, implying that $\delta_P(Q, k) = \delta_S(Q, k)$, and that P contains $N_S(Q, k)$. Thus, the algorithm returns the correct result for the last case, where $t \geq \mathrm{MED}(q^*, k+1)$.

Now, assume $0 < t < \mathrm{MED}(q^*, k+1)$. From Line 24, we have $k_{in} \geq 1$. Together with Inequality (4.2), we get $(r_{out}/r_{in})^t < k_{out}+1$. Letting $\wp = k_{out}+1$, we obtain $r_{out} < \sqrt[t]{\wp} \cdot r_{in}$. Since $r_{in} = \delta_P(q^*, k_{in})$ (Line 25) and $k_{in} = |R_P^{\leq}(q^*, \gamma - r^*)|$ (Line 23), $r_{in} < \gamma - r^*$ must hold. Then, $r_{out} < \sqrt[t]{\wp} \cdot r_{in} < \sqrt[t]{\wp}(\gamma - r^*)$ must hold. Applying $r_{out} = \delta_P(Q, k) + r^*$ (Line 26), we get

$$\delta_P(Q, k) + r^* < \sqrt[t]{\wp}(\gamma - r^*). \tag{4.6}$$

We know that r^* is the radius of $\mathrm{MEB}(Q)$, where $|Q| > 1$. Obviously, $r^* > 0$. Consequently, $\delta_P(Q, k) < \sqrt[t]{\wp} \cdot \gamma$ must hold. From Lemma 3 and Lemma 4, we observe that $\gamma \leq D_\phi(Q, v)$ for any object $v \in S \setminus N_S(Q, k')$. Together with $k' < k$ (Line 22), we obtain $\gamma \leq \delta_S(Q, k)$.

Therefore, $\delta_P(Q,k) < \sqrt[t]{\wp} \cdot \gamma \leq \sqrt[t]{\wp} \cdot \delta_S(Q,k)$ must be true. We thus conclude that the algorithm returns a $\sqrt[t]{\wp}$-approximate result, where $\wp = k_{\text{out}} + 1$, for the last case if $0 < t < \text{MED}(q^*, k+1)$. $\square$

Note that since $k \leq k_{\text{out}} < |S|$ (Inequality (4.4)), the approximation ratio $\sqrt[t]{\wp}$ with $\wp = k_{\text{out}} + 1$ is within the range $[\sqrt[t]{k+1}, \sqrt[t]{|S|}]$. That is, the approximation ratio is $\sqrt[t]{|S|}$ in the worst case.

In practice, the value of $\text{MED}(q^*, k+1)$ is typically not known beforehand, and thus cannot be used to indicate a choice of termination parameter t. However, the approximation ratio stated in Theorem 2 can definitely serve as a guide for choosing t. Specifically, for a desired approximation ratio $\eta > 1$, t should be chosen to be the smallest value such that $\sqrt[t]{|S|} \leq \eta$, which is $\log_\eta |S|$. With this choice of t, Algorithm FADT_AVG is guaranteed to produce results with approximation ratios no larger than the desired η.

4.3.3 Variants of FADT_AVG

In this subsection, several variants of FADT_AVG are discussed.

Proportional Reduction of the MEB Radius. For Algorithm FADT_AVG, in order to obtain a theoretical guarantee of correctness, a term equal to the radius r^* of $\text{MEB}(Q)$ is used in the determination of two closed balls when performing the dimensional test in Lines 23-28. The use of this term is motivated purely by the analysis, in that it allows the triangle inequality to be invoked in the proof of Lemma 5.

In practice, however, we may sacrifice the guarantee of correctness in the hope of gaining improvements in performance. One way to achieve this is by eliminating the MEB radius term r^* in Lines 23 and 26, or by using a proportion $\rho \cdot r^*$ of the radius, for some choice $0 \leq \rho \leq 1$. Decreasing the proportion of r^* will increase the likelihood of success of the dimensional test, and thus possibly lead to an earlier termination of the algorithm. The variant of FADT_AVG due to the use of the proportional factor ρ will be referred to as FADT_AVG1.

Clearly, the theoretical guarantee of correctness is lost whenever the proportion ρ is chosen to be less than 1. Nevertheless, we can still show that if $t > 0$, then FADT_AVG1 returns a $\sqrt[t]{\wp}$-approximate query result, where $\wp = k_{\text{out}} + 1$ at termination, regardless of the value of ρ.

Corollary 1 *If* $t > 0$, *then FADT_AVG1 returns a* $\sqrt[t]{\wp}$*-approximate query result, where* $\wp = k_{\text{out}} + 1$ *at termination, regardless of the value of* ρ.

Proof: The argument is similar to that of the proof of Theorem 2, with Inequality (4.6) replaced by

$$\delta_P(Q,k) + \rho r^* < \sqrt[t]{\wp}(\gamma - \rho r^*).$$

Since $\rho r^* \geq 0$, the inequality $\delta_P(Q,k) < \sqrt[t]{\wp} \cdot \gamma$ still holds. □

Restricted-Size Buffer for Candidate Objects. During the execution of Algorithm FADT_AVG, in the worst case, we may need to store a very large number of candidate objects, since the outer ball determined at each iteration (Lines 26 and 27) may potentially contain a number of points proportional to the size of the entire database. This may be a serious issue for those applications in which the system memory is scarce. We therefore propose a variant of FADT_AVG, called FADT_AVG2, in which the capacity of the buffer of potential candidates P is restricted to be a small multiple of k — in particular, zk for some small constant choice of $z \geq 1$. Consequently, from among all objects visited, only zk nearest neighbors of q^* (the center of the outer ball) are maintained as candidates when determining the membership of the outer ball. With this restriction, the chance of passing the dimensional test may decrease somewhat, and as a result, the termination of the algorithm may be delayed. Note that in the FADT_AVG2 variant, we must also maintain as a tentative query result the k-NN set of Q from among all the objects visited by the search.

Again, by restricting the size of the candidate buffer P, we lose the theoretical guarantee of the correctness of the algorithm. However, we can nevertheless show that if $t > 0$, then FADT_AVG2 returns an approximate query result with the bound dependent on the buffer size and on t.

Corollary 2 *If* $t > 0$*, FADT_AVG2 returns a* $\sqrt[t]{zk+1}$*-approximate query result.*

Proof: The argument is similar to that of the proof of Theorem 2, but with the value of k_{out} bounded from above by zk. □

Estimation of Lower-Bounding Ratios. In practice, it may be difficult for the user to determine values for the lower-bounding ratios λ_i $(1 \leq i \leq m)$ with which FADT_AVG can perform well. If the ratios are too small, a large number of candidates for the query result can be expected; if they are too large, the accuracy of the query result may suffer. In order to gain more control over the tradeoff between execution cost and query result accuracy, we modify Algorithm FADT_AVG so as to dynamically estimate an appropriate value for each of the lower bounding ratios λ_i $(1 \leq i \leq m)$. The extension, called FADT_AVG3, is described in Figure 4.6. Observed distance values are used to estimate the lower-bounding ratios; more precisely, after each neighborhood v_i of the geometric median M_i and distance $\beta_i = d(M_i, v_i)$ are retrieved from the underlying index, the sum of the ϕ-aggregate distance $D_\phi(Q, v_i)$ and some constant α_i is computed first, followed by the ratio of the sum over β_i. The smallest such ratio encountered is stored in λ_{e_i} as the current estimate of λ_i.

The estimation of λ_i can be expected to perform well if no objects possess both small ϕ-aggregate distance to Q and large distance to M_i. Such objects, if they exist, could lead to an undesirable underestimation of the value of λ_i; as a result, the termination of the algorithm may be significantly delayed. To avoid the underestimation, any such objects are treated as outliers, and excluded from the estimation. Before allowing a given object to contribute to the estimation of λ_i, we first check whether the object is in the most-recently computed tentative k-NN result set for group query Q. If the object is present, we can conclude that: (1) the aggregate distance from the object to Q is small; and (2) the object has already been retrieved from the neighborhood of some geometric median other than

```
Algorithm FADT_AVG3 (database $S$, group query $Q$, target proportion $\phi$, aggregation function avg, neighborhood size $k$, termination parameter $t$)
 6: Initialize $\gamma_i \leftarrow -\infty$ and $\lambda_{e_i} \leftarrow +\infty$, for $i = 1, \ldots, m$.
10: $(v_i, \beta_i) \leftarrow I.getnext(M_i)$.    // $\beta_i = d(M_i, v_i)$
    if $v_i \notin N_P(Q, k)$ then
      $\lambda_{e_i} \leftarrow \min\{\lambda_{e_i}, (D_\phi(Q, v_i) + \alpha_i)/\beta_i\}$.
    end if
11: $\gamma_i \leftarrow \lambda_{e_i} \cdot \beta_i - \alpha_i$.    // lower distance bound
18: $\gamma \leftarrow \min\{\gamma_1, \ldots, \gamma_m\}$.
```

Figure 4.6 The description of Algorithm FADT_AVG3. Only those steps for which changes have been made are shown; all other steps are identical to those of Algorithm FADT_AVG (see Figure 4.4).

M_i, indicating that the distance from the object to M_i may be unusually large. When both these conditions hold, the object is likely to be an outlier or noise element. We therefore exclude any such objects from the estimation. This exclusion is performed at Line 10 of Algorithm FADT_AVG3 (see Figure 4.6).

The estimate of λ_i thus obtained is in turn used to determine the lower distance bound γ_i. In order to determine a query result with reasonable accuracy, during each iteration of the algorithm, we conservatively select the smallest (most-restrictive) of the m lower bounds γ_i $(1 \leq i \leq m)$.

4.4 The *Max* Variant of FANN

We now present an algorithm for the *max* variant of the FANN problem, as well as several extensions along similar lines as was presented in the previous section for the *avg*/*sum* variant.

4.4.1 Algorithm FADT_MAX

The algorithm for the *max* variant, which we refer to as FADT_MAX, is very similar to FADT_AVG. A list of steps that differ from those of FADT_AVG is shown in Figure 4.7. The change of aggregation function from *avg* by *max* necessitates two modifications of the

```
Algorithm FADT_MAX (database $S$, group query $Q$, target proportion $\phi$, aggregation function max, neighborhood size $k$, lower-bounding ratios $\lambda_1,\ldots,\lambda_m$, termination parameter $t$)
 3: For each $q_i \in Q$, find the center $C_i$ of minimum enclosing ball of $N_Q(q_i,\phi m)$.
 4: Let $\alpha_i$ be the $\phi m$-th largest distance from $C_i$ to all the query points in $Q$,
    for $i = 1,\ldots,m$.
10: $(v_i,\beta_i) \leftarrow I.getnext(C_i)$.    // $\beta_i = d(C_i,v_i)$
```

Figure 4.7 The description of Algorithm FADT_MAX, showing only those steps that differ from the corresponding steps of Algorithm FADT_AVG; all other steps are identical (see Figure 4.4).

algorithm: the set of objects from which to expand the search for tentative query results, and the values of α_i ($1 \le i \le m$) used in the computation of the lower bounds γ_i. Instead of expanding the search from geometric medians of subsets of Q (as in FADT_AVG), Algorithm FADT_MAX initiates its searches from the center C_i of the minimum enclosing ball of query set neighborhood $N_Q(q_i,\phi m)$, for each $q_i \in Q$. The center C_i is the point minimizing the maximum distance to any point in the query set neighborhood. FADT_MAX sets the value α_i to be the ϕm-th largest distance to C_i from among the query points of Q. These changes in turn require a slight modification of the relationship between the pruning distance and the aggregation.

Lower-Bounding Relationship. For every center C_i ($1 \le i \le m$), every query object $q_j \in Q$ ($1 \le j \le m$), and any data object $v \in \mathbb{U}$, we have the following triangle inequality:

$$d(C_i,v) \le d(q_j,v) + d(C_i,q_j).$$

Together with the definition of ϕ-aggregate distance D_ϕ, a *max* aggregation of these bounds on $d(C_i,v)$ over every $q \in N_Q(v,\phi m)$ yields:

$$d(C_i,v) \le D_\phi(Q,v) + d(C_i,q).$$

As per the definition in the description of FADT_MAX, let α_i be the ϕm-th largest distance from C_i to all the query points in Q. Clearly, it cannot be the case that all ϕm query points in the set $N_Q(v, \phi m)$ have distance to C_i greater than α_i. In other words, there must be at least one query point $q \in N_Q(v, \phi m)$ satisfying $d(C_i, q) \leq \alpha_i$. We can therefore derive the following lower-bounding relationship:

$$d(C_i, v) \leq D_\phi(Q, v) + \alpha_i.$$

Again, we may view $d(C_i, v)$ as an approximation of $D_\phi(Q, v) + \alpha_i$. Since the performance of FADT_MAX depends on the tightness of the approximation, we again introduce a lower-bounding ratio. In this case, we set $\lambda_i \geq 1$ such that

$$\lambda_i d(C_i, v) \leq D_\phi(Q, v) + \alpha_i.$$

We shall now give a formal theoretical analysis of FADT_MAX along the same lines as that of FADT_AVG.

Theorem 3 *If MED*$(q^*, k+1)$ *has no definite value, then FADT_MAX returns the correct query result regardless of the value of* t. *Otherwise, if* $t \geq$ *MED*$(q^*, k+1)$, *FADT_MAX returns the correct query result; if* $0 < t <$ *MED*$(q^*, k+1)$, *FADT_MAX returns a* $\sqrt[t]{\wp}$-*approximate query result with* $\wp = k_{\text{out}} + 1$ *at termination.*

Proof: Omitted; the proof is similar to that of Theorem 2. □

4.4.2 Extensions of FADT_MAX

As with Algorithm FADT_AVG, we consider several extensions of Algorithm FADT_MAX.

Using a Proportion of the Radius Value r*. We extend Algorithm FADT_MAX by using only a fixed proportion of the value r^* in Lines 23 and 26. For this extension, which we will refer to as FADT_MAX1, we use $0 \leq \rho \leq 1$ to denote the proportion to be used.

Corollary 3 *If* $t > 0$, *then FADT_MAX1 returns a* $\sqrt[t]{\wp}$-*approximate query result, where* $\wp = k_{\text{out}} + 1$ *at termination, regardless of the value of* ρ.

Algorithm **FADT_MAX3** (*database* S, *group query* Q, *target proportion* ϕ, *aggregation function* max, *neighborhood size* k, *termination parameter* t)

```
6: Initialize $\gamma_i \leftarrow -\infty$ and $\lambda_{e_i} \leftarrow +\infty$, for $i = 1, \ldots, m$.
10: $(v_i, \beta_i) \leftarrow I.getnext(C_i)$.    // $\beta_i = d(C_i, v_i)$
    if $v_i \notin N_P(Q, k)$ then
        $\lambda_{e_i} \leftarrow \min\{\lambda_{e_i}, (D_\phi(Q, v_i) + \alpha_i)/\beta_i\}$.
    end if
11: $\gamma_i \leftarrow \lambda_{e_i} \cdot \beta_i - \alpha_i$.    // lower distance bound
18: $\gamma \leftarrow \min\{\gamma_1, \ldots, \gamma_m\}$.
```

Figure 4.8 The description of Algorithm FADT_MAX3, showing only those steps that differ from the corresponding steps of Algorithm FADT_MAX; all other steps are identical (see Figure 4.7).

Proof: The approximation bound can be verified to hold from the arguments contained in the proof of Theorems 2 and 3. □

Limiting the Capacity of the Object Buffer. The objects visited in the course of the search must be maintained in a buffer, denoted as P in the description of the algorithms. We extend Algorithm FADT_MAX by limiting the maximum size of P to be a constant multiple zk of k, where $z \geq 1$. Consequently, from among all objects visited, only the zk-nearest neighbors of the center q^* are used in the determination of the outer ball. We call this extension FADT_MAX2.

Corollary 4 *If* $t > 0$, *FADT_MAX2 returns a* $\sqrt[t]{zk+1}$*-approximate query result.*

Dynamic Estimation of Lower-Bounding Ratios. We extend Algorithm FADT_MAX so as to dynamically estimate values for each of the lower bounding ratios λ_i ($1 \leq i \leq m$). The extension, which we call FADT_MAX3, is described in Figure 4.8.

4.5 Experimental Results

In this section, we present the results of our experimentation. We compared our algorithms with the state-of-the-art approaches List, ASUM and AMAX. All algorithms were implemented in C++. The experiments were run on a desktop computer with a quad-core 2.4GHz CPU and 8GB of memory.

4.5.1 Experimental Framework

Before presenting our experimental results, we first describe the data sets, query types, and methodology to be followed.

Data Sets. Four publicly-available data sets were considered for the experimentation, so as to compare across a variety of set sizes, representational dimensions and data domains.

- The MNIST data set [LeCun et al., 1998] consists of $70,000$ images of handwritten digits from 500 different writers, with each image represented by 784 gray-scale texture values.
- The Amsterdam Library of Object Images (ALOI) [Geusebroek et al., 2005] consists of $110,250$ images of 1000 small objects taken under different conditions, such as differing viewpoints and illumination directions. The images are represented by 641-dimensional feature vectors based on color and texture histograms (for a detailed description of how the vectors were produced, see [Boujemaa et al., 2001]).
- The Cortina data set [1] consists of $1,088,864$ images gathered from the World Wide Web. Each image is represented by a 74-dimensional feature vector based on homogeneous texture, dominant color and edge histograms.
- The Reuters Corpus Vol. 2 news article data set (RCV2) [2] consists of $554,651$ sparse document vectors spanning $320,647$ keyword dimensions, with TF-IDF weighting followed by vector normalization.

FANN Queries. To generate a FANN query, several factors should be taken into account, including the position of the center of $\mathrm{MEB}(Q)$ (minimum enclosing ball of group query Q), the radius of $\mathrm{MEB}(Q)$, the size $m = |Q|$, the target neighborhood size k, and the target proportion ϕ. For our experiments, we generated a group query Q in the following way. We first determined a ball with a randomly-chosen center covering a certain number c of points in the data space. Then, within this ball, we randomly selected m data points as the group query. Parameter c can thus be regarded as governing the dispersion of the query object sets

[1] http://www.scl.ece.ucsb.edu/datasets/index.htm (accessed on November 9, 2014).
[2] http://trec.nist.gov/data/reuters/reuters.html (accessed on November 9, 2014).

generated for the experimentation. Unless stated otherwise, the default parameters $m = 16$, $k = 50$, $\phi = 0.5$ and $c = 10,000$ were used in generating FANN queries.

Methodology. For each test, 100 random FANN queries were generated. Three quantities were measured for the evaluation: query result accuracy, number of candidates visited, and execution time. The results were reported as averages over the 100 queries performed. The number of candidates was calculated as the number of data objects (including duplicates) retrieved from the underlying index in the course of computing the final query result. For one group query Q, the accuracy of its k-NN query result is defined as

$$\frac{|\{v \in Y \mid D_\phi(Q,v) \le \delta_S(Q,k)\}|}{k},$$

where Y denotes the k-NN query result of Q ($|Y| = k$). For ease of comparison among the competing methods, we ignored the costs associated with producing candidate elements from the underlying index structure I, and instead precomputed all required neighborhoods. However, in order to gauge the practicality of the methods tested, in the last set of experiments, we report the total query cost, including the costs associated with the production of neighborhood lists. For these experiments, we used SASH [Houle and Sakuma, 2005] as the underlying index I, for its ability to answer approximate k-NN queries in very high dimensional settings with good accuracy and efficiency. SASH does not directly support the incremental queries needed by our algorithms and by List. However, using an implementation of SASH with an internal buffer storing previously-computed distances, neighborhoods can be efficiently extended by performing $(2^i k)$-NN queries for successively larger choices of i, until a sufficient number of candidates are produced. The Euclidean distance was used for all experiments. For dataset RCV2, the normalization of data vectors ensures that the query results produced using the Euclidean distance are consistent with those that would have been obtained using the cosine similarity measure.

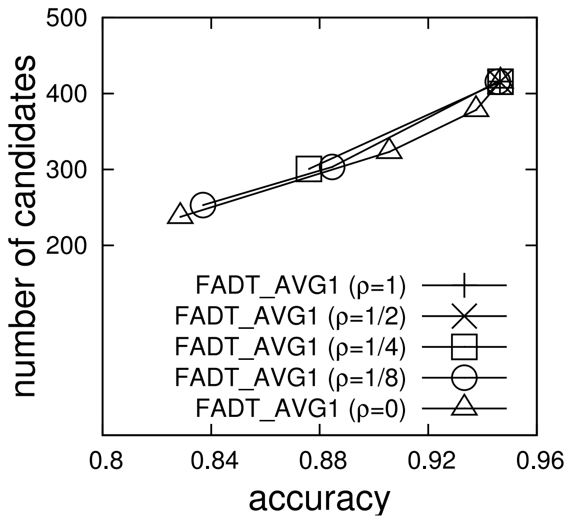
4.5.2 Comparison of Variants of FADT_AVG and FADT_MAX

We conducted several sets of experiments to compare the relative performance of our proposed methods, using all four data sets described above. Since we observed the same trends in the results for all the data sets tested, we will show only the results for the MNIST data set. The execution times for all experiments were proportional to the number of candidates processed by the methods; accordingly, we will report the execution costs only in terms of these numbers of candidates.

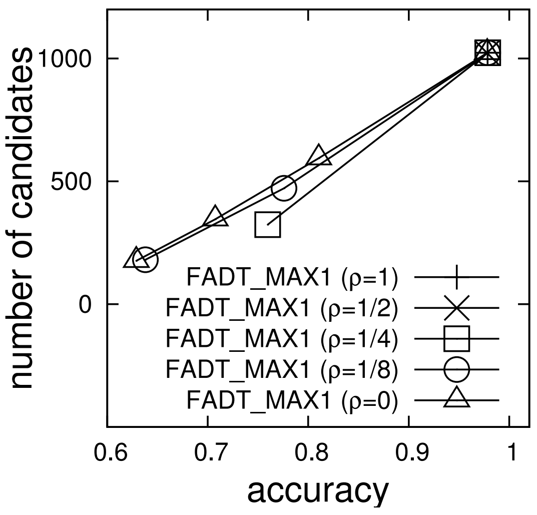
For the experiments, we selected a common value $\lambda = \lambda_i$ for all lower-bounding ratios ($1 \leq i \leq m$). In order to cover as wide a range of result accuracies as possible, the values of termination parameter t were chosen to be of the form 2^i, for all choices of integer i in the range $[-6, 6]$. Note that for readability, the results for some of the more extreme values of t may be omitted from some plots.

Using a Proportion of the Radius Value r^*. First, we tested the effect of using a proportion $0 \leq \rho \leq 1$ of the value of radius r^*, which was proposed for the algorithm variants FADT_AVG1 and FADT_MAX1. Three values of $c = 100$, 1000 and $10,000$ were chosen in the generation of queries, in order to test the effect over a range of values of r^*. Note that the value of r^* tends to grow as the value of c increases. Again, since we observed similar trends for all three choices of c, we report as an example only the results for $c = 100$, in Figure 4.9. As expected, for both FADT_AVG1 and FADT_MAX1, the performance tradeoff between execution cost and accuracy improves as the value of ρ decreases. This trend justifies the heuristic choice of $\rho = 0$ in practice. In particular, for all remaining experiments, we use $\rho = 0$; that is, the value of r^* is treated as 0.

Limiting the Capacity of the Object Buffer. We next tested the effect of maintaining an object buffer with limited capacity zk for the determination of the outer ball, which was proposed as variants FADT_AVG2 and FADT_MAX2. On top of these two variants, we heuristically set the value of r^* to 0 as justified previously. We call the new variants



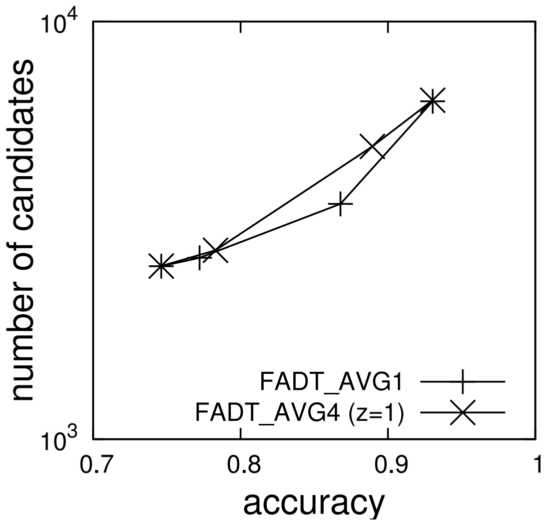
(a) *sum* FANN ($\lambda = 2.4$)



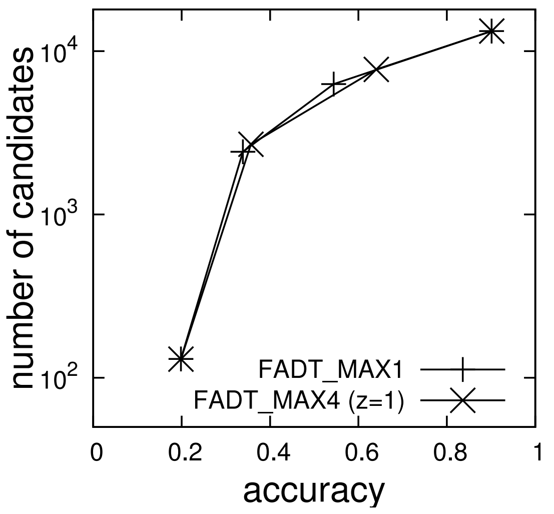
(b) *max* FANN ($\lambda = 2.2$)

Figure 4.9 The effect of using a proportion of the value of radius $\mathbf{r}^*$, with dataset MNIST.

FADT_AVG4 and FADT_MAX4. As shown in Figure 4.10, we observed in our experiments that even with the choice $z = 1$ (yielding the minimal buffer size k), the performances of FADT_AVG4 and FADT_MAX4 are nearly identical to that of FADT_AVG1 and FADT_MAX1, respectively. This observation indicates that the real buffer size needed for the determination of the outer ball is typically quite small. Nevertheless, in order to



(a) *sum* FANN ($\lambda = 2.5$)

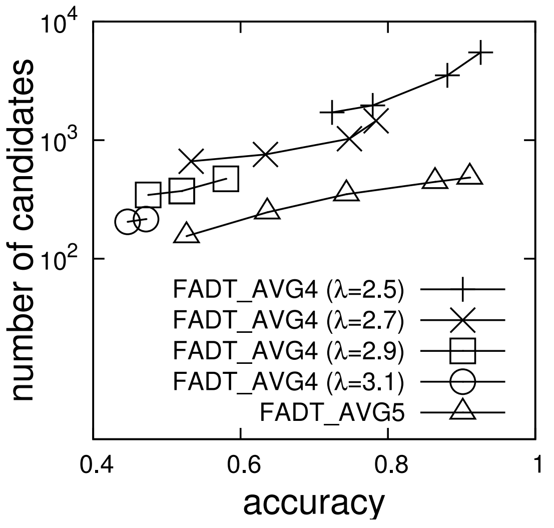


(b) *max* FANN ($\lambda = 2.3$)

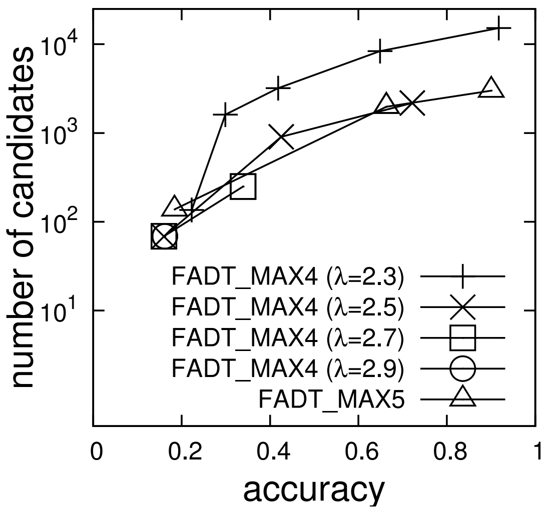
Figure 4.10 The effect of limiting the capacity of the object buffer, with dataset MNIST.

have a fair comparison with other methods, for all remaining experiments we maintain a buffer of at most k visited objects for the determination of the outer ball.

Estimation of Lower-Bounding Ratios. We then tested the effect of dynamic estimation of lower-bounding ratios, which was proposed as the algorithm variants FADT_AVG3 and FADT_MAX3. As justified in previous experiments, on top of these two variants, we set the value of r^* to 0 and maintain an object buffer with size k for the determination of



(a) *sum* FANN



(b) *max* FANN

Figure 4.11 The effect of dynamic estimation of lower-bounding ratios, with dataset MNIST.

the outer ball. The new variants are called FADT_AVG5 and FADT_MAX5. The results, shown in Figure 4.11, indicate that FADT_AVG5 and FADT_MAX5 generally improve upon FADT_AVG4 and FADT_MAX4, respectively, over a range of fixed choices of λ. This can be explained by the fact that for different queries, the 'best' choices of λ that lead to a given accuracy may vary considerably. In other words, we may not be able to find a fixed value of λ that is good for all queries. Therefore, in all remaining experiments with competing methods, we will present results for FADT_AVG5 and FADT_MAX5 as the canonical representatives of our proposed search strategies.

4.5.3 Comparison with Other Methods

In this subsection, we compare the best variants of our FANN methods, FADT_AVG5 and FADT_MAX5, with the exact method List, as well as the two approximation methods ASUM and AMAX. No sampling was utilized in the implementation of ASUM and AMAX, in order to ensure that they produced query results with the highest accuracies possible for these methods.

Using Precomputed Neighborhoods. On all the considered datasets, we conducted 4 sets of experiments for the comparison, varying one of the 4 parameters for generating FANN queries respectively, while fixing the rest at their default values ($m = 16$, $k = 50$, $\phi = 0.5$ and $c = 10,000$). Specifically, we varied m from 4 to 128, k from 20 to 100, ϕ from 0.1 to 1, and c from 100 to 30,000. Since similar conclusions can be drawn from all the results, we report only the results for the MNIST dataset. In addition, for all the datasets, we report the results obtained when all parameters are set to the default values stated above.

Variation in the Query Set Size m. The results of varying m are shown in Figure 4.12. We observe that in all cases, for both the *sum* (or *avg*) FANN problem and the *max* FANN problem, our algorithms FADT_AVG5 and FADT_MAX5 are able to produce query results with reasonable accuracy, while at the same time being very efficient. On average, our algorithms outperform List by approximately one order of magnitude, in terms of both the number of candidates and the execution time. This can be explained by their minimization of aggregate distances through the use of geometric medians or the centers of minimum enclosing balls. Although ASUM and AMAX are also very efficient, for some cases they failed to produce query results with reasonable accuracy. Their failure is likely due to their strategy of always limiting the search to at most km candidates. Such overly-aggressive

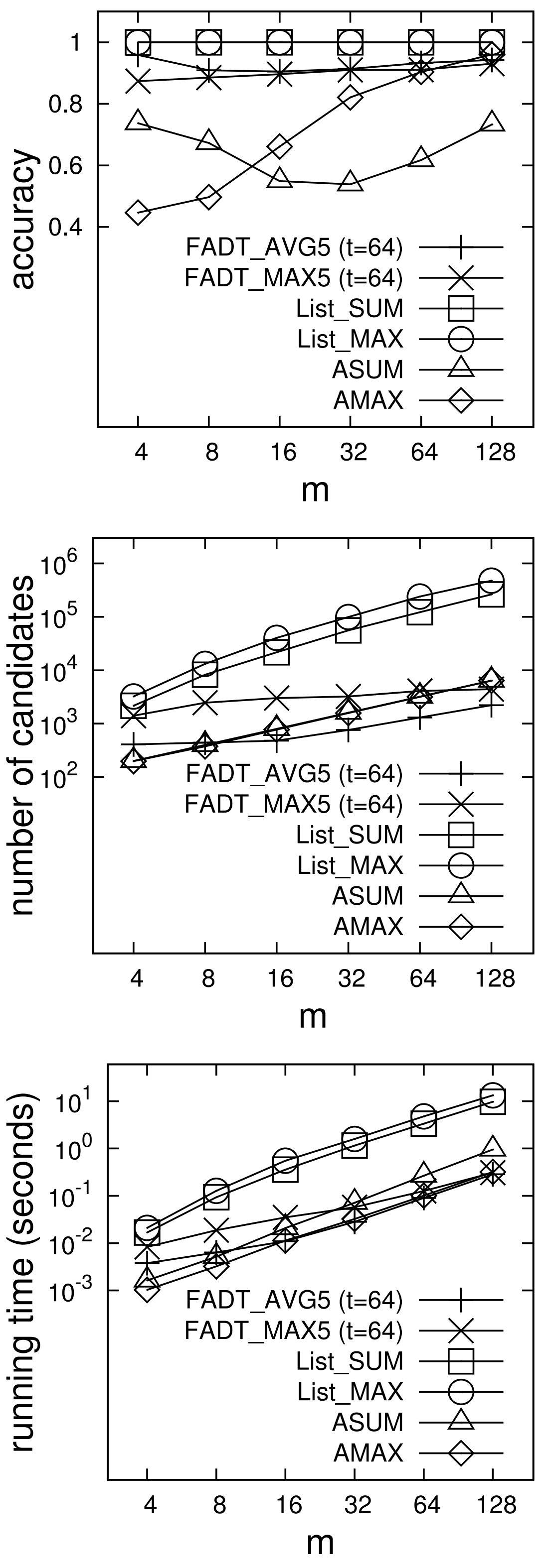


Figure 4.12 The results of varying m for generating FANN queries on dataset MNIST.

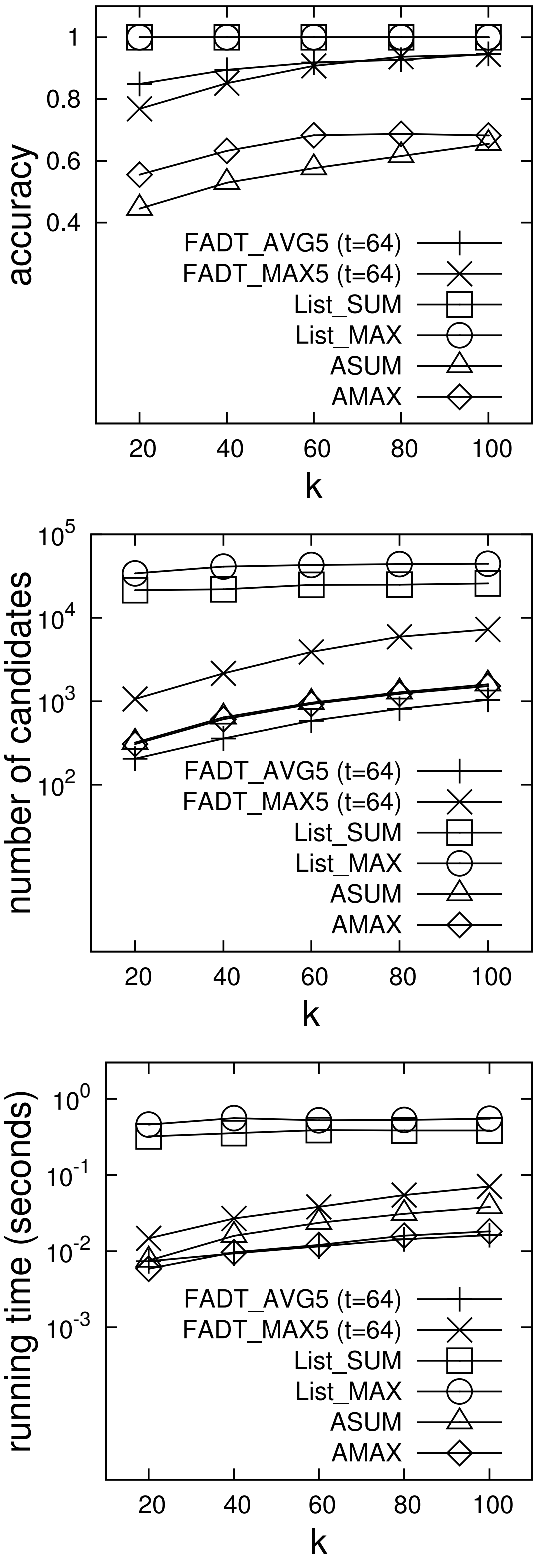


Figure 4.13 The results of varying k for generating FANN queries on dataset MNIST.

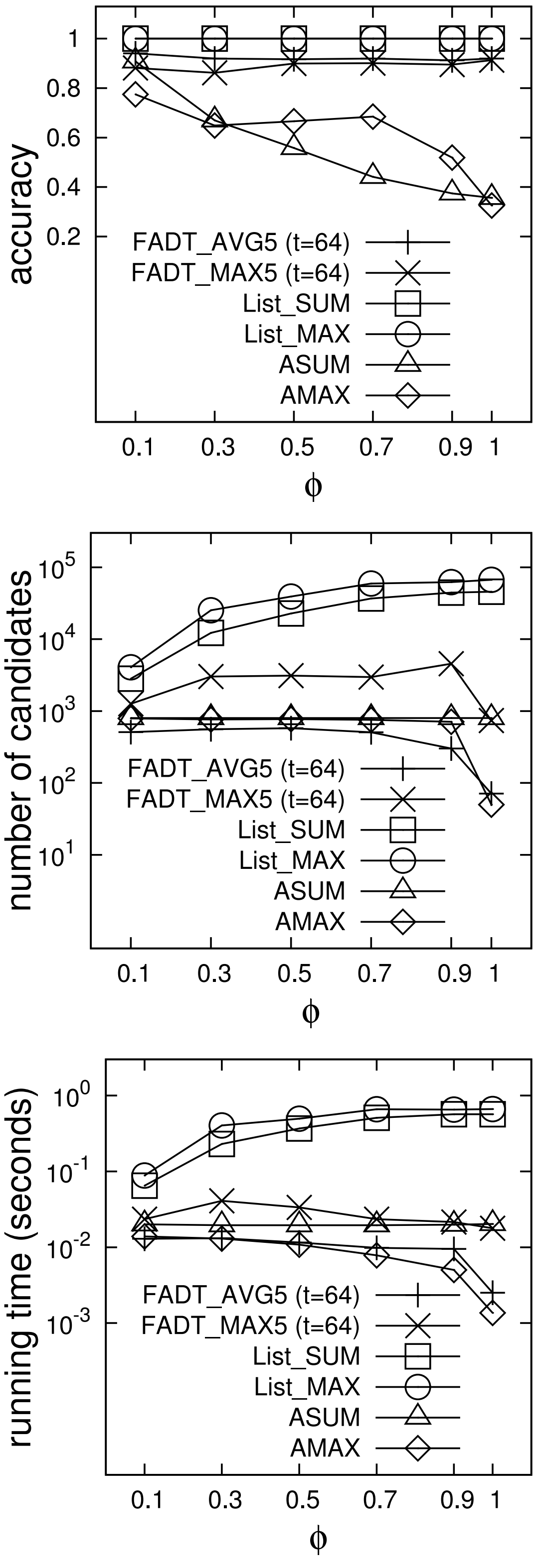


Figure 4.14 The results of varying ϕ for generating FANN queries on dataset MNIST.

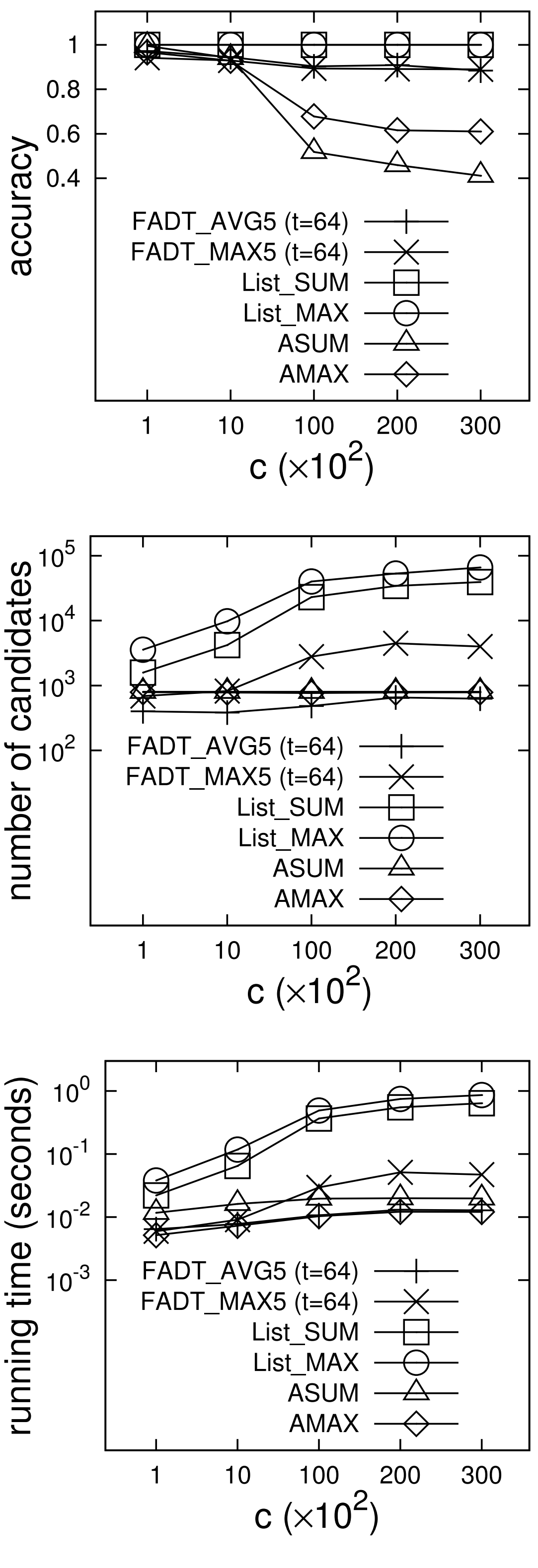


Figure 4.15 The results of varying c for generating FANN queries on dataset MNIST.

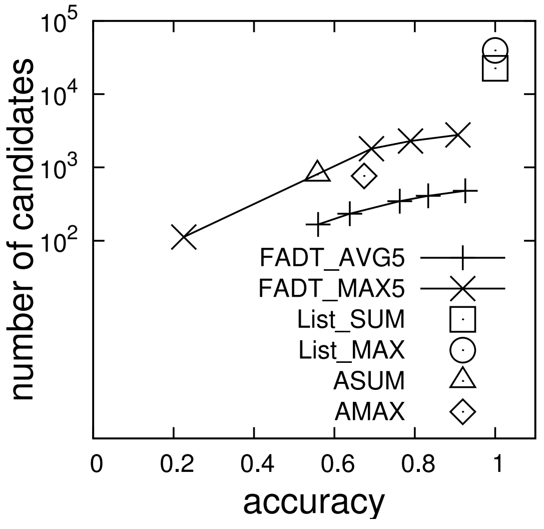
search strategies may not be able to produce the query result with reasonable accuracy, as justified here and in the following experiments.

Variation in the Neighborhood Size k. Figure 4.13 shows the results of varying k. We again observe that for all choices of k, our algorithms FADT_AVG5 and FADT_MAX5 maintain their superiority over List by roughly 1 to 2 orders of magnitude, in terms of both the number of candidates and the execution time. We also note that ASUM and AMAX may not be able to produce query results with reasonably good accuracy.

Variation in the Target Proportion ϕ. The results of varying ϕ are shown in Figure 4.14. Again, FADT_AVG5 and FADT_MAX5 outperform List in terms of both number of candidates and running time for all choices of ϕ, and again, ASUM and AMAX cannot guarantee query results with reasonable accuracy. The superiority of our algorithms over List becomes more and more evident as ϕ increases. In particular, when $\phi = 1$, our algorithms outperform List by approximately 2 to 3 orders of magnitude, in terms of both the number of candidates and the execution time. This is because that our algorithms benefit more from utilizing geometric medians or centers of MEB as ϕ grows.

Variation in the Query Set Dispersion Parameter c. Figure 4.15 shows the results of varying c. Similarly, algorithms FADT_AVG5 and FADT_MAX5 show their superiority over List in terms of both number of candidates and running time, and ASUM and AMAX may not be able to produce query results with good accuracy. The superiority of our algorithms over List becomes more and more evident as c increases. Especially, when $c = 30,000$, our algorithms outperform List by approximately 1.5 orders of magnitude in terms of both number of candidates and running time. The behaviors of our algorithms are relatively stable with respect to c, again due to the benefits of utilizing geometric medians or centers of MEB. In contrast, the efficiency of List drops fast as c increases.

Default Parameters. Here, we show the results for all datasets using the default parameters in generating and processing FANN queries. For our algorithms, parameter t was chosen to



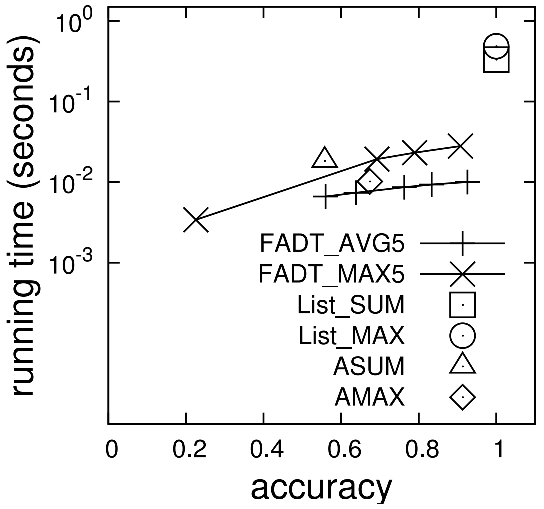


Figure 4.16 The results of using default parameters on dataset MNIST.

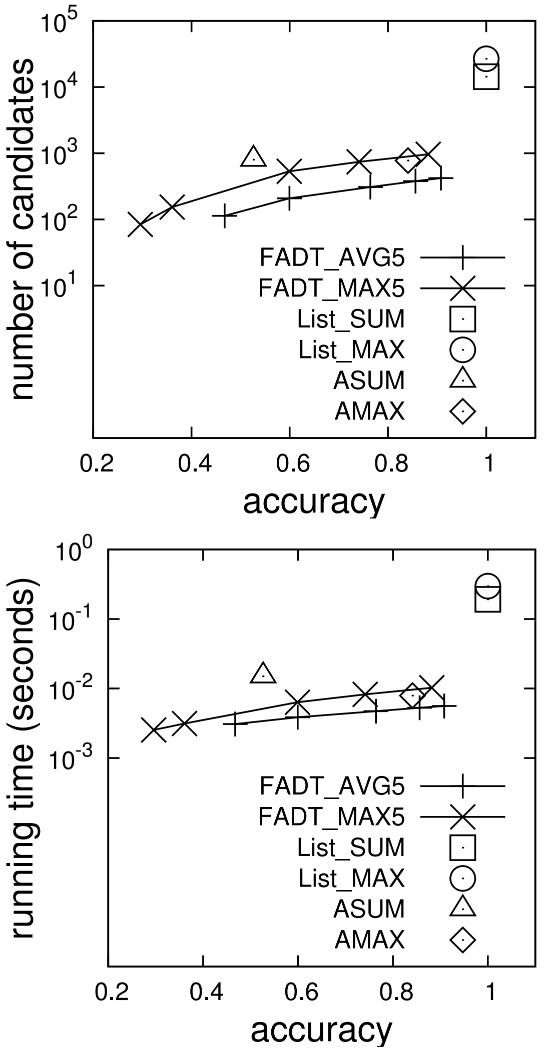
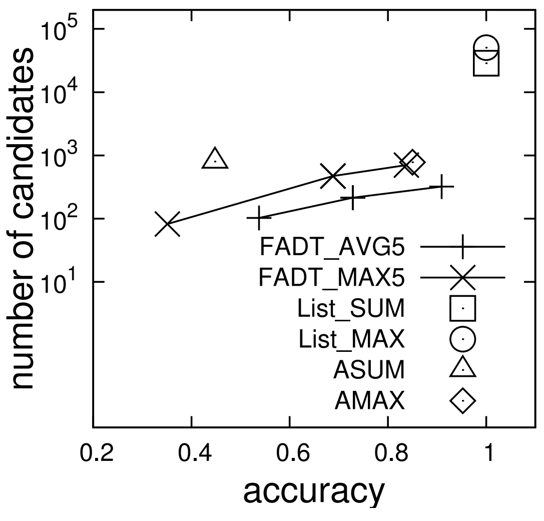


Figure 4.17 The results of using default parameters on dataset ALOI.



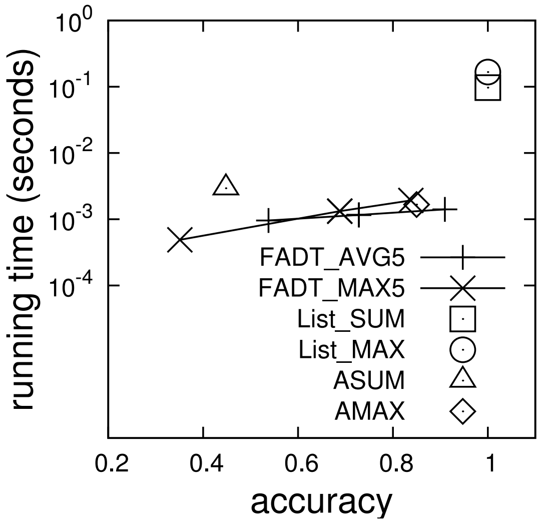
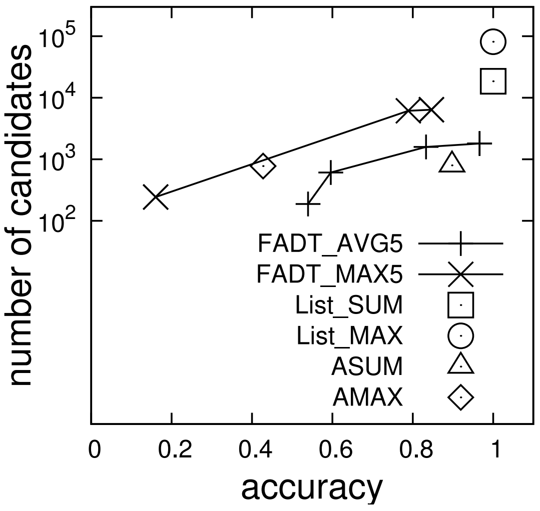


Figure 4.18 The results of using default parameters on dataset Cortina.



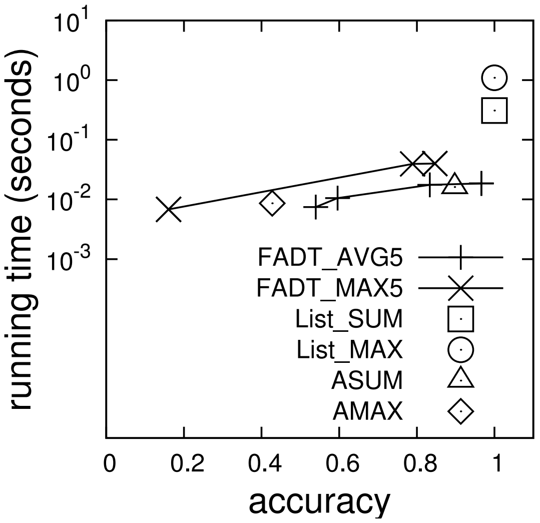


Figure 4.19 The results of using default parameters on dataset RCV2.

be 2^i for all integer choices of i in the range $[-6, 6]$, so as to cover a reasonably wide range of result accuracies. Again, some of the plots have been cropped for the sake of readability.

The results on MNIST, ALOI, Cortina and RCV2 are shown in Figure 4.16, 4.17, 4.18 and 4.19, respectively. Once again, we find that algorithms FADT_AVG5 and FADT_MAX5 outperform List by 1 to 2 orders of magnitude, in terms of both the number of candidates and the execution time, and again we observe the inability of ASUM and AMAX to achieve reasonably high accuracies.

Online Neighborhood Generation. For our last set of experiments, for all algorithm-dataset combinations, we measured the execution costs of queries using the SASH as the underlying index for online neighborhood generation. As a baseline comparison, we also computed the query results using a 'brute force' search (BF) in which the ϕ-aggregate distance from Q is explicitly computed for each object in the database. In the plots, the running time is presented as a proportion of the time required by BF. Again, for our algorithms, parameter t was chosen to be 2^i for each choice of $i \in [-6, 6]$, and again, some of the performance plots have been cropped for the sake of readability.

The results on MNIST, ALOI, Cortina and RCV2 are shown in Figure 4.20, 4.21, 4.22 and 4.23, respectively. Due to the high computational cost associated with online neighborhood generation, in terms of the execution time, the superiority of our algorithms over List is less significant than when neighbor lists have been precomputed. Nevertheless, FADT_AVG5 and FADT_MAX5 still outperform List by approximately 0.5 to 1.5 orders of magnitude. Compared to the baseline method BF, our algorithms maintain their superiority in execution cost by roughly 1 to 2 orders of magnitude, whereas the cost of Algorithm List may approach that of BF.

4.6 Conclusion

We have presented two approximation algorithms for flexible aggregate similarity search (FANN), in which the multi-step search strategy and tests of intrinsic dimensionality are combined. Theoretical analysis of the two algorithms shows both conditions under which an exact result can be guaranteed, and conditions under which an approximate result can be guaranteed with respect to a variable distance approximation ratio. Several practical heuristic extensions have also been developed. Our extensive experimental study showed that our algorithms are able to produce query results with good accuracy, while at the same time being very efficient, compared to List, ASUM and AMAX. Specifically, compared to List, our algorithms are typically more than one order of magnitude faster, while losing only approximately 10% in the accuracy of query results. While ASUM and AMAX are also quite efficient, they often fail to produce query results with high accuracy.

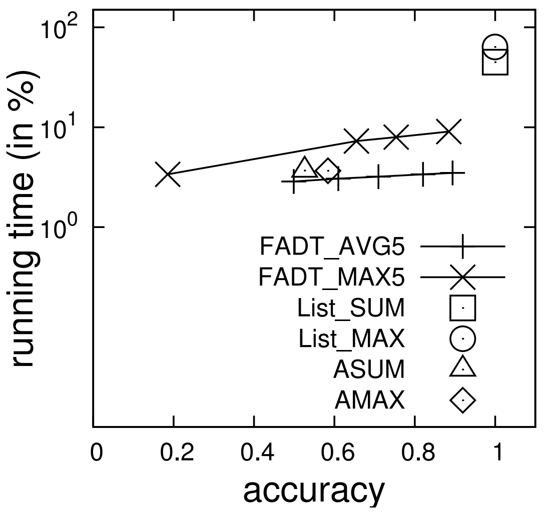


Figure 4.20 The results of using default parameters on dataset MNIST with SASH as the underlying index. The average running time of BF is about 1.67 seconds.

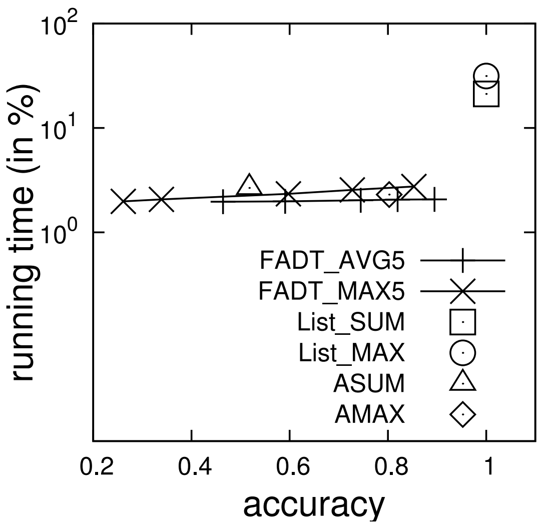


Figure 4.21 The results of using default parameters on dataset ALOI with SASH as the underlying index. The average running time of BF is about 2.16 seconds.

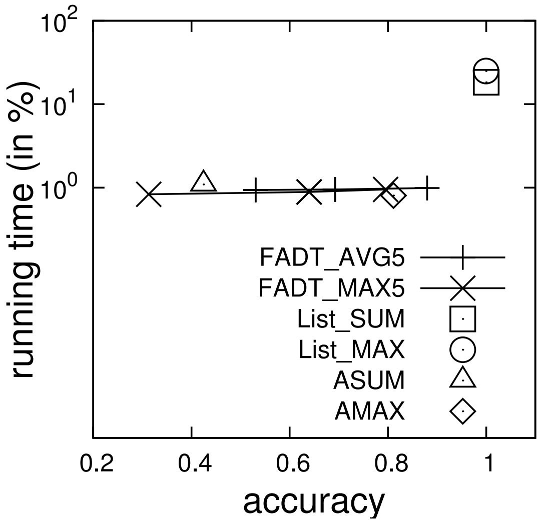


Figure 4.22 The results of using default parameters on dataset Cortina with SASH as the underlying index. The average running time of BF is about 3.92 seconds.

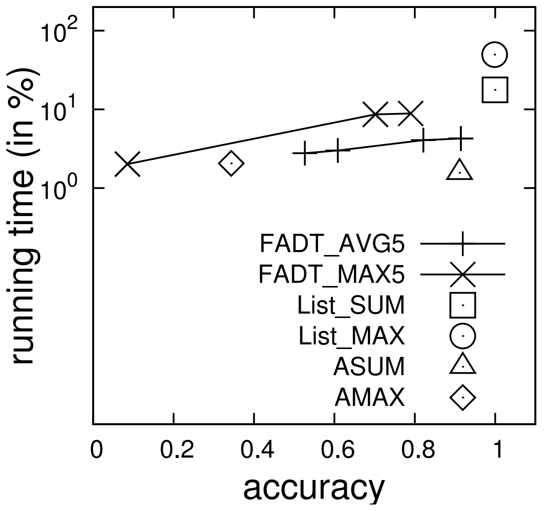


Figure 4.23 The results of using default parameters on dataset RCV2 with SASH as the underlying index. The average running time of BF is about 10.24 seconds.

CHAPTER 5

EFFICIENT ALGORITHMS FOR SIMILARITY SEARCH IN AXIS-ALIGNED SUBSPACES

In this chapter, we present the work on subspace similarity search, which aims to retrieve from the database the most similar objects to a query object with the similarity distance function being defined over a subset of features (an axis-aligned projective subspace). In particular, each query may specify an arbitrary subspace with an arbitrary number of features.

Of the two main types of similarity queries (k-NN queries and range queries), k-NN queries are often more important, due to the difficulty faced by the user in deciding range thresholds. This is especially the case for the search in subspaces, since the range values of interest will typically depend on the number of features associated with the subspace. In this work, we focus only on k-NN queries.

We now formally define the subspace search problem for k-NN queries. Given an object domain $\mathbb{U}$, let $S \subseteq \mathbb{U}$ denote a set of database objects represented as feature vectors in $\mathbb{R}^D$. The set of features will be denoted simply as $F = \{1, 2, \cdots, D\}$, with feature $i \in F$ corresponding to the i-th coordinate in the vector representation. Let $d : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ be a distance function defined for the vector space. Given an object vector $u = (u_1, \ldots, u_{|F|}) \in S$, its projection with respect to a feature subset $F' \subseteq F$ is the vector $u' = (u'_1, \ldots, u'_{|F|})$ such that for all $i \in F$, $u'_i = u_i$ whenever $i \in F'$, and $u'_i = 0$ otherwise. The feature set F' thus indicates a unique axis-aligned projective subspace to which distance calculations can be restricted.

Definition 1 (Subspace k-NN Query) *Given a query object $q \in \mathbb{U}$, a query subspace $F' \subseteq F$, and a query neighborhood size k, a subspace k-NN query $\langle q, F', k \rangle$ returns the k objects of S most similar to q, for the distance function $d_{F'}(q, u) \triangleq d(q', u')$, where q' and u' are the projections of q and u with respect to F'.*

As an example of a subspace distance function, for any given $p \in [1, \infty)$, the L_p distance between two objects $q, u \in \mathbb{U}$ restricted to the axis-aligned projective subspace F' is defined as

$$d_{F'}(q,u) = \left(\sum_{i \in F'} |q_i - u_i|^p \right)^{\frac{1}{p}} .$$

Existing solutions for the subspace search problem include the Partial VA-file (PVA) [Kriegel et al., 2006], the Dimension-Merge Index (DMI) [Bernecker et al., 2010b], the Projected R-Tree (PT) [Bernecker et al., 2010a], and the one for range queries [Lian and Chen, 2008]. Since k-NN queries are not directly supported by the last method, for the experimental comparison in Section 5.3, we restrict our attention to PVA, DMI and PT.

In this work, we present algorithms for subspace similarity search following the multi-step search strategy [Seidl and Kriegel, 1998; Houle et al., 2012b] (see Sections 3.1 and 3.2 in Chapter 3), utilizing 1-dimensional distances as lower bounds to efficiently prune the search space. The main contributions are:

- algorithms specifically tailored for subspace similarity search, both exact and approximate;
- a guide to the practical choice of an important algorithm parameter, based on a theoretical analysis of sample properties;
- an experimental evaluation across data sets of a variety of types and sizes, showing the efficiency and competitiveness of our algorithms.

The remainder of this chapter is organized as follows. Section 5.1 discusses Algorithm PVA, DMI and PT in detail. Our proposed algorithms are presented in Section 5.2. In Section 5.3, through experiments on several real-world datasets, we contrast the performance of our methods with those of existing methods. The discussion is concluded in Section 5.4.

5.1 Algorithm PVA, DMI and PT

In [Kriegel et al., 2006], PVA was proposed, which adapts the vector approximation file (VA-file) [Weber et al., 1998] to support subspace queries. The VA-file, designed for fixed-space similarity search, stores a compressed approximation of the data as a single file. At query time, the compressed approximation is scanned in its entirety, and the information is used to derive upper and lower bounds of the true distances between data objects and the query object. If there are still candidates that cannot be pruned or reported using the upper and lower distance bounds, the exact information of the candidates are accessed to get the true distances. PVA, on the other hand, stores an approximation of data on each dimension separately, and processes the search using only those 1-dimensional VA-files that correspond to dimensions involved in the query.

In [Bernecker et al., 2010b], DMI was developed, which combines multiple 1-dimensional index structures to answer subspace queries. DMI builds an index for each dimension separately (of any desired type), and utilizes those indexes with respect to the query dimensions to perform the search. Specifically, DMI retrieves candidates for the query result from the neighborhood associated with each of the query dimensions in a round robin fashion, while maintaining a lower bound for the best possible subspace distance of all the unseen objects. The algorithm terminates when at least k of the objects visited have subspace distances no greater than the lower bound. Since no unseen objects would have subspace distances smaller than the lower bound, the algorithm is guaranteed to return exact query results.

In [Bernecker et al., 2010a], PT was proposed as a redefinition of the classical search structure R-tree [Guttman, 1984] for subspace similarity search. Instead of integrating results of queries on 1-dimensional indices, PT utilizes a single index built on the full feature space (an R-tree) to answer queries with respect to subspaces. Since the minimum distance between minimum bounding rectangles (minimum bounding rectangles are used to organize data in R-trees) and the query object q in subspace F' is properly defined, the best-first search algorithm can be applied without any changes. The minimum distance (L_p distance) between a minimum bounding rectangle R and the query object q in subspace F' is defined as

$$\mathrm{mindist}_{F'}(q,R) = \left(\sum_{i \in F'} \begin{cases} (R_i^{\min} - q_i)^p & \text{if } R_i^{\min} > q_i \\ (q_i - R_i^{\max})^p & \text{if } R_i^{\max} < q_i \\ 0 & \text{otherwise} \end{cases} \right)^{\frac{1}{p}},$$

where $R_i^{\min}$ and $R_i^{\max}$ are the minimum and maximum value of rectangle R in the i-th dimension.

All the subspace similarity search methods mentioned above produce exact query results; however, as we shall see in Section 5.3, all tend to suffer greatly in terms of their computational cost.

5.2 Algorithm Design

We now present our solutions to the subspace similarity search problem. Let us first introduce some additional notation. For any object $q \in \mathbb{U}$ and any subspace $F' \subseteq F$, let $N_{F'}(q,k)$ denote the set of k-nearest neighbors of q within database S with respect to subspace distance $d_{F'}$. Ties are broken arbitrarily but consistently. Let $\delta_{F'}(q,k)$ denote the k-th smallest subspace distance (with respect to F') from q to the objects in S.

The strategy underlying our methods involves the application of multi-step search, using a lower-bounding distance function to filter a candidate set from the database, and using the target distance function to refine the candidate set to obtain the final query result. The main concern here is the determination at query time of a lower-bounding distance function suitable for the indicated subspace. Due to the exponential number of possible subspaces, it is impossible to explicitly preprocess the data for every subspace. Instead, as potential lower-bounding distance functions, we consider only the 1-dimensional distance $d_{\{i\}}$ associated with each feature $i \in F$. Assuming that the lower-bounding property holds

```
Algorithm SK_SR (query $q$, subspace $F'$, target neighborhood size $k$)
   // Preprocessing step: obtain a single ranking of all dimensions.
1: for each dimension $i \in F$ do
2:     $\mu_i \leftarrow \frac{1}{|S|} \sum_{u \in S} u_i$.
3:     $\mathrm{Var}_i \leftarrow \frac{1}{|S|} \sum_{u \in S} (u_i - \mu_i)^2$.
4: end for
5: Rank all dimensions $i \in F$ in decreasing order of $\mathrm{Var}_i$. Let $\mathfrak{R}(F)$ denote this ranking.
   // Query processing step: perform a multi-step search.
6: Among all the dimensions in subspace $F'$, select the dimension $i^*$ with the highest
   ranking according to $\mathfrak{R}(F)$.
7: Call $\mathrm{SK}(q, k)$ to produce the query result, with $d_{\{i^*\}}$ as the lower-bounding distance
   function, and $d_{F'}$ as the target distance function.
```

Figure 5.1 The description of algorithm SK_SR.

between $d_{\{i\}}$ and subspace distance $d_{F'}$ for all $i \in F'$ (which is the case for many practical distance measures, including the Euclidean distance), there are $|F'|$ lower-bounding distance functions that can be used in the search. However, practical performance may vary considerably according to the choice of $d_{\{i\}}$. In order to minimize the risk of choosing a poorly-performing lower-bounding distance, we select the distance function corresponding to the most discriminative query dimension. This is done by ranking the dimensions based on data variance, a simple yet effective ranking technique. Two ranking strategies are proposed: Single Ranking (SR) and Multiple Ranking (MR).

5.2.1 Single Ranking Strategy

The first of our proposed algorithms — SK_SR, described in Figure 5.1 — employs a single overall ranking of dimensions based on variance. There are two main phases: a preprocessing phase and a query processing phase. In the preprocessing phase, the algorithm generates a single ranking of the dimensions, in terms of the variances of the data values computed separately for each of the dimensional coordinates — the larger the data variance for a given dimension, the higher the ranking of that dimension. In the query processing phase, as the lower-bounding distance function used in multi-step search, the algorithm chooses the dimension of highest rank from among the query dimensions. When Algorithm SK (see Figure 3.1 on Page 36) is used for performing the multi-step search (in Line 7), the query result is guaranteed to be correct. As an alternative, we may also utilize the approximate multi-step algorithm MAET+ (see Figure 3.5 on Page 45); this variant of subspace similarity search will be referred to as MAET+_SR. Specifically, we make a call to MAET+(q, k, t), where $t > 0$ is a parameter governing an early termination criterion. Larger choices of t can be expected to yield query results with higher accuracies at the possible expense of computational cost. A sampling method has been designed for choosing t so that a desired proportion of potential queries can be correctly answered with high probability [Houle et al., 2012b] (see Section 3.2.3 in Chapter 3).

Note that like DMI, our search strategy requires the construction of a separate index for each of the dimensions. However, unlike DMI, our algorithms access only a single index per query, namely the most discriminative query dimension in terms of variance.

5.2.2 Multiple Ranking Strategy

The single ranking strategy has the advantage of being straightforward to apply. However, its effectiveness may be limited whenever the variance of a particular dimension differs greatly when restricted to the vicinity of differing query objects. For this reason, we have also designed a multiple ranking strategy that takes the query object into account when generating a ranking of dimensions.

Our multiple ranking strategy for subspace similarity search, SK_MR, is described in Figure 5.2. In the preprocessing step, the algorithm first samples m reference points from the database. Then, with respect to each reference point v, the algorithm determines a ranking (from highest to lowest) of all dimensions based on the variance of the coordinate values for the dimension in question, this time computed over a neighbor set of v (instead of over the entire dataset S). In the query processing step, the algorithm first finds the nearest reference point v^* of q in the query subspace (using sequential search within the reference set), and then uses the ranking of dimensions precomputed for v^* in the processing of

Algorithm **SK_MR** ($($*query* q, *subspace* F', *target neighborhood size* k, *sample size* m, *variance neighborhood size* $K)$

```
    // Preprocessing step: create multiple rankings of dimensions.
 1: Create a reference set $R \subseteq S$ by sampling $m$ points from the database, uniformly at
    random and without replacement.
 2: for each reference point $v \in R$ do
 3:     for each dimension $i \in F$ do
 4:         $\mu_{v,i} \leftarrow \frac{1}{|K|} \sum_{u \in N_{\{i\}}(v,K)} u_i$.
 5:         $\mathrm{Var}_{v,i} \leftarrow \frac{1}{|K|} \sum_{u \in N_{\{i\}}(v,K)} (u_i - \mu_{v,i})^2$.
 6:     end for
 7:     Rank all dimensions $i \in F$ in decreasing order of $\mathrm{Var}_{v,i}$. Let $\mathfrak{R}_v(F)$ denote this
    ranking.
 8: end for
    // Query processing step: perform a multi-step search.
 9: Linearly scan $R$ to find $v^*$, the nearest reference point to $q$ with respect to $d_{F'}$.
10: Select the query dimension $i^* \in F'$ with the highest ranking according to $\mathfrak{R}_{v^*}(F)$.
11: Call $\mathrm{SK}(q, k)$ to produce the query result, with $d_{\{i^*\}}$ being the lower-bounding distance
    function and $d_{F'}$ being the target distance function.
```

Figure 5.2 The description of algorithm SK_MR.

query q. Again, we may replace SK with MAET+ to derive an approximation variant, MAET+_MR.

Two parameter choices must be considered when applying the multiple ranking strategy: the number of reference points m, and the size K of the neighborhoods within which data variance is computed. As will be shown in Section 5.3, the choice of K does not greatly affect the performance, provided that it is small relative to the dataset size $|S|$. On the other hand, the number of reference points m must be chosen with more care. If m is too large, the identification of the most discriminative query dimension may become unaffordable. If m is too small, the dimension i^* selected for multi-step search may not be very discriminative for the query. We next discuss how to choose a reasonable value for m.

Determining the Reference Set Size. For the multiple ranking strategy to be effective, for any given query point q, its nearest reference point v^* should be among the nearest neighbors of q within S (all with respect to the query subspace). Otherwise, the ranking of dimensions based at v^* may fail to approximate the ranking based at q. Fortunately, the following technical lemma shows that with even a relatively small number of reference points, v^* can lie in the local neighborhood of q with high probability.

Lemma 7 (From [Houle et al., 2012b]) *Let A be a set of positive integers, and let $A' \subseteq A$ be a subset sampled uniformly at random without replacement. Given a threshold τ, let a and a' refer to the number of elements in A and A', respectively, that are no greater than τ. Take η and η' to refer to the proportion of those elements within A and A', respectively. For any real number $\phi \geq 0$, we have $\mathsf{Pr}[|\eta - \eta'| \geq \phi] \leq 2e^{-2\phi^2|A'|}$.*

Proof: Since A' is generated by uniform selection from A, random variable a' follows the hypergeometric distribution with expectation $\mathsf{E}[a'] = a|A'|/|A|$. In [Chvátal, 1979], Chvátal showed that random variable a' satisfies both $\mathsf{Pr}[\mathsf{E}[a'] \geq a' + \phi|A'|] \leq e^{-2\phi^2|A'|}$ and $\mathsf{Pr}[\mathsf{E}[a'] \leq a' - \phi|A'|] \leq e^{-2\phi^2|A'|}$. Both inequalities can be combined to yield the following error bound:

$$\begin{aligned}\mathsf{Pr}\left[|\eta - \eta'| \geq \phi\right] &= \mathsf{Pr}\left[\left|\frac{a}{|A|} - \frac{a'}{|A'|}\right| \geq \phi\right] = \mathsf{Pr}\left[\left|\frac{\mathsf{E}[a']}{|A'|} - \frac{a'}{|A'|}\right| \geq \phi\right] \\ &= \mathsf{Pr}\left[\left|\mathsf{E}\left[a'\right] - a'\right| \geq \phi|A'|\right] \leq 2e^{-2\phi^2|A'|}.\end{aligned}$$

□

To apply this lemma to the analysis of the choice of reference set size, let $A = \{1, 2, 3, \ldots, |S|\}$ represent the ranks of all the objects in S with respect to a query object q, and let $A' \subseteq A$ store the ranks of all the reference points ($|A'| = m$). Also, let τ be the rank of the reference point v^*, which implies that $\eta' = 1/|A'|$. A small value of η would therefore indicate that v^* is in the local neighborhood of q, as desired. From Lemma 7, we know that the probability of η deviating from $\eta' = 1/|A'|$ by more than $\phi \geq 0$ is at most $2e^{-2\phi^2|A'|}$. That is, the probability of η being significantly larger than $1/|A'|$ vanishes quickly as the sample size $|A'|$ grows. In practice, even small sample sizes allow us to obtain reasonably small values of η with high probability. For example, if $|A'| = 5{,}000$

and $\phi = 0.02$, the lemma indicates that the probability of $\eta \geq 0.0202$ is at most 0.037, or equivalently, the probability of $\eta < 0.0202$ is at least 0.963.

5.3 Experimental Results

In this section, we present the results of our experimentation. We compared our algorithms with the state-of-the-art approaches PVA, PT and DMI.

5.3.1 Experimental Framework

Data Sets. Five publicly-available data sets were considered for the experimentation, so as to compare across a variety of set sizes, dimensions and data types.

- The Amsterdam Library of Object Images (ALOI) [Geusebroek et al., 2005] consists of $110,250$ images of 1000 small objects taken from different viewpoints and illumination directions. The images are represented by 641-dimensional feature vectors based on color and texture histograms (for a detailed description of the image features, see [Boujemaa et al., 2001]).
- The MNIST data set [LeCun et al., 1998] consists of $70,000$ images of handwritten digits from 500 different writers, with each image represented by 784 gray-scale texture values.
- The Cortina data set [1] consists of $1,088,864$ images gathered from the World Wide Web. Each image is represented by a 74-dimensional feature vector based on homogeneous texture, dominant color and edge histograms.
- The Forest Cover Type set (FCT) [Bache and Lichman, 2013] consists of $581,012$ data points, with each representing a 30×30 square meter area of forest. Each point is represented by 54 attributes, associated with elevation, aspect, slope and other geographical characteristics.
- The ANN_SIFT data set [Jégou et al., 2011] consists of 10^7 SIFT descriptors [Lowe, 2004] of 128 dimensions. The SIFT descriptors were extracted from approximately 10^6 general images.

Methodology. For each test, 1000 queries were generated at random, each consisting of an object q selected from the database, and a query subspace F'. Unless stated otherwise, the number of query dimensions was $|F'| = 8$, and the target neighborhood size was $k = 10$.

[1]http://www.scl.ece.ucsb.edu/datasets/index.htm (accessed on November 9, 2014).

Two quantities were measured for the evaluation: query result accuracy and execution time. The results were reported as averages over the 1000 queries performed. The execution time is shown as a proportion of the time needed for a sequential search of the entire dataset. For each query, the accuracy of its k-NN query result is defined as the proportion of the result falling within the true k-NN (subspace) distance to q:

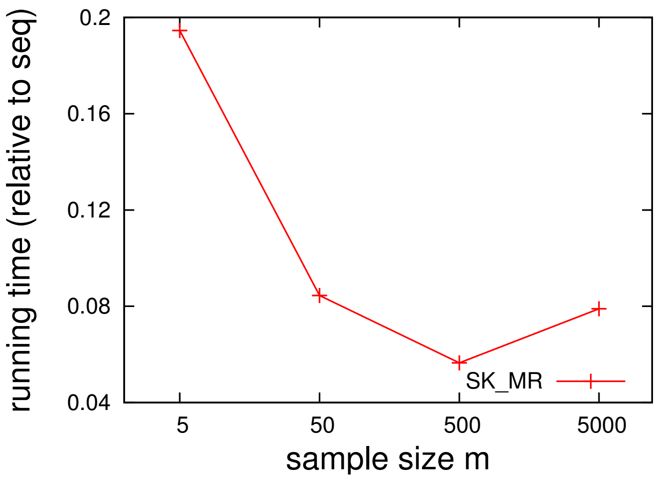
$$\frac{|\{v \in Y \mid d_{F'}(q, v) \leq \delta_{F'}(q, k)\}|}{k},$$

where Y denotes the k-NN query result of q in subspace F' ($|Y| = k$). The Euclidean distance was used for all experiments.

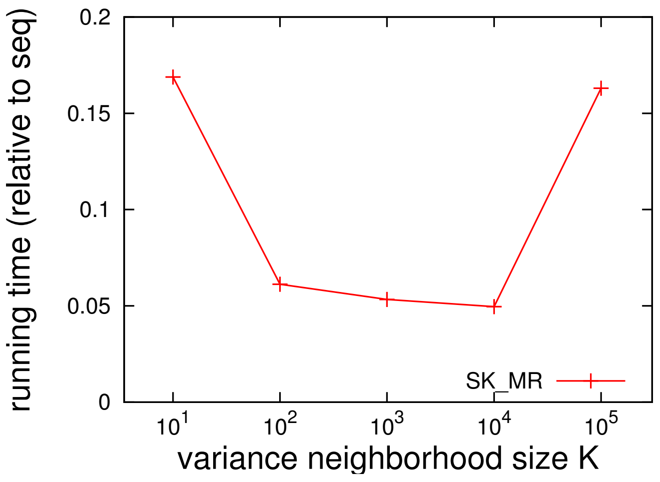
5.3.2 Effects of Varying m and K on the Multiple Ranking Strategy

For the first set of experiments, for all of the datasets under consideration, we tested the effects on the multiple ranking strategy due to variation of the sample size m and variance neighborhood size K. When varying the sample size m, the variance neighborhood size K was chosen to be approximately 1% of the dataset size: specifically, the choices were $K = 10^3$ for ALOI and MNIST, $K = 10^4$ for Cortina and FCT, and $K = 10^5$ for ANN_SIFT. When varying K, the sample size m was fixed at 500 for all datasets tested. Since we observed similar trends in the results for all datasets, we show the results of varying m and K only for the ALOI dataset.

The results for varying m are shown in Figure 5.3(a). Here, we see that $m = 500$ is a sufficiently-large sample size for multiple ranking strategy to be effective, which is better than indicated by the theoretical analysis. From Lemma 7, we know that if $m = 500$, then for any dataset with any number of data points, the probability of $\eta < 0.062$ is at least 0.945 ($\phi = 0.06$). Recall that the effectiveness of the multiple ranking strategy is expected to increase as η diminishes. Our experimental findings show that the value of η in practice is typically much smaller than what the analysis indicates. In order to reduce the computational cost of the experimentation, we therefore set $m = 500$ for all remaining experiments.



(a) $K = 1000$



(b) $m = 500$

Figure 5.3 The effects of varying m and K for the multiple ranking strategy, with dataset ALOI.

Figure 5.3(b) shows the results of varying K. As expected, the variance neighborhood size K does not greatly affect the performance, provided that it is set to reasonably small values relative to the dataset size. For all remaining experiments, we set $K = 10^3$ for ALOI and MNIST, $K = 10^4$ for Cortina and FCT, and $K = 10^5$ for ANN_SIFT.

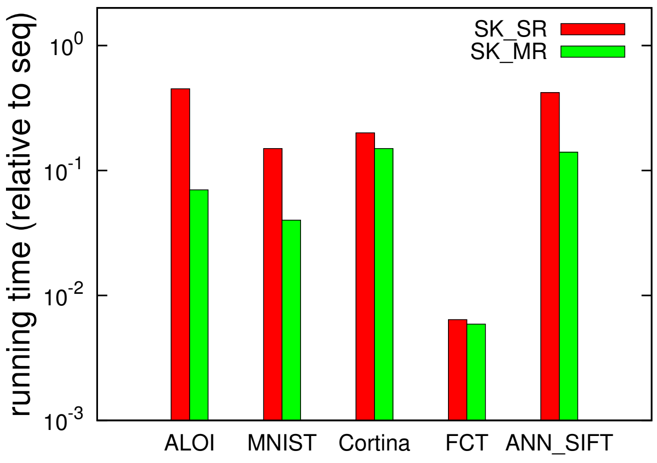


Figure 5.4 The comparison of single ranking and multiple ranking on all datasets tested.

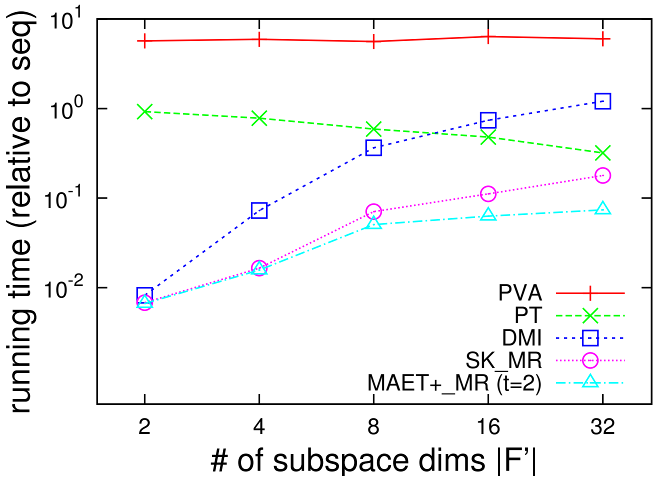


Figure 5.5 The results of varying $|F'|$ on dataset ALOI, with $k = 10$. The average accuracy of MAET+_MR with $t = 2$ is approximately 92%.

5.3.3 Comparison of Single Ranking and Multiple Ranking

We next compared the performance of the single ranking and multiple ranking strategies; the results are shown in Figure 5.4. Unsurprisingly, multiple ranking outperformed single ranking for all datasets tested. In all experiments involving competing methods, we show a comparison of results only for multiple ranking.

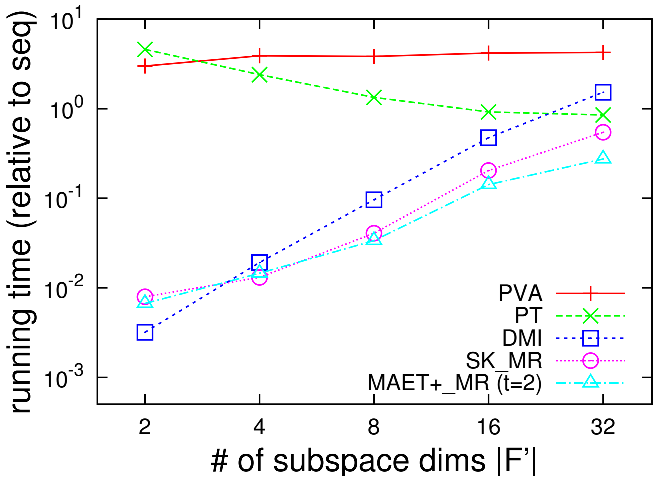


Figure 5.6 The results of varying $|F'|$ on dataset MNIST, with $k = 10$. The average accuracy of MAET+_MR with $t = 2$ is approximately 90%.

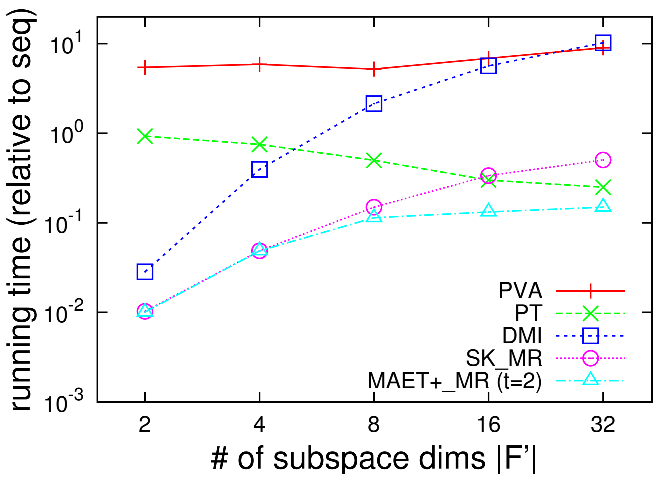


Figure 5.7 The results of varying $|F'|$ on dataset Cortina, with $k = 10$. The average accuracy of MAET+_MR with $t = 2$ is approximately 88%.

5.3.4 Comparison with Other Methods

We conducted two sets of experiments for the comparison of our algorithms with competing methods, varying each of two parameters in turn: the number of subspace dimensions $|F'|$, and the target neighborhood size k. Specifically, we varied $|F'|$ from 2 to 32 while fixing $k = 10$, and varied k from 5 to 40 while fixing $|F'| = 8$.

The results of varying $|F'|$ on dataset ALOI, MNIST, Cortina, FCT and ANN_SIFT are shown in Figure 5.5, 5.6, 5.7, 5.8 and 5.9, respectively. For all datasets and all

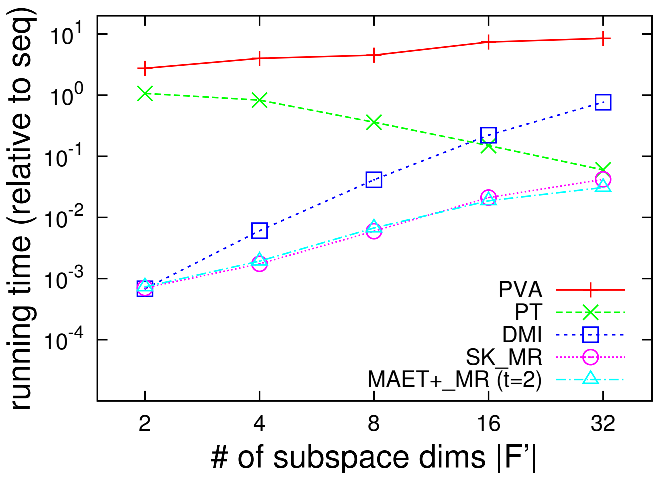


Figure 5.8 The results of varying $|F'|$ on dataset FCT, with $k = 10$. The average accuracy of MAET+_MR with $t = 2$ is approximately 97%.

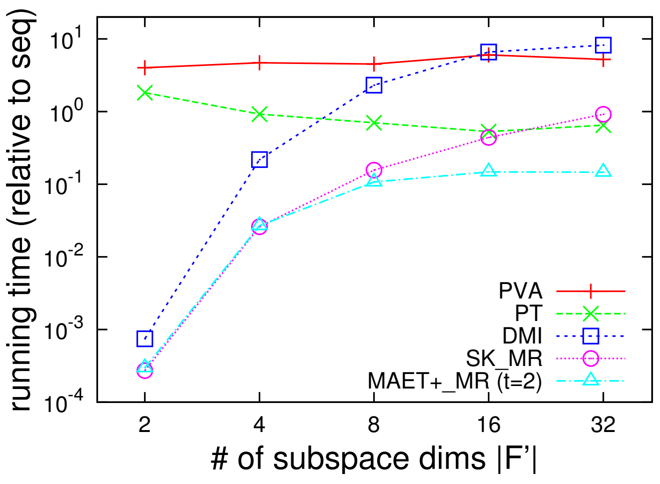


Figure 5.9 The results of varying $|F'|$ on dataset ANN_SIFT, with $k = 10$. The average accuracy of MAET+_MR with $t = 2$ is approximately 90%.

choices of $|F'|$, our proposed methods generally outperform their competitors. Among all the methods tested, PVA is the most expensive, perhaps due to its use of sequential scan.

PT utilizes an R-tree built on the full-dimensional space to answer queries in subspaces; consequently, one would expect it to be less effective for subspaces in which $d_{F'}$ differs greatly from d_F. This can explain the improvement in the performance of PT as the number of subspace dimensions increases. Nevertheless, due to the limits on the performance of R-trees for spaces of even moderate dimensionality, PT will still become prohibitively expensive as the number of subspace dimensions grows.

DMI processes queries by aggregating partial results across neighborhoods with respect to every query dimension. The aggregation may become prohibitively expensive as the number of subspace dimensions increases. In contrast, our algorithms avoid expensive aggregation by restricting the processing to a single query dimension.

Relative to SK_MR, we observe that for high subspace dimensionality, MAET+_MR can achieve a significant improvement in running time while still achieving a high level of accuracy. We note that as the value of $|F'|$ increases, the computational cost of all tested methods must eventually tend to that of sequential search, as one would expect due to the curse of dimensionality.

The results of varying k on dataset ALOI, MNIST, Cortina, FCT and ANN_SIFT are shown in Figure 5.10, 5.11, 5.12, 5.13 and 5.14, respectively. Again, our proposed methods generally outperform their competitors, with MAET+_MR achieving a slight improvement in running time over SK_MR, at the cost of a slight loss of accuracy. We also observe that the behaviors of all tested methods are quite stable with respect to k.

Finally, Figure 5.15 shows the preprocessing costs of all methods considered in our experimentation. While the preprocessing costs of our methods is substantial, the costs are justifiable in light of their improved performance at query time.

5.4 Conclusion

We have presented new solutions for the subspace similarity search problem based on multi-step search, utilizing 1-dimensional lower-bounding distance functions associated with individual features for the efficient pruning of the search space. Among possible choices of lower-bounding distance functions that can be used for the specified subspace, we select the one corresponding to the most discriminative query dimension. This is done by ranking the dimensions based on data variance. Two ranking strategies have been proposed: Single Ranking (SR) and Multiple Ranking (MR).

To apply the MR strategy, two parameter choices must be considered: the sample size m and the variance neighborhood size K. The choice of K does not greatly affect the performance, provided that it is small relative to the dataset size $|S|$, for example, $K = 0.01 \times |S|$. On the other hand, the sample size m must be chosen with more care. As disscussed in Section 5.2.2, the choice of m can be guided by the analysis of sampling properties.

Our extensive experimental study showed that MR is able to perform better than SR for all datasets tested, and our algorithms (SK_MR and MAET+_MR) are able to outperform their state-of-the-art competitors (PVA, PT and DMI) for a relatively wide range of subspace dimensions. Relative to SK_MR, we observed that for high subspace dimensionality, MAET+_MR can achieve a significant improvement in running time while losing only approximately 10% in the accuracy of query results.

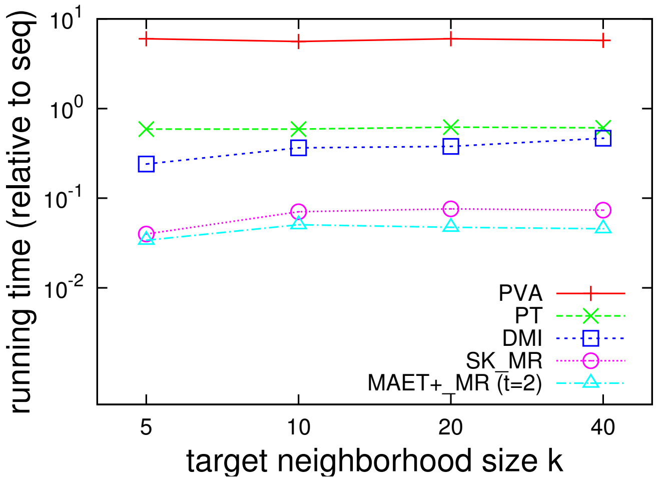


Figure 5.10 The results of varying k on dataset ALOI, with $|F'| = 8$. The average accuracy of MAET+_MR with $t = 2$ is approximately 88%.

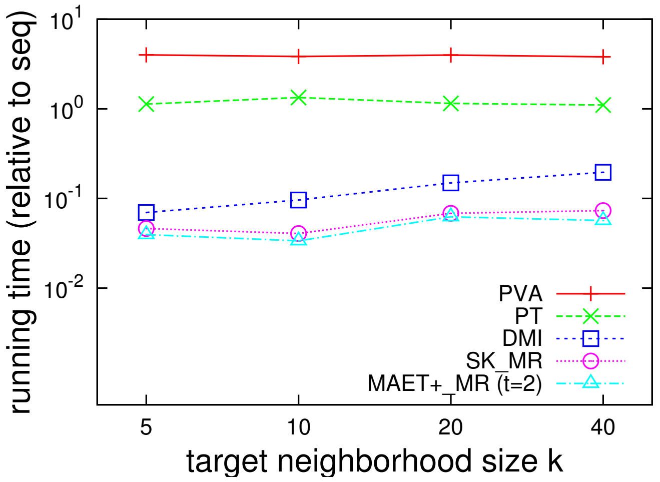


Figure 5.11 The results of varying k on dataset MNIST, with $|F'| = 8$. The average accuracy of MAET+_MR with $t = 2$ is approximately 96%.

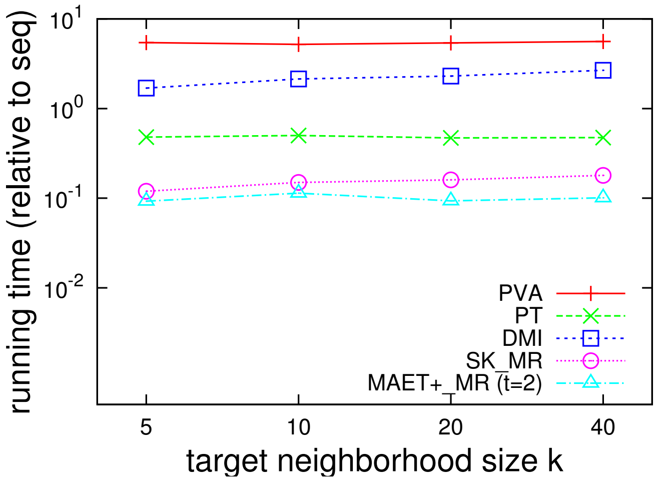


Figure 5.12 The results of varying k on dataset Cortina, with $|F'| = 8$. The average accuracy of MAET+_MR with $t = 2$ is approximately 89%.

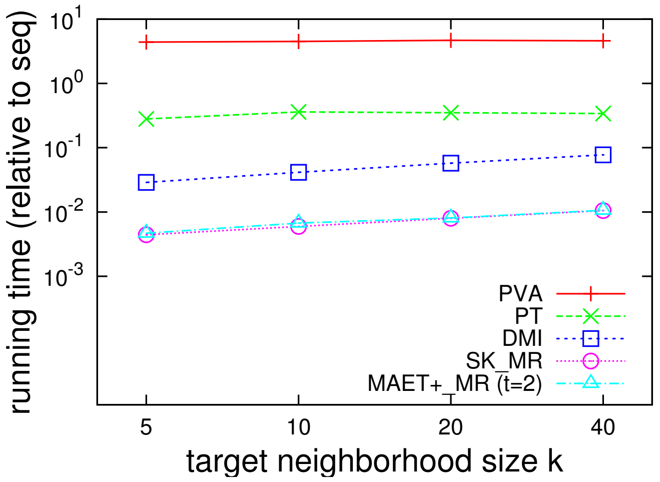


Figure 5.13 The results of varying k on dataset FCT, with $|F'| = 8$. The average accuracy of MAET+_MR with $t = 2$ is approximately 98%.

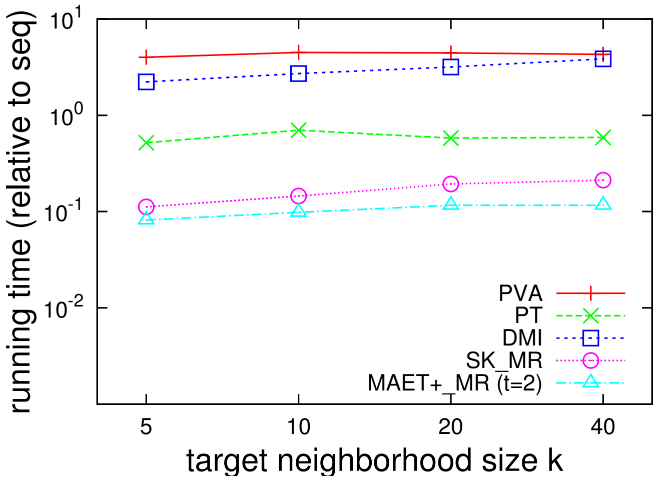


Figure 5.14 The results of varying k on dataset ANN_SIFT, with $|F'| = 8$. The average accuracy of MAET+_MR with $t = 2$ is approximately 92%.

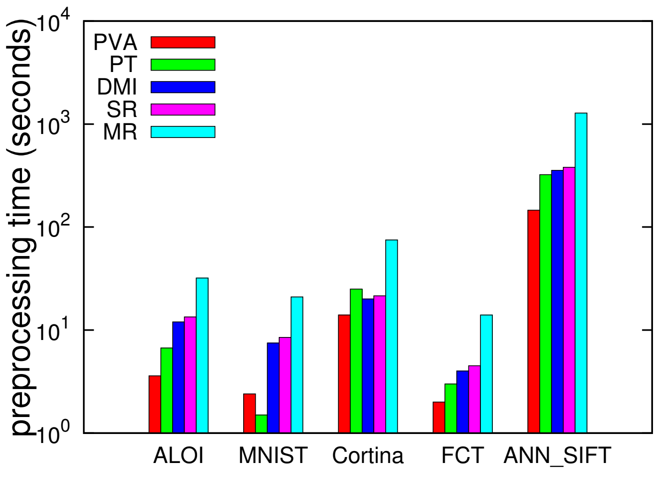


Figure 5.15 Preprocessing costs for all datasets tested.

CHAPTER 6

CONCLUSION AND FUTURE WORK

This dissertation mainly investigates the possibility of utilizing intrinsic dimension in the design and analysis of similarity search algorithms. Based on the generalized expansion dimension (GED) model [Houle et al., 2012a], we have designed solutions for three variants of the similarity search problem, including adaptive similarity search, flexible aggregate similarity search, and subspace similarity search. Both the theoretical analysis and the experimental results presented in this dissertation give evidence for the potential benefits that come with the knowledge of intrinsic dimension of data.

To solve the adaptive similarity search problem, we have designed an approximate multi-step search algorithm. We have shown that our approach is able to achieve better performance than previous multi-step algorithms. However, all multi-step algorithms rely on the existence of a lower-bounding distance function that provides reasonable approximations of target distance functions. If such a lower-bounding distance function does not exist, then the multi-step algorithms may fail to efficiently handle adaptive similarity queries. How to solve the adaptive similarity search problem in such situations is an open research question.

For the flexible aggregate similarity search (FANN) problem, We have presented two approximation algorithms, one for the *sum* variant and the other for the *max* variant. Our experimental study showed that our algorithms are able to produce query results with good accuracy, while at the same time being very efficient, compared to previous approaches. One limitation of our proposed algorithms is that they can only be applied in L_p spaces, since they require the computation of the minimum enclosing ball. How to design an effective and efficient algorithm for flexible aggregate similarity search in general metric spaces is an interesting research problem.

In our proposed multi-step subspace search algorithms, 1-dimensional lower-bounding distances are utilized in the pruning of the search space. One possible direction for future research may include the investigation of multi-dimensional lower-bounding distances for pruning, such as 2-dimensional or 3-dimensional distances. Although multi-dimensional distances are expected to provide a tighter lower bound on the target distance, they cover fewer combinations of query dimensions, and thus may be only of limited practicality. For example, if we consider using 2-dimensional lower-bounding distances for pruning, we must build an index for each combination of two dimensions, in order to be able to handle all possible choices of target distances. Suppose there are D dimensions in total, we need to build exactly $\mathrm{D}(\mathrm{D}-1)/2$ indexes, which may not be practical when the value of D is large. The practicality issue will get more severe if we consider using 3-dimensional or other higher-dimensional distances for pruning. How to practically utilize multi-dimensional lower-bounding distances for pruning is an interesting research issue to follow.

In this dissertation, the GED model is utilized as the basis in the design of search algorithms. Specifically, tests of GED are used to dynamically guide the decisions made by search algorithms. GED may not be able to provide the most stable estimates of intrinsic dimension of data; however, it can be evaluated quite fast. Therefore, we consider GED as a good choice for integration into the search, since GED test introduces little overhead. One may consider integrating other models of intrinsic dimension into the search so long as the models are able to provide stable estimates of intrinsic dimension, and will cause little overhead.

The application of intrinsic dimension is not limited to the area of similarity search. For other areas such as feature selection, one may find applications of intrinsic dimension as well. In the following, we give a brief discussion about how to utilize intrinsic dimension for feature selection.

Given a set of data objects represented as finite-dimensional feature vectors, the goal of feature selection is to find a projection of the data onto a subset of features that maximizes some objective function. In unsupervised settings, such objective functions may reward the preservation of object-to-object distances; in supervised settings, such objective functions may penalize small distances between objects from different classes. In [Houle, 2013], Houle showed that intrinsic dimension and indiscriminability are identical within the framework of continuous intrinsic dimension. Accordingly, when trying to select a subset of features that maximizes the degree of discriminability, one may choose the features with the lowest intrinsic dimension. Some initial (unpublished) results obtained by Chelly and Houle have demonstrated the promising aspects of this new approach to feature selection.

All the applications of intrinsic dimension discussed in this dissertation demonstrate that the study of intrinsic dimension is very important and more investigation should be made both within and beyond the area of similarity search.

BIBLIOGRAPHY

[Agrawal et al., 1993] Agrawal, R., Faloutsos, C., and Swami, A. N. (1993). Efficient similarity search in sequence databases. In *International Conference on Foundations of Data Organization and Algorithms*, pages 69–84.

[Amsaleg et al., 2014] Amsaleg, L., Chelly, O., Furon, T., Girard, S., Houle, M. E., and Nett, M. (2014). Estimating continuous intrinsic dimensionality. Technical Report NII-2014-001E, National Institute of Informatics, Tokyo, Japan.

[Andoni et al., 2008] Andoni, A., Fagin, R., Kumar, R., Pătraşcu, M., and Sivakumar, D. (2008). Corrigendum to "efficient similarity search and classification via rank aggregation" by R. Fagin, R. Kumar and D. Sivakumar (SIGMOD'03). In *International Conference on Management of Data*, pages 1375–1376.

[Andoni and Indyk, 2006] Andoni, A. and Indyk, P. (2006). Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Symposium on Foundations of Computer Science*, pages 459–468.

[Anh et al., 2001] Anh, V. N., de Kretser, O., and Moffat, A. (2001). Vector-space ranking with effective early termination. In *International Conference on Research and Development in Information Retrieval*, pages 35–42.

[Assent et al., 2006] Assent, I., Wenning, A., and Seidl, T. (2006). Approximation techniques for indexing the earth mover's distance in multimedia databases. In *International Conference on Data Engineering*.

[Bache and Lichman, 2013] Bache, K. and Lichman, M. (2013). UCI machine learning repository. http://archive.ics.uci.edu/ml/ (accessed on November 9, 2014).

[Bartolini et al., 2008] Bartolini, I., Ciaccia, P., and Patella, M. (2008). Efficient sort-based skyline evaluation. *Transactions on Database Systems*, 33(4):31:1–31:49.

[Bawa et al., 2003] Bawa, M., Bayardo, R. J., Jr., and Rajagopalan, S. (2003). Make it fresh, make it quick - searching a network of personal webservers. In *International World Wide Web Conference*, pages 577–586.

[Beckmann et al., 1990] Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B. (1990). The R*-tree: an efficient and robust access method for points and rectangles. In *International Conference on Management of Data*, pages 322–331.

[Berchtold et al., 1996] Berchtold, S., Keim, D. A., and Kriegel, H.-P. (1996). The X-tree : an index structure for high-dimensional data. In *International Conference on Very Large Data Bases*, pages 28–39.

[Bernecker et al., 2010a] Bernecker, T., Emrich, T., Graf, F., Kriegel, H.-P., Kröger, P., Renz, M., Schubert, E., and Zimek, A. (2010a). Subspace similarity search: efficient k-NN queries in arbitrary subspaces. In *International Conference on Scientific and Statistical Database Management*, pages 555–564.

[Bernecker et al., 2010b] Bernecker, T., Emrich, T., Graf, F., Kriegel, H.-P., Kröger, P., Renz, M., Schubert, E., and Zimek, A. (2010b). Subspace similarity search using the ideas of ranking and top-k retrieval. In *ICDE Workshop DBRank*, pages 4–9.

[Bernecker et al., 2011] Bernecker, T., Houle, M. E., Kriegel, H.-P., Kröger, P., Renz, M., Schubert, E., and Zimek, A. (2011). Quality of similarity rankings in time series. In *International Symposium on Advances in Spatial and Temporal Databases*, pages 422–440.

[Beyer et al., 1999] Beyer, K. S., Goldstein, J., Ramakrishnan, R., and Shaft, U. (1999). When is "nearest neighbor" meaningful? In *International Conference on Database Theory*, pages 217–235.

[Beygelzimer et al., 2006] Beygelzimer, A., Kakade, S., and Langford, J. (2006). Cover trees for nearest neighbor. In *International Conference on Machine Learning*, pages 97–104.

[Borzsony et al., 2001] Borzsony, S., Kossmann, D., and Stocker, K. (2001). The skyline operator. In *International Conference on Data Engineering*, pages 421–430.

[Boujemaa et al., 2001] Boujemaa, N., Fauqueur, J., Ferecatu, M., Fleuret, F., Gouet, V., Saux, B. L., and Sahbi, H. (2001). IKONA: interactive generic and specific image retrieval. In *International Workshop on Multimedia Content-Based Indexing and Retrieval*.

[Breunig et al., 2000] Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. (2000). LOF: identifying density-based local outliers. *SIGMOD Record*, 29(2):93–104.

[Brin, 1995] Brin, S. (1995). Near neighbor search in large metric spaces. In *International Conference on Very Large Data Bases*, pages 574–584.

[Bruno et al., 2002] Bruno, N., Gravano, L., and Marian, A. (2002). Evaluating top-k queries over web-accessible databases. In *International Conference on Data Engineering*, pages 369–380.

[Bruske and Sommer, 1998] Bruske, J. and Sommer, G. (1998). Intrinsic dimensionality estimation with optimally topology preserving maps. *Transactions on Pattern Analysis and Machine Intelligence*, 20(5):572–575.

[Camastra and Vinciarelli, 2002] Camastra, F. and Vinciarelli, A. (2002). Estimating the intrinsic dimension of data with a fractal-based method. *Transactions on Pattern Analysis and Machine Intelligence*, 24(10):1404–1407.

[Carmel et al., 2001] Carmel, D., Cohen, D., Fagin, R., Farchi, E., Herscovici, M., Maarek, Y. S., and Soffer, A. (2001). Static index pruning for information retrieval systems. In *International Conference on Research and Development in Information Retrieval*, pages 43–50.

[Chandola et al., 2009] Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: a survey. *ACM Computing Surveys*, 41(3):15:1–15:58.

[Chandrasekaran and Tamir, 1989] Chandrasekaran, R. and Tamir, A. (1989). Open questions concerning weiszfeld's algorithm for the fermat-weber location problem. *Mathematical Programming, Series A*, 44:293–295.

[Chang and Hwang, 2002] Chang, K. C. and Hwang, S. (2002). Minimal probing: supporting expensive predicates for top-k queries. In *International Conference on Management of Data*, pages 346–357.

[Chang et al., 2000] Chang, Y.-C., Bergman, L., Castelli, V., Li, C.-S., Lo, M.-L., and Smith, J. R. (2000). The onion technique: indexing for linear optimization queries. In *International Conference on Management of Data*, pages 391–402.

[Charikar, 2002] Charikar, M. (2002). Similarity estimation techniques from rounding algorithms. In *Symposium on Theory of Computing*, pages 380–388.

[Chaudhuri et al., 2004] Chaudhuri, S., Gravano, L., and Marian, A. (2004). Optimizing top-k selection queries over multimedia repositories. *Transactions on Knowledge and Data Engineering*, 16(8):992–1009.

[Chávez et al., 2001] Chávez, E., Navarro, G., Baeza-Yates, R., and Marroquín, J. L. (2001). Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321.

[Chomicki et al., 2003] Chomicki, J., Godfrey, P., Gryz, J., and Liang, D. (2003). Skyline with presorting. In *International Conference on Data Engineering*, pages 717–719.

[Chvátal, 1979] Chvátal, V. (1979). The tail of the hypergeometric distribution. *Discrete Mathematics*, 25:285–287.

[Ciaccia et al., 1997] Ciaccia, P., Patella, M., and Zezula, P. (1997). M-tree: an efficient access method for similarity search in metric spaces. In *International Conference on Very Large Data Bases*, pages 426–435.

[de Berg et al., 1997] de Berg, M., van Kreveld, M., Overmars, M., and Schwarzkopf, O. (1997). *Computational geometry: algorithms and applications*. Secaucus, NJ, USA: Springer.

[Fagin, 1999] Fagin, R. (1999). Combining fuzzy information from multiple systems. *Journal of Computer and System Sciences*, 58(1):83–99.

[Fagin et al., 2001] Fagin, R., Lotem, A., and Naor, M. (2001). Optimal aggregation algorithms for middleware. In *Symposium on Principles of Database Systems*, pages 102–113.

[Fagin et al., 2003] Fagin, R., Lotem, A., and Naor, M. (2003). Optimal aggregation algorithms for middleware. *Journal of Computer and System Sciences*, 66(4):614–656.

[Faloutsos et al., 1994] Faloutsos, C., Barber, R., Flickner, M., Hafner, J., Niblack, W., Petkovic, D., and Equitz, W. (1994). Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3(3-4):231–262.

[Faloutsos and Kamel, 1994] Faloutsos, C. and Kamel, I. (1994). Beyond uniformity and independence: analysis of R-trees using the concept of fractal dimension. In *Symposium on Principles of Database Systems*, pages 4–13.

[Ferhatosmanoglu et al., 2000] Ferhatosmanoglu, H., Tuncel, E., Agrawal, D., and El Abbadi, A. (2000). Vector approximation based indexing for non-uniform high dimensional data sets. In *International Conference on Information and Knowledge Management*, pages 202–209.

[Fukunaga, 1990] Fukunaga, K. (1990). *Introduction to statistical pattern recognition*. San Diego, CA, USA: Elsevier.

[Fukunaga and Olsen, 1971] Fukunaga, K. and Olsen, D. (1971). An algorithm for finding intrinsic dimensionality of data. *Transactions on Computers*, C-20(2):176–183.

[Gershenfeld, 1999] Gershenfeld, N. A. (1999). *The nature of mathematical modeling*. New York, NY, USA: Cambridge University Press.

[Geusebroek et al., 2005] Geusebroek, J. M., Burghouts, G. J., and Smeulders, A. W. M. (2005). The amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112.

[Godfrey et al., 2005] Godfrey, P., Shipley, R., and Gryz, J. (2005). Maximal vector computation in large data sets. In *International Conference on Very Large Data Bases*, pages 229–240.

[Golub and Van Loan, 1996] Golub, G. H. and Van Loan, C. F. (1996). *Matrix Computations*. Baltimore, MD, USA: Johns Hopkins University Press.

[Gromov and Milman, 1983] Gromov, M. and Milman, V. D. (1983). A topological application of the isoperimetric inequality. *American Journal of Mathematics*, 105(4):843–854.

[Güntzer et al., 2000] Güntzer, U., Balke, W.-T., and Kießling, W. (2000). Optimizing multi-feature queries for image databases. In *International Conference on Very Large Data Bases*, pages 419–428.

[Güntzer et al., 2001] Güntzer, U., Balke, W.-T., and Kießling, W. (2001). Towards efficient multi-feature queries in heterogeneous environments. In *International Conference on Information Technology: coding and Computing*, pages 622–628.

[Gupta et al., 2003] Gupta, A., Krauthgamer, R., and Lee, J. (2003). Bounded geometries, fractals, and low-distortion embeddings. In *Symposium on Foundations of Computer Science*, pages 534–543.

[Guttman, 1984] Guttman, A. (1984). R-trees: a dynamic index structure for spatial searching. In *International Conference on Management of Data*, pages 47–57.

[Hafner et al., 1995] Hafner, J., Sawhney, H. S., Equitz, W., Flickner, M., and Niblack, W. (1995). Efficient color histogram indexing for quadratic form distance functions. *Transactions on Pattern Analysis and Machine Intelligence*, 17:729–736.

[Han and Kamber, 2006] Han, J. and Kamber, M. (2006). *Data mining: concepts and techniques*. San Francisco, CA, USA: Morgan Kaufmann.

[Hellerstein et al., 1995] Hellerstein, J. M., Naughton, J. F., and Pfeffer, A. (1995). Generalized search trees for database systems. In *International Conference on Very Large Data Bases*, pages 562–573.

[Hjaltason and Samet, 1995] Hjaltason, G. R. and Samet, H. (1995). Ranking in spatial databases. In *International Symposium on Advances in Spatial Databases*, pages 83–95.

[Houle, 2013] Houle, M. (2013). Dimensionality, discriminability, density and distance distributions. In *International Conference on Data Mining Workshops*, pages 468–473.

[Houle et al., 2012a] Houle, M. E., Kashima, H., and Nett, M. (2012a). Generalized expansion dimension. In *ICDM Workshop on Practical Theories for Exploratory Data Mining*, pages 587–594.

[Houle et al., 2010] Houle, M. E., Kriegel, H.-P., Kröger, P., Schubert, E., and Zimek, A. (2010). Can shared-neighbor distances defeat the curse of dimensionality? In *International Conference on Scientific and Statistical Database Management*, pages 482–500.

[Houle et al., 2012b] Houle, M. E., Ma, X., Nett, M., and Oria, V. (2012b). Dimensional testing for multi-step similarity search. In *International Conference on Data Mining*, pages 299–308.

[Houle and Nett, 2013] Houle, M. E. and Nett, M. (2013). Rank cover trees for nearest neighbor search. In *International Conference on Similarity Search and Applications*, pages 16–29.

[Houle and Nett, 2014] Houle, M. E. and Nett, M. (2014). Rank-based similarity search: reducing the dimensional dependence. *Transactions on Pattern Analysis and Machine Intelligence*.

[Houle and Sakuma, 2005] Houle, M. E. and Sakuma, J. (2005). Fast approximate similarity search in extremely high-dimensional data sets. In *International Conference on Data Engineering*, pages 619–630.

[Hristidis and Papakonstantinou, 2004] Hristidis, V. and Papakonstantinou, Y. (2004). Algorithms and applications for answering ranked queries using ranked views. *The VLDB Journal*, 13(1):49–70.

[Indyk and Motwani, 1998] Indyk, P. and Motwani, R. (1998). Approximate nearest neighbors: towards removing the curse of dimensionality. In *Symposium on Theory of Computing*, pages 604–613.

[Ishikawa et al., 1998] Ishikawa, Y., Subramanya, R., and Faloutsos, C. (1998). Mindreader: querying databases through multiple examples. In *International Conference on Very Large Data Bases*, pages 218–227.

[Jagadish et al., 2005] Jagadish, H. V., Ooi, B. C., Tan, K.-L., Yu, C., and Zhang, R. (2005). iDistance: an adaptive B$^+$-tree based indexing method for nearest neighbor search. *Transactions on Database Systems*, 30:364–397.

[Jégou et al., 2011] Jégou, H., Tavenard, R., Douze, M., and Amsaleg, L. (2011). Searching in one billion vectors: re-rank with source coding. In *International Conference on Acoustics, Speech and Signal Processing*, pages 861–864.

[Kanth et al., 1999] Kanth, K. R., Agrawal, D., Abbadi, A. E., and Singh, A. (1999). Dimensionality reduction for similarity searching in dynamic databases. *Computer Vision and Image Understanding*, 75(12):59–72.

[Karger and Ruhl, 2002] Karger, D. R. and Ruhl, M. (2002). Finding nearest neighbors in growth-restricted metrics. In *Symposium on Theory of Computing*, pages 741–750.

[Karhunen and Joutsensalo, 1994] Karhunen, J. and Joutsensalo, J. (1994). Representation and separation of signals using nonlinear PCA type learning. *Neural Networks*, 7(1):113–127.

[Katayama and Satoh, 1997] Katayama, N. and Satoh, S. (1997). The SR-tree: an index structure for high-dimensional nearest neighbor queries. In *International Conference on Management of Data*, pages 369–380.

[Kim and Lee, 2002] Kim, H.-J. and Lee, S.-G. (2002). An effective document clustering method using user-adaptable distance metrics. In *Symposium on Applied Computing*, pages 16–20.

[Kohavi and John, 1997] Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324.

[Korn et al., 1996] Korn, F., Sidiropoulos, N., Faloutsos, C., Siegel, E., and Protopapas, Z. (1996). Fast nearest neighbor search in medical image databases. In *International Conference on Very Large Data Bases*, pages 215–226.

[Kossmann et al., 2002] Kossmann, D., Ramsak, F., and Rost, S. (2002). Shooting stars in the sky: an online algorithm for skyline queries. In *International Conference on Very Large Data Bases*, pages 275–286.

[Kriegel et al., 2007] Kriegel, H.-P., Kröger, P., Kunath, P., and Renz, M. (2007). Generalizing the optimality of multi-step k-nearest neighbor query processing. In *Symposium on Spatial and Temporal Databases*, pages 75–92.

[Kriegel et al., 2006] Kriegel, H.-P., Kröger, P., Schubert, M., and Zhu, Z. (2006). Efficient query processing in arbitrary subspaces using vector approximations. In *International Conference on Scientific and Statistical Database Management*, pages 184–190.

[Kriegel et al., 2012] Kriegel, H.-P., Kröger, P., and Zimek, A. (2012). Subspace clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(4):351–364.

[Kumar et al., 2003] Kumar, P., Mitchell, J. S. B., and Yildirim, E. A. (2003). Approximate minimum enclosing balls in high dimensions using core-sets. *ACM Journal of Experimental Algorithmics*, 8.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[Lee et al., 2007] Lee, K. C. K., Zheng, B., Li, H., and Lee, W.-C. (2007). Approaching the skyline in Z order. In *International Conference on Very Large Data Bases*, pages 279–290.

[Li et al., 2011a] Li, F., Yao, B., and Kumar, P. (2011a). Group enclosing queries. *Transactions on Knowledge and Data Engineering*, 23(10):1526–1540.

[Li et al., 2011b] Li, Y., Li, F., Yi, K., Yao, B., and Wang, M. (2011b). Flexible aggregate similarity search. In *International Conference on Management of Data*, pages 1009–1020.

[Lian and Chen, 2008] Lian, X. and Chen, L. (2008). Similarity search in arbitrary subspaces under L_p-norm. In *International Conference on Data Engineering*, pages 317–326.

[Long and Suel, 2003] Long, X. and Suel, T. (2003). Optimized query execution in large search engines with global page ordering. In *International Conference on Very Large Data Bases*, pages 129–140.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

[Natsev et al., 2001] Natsev, A., Chang, Y.-C., Smith, J. R., Li, C.-S., and Vitter, J. S. (2001). Supporting incremental join queries on ranked inputs. In *International Conference on Very Large Data Bases*, pages 281–290.

[Nepal and Ramakrishna, 1999] Nepal, S. and Ramakrishna, M. V. (1999). Query processing issues in image (multimedia) databases. In *International Conference on Data Engineering*, pages 22–29.

[Papadias et al., 2004] Papadias, D., Shen, Q., Tao, Y., and Mouratidis, K. (2004). Group nearest neighbor queries. In *International Conference on Data Engineering*, pages 301–312.

[Papadias et al., 2005a] Papadias, D., Tao, Y., Fu, G., and Seeger, B. (2005a). Progressive skyline computation in database systems. *Transactions on Database Systems*, 30(1):41–82.

[Papadias et al., 2005b] Papadias, D., Tao, Y., Mouratidis, K., and Hui, C. K. (2005b). Aggregate nearest neighbor queries in spatial databases. *Transactions on Database Systems*, 30(2):529–576.

[Persin et al., 1996] Persin, M., Zobel, J., and Sacks-davis, R. (1996). Filtered document retrieval with frequency-sorted indexes. *Journal of the American Society for Information Science*, 47(10):749–764.

[Pestov, 2000] Pestov, V. (2000). On the geometry of similarity search: dimensionality curse and concentration of measure. *Information Processing Letters*, 73(12):47–51.

[Pettis et al., 1979] Pettis, K. W., Bailey, T. A., Jain, A. K., and Dubes, R. C. (1979). An intrinsic dimensionality estimator from near-neighbor information. *Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(1):25–37.

[Razente et al., 2008a] Razente, H. L., Barioni, M. C. N., Traina, A. J. M., Faloutsos, C., and Traina, Jr., C. (2008a). A novel optimization approach to efficiently process aggregate similarity queries in metric access methods. In *International Conference on Information and Knowledge Management*, pages 193–202.

[Razente et al., 2008b] Razente, H. L., Barioni, M. C. N., Traina, A. J. M., and Traina, Jr., C. (2008b). Aggregate similarity queries in relevance feedback methods for content-based image retrieval. In *Symposium on Applied Computing*, pages 869–874.

[Roweis and Saul, 2000] Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

[Rubner et al., 1998] Rubner, Y., Tomasi, C., and Guibas, L. J. (1998). A metric for distributions with applications to image databases. In *International Conference on Computer Vision*, pages 59–66.

[Sakurai et al., 2002] Sakurai, Y., Yoshikawa, M., Uemura, S., and Kojima, H. (2002). Spatial indexing of high-dimensional data based on relative approximation. *The VLDB Journal*, 11(2):93–108.

[Samet, 2006] Samet, H. (2006). *Foundations of multidimensional and metric data structures*. San Francisco, CA, USA: Morgan Kaufmann.

[Sarwar et al., 2000] Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. T. (2000). Application of dimensionality reduction in recommender system – a case study. In *ACM WebKDD Workshop*.

[Schlkopf et al., 1998] Schlkopf, B., Smola, A., Smola, E., and Mller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319.

[Seidl and Kriegel, 1997] Seidl, T. and Kriegel, H.-P. (1997). Efficient user-adaptable similarity search in large multimedia databases. In *International Conference on Very Large Data Bases*, pages 506–515.

[Seidl and Kriegel, 1998] Seidl, T. and Kriegel, H.-P. (1998). Optimal multi-step k-nearest neighbor search. In *International Conference on Management of Data*, pages 154–165.

[Sellis et al., 1987] Sellis, T. K., Roussopoulos, N., and Faloutsos, C. (1987). The R^{+}-tree: a dynamic index for multi-dimensional objects. In *International Conference on Very Large Data Bases*, pages 507–518.

[Tan et al., 2001] Tan, K.-L., Eng, P.-K., and Ooi, B. C. (2001). Efficient progressive skyline computation. In *International Conference on Very Large Data Bases*, pages 301–310.

[Tao et al., 2007] Tao, Y., Papadias, D., Hristidis, V., and Papakonstantinou, Y. (2007). Branch-and-bound processing of ranked queries. *Information Systems*, 32:424–445.

[Theobald et al., 2004] Theobald, M., Weikum, G., and Schenkel, R. (2004). Top-k query evaluation with probabilistic guarantees. In *International Conference on Very Large Data Bases*, pages 648–659.

[Tsaparas et al., 2003] Tsaparas, P., Koudas, N., Kotidis, Y., Palpanas, T., and Srivastava, D. (2003). Ranked join indices. In *International Conference on Data Engineering*, pages 277–288.

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

[Venna and Kaski, 2006] Venna, J. and Kaski, S. (2006). Local multidimensional scaling. *Neural Networks*, 19(67):889–899.

[Verveer and Duin, 1995] Verveer, P. and Duin, R. (1995). An evaluation of intrinsic dimensionality estimators. *Transactions on Pattern Analysis and Machine Intelligence*, 17(1):81–86.

[Vlachos et al., 2006] Vlachos, M., Hadjieleftheriou, M., Gunopulos, D., and Keogh, E. (2006). Indexing multidimensional time-series. *The VLDB Journal*, 15:1–20.

[Volnyansky and Pestov, 2009] Volnyansky, I. and Pestov, V. (2009). Curse of dimensionality in pivot based indexes. In *International Workshop on Similarity Search and Applications*, pages 39–46.

[Weber et al., 1998] Weber, R., Schek, H.-J., and Blott, S. (1998). A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *International Conference on Very Large Data Bases*, pages 194–205.

[White and Jain, 1996] White, D. A. and Jain, R. (1996). Similarity indexing with the SS-tree. In *International Conference on Data Engineering*, pages 516–523.

[Wu et al., 1996] Wu, D., Agrawal, D., Abbadi, A. E., and Singh, A. (1996). Efficient retrieval for browsing large image databases. In *International Conference on Information and Knowledge Management*, pages 11–18.

[Yi et al., 2003] Yi, K., Yu, H., Yang, J., Xia, G., and Chen, Y. (2003). Efficient maintenance of materialized top-k views. In *International Conference on Data Engineering*, pages 189–200.

[Yianilos, 1993] Yianilos, P. N. (1993). Data structures and algorithms for nearest neighbor search in general metric spaces. In *Symposium on Discrete Algorithms*, pages 311–321.

[Yu et al., 2003] Yu, C., Philip, G., and Meng, W. (2003). Distributed top-N query processing with possibly uncooperative local systems. In *International Conference on Very Large Data Bases,*, pages 117–128.

[Zezula et al., 2005] Zezula, P., Amato, G., Dohnal, V., and Batko, M. (2005). *Similarity search: the metric space approach (advances in database systems)*. Secaucus, NJ, USA: Springer.

[Zhang et al., 2009] Zhang, S., Mamoulis, N., and Cheung, D. W. (2009). Scalable skyline computation using object-based space partitioning. In *International Conference on Management of Data*, pages 483–494.