Counterexamples expose gaps in the proof of time complexity for cover trees introduced in 2006

Yury Elkin*
Department of Computer Science, University of Liverpool

Vitaliy Kurlin†
Department of Computer Science, University of Liverpool.

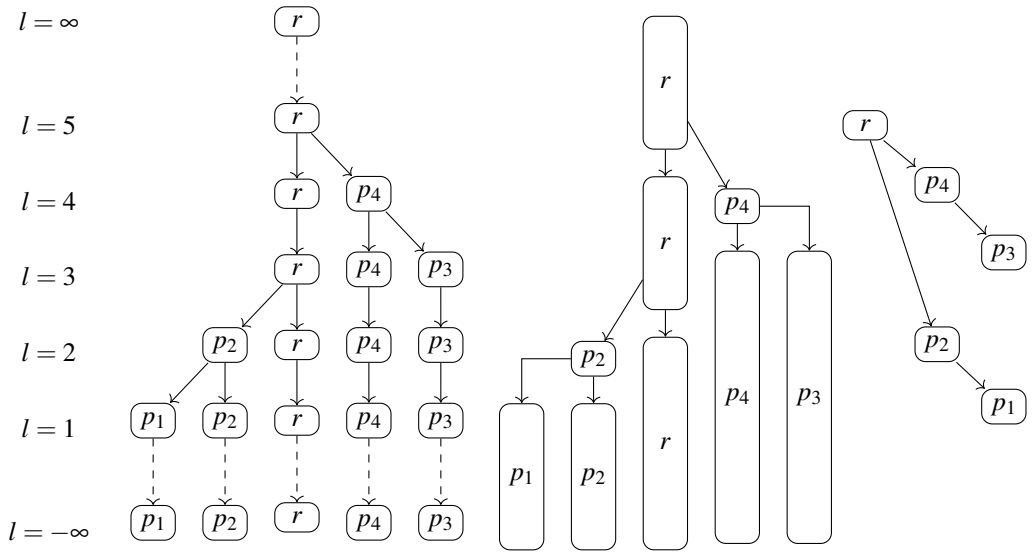

Figure 1: A comparison of different cover trees built on datasets from Example 2.4. **Left:** an implicit cover tree introduced in 2006 contains infinite repetitions of given points, see Definition 2.1. **Middle:** an explicit cover tree still includes repeated points, see Definition 2.2. **Right:** a new compressed cover tree is much smaller and includes each point only once, see [18, Definition 3.5].

ABSTRACT

This paper is motivated by the k-nearest neighbors search: given an arbitrary metric space, and its finite subsets (a reference set R and a query set Q), design a fast algorithm to find all k-nearest neighbors in R for every point $q \in Q$. In 2006, Beygelzimer, Kakade, and Langford introduced cover trees to justify a near-linear time complexity for the neighbor search in the sizes of Q, R.

Section 5.3 of Curtin's PhD (2015) pointed out that the proof of this result was wrong. The key step in the original proof attempted to show that the number of iterations can be estimated by multiplying the length of the longest root-to-leaf path in a cover tree by a constant factor. However, this estimate can miss many potential nodes in several branches of a cover tree, that should be considered during the neighbor search. The same argument was unfortunately repeated in several subsequent papers using cover trees from 2006.

This paper explicitly constructs challenging datasets that provide counterexamples to the past proofs of time complexity for the cover tree construction, the k-nearest neighbor search presented at ICML 2006, and the dual-tree search algorithm published in NIPS 2009.

The corrected near-linear time complexities with extra parameters are proved in another forthcoming paper by using a new compressed cover tree simplifying the original tree structure.

*e-mail: yura.elkin@gmail.com
†e-mail: vitaliy.kurlin@gmail.com

1 INTRODUCTION: NEIGHBOR PROBLEM AND PAST WORK

The search for nearest neighbors was one of the first data-driven problems and led to the neighbor rule for classification [13].

In a modern formulation, the problem is to find all nearest neighbors in a reference set R for all points from a query set Q. Both sets live in an ambient space X with a distance d satisfying all metric axioms. The simplest example is $X = \mathbb{R}^n$ with the Euclidean metric, where a query set Q can be a single point or a subset of a larger set R.

Definition 1.1 (k nearest neighbors)**.** For any fixed point $q \in Q$, let $d_1 \le \cdots \le d_{|R|}$ be ordered distances from q to all points of R, where $|R|$ is the number of points in R. For any $k \ge 1$, the k-nearest neighbor sets $\mathrm{NN}_k(q;R)$ consists of all points $u \in R$ with $d(q,u) \le d_k$. ■

For $Q = R = \{0,1,2,3\}$, the point $q = 1$ has ordered distances $d_1 = 0 < d_2 = 1 = d_3 < d_4 = 2$. The nearest neighbor sets are $\mathrm{NN}_1(1;R) = \{1\}, \mathrm{NN}_2(1;R) = \{0,1,2\} = \mathrm{NN}_3(1;R), \mathrm{NN}_4(1;R) = R$. So 0 can be a 2nd neighbor of 1, then 2 becomes a 3rd neighbor of 1, or these neighbors of 0 can be found in a different order.

Problem 1.2 (all nearest neighbors search)**.** Let Q, R be finite subsets of query and reference points in a metric space (X,d). For any fixed $k \ge 1$, design an algorithm to exactly find k distinct points from $\mathrm{NN}_k(q;R)$ for all $q \in Q$ so that the parametrized worst-case time complexity is near-linear in $\max\{|Q|,|R|\}$, where hidden constants may depend on structures of Q, R but not on their sizes $|Q|, |R|$. ■

Spacial data structures. It is well known that the time complexity of a brute-force approach of finding all 1st nearest neighbors of points from Q within R is proportional to the product $|Q| \cdot |R|$ of the sizes of Q, R. Already by the mid of 1970s real data was big enough to motivate faster algorithms and sophisticated data structures.

Table 1: Results for building structures in terms of the expansion constant $c(R)$ in Definition 1.3 or KR-type constant $2^{\dim_{KR}}$ in [24, Section 2.1]

Data structure, reference	time complexity	space	proofs
Navigating nets [24]	$O\big(2^{O(\dim_{KR})} \cdot \lvert R\rvert \log(\lvert R\rvert)\log(\log\lvert R\rvert)\big)$, [24, Theorem 2.6]	$O(2^{O(\dim)}\lvert R\rvert)$	Not available
Cover tree [9]	$O(c(R)^{O(1)} \cdot \lvert R\rvert \cdot \log\lvert R\rvert)$, [9, Theorem 6]	$O(\lvert R\rvert)$	Counterexample 4.2 shows that the past proof is incorrect
Compressed cover tree [18]	$O\big(c(R)^{O(1)} \cdot \lvert R\rvert \cdot \log\lvert R\rvert\big)$	$O(\lvert R\rvert)$	[18, Theorem 3.52].

Table 2: Results for finding exact all k-nearest neighbors of one query point $q \in Q$ in terms of the expansion constant $c(R)$ in Definition 1.3 or KR-type constant $2^{\dim_{KR}}$ in [24, Section 2.1], assuming that all data structures are already built.

Data structure, reference	time complexity	space	proofs
Navigating nets [24]	$O\big(2^{O(\dim_{KR})}(k+\log\lvert R\rvert)\big)$ for $k \geq 1$ [24, Theorem 2.7]	$O(2^{O(\dim)} \cdot \lvert R\rvert)$	Not available
Cover tree [9]	$O\big(c(R)^{O(1)}\log\lvert R\rvert\big)$ for $k=1$, [9, Theorem 5]	$O(\lvert R\rvert)$	Counterexample 5.2 shows that the past proof is incorrect
Compressed cover tree [18]	$O\big(c(R)^{O(1)} \cdot \log(k) \cdot (k+\log\lvert R\rvert)\big)$	$O(\lvert R\rvert)$	[18, Theorem 3.84]
Dual cover tree [38]	$O\big(c(R)^{O(1)} \cdot c(Q)^{O(1)} \cdot \max\{\lvert R\rvert, \lvert Q\rvert\}\big)$ for $k=1$ and large set Q, [38, Theorem 3.1]	$O(\lvert R\rvert)$	Counterexample 6.5 shows that the past proof is incorrect

One of the first spacial data structure, a *quadtree* [19], hierarchically indexes a reference set $R \subset \mathbb{R}^2$ by subdividing its bounding box (a root) into four smaller boxes (children), which are recursively subdivided until final boxes (leaf nodes) contain only a small number of reference points. A generalization of the quadtree to $\mathbb{R}^n$ exposes an exponential dependence of its computational complexity on n, because the n-dimensional box is subdivided into 2^n smaller boxes.

The first attempt to overcome this dimensionality curse was the *kd*-tree [7] that subdivides a subset of the reference set R at every recursion step into two subsets instead of 2^n subsets.

Then more advanced algorithms utilizing spatial data structures have positively impacted various related research areas such as a minimum spanning tree [6], range search [35], k-means clustering [35], and ray tracing [21]. The spacial data structures for finding nearest neighbors in the chronological order are k-means tree [20], R tree [5], ball tree [34], R^* tree [5], vantage-point tree [43], TV trees [29], X trees [8], principal axis tree [32], spill tree [30], cover tree [9], cosine tree [22], max-margin tree [37], cone tree [36].

Expansion constant. The past work starting from [23] expressed the time complexities of neighbor search in terms of a dimensionality constant for a finite metric space X. This constant was denoted by $2^{\dim_{KR}}$ in [24, Section 2.1] and by c in [9, Section 1]. In any metric space X, let $\bar{B}(p,t) \subseteq X$ be the closed ball with a center p and a radius t. Let $|\bar{B}(p,t)|$ be the number (if finite) of points in $\bar{B}(p,t)$.

Definition 1.3 (expansion constant $c(R)$, [18, Definition 3.4])**.** Let R be a finite set in a metric space X. The *expansion constant* $c(R)$ is the smallest $c(R) \geq 2$ such that $|\bar{B}(p,2t)| \leq c(R) \cdot |\bar{B}(p,t)|$ for any point $p \in R$ and radius $t \geq 0$.

Typically, uniformly distributed datasets have small expansion constants. Using arguments of [18, Section 4.3] it can be shown that if R is a uniformly distributed point cloud of $\mathbb{R}^m$ we have $c(R) = 2^m$. However, if a dataset contains even a single outlier, say $R = \{1,2,3,\dots,m,2m\}$, then $c(R) = |R|$.

The data structures described below were designed to justify a near-linear time complexity for finding k-nearest neighbors.

Navigating nets. In 2004 a new data structure was introduced that was a sequence of progressively finer ε-nets on the dataset R. In [24, Theorem 2.7] it was claimed that all k-nearest neighbors of a query point q are found by navigating nets in time $2^{O(\dim_{KR}(R\cup\{q\}))}(k+\log|R|)$, where $\dim_{KR}(R\cup\{q\})$ is an expansion rate of [24, Section 1.2]. All proofs and pseudocodes were omitted. The authors did not reply to our request for details.

Modified navigating nets [12] were used in 2006 to claim the worst-case time complexity $O(\log(|R|)+(1/\varepsilon)^{O(1)})$ for finding the first $(1+\varepsilon)$-approximate neighbor parameterized by a constant that depends on a doubling dimension of the ambient space. However, only sketch of proof of this result was given.

Cover trees. In 2006, [9] introduced a cover tree inspired by the navigating nets [24]. This cover tree was designed to prove a worst-case time complexity in the size $|R|$ and the expansion constant c from Definition 1.3. In particular, [9, Theorem 5] claimed that cover trees help solve Problem 1.2 for $k=1$ could be solved in $O(c^{12}\log|R|)$ time.

Past challenges. In 2015, Curtin's PhD [14, section 5.3] pointed out that the proof of [9, Theorem 5] had a mistake. It was incorrectly claimed that the number of performed iterations of the nearest neighbors algorithm [9, Algorithm 1] can be bounded by multiplying the depth of cover tree by some constant factor. This claim is false because many potential branches at different levels of a cover tree can be missed. The similar mistake was repeated in proof of time complexity method of Insert() method [9, Theorem 6], as well as in several subsequent papers: for a dual-tree based all-nearest neighbor search [38, Theorem 3.1], for a Minimum Spanning Tree [31, Theorem 5.1], for a fast exact max-kernel search [16, Lemma 5.2].

Counterexamples. To confirm the discovery of Ryan Curtin [14, section 5.3], Example 3.1 will describe a finite metric space (R,d) and its cover tree $\mathscr{C}(R)$, for which the maximal root-to-node path is bounded by $O(\sqrt{|R|})$, but that forces both Algorithm 1 and Algorithm 2 of [9] iterate over all $|R|$ levels of the cover tree $\mathscr{C}(R)$. The contradiction will follow by noting that the number of iterations in one particular example has a lower bound $|R|$ despite the claimed upper bound $O(\sqrt{|R|})$ for all datasets.

Here is the summary of the found counterexamples:

- Counterexample 4.2 to the proof of [9, Theorem 2],

- Counterexample 5.2 to the proof of [9, Theorem 1],
- Counterexample 6.5 to the proof of [38, Theorem 3.1].

Counterexamples for the time complexity of a minimum spanning tree [31, Theorem 5.1] can be found in [18, Section 4.2].

New results and compressed cover trees. All issues of past approaches are resolved in [18, Chapter 3] by defining a new data structure, a compressed cover tree [18, Definition 3.5], which combines the explicit and implicit cover tree into a single simpler structure. New near-linear time algorithms for building a compressed cover tree and for finding k-nearest neighbors are described in [18, Algorithm 3.5.3] and [18, Algorithm 3.7.2].

To overcome the past issues, we estimate the number of iterations in [18, Algorithm 3.5.4] and [18, Algorithm 3.7.2] in [18, Lemma 3.5.9] and [18, Lemma 3.7.13] , respectively. In [18, Corollary 3.5.11] it is shown that a compressed cover tree can be constructed in time $O(c(R)^{10}\log(|R|)|R|)$ and [18, Theorem 3.7.14] shows that k nearest neighbors of any point q can be found in time

$$O(c(R\cup\{q\})^3\cdot\log_2(k)\cdot(c(R\cup\{q\})^9\cdot\log_2(|R|)+k)).$$

Tables 1 and 2 summarize all known cover tree methods and their contributions for k-nearest neighborhood search into two tables.

2 Original cover trees introduced in 2006

To resolve Problem 1.2 effectively [9] introduced a new data structure, cover tree, the idea of which was to encode data of the reference set R into a leveled tree. Using this tree, a new algorithm [9, Algorithm 1] was introduced, which was used to find the nearest neighbor of a given query point q. The idea was to travel from the root node of the tree, located on the highest level towards the leaf nodes on the lowest level, memorizing the current best candidate for the nearest neighbor and eliminating the branches, which were clearly too far from the query point. Compared to the brute-force search, the benefit of this procedure is that we avoid computing the distance of a query point to a large number of points which are eliminated in large batches during the search.

Implicit and explicit cover trees are visualizations of finite metric spaces, that were discovered in [9, Section 2]. However, only the definition of the implicit cover tree was formally stated.

Definition 2.1 (Implicit cover tree $\mathscr{I}(R)$, [9, Section 2])**.** Let R be a finite set in a metric space (X,d). *An implicit cover tree* $\mathscr{I}(R)$ is a tree on a subset of $R\times\mathbb{Z}\cup\{-\infty,+\infty\}$ with a root $r\in R$ and a *level* function $l:R\to\mathbb{Z}$ satisfying the conditions below.

(2.1a) *Root condition* : The level of the root node r is $l(r)=\infty$.

(2.1b) *Node condition* : For all points $p\in R$ and for all indices $i\in(-\infty,l(p)+1)$ there exists a node (p,i) in the tree $\mathscr{I}(R)$.

(2.1c) *Covering condition* : for every node $(q,i)\in\mathscr{I}(R)$ there exists a *parent* $(p,i+1)\in\mathscr{I}(R)$ such that $d(q,p)\le 2^{i+1}$, this parent node p has a single link to its *child* node q in the tree $\mathscr{I}(R)$.

(2.1d) *Separation condition* : for $i\in\mathbb{Z}$ and the *cover set* $C_i=\{p\in R\mid l(p)\ge i\}$, the minimum inter-point distance $d_{\min}(C_i)=\min\limits_{p\in C_i}\min\limits_{q\in C_i\setminus\{p\}}d(p,q)$ is larger than 2^i.

For any node $p\in\mathscr{I}(R)$, Children(p,i) denotes the set consisting of all children of the node (p,i), including the node $(p,i-1)$ on the level below. For any node $p\in\mathscr{I}(R)$, define the *node-to-root* path as a unique sequence of nodes $w_0,\dots,w_m$ such that $w_0=p$, w_m is the root and w_{j+1} is the parent of w_j for $j=0,...,m-1$. A node $q\in\mathscr{I}(R)$ is a *descendant* of a node p if p is in the node-to-root path of q. A node p is an *ancestor* of q if q is in the node-to-root path of p. Let Descendants(p,i) be the set of all descendants of node (p,i), including $(p,i-1)$. ■

The explicit cover tree is obtained from an implicit cover tree by collapsing into a single node all nodes from any infinite non-branched path $(p,i)\to(p,i-1)\to\cdots\to(p,-\infty)$, see the left and middle pictures of Fig. 1, as formalized below.

Definition 2.2 (Explicit cover tree $\mathscr{E}(R)$, [9, Section 2])**.** Let R be a finite set in a metric space (X,d). Let $\mathscr{I}(R)$ be implicit cover tree of Definition 2.1. An *explicit cover tree* $\mathscr{E}(R)$ is a quotient tree $\mathscr{I}(R)/\backsim$, where $(p,i)\backsim(q,j)$, if $p=q$ and Children(p,t) consist of the nodes $(p,t-1)$ for all $t\in[\min(i,j)+1,\max(i,j)]$.

Since nodes containing different points are never glued together, we denote an arbitrary node of explicit cover tree $\mathscr{E}(R)$ by $(p,[i])$, where $p\in R$ is the point stored in the node and $[i]$ is equivalence class of (p,i) in $[\backsim]$.

Example 2.3 (Three point example)**.** Let $R=\{0,1,2^i\}$ for some large $i\in\mathbb{Z}_+$ and let $d(x,y)=|x-y|$ be the Euclidean metric on $\mathbb{R}$. There are multiple ways to construct an implicit cover tree $\mathscr{I}(R)$. Assume that 2^i is chosen to be the root node. Then $\mathscr{I}(R)$ will contain an infinite chain $\{(2^i,j)\mid j\in\mathbb{Z}\}$, in such a way that for all j node $(2^i,j)$ is parent of $(2^i,j-1)$.

Let us now insert points $\{0,1\}$. Since $d(2^i,1)=2^i-1$ and $d(2^i,0)=2^i$, by conditions (2.1b) and (2.1c) either $l(0)=i-1$ or $l(1)=i-1$. Let us choose $l(0)=i-1$, then $\mathscr{I}(R)$ will contain chain $\{(0,j)\mid j\in(-\infty,i-1]\cap\mathbb{Z}\}$ in its vertex set, where $(0,j)$ will be parent of $(0,j-1)$ for all $j\in(-\infty,i-1]$ and $(2^i,i)$ will be parent of $(0,i-1)$. Since $d(0,1)=1$ and point 0 minimizes the distance $d(1,\{0,2^i\})$ we have $l(1)=-1$. Therefore $\mathscr{I}(R)$ will contain chain $\{(1,j)\mid j\in(-\infty,-1]\cap\mathbb{Z}\}$ and $(1,-1)$ will be child of $(0,0)$.

The compressed representation of $\mathscr{I}(R)$ is illustrated in Figure 2 (middle). Explicit cover tree $\mathscr{E}(R)$ consists of nodes: $(2^i,[i])$, $(2^i,[i-1])$, $(0,[i-1])$, $(0,[-1])$, $(1,[-1])$, where

- $(2^i,[i])$ has two children $(2^i,[i-1])$, $(0,[i-1])$ on level $i-1$.
- $(0,[i-1])$ has two children $(0,[-1])$, $(1,[-1])$ on level -1.
- No other children are present.

Example 2.4 (a short train line tree)**.** Let G be the unoriented metric graph consisting of two vertices r,q connected by three different edges e,h,g of lengths $|e|=2^6$, $|h|=2^3$, $|g|=1$. Let p_4 be the middle point of the edge e. Let p_3 be the middle point of the subedge (p_4,q). Let p_2 be the middle point of the edge h. Let p_1 be the middle point of the subedge (p_2,q). Let $R=\{p_1,p_2,p_3,p_4,r\}$. We construct an implicit cover tree $\mathscr{I}(R)$ by choosing the level $l(p_i)=i$ and by setting the root to be r. Then $\mathscr{I}(R)$ satisfies all the conditions of Definition 2.1, see a comparison of the three cover trees in Fig. 1. ■

3 Challenging datasets for original cover trees

In this section Example 3.1 introduces a dataset R and its cover tree $\mathscr{I}(R)$, which will be used to show that key steps in proofs of time complexity estimates of cover tree construction algorithm [9, Theorem 5] and the nearest neighbor search algorithm [9, Theorem 6] are incorrect. Since the same false arguments were later repeated in the papers [31] and [38], we provide a detailed counterexamples in Sections 4 and 5 that expose the contradiction within each of the proof of the theorems.

Example 3.1 (tall imbalanced tree)**.** For any integer $m>10$, let G be a metric graph pictured in Figure 3 that has two vertices r,q and $m+1$ edges (e_i) for $i\in\{0,...,m\}$, and the length of each edge e_i is $|e_i|=2^{m\cdot i+2}$ for $i\ge 1$. Finally, set $|e_0|=1$. For every $i\in\{1,...,m^2\}$ if i is divisible by m we set p_i be the middle point of $e_{i/m}$ and for

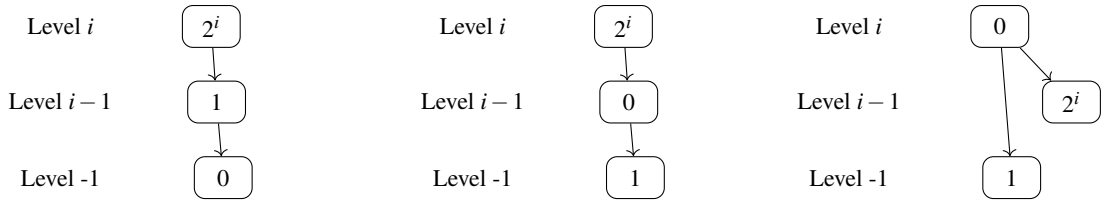

Figure 2: Compressed representations of three different implicit cover trees $\mathscr{I}(R)$ built on same set $R = \{0, 1, 2^i\}$. See Definition 2.1.

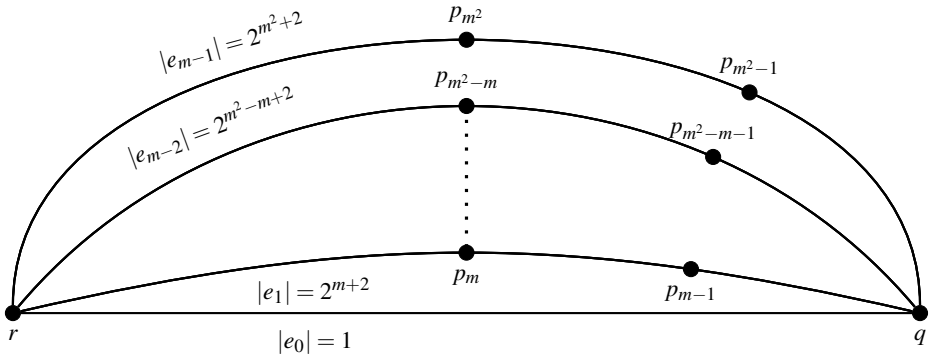

Figure 3: The graph G and the dataset R defined in Example 3.1

every other i we define p_i to be the middle point of segment (p_{i+1}, q). Let d be the induced shortest path metric on the continuous graph G. Then $d(q,r) = 1$, $d(r,p_i) = 2^{i+1}+1$, $d(q,p_i) = 2^i$. If $i > j$ and $\lceil \frac{i}{m} \rceil = \lceil \frac{j}{m} \rceil$, then

$$d(p_j, p_i) = \sum_{t=j+1}^{i} 2^t.$$

We consider the reference set $R = \{r\} \cup \{p_i \mid i = 1,2,3,...,m^2\}$ with the metric d.

Let us define an implicit cover tree $\mathscr{I}(R)$ by setting r to be the root node and $l(p_i) = i$ for all i. For all $i \in 1,...,m^2$: If i is divisible by m, we set $(r, i+1)$ to be the parent of (p_i, i). If i is not divisible by m, we set $(p_{i+1}, i+1)$ to be the parent of (p_i, i). For every i divisible by m, the point p_i is in the middle of edge $e_{i/m}$, hence $d(p_i, r) \le 2^{i+1}$. For every i not divisible by m, by definition, p_i is the middle point of (p_{i+1}, q). Therefore, we have $d(p_i, p_{i+1}) \le 2^{i+1}$. Since for any point p_i distance to its parent is at most 2^{i+1}, the tree $\mathscr{I}(R)$ satisfies covering condition (2.1b). For any integer t, the cover set is $C_t = \{r\} \cup \{p_i \mid i \ge t\}$. We will prove that C_t satisfies (2.1c). Let $p_i \in C_t$. If i is divisible by m, then $d(r, p_i) = 2^{i+1} \ge 2^{t+1} > 2^t$. If i is not divisible by m, then $d(r, p_i) = d(r,q) + d(q, p_i) = 1 + 2^{i+1} > 2^t$. Then the root r is separated from the other points by the distance 2^t. Consider arbitrary points p_i and p_j with indices $i > j \ge t$ and $\lceil \frac{i}{m} \rceil = \lceil \frac{j}{m} \rceil$. Then

$$d(p_i, p_j) = \sum_{s=j+1}^{i} 2^s \ge 2^{j+1} \ge 2^{t+1} > 2^t.$$

On the other hand, if $i > j \ge t$ and $\lceil \frac{i}{m} \rceil \ne \lceil \frac{j}{m} \rceil$, then

$$d(p_i, p_j) = d(p_i, q) + d(p_j, q) \ge 2^i + 2^j \ge 2^{j+1} \ge 2^{t+1} > 2^t.$$

For any t, we have shown that all pairwise combinations of points of C_t satisfy condition (2.1c). Hence this condition holds for the whole tree $\mathscr{I}(R)$. ■

Let us now define the explicit depth, that corresponds to maximal root-to-node path of any cover tree. By Definition 2.2 an explicit cover tree $\mathscr{E}(R)$ is a quotient of $\mathscr{I}(R)/\backsim$, where we collapse all the chains having only a single self-child into a single node. Nodes of $\mathscr{I}(R)/\backsim$ are denoted by $(p, [i])$, where $[i]$ is the equivalence class of integer i in $\backsim$. By [9, Lemma 4.3] the depth of any node $(p, [i])$ is "defined as the number of explicit grandparent nodes on the path from the root to p in the lowest level in which p is explicit". The explicit depth of a node $(p, [i])$ in any explicit cover tree $\mathscr{E}$ is introduced in Definition 3.2 below using the most natural interpretation of the aforementioned quotes.

Definition 3.2 (Explicit depth for explicit cover tree)**.** Let R be a finite subset of a metric space with a metric d. Let $\mathscr{E}(R)$ be an explicit cover tree on R. For any $(p, [j]) \in \mathscr{E}(R)$, let $s = (w_0, ..., w_m)$ be a node-to-root path of $(p, [j])$, where $w_0 = (r, [+\infty])$ and $w_m = (p, [j])$. We define $D(p, [j])$ to be the number of nodes $|s|$ in the path s. The *explicit depth* of a cover tree is defined as the size of maximal node-to-root path

$$D(\mathscr{E}(R)) = \max_{(p,[j]) \in \mathscr{E}(R)} D(p, [j]).$$

■

Lemma 3.3 shows that the cover tree of Example 3.1 the maximal node-to-root has $2 \cdot \sqrt{|R|}$ size, where $|R|$ is the size of dataset.

Lemma 3.3. *Let $\mathscr{E}(R)$ be a compressed cover tree on the set R from Example 3.1 for some $m \in \mathbb{Z}$. The explicit depth $D(\mathscr{E}(R))$ of Definition 3.2 has the upper bound $2m+1$.* ■

Proof. Note first that root node r contains exactly m non-trivial children. Consider arbitrary node p_i. If i is divisible by m, then r is the parent of p_i. It follows that we can reach root node $(r, [+\infty])$ in at most $m+1$ steps from p_i.

Let us now consider an index i that is not divisible by m. Note that $(p_{j+1}, [j+1])$ is the parent of $(p_j, [j])$ for all $j \in [i, m \cdot \lceil i/m \rceil -$

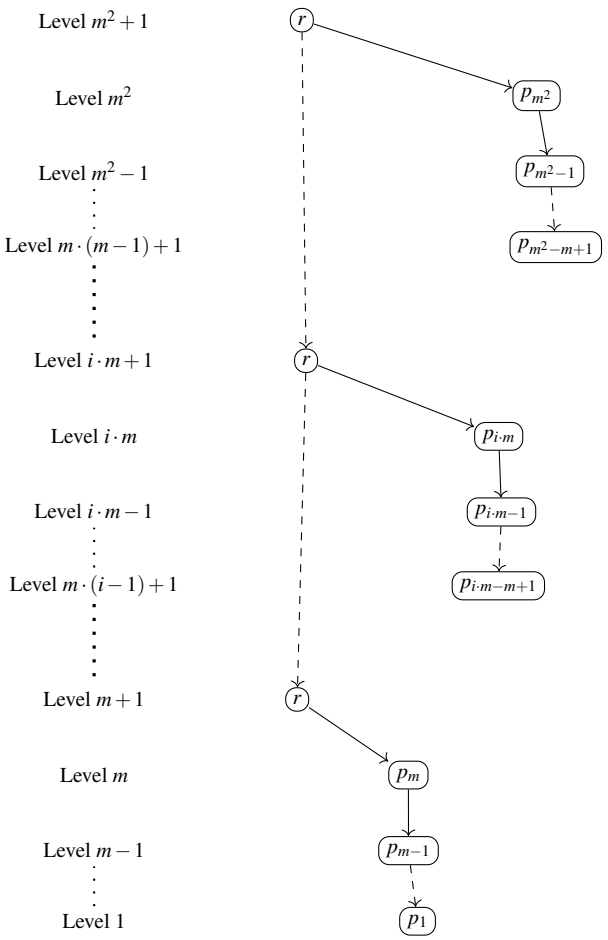

Figure 4: An explicit cover tree built on the dataset R in Example 3.1. The node r at the level $i\cdot m+1$ corresponds to the node $(r,[i\cdot m+1])$ in the explicit cover tree $\mathscr{E}(R)$.

1]. Then the path s consisting of all ancestors of p_i from p_i to $p_{m\cdot\lceil i/m\rceil}$ has the form $((p_i,[i]),(p_{i+1},[i+1]),...,(p_{m\cdot\lceil i/m\rceil},[m\cdot\lceil i/m\rceil]))$. Note that $|s|\le m$. Since $m\cdot\lceil i/m\rceil$ is divisible by m, by the first paragraph the node to root path l from $(p_{m\cdot\lceil i/m\rceil},[m\cdot\lceil i/m\rceil])$ to $(r,[+\infty])$ takes at most $m+1$ steps. Therefore

$$D(p_i,[i])\le|s\cup l|\le|s|+|l|\le(m+1)+m\le 2m+1,$$

which proves the claim. □

4 COVER TREE CONSTRUCTION

Counterexample 4.2 shows that the proof of worst-case time complexity of the Insert() operation for an implicit cover tree [9, Theorem 6] is incorrect. A correct time complexity for a new compressed cover tree is given in [18, Corollary 3.53].

Counterexample 4.2 (for a step in the proof of [9, Theorem 6])**.** The idea is based on adding a new point q of Figure 3 to the tree $\mathscr{E}(R)$ of Example 3.1 that lures the Algorithm 4.1 into using all branches of $\mathscr{E}(R)$. It follows that the Algorithm 4.1 is launched $O(|R|)$ times. However, in the proof of [9, Theorem 6] it was claimed that Algorithm 4.1 is launched at most $4\cdot D(\mathscr{E}(R))$, where $D(\mathscr{E}(R))$ is the explicit depth of explicit cover tree $\mathscr{E}(R)$. This is a contradiction, since $D(\mathscr{E}(R))\le 2\sqrt{|R|}+1$ but Algorithm 4.1 runs $O(|R|)$ times.

1: **Insert**(point p, cover set Q_i, level i)
2: Set $Q=\{\text{Children}(q)\mid q\in Q_i\}$
3: **if** $d(p,Q)>2^i$ **then**
4: **return** "no parent found"
5: **else**
6: Set $Q_{i-1}=\{q\in Q\mid d(p,q)\le 2^i\}$
7: **if Insert**$(p,Q_{i-1},i-1)$ = "no parent found" and $d(p,Q_i)\le 2^i$ **then**
8: Pick $q\in Q_i$ satisfying $d(p,q)\le 2^i$ and insert p into Children(q), **return** "parent found"
9: **else**
10: **return** "no parent found"
11: **end if**
12: **end if**

Algorithm 4.1: Copy-pasted Insert() algorithm for inserting a point p into an implicit cover tree T [9, Algorithm 2]. This algorithm is launched with $i=l_{\max}$ and $Q_i=\{r\}$, where r is the root node of T.

For more details, we cite a part of the proof of [9, Theorem 6]:

"**Theorem 6** Any insertion or removal takes time at most $O(c^6\log(n))$" [In other words the run time of Algorithm 4.1 is $O(c^6\log(n))$, where n is the number points of original dataset S on which tree T was constructed.]

[*Partial proof:*]: *" Let $k=c^2\log(|S|)$ be the maximum explicit depth of any point, given by Lemma 4.3. Then the total number of cover sets with explicit nodes is at most $3k+k=4k$, where the first term follows from the fact that any node that is not removed must be explicit at least once every three iterations, and the additional k accounts for a single point that may be implicit for many iterations. Thus the total amount of work in Steps 1 [Our line 2] and 2 [Our lines 3-5] is proportional to $O(k\cdot\max_i|Q_i|)$. Step 3 [Our lines 5-11] requires work no greater than step 1 [Our line 2]."*

In our interpretation the above arguments says that the total number of times line 1 **[our line 2] was called during the algorithm has the upper bound** $4\cdot D(\mathscr{E}(R))$ **, where** $D(\mathscr{E}(R))$ **is the explicit depth of** $\mathscr{E}(R)$**, see Definition 3.2. In this Counterexample we will show that** $\mathscr{E}(R)$ **from Example 3.1 does not satisfy the claimed inequality.**

Take the reference set R, the compressed cover tree $\mathscr{E}(R)$ and the point q from Example 3.1 for any parameter $m>200$. Assume that we have already constructed tree $\mathscr{E}(R)$. Let us show that $\mathscr{E}(R\cup q)$ constructed by Algorithm 4.1 from the input $q, i=m^2+1, Q_i=\{(r,[+\infty])\}$ runs at least m^2-2 self-recursions. This will lead to a contradiction since by Lemma 3.3 we have $D(\mathscr{E}(R))\le 2m+1$.

We show by induction on m going down that, for every step $i\in[1,m^2]$, we have $Q_i=\{(r,[i]),(p_i,[i])\}$. The proof for the base case $i=m^2$ is similar to the induction step and thus will be omitted. Assume that Q_i has the desired form for some i. Let us show that the claim holds for $i-1$. For all levels $i-1$ divisible by m, the node $(p_{i-1},[i-1])$ is a child of node $(r,[i])$. For all levels $i-1$ not divisible by m, the node $(p_{i-1},[i-1])$ is a child of p_i. Since $\mathscr{E}(R)$ contains exactly one node at each level, in both cases we have $Q=\{(r,[i]),(p_i,[i]),(p_{i-1},[i-1])\}$. Since $d(q,r)=1$, $d(q,p_i)=2^{i+1}$ and $d(q,p_{i-1})=2^i$ we have

$$Q_{i-1}=\{p\in Q_i\mid d(p,q)\le 2^i\}=\{(r,[i-1]),(p_{i-1},[i-1])\}.$$

The actual implementation of algorithm 4.1 iterates over all levels

i for which there exists a node in Q_i that contains at least one non-trivial child on level $i-1$ and for which the condition in line 7 is satisfied. Since for every index $i\in[2,m^2+1]$ we have $Q_i=$

$\{(r,[i]),(p_i,[i])\}$ and since either r or p_i has a child at level $i-1$ and the condition in line 7 is always satisfied, it follows that m^2-2 is a low bound for the number ξ of self-recursions. Therefore the contradiction follows from the inequality:

$$m^2-2 \le \xi \le 4\cdot D(\mathscr{E}(R)) \le 8\cdot(2m+1) \le 16\cdot m+8$$

where $m>20$. ■

5 NEAREST NEIGHBOR SEARCH

Counterexample 5.2 shows that the proof of [9, Theorem 5], which gives an upper bound for the complexity of Algorithm 5.1 is incorrect. A correct time complexity estimate for a new k-nn algorithm using compressed cover tree is given in [18, Corollary 3.84].

1: **Input** : implicit cover tree T, a query point p
2: Set $Q_\infty = C_\infty$ where C_∞ is the root level of T
3: **for** i from ∞ down to $-\infty$ **do**
4: Set $Q = \{\text{Children}(q) \mid q \in Q_i\}$.
5: Form cover set $Q_{i-1} = \{q \in Q \mid d(p,q) \le d(p,Q)+2^i\}$
6: **end for**
7: **return** $\text{argmin}_{q\in Q_{-\infty}} d(p,q)$

Algorithm 5.1: Copy-pasted [9, Algorithm 1] based on an implicit cover tree T [9, Section 2] for nearest neighbor search, which is used in Counterexample 5.2. The children of a node q of an implicit cover tree are defined as the nodes at one level below q that have q as their parent. In the actual implementation the loop in lines 3-6 runs only for the levels containing nodes with non-trivial children (not coinciding with their parents).

Counterexample 5.2 (for a step in the proof of [9, Theorem 5])**.** Counterexample 5.2 shows that there is a gap in proof of [9, Theorem 6]. The counterexample is obtained by running Algorithm 5.1 for node q of Figure 3 and tree $\mathscr{E}(R)$ of Example 3.1. It it shown that Algorithm 5.1 iterates over all branches of $\mathscr{E}(R)$, therefore lines 3-6 are considered exactly $|R|$ times. However, the proof of [9, Theorem 6] claimed that the number of times lines 3-6 are considered is bounded by multiplication $\max_i|R_i|\cdot D(\mathscr{E}(R))$, where $D(\mathscr{E}(R))$ is the maximal path-to-root path that has an upper bound $2\sqrt{|R|}$. In this counterexample it will be also shown that $\max_i|R_i| \le 3$ during the whole iteration of the algorithm, which will lead to contradiction $|R| \le 3\cdot 2\sqrt{|R|}$, when $|R|$ is sufficiently big.

For more detailed exhibition let us first cite a part of the proof of [9, Theorem 5].

"**Theorem 5** If the dataset $S\cup\{p\}$ has expansion constant c, the nearest neighbor of p can be found in time $O(c^{12}\log(n))$."

*[*Partial proof:*] "Let Q^* be the last Q considered by the Algorithm 5.1 (so Q^* consists only of lead nodes with scale $-\infty$). Lemma 4.3 bounds the explicit depth of any node in the tree (and in particular any node in Q^*) by $k = O(c^2\log(N))$. Consequently the number of iterations is at most $k|Q^*| \le k\max_i|Q_i|$."*

By our interpretation the above argument claims that the total number ξ of times when Algorithm 5.1 runs lines 3-6 has an upper bound $\xi \le D(\mathscr{E}(R))\cdot\max_i|Q_i|$. Contradiction will be obtained by showing that $\mathscr{E}(R)$ from Example 3.1 does not satisfy this inequality.

Take $R,\mathscr{E}(R)$ and q from Example 3.1. We will apply Algorithm 5.1 to the tree $\mathscr{E}(R)$ and query point q. By Lemma 3.3 the cover tree $\mathscr{E}(R)$ having parameter m has $D(p) \le 2m+1$ for all $p\in R$. A contradiction to the original argument will follow after showing that $\max|Q_i| \le 2$ and $\xi \ge m^2-2$.

Let us first estimate $\max_i|Q_i|$. Similarly to Counterexample 4.2 we will show that, for every iteration $i\in[1,m^2]$ of lines 3-5 of Algorithm 5.1, we have $Q_i = \{(r,[i]),(p_i,[i])\}$. The proof for the basecase $i=m^2$ is similar to the induction step and thus will be omitted. Assume that Q_i has the desired form for some i. Let us show that the claim holds for $i-1$. For all levels $i-1$ divisible by m, the node $(p_{i-1},[i-1]$ is a child of the root $(r,[i])$. For all levels $i-1$ not divisible by m, the node $(p_{i-1},[i-1])$ is a child of $(p_i,[i])$. Since $\mathscr{E}(R)$ contains exactly one node at each level, in both cases we have $Q = \{(r,[i-1]),(p_i,[i-1]).(p_{i-1},[i-1])\}$. Since $d(q,r)=1$, $d(q,p_i)=2^{i+1}$ and $d(q,p_{i-1})=2^i$, we have

$$Q_{i-1} = \{p\in Q_t \mid d(p,q)\le 2^i+1\} = \{(r,[i-1]),(p_{i-1},[i-1])\}$$

Therefore it follows that $|Q_i|\le 2$ for all $i\in[1,m^2]$.

The actual implementation of algorithm 5.1iterates over all levels i for which there exists a node in Q_i containing at least one non-trivial child at level $i-1$. Since $Q_i=\{r,p_i\}$ and for every index $i\in[2,m^2+1]$, either $(r,[i])$ or $(p_i,[i])$ has a child on level $i-1$, it follows that m^2-2 is a low bound for the number ξ of iterations. A contradiction follows from

$$m^2-2 \le \xi \le D(\mathscr{E}(R))\cdot\max_i|Q_i| \le (2m+1)\cdot 2 \le 4m+2,$$

for any $m>20$.

■

6 CHALLENGES OF THE NEAREST NEIGHBOR SEARCH BASED ON PAIRED TREES

In 2009 [38, Theorem 3.1] revisited the time complexity for all 1st nearest neighbors and claimed the upper bound $O(c(R)^{12}c(Q)^{4\kappa}\max\{|Q|,|R|\})$, where $c(Q),c(R)$ are expansion constants of the query set Q and reference set R. The degree of bichromaticity κ is a parameter of both sets Q,R, see [38, Definition 3.1]. We have found the following issues.

First, Counterexample 6.2 shows that [38, Algorithm 1] for $Q=R$ returns for any query point $q\in Q$ the same point q as its first neighbor. Second, Remark 6.4 explains several possible interpretations of [38, Definition 3.1] for the parameter κ. Third, [38, Theorem 3.1] similarly to [9, Theorem 5] relied on the same estimate of recursions in the proof of [9, Lemma 4.3]. Counterexample 6.5 explains step-by-step why the proof of the time complexity result of [38, Algorithm 1] is incorrect and requires a clearer definition of κ.

In 2015 Curtin with the authors above [15] introduced other parameters: the imbalance I_t in [15, Definition 3] and θ in [15, Definition 4]. These parameters measured extra recursions that occurred due to possible imbalances in trees built on Q,R, which was missed in the past. [15, Theorem 2] shows that, for constructed cover trees on a query set Q and a reference set R, Problem 1.2 for $k=1$ (only 1st nearest neighbors) can be solved in time

$$O\Big(c^{O(1)}\big(|R|+|Q|+I_t+\theta\big)\Big). \qquad (*),$$

where c is expansion constant that depends on Q and R. The problem with this approach is that in worst case I_t is quadratic $O(|R|^2)$. To make the time complexity linear, we would have to show $I_t = O(c^{O(1)}\cdot\max\{|R|,|Q|\})$. However, no such result exist at the moment.

The step-by-step execution of Algorithm 6.1 will show that the number of reference expansions has a lower bound $O(\max\{|Q|,|R|\}^2)$. Recall that [38, End of Section 1] defined the all-nearest-neighbor problem as follows. "**All Nearest-neighbors:** For all queries $q\in Q$

1: **Function** FindAllNN(a node $q_j \in T(Q)$, a subset R_i of a cover set C_i of $T(R)$).
2: **if** $i = -\infty$ **then**
3: for each $q_j \in L(q_j)$ **return** $\text{argmin}_{r \in R_{-\infty}} d(q,r)$
4: {here $L(q_j)$ is the set of all descendants of the node q_j}
5: **else if** $j < i$ **then**
6: $\mathscr{C}(R_i) = \{\text{Children}(r) \mid r \in R_i\}$
7: $R_{i-1} = \{r \in R \mid d(q_j,r) \le d(q_j,R) + 2^i + 2^{j+2}\}$
8: FindAllNN(q_{j-1}, R_i) { q_{j-1} is the same point as q_j on one level below}
9: **else**
10: for each $p_{j-1} \in \text{Children}(q_j)$ FindAllNN(p_{j-1}, R_i)
11: **end if**

Algorithm 6.1: Copy-pasted [38, Algorithm 1] is analyzed in Counterexamples 6.2 and 6.5.

find $r^*(q) \in R$ such that $r^*(q) = \text{argmin}_{r \in R} d(q,r)$". For $Q = R$, the last formula produces trivial self-neighbors.

In original Algorithm 6.1, the node q_j has a level j, a reference subset $R_i \subset R$ is a subset of C_i for an explicit cover tree $\mathscr{E}(R)$. The algorithm is called for a pair $q_j, R_i = \{(r, [+\infty])\}$, where q_j is the root of the query tree at the maximal level $j = +\infty$, and r is the root of the reference tree at the maximal level $i = +\infty$.

Split Algorithm 6.1 into these blocks: **lines 2-4** : FinalCandidates, **lines 5-9** : reference expansion, **lines 9-11** : query expansion.

Counterexample 6.2. In the notations of Example 3.1, m is a parameter of R. Build a compressed cover tree $\mathscr{E}(R)$ as in Figure 4. Set $Q = R$. First we show that Algorithm 6.1 returns the trivial neighbor when $\mathscr{E}(Q) = \mathscr{E}(R)$. We start the simulation with the query node r on the level m^2+1, which has the reference subset $R_{m^2+1} = \{(r, [m^2+1])\}$. The query node and the reference set are at the same levels, so we run the query expansions (lines 9-11). The node r has p_{m^2} and r as its children. Hence the algorithm goes into the branches that have p_{m^2} as the query node and into the branches that have r as the query node. Let us focus on all recursions having p_{m^2} as the query node. In the first recursion involving the node p_{m^2}, we have $i = m^2+1, j = m^2$. Thus $j < i$ and we run reference expansions (lines 5-9). The node $(r, [m^2+1])$ has two children at the level m^2, so $\mathscr{C}(R_i) = \{(p_{m^2}, [m^2]), (r, [m^2])\}$. Since $d(p_{m^2}, p_{m^2}) = 0$ and $d(p_{m^2}, r) = 2^{m^2+1}$ on line 7, we have:

$$\begin{aligned} R_{m^2} &= \{r \in \mathscr{C}(R_i) \mid d(q_j,r) \le 2^{m^2+1} + 2^{m^2+2}\} \\ &= \{(p_{m^2}, [m^2]), (r, [m^2])\} \end{aligned}$$

Similarly, for $i = m^2, j = m^2-1, q_j = p_{m^2}$, we have

$$\mathscr{C}(R_i) = \{(p_{m^2}, [m^2]), (p_{m^2-1}, [m^2-1]), (r, [m^2+1])\}$$

and since $d(p_{m^2-1}, p_{m^2}) = 2^{m^2}$ and $d(r, p_{m^2}) = 2^{m^2+1}$ we have:

$$\begin{aligned} R_{m^2-1} &= \{r \in \mathscr{C}(R_i) \mid d(q_j,r) \le 2^{m^2} + 2^{m^2+1}\} \\ &= \{(p_{m^2}, [m^2-1]), (p_{m^2-1}, [m^2-1])\}. \end{aligned}$$

For $i = m^2-1, j = m^2-2, q_j = p_{m^2}$, we have

$$\mathscr{C}(R_i) = \{(p_{m^2}, [m^2-2]), (p_{m^2-1}, [m^2-2]), (p_{m^2-2}, [m^2-2])\}.$$

Since $d(p_{m^2-1}, p_{m^2}) = 2^{m^2}$ and $d(p_{m^2-2}, p_{m^2}) = 2^{m^2} + 2^{m^2-1}$, we have:

$$\begin{aligned} R_{m^2-2} &= \{r \in \mathscr{C}(R_i) \mid d(q_j,r) \le 2^{m^2-1} + 2^{m^2}\} \\ &= \{(p_{m^2}, [m^2]), (p_{m^2-1}, [m^2-1]), (p_{m^2-2}, [m^2-2])\}. \end{aligned}$$

Finally, for $i = m^2-2, j = m^2-3, q_j = p_{m^2}$, we have

$$\mathscr{C}(R_i) = \{(p_{m^2}, [m^2-3]), ..., (p_{m^3-3}, [m^2-3])\}$$

and $d(p_{m^2}, p_{m^3-3}) = 2^{m^2} + 2^{m^2-1} + 2^{m^2-2}$. The previous inequalities imply that

$$R_{m^2-3} = \{r \in \mathscr{C}(R_i) \mid d(q_j,r) \le 2^{m^2-2} + 2^{m^2-1}\} = \{(p_{m^2}, [m^2-3])\}.$$

Since $R_{m^2-3} = \{p_{m^2}\}$, the nearest neighbor of p_{m^2} will be chosen to be p_{m^2}. The same argument can be repeated for all $p_t \in R$. It follows that Algorithm 6.1 finds trivial nearest neighbor for every point $p_t \in R$. ■

Example 6.3. To avoid the issue of finding trivial nearest neighbors as in Counterexample 6.2, we will modify Example 3.1. For any integer $m > 100$, let G be a metric graph that has 2 vertices r and q and $2m-1$ edges $\{e_0\} \cup \{e_1, ..., e_{m-1}, h_1, ..., h_{m-1}\}$. The edge-lengths are $|e_i| = 2^{i \cdot m+2}$ and $|h_i| = 2^{i \cdot m+2}$ for all $i \in [1,m]$, finally $|e_0| = 1$.

For every $i \in \{1, ..., m^2\}$, if i is divisible by m, we set q_i to be the middle point of $e_{i/m}$ and r_i to be the middle point of $h_{i/m}$. For every other i not divisible by m, we define q_i to be the middle point of segment (q_{i+1}, q) and r_i to be the middle point of segment (r_{i+1}, q).

Let d be the shortest path metric on the graph G. Then $d(q_i, r) = d(q_i, q) + 1 = 2^{i+1} + 1$, $d(q_i, r_j) = 2^{i+1} + 2^{j+1}$ and $d(q,r) = 1$. Let $R = \{r, r_{m^2}, r_{m^2-1}, ..., r_1\}$ and let $Q = \{r, q_{m^2}, q_{m^2-1}, ..., q_1\}$. Let compressed cover trees $\mathscr{E}(Q), \mathscr{E}(R)$ have the same structure as the compressed cover tree $\mathscr{E}(R)$ in Example 3.1. ■

Remark 6.4. [38, Definition 3.1] introduced the degree of bichromaticity κ as follows. "**Definition 3.1** Let S and T be cover trees built on query set Q and reference set R respectively. Consider a dual-tree algorithm with the property that the scales of S and T are kept as close as possible – i.e. the tree with the larger scale is always descended. Then, the degree of bichromaticity κ of the query-reference pair (Q,R) is the maximum number of descends in S between any two descends in T". There are at least two different interpretations of this definition. Our best interpretation is that κ is the maximal number of levels in T containing at least one node between any two consecutive levels of S. However, if q is a leaf node of S, but there are still many levels between level of q and $l_{\min}(T)$, it is not clear from the definition if κ includes these levels. [15, page 3284] pointed out that " Our results are similar to that of Ram et al. (2009a), but those results depend on a quantity called the constant of bichromaticity, denoted κ, which has unclear relation to cover tree imbalance. The dependence on κ is given as $c_q^{4\kappa}$, which is not a good bound, especially because κ may be much greater than 1 in the bichromatic case (where $S_q = S_r$)".

To keep track of the indices i, j the function call FindAllNN(q_j, R_i) will be expressed as FindAllNN(i, j, q_j, R_i) in Counterexample 6.2.

Counterexample 6.5 (for a step in the proof of [38, Theorem 3.1])**.** We will now show that in addition to the problems in the pseudocode the proof of [38, Theorem 3.1] is incorrect. Let us consider the following quote from its proof.

*"**Theorem 3.1** Given a reference set R of size N and expansion constant c_R, a query set Q of size $O(N)$ and expansion constant c_Q, and bounded degree of bichromaticity κ of the (Q,R) pair, the FindAllNN subroutine of Algorithm 1 computes the nearest neighbor in R of each point in Q in $O(c_R^{12} c_Q^{4\kappa} N)$ time.*

*[*Partial proof:*] Since at any level of recursion, the size of R [Corresponding to $\mathscr{C}(R_i)$ in Algorithm 6.1] is bounded by $c_R^4 \max_i R_i$*

(width bound), and the maximum depth of any point in the explicit tree is $O(c_R^2 \log(N))$ (depth bound), the number of nodes encountered in Line 6 is $O(c_R^6 \max_i |R_i| \log(N))$. Since the traversal down the query tree causes duplication, and the duplication of any reference node is upper bounded by $c_Q^{4\kappa}$, Line 6 [corresponds to line 8 in Algorithm 6.1] takes at most $c_Q^{4\kappa} c_R^6 \max_i |R_i| \log(N)$ in the whole algorithm. "

The above arguments claimed the algorithm runs Line 8 at most this number of times:

$$\#(\text{Line 8}) \leq D(\mathscr{E}(R)) \cdot \max_i \mathscr{C}(R_i) \cdot (\text{number of duplications}). \quad (1)$$

It will be shown that cover tree $\mathscr{E}(R)$ from Example 6.3 does not satisfy the inequality above.

Let $X, \mathscr{E}(R), \mathscr{E}(Q), R, Q$ be as in Example 6.3 for some parameter m. We will consider the simulation of Algorithm 6.1 on pair $(\mathscr{E}(Q), \mathscr{E}(R))$. We note first Lemma 3.3 applied on $\mathscr{E}(R)$ provides $\max_{p \in R} D(p) \leq 2m+1$ As in Counterexample 5.2, a contradiction will be achieved by showing that R_i and a set of its children $\mathscr{C}(R_i)$ have a constant size bound on any recursion (i, j) of Algorithm 6.1.

Since $\mathscr{E}(R)$ contains at most one children on every level i we have $|\mathscr{C}(R_i)| \leq |R_i| + 1$ for any recursion of FindAllNN algorithm. For any $i > m^2$ denote r_i and q_i to be r. Note first that since $l(q_t) = t$ for any $t \in [1, m^2]$, then q_t is recursed into from FindAllNN($t+1, t+1, p, R_i$), where p is parent node of q_t. Therefore it follows that $t \geq i+1$ in any stage of the recursion. Let us prove that for any $i \in [1, m^2+1]$ following two claims hold: (1) Function FindAllNN(i , $j = i-1$, q_t, R_i) is called for all $t \geq i+1$ and (2) We have $R_i = \{r_{i+1}, r_i, r\}$ in this stage of the algorithm. The claim will be proved by induction on i. Let us first prove case $i = 2m+1$. Note that Algorithm 6.1 is originally launched from FindAllNN($2m+1, 2m+1, r, \{r\}$), therefore the first claim holds. Second claim holds trivially since $r_{2m+2} = r$ and $r_{2m+1} = r$.

Let the claim hold for some i, let us show that the claim will always hold for $i-1$. Assume that FindAllNN($i, j = i-1, q_t, R_i$) was called for some $t \geq i+1$. Since $j = i-1$, we perform a reference expansion (lines 5-9). By line 6 and induction assumption we have

$$\mathscr{C}(R_i) = \{(r, [i-1]), (r_{i+1}, [i-1]), (r_i, [i-1]), (r_{i-1}, [i-1])\}.$$

Assume first that $q_t = r$. Recall that for any $u \in [1, m^2]$ we have $d(r, r_u) \geq 2^{u+1}$. It follows that

$$\begin{aligned} R_{i-1} &= \{r' \in \mathscr{C}(R_i) \mid d(r, r') \leq 2^i + 2^{i+1}\} \\ &= \{(r, [i-1]), (r_i, [i-1]), (r_{i-1}, [i-1])\} \end{aligned}$$

Let us now consider case $q_t \neq r$. We have $d(r, q_t) = 2^{t+1}$ and $d(q_t, r_{u+1}) = 2^{t+1} + 2^{u+2}$ for any $u \in [1, m^2+1]$. Therefore

$$R_{i-1} = \{r' \in C_{i-1} \mid d(q_t, r') \leq d(q_t, r) + 2^i + 2^{i+1} \leq 2^{t+1} + 2^i + 2^{i+1}\}.$$

It follows that $R_{i-1} = \{(r, [i-1]), (r_i, [i-1]), (r_{i-1}, [i-1])\}$. In both cases we proceed to line 8 where we launch FindAllNN($i-1, i-1, q_t, R_{i-1}$). After proceeding into the recursion we have $j = i$ and therefore query-expansion (lines 9-11) will be performed. Note that q_t was chosen so that $t \geq i+1$. Since every q_{t-1} is either a child of r or q_t it follows that FindAllNN($i-1, i-2, q_{t'}, R_{i-1}$) will be called for all $t' \geq t-1 \geq i$. Then condition (2) of the induction claim holds as well.

It remains to show that Algorithm 6.1 $(q, R_i = \{r\})$ has $O(m^4)$ low bound on the number of times reference expansions (lines 5-9) are performed. Let ξ be the number of times Algorithm 6.1 performs reference expansions. For every $q' \in Q$ denote $\xi(q')$ to be the total number of reference expansions performed for q'. Recall that any query node q' is introduced in the query expansion (lines 9-11) for parameters $(i = u+1, j = u+1, p, R_i)$, where p is the parent node of q'. Since R_i is non empty for all levels $[1, u]$ we have $\xi(q_u) \geq u-1$ for all u. Thus

$$\xi = \sum_{q' \in Q} \xi(q') \geq \sum_{u=2}^{m^2+1} u - 2 = O(m^4).$$

There are different interpretations for the number of duplications. Note that the query tree $\mathscr{E}(Q)$ has exactly one new child on every level and that trees $\mathscr{E}(Q)$ and $\mathscr{E}(R)$ contain exactly the same levels. By using the definitions the number of duplications should be 2. However, since there can be other interpretations for the number of duplications, we make a rough estimate that the number of duplications is upper bounded by the number of nodes in query tree $O(m^2)$. By using Inequality (1), we obtain the following contradiction:

$$\begin{aligned} O(m^4) = \xi &\leq \max_{p \in R} D(p) \cdot (\max_i \mathscr{C}(R_i)) \cdot (\text{number of duplications}) \\ &\leq (2m+1) \cdot 4 \cdot m^2 \leq O(m^3) \end{aligned}$$

7 Conclusions and further work

The motivations for this paper were the past gaps in the proofs of time complexities in [9, Theorem 5], [9, Theorem 6], [38, Theorem 3.1], [31, Theorem 5.1]. In this paper, Example 3.1 introduced a dataset R and its cover tree $\mathscr{E}(R)$, where each node appears in a separate level, so the tree is split into $\sqrt{|R|}$ different branches and its maximal depth $D(\mathscr{E}(R))$ is $2\sqrt{|R|}$.

Counterexample 4.2 shows that the proof of the time complexity [9, Theorem 6] for the Insert() operation [9, Algorithm 2] is incorrect for the explicit cover tree $\mathscr{E}(R)$ in Example 3.1. Similarly, [9, Theorem 5] giving time complexity bound for the nearest neighbors search algorithm [9, Algorithm 1] has a similar gap in the proof when used on $\mathscr{E}(R)$. Counterexample 6.5 shows that the same mistake was later repeated in the dual-tree approach for all nearest neighbor search [38, Theorem 3.1].

Another forthcoming paper based on the PhD thesis [18] studies a new compressed cover tree that overcomes the past obstacles in [9, Theorem 5], [9, Theorem 6] and proves the parameterized near-linear time complexities for the compressed cover tree construction and the k-nearest neighbor search for any $k \geq 1$. In [18, Corollary 3.5.11] it is shown that a compressed cover tree can be constructed in $O(c(R)^{10} \cdot \log_2(|R|) \cdot |R|)$ and [18, Theorem 3.7.14] shows that using compressed cover tree k-nearest neighbors of any point q can be found in time

$$O(c(R \cup \{q\})^3 \cdot \log_2(k) \cdot (c(R \cup \{q\})^9 \cdot \log_2(|R|) + k)).$$

The near-linear complexities above have helped justify the fast neighbor-based algorithms for computing generically complete isometry invariants of periodic point sets [41, 42]. The resulting ultra-fast distinguished all periodic crystals in the Cambridge Structural Database (the world's largest collection of real materials) through more than 200 billion comparisons only over two days on a modest desktop. The newly established *Crystal Isometry Principle* says that all real periodic crystals have unique locations in a common *Crystal Isometry Space* (CRISP) parameterized by complete invariants.

The first maps of CRISP are visualized for millions of 2-dimensional [10, 28] and 3-dimensional [11, 25] lattices extracted from real crystals. Though the data ambiguity of traditional crystal representations

was a long-standing challenge in crystallography [1, 27, 33] and despite the substantial progress in developing continuous invariants such as density functions [4, 17, 40],The the recent complete invariant isosets [2,3] allowed only approximate algorithms. For finite sets of unlabeled points representing bounded rigid structures, complete isometry invariants were recently introduced with computable and continuous metrics in [26] motivated by generic families of point sets [39] with identical 1-dimensional persistence.

This research opened a new wider area of Geometric Data Science supported by the £3.5M EPSRC grant 'Application-driven Topological Data Analysis' with Oxford (2018-2023, EP/R018472/1), the Royal Academy of Engineering Industrial Fellowship 'Data Science for Next Generation Engineering of Solid Crystalline Materials' (2021-2023, IF2122/186), and the EPSRC New Horizons grant 'Inverse design of periodic crystals' (2022-2024, EP/X018474/1).

We thank all reviewers for their valuable time and suggestions.

REFERENCES

[1] O. Anosova and V. Kurlin. Introduction to periodic geometry and topology. *arXiv:2103.02749*, 2021.

[2] O. Anosova and V. Kurlin. An isometry classification of periodic point sets. In *LNCS Proceedings of Discrete Geometry and Mathematical Morphology*, pp. 229–241, 2021.

[3] O. Anosova and V. Kurlin. Algorithms for continuous metrics on periodic crystals. *arXiv:2205.15298*, 2022.

[4] O. Anosova and V. Kurlin. Density functions of periodic sequences. In *Discrete Geometry and Mathematical Morphology*, 2022.

[5] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the ACM SIGMOD International Vonference on Management of Data*, pp. 322–331, 1990.

[6] J. Bentley and J. Friedman. Fast algorithms for constructing minimal spanning trees in coordinate spaces. *IEEE Transactions on Computers*, 27(02):97–105, 1978.

[7] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[8] S. Berchtold, D. Keim, and H. Kriegel. The x-tree: An index structure for high-dimensional data. In *Very Large Data-Bases*, pp. 28–39, 1996.

[9] A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *Proceedings of ICML*, pp. 97–104, 2006.

[10] M. Bright, A. Cooper, and V. Kurlin. Geographic-style maps for 2-dimensional lattices. *arxiv:2109.10885*, 2021.

[11] M. Bright, A. Cooper, and V. Kurlin. Welcome to a continuous world of 3-dimensional lattices. *arxiv:2109.11538*, 2021.

[12] R. Cole and L.-A. Gottlieb. Searching dynamic point sets in spaces with bounded doubling dimension. In *Proceedings of ACM STOC*, pp. 574–583, 2006.

[13] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on information theory*, 13(1):21–27, 1967.

[14] R. R. Curtin. Improving dual-tree algorithms. *PhD at Georgia Institute of Technology, http://ratml.org/pub/pdf/2015improving.pdf*, 2015.

[15] R. R. Curtin, D. Lee, W. B. March, and P. Ram. Plug-and-play dual-tree algorithm runtime analysis. *J. Mach. Learn. Res.*, 16:3269–3297, 2015.

[16] R. R. Curtin, P. Ram, and A. G. Gray. Fast exact max-kernel search. In *Proceedings of SIAM ICDM*, pp. 1–9, 2013.

[17] H. Edelsbrunner, T. Heiss, V. Kurlin, P. Smith, and M. Wintraecken. The density fingerprint of a periodic point set. In *Symp. Comp. Geom.*, pp. 32:1–32:16, 2021.

[18] Y. Elkin. New compressed cover tree for k-nearest neighbor search. *arXiv:2205.10194*, 2022.

[19] R. A. Finkel and J. L. Bentley. Quad trees a data structure for retrieval on composite keys. *Acta informatica*, 4(1):1–9, 1974.

[20] K. Fukunaga and P. M. Narendra. A branch and bound algorithm for computing k-nearest neighbors. *IEEE Transactions on computers*, 100(7):750–753, 1975.

[21] D. Fussell and K. R. Subramanian. *Fast ray tracing using kd trees*. University of Texas at Austin, Department of Computer Sciences, 1988.

[22] M. P. Holmes, A. G. Gray, and C. L. Isbell Jr. QUIC-SVD: Fast SVD using cosine trees. In *Proceedings of NIPS*, pp. 673–680, 2008.

[23] D. R. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. In *Proceedings of ACM STOC*, pp. 741–750, 2002.

[24] R. Krauthgamer and J. R. Lee. Navigating nets: Simple algorithms for proximity search. In *Proceedings of SODA*, pp. 798–807. Citeseer, 2004.

[25] V. Kurlin. A complete isometry classification of 3-dimensional lattices. *arxiv:2201.10543*, 2022.

[26] V. Kurlin. Computable complete invariants for finite clouds of unlabeled points. *arxiv:2207.08502*, 2022.

[27] V. Kurlin. Exactly computable and continuous metrics on isometry classes of finite and 1-periodic sequences. *arXiv:2205.04388*, 2022.

[28] V. Kurlin. Mathematics of 2-dimensional lattices. *arxiv:2201.05150*, 2022.

[29] K. Lin, H. Jagadish, and C. Faloutsos. The tv-tree: An index structure for high-dimensional data. *The VLDB Journal*, 3(4):517–542, 1994.

[30] T. Liu, A. W. Moore, A. G. Gray, and K. Yang. An investigation of practical approximate nearest neighbor algorithms. In *Proceedings of NIPS*, vol. 12, 2004.

[31] W. B. March, P. Ram, and A. G. Gray. Fast euclidean minimum spanning tree: algorithm, analysis, and applications. In *Proceedings of SIGKDD*, pp. 603–612, 2010.

[32] J. McNames. A fast nearest-neighbor algorithm based on a principal axis search tree. *IEEE Transactions on pattern analysis and machine intelligence*, 23(9):964–976, 2001.

[33] M. Mosca and V. Kurlin. Voronoi-based similarity distances between arbitrary crystal lattices. *Crystal Research and Technology*, 55(5):1900197, 2020.

[34] S. M. Omohundro. *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

[35] D. Pelleg and A. Moore. Accelerating exact k-means algorithms with geometric reasoning. In *Proceedings of ACM SIGKDD*, pp. 277–281, 1999.

[36] P. Ram and A. G. Gray. Maximum inner-product search using cone trees. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 931–939, 2012.

[37] P. Ram, D. Lee, and A. G. Gray. Nearest-neighbor search on a time budget via max-margin trees. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pp. 1011–1022. SIAM, 2012.

[38] P. Ram, D. Lee, W. March, and A. Gray. Linear-time algorithms for pairwise statistical problems. *Advances in Neural Information Processing Systems*, 22:1527–1535, 2009.

[39] P. Smith and V. Kurlin. Families of point sets with identical 1d persistence. *arxiv:2202.00577*, 2022.

[40] P. Smith and V. Kurlin. A practical algorithm for degree-k voronoi domains of three-dimensional periodic point sets. In *Proceedings of ISVC*, 2022.

[41] D. Widdowson and V. Kurlin. Resolving the data ambiguity for periodic crystals. In *Proceedings of NeurIPS: Neural Information Processing Systems (arXiv:2108.04798)*, 2022.

[42] D. Widdowson, M. Mosca, A. Pulido, V. Kurlin, and A. Cooper. Average minimum distances of periodic point sets - fundamental invariants for mapping all periodic crystals. *MATCH Comm. in Mathematical and in Computer Chemistry*, 87:529–559, 2022.

[43] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of SODA*, vol. 93, pp. 311–21, 1993.