Dimensional Testing for Reverse k-Nearest Neighbor Search

Guillaume Casanova
ONERA-DCSD, France
guillaume.casanova@onera.fr

Elias Englmeier
LMU Munich, Germany
elias_englmeier@yahoo.de

Michael E. Houle
NII, Tokyo, Japan
meh@nii.ac.jp

Peer Kröger
LMU Munich, Germany
kroeger@dbs.ifi.lmu.de

Michael Nett
Google Japan
mnett@google.com

Erich Schubert
Heidelberg U., Germany
schubert@informatik.uni-heidelberg.de

Arthur Zimek
SDU, Odense, Denmark
zimek@imada.sdu.dk

ABSTRACT

Given a query object q, reverse k-nearest neighbor (RkNN) search aims to locate those objects of the database that have q among their k-nearest neighbors. In this paper, we propose an approximation method for solving RkNN queries, where the pruning operations and termination tests are guided by a characterization of the intrinsic dimensionality of the data. The method can accommodate any index structure supporting incremental (forward) nearest-neighbor search for the generation and verification of candidates, while avoiding impractically-high preprocessing costs. We also provide experimental evidence that our method significantly outperforms its competitors in terms of the tradeoff between execution time and the quality of the approximation. Our approach thus addresses many of the scalability issues surrounding the use of previous methods in data mining.

1. INTRODUCTION

The reverse k-nearest neighbor (RkNN) similarity query — that is, the computation of all objects of a data set that have a given query object amongst their respective k-nearest neighbors (kNNs) — is a fundamental operation in data mining. Even though RkNN and kNN queries may seem to be equivalent at first glance, they require different algorithms and data structures for their implementation. Intuitively speaking, RkNN queries attempt to determine those data objects that are most 'influenced' by the query object. Such notions of influence arise in many important applications. In graph mining, the degree of hubness of a node [46] can be computed by means of RkNN queries. RkNN queries are also crucial for many existing data mining models, particularly in the areas of clustering and outlier detection [18, 27, 37]. For dynamic scenarios such as data warehouses and data streams, RkNN queries serve as a basic operation for determining those objects that would potentially be affected by a particular data update operation, particularly for those applications that aim to track the changes of clusters and outliers [1, 36, 35]. Bichromatic RkNN queries have also been proposed, in which the data objects are divided into two types, where queries on the objects of the first type are to return reverse neighbors drawn from the set of objects of the second type [29, 48, 50]. Typical applications of bichromatic queries arise in situations where one object type represents services, and the other represents clients. The methods suggested in [49] and [9] provide ways of computing monochromatic or bichromatic RkNN queries for 2-dimensional location data.

Virtually all existing RkNN query strategies aggregate information from the kNN sets (the 'forward' neighborhoods) of data objects in order to determine and verify RkNN candidate objects. Some strategies require the computation of exact or approximate kNN distances for all data objects [29, 51, 3, 44, 2, 4], whereas others apply distance-based pruning techniques with index structures such as R-trees [17] for Euclidean data, or M-trees [10] for general metric data [41, 33, 40, 43]. Although they are required for the determination of exact RkNN query results, the use of exact kNN sets generally involves significant overheads for precomputation and storage, especially if the value of k is not known beforehand.

In recent years, data sets have grown considerably in complexity, with the feature spaces often being of high dimensionality. For the many data mining applications that depend on indices for similarity search, overall performance can be sharply limited by the *curse of dimensionality*. For reverse neighborhood search, the impact of the curse of dimensionality is felt in the high cost of computing the forward kNN distances needed for the generation and verification of RkNN candidates. Most practical RkNN approaches work with approximations to the true kNN distance values. The quality of the approximate RkNN results, and thus the quality of the method as a whole, naturally depends heavily on the accuracy of the kNN distance approximation.

Theoreticians and practitioners have put much effort into finding workarounds for the difficulties caused by the curse of dimensionality. One approach to the problem has been to

Proceedings of the VLDB Endowment, Vol. 10, No. 7

characterize the difficulty of datasets not in terms of the representational dimension of the data, but instead in terms of a more compact model of the data. Such models have often (explicitly or implicitly) relied on such parameters dimension of the surface or manifold that best approximates the data, or of the minimum number of latent variables needed to represent the data. The numbers of such parameters can be regarded as an indication of the intrinsic dimensionality (ID) of the data set, which is typically much lower than the representational dimension. Intrinsic dimensionality thus serves as an important natural measure of the complexity of data. For a general reference on intrinsic dimensional measures, see for example [13].

Recently, intrinsic dimensionality has been considered in the design and analysis of similarity search applications, in the form of the expansion dimension of Karger and Ruhl [28] and its variant, the generalized expansion dimension (GED) [22]. The GED is a measure of dimensionality based on the rate of growth in the cardinality of a neighborhood as its radius increases — it can be regarded as an estimator of the extreme-value-theoretic local intrinsic dimension (LID) of distance distributions [20, 5], which assesses the rate of growth of the probability measure ('intrinsic volume') captured by a ball with expanding radius centered at a location of interest. LID is in turn related to certain measures of the overall intrinsic dimensionality of data sets, including the classical Hausdorff dimension [34], correlation dimension [16, 42], and other forms of fractal dimension [13]. Originally formulated solely for the analysis of similarity search indices [6, 28], the expansion dimension and its variants have recently been used in the analysis of the performance of a projection-based heuristic for outlier detection [12], and of approximation algorithms for multistep similarity search with lower-bounding distances [23, 24, 25]. The latter methods rely on a runtime test condition for deciding the early termination of an expanding ring search, based on estimation of the intrinsic dimensionality of the search neighborhood.

In this paper, we propose a filter-refinement algorithm in which run-time tests of intrinsic dimensionality can be used to significantly speed-up RkNN queries on high-dimensional data sets. The algorithm performs an expanding search from the query object, in which tests of the local intrinsic dimensionality are used to determine whether as-yet-unvisited objects could possibly be reverse k-nearest neighbors of the query. Compared to existing state-of-the-art approaches, our algorithm has the following advantages: (i) it does not rely on tree-like index structures that typically work well only in spaces of moderate dimensionality; (ii) in addition to early pruning of true negatives, our method is capable of identifying true positives early; (iii) the method is able to make effective use of approximate neighbor rankings, and thus can be supported by recent efficient similarity search methods such as [26, 15], even in high dimensional settings; (iv) the method admits a theoretical analysis of performance, including conditions for which the accuracy of the query result can be guaranteed; (v) an experimental comparison against state-of-the-art methods shows a consistent and significant improvement over all approaches tested, particularly as the data dimensionality rises.

The remainder of the paper is organized as follows. Section 2 surveys the research literature related to reverse k-nearest neighbor search. Section 3 provides background on intrinsic dimensionality, as well as supporting notation and terminology. The algorithm is presented in Section 4, followed by its analysis in Section 5. Section 7 describes the experimental framework used for the comparison of methods, and the results of the experimentation are discussed in Section 8. The presentation concludes in Section 9.

2. RELATED WORK

2.1 Methods with Heavy Precomputation

The basic observation motivating all RkNN methods is that if the distance of an object v to the query object q is smaller than the actual kNN distance of v, then v is a RkNN of q. This relationship between RkNN queries and the kNN distances of data objects was first formalized in [29]. The authors proposed a solution (originally for the special case $k=1$) in which each database object is transformed into the smallest possible hypersphere covering the k-neighborhood of the object. For this purpose, the (exact) kNN distance of each object must be precomputed. By indexing the obtained collection of hyperspheres in an R-Tree [17], the authors reduced the RkNN problem to that of answering point containment queries for hyperspheres. The authors of the *RdNN-Tree* [51] extend this idea: at each index node, the maximum of the kNN distances of the points (hypersphere radii) is aggregated within the subtree rooted at this node. In this way, smaller bounding rectangles can be maintained at the R-Tree nodes, with the potential of higher pruning power when processing queries.

An obvious limitation of the aforementioned solutions is that an independent R-Tree would be required for each possible value of k — which is generally infeasible in practice. Approaches such as [3, 44, 2, 4] attempt to overcome this limitation by allowing approximate kNN distances (for any choice of k up to a predefined maximum). Exact RkNN results can still be achieved if the approximate kNN distances are guaranteed to be lower bounds of the exact distances. If so, the method returns a superset of the exact query result as candidates; the query result can then be isolated from the candidate set in a subsequent refinement phase. However, determining the final exact result requires a kNN query to be performed for each candidate against the entire database. Most methods for RkNN search use such lower bounding approximations.

Of these methods, the *MRkNNCoP* algorithm [3] is of special interest to us, as it is the only RkNN method presented to date that implicitly uses estimation of intrinsic dimensionality to improve the pruning power of the index. *MRkNNCoP* uses a progressive refinement of upper and lower bounds to kNN distances to the query object so as to identify database objects that can be safely discarded. The pruning strategy relies on the assumption that the kNN distances for query $\mathbf{q}$ fit a formula for the fractal dimension FD involving the neighborhood size k: more precisely, the kNN distance $d_k(\mathbf{q})$ is determined according to a relationship of the form $\mathrm{FD} = c \log k / \log d_k(\mathbf{q})$ for some constant c.

All these solutions have difficulties in handling high dimensional data. As the representational dimension of the feature space grows, the R-Tree loses much of its pruning power [47]. As a consequence, in such scenarios, the set of candidates generated before refinement is typically very large, and the costs of computing the set increase considerably. In the case of approximate RkNN search, the quality of

the query result decreases significantly; in the case of exact search, the additional refinement phase is extremely costly.

2.2 Dynamic Methods

In order to avoid the inflexibility and the heavy cost associated with the precomputation of distance information, a second class of methods for RkNN query processing explore the neighborhoods of candidate objects at query time. As a consequence, these methods are typically more flexible with respect to the choice of k, and are more suitable for dynamic scenarios in which the data changes frequently. This flexibility is typically achieved through the use of complex hyperplane arrangements during index traversal to quickly eliminate search paths from consideration [41, 33, 40, 43]. The solutions in [41], [49] and [9] are designed for 2D data.

Singh et al. [40] attempt to alleviate the execution cost of RkNN queries by abandoning exactness in *SFT*. Query processing begins with the extraction of an αk-NN set (for $\alpha \geq 1$) of the query point as an initial set of candidates. The algorithm subsequently employs two refinement strategies for the removal of false positives from the candidate set: the outcome of *local distance computations* among pairs of candidate points is first used for filtering, and the remaining false positives are then eliminated using *count range queries*. This method naturally offers support for flexible choices of k. Generally speaking, the proportion of query answers found by this approach is subject to the particular choice of α; if the number of initial candidates is sufficiently large, then the heuristic is guaranteed to retrieve all RkNNs of a given query point. However, other than that the forward rank of a RkNN grows at most exponentially with the representational dimension of the feature space, the relationship between α and the recall rate is not well understood.

As with previously discussed methods, *TPL* [43] relies on an index supporting kNN queries. The generation and refinement of candidates is handled within a single traversal of the underlying index structure. *TPL* discards candidates and search paths using arrangements of perpendicular bisectors between the query point and candidate objects. However, the performance of the pruning procedure rapidly diminishes as either the neighborhood rank k or the data dimensionality grows. Extensions of the *TPL* method include an incremental method with pruning based on minimum and maximum distances to bounding rectangles [30].

In general, both precomputation-heavy methods and dynamic methods also suffer from their explicit use of index structures (such as the R-Tree family) that are effective only for data of low to moderate dimensionality.

3. PRELIMINARIES

3.1 Notation

Let $(\mathbb{R}^m, d)$ refer to the Euclidean space of representational dimension $m \in \mathbb{N}$ ($m > 0$). We denote points in $\mathbb{R}^m$ using bold lower-case letters. For the sake of convenience, we identify data objects with their associated feature vectors in $\mathbb{R}^m$. Given a pair of points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$, recall the definition of the Euclidean distance:

$$d(\mathbf{x},\mathbf{y}) \triangleq \left(\sum_{i=1}^{m} |x_i - y_i|^2 \right)^{\frac{1}{2}}$$

Let $S \subseteq \mathbb{R}^m$ be a finite point set containing $n > 0$ elements. For any reference point $\mathbf{q} \in \mathbb{R}^m$ and positive distance threshold $r > 0$, let

$$B_S^{\leqslant}(\mathbf{q}, r) \triangleq \{\, \mathbf{x} \in S : d(\mathbf{q},\mathbf{x}) \leq r \,\}$$

refer to the subset of points from S contained in the *closed ball* of radius r centered at $\mathbf{q}$. Accordingly, we define the *rank* of an object $\mathbf{x} \in S$ with respect to $\mathbf{q}$ as the number of points in the smallest ball that both contains $\mathbf{x}$ and is centered at $\mathbf{q}$:

$$\rho_S(\mathbf{q},\mathbf{x}) \triangleq \left| B_S^{\leqslant}(\mathbf{q}, d(\mathbf{q},\mathbf{x})) \right|.$$

Note that when distances are tied, this definition assigns the maximum rank to each of the objects. We use the notation $d_k(\mathbf{q})$ to refer to the distance from $\mathbf{q}$ to its k-nearest neighbor. For any reference point $\mathbf{q} \in \mathbb{R}^m$ and any *neighborhood size* $k \in \mathbb{N}$, the set of kNNs is given by

$$\begin{aligned} N_S(\mathbf{q},k) &\triangleq \{\, \mathbf{x} \in S : \rho_S(\mathbf{q},\mathbf{x}) \leq k \,\} \\ &= \{\, \mathbf{x} \in S : d(\mathbf{q},\mathbf{x}) \leq d_k(\mathbf{q}) \,\}. \end{aligned}$$

The set of RkNNs, the efficient construction of which is central to the theme of this paper, can be defined as

$$\begin{aligned} N_S^{-1}(\mathbf{q},k) &\triangleq \{\, \mathbf{x} \in S : \rho_S(\mathbf{x},\mathbf{q}) \leq k \,\} \\ &= \{\, \mathbf{x} \in S : q \in N_S(\mathbf{x},k) \,\}. \end{aligned}$$

3.2 Generalized Expansion Dimension

Karger and Ruhl [28] proposed a measure of intrinsic dimensionality supporting the analysis of a similarity search strategy. The execution cost of their method depends heavily on the rate at which the number of discovered data objects grows as the search progresses. Accordingly, the authors restricted their analysis to point configurations satisfying the following *smooth growth property*.

Definition 1 (Smooth Growth Property [28]). *Let S be a set of objects in a space equipped with a distance measure d. S is said to have (b,δ)-expansion if it satisfies*

$$\left| B_S^{\leqslant}(\mathbf{q},r) \right| \geq b \Longrightarrow \left| B_S^{\leqslant}(\mathbf{q},2r) \right| \leq \delta \cdot \left| B_S^{\leqslant}(\mathbf{q},r) \right|,$$

for any reference point $\mathbf{q} \in \mathbb{R}^m$ and any distance threshold $r > 0$, subject to the choice of minimum ball size $b \in \mathbb{N}$.

The *expansion rate* of S is the minimum value of δ, such that S has (b,δ)-expansion. Karger and Ruhl [28] chose a minimum ball size of $b \in \mathbf{O}(\log n)$.

The expansion rate was introduced in order to support the analysis of the complexity of search indices that rely on the triangle inequality of Euclidean space; the upper-bounding arguments used require that the outer sphere have double the radius of the inner sphere [28, 6]. More generally, however, expansion can be revealed by any two concentric ball placements of distinct, strictly positive radii.

In its more general form, the expansion rate can be related to the dimensionality of the underlying space. To motivate this, we consider the situation in Euclidean space, as follows. Let $B_S^{\leqslant}(\mathbf{q},r_1)$ and $B_S^{\leqslant}(\mathbf{q},r_2)$ be two spheres centered at $\mathbf{q} \in \mathbb{R}^m$ with $0 < r_1 < r_2$. The volumes of these balls are

$$V_i \triangleq \int_{B_S^{\leqslant}(\mathbf{q},r_i)} \mathrm{d}\mathbf{x} = \frac{\pi^{m/2}}{\Gamma\left(\frac{m}{2}+1\right)} r_i^m \quad \text{for } i \in \{1,2\}.$$

Solving for the dimension gives

$$m = \frac{\log(V_2/V_1)}{\log(r_2/r_1)}. \tag{1}$$

Equation 1 is not in itself immediately useful, as the representational dimension m of a feature space is almost always

known. However, under a distributional point of view in which the dataset S is regarded as having been produced by a statistical generation process, the formulation of Equation 1 leads to a measure of the *intrinsic dimensionality* (ID) of S. The ratio of the volumes V_2/V_1 of the neighborhood balls $B_S^{\leq}(\mathbf{q}, r_i)$ can be substituted by the ratio of the probability measures associated with the two balls. Under this substitution, with appropriate assumptions on the continuity of the distribution of distances with respect to $\mathbf{q}$, as $r_2 \to r_1 \to 0$, m tends to the *local intrinsic dimensionality* (LID) [20, 5], which can be defined as follows:

$$\text{LID} \triangleq \lim_{r \to 0^+} \lim_{\epsilon \to 0^+} \frac{\log\left(F\left((1+\epsilon)r\right)/F(r)\right)}{\log(1+\epsilon)},$$

where $F(r)$ represents the probability measure associated with the ball centered at $\mathbf{q}$ with radius r. Local intrinsic dimensionality has been shown to have an interpretation in terms of extreme value theory, as the LID value coincides with the scale parameter describing the rate of growth in the lower tail of the distance distribution [21, 5]. This relationship holds true for any distance measure (not just Euclidean) for which the continuity assumptions are satisfied.

The ratio of the probabilities associated with the neighborhood balls $B_S^{\leq}(\mathbf{q}, r_i)$ is equal to the ratio of the expected number of points of the data samples contained in these balls. For sufficiently small values of r_1 and r_2, an estimate of the LID can thus be obtained by substituting V_1 and V_2 by the numbers of points of S contained in the respective balls. Denoting these numbers of points by k_1 and k_2, respectively, we arrive at an estimator of the LID referred to as the *generalized expansion dimension* of S [22, 5]:

$$\textsc{Ged}\left(B_S^{\leq}(\mathbf{q}, r_1), B_S^{\leq}(\mathbf{q}, r_2)\right) \triangleq \frac{\log(k_2/k_1)}{\log(r_2/r_1)}.$$

For the special case of $r_2 = 2r_1$, we have the *expansion dimension* as originally proposed by Karger and Ruhl [28]. Note that as the GED can be regarded as an estimator of LID, its use is not restricted to the Euclidean distance.

The choices of neighborhood balls $B_S^{\leq}(\mathbf{q}, r_1)$ and $B_S^{\leq}(\mathbf{q}, r_2)$ determine two distance-rank pairs, (r_1, k_1) and (r_2, k_2), that together allow an assessment of the dimensional structure of the data points in the vicinity of $\mathbf{q}$. In general, we consider $\textsc{Ged}\left(B_S^{\leq}(\mathbf{q}, r_1), B_S^{\leq}(\mathbf{q}, r_2)\right)$ to be a *dimensional test value* as determined by $B_S^{\leq}(\mathbf{q}, r_1)$ and $B_S^{\leq}(\mathbf{q}, r_2)$. Varying the neighborhood balls is likely to impact the value of a dimensional test; however, in [22] the authors discuss how an aggregation of the results of multiple tests over a range of ball sizes can produce a representative estimate of the intrinsic dimension in the vicinity of individual query locations.

The analysis presented in Section 5 of this paper makes use of an upper bound on estimates of the local intrinsic dimensionality generated throughout the search. Accordingly, for a set of points S and a target neighborhood size $k \in \mathbb{N}$, we define the *maximum generalized expansion dimension* (maximum GED, or MaxGed) as

$$\textsc{MaxGed}(S, k) \triangleq \max_{\substack{\mathbf{q} \in S, \\ k < s \leq |S| \\ d_k(\mathbf{q}) \neq d_s(\mathbf{q})}} \left\{ \textsc{Ged}\left(B_S^{\leq}(\mathbf{q}, d_s(\mathbf{q})), B_S^{\leq}(\mathbf{q}, d_k(\mathbf{q}))\right) \right\}$$

Note that this definition differs slightly from the one proposed in [23] for lower-bounding distance search.

Previously, models of intrinsic dimension, such as *expansion rate* [28] and *aspect ratio* [11], have typically been used to support *a posteriori* analyses of the performance of different similarity search structures. In the following section, we demonstrate how outcomes of tests of the generalized expansion dimension can be used to guide algorithmic decisions in processing reverse k-nearest neighbor queries.

Algorithm 1 Our algorithm for processing reverse k-nearest neighbor queries centered at q, with respect to a data set S. The algorithm requires an index $\mathcal{I}$ supporting (incremental) forward nearest-neighbor queries. It also accepts a non-negative scale parameter t governing the trade-off between time and accuracy. Shading is used to indicate those operations involving witness sets.

```
 1: procedure RDT(q, k, t, I)
 2:     Initialize the upper bound ω at ∞.
 3:     Initialize an empty filter set F and result set R.
 4:     Initialize the neighborhood size s at 0.
 5:     repeat
 6:         Using the supplied index I, retrieve the next
            neighbor v ∈ S of q.
 7:         Let s ← ρ_S(q, v).  //Lemma 1.
 8:         Initialize the witness counter W(v) ← 0.
 9:         for x ∈ F do
10:             if d(q, x) > d(v, x) then
11:                 Increment the witness counter W(v).
12:             end if
13:             if d(q, v) > d(v, x) then
14:                 Increment the witness counter W(x).
15:             end if
16:             if W(x) < k and d(q, v) ≥ 2d(q, x) then
17:                 Insert x into R.
18:             end if
19:         end for
20:         Insert v into the filter set F.
21:         if s > k and d(q, v) > 0 then  //Theorem 1.
22:             Update ω ← min{ω, d(q,v) / (s/k)^{1/t} − 1}.
23:         end if
24:     until d(q, v) > ω or s ≥ min{n, ⌊2^t k⌋}
25:     for v ∈ F \ R do
26:         if W(v) < k then
27:             if d_k(v) ≥ d(q, v) then
28:                 Insert v into the result set R.
29:             end if
30:         end if
31:     end for
32:     Return R.
33: end procedure
```

4. ALGORITHM

This section is concerned with the main contribution of the article, a scalable heuristic for answering **R**everse k-nearest neighbor queries by **D**imensional **T**esting (*RDT*). More specifically, we propose an algorithm that, given a data set $S \subseteq \mathbb{R}^m$, a query object $\mathbf{q} \in S$, and a target neighbor rank $k \in \mathbb{N}$, returns an approximation to the set of reverse k-nearest neighbors $N_S^{-1}(\mathbf{q}, k)$.

We designed our algorithm to easily integrate into any existing framework that supports (forward) k-nearest neighbor queries. The algorithm requires only that it be provided with some auxiliary index structure that can efficiently process incremental nearest neighbor queries with respect to S. Assuming these requirements are fulfilled, our algorithm does not impose any significant additional storage cost in the processing of RkNN queries, and any computational

overheads incurred are independent of the data set size n. Furthermore, our method supports dynamic insertion and deletion of data points, in that no additional costs are incurred other than those due to changes made to the auxiliary "forward kNN" index structure.

The algorithm, a formal description of which is provided as Algorithm 1, aims to locate reverse k-nearest neighbors in S using a filter-refinement approach. In order to facilitate processing of RkNN queries, the algorithm also accepts a *scale parameter* $t > 0$ controlling the tradeoff between approximation quality and execution cost: increasing the value of t will eventually lower the number of false negatives to zero, at the expense of additional computation. In Algorithm 1, processing of a query located at $\mathbf{q} \in \mathbb{R}^m$ begins with a filtering step (lines 2–24) that discovers a set F of candidate points from S. The subsequent refinement phase of the algorithm (lines 25–32) eliminates false positives from F and returns the remainder as the query result.

4.1 Filter Phase

The discovery of potential reverse k-nearest neighbors in the filter phase is facilitated by means of a search expanding outwards from the query $\mathbf{q}$. The expansion process is eventually interrupted by a termination condition, the *dimensional test*, which uses distance and rank information within the neighborhood of q in order to judge whether any other reverse k-nearest neighbor objects remain to be discovered (line 24). The test makes use of distance and rank information within the neighborhood, as well as a user-supplied estimate of the maximum local intrinsic dimension in the form of the scale parameter t. The test uses this information to deduce whether an undiscovered reverse k-nearest neighbor could exist without violating the heuristic assumption that t is an upper bound on the intrinsic dimension; if the test succeeds, the algorithm deems that no undiscovered reverse neighbors exist, and the execution terminates. Larger choices of t tend to increase the quality of the query result at the expense of execution time. The dimensional test implicitly assesses the order of magnitude of the neighborhood growth rate, and not the local density. Dimensional testing thus allows the algorithm to dynamically adapt its behavior to data regions of varying density. A rigorous analysis of the termination criterion is provided in Section 5.

By expanding on a technique used in [40], our algorithm avoids unnecessary computation of forward kNN sets by maintaining partial lists of the neighborhoods for each of the candidate points. Let $\mathbf{x}, \mathbf{y} \in F$ be two distinct points discovered in the filter phase. We refer to $\mathbf{y}$ as a *witness* of $\mathbf{x}$ if $d(\mathbf{y}, \mathbf{x}) < d(\mathbf{q}, \mathbf{x})$. In particular, for each candidate point $\mathbf{x} \in F$, our algorithm tracks the number of witnesses of $\mathbf{x}$, defined as $W(\mathbf{x}) \triangleq |\{\, \mathbf{y} \in F : d(\mathbf{x}, \mathbf{y}) < d(\mathbf{x}, \mathbf{q}) \,\}|$. If the number of witnesses of $\mathbf{x}$ ever reaches or exceeds k, then $\mathbf{x}$ can be eliminated as a reverse k-nearest neighbor of $\mathbf{q}$, without needing to perform an expensive verification of the forward k-nearest neighborhood of $\mathbf{x}$.

Assertion 1 (Lazy Reject). *Let F be a candidate set generated by Algorithm 1. Any point $\mathbf{x} \in F$ satisfying $W(\mathbf{x}) \geq k$ cannot be a member of $N_S^{-1}(\mathbf{q}, k)$.*

Assertion 2 (Lazy Accept). *Let $\mathbf{x} \in F$ have query distance $d(\mathbf{q}, \mathbf{x})$. If $W(\mathbf{x}) < k$, and if the expanding search progresses to a distance greater than $2d(\mathbf{q}, \mathbf{x})$, then $\mathbf{x}$ must be a member of $N_S^{-1}(\mathbf{q}, k)$.*

Assertion 1 follows from the observation that the existence of k witnesses implies that the query point $\mathbf{q}$ is outside of the k-nearest neighbor ball of $\mathbf{x}$ with respect to S. Accordingly, $\mathbf{x}$ cannot be a reverse k-nearest neighbor of $\mathbf{q}$. Conversely, if the expanding search progresses to a distance of $2d(\mathbf{q}, \mathbf{x})$ without producing at least k witnesses for $\mathbf{x}$, then a neighborhood of $\mathbf{x}$ of size $W(\mathbf{x}) + 1 \leq k$ has been completely searched, and the query point $\mathbf{q}$ would be contained in it. In other words, $\mathbf{x}$ is a reverse k-nearest neighbor of $\mathbf{q}$.

The set of witnesses of any point $\mathbf{x} \in F$ is updated whenever a new candidate enters the set F. The total cost of these updates is bounded from above by $\binom{s}{2}$, where s is the number of candidates discovered during the filter phase. Although the cost of maintenance is quadratic in the size of the candidate set F, the use of witness sets can significantly accelerate the refinement phase.

4.2 Refinement Phase

Individual members of the candidate set F discovered during the filter phase may satisfy neither of the assertions stated above. In Algorithm 1 the treatment of such points is delayed until the refinement phase (lines 25–32). For any such candidate $\mathbf{x} \in F$, the refinement phase executes a (forward) k-nearest neighbor query located at $\mathbf{x}$. A candidate $\mathbf{x} \in F$ is only retained in the result set R if it is verified to satisfy $d_k(\mathbf{x}) \geq d(\mathbf{q}, \mathbf{x})$. Generally speaking, the cost of neighborhood verification is the most expensive operation in our algorithm.

4.3 Candidate Set Reduction

In Algorithm 1, many unnecessary verification operations can be avoided through the use of witnesses. However, as the number of candidates increases, the cost of maintaining witnesses can become too expensive, particularly when the dataset size or dimensionality is large. To overcome this problem, we introduce the following heuristic variation, which we refer to as *RDT+*: in order to reduce the costs associated with witness operations, any newly-retrieved object v that accumulates k or more witnesses in its first cycle through the witness procedure (Lines 8-19 of Algorithm 1) is excluded from the filter set. More precisely, Line 20 is not executed unless $W(\mathbf{v}) < k$. The rationale is that objects that have been excluded as potential reverse k-nearest neighbors are themselves unlikely to be decisive witnesses for the rejection of other objects. The exclusion criterion prevents a significant number of false positives from increasing the witness sets of other candidates, at the risk of a drop in precision in the query result. The criterion is not applied to the first k candidates, as it is impossible for them to immediately reach the rejection threshold of $W(x) \geq k$. It should be emphasized that even when an item is excluded as a witness by *RDT+*, it is still eligible for lazy rejection if its own witness count reaches k.

At any given stage of the search, the asymptotic execution time taken by the witness procedure of *RDT* and *RDT+* is quadratic in the size of the filter set (that is, $O(|F|^2)$). In the theoretical worst case, $|F|$ can be linear in n; however, in practice $|F|$ can be expected to be much lower.

5. ANALYSIS

We present a formal analysis that derives conditions under which Algorithm 1 is guaranteed to return an exact query

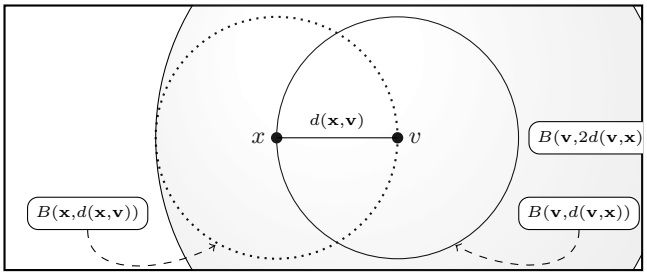

Figure 1: Visualization of the ball placements used in the proof of Lemma 1. The forward rank $\rho(\mathbf{x},\mathbf{v})$ and the reverse rank $\rho(\mathbf{v},\mathbf{x})$ are determined by the number of data points from S captured by the corresponding balls.

result. The theoretical result will be seen to imply a relationship between the scale parameter t and the intrinsic dimensionality of the data set. The analysis will be seen to hold for any distance metric; that is, any distance measure for which the triangle inequality holds. We begin the analysis by demonstrating a useful relationship between the forward and reverse ranks between any two given points, in terms of maximum generalized expansion dimension.

LEMMA 1 (REVERSE RANK BOUND). *Let S be a set of points and let $\mathbf{x},\mathbf{v}$ be a pair of distinct points from S, such that $\rho_S(\mathbf{x},\mathbf{v})=k$ for some $k\in\mathbb{N}$. Let t be an upper bound on the maximum generalized expansion dimension* $\textsc{MaxGed}(S,k)$. *The forward rank $\rho_S(\mathbf{x},\mathbf{v})$ is bounded from above by $2^t\rho_S(\mathbf{v},\mathbf{x})$.*

PROOF. In the following, we consider certain placements of neighborhood balls within the framework of generalized expansion dimension. An illustration of the specific ball placements relevant to the proof is given in Figure 1.

Under the metric distance assumption, given $\mathbf{x},\mathbf{v}\in S$, we note that $B_S^{\leq}(\mathbf{v},2d(\mathbf{v},\mathbf{x}))$ is the smallest ball centered at $\mathbf{v}$ that completely covers the ball $B_S^{\leq}(\mathbf{x},d(\mathbf{x},\mathbf{v}))$. By assumption the maximum generalized expansion dimension of S is no greater than t. This implies that

$$\begin{aligned} t &\geq \textsc{Ged}\big(B_S^{\leq}(\mathbf{v},2d(\mathbf{v},\mathbf{x})),B_S^{\leq}(\mathbf{v},d(\mathbf{v},\mathbf{x}))\big) \\ &= \frac{\log\big|B_S^{\leq}(\mathbf{v},2d(\mathbf{v},\mathbf{x}))\big|-\log\big|B_S^{\leq}(\mathbf{v},d(\mathbf{v},\mathbf{x}))\big|}{\log(2d(\mathbf{v},\mathbf{x}))-\log d(\mathbf{v},\mathbf{x})} \\ &= \frac{\log\big|B_S^{\leq}(\mathbf{v},2d(\mathbf{v},\mathbf{x}))\big|-\log\rho_S(\mathbf{v},\mathbf{x})}{\log 2}. \end{aligned}$$

Rearranging the above inequality, we obtain

$$t\log 2\geq\log\frac{\big|B_S^{\leq}(\mathbf{v},2d(\mathbf{x},\mathbf{v}))\big|}{\rho_S(\mathbf{v},\mathbf{x})}. \tag{2}$$

Recall that by construction $B_S^{\leq}(\mathbf{v},2d(\mathbf{v},\mathbf{x}))$ completely encloses $B_S^{\leq}(\mathbf{x},d(\mathbf{x},\mathbf{v}))$. Therefore, the number of points captured by the enclosing ball satisfies

$$\big|B_S^{\leq}(\mathbf{v},2d(\mathbf{v},\mathbf{x}))\big|\;\geq\;\big|B_S^{\leq}(\mathbf{x},d(\mathbf{x},\mathbf{v}))\big|=\rho_S(\mathbf{x},\mathbf{v}).$$

Accordingly, substituting the above into Equation 2 gives

$$t\log 2\geq\log\frac{\rho_S(\mathbf{x},\mathbf{v})}{\rho_S(\mathbf{v},\mathbf{x})}.$$

We conclude the proof of Lemma 1 by exponentiating and solving the above inequality for $\rho_S(\mathbf{x},\mathbf{v})$:

$$\rho_S(\mathbf{x},\mathbf{v})\leq 2^t\cdot\rho_S(\mathbf{v},\mathbf{x}). \quad \square$$

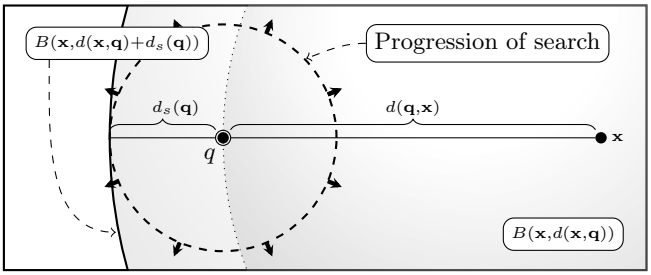

Figure 2: Visualization of the ball placements used in the early termination criterion applied in Algorithm 1. The situation illustrates a reverse k-nearest neighbor query located at q in which the expanding search has progressed to a distance of $d_s(q)$. The object $x\in S$ is a reverse neighbor that has yet to be discovered by the search.

The remainder of this section is concerned with an analysis of the quality of the query results returned by Algorithm 1. In particular, we show that the algorithm will report every reverse k-nearest neighbor whose distance to query q falls below a threshold that depends on the termination parameter t, the neighbor rank k, and the $(k+1)$-nearest neighbor distance to q; by increasing the value of t, the minimum RkNN threshold distance also increases. Given a choice of t, a guarantee on the quality of the result can therefore be determined in advance of the query simply by calculating the $(k+1)$-nearest neighbor distance to q. In addition, if t is chosen to be at least as large as the maximum generalized expansion dimension $\textsc{MaxGed}(S\cup\{\mathbf{q}\},k)$, the accuracy of the entire query result is guaranteed. The analysis is formalized in the following theorem.

THEOREM 1. *Let $\mathbf{q}\in\mathbb{R}^m$ be a query and let $k\in\mathbb{N}$. Let $R\subseteq S$ denote the set of query answers returned by Algorithm 1. Any reverse k-nearest neighbor $x\in N_S^{-1}(\mathbf{q},k)$ not contained in R must have distance to q satisfying*

$$d(\mathbf{x},\mathbf{q})>\frac{d_{k+1}(\mathbf{q})}{\left(\frac{s}{k}\right)^{\frac{1}{t}}-1}\geq\frac{d_{k+1}(\mathbf{q})}{\left(\frac{n}{k}\right)^{\frac{1}{t}}-1},$$

where $s=|F|\in\mathbb{N}$ is the number of data objects discovered before the expanding search in Algorithm 1 terminates. Moreover, if the scale parameter t is no less than $\textsc{MaxGed}(S\cup\{\mathbf{q}\},k)$, *then $R=N_S^{-1}(\mathbf{q},k)$.*

PROOF. We prove Theorem 1 by analyzing the individual situations that can result in the termination of Algorithm 1. From the loop invariant in line 24, either $s\geq\min\{n,2^tk\}$ or $d_s(q)>\omega$ must hold at termination.

Case 1. ($s=\min\{n,2^tk\}$ at termination.)
If the minimum is determined by n, the expanding search has explored the whole data set, and therefore F must contain all reverse k-nearest neighbors. If instead $s=\lfloor 2^tk\rfloor$, we use Lemma 1 to conclude that the (forward) rank $\rho_S(\mathbf{q},\mathbf{x})$ of any reverse k-nearest neighbor $\mathbf{x}\in S$ cannot exceed s. Accordingly, independently of how the minimum is determined, all reverse k-nearest neighbors must be contained in the candidate set F, and therefore $N_S^{-1}(\mathbf{q},k)\subseteq F$.

Case 2. ($d_s(\mathbf{q})>\omega$ at termination.)
First, observe that the particular (finite) value of ω at termination must have been realized in some iteration $\tilde{s}$, where

$k < \tilde{s} \leq s$. Accordingly,

$$\omega = \frac{d_{\tilde{s}}(\mathbf{q})}{\left(\frac{\tilde{s}}{k}\right)^{\frac{1}{t}} - 1}.$$

Suppose now that there exists a reverse k-nearest neighbor $\mathbf{x} \in N_S^{-1}(\mathbf{q}, k)$ that had not been reached by the expanding search at the time of termination. Any such point must have

$$d(\mathbf{x}, \mathbf{q}) > \frac{d_{\tilde{s}}(\mathbf{q})}{\left(\frac{\tilde{s}}{k}\right)^{\frac{1}{t}} - 1}$$

Grouping rank and distance-related terms, this becomes

$$\left(\frac{\tilde{s}}{k}\right)^{\frac{1}{t}} > \frac{d_{\tilde{s}}(\mathbf{q}) + d(\mathbf{x}, \mathbf{q})}{d(\mathbf{x}, \mathbf{q})}.$$

By taking the logarithms and simplifying, we obtain

$$\frac{1}{t}\left(\log \tilde{s} - \log k\right) > \log\left(d_{\tilde{s}}(\mathbf{q}) + d(\mathbf{x}, \mathbf{q})\right) - \log d(\mathbf{x}, \mathbf{q}).$$

Noting that the metric assumption guarantees that the ball $B_S^{\leq}(\mathbf{x}, d_{\tilde{s}}(\mathbf{q}) + d(\mathbf{x}, \mathbf{q}))$ contains at least $\tilde{s}$ points, solving for t yields

$$\begin{aligned} t &< \frac{\log \tilde{s} - \log k}{\log\left(d_{\tilde{s}}(\mathbf{q}) + d(\mathbf{x}, \mathbf{q})\right) - \log d(\mathbf{x}, \mathbf{q})} \\ &\leq \textsc{Ged}\left(B_S^{\leq}(\mathbf{x}, d_{\tilde{s}}(\mathbf{q}) + d(\mathbf{x}, \mathbf{q})), B_S^{\leq}(\mathbf{x}, d(\mathbf{x}, \mathbf{q}))\right) \\ &< \textsc{MaxGed}(S \cup \{\mathbf{q}\}, k). \end{aligned}$$

This, however, violates the assumption that t is an upper bound on the maximum generalized expansion dimension. Therefore, no reverse k-nearest neighbor $\mathbf{x} \in N_S^{-1}(\mathbf{q}, k)$ can have been missed by the expanding search. □

6. SCALE PARAMETER ESTIMATION

From the analysis in the previous section, one may be tempted to conclude that the most effective way of choosing values for the scale parameter t in practice is by estimating MaxGed. However, estimation of MaxGed is extremely impractical due to the potentially infinite number of query locations over which the maximum GED value is to be estimated. Moreover, MaxGed is an extremely conservative and loose upper bound on the intrinsic dimensionality in the vicinity of the query. Instead, a more favorable tradeoff between accuracy and execution time may be achieved through the direct estimation of the intrinsic dimension itself, and not through an upper bound on GED estimates. For this purpose, we precompute estimates of the intrinsic dimensionality for the entire dataset using existing estimators for other expansion-based models: the maximum likelihood estimator (MLE) for the local intrinsic dimensionality (LID) [5], and two estimators of the correlation dimension (CD), the Grassberger-Procaccia algorithm [16] and the Takens estimator [45, 42]. With these estimators guiding the choice of t, the *RDT* termination criterion based on Theorem 1 is no longer a guarantee but a heuristic requiring experimental validation.

An overall measure of ID can be obtained by averaging LID values over a sufficiently large sample of points drawn from the dataset S. For this purpose, we chose the MLE (Hill) estimator described in [5], due to its relative stability and convergence properties:

$$ID_x = -\left(\frac{1}{n}\sum_{i=1}^{n} \ln \frac{x_n}{w}\right)^{-1},$$

where $x_1 ... x_n$ are observations of a random distance variable X taking values in the range $(0, w]$. The experimental results of [5] show that a neighborhood set size of $n = 100$ is generally sufficient for convergence. The runtime of the estimator scales linearly.

The Grassberger-Procaccia (GP) algorithm [16] estimates the dimension in terms of the proportion of pairwise distance values falling below a supplied small threshold $r > 0$:

$$C(r) = \frac{2}{N(N-1)} \sum_{i<j} H(r - \|x_i - x_j\|),$$

where H is the Heaviside step function that is 0 for values below 0, and 1 for values greater or equal to zero. The correlation dimension is then defined as:

$$\mathrm{CD} = \lim_{r \to 0} \frac{\log C(r)}{\log\ r}.$$

In practice, the limit is estimated by fitting a straight line to a log-log curve of $C(r)$ versus r, over the smallest values of r. The slope of this line is then the estimate of CD.

The Takens estimator of correlation dimension is given by

$$v(r) = 1 / \langle log(r_{ij}/r) \rangle,$$

where the angle brackets denote the average over all pairwise distances r_{ij} which are less than a supplied small threshold value $r > 0$. The *Takens* and *GP* estimators both compute values for all pairs of distances taken from the supplied Index $\mathcal{I}$, leading to a quadratic runtime.

7. EXPERIMENTAL FRAMEWORK

7.1 Comparison to Prior Work

The evaluation begins with an in-depth comparison of *RDT* and *RDT+* with competing methods chosen from those reviewed in Section 2. In particular, we compare against the approximate *SFT* heuristic [40], the *MRkNNCoP* algorithm [3], the *RdNN-Tree* [51], and a variant of *TPL* based on *k-trim* and a *Hilbert heuristic* [43]. For the approximation methods (*RDT*, *RDT+* and *SFT*), we generate time-accuracy tradeoff curves by executing each method over a range of choices of their respective tuning parameters. We also show the performance of *RDT+* for fixed values of the scale parameter t as determined using the three different estimators of intrinsic dimensionality described in Section 6. These results will be depicted in the plots in Section 8 as *RDT+*(MLE), *RDT+*(Takens) and *RDT+*(GP).

All methods in the experimental study were implemented in Java using the ELKI data mining framework [39].

Excluding the cost of maintaining the index supporting forward kNN queries, the asymptotic storage cost of the algorithms is dominated by the number of candidates generated throughout the execution. As explained in Section 4, the number of candidates generated depends on the input parameters t and k, and is linear in n in the worst case.

The estimators of correlation dimension were implemented by adapting the source code[1] used in [19]. To produce the final averaged estimate of local intrinsic dimensionality, *MLE* estimation was performed over a random sample of size equal to ten percent of each dataset; for each estimate, 100 nearest neighbors were used. Back-end support for incremental kNN queries can be provided in a great variety of ways. For our experimentation, we chose as examples two different methods: the *Cover Tree* [6], and straightforward sequential database scan. For all datasets except for *MNIST*

[1] `www.ml.uni-saarland.de/code/IntDim/IntDim.htm`

and *Imagenet* (described below), the *Cover Tree* was found to perform substantially better than sequential scan. Thus, for these two sets, all experimental results were reported using sequential scan, while for the remaining sets, the results reported are for the *Cover Tree*. So that our method could be tested against competitors that require it, the Euclidean distance was used for all experiments.

In the literature on RkNN search, experimentation usually targeted combinations of datasets and queries of low scale in terms of the reverse neighbor rank k, as well as the dataset size and representational dimension. We based the first group of experiments on the following collection of publicly available datasets of low to medium scale, selected so as to span a variety of potential target applications. From among the objects contained in each dataset, we extracted 100 randomly chosen points to serve as query objects for RkNN queries using $k \in \{10, 50, 100\}$.

- The *Sequoia* dataset contains a total of 62,174 locations of interest from within California[2]. Individual locations in the dataset are represented by a normalized pair of latitude and longitude values.
- The *Amsterdam Library of Object Images* (*ALOI*) [14] consists of 110,250 images of physical objects taken under varying lighting conditions. Each element is described by a vector spanning 641 features, including texture and color histograms. Details regarding the feature generation process are available in [8].
- The *Forest Cover Type* (*FCT*) dataset [7], available as part of the UCI Machine Learning Repository[2] [32], describes the topographical features of 581,012 forest cells, each 900m^2 in area. The 53 attributes contain information on the elevation, slope and vegetation cover types within the cell. We normalized each feature to standard scores.
- The *MNIST Database of Handwritten Digits* contains 70,000 recordings of hand-written digits by 500 distinct writers. The digits are normalized to a 28×28 grid, resulting in a total of 784 feature dimensions [31].

7.2 Efficacy of Lazy Acceptance and Lazy Rejection Criteria

As noted in Section 2, the execution time of individual RkNN queries is typically dominated by the cost of calculating k-nearest neighbor distances within the refinement phase of Algorithm 1. Consequently, the savings from accepting or rejecting a candidate based on Assertion 1 or 2 can potentially improve the overall execution cost significantly. In fact, the generation of a larger number of candidates during the filter phase can potentially provide more opportunities for the lazy rejection of individual candidate points; at the same time, unfortunately, the cost of maintaining the witness counts would also increase. These conflicting influences could produce considerable variations in the tradeoff between time and accuracy of the method, potentially confusing the contributions of lazy acceptance and lazy rejection. We therefore recorded the proportion of candidates affected by each criterion separately. We present our findings in Section 8.2.

[2]`http://archive.ics.uci.edu/ml`

7.3 Scalability

In this part of our experimentation, we investigate the performance of the individual methods for applications at scales larger than those targeted by prior work (see Section 2). The competing methods tested in this section are the *MRkNN-CoP* algorithm [3] and the *RdNN-Tree* [51]. We evaluate the performance of the individual methods on choices of $k \in \{10, 50\}$ and report the mean performance of a sample of 100 queries drawn uniformly at random from within the data points of the data set. The dataset used in this second batch of experiments is described as follows.

The *Imagenet* dataset [38] consists of 1,281,167 images collected from *flickr* and other image databases[3]. Every image was labeled with the absence or presence of objects from 1000 categories. For each image, 4096 high-level features were extracted using a 19-layer deep convolutional network.

Note that the *RdNN-Tree* is based on the R*-tree index structure, whose performance is known to severely deteriorate for data with representational dimension of roughly 20 or more [47]. This significantly affects the preprocessing costs of the *RdNN-Tree*. To illustrate this effect we investigate the rate at which the performance of the competing methods diminishes when considering subsamples of various sizes drawn from *Imagenet*. We downsized the full dataset by uniform random selection to subsets of 100,000 (*Imagenet*100), 250,000 (*Imagenet*250) and 500,000 (*Imagenet*500) objects. For all experiments on these sets, we limit our comparison to those methods whose preprocessing computation time is less than one week (168 hours).

8. EXPERIMENTAL RESULTS

8.1 Comparison to Prior Work

In Figures 3–6 we present the result of our comparison of *RDT* and *RDT+* with their competitors *TPL*, *MRkNN-CoP*, *SFT* and the *RdNN-Tree*. The figures show the respective results for $k \in \{10, 50, 100\}$ on the *Sequoia*, *FCT*, *ALOI* and *MNIST* datasets. On all these sets, among the exact methods, the query times of *TPL* are not competitive with *MRkNNCoP* and the *RdNN-Tree*, particularly for larger values of k; however, *TPL* had the smallest preprocessing cost of all methods studied. At the other end of the spectrum, on most of the execution runs, the *RdNN-Tree* and *MRkNNCoP* achieved the smallest query times. However, this performance comes at an enormous cost in preprocessing time, many orders of magnitude higher than those of the heuristic methods studied. The *RdNN-Tree* also has the deficiency that a new tree must be constructed for each value of k tested.

Of the three heuristic methods, in terms of tradeoff between speed and accuracy, *RDT+* strictly outperforms *RDT*, which in turn outperforms *SFT*, over all but the smallest datasets studied. On the *FCT* dataset, *SFT* has a slight edge over *RDT+* and *RDT* for certain values of k. This can be explained by the extremely fast kNN query support provided by the *Cover Tree*. While the computational overheads due to their lazy accept and lazy reject mechanism reduce the competitiveness of *RDT+* and *RDT* for the smaller datasets, for the larger sets these refinement strategies give them a distinct advantange over *SFT*.

[3]`http://www.image-net.org/challenges/LSVRC/2012`

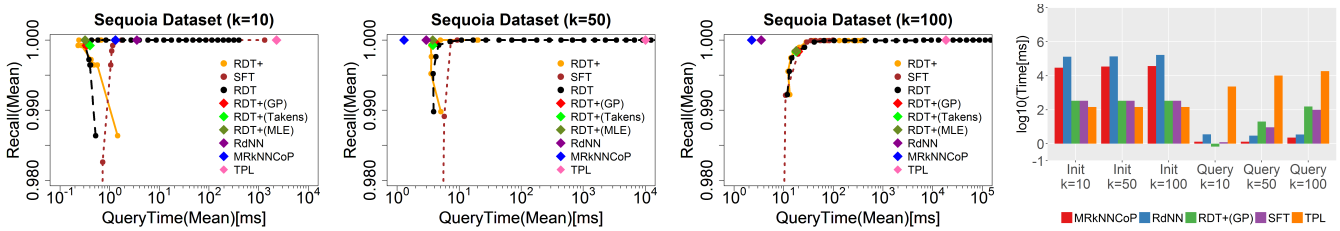

Figure 3: Comparison of recall rates and query times achieved by the competing algorithms on the *Sequoia* dataset for $k \in \{10, 50, 100\}$. The curves for *RDT*, *RDT+* and *SFT* were generated by varying the parameters t and α, respectively. We also show a comparison of the query and precomputation times on a log scale.

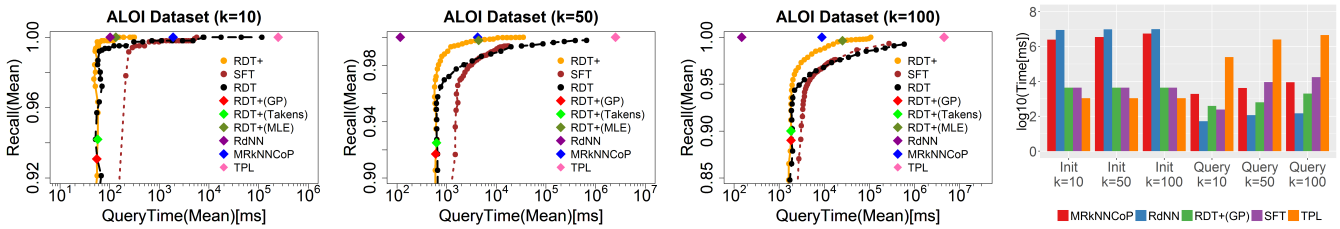

Figure 4: Comparison of recall rates and query times achieved by the competing algorithms on the *ALOI* dataset for $k \in \{10, 50, 100\}$. The curves for *RDT*, *RDT+* and *SFT* were generated by varying the parameters t and α, respectively. We also show a comparison of the query and precomputation times on a log scale.

However, for these relatively small-scale datasets, the improvements in query response time are within one order of magnitude for any given level of recall. The mean recall rates achieved by *RDT+*, *RDT* and *SFT* grow monotonically with the choices of the respective parameters t and α; unsurprisingly, as the numbers of candidates generated in the filter phase of all three algorithms is increasing. As observed in Section 7.2, the conflicting influences between increased costs of maintaining pruning information and lowered numbers of kNN verifications can be expected to lead to fluctuations in the overall query execution times. In fact, we observe such execution time fluctuations to varying extents across all the performance curves in Figures 3–6.

When the choice of the reverse neighbor rank k is low and the dataset is very small (*Sequoia*),the heuristic methods outperform their exact competitors as recall rates approach 100%. For experiments in which k is high, the query performance of *MRkNNCoP* becomes quite competitive across all datasets, outperforming *RDT* and *RDT+* easily on some. For the sets of medium scale (*FCT*, *ALOI*, *MNIST*), we observe that the advantages of *MRkNNCoP* diminish, as it is generally outperformed by *RDT+* for the sets of lower intrinsic dimensionality (ALOI and FCT). This trend can be explained by the advantage that *MRkNNCoP* gains by using precomputed upper bounds on the k-nearest neighbor distances of the data points, allowing earlier pruning of potential search paths in the underlying M-tree index.

Figures 3–6 indicate that automatic determination of the value of the scale parameter t is effective, particularly when the datasets are small. For the variants of *RDT+* in which estimators of intrinsic dimensionality are used to determine the value of the scale parameter t, this estimation is shown to be effective, particularly for the estimators of correlation dimension (*Takens*, *GP*), and particularly when intrinsic dimensionality of the dataset is relatively small (*ALOI*, *FCT*). For the *MNIST* dataset, the intrinsic dimension is overestimated by *MLE*, leading to high query times with virtually exact query results. Overall the estimators of correlation dimension seem more suitable, as they provide a single, global estimate of the intrinsic dimensionality (see Table 1).

Table 1: The intrinsic dimensionality of each data set as estimated by the different estimators used in our experiments, together with their representational dimensions (D). The average execution times (in minutes) of the estimators are shown in parentheses. The execution times of the *Takens* estimator are not explicitly shown here, as they are extremely close to those of the *GP* algorithm.

Data Set	D	MLE		GP		Takens
Sequoia	2	1.84	(0.01)	1.79	(1.65)	1.78
FCT	53	3.54	(1.08)	3.87	(562.19)	3.65
ALOI	641	7.71	(3.12)	1.98	(75.00)	2.16
MNIST	784	12.15	(10.05)	4.39	(119.08)	4.68

8.2 Lazy Acceptance and Lazy Rejection

We next examine the efficacy of the acceptance and rejection mechanisms within *RDT* and *RDT+*, which are based on the outcome of local distance computations. Figure 7 shows the proportion of candidate points that are treated by either of the aforementioned techniques. Throughout the range of values of t, the proportion of candidates accepted due to Assertion 2 remains relatively small (yet still significant). For low values of t, when the number of processed candidates is low, we see that the explicit verification mechanism accounts for a relatively large proportion of the cases. However, for larger choices of the scale parameter t, as the search expands further into the dataset and more candidates are discovered, *RDT* is given more opportunities to apply Assertion 1 to eliminate true negatives. The lazy reject mechanism, which represents a fundamental difference between *RDT* and *SFT*, incurs only constant-time overheads. With this in mind, the significant performance improvements of *RDT* over *SFT* can be explained by the ob-

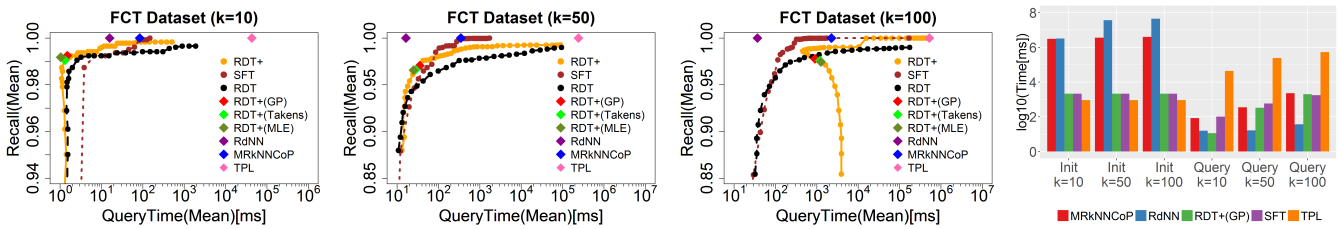

Figure 5: Comparison of recall rates and query times achieved by the competing algorithms on the *FCT* dataset for $k \in \{10, 50, 100\}$. The curves for *RDT*, *RDT+* and *SFT* were generated by varying the parameters t and α, respectively. We also show a comparison of the query and precomputation times on a log scale.

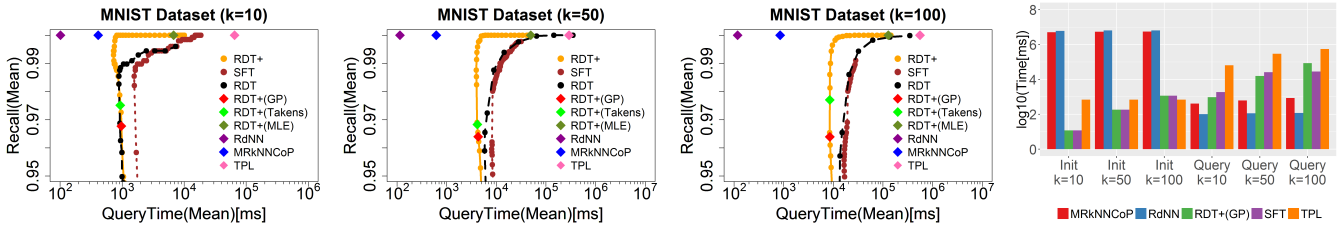

Figure 6: Comparison of recall rates and query times achieved by the competing algorithms on the *MNIST* dataset for $k \in \{10, 50, 100\}$. The curves for *RDT*, *RDT+* and *SFT* were generated by varying the parameters t and α, respectively. We also show a comparison of the query and precomputation times on a log scale.

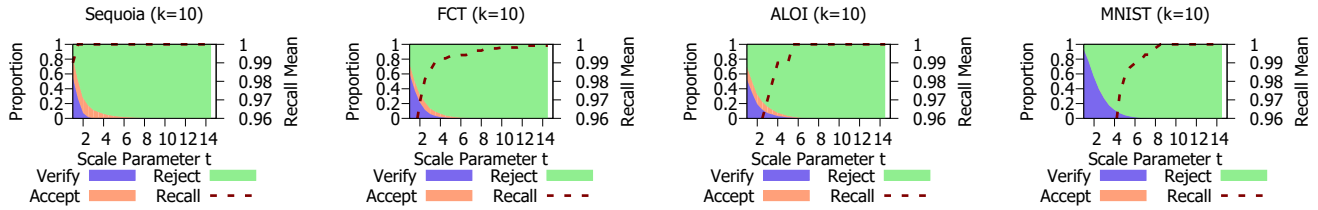

Figure 7: Comparison of the proportion of *lazy accepts*, *lazy rejects* and explicitly verified candidates performed by *RDT+* as a function of the scale parameter t, for a fixed reverse neighbor rank of $k = 10$. The dashed line represents the achieved levels of recall. The values represent averages across all 100 queries.

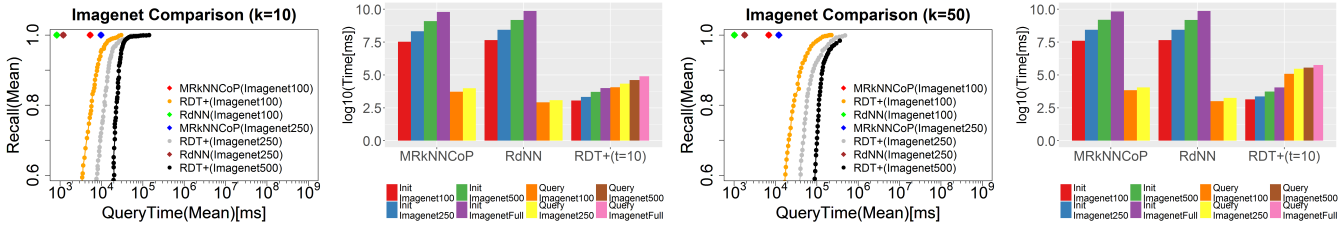

Figure 8: Comparison of the performance curves of *RDT+* with those of its competitors on subsets of the *Imagenet* dataset. The curves were generated by varying the scale parameter $t > 0$ for choices of the reverse neighbor rank $k \in \{10, 50\}$. We also compare initialization and query times.

servation that for increasingly large numbers of candidates, the majority of points are rejected by this mechanism.

8.3 Scalability

Our final set of experiments is concerned with demonstrating the scalability of the competing methods on the larger, higher-dimensional *Imagenet* dataset outlined in Section 7. In contrast to the experimentation for datasets of small to medium scale, here we only show the performance of the *RDT+* method — *SFT* and *RDT* are excluded from the presentation as both are variants of *RDT+* whose performance turned out to be consistently worse on those datasets where the *Cover Tree* can not be used for refinement. The precomputation times required by the exact *RdNN-Tree* and *MRkNNCoP* methods rendered them totally impractical for *Imagenet*: for both methods the estimated precomputation time, which we derived after precomputing a small portion of the full set, would exceed two months. In contrast, preprocessing for *RDT+* required only 11 seconds.

To demonstrate how the precomputation time scales with database size and dimensionality, we downsized the *Imagenet* dataset as described in Section 7. As can be seen from Figure 9, for the smaller samples (*Imagenet*100 and *Imagenet*250), *RdNN-Tree* and *MRkNNCoP* still outperform

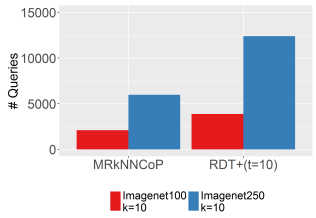

Figure 9: Comparison of the number of queries that can be processed in the time required by *RdNN-Tree* for precomputation, for the *Imagenet*100 and *Imagenet*250 datasets with reverse nearest neighbor rank of $k = 10$.

RDT+ in terms of their query performance. However, as the sample size is increased, the precomputation time for both methods rises and quickly becomes prohibitive — from approximately 60 hours for *Imagenet*250 to more than two weeks for *Imagenet*500. For this reason, *MRkNNCoP* and the *RdNN-Tree* were excluded from the experimental comparison for *Imagenet*500 and the full *Imagenet* dataset. For the full set we are unable to compute true positives in reasonable time by any method. Therefore we show the query time of *RDT+* for the value $t = 10$, for which we expect (based on the results for the smaller samples) an average recall rate of roughly 0.90. So as to give an impression of how enormous the precomputation times of *MRkNNCoP* and the *RdNN-Tree* can be, in Figure 9 we show for *Imagenet*100 and *Imagenet*250 the number of queries for each method that can be performed during the same amount of time required for the precomputation of the *RdNN-Tree*.

9. CONCLUSION

In this paper we proposed the *RDT* heuristic for finding reverse k-nearest neighbors in large data sets of high dimensionality. The quality of the approximations provided by our filter-refinement strategy is subject to the choice of scale parameter t. *RDT* is the first algorithm whose performance has been directly related to the intrinsic dimensionality of the data, in that *RDT* is guaranteed to return an exact query result whenever t exceeds the maximum generalized expansion dimension of the data set. To the best of our knowledge, our proposed algorithm is the first method to provide non-trivial theoretical guarantees on the performance of approximate reverse k-nearest neighbor queries.

The parameter α used by *SFT* (the other approximation method in our study), and the parameter t used by our method *RDT*, both determine the number of candidates to be processed. If this number of candidates is the same, then the two methods would return the same query result set. However, whereas *SFT* expresses this target number only as a multiple (α) of k, our method is sensitive to the distribution of distances encountered as measured from the query point q. As shown in Theorem 1, our algorithm is guaranteed to find all reverse k-nearest neighbor distances within a distance threshold that is a function of k, t, and the forward $(k+1)$-nearest neighbor distance from q.

We also developed a heuristic variant *RDT+* that is capable of handling data of large size and high dimensionality. To our knowledge, *RDT+* is the first method able to efficiently handle datasets of this scale. The estimators of intrinsic dimensionality described in Section 6 allow the automatic determination of the scale parameter t at a useful level of efficiency. We provided experimental results which show that with automatic determination of the scale parameter value, *RDT+* is competitive with state-of-the-art methods even for sets of low to moderate size, with the important advantage of requiring much less precomputation time. For future work, it would be interesting to study the behavior of *RDT* and *RDT+* when the value of t is dynamically adjusted during the execution of individual queries.

RDT and *RDT+* are both main-memory indices, and as such do not target applications of extreme scale. It is an interesting and challenging open problem to develop efficient parallelizable solutions for reverse k-NN search that can cope with the extreme requirements of 'big data' applications.

10. ACKNOWLEDGMENTS

M. E. Houle acknowledges the financial support of JSPS Kakenhi Kiban (A) Research Grant 25240036 and Kiban (B) Research Grant 15H02736.

11. REFERENCES

[1] E. Achtert, C. Böhm, H.-P. Kriegel, and P. Kröger. Online hierarchical clustering in a data warehouse environment. In *ICDM*, pages 10–17, 2005.

[2] E. Achtert, C. Böhm, P. Kröger, P. Kunath, A. Pryakhin, and M. Renz. Approximate reverse k-nearest neighbor queries in general metric spaces. In *CIKM*, pages 788–789, 2006.

[3] E. Achtert, C. Böhm, P. Kröger, P. Kunath, A. Pryakhin, and M. Renz. Efficient reverse k-nearest neighbor search in arbitrary metric spaces. In *SIGMOD*, pages 515–526, 2006.

[4] E. Achtert, C. Böhm, P. Kröger, P. Kunath, A. Pryakhin, and M. Renz. Efficient reverse k-nearest neighbor estimation. In *BTW*, pages 344–363, 2007.

[5] L. Amsaleg, O. Chelly, T. Furon, S. Girard, M. E. Houle, K. Kawarabayashi, and M. Nett. Estimating local intrinsic dimensionality. In *KDD*, pages 29–38, 2015.

[6] A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *ICML*, pages 97–104, 2006.

[7] J. A. Blackard and D. J. Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *CompAg*, 24:131–151, 1999.

[8] N. Boujemaa, J. Fauqueur, M. Ferecatu, F. Fleuret, V. Gouet, B. LeSaux, and H. Sahbi. IKONA: Interactive specific and generic image retrieval. In *MMCBIR*, 2001.

[9] M. A. Cheema, X. Lin, W. Zhang, and Y. Zhang. Influence zone: Efficiently processing reverse k nearest neighbors queries. In *ICDE*, pages 577–588, 2011.

[10] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *VLDB*, pages 426–435, 1997.

[11] K. L. Clarkson. Nearest neighbor queries in metric spaces. *DCG*, 22(1):63–93, 1999.

[12] T. de Vries, S. Chawla, and M. E. Houle. Finding local anomalies in very high dimensional space. In *ICDM*, pages 128–137, 2010.

[13] K. Falconer. *Fractal Geometry: Mathematical Foundations and Applications*. Wiley, 2003.

[14] J. M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders. The Amsterdam Library of Object Images. *Intern. J. Computer Vision*, 61(1):103–112, 2005.

[15] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, pages 518–529, 1999.

[16] P. Grassberger and I. Procaccia. Measuring the strangeness of strange attractors. In *The Theory of Chaotic Attractors*, pages 170–189. Springer, 2004.

[17] A. Guttman. R-Trees: A dynamic index structure for spatial searching. In *SIGMOD*, pages 47–57, 1984.

[18] V. Hautamäki, I. Kärkkäinen, and P. Fränti. Outlier detection using k-nearest neighbour graph. In *ICPR*, pages 430–433, 2004.

[19] M. Hein and J.-Y. Audibert. Intrinsic dimensionality estimation of submanifolds in R^d. In *ICML*, pages 289–296, 2005.

[20] M. E. Houle. Dimensionality, discriminability, density & distance distributions. In *ICDMW*, pages 468–473, 2013.

[21] M. E. Houle. Inlierness, outlierness, hubness and discriminability: an extreme-value-theoretic foundation. Technical Report NII-2015-002E, 2015.

[22] M. E. Houle, H. Kashima, and M. Nett. Generalized expansion dimension. In *ICDMW*, pages 587–594, 2012.

[23] M. E. Houle, X. Ma, M. Nett, and V. Oria. Dimensional testing for multi-step similarity search. In *ICDM*, pages 299–308, 2012.

[24] M. E. Houle, X. Ma, and V. Oria. Effective and efficient algorithms for flexible aggregate similarity search in high dimensional spaces. *TKDE*, 27(12):3258–3273, 2015.

[25] M. E. Houle, X. Ma, V. Oria, and J. Sun. Efficient similarity search within user-specified projective subspaces. *Inf. Syst.*, 59:2–14, 2016.

[26] M. E. Houle and J. Sakuma. Fast approximate similarity search in extremely high-dimensional data sets. In *ICDE*, pages 619–630, 2005.

[27] W. Jin, A. K. H. Tung, J. Han, and W. Wang. Ranking outliers using symmetric neighborhood relationship. In *PAKDD*, pages 577–593, 2006.

[28] D. R. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. In *STOC*, pages 741–750, 2002.

[29] F. Korn and S. Muthukrishnan. Influence sets based on reverse nearest neighbor queries. In *SIGMOD*, pages 201–212, 2000.

[30] H.-P. Kriegel, P. Kröger, M. Renz, A. Züfle, and A. Katzdobler. Incremental reverse nearest neighbor ranking. In *ICDE*, pages 1560–1567, 2009.

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

[32] M. Lichman. UCI machine learning repository, 2013. University of California, Irvine, School of Information and Computer Sciences.

[33] A. Maheshwari, J. Vahrenhold, and N. Zeh. On reverse nearest neighbor queries. In *CCCG*, pages 128–132, 2002.

[34] P. Mattila. Hausdorff dimension, orthogonal projections and intersections with planes. *Ann. Acad. Sci. Fenn. A Math.*, 1:227–244, 1975.

[35] I. Ntoutsi, A. Zimek, T. Palpanas, P. Kröger, and H.-P. Kriegel. Density-based projected clustering over high dimensional data streams. In *SDM*, pages 987–998, 2012.

[36] D. Pokrajac, A. Lazarevic, and L. J. Latecki. Incremental local outlier detection for data streams. In *CIDM*, pages 504–515, 2007.

[37] M. Radovanovic, A. Nanopoulos, and M. Ivanovic. Reverse nearest neighbors in unsupervised distance-based outlier detection. *TKDE*, 27(5):1369–1382, 2015.

[38] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[39] E. Schubert, A. Koos, T. Emrich, A. Züfle, K. A. Schmid, and A. Zimek. A framework for clustering uncertain data. *PVLDB*, 8(12):1976–1987, 2015.

[40] A. Singh, H. Ferhatosmanoglu, and A. S. Tosun. High dimensional reverse nearest neighbor queries. In *CIKM*, pages 91–98, 2003.

[41] I. Stanoi, D. Agrawal, and A. El Abbadi. Reverse nearest neighbor queries for dynamic databases. In *SIGMODW*, pages 44–53, 2000.

[42] F. Takens. *Detecting strange attractors in turbulence*. Springer, 1981.

[43] Y. Tao, D. Papadias, and X. Lian. Reverse kNN search in arbitrary dimensionality. In *VLDB*, pages 744–755, 2004.

[44] Y. Tao, M. L. Yiu, and N. Mamoulis. Reverse nearest neighbor search in metric spaces. *TKDE*, 18(9):1239–1252, 2006.

[45] J. Theiler. Lacunarity in a best estimator of fractal dimension. *Phys. Lett. A*, 133(4):195–200, 1988.

[46] N. Tomasev, M. Radovanovic, D. Mladenic, and M. Ivanovic. The role of hubness in clustering high-dimensional data. *TKDE*, 26(3):739–751, 2014.

[47] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, pages 194–205, 1998.

[48] R. C.-W. Wong, M. T. Özsu, A. W.-C. Fu, P. S. Yu, L. Liu, and Y. Liu. Maximizing bichromatic reverse nearest neighbor for l_p -norm in two- and three-dimensional spaces. *VLDB J.*, 20(6):893–919, 2011.

[49] W. Wu, F. Yang, C. Y. Chan, and K.-L. Tan. FINCH: Evaluating reverse k-nearest-neighbor queries on location data. *PVLDB*, 1(1):1056–1067, 2008.

[50] T. Xia, D. Zhang, E. Kanoulas, and Y. Du. On computing top-t most influential spatial sites. In *VLDB*, pages 946–957, 2005.

[51] C. Yang and K.-I. Lin. An index structure for efficient reverse nearest neighbor queries. In *ICDE*, pages 485–492, 2001.