Rank-Based Similarity Search: Reducing the Dimensional Dependence

Michael E. Houle and Michael Nett

Abstract—This paper introduces a data structure for k-NN search, the *Rank Cover Tree* (RCT), whose pruning tests rely solely on the comparison of similarity values; other properties of the underlying space, such as the triangle inequality, are not employed. Objects are selected according to their ranks with respect to the query object, allowing much tighter control on the overall execution costs. A formal theoretical analysis shows that with very high probability, the RCT returns a correct query result in time that depends very competitively on a measure of the intrinsic dimensionality of the data set. The experimental results for the RCT show that non-metric pruning strategies for similarity search can be practical even when the representational dimension of the data is extremely high. They also show that the RCT is capable of meeting or exceeding the level of performance of state-of-the-art methods that make use of metric pruning or other selection tests involving numerical constraints on distance values.

Index Terms—Nearest neighbor search, intrinsic dimensionality, rank-based search

1 Introduction

Of the fundamental operations employed in data mining tasks such as classification, cluster analysis, and anomaly detection, perhaps the most widely-encountered is that of similarity search. Similarity search is the foundation of k-nearest-neighbor (k-NN) classification, which often produces competitively-low error rates in practice, particularly when the number of classes is large [26]. The error rate of nearest-neighbor classification has been shown to be 'asymptotically optimal' as the training set size increases [14], [47]. For clustering, many of the most effective and popular strategies require the determination of neighbor sets based at a substantial proportion of the data set objects [26]: examples include hierarchical (agglomerative) methods such as ROCK [22] and CURE [23]; density-based methods such as DBSCAN [17], OPTICS [3], and SNN [16]; and non-agglomerative shared-neighbor clustering [27]. Content-based filtering methods for recommender systems [45] and anomaly detection methods [11] 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 very popular density-based measure, the Local Outlier Factor (LOF), relies heavily on k-NN set computation to determine the relative density of the data in the vicinity of the test point [8].

For data mining applications based on similarity search, data objects are typically modeled as feature vectors of attributes for which some measure of similarity is defined. Often, the data can be modeled as a subset $S \subset \mathcal{U}$ belonging to a metric space $\mathcal{M} = (\mathcal{U}, d)$ over some domain $\mathcal{U}$, with distance measure $d : \mathcal{U} \times \mathcal{U} \mapsto \mathbb{R}^+$ satisfying the metric postulates. Given a query point $q \in \mathcal{U}$, similarity queries over S are of two general types:

- *k-nearest neighbor queries* report a set $U \subseteq S$ of size k elements satisfying $d(q,u) \leq d(q,v)$ for all $u \in U$ and $v \in S \setminus U$.
- Given a real value $r \geq 0$, *range queries* report the set $\{v \in S \,|\, d(q,v) \leq r\}$.

While a k-**NN** query result is not necessarily unique, the range query result clearly is.

Motivated at least in part by the impact of similarity search on problems in data mining, machine learning, pattern recognition, and statistics, the design and analysis of scalable and effective similarity search structures has been the subject of intensive research for many decades. Until relatively recently, most data structures for similarity search targeted low-dimensional real vector space representations and the euclidean or other L_p distance metrics [44]. However, many public and commercial data sets available today are more naturally represented as vectors spanning many hundreds or thousands of feature attributes, that can be real or integer-valued, ordinal or categorical, or even a mixture of these types. This has spurred the development of search structures for more general metric spaces, such as the *Multi-Vantage-Point Tree* [7], the *Geometric Near-neighbor Access Tree (GNAT)* [9], *Spatial Approximation Tree (SAT)* [40], the *M-tree* [13], and (more recently) the *Cover Tree (CT)* [6].

Despite their various advantages, spatial and metric search structures are both limited by an effect often referred to as the *curse of dimensionality*. One way in which the curse may manifest itself is in a tendency of distances to concentrate strongly around their mean values as the dimension increases. Consequently, most pairwise distances become difficult to distinguish, and the triangle inequality can no longer be effectively used to eliminate candidates from consideration along search paths. Evidence suggests that when the representational dimension of feature vectors is high

- *M. E. Houle is with the National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan. E-mail: meh@nii.ac.jp.*
- *M. Nett is with Google Japan, 6-10-1 Roppongi, Minato-ku, Tokyo 106-6126, Japan. E-mail: mnett@google.com.*

Manuscript received 13 Sept. 2012; revised 14 Apr. 2014; accepted 27 May 2014. Date of publication 25 July 2014; date of current version 5 Dec. 2014. Recommended for acceptance by G. Lanckriet.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2014.2343223

(roughly 20 or more [5], [49]), traditional similarity search accesses an unacceptably-high proportion of the data elements, unless the underlying data distribution has special properties [6], [12], [41]. Even though the local neighborhood information employed by data mining applications is both useful and meaningful, high data dimensionality tends to make this local information very expensive to obtain.

The performance of similarity search indices depends crucially on the way in which they use similarity information for the identification and selection of objects relevant to the query. Virtually all existing indices make use of numerical constraints for pruning and selection. Such constraints include the triangle inequality (a linear constraint on three distance values), other bounding surfaces defined in terms of distance (such as hypercubes or hyperspheres) [25], [32], range queries involving approximation factors as in Locality-Sensitive Hashing (LSH) [19], [30], or absolute quantities as additive distance terms [6]. One serious drawback of such operations based on numerical constraints such as the triangle inequality or distance ranges is that the number of objects actually examined can be highly variable, so much so that the overall execution time cannot be easily predicted.

In an attempt to improve the scalability of applications that depend upon similarity search, researchers and practitioners have investigated practical methods for speeding up the computation of neighborhood information at the expense of accuracy. For data mining applications, the approaches considered have included feature sampling for local outlier detection [15], data sampling for clustering [50], and approximate similarity search for k-NN classification (as well as in its own right). Examples of fast approximate similarity search indices include the *BD-Tree*, a widely-recognized benchmark for approximate k-NN search; it makes use of splitting rules and early termination to improve upon the performance of the basic KD-Tree. One of the most popular methods for indexing, Locality-Sensitive Hashing [19], [30], can also achieve good practical search performance for range queries by managing parameters that influence a tradeoff between accuracy and time. The spatial approximation sample hierarchy (SASH) similarity search index [29] has had practical success in accelerating the performance of a shared-neighbor clustering algorithm [27], for a variety of data types.

In this paper, we propose a new similarity search structure, the *Rank Cover Tree* (RCT), whose internal operations completely avoid the use of numerical constraints involving similarity values, such as distance bounds and the triangle inequality. Instead, all internal selection operations of the RCT can be regarded as *ordinal* or *rank-based*, in that objects are selected or pruned solely according to their rank with respect to the sorted order of distance to the query object. Rank thresholds precisely determine the number of objects to be selected, thereby avoiding a major source of variation in the overall query execution time. This precision makes ordinal pruning particularly well-suited to those data mining applications, such as k-NN classification and LOF outlier detection, in which the desired size of the neighborhood sets is limited. As ordinal pruning involves only direct pairwise comparisons between similarity values, the RCT is also an example of a *combinatorial* similarity search algorithm [21].

The main contributions of this paper are as follows:

- A similarity search index in which only ordinal pruning is used for node selection—no use is made of metric pruning or of other constraints involving distance values.
- Experimental evidence indicating that for practical k-NN search applications, our rank-based method is very competitive with approaches that make explicit use of similarity constraints. In particular, it comprehensively outperforms state-of-the-art implementations of both LSH and the BD-Tree, and is capable of achieving practical speedups even for data sets of extremely high representational dimensionality.
- A formal theoretical analysis of performance showing that RCT k-NN queries efficiently produce correct results with very high probability. The performance bounds are expressed in terms of a measure of intrinsic dimensionality (the expansion rate [31]), independently of the full representational dimension of the data set. The analysis shows that by accepting polynomial sublinear dependence of query cost in terms of the number of data objects n, the dependence on the intrinsic dimensionality is lower than any other known search index achieving sublinear performance in n, while still achieving very high accuracy.

To the best of our knowledge, the RCT is the first practical similarity search index that both depends solely on ordinal pruning, and admits a formal theoretical analysis of correctness and performance. A preliminary version of this work has appeared in [28].

The remainder of this paper is organized as follows. Section 2 briefly introduces two search structures whose designs are most-closely related to that of the RCT: the rank-based SASH approximate similarity search structure [29], and the distance-based Cover Tree for exact similarity search [6]. Section 3 introduces basic concepts and tools needed to describe the RCT. Section 4 presents the algorithmic details of the RCT index. In Section 5 we provide a formal theoretical analysis of the theoretical performance guarantees of the RCT. Section 6 compares the practical performance of the RCT index with that of other popular methods for similarity search. Concluding remarks are made in Section 7.

2 Related Work

This paper will be concerned with two recently-proposed approaches that on the surface seem quite dissimilar: the SASH heuristic for approximate similarity search [29], and the Cover Tree for exact similarity search [6]. Of the two, the SASH can be regarded as combinatorial, whereas the Cover Tree makes use of numerical constraints. Before formally stating the new results, we first provide an overview of both the SASH and the Cover Tree.

2.1 SASH

A (SASH) is a multi-level structure recursively constructed by building a SASH on a half-sized random sample $S' \subset S$ of the object set S, and then connecting each object remaining

Algorithm **SASH-Find-Near-Neighbors** (*SASH T, query point q, query size k*)

1) Initialize the root-level set, V_h, to contain the root node of T.
2) For j from $h-1$ down to 1, do:
   a) Let V_j^* be the set of all children of nodes of V_{j+1}, and let k_j be a positive real value depending on k and j.
      i) If $|V_j^*| \leq k_j$, then choose $V_j \leftarrow V_j^*$.
      ii) Otherwise, let V_j be the set of items of V_j^* attaining the $\lfloor k_j \rfloor$ smallest distances from q — that is, satisfying $|V_j| = \lfloor k_j \rfloor$ and $d(q,v) < d(q,w)$ for all $v \in V_j$ and $w \in V_j^* \setminus V_j$.
3) Return the k elements of $V_1 \cup V_2 \cup \ldots \cup V_h$ closest to q. If the set contains fewer than k elements, return the entire set.

Fig. 1. SASH routine for finding approximate k-nearest neighbors.

outside S' to several of its approximate nearest neighbors from within S'. Queries are processed by first locating approximate neighbors within sample S', and then using the pre-established connections to discover neighbors within the remainder of the data set. The SASH index relies on a pairwise distance measure, but otherwise makes no assumptions regarding the representation of the data, and does not use the triangle inequality for pruning of search paths.

SASH construction is in batch fashion, with points inserted in level order. Each node v appears at only one level of the structure: if the leaf level is level 1, the probability of v being assigned to level j is $\frac{1}{2^j}$. Each node v is attached to at most p parent nodes, for some constant $p \geq 1$, chosen as approximate nearest neighbors from among the nodes at one level higher than v. The SASH guarantees a constant degree for each node by ensuring that each can serve as the parent of at most $c = 4p$ children; any attempt to attach more than c children to an individual parent w is resolved by accepting only the c closest children to w, and reassigning rejected children to nearby surrogate parents whenever necessary.

Similarity queries are performed by establishing an upper limit k_j on the number of neighbor candidates to be retained from level j of the SASH, dependent on both j and the number of desired neighbors k (see Fig. 1). The search starts from the root and progresses by successively visiting all children of the retained set for the current level, and then reducing the set of children to meet the quota for the new level, by selecting the k_j elements closest to the query point. In the case of k-**NN** search, when the quota values are chosen as

$$k_j = \max\left\{k^{1-\frac{j}{\log_2 n}}, \frac{1}{2}pc\right\},$$

the total number of nodes visited is bounded by

$$\frac{k^{1+\frac{1}{\log_2 n}}}{k^{\frac{1}{\log_2 n}} - 1} + \frac{pc^2}{2}\log_2 n = \tilde{\mathbf{O}}(k + \log n).$$

SASH construction can be performed in $\mathbf{O}(pcn \log n)$ time and requires $\mathbf{O}(cn)$ space. The SASH was proposed as a heuristic structure with no formal analysis of query accuracy; however, its lack of dependence on the representational dimension of the data, together with tight control on the execution time, allow it to achieve very substantial speed-ups (typically 1-3 orders of magnitude) for real data sets of sizes and representational dimensions extending into the millions, while still consistently achieving high accuracy rates [27], [29].

Very few other efficient similarity search methods are known to use rank information as the sole basis for access or pruning. The recent algorithm of [48] proposes a rank-based hashing scheme, in which similarity computation relies on rank averaging and other arithmetic operations. No experimental results for this method have appeared in the research literature as yet. Another algorithm we considered, the combinatorial random connection graph search method *RanWalk* [21], is mainly of theoretical interest, since the preprocessing time and space required is quadratic in the data set size. Due to these impracticalities, both methods are excluded from the experimentation presented in Section 6.

2.2 Cover Trees and the Expansion Rate

In [31], Karger and Ruhl introduced a measure of intrinsic dimensionality as a means of analyzing the performance of a local search strategy for handling nearest neighbor queries. In their method, a randomized structure resembling a skip list [42] is used to retrieve pre-computed samples of elements in the vicinity of points of interest. Eventual navigation to the query is then possible by repeatedly shifting the focus to those sample elements closest to the query, and retrieving new samples in the vicinity of the new points of interest. 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

$$B_S(v,r) = \{w \in S \,|\, d(v,w) \leq r\}$$

be the set of elements of S contained in the closed ball of radius r centered at $v \in S$. Given a query set $\mathcal{U}$, S is said to have (b,δ)-expansion if for all $q \in \mathcal{U}$ and $r > 0$,

$$|B_S(q,r)| \geq b \Longrightarrow |B_S(q,2r)| \leq \delta \cdot |B_S(q,r)|.$$

The *expansion rate* of S is the minimum value of δ such that the above condition holds, subject to the choice of minimum ball set size b (in their analysis, Karger and Ruhl chose $b = \mathbf{O}(\log|S|)$). Karger and Ruhl's expansion rates have since been applied to the design and analysis of routing algorithms and distance labeling schemes [1], [10], [46].

One can consider the value $\log_2 \delta$ to be a measure of the intrinsic dimension, by observing that for the euclidean distance metric in $\mathbb{R}^m$, doubling the radius of a sphere increases its volume by a factor of 2^m. When sampling $\mathbb{R}^m$ by a uniformly distributed point set, the expanded sphere would contain proportionally as many points.

However, as pointed out by the authors themselves, low-dimensional subsets in very high-dimensional spaces can

Algorithm **CT-Find-Nearest** (*Cover Tree* T, *query point* q)

1) Initialize the root-level cover set, V_h, to contain all nodes at the root level of T.
2) For j from $h-1$ down to 0, do:
   a) Let V_j^* be the set of all children of nodes of V_{j+1}.
   b) Form cover set $V_j = \{v \in V_j^* \mid d(q,v) \leq d(q,V_j^*) + 2^j\}$, where $d(q,V_j^*)$ is the minimum distance between q and the points of V_j^*.
3) Return as the nearest neighbor the point $u \in V_0$ minimizing $d(q,u)$.

Fig. 2. Cover tree routine for finding the nearest neighbor of q

have very low expansion rates, whereas even for one-dimensional data the expansion rate can be linear in the size of S. The *expansion dimension* $\log_2 \delta$ is also not a robust measure of intrinsic dimensionality, in that the insertion or deletion of even a single point can cause an arbitrarily-large increase or decrease.

Subsequently, Krauthgamer and Lee [34] proposed a structure called a *Navigating Net*, consisting of a hierarchy of progressively finer ε-nets of S, with pointers that allow navigation from one level of the hierarchy to the next. Their analysis of the structure involved a closely-related alternative to the expansion dimension that does not depend on the distribution of S. The *doubling constant* is defined as the minimum value δ_* such that every ball $B_S(q,2r)$ can be entirely covered by δ_* balls of radius r. Analogously, the *doubling dimension* is then defined as $\log_2 \delta_*$. Although Gupta et al. [24] have shown that the doubling dimension is more general than the expansion dimension, the latter is more sensitive to the actual distribution of the point set S. In [33], Krauthgamer and Lee furthermore showed that without additional assumptions, nearest neighbor queries cannot be approximated within a factor of less than $\frac{7}{5}$, unless $\log \delta_* \in \mathbf{O}(\log \log n)$.

For real-world data sets, the values of the expansion rate can be very large, typically greatly in excess of $\log_2 n$ [6]. In terms of δ, the execution costs associated with Navigating Nets are prohibitively high. Beygelzimer et al. [6] have proposed an improvement upon the Navigating Net with execution costs depending only on a constant number of factors of δ. They showed that the densely-interconnected graph structure of a Navigating Net could be thinned into a tree satisfying the following invariant *covering tree* condition: for every node u at level $l-1$ of the structure, its parent v at level l is such that $d(u,v) < 2^l$. Nearest-neighbor search in the resulting structure, called a *Cover Tree*, progresses by identifying a *cover set* of nodes C_l at every tree level l whose descendants are guaranteed to include all nearest neighbors. Fig. 2 shows the method by which the cover sets are generated, adapted from the original description in [6]. It assumes that the closest pair of points of S have unit distance.

The asymptotic complexities of Cover Trees and Navigating Nets are summarized in Fig. 3. Both methods have complexities that are optimal in terms of $n = |S|$; however, in the case of Navigating Nets, the large (polynomial) dependence on δ is impractically high. Experimental results for the Cover Tree show good practical performance for many real data sets [6], but the formal analysis still depends very heavily on δ.

The Cover Tree can also be used to answer approximate similarity queries using an early termination strategy. However, the approximation guarantees only that the resulting neighbors are within a factor of $1+\varepsilon$ of the optimal distances, for some error value $\varepsilon > 0$. The dependence on distance values does not scale well to higher dimensional settings, as small variations in the value of ε may lead to great variations in the number of neighbors having distances within the range.

3 Preliminaries

In this section, we introduce some of the basic concepts needed for the description of the RCT structure, and prove a technical lemma needed for the analysis.

3.1 Level Sets

The organization of nodes in a Rank Cover Tree for S is similar to that of the skip list data structure [37], [42]. Each node

	NavNet	CT	RCT ($h=3$)	RCT ($h=4$)	RCT ($h=5$)	LSH
Space	$\delta_*^{\mathbf{O}(1)} n$	n	n	n	n	$\approx n^{1+\rho}$
Build	$\delta_*^{\mathbf{O}(1)} n \log n$	$\delta^6 n \log n$	$c\delta^4 n^{5/3}$	$c\delta^5 n^{3/2}$	$c\delta^6 n^{7/5}$	$\approx n^{1+\rho} \log n$
Query	$\delta_*^{\mathbf{O}(1)} \log n$	$\delta^{12} \log n$	$c\delta^4 k n^{2/3}$	$c\delta^5 k n^{1/2}$	$c\delta^6 k n^{2/5}$	$\approx n^{\rho} \log n$

	RCT (fixed $\Delta \geq 2, h \approx \log_\Delta n$)	RCT (fixed $h \geq 3$)	RanWalk
Space	n	n	n^2
Build	$c\delta^{1+\lfloor \log_\phi(\sqrt{5}\,h)\rfloor} n \log^2 n$	$c\delta^{1+\lfloor \log_\phi(\sqrt{5}\,h)\rfloor} n^{1+(2/h)}$	$n^2 \log n$
Query	$c\delta^{1+\lfloor \log_\phi(\sqrt{5}\,h)\rfloor} (k + \log n) \log n$	$c\delta^{1+\lfloor \log_\phi(\sqrt{5}\,h)\rfloor} k n^{2/h}$	$\approx \delta^2 \log n \log\log n + \delta^6$

Fig. 3. Asymptotic complexities of Rank Cover Tree, Cover Tree, Navigating Nets (NavNet), RanWalk, and LSH, stated in terms of $n = |S|$, neighbor set size k, and 2-expansion rate δ. The complexity of NavNet is reported in terms of the doubling constant δ_*. For the RCT, we show the k-**NN** complexity bounds derived in Section 5 for several sample rates, both constant and sublinear. ϕ is the golden ratio $(1+\sqrt{5})/2$. For the stated bounds, the RCT results are correct with probability at least $1 - \frac{1}{n^c}$. The query complexities stated for the CT and NavNet algorithms are for single nearest-neighbor (1-**NN**) search, in accordance with the published descriptions and analyses for these methods [6], [34]. Although a scheme exists for applying LSH to handle k-**NN** queries (see Section 6), LSH in fact performs $(1+\varepsilon)$-approximate range search. Accordingly, the complexities stated for LSH are for range search; only the approximate dependence on n is shown, in terms of ρ, a positive-valued parameter that depends on the sensitivity of the family of hash functions employed. The dimensional dependence is hidden in the cost of evaluating distances and hash functions (not shown), as well as the value of ρ. The query bound for RanWalk is the expected time required to return the exact 1-**NN**.

of the bottom level of the RCT (L_0) is associated with a unique element of S.

Definition 1. *A* random leveling *of a set S is a sequence $\mathcal{L} = (L_0, L_1, \ldots)$ of subsets of S, where for any integer $j \geq 0$ the membership of L_{j+1} is determined by independently selecting each node $v \in L_j$ for inclusion with probability $\frac{1}{\Delta}$, for some real-valued constant $\Delta > 1$.*

The *level* $\lambda(v)$ of an element $v \in S$ is defined as the maximum index j such that $v \in L_j$; a copy of v appears at every level in the range $L_0, \ldots, L_j$. The smallest index h such that $L_h = \emptyset$ is the *height* of the random leveling. Other properties that will prove useful in analyzing the performance of Rank Cover Trees include:

- $\lambda(v)$ is a geometrically-distributed random variable with expected value $\mathsf{E}[\lambda(v)] = \frac{\Delta}{\Delta - 1}$.
- The probability that $v \in S$ belongs to the non-empty level set L_j is $\mathsf{Pr}[\lambda(v) \geq j] = \frac{1}{\Delta^j}$.
- The size of a random leveling, that is the sum of the cardinalities of its level sets, has expected value

$$\mathsf{E}\left[\sum_{v \in S} \lambda(v)\right] = |S| \frac{\Delta}{\Delta - 1}.$$

- The expected height of a random leveling is logarithmic in $|S|$; moreover, the probability of h greatly exceeding its expected value is vanishingly small (see [37]):

$$\mathsf{Pr}[h > (c+1)\mathsf{E}[h]] \leq 1/|S|^c.$$

3.2 Rank Function

Tree-based strategies for proximity search typically use a distance metric in two different ways: as a numerical (linear) constraint on the distances among three data objects (or the query object and two data objects), as exemplified by the triangle inequality, or as an numerical (absolute) constraint on the distance of candidates from a reference point. The proposed Rank Cover Tree differs from most other search structures in that it makes use of the distance metric solely for ordinal pruning, thereby avoiding many of the difficulties associated with traditional approaches in high-dimensional settings, such as the loss of effectiveness of the triangle inequality for pruning search paths [12].

Let $\mathcal{U}$ be some domain containing the point set S and the set of all possible queries. We assume the existence of an oracle which, given a query point and two objects in S, determines the object most similar to the query. Note that this ordering is assumed to be consistent with some underlying total order of the data objects with respect to the query. Based on such an oracle we provide an abstract formulation of ranks.

Definition 2. *Let $q \in \mathcal{U}$ be a query. The* rank function *$\rho_S : \mathcal{U} \times S \to \mathbb{N}$ yields $\rho_S(q, v) = i$ if and only if $(v_1, \ldots, v_n)$ is the ordering provided by the oracle for q and $v = v_i$. The k-nearest neighbor (k-**NN**) set of q is defined as $\mathbf{NN}(q, k) = \{v \in S \mid \rho_S(q, v) \leq k\}$.*

To facilitate the analysis of the RCT, we assume that the oracle rankings are induced by a distance metric on the items of S relative to any given query point $q \in \mathcal{U}$. In this paper, for the sake of simplicity, we will assume that no pair of items of S have identical distances to any point in $\mathcal{U}$. When desired, the uniqueness of distance values can be achieved by means of a tie-breaking perturbation scheme [20]. Furthermore, the total ranking of S can also serve to rank any subset of S. For each level set $L_j \in \mathcal{L}$, we define the rank function $\rho_{L_j} : \mathcal{U} \times L_j \to \mathbb{N}$ as $\rho_{L_j}(q, v) = |\{u \in L_j \mid \rho(q, u) \leq \rho(q, v)\}|$, and henceforth we take ρ_j and ρ to refer to ρ_{L_j} and ρ_S, respectively.

3.3 Expansion Rate

As with the Cover Tree, the RCT is analyzed in terms of Karger and Ruhl's expansion rate. With respect to a query set $\mathcal{U}$, a random leveling $\mathcal{L}$ that, for all $q \in \mathcal{U}$ and $L_j \in \mathcal{L}$, satisfies $|B_{L_j}(q, ir)| \leq \delta_i \cdot |B_{L_j}(q, r)|$ for some real value $\delta_i > 0$ and some integer $i > 1$, is said to have an i-expansion of δ_i. We consider random levelings with 2- and 3-expansion rates δ_2 and δ_3, respectively. For simplicity, in this version of the paper we will place no constraints on the minimum ball size, although such constraints can easily be accommodated in the analysis if desired. Henceforth, we shall take $B_j(q, r)$ to refer to $B_{L_j}(q, r)$. Note also that the expansion rates of random levelings may be higher than those based only on the data set itself (level 0).

One of the difficulties in attempting to analyze similarity search performance in terms of ranks is the fact that rank information, unlike distance, does not satisfy the triangle inequality in general. To this end, Goyal et al. [21] introduced the *disorder inequality*, which can be seen as a relaxed, combinatorial generalization of the triangle inequality. A point set S has real-valued disorder constant D if all triples of points $x, y, z \in S$ satisfy

$$\rho(x, y) \leq D \cdot (\rho(z, x) + \rho(z, y)),$$

and D is minimal with this property. We can derive a similar relationship in terms of expansion rates of a set S.

Lemma 1 (Rank Triangle Inequality). *Let δ_2 and δ_3 be the 2- and 3-expansion rates for $\mathcal{U}$ and the random leveling $\mathcal{L}$ of S. Then for any level set $L_l \in \mathcal{L}$, and any query object $q \in \mathcal{U}$ and elements $u, v \in S$,*

$$\rho_l(q, v) \leq \max\{\delta_2 \cdot \rho_l(q, u), \min\{\delta_2^2, \delta_3\} \cdot \rho_l(u, v)\}.$$

Proof. From the triangle inequality, we know that

$$d(q, v) \leq d(q, u) + d(u, v).$$

There are two cases to consider, depending on the relationship between $d(q, u)$ and $d(q, v)$. First, let us suppose that $d(q, u) \geq d(u, v)$. In this case, $d(q, v) \leq 2d(q, u)$. Since $B_l(q, d(q, v)) \subseteq B_l(q, 2d(q, u))$, and since $\rho_l(q, v) = |B_l(q, d(q, v))|$, we have

$$\begin{aligned} \rho_l(q, v) &\leq |B_l(q, 2d(q, u))| \\ &\leq \delta_2 |B_l(q, d(q, u))| \leq \delta_2 \, \rho_l(q, u). \end{aligned}$$

Algorithm **RCT-Build** (*leveling $\mathcal{L}$, coverage parameter ω*)

1) *Level assignment.*
   For each item $u \in S$, determine level $\lambda(u)$. Introduce u into each of the level sets $L_0, L_1, \ldots, L_{\lambda(u)}$.
2) Let h be the smallest level index such that $L_h = \emptyset$. Produce partial RCT T_{h-1} by connecting an artificial node notionally at level h to each of the items of L_{h-1}.
3) *Insertion loop.*
   For $j = h-2$ down to 0, and for every item $u \in L_j$, insert u into T_j as follows:
   a) If a copy of u also belongs to level set L_{j+1}, then set this copy to be the parent of u.
   b) Otherwise, set the parent of u to be the element $v =$ **RCT-Find-k-Nearest**$(T_{j+1}, u, 1, \omega)$.
4) Return tree T_0 as the RCT for $\mathcal{L}$.

Fig. 4. Offline construction routine for the Rank Cover Tree.

Algorithm **RCT-Find-k-Nearest** (*RCT T, query point q, neighborhood size k, coverage parameter ω*)

1) Set $V_{h-1} = L_{h-1}$.
2) For j from $h-2$ down to 0 do:
   a) Consider the set of children of V_{j+1}: $V_j^* = \bigcup_{v \in V_{j+1}} C(v)$.
   b) Let $k_j = \omega \max\left\{\frac{k}{\Delta^j}, 1\right\}$ be the quota of children to be retained at level j.
      i) If $\left|V_j^*\right| \le k_j$, then choose $V_j \leftarrow V_j^*$.
      ii) Otherwise, let V_j be the set of items of V_j^* attaining the $\lfloor k_j \rfloor$ smallest distances from q — that is, satisfying $|V_j| = \lfloor k_j \rfloor$ and $d(q,v) < d(q,w)$ for all $v \in V_j$ and $w \in V_j^* \setminus V_j$.
3) Return the k elements of V_0 closest to q according to d.

Fig. 5. k-nearest neighbor search for the Rank Cover Tree.

Now assume that $d(q,u) \le d(u,v)$. Let w be any point contained in the ball $B_l(q, d(q,v))$. Since

$$\begin{aligned} d(u,w) &\le d(u,q) + d(q,w) \le d(q,u) + d(q,v) \\ &\le d(q,u) + d(q,u) + d(u,v) \le 3d(u,v), \end{aligned}$$

the ball $B_l(u, 3d(u,v))$ entirely encloses $B_l(q, d(q,v))$. Therefore

$$\begin{aligned} \rho_l(q,v) &\le |B_l(u, 3d(u,v)| \\ &\le \delta_3 |B_l(u, d(u,v))| \le \delta_3 \, \rho_l(u,v). \end{aligned}$$

Alternatively,

$$\begin{aligned} \rho_l(q,v) &\le |B_l(u, 4d(u,v)| \\ &\le \delta_2^2 |B_l(u, d(u,v))| \le \delta_2^2 \, \rho_l(u,v). \end{aligned}$$

Combining both bounds on $\rho_l(q,v)$, the result follows. □

Henceforth, as a simplification, we shall take δ to refer to δ_2, and use only the weaker claim that $\rho_l(q,v) \le \max\{\delta \cdot \rho_l(q,u), \delta^2 \cdot \rho_l(u,v)\}$. In [36], Lifshits and Zhang showed that disorder inequalities and expansion rates are related, in that $D \le \delta^2$. Note that our bound is tighter than that presented in [36]; this improvement will be crucial to the analysis of the RCT.

4 Rank Cover Tree

The proposed Rank Cover Tree blends some of the design features of the SASH similarity search structure and the Cover Tree. Like the SASH (and unlike the Cover Tree), we shall see that its use of ordinal pruning allows for tight control on the execution costs associated with approximate search queries. By restricting the number of neighboring nodes to be visited at each level of the structure, the user can reduce the average execution time at the expense of query accuracy. The algorithmic descriptions of RCT construction and query processing are outlined in Figs. 4 and 5 respectively.

The underlying structure of the RCT is that of a tree T imposed on a random leveling $\mathcal{L}$ of the data set S. Given any $0 \le l < h$, the subtree $T_l \subset T$ spanning only the level sets $L_l, L_{l+1}, \ldots, L_{h-1} \in \mathcal{L}$ is also a Rank Cover Tree for the set L_l; T_l will be referred to as the *partial Rank Cover Tree* of T for level l. Now, for any $u \in L_l$ and any choice of $l < j < h$, we define $a_j(u) \in L_j$ to be the unique ancestor of u in T_l at level j.

RCT search proceeds from the root of the tree, by identifying at each level j a set of nodes V_j (the *cover set*) whose subtrees will be explored at the next iteration. For an item u to appear in the query result, its ancestor at level j must appear in the cover set associated with level j. V_j is chosen so that, with high probability, each true k-nearest neighbor u satisfies the following conditions: the ancestor $u_j = a_j(u)$ of u is contained in V_j, and for any query point q, the rank $\rho_j(q, u_j)$ of u_j with respect to L_j is at most a level-dependent *coverage quota* $k_j = \omega \max\{\frac{k}{\Delta^j}, 1\}$.

The real-valued parameter ω is the *coverage parameter*. It influences the extent to which the number of requested neighbors k impacts upon the accuracy and execution performance of RCT construction and search, while also establishing a minimum amount of coverage independent of k. The effects of this parameter on RCT performance is the subject of the analysis in Section 5.

Offline construction of the RCT is performed by level-ordered insertion of the items of the random leveling, with the insertion of node $v \in L_j$ performed by first searching for its nearest neighbor $w \in L_{j+1}$ using the partial Rank Cover Tree T_{j+1}, and then linking v to w. The construction steps are summarized in Fig. 4.

A Rank Cover Tree T will be said to be *well-formed* if for all nodes $u \in T$ with parent $v \ne u$, the parent v is the nearest neighbor of u from among the nodes at that level—that is, if $\rho_{\lambda(u)+1}(u,v) = 1$. In the analysis to follow, we determine conditions upon the choice of the coverage parameter ω for which the construction algorithm produces a well-formed RCT with high probability. We also derive bounds on the error probability for k-**NN** queries on a well-formed RCT.

5 Analysis

This section is concerned with proving the following theorems stating performance guarantees for the RCT query and construction operations. Here, n is the size of the data set S, $h \ge 3$ is the height of the random leveling $\mathcal{L}$, $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio, and δ is the maximum over the expansion rates of S and each level of $\mathcal{L}$.

Theorem 1. *Let T be a well-formed Rank Cover Tree of height $h = \log_\Delta n$ on data set S. For any given query item $q \in \mathcal{U}$ and size $1 \le k \le n$, consider the k-nearest neighbor set U of q with respect to S. Given some constant $c > 0$, if the coverage parameter ω is chosen such that*

$$\omega \ge \chi_{h-1}(ch + \max\{2h, e\Delta\}),$$

then with probability at least $1 - \frac{1}{n^c}$, a call to ***RCT-Find-k-Nearest*** *(T, q, k, ω) correctly returns the entire set U, with the expected number of calls to the distance oracle being in $\mathbf{O}(\omega\Delta\delta(k + h))$.*

Theorem 2. *Let $S \subseteq \mathcal{U}$ be a finite set, and let $\mathcal{L}$ be a random leveling for S of height $h = \log_\Delta n > 1$. Let the two-expansion rates of S and any level set $L \in \mathcal{L}$ be no greater than δ. Consider the Rank Cover Tree T produced via a call to* ***RCT-Build*** *$(h, \mathcal{L}, \omega)$. If the coverage parameter is chosen such that*

$$\omega \ge \chi_{h-1}(ch + \max\{2h, e\Delta\}),$$

then with probability at least $1 - \frac{1}{n^c}$, T is well-formed, and the execution time required is in $\mathbf{O}(\delta\Delta\omega nh)$.

We begin the RCT analysis with two technical lemmas, one of which relates the ranks of a query-neighbor pair with respect to two different level sets. The other bounds the average degree of nodes in an RCT.

Lemma 2. *Let v be any item of level set L_j, $0 < j < h$. Let $\alpha > e$ and $\beta > 0$ be real-valued constants. For any query item $q \in \mathcal{U}$, with respect to any level set L_l with $0 \le l < j$, we have*

$$\mathsf{Pr}\left[\rho_j(q, v) > \max\left\{\frac{\alpha}{\Delta^{j-l}}\rho_l(q, v), \beta\right\}\right] \le \max\left\{\left(\frac{e}{\alpha}\right)^{\frac{\alpha\cdot\rho_l(q,v)}{\Delta^{j-l}}}, \left(\frac{e\cdot\rho_l(q,v)}{\beta\cdot\Delta^{j-l}}\right)^{\beta}\right\}.$$

Proof. Let $U = \{u \in L_l \mid \rho_l(q, u) \le \rho_l(q, v)\}$ be the set of $\rho_l(q, v)$-nearest neighbors of q in L_l. With respect to q, the rank of v restricted to L_j is thus $\rho_j(q, v) = |U \cap L_j|$. Since L_j is formed via independent selection of the members of L_l with probability $1/\Delta^{j-l}$, the expected size of $U \cap L_j$ is $\mu = \rho_l(q, v)/\Delta^{j-l}$. We analyze two cases according to how the maximum of $\alpha\mu$ and β is determined.

Suppose that $\alpha\mu \ge \beta$. Applying the standard Chernoff bound technique [37] for the upper tail probability of $\rho_j(q, v)$ yields

$$\mathsf{Pr}\big[\rho_j(q, v) > \alpha\mu\big] < \frac{1}{e^\mu}\left(\frac{e}{\alpha}\right)^{\alpha\mu} < \left(\frac{e}{\alpha}\right)^{\alpha\mu}.$$

If on the other hand $\alpha\mu \le \beta$, by using the Chernoff bound technique again, we obtain

$$\mathsf{Pr}\big[\rho_j(q, v) > \beta\big] = \mathsf{Pr}\left[\rho_j(q, u) > \frac{\beta}{\mu}\cdot\mu\right] \le \left(\frac{e\mu}{\beta}\right)^{\beta}.$$

□

Lemma 3. *Let T be a well-formed Rank Cover Tree, and let v be a node of T at level j, where $h > j > 0$. Then the expected number of children of v is at most $\delta\Delta$.*

Proof. Let $C(v) \subseteq L_{j-1}$ be the set of children of v. Let $u \in C(v)$ be a child of v such that $d(u, v) \ge d(w, v)$ for any $w \in C(v)$. By the triangle inequality, any child w satisfies

$$d(u, w) \le d(u, v) + d(v, w) \le 2d(u, v)$$

and is therefore contained in the ball $B_{j-1}(u, 2d(u, v))$. Therefore,

$$|C(v)| \le \big|B_{j-1}(u, 2d(u, v))\big| \le \delta\big|B_{j-1}(u, d(u, v))\big| = \delta\rho_{j-1}(u, v).$$

Since L_j is generated by uniform random selection of elements from L_{j-1}, $\mathsf{E}[|C(v)|] = \Delta|C(v)| \le \Delta\delta\rho_{j-1}(u, v)$. The assumption that T is well-formed implies that $\rho_{j-1}(u, v) = 1$, and therefore that $\mathsf{E}[|C(v)|] \le \Delta\delta$. □

Let $u \in S$ be a k-nearest neighbor of a query point $q \in \mathcal{U}$. Consider the unique ancestors $u_1, u_2, \ldots, u_h$ of u on each level. Since the RCT is a tree, the search finds u if and only if the coverage parameter ω is chosen such that the $k_j \ge \rho_j(q, u_j)$ for each level $0 \le j < h$ (see Fig. 5). The following lemma provides a useful bound on the ranks of these ancestors with respect to their levels.

Lemma 4. *Let T be a well-formed Rank Cover Tree of height $h > 0$ on data set S. For any given query item $q \in \mathcal{U}$ and size $k \ge 1$, let u be a k-nearest neighbor of q with respect to $L_0 = S$. Furthermore, let $u_j = a_j(u)$ be the unique ancestor of u at level j, for $0 \le j < h$, and let $u_{-1} = q$. The rank of u_j with respect to q is at most*

$$\rho_j(q, u_j) \le \chi_j \cdot \max_{0 \le i \le j} \rho_j(u_{i-1}, u_i),$$

where $\chi_j \le \delta^{\lfloor \log_\phi(\sqrt{5}(j+1))\rfloor}$, and $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio.

Proof. Consider the sequence $u_{-1}, \ldots, u_j$, where $u_{-1} = q$. By applying the rank triangle inequality (Lemma 1) with respect to $q = u_{-1}$, u_i, and u_j, for any choice of $0 \le i < j$, we obtain the bound

$$\rho_j(q, u_j) \le \min_{0 \le i < j} \max\{\delta \cdot \rho_j(u_{-1}, u_i), \delta^2 \cdot \rho_j(u_i, u_j).\}$$

Repeated application of the rank triangle inequality over the resulting subsequences ultimately yields a bound of the form

$$\rho_j(q, u_j) \le \max_{0 \le i \le j}\{\delta^{a_i} \cdot \rho_j(u_{i-1}, u_i)\},$$

where a_i is the maximum number of factors of δ accumulated by the term $\rho_j(u_{i-1}, u_i)$ during the expansion. Let $f(m)$ be the minimum worst-case number of factors of δ necessarily incurred in deriving a bound of the form stated above, for a sequence of $m + 1$ points (or m gaps). The bound would then become

$$\rho_j(q, u_j) \le \delta^{f(j+1)} \cdot \max_{0 \le i \le j}\{\rho_j(u_{i-1}, u_i)\},$$

and therefore it suffices to show that

$$f(m) \le \lfloor \log_\phi(\sqrt{5}\, m)\rfloor.$$

Clearly, the bound holds for $m = 1$.

Suppose for some $s \in \mathbb{N}$ that m is the sth Fibonacci number F_s, which satisfies $F_1 = 1$, $F_2 = 1$, and for any integer $s > 2$,

$$F_s = \frac{1}{\sqrt{5}}\left(\phi^s - \left(-\frac{1}{\phi}\right)^s\right) = F_{s-1} + F_{s-2}.$$

Observe that $f(F_1) = f(F_2) = f(1) = 0$ and $f(F_3) = f(2) = 2$. Over a subsequence (u_a, u_{a+F_b}) with $b > 2$, we may apply the rank triangle inequality at u_a, $u_{a+F_{b-1}}$, and u_{a+F_b}, yielding the recurrence

$$f(F_s) \leq \max\{1 + f(F_{s-1}), 2 + f(F_{s-2})\}.$$

This recurrence has solution $f(F_s) \leq s - 1$ for all $s \in \mathbb{N}$.

Even if m is not a Fibonacci number, for $m \geq 2$ one can always find a unique integer $s \geq 3$ such that

$$F_{s-1} + 1 \leq m \leq F_s. \tag{1}$$

Since f is a monotonically non-decreasing function,

$$f(m) \leq f(F_s) \leq s - 1.$$

It now suffices to show that

$$\lfloor \log_\phi(\sqrt{5}\, m) \rfloor \geq s - 1.$$

Substituting for F_{s-1} in Inequality 1,

$$\begin{aligned} m &\geq \frac{1}{\sqrt{5}}\left(\phi^{s-1} - \left(-\frac{1}{\phi}\right)^{s-1}\right) + 1 \\ \sqrt{5}\, m &\geq \phi^{s-1} - \frac{1}{\phi^{s-1}} + \sqrt{5} \\ &> \phi^{s-1}, \end{aligned}$$

since $\phi^{s-1} > 1$ for all $s \geq 2$. Therefore

$$\log_\phi(\sqrt{5}\, m) > s - 1,$$

and

$$f(m) \leq s - 1 \leq \lfloor \log_\phi(\sqrt{5}\, m) \rfloor$$

as required. □

On its own, Lemma 4 is not sufficient to provide a deterministic performance bound, since neither u nor its ancestors are known until the search has reached the leaf level of the RCT. However, by combining Lemmas 2 and 4, we can derive a bound on the probability of an ancestor u_j of a true k-nearest neighbor u not being found at level j. This error probability decreases exponentially with ω.

Lemma 5. *Let T be a well-formed Rank Cover Tree of height $h > 0$ on data set S. For any given query item $q \in \mathcal{U}$ and size $k \geq 1$, let u be an element of the k-nearest-neighbor set U of q with respect to $L_0 = S$. Consider the cover sets V_j $(0 \leq j < h)$ generated as a result of a call to*

RCT-Find-k-Nearest *(T, q, k, ω). If the coverage parameter satisfies $\omega > e\chi_j$, the probability that ancestor $u_j = a_j(u)$ fails to appear in cover set V_j whenever ancestor $u_{j+1} = a_{j+1}(u)$ appears in V_{j+1} is at most*

$$\mathsf{Pr}\big[u_j \notin V_j \,|\, u_{j+1} \in V_{j+1}\big] \leq \left(\frac{e\chi_j}{\omega}\right)^{\frac{\omega}{\chi_j}}.$$

Proof. We assume that V_{j+1} contains the ancestor u_{j+1} of u_l. By definition, the set V_j^* defined at step 2b consists of all children of the nodes of V_{j+1}, and therefore must contain the ancestor u_j. The probability that u_j is not retained as a member of V_j is

$$\begin{aligned} &\mathsf{Pr}\big[u_j \notin V_j \,|\, u_{j+1} \in V_{j+1}\big] \\ &\quad = \mathsf{Pr}\big[\rho_j(q, u_j) > k_j\big] \\ &\quad = \mathsf{Pr}\left[\rho_j(q, u_j) > \omega \max\left\{\frac{k}{\Delta^j}, 1\right\}\right]. \end{aligned}$$

Applying Lemma 4 to the pair of points (q, u_j) yields

$$\rho_j(q, u_j) \leq \chi_j \max_{1 \leq i \leq j}\big\{\rho_j(q, u_0), \rho_j(u_{i-1}, u_i)\big\}.$$

Combining with the error probability bound, we obtain

$$\begin{aligned} &\mathsf{Pr}\big[u_j \notin V_j \,|\, u_{j+1} \in V_{j+1}\big] \\ &\leq \mathsf{Pr}\left[\chi_j \max_{0 \leq i < j}\big\{\rho_j(q, u_0), \rho_j(u_i, u_{i+1})\big\} > \omega \max\left\{\frac{k}{\Delta^j}, 1\right\}\right] \\ &\leq \max_{0 \leq i < j}\left\{\mathsf{Pr}\left[\rho_j(q, u_0) > \omega \max\left\{\frac{\rho_0(q, u_0)}{\Delta^j \chi_j}, \frac{1}{\chi_j}\right\}\right],\right. \\ &\quad \left.\mathsf{Pr}\big[\rho_j(u_i, u_{i+1}) \right. \\ &\quad \left. > \omega \max\left\{\frac{1}{\chi_j}, \frac{k}{\Delta^{i+1}\chi_j} \cdot \frac{\rho_{i+1}(u_i, u_{i+1})}{\Delta^{j-i-1}}\right\}\right]\right\}. \end{aligned}$$

This last inequality holds since $u = u_0$ is a k-nearest-neighbor of q, implying that $\rho_0(q, u_0) \leq k$, and since T is well-formed, implying that $\rho_i(u_{i-1}, u_i) \leq 1$. Applying Lemma 2 to the probability gives

$$\begin{aligned} &\mathsf{Pr}\big[u_j \notin V_j \,|\, u_{j+1} \in V_{j+1}\big] \\ &\quad \leq \max_{0 \leq i < j}\left\{\left(\frac{e\chi_j}{\omega}\right)^{\frac{\omega}{\chi_j}}, \left(\frac{e\chi_j}{\Delta^{j-i-1}\omega}\right)^{\frac{\omega}{\chi_j}}\right\} = \left(\frac{e\chi_j}{\omega}\right)^{\frac{\omega}{\chi_j}}. \end{aligned}$$

□

We are now ready for the proofs of Theorems 1 and 2. As with the previous lemma, which bounds the probability of an ancestor of a neighbor failing to belong to the cover set at a given level, we can bound the overall probability of the failure of the search and construction operations by summing these individual error rates over all ancestors involved in the operation.

Proof of Theorem 1. First, we note that the number of calls to the oracle is proportional to the total number of nodes visited during the descent of the RCT. Lemma 3 implies that for each node retained in the cover set at a given level, the expected number of children visited is at most $\delta\Delta$. Since the total number of nodes in the cover sets is in $\mathbf{O}(\omega(k + h))$, the complexity bound follows.

Noting that $k < n$ and $h \leq n$, Lemma 5 implies that a search in any RCT of height h satisfies

$$\begin{aligned} &\mathsf{Pr}[\text{ U incorrectly returned}] \\ &\quad \leq k \cdot (h-1) \cdot \left(\frac{e\chi_{h-1}}{\omega}\right)^{\frac{\omega}{\chi_{h-1}}} \\ &\quad \leq n^2 \cdot \left(\frac{e\chi_{h-1}}{\omega}\right)^{\frac{\omega}{\chi_{h-1}}}. \end{aligned} \tag{2}$$

Since

$$\omega \geq \chi_{h-1}(c\lceil \log_\Delta n \rceil + \max\{2\lceil \log_\Delta n \rceil, e\Delta\}) \geq \chi_{h-1} t$$

for

$$t = c \log_\Delta n + \max\{2 \log_\Delta n, e\Delta\},$$

and since $\Delta > 1$, t is at least $e \cdot \chi_{h-1}$. Therefore, Inequality 2 can be extended as follows.

$$n^2 \left(\frac{e\chi_{h-1}}{\omega}\right)^{\frac{\omega}{\chi_{h-1}}} \leq n^2 \left(\frac{e}{t}\right)^t. \tag{3}$$

Subsequently, Inequality 3 is transformed by applying the (monotonic) logarithm function with base Δ. This yields

$$\begin{aligned} &\log_\Delta \mathsf{Pr}[U \text{ incorrectly returned}] && (4)\\ &\leq \log_\Delta \left(n^2 \left(\frac{e}{t}\right)^t\right) \\ &= 2 \cdot \log_\Delta n + t \cdot \log_\Delta \frac{e}{t} \\ &= 2 \cdot \log_\Delta n - t \cdot \log_\Delta \frac{t}{e} && (5)\\ &= \log_\Delta n \left(2 - \frac{t}{\log_\Delta n} \cdot \log_\Delta \frac{t}{e}\right). \end{aligned}$$

By definition, t is at least $\Delta \cdot e$, which gives

$$\log_\Delta \frac{t}{e} \geq 1.$$

Therefore, the derivations up to Inequality 5 can be extended in the following manner:

$$\log_\Delta n \left(2 - \frac{t}{\log_\Delta n} \cdot \log_\Delta \frac{t}{e}\right) \leq \log_\Delta n \left(2 - \frac{t}{\log_\Delta n}\right).$$

Subsequently, using the fact that

$$\max\left\{2, \frac{e \cdot \Delta}{\log_\Delta n}\right\} \geq 2,$$

and by reinserting t into the above inequality, one obtains

$$\begin{aligned} &\log_\Delta \mathsf{Pr}[U \text{ incorrectly returned}] \\ &\leq \log_\Delta n \left(2 - \left(c + \max\left\{2, \frac{e\Delta}{\log_\Delta n}\right\}\right)\right) \\ &\leq \log_\Delta n(2 - (c+2)) \\ &= -c \log_\Delta n \\ &= \log_\Delta \frac{1}{n^c}. \end{aligned}$$

Since $\Delta > 1$, exponentiation yields the result of Theorem 1:

$$\mathsf{Pr}[U \text{ incorrectly returned}] \leq \frac{1}{n^c}.$$

□

Note that Theorem 1 bounds the number of calls to the oracle made during the processing of a query. The asymptotic complexity bounds also apply to the additional cost incurred at runtime, such as that of maintaining a set of tentative nearest neighbors.

Proof of Theorem 2. Analogous to Theorem 1. Since the construction requires exactly one search query per point $v \in S$ the complexity is an immediate result of the query time complexity in Theorem 1. □

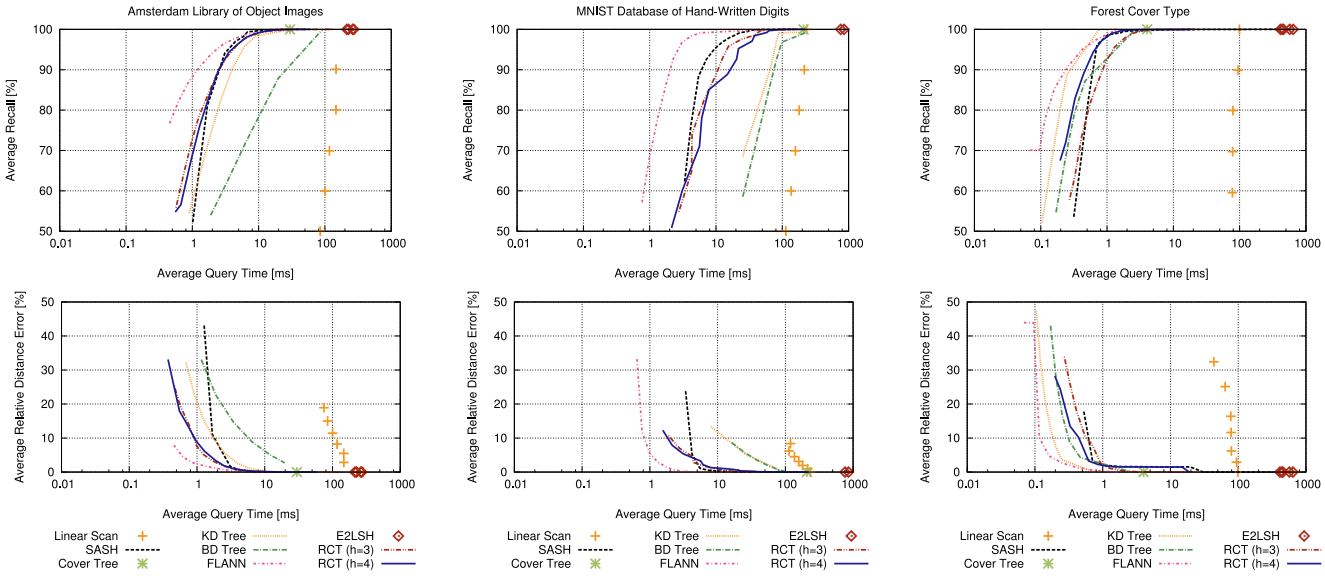
The RCT complexity bounds can be further simplified if one assumes either that the sampling rate Δ is constant, or the number of levels h is constant ($\Delta = n^{\frac{1}{h}}$). These cases are shown in Fig. 3. It should be noted that when the number of levels is fixed to $h = 3$ or $h = 4$, the dependence of the RCT search time bound on δ is δ^4 or δ^5, respectively. For small fixed choices of h, the dependence is much less than that of the Cover Tree (at δ^{12}), while still maintaining sublinear dependence on n.

6 Evaluation

We investigated and compared the performance of different approaches to exact and approximate k-nearest neighbor search in general metric spaces. The experimental evaluation assessed the query performance of the various methods in terms of the tradeoff between accuracy and execution time; the results are presented in Figs. 6a–7, 8, 9, 10, and 11, with accuracy plotted in terms of both recall and relative distance error.

6.1 Methods

We compared the fixed-height variant of the RCT (for heights 3, 4, 8 and 16) against the Cover Tree, the SASH, and (for those instances in which the L_2-norm is applicable) the popular E^{2}LSH [2] implementation of LSH. The performance of the RCT may be influenced by scaling the coverage parameter introduced in Section 4, and the accuracy of LSH is subject to the choice of a parameter governing failure probability. For the Cover Tree, we used an implementation provided by the authors [6] that is capable of handling k-**NN** queries. In addition, we compared the performance of RCT against two libraries for approximate k-NN search based on the KD-Tree: the first library, ANN [38], provides implementations of the KD-Tree and BD-Tree (box decomposition tree). The BD-Tree was run as recommended by the authors, using the 'sliding midpoint' and 'simple shrinking' rules. The accuracies of the KD-Tree and BD-Tree may be influenced by parameters governing admissible distance error. The second library, FLANN [39], uses an ensemble of KD-Tree indices and clustering-based KMeans trees, along with automatic parameter tuning of these indices for optimized performance. The experimentation was performed with the FLANN default parameter settings. The accuracy of FLANN may further be influenced by varying a parameter that governs the maximum number of leaf nodes that can be searched. As a baseline, we also tested the performance of sequential search (linear scan) over random samples of the data, of varying sizes.



(a) The *Amsterdam Library of Object Images* [18] contains **110,250** vectors, each containing a total of **641** features consisting of color and texture histograms.

(b) The *MNIST Database of Hand-written Digits* [35] accommodates **70,000** recordings of hand-written digits by 500 writers. The digits are normalized to a 28×28 grid, or **784** features.

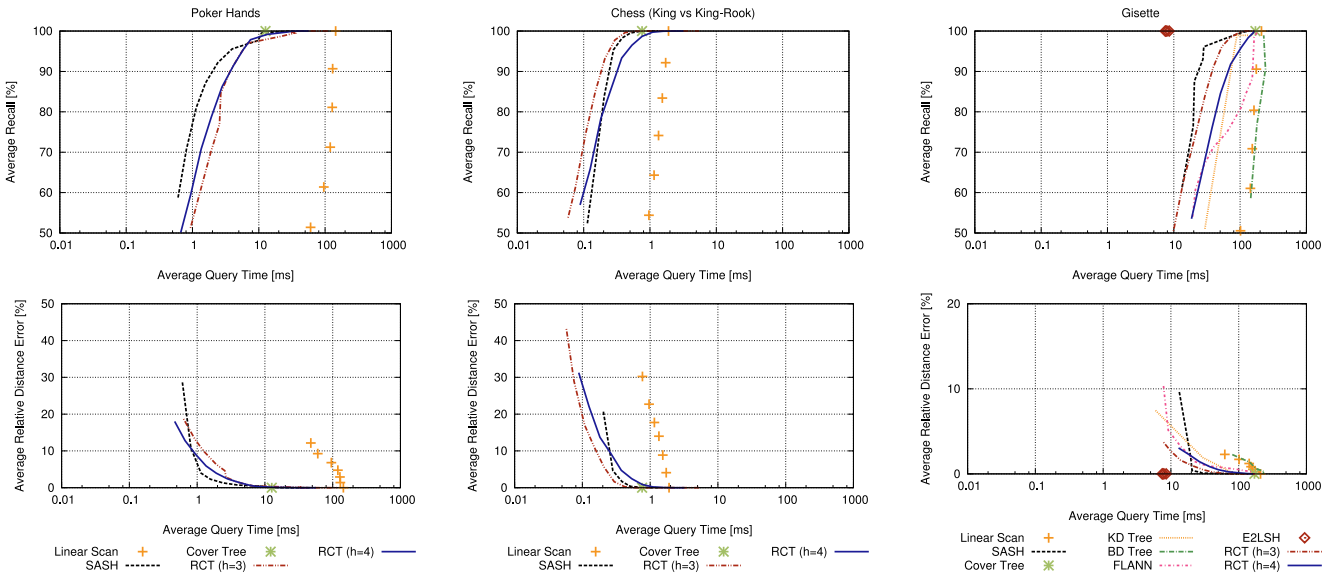
(c) The *Forest CoverType* [4] consists of **581,012** instances of topographical information on forest cells of $900m^2$ each. The **53** attributes include elevation, slope, soil and vegetation cover types.

Fig. 6. Performance curves obtained for the tested methods on the *Amsterdam Library of Object Images*, *MNIST Database of Hand-Written Digits*, and the *Forest Cover Type* data sets.

Unfortunately, E^{2}LSH offers native support exclusively for range queries. For k-nearest neighbor search, the E^{2}LSH manual [2] suggests the construction of a hierarchy of structures for query ranges r, $\frac{r}{2}$, $\frac{r}{4}$, ..., with r being the largest pairwise distance within the data set. We constructed a hierarchy of eight such structures, each given 1 Gb of memory, and queried them in ascending order of their associated range sizes, until at least k results were returned. However, as the size of the neighborhoods obtained in this manner were often greatly in excess of k, we introduced an additional filtering step in which the k-nearest neighbors were extracted from among the elements of the result sets. The query times presented for E^{2}LSH include the time spent on the additional filtering. We justify this by the fact that the range query results obtained were frequently orders of magnitude larger than the number of requested neighbors.

6.2 Data Sets

We chose a wide variety of publicly available data sets in order to demonstrate the behavior of the investigated



(a) The *Poker Hand* data set [4] consists of **1,025,010** poker hands. A hand is represented by **10** categorical attributes describing the suit and rank of each individual card in the hand. The mismatch distance was used.

(b) The *Chess (King vs. King-Rook)* set [4] contains **28,056** chess end-game situations involving both kings and a rook. The vectors span **6** categorical features. The mismatch distance was used as the dissimilarity measure.

(c) The *Gisette* data set [4] contains **13,500** recordings of the handwritten digits 4 and 9. The **5,000** features also includes artificial noise and distractors.

Fig. 7. Performance curves obtained for the tested methods on the *Poker Hands*, *Chess (King versus King-Rook)*, and the *Gisette* data sets.

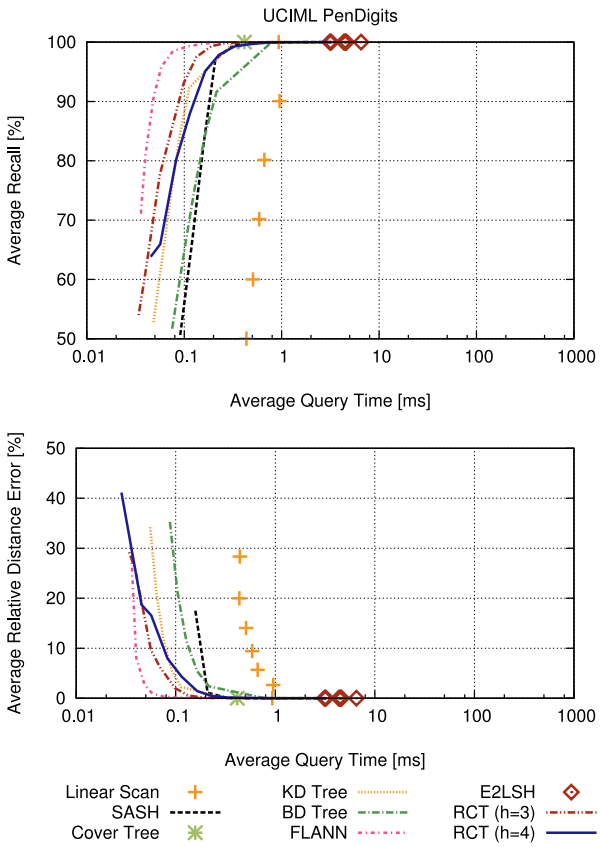


Fig. 8. The *Pen-Based Recognition of Handwritten Digits Data Set* [4] contains **10,992** recordings of handwritten digits. The **16** features represent coordinates obtained by spatial resampling.

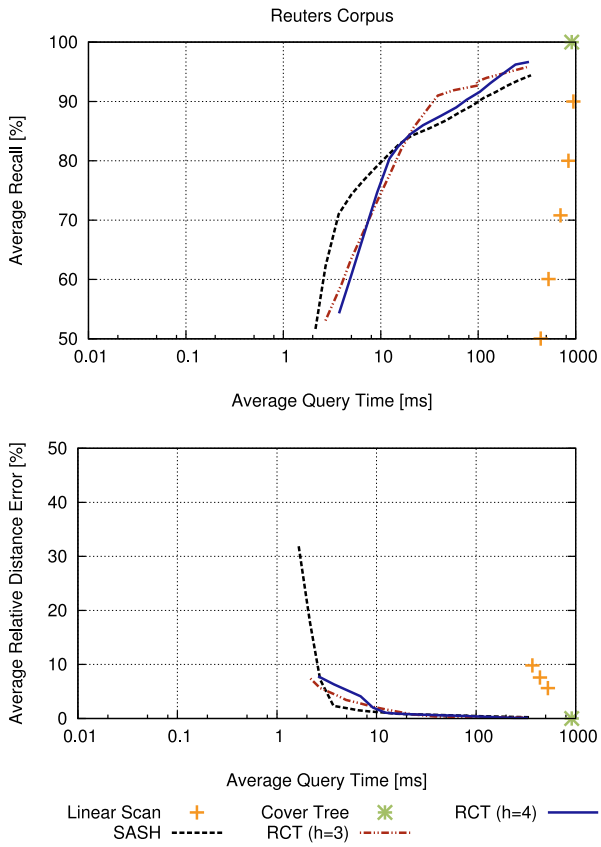


Fig. 9. A selection of **554,651** documents drawn from the *Reuters Corpus Vol. 2* [43] newspaper article data set (RCV2). The (sparse) vectors spanned **320,647** keyword dimensions, with TF-IDF weighting. The vector angle distance (equivalent to cosine similarity) was used as the similarity measure.

methods across different data types, set sizes, representational dimensions and similarity measures. For each individual data set, we selected 100 objects uniformly at random as query locations.

We measured the average time required to find the 100-nearest neighbors for each of these queries. For those methods making use of randomization, we averaged the results across 10 successive builds of the index structure. Except when otherwise stated, the distance measure employed was the euclidean distance. For those data sets for which some other distance measure is more appropriate, the E^2LSH, KD-Tree, BD-Tree and FLANN were not evaluated (as the available implementations require the use of the L_2 or L_p norms). We furthermore investigated the variation in query time and achieved recall rates across different queries for selected data sets.

So as to compare the relative performance of FLANN and RCT (with $h = 4$) on data sets of extremely high (but sparse) dimensionality, we compared their performance on projections of a document data set along randomly-selected dimensions between 512 and 4,096. In order that FLANN could be applied, the resulting vectors were normalized, and the L_2 distance was employed as the similarity measure for both methods.

6.3 Measure of Accuracy

We measured the accuracy of the methods in terms of both distance error and recall, the latter perhaps being a more appropriate measure for k-NN query performance. The recall is defined as the proportion of true nearest neighbors returned by an index structure. That is, given a query q and an approximate k-nearest neighbor list $U^* = (u_k^*)$ and its exact counter-part $U = (u_k)$, the recall is

$$\frac{|\{d(q,u)|u \in U\} \cap \{d(q,u^*)|u^* \in U^*\}|}{|U|}.$$

Note that we do not penalize missing a true k-nearest neighbor, if instead another point with identical query-distance is returned. The relative distance error of that particular query result is defined as the smallest value $\varepsilon \geq 0$ satisfying

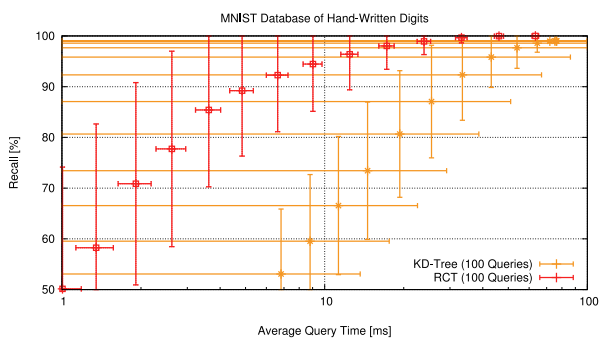


Fig. 10. Average recall and query times achieved by the RCT ($h = 4$) and KD-Tree over 100 queries on the *MNIST* [35] handwritten digit data set. The error bars represent one standard deviation.

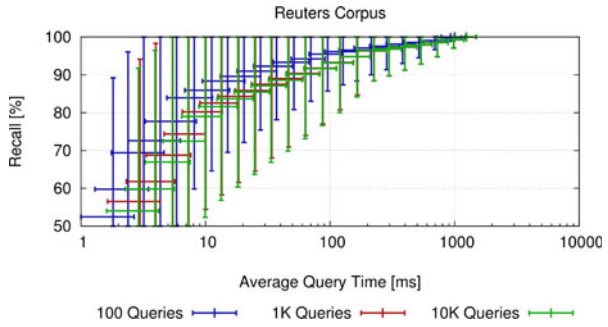


Fig. 11. Average recall and query times achieved by the RCT ($h = 4$) over different numbers of queries on the *Reuters Corpus Vol. 2* [43] newspaper article data set. The error bars represent one standard deviation.

$$d(q, u_k) \leq (1 + \varepsilon) \cdot d(q, u_k^*).$$

Average distance errors will be reported only for those queries for which no fewer than k items appear in the result set.

6.4 Results

The experimental results are presented in Figs. 6, 7, 8, 9, 10, and 11. In all instances tested, the fixed-height variants of the RCT for heights 3 and 4 performed very well, in that speed-ups over sequential search of more than 10 times (and in some cases, 100 times or more) were obtained for recall rates in excess of 90 percent. For no datasets were the RCT performances for heights $h = 8$ and $h = 16$ competitive with those for $h = 3$ and $h = 4$; these results were omitted from the plots for the sake of clarity of exposition. In general, we recommend the use of $h = 4$ for all data sets of scales similar to those considered in this paper.

The performances of the RCT and the SASH were generally competitive with those of the other methods for the lower-dimensional data sets; for the higher-dimensional sets, the RCT and SASH dominated. Their main competitor was the ensemble method FLANN, which tended to outperform RCT for those data sets for which the Euclidean distance was appropriate. However, it must be noted that these data sets were of relatively small representational dimension. Furthermore, although the accuracies achieved by FLANN were high, the results show that FLANN query times become impractically large as the dimensionality rises, due to their computation of dense mean vectors (see Fig. 12). The RCT is capable of good performance even for data sets of extremely high representational dimensionality, as can be seen from Fig. 9.

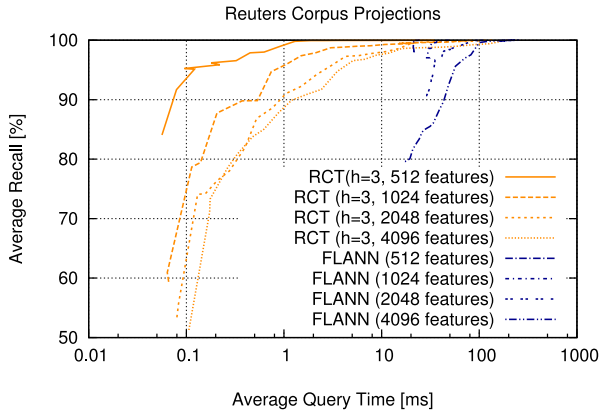


Fig. 12. A comparison of FLANN and RCT on projections of the RCV2 set consisting of 512, 1,024, 2,048 and 4,096 dimensions, selected randomly. After projection, vectors were normalized (after eliminating any zero-length vectors). The accuracies and execution times were averaged over 100 queries, using the euclidean distance as the dissimilarity measure.

The RCT and SASH query times are generally much more consistent than the other approximation algorithms tested. Although space limitations prevent a full exposition of the variation in query performance, as an example we contrast the performance of the RCT and KD-Tree on the MNIST data set by showing the variation in query time and recall rate across 100 queries, in Fig. 10. For both methods, the degree of variation in the recall rate increases as the mean recall rate decreases; however, the variation in query time is much lower for the RCT. In Fig. 11, where plots are shown for the RCV2 set over 100, 1,000, and 10,000 queries, we see that there is little change in the performance curves as the number of queries increases.

Although LSH is known to provide fast approximations of high quality for range-based queries, it performed rather poorly on all but one of the sets tested (those for which the Euclidean distance was used as a similarity measure). The time-accuracy trade-off controlled by the failure probability parameter of the LSH was in many cases so erratic that performance curves could not be generated; instead, the measurements are displayed as a scatter plot.

The performance of the Cover Tree was more competitive than E^{2}LSH, substantially improving upon sequential

TABLE 1
Construction Times (in Seconds) for the Various Methods Tested

			ANN Framework			E^{2}LSH			RCT	
Data Set	Size	Dim.	BD-Tree	KD-Tree	Cover Tree	$\psi = 0.9$	FLANN	SASH	$h = 3$	$h = 4$
ALOI	109,500	641	164.14	2.55	12.62	6787.75	140.00	58.07	399.68	340.63
Chess	28,056	6	—	—	0.06	—	—	0.98	2.34	2.57
Cover Type	581,012	55	53.35	4.72	5.57	3097.30	93.38	82.04	884.09	498.26
Gisette	7,000	5,000	11.34	3.68	243.20	3106.68	36.41	16.10	25.21	42.88
MNIST	70,000	784	110.29	33.18	145.28	8236.89	94.19	43.69	219.48	201.81
Pendigits	10,992	16	0.05	0.02	0.03	50.22	0.31	0.31	0.57	0.82
Poker	1022,771	10	—	—	5.94	—	—	112.76	1044.63	641.38
Reuters Corpus	554,651	32,0647	—	—	97435.90	—	—	415.09	4393.51	2736.37

For the RCT experiments, the trees were constructed using a coverage parameter value of $\omega = 64$. The measurements presented for E^{2}LSH were obtained by setting the success probability ψ to 90 percent.

TABLE 2
Comparison of Construction Times (in Seconds) between the FLANN and the RCT on the Subspace Projections of the Reuters Corpus

			RCT	
Size	Dim.	FLANN	$h = 3$	$h = 4$
92,827	512 (426)	22.53	22.14	19.28
150,257	1024 (832)	110.69	60.64	42.42
242,320	2,048 (1657)	528.93	156.04	88.76
321,547	4096 (3285)	2481.48	269.87	131.35

The sizes listed are the numbers of non-zero vectors in the projected data sets. The dimensions listed in parentheses are the numbers of attributes for which at least one vector achieved a non-zero value. RCT construction was performed using a coverage parameter value of $\omega = 64$.

search for the Forest Cover Type (Fig. 6c) and the Poker Hand (Fig. 7a) data sets. However, it was generally outperformed by the RCT, even for very high recall rates.

Trends observable among the results for the KD-Tree show that it is somewhat competitive on data sets of low representational dimensionality. Its performance, however, degrades to that of the sequential scan on data sets of moderate to high representational dimension. On only one of the investigated instances did the KD-Tree outperform the RCT variants of heights 3 and 4 (the 53-dimensional Forest Cover Type set, shown in Fig. 6c).

The comparable performance of the SASH and the RCT may be explained in light of the similarities between the two methods. The SASH can be regarded as a heuristic variant of the RCT, with additional parent pointers introduced as a way of recovering from errors at search time, and with higher query costs due to the extra overheads associated with the multiple parent pointers. The redundancy sometimes allowed the SASH to outperform some of the RCT variants over the very highest ranges of accuracy (recall rates of roughly 90 percent or more), but in most situations it was rendered less competitive than the RCT when tuned for faster, less accurate performance.

The construction costs for the various indices are reported in Tables 1 and 2. The time taken for RCT construction was generally greater than that of the BD-Tree, KD-Tree, and SASH, it was consistently much less than that of E^2LSH and (for higher-dimensional data sets) FLANN. Like the SASH, the RCT does not suffer from the excessively high construction costs of the other methods. For the full-dimensional Reuters Corpus data set, these methods were the only two to successfully complete within one day (the Cover Tree required slightly more than 27 hours). In order to show the effect of increasing dimensionality on construction time, we listed the RCT and FLANN construction times for the reduced-dimensional Reuters data sets, in Table 2. The results confirm the superior performance of RCT over FLANN for very high-dimensional data.

Finally, we note that the consistently good RCT performance for $h = 3$ and $h = 4$ compared with higher choices of h reflects the importance of reducing the dependence of the execution cost on the (intrinsic) dimensionality of the data. This trend in the performance of the fixed-height variants of the RCT has been validated on categorical data as well as numerical data, and for text data as well as image data and other data types.

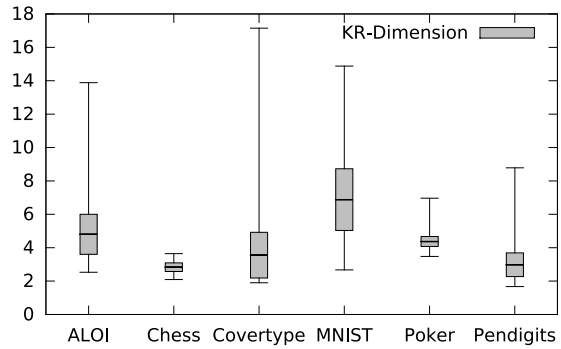


Fig. 13. Box-and-whisker plots of $\log_2 \delta$ estimated for several data sets. Maximum and minimum values are indicated by the whiskers, whereas the box indicates the range spanned by values falling within one standard deviation of the mean. The estimation was performed using every item as a center, over a range of inner ball sizes, from $\log_2 n$ to 100.

7 Conclusion

We have presented a new structure for similarity search, the Rank Cover Tree, whose ordinal pruning strategy makes use only of direct comparisons between distance values. The RCT construction and query execution costs do not explicitly depend on the representational dimension of the data, but can be analyzed probabilistically in terms of a measure of intrinsic dimensionality, the expansion rate. The RCT is the first practical rank-based similarity search index with a formal theoretical performance analysis in terms of the expansion rate; for small choices of parameter h, its fixed-height variant achieves a polynomial dependence on the expansion rate of much smaller degree than attained by the only other practical polynomially-dependent structure known to date (the Cover Tree), while still maintaining sublinear dependence on the number of data objects (as does LSH). An estimation of the values of the expansion rates is shown in Fig. 13 for several of the data sets considered in the experimentation; they show that in most cases, the ability to trade away many factors of the expansion rate more than justifies the acceptance of a polynomial cost in terms of n. The experimental results support the theoretical analysis, as they clearly indicate that the RCT outperforms its two closest relatives—the Cover Tree and SASH structures—in many cases, and consistently outperforms the E^2LSH implementation of LSH, classical indices such as the KD-Tree and BD-Tree, and—for data sets of high (but sparse) dimensionality—the KD-Tree ensemble method FLANN.

Acknowledgments

Michael Houle acknowledges the financial support of JSPS Kakenhi Kiban (A) Research Grant 25240036, the JST ERATO Kawarabayashi Large Graph Project, and the JST ERATO Minato Discrete Structure Manipulation System Project. We are also grateful to Ilia Zvedeniouk for providing us with patches that allowed us to include the ANN library in our experimental evaluation. Much of Nett's work on this paper was conducted while he was with the University of Tokyo, Japan, and the National Institute of Informatics, Japan.

REFERENCES

[1] I. Abraham, D. Malkhi, and O. Dobzinski, "LAND: Stretch (1 + epsilon) locality-aware networks for DHTs," in *Proc. 15th Annu. ACM-SIAM Symp. Discrete Algorithm*, 2004, pp. 550–559.

[2] A. Andoni and P. Indyk. (2005). *E^2LSH 0.1: User Manual*. [Online]. Available: www.mit.edu/andoni/LSH/, 2005.

[3] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 1999, pp. 49–60.

[4] A. Asuncion and D. J. Newman. (2007). UCI machine learning repository. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[5] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "nearest neighbor" meaningful?" in *Proc. 7th Int. Conf. Database Theory*, 1999, pp. 217–235.

[6] A. Beygelzimer, S. Kakade, and J. Langford, "Cover trees for nearest neighbor," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 97–104.

[7] T. Bozkaya and M. Ozsoyoglu, "Indexing large metric spaces for similarity search queries," *ACM Trans. Database Syst.*, vol. 24, no. 3, pp. 361–404, 1999.

[8] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," *SIGMOD Rec.*, vol. 29, no. 2, pp. 93–104, 2000.

[9] S. Brin, "Near neighbor search in large metric spaces," in *Proc. 21th Int. Conf. Very Large Data Bases*, 1995, pp. 574–584.

[10] H. T.-H. Chan, A. Gupta, B. M. Maggs, and S. Zhou, "On hierarchical routing in doubling metrics," in *Proc. 15th Annu. ACM-SIAM Symp. Discrete Algorithm*, 2005, pp. 762–771.

[11] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, 2009.

[12] E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín, "Searching in metric spaces," *ACM Comput. Surv.*, vol. 33, no. 3, pp. 273–321, 2001.

[13] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An efficient access method for similarity search in metric spaces," in *Proc. 23rd Int. Conf. Very Large Data Bases*, 1997, pp. 426–435.

[14] T. Cover, and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.

[15] T. de Vries, S. Chawla, and M. E. Houle, "Finding local anomalies in very high dimensional space," in *Proc. IEEE Int. Conf. Data Mining*, 2010, pp. 128–137.

[16] L. Ertöz, M. Steinbach, and V. Kumar, "Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data," in *Proc. 3rd SIAM Int. Conf. Data Mining*, 2003, p. 1.

[17] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discov. Data Mining*, 1996, pp. 226–231.

[18] J. M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders, "The Amsterdam library of object images," *Int. J. Comput. Vis.*, vol. 61, no. 1, pp. 103–112, 2005.

[19] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. 25th Int. Conf. Very Large Data Bases*, 1999, pp. 518–529.

[20] J. E. Goodman and J. O'Rourke, Eds., *Handbook of Discrete and Computational Geometry*. Cleveland, OH, USA: CRC Press, 1997.

[21] N. Goyal, Y. Lifshits, and H. Schütze, "Disorder inequality: A combinatorial approach to nearest neighbor search," in *Proc. Intern. Conf. Web Search Web Data Mining*, 2008, pp. 25–32.

[22] S. Guha, R. Rastogi, and K. Shim, "ROCK: A robust clustering algorithm for categorical attributes," *Inf. Syst.*, vol. 25, no. 5, pp. 345–366, 2000.

[23] S. Guha, R. Rastogi, and K. Shim, "CURE: An efficient clustering algorithm for large databases," *Inf. Syst.*, vol. 26, no. 1, pp. 35–58, 2001.

[24] A. Gupta, R. Krauthgamer, and J. R. Lee, "Bounded geometries, fractals, and low-distortion embeddings," in *Proc. 44th Annu. IEEE Symp. Foundations Comput. Sci.*, 2003, p. 534.

[25] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proc. Annu. Meeting*, 1984, pp. 47–57.

[26] J. Han, and M. Kamber, *Data Mining: Concepts and Techniques*, San Francisco, CA, USA: Morgan Kaufmann, 2006.

[27] M. E. Houle, "The relevant set correlation model for data clustering," *Statist. Anal. Data Mining*, vol. 1, no. 3, pp. 157–176, 2008.

[28] M. E. Houle and M. Nett, "Rank cover trees for nearest neighbor search," in *Proc. Int. Conf. Similarity Search Appl.*, 2013, pp. 16–29.

[29] M. E. Houle and J. Sakuma, "Fast approximate similarity search in extremely high-dimensional data sets," in *Proc. 21st Intern. Conf. Data Eng.*, 2005, pp. 619–630.

[30] P. Indyk, and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proc. 30th ACM Symp. Theory Comput.*, 1998, pp. 604–613.

[31] D. R. Karger, and M. Ruhl, "Finding nearest neighbors in growth-restricted metrics," in *Proc. 34th ACM Symp. Theory Comput.*, 2002, pp. 741–750.

[32] N. Katayama and S. Satoh, "The SR-tree: An index structure for high-dimensional nearest neighbor queries," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, May 1997, pp. 369–380.

[33] R. Krauthgamer and J. R. Lee, "The black-box complexity of nearest neighbor search," in *Proc. 31st Int. Colloquium Automata, Lang. Programm.*, 2004, pp. 858–869.

[34] R. Krauthgamer and J. R. Lee, "Navigating nets: Simple algorithms for proximity search," in *Proc. 15th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2004, pp. 798–807.

[35] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[36] Y. Lifshits and S. Zhang, "Combinatorial algorithms for nearest neighbors, near-duplicates and small-world design," in *Proc. 15th Annu. ACM-SIAM Symp. Discrete Algorithm*, 2009, pp. 318–326.

[37] R. Motwani and P. Raghavan, "Randomized algorithms," *ACM Comput. Surv.*, vol. 28, no. 1, pp. 33–37, 1996.

[38] D. M. Mount and S. Arya. (2010). ANN: A library for approximate nearest neighbor searching. [Online]. Available: http://www.cs.umd.edu/ mount/ANN/

[39] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Proc. Int. Conf. Comput. Vis. Theory Appl.*, 2009, pp. 331–340.

[40] G. Navarro, "Searching in metric spaces by spatial approximation," in *Proc. String Process. Inf. Retrieval Symp. Int. Workshop Groupware*, 1999, p.141.

[41] V. Pestov, "On the geometry of similarity search: Dimensionality curse and concentration of measure," *Inf. Process. Lett.*, vol. 73, nos. 1/2, pp. 47–51, 2000.

[42] W. Pugh, "Skip lists: A probabilistic alternative to balanced trees," *Commun. ACM*, vol. 33, no. 6, pp. 668–676, 1990.

[43] Reuters Ltd. Reuters corpus, volume 2, multilingual corpus. [Online]. Available: http://trec.nist.gov/data/reuters/reuters.html, 2005.

[44] H. Samet, *Foundations of Multidimensional and Metric Data Structures*. San Francisco, CA, USA: Morgan Kaufmann, 2006.

[45] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl, "Application of dimensionality reduction in recommender system—A case study," in *Proc. ACM WebKDD Workshop*, 2000, pp. 1–12.

[46] A. Slivkins, "Distance estimation and object location via rings of neighbors," in *Proc. 24th Annu. ACM Symp. Principles Distrib. Comput.*, 2005, pp. 41–50.

[47] C. J. Stone, "Consistent parametric regression," *Ann. Statist.*, vol. 5, no. 4, pp. 595–645, 1977.

[48] D. Tschopp, S. N. Diggavi, P. Delgosha, and S. Mohajer, "Randomized algorithms for comparison-based search," in *Proc. Adv. Neural Inf. Process. Syst. 24*, 2011, pp. 2231–2239.

[49] R. Weber, H.-J. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," in *Proc. 24rd Int. Conf. Very Large Data Bases*, 1998, pp. 194–205.

[50] N. Ye, *The Handbook of Data Mining*. Mahwah, NJ, USA: Lawrence Erlbaum, 2003.

Michael E. Houle received the PhD degree from McGill University in the area of computational geometry in 1989. Since then, he developed research interests in algorithmics, data structures, and relational visualization, first as a research associate at Kyushu University and the University of Tokyo in Japan, and from 1992 at the University of Newcastle and the University of Sydney in Australia. From 2001 to 2004, he was a visiting scientist at IBM Japan's Tokyo Research Laboratory, where he first began working on approximate similarity search and shared-neighbor clustering methods for data mining applications. Since then, his research interests have expanded to include dimensionality and scalability in the context of fundamental data mining tasks such as search, clustering, classification, and outlier detection. Currently, he is a visiting professor at the National Institute of Informatics (NII), Japan.

Michael Nett received the diploma in computer science from RWTH Aachen University in 2011, and the PhD degree from the Graduate School of Information Sciences and Technology, University of Tokyo in 2014. From 2011 to 2014, he served as a research assistant at both the University of Tokyo and the National Institute of Informatics, where his research focused on the tractability of data mining and machine learning tasks with regard to models of intrinsic dimensionality. He currently occupies an engineering position at Google Japan.