A new compressed cover tree for k-nearest neighbour search and the stable-under-noise mergegram of a point cloud

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy by

Yury Elkin

December 2022

Abstract

The analysis of data sets mathematically representable as finite metric spaces plays a significant role in every scientific study. In this thesis we focus on constructing new effective algorithms in the area of computational geometry that can be effectively deployed for the study and classification of large data sets prevalent in natural science, economic analysis, medicine, environmental protection etc.

The first major contribution of this thesis is a new near-linear time algorithm, that resolves the classical problem of finding k-nearest neighbors (KNN) to of query set Q in a larger reference set R, where Q and R both belong to some metric space X. This project was inspired by the work of Beygelzimer, Kakade, and Langford in ICML 2006 that attempted to show that such problem is resolvable for $k = 1$ having a near-linear time complexity. However, in 2015 it was pointed out that the proof of their time complexity might contain mistakes, which has been ascertained in this thesis by showing that the proposed proof does not withstand a concrete counterexample. An important application of the KNN algorithm is a KNN graph on a finite metric space R whose edge set is formed by connecting every point $p \in R$ with its k-nearest neighbors. The KNN graph finds its application in areas of data-skeletonization, where it can serve as an initial skeleton of the data set, or in cluster analysis, where connected components of the KNN graph can represent the clusters.

Another application of the the KNN algorithm is Minimum spanning tree (MST), which is an efficient way to visualize any unstructured data while knowing only distances, for example any metric graph connecting abstract data points. Although many efficient algorithms for the MST in metric spaces have been devised, there existed only one past attempt to justify a near-linear time complexity in the size of a given metric space. In 2010 March, Ram, and Gray claimed that MST of any finite metric space can be built in a parametrized near-linear time. In this work we have demonstrated, with multiple counterexamples, that the attempted proof was incorrect by showing that one of its step fails. Encouraged by the results of the work of 2010 this thesis produces a new algorithm that is based on Boruvka algorithm, which is combined with the KNN method to resolve the metric MST problem in a near-linear time complexity.

In the thesis final chapter the MST algorithm is applied in the computation of a new isometry invariant mergegram of Topological Data Analysis (TDA). TDA quantifies topological shapes hidden in unorganized data such as clouds of unordered points. In the 0-dimensional (0D) case,

the distance-based persistence is determined by a single-linkage (SL) clustering of a finite set in a metric space. Equivalently, the $0D$ persistence captures only edge lengths of a Minimum Spanning Tree (MST). Both the SL dendrogram and the MST are unstable under perturbations of points. In this thesis, we define the new stable-under-noise mergegram which outperforms previous isometry invariants on a classification of point clouds.

In conclusion, the developed fast algorithms of this thesis can cater to a vast varieties of tasks in data science and beyond. The newly proposed corrected time complexity analysis of KNN and MST not only rectifies the past issues in their theoretical justifications but also gives a way to fix analogous issues in other similar methods based on the cover tree data structure.

Acknowledgements

First of all, I would like to thank my primary supervisor, Dr. Vitaliy Kurlin, for giving me the opportunity to deeply study in a "high-voltage" field of mathematics about which I remain to be passionate as ever. His incredible experience has been inspiring and encouraging to me. His persistent moral support during all these years was a significant factor to my motivation. I am so grateful for all our fruitful conversations that greatly aided to my research results!

I would also like to thank my second supervisor, Dr. Marja Kankaanrinta, for all her guidance, her patience, and for sharing her knowledge with me. Her feedback has been fundamental to my research and steered me steadily the right course.

Finally, I can not put in words how much I owe to my family, for their commitment to assist, uplift and back me up.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Topological Data Analysis (TDA) is a recent modern branch of mathematics that appeared from various works in applied topology and computational geometry during the early 2000s. Although one can trace back geometric approaches for data analysis quite far in the past, TDA started as a field with the pioneering works of Edelsbrunner et al. [29] and Zomorodian and Carlsson [94] in persistent homology and was later popularized in a landmark paper in 2009 Carlsson [14]. TDA is mainly motivated by the idea that topology and geometry provide a a powerful approach to access information about the structure of data [19].

TDA aims to provide adequate mathematical, statistical, and algorithmic methods to derive, analyze and exploit the underlying complex topological and geometric structures in given data. The data is often represented as point clouds in Euclidean or more complex metric spaces. During last years, gigantic efforts have been made to provide TDA with powerful data structures and algorithms that are now implemented and ready to use, this involves skeletonization methods such as [83] Mapper, Homologically Persistent Skeleton [84, 58], and fast approximate skeleton [34], as well as libraries allowing us to compute persistent homology such as the Gudhi library (C++ and Python) Clément Maria et al [64], as well as a set of efficient tools that can be used in combination or complementary to other data sciences tools, such as deep learning [52].

There now exist a large variety of methods and techniques inspired by topological and geometric advances. Providing a complete analysis of all these existing advances is beyond the scope of this Ph.D. thesis. However, most of them rely on the following typical pipeline pattern that constitutes

as a backbone for topological data analysis:

1. The input is assumed to be a finite set of points coming with a notion of distance. This distance can be induced by the metric in the underlying space (e.g. the Euclidean metric when the data are embedded in $\mathbb{R}^m$) or come as an arbitrarily defined metric for instance described as a pairwise distance matrix. The definition of the metric on the data is commonly given as an input advised by the application. It should be noted that the choice of metric might be crucial for bringing to light the thought-provoking topological and geometric features of the data.

2. A "continuous" shape is constructed on top of the data to reformulate the task in terms of topology or geometry. This is often a CW-complex or a nested family of complexes, called a filtration, that highlights the formation of the data at different scales. The complexes can be seen as a higher-dimensional analog of neighboring graphs that are in normal circumstances built on top of data which is the case in the classical data analysis or learning procedures. The main challenge of this step is to come up with an appropriate structure that will outline the essential information.

3. Topological or geometric information is extracted from the output of step 2. This can result in full reconstruction, typically a skeleton of the shape, from which geometric features can be easily extracted or, in approximate summaries, or so-called algebraic invariants. The latter highlights relevant information, such as persistence diagrams that are isometric invariant of the original data and that can be often quickly computed in a modern GPU-parallel way for example using a Ripser algorithm [4]. Beyond getting interesting topological/geometric information or its interpretation the goal is to prove its relevance, in particular its stability with respect to noise in the input data.

4. The summaries produced in step 3 may generate additional sets of features and descriptors of the initial data points. They can be used to improve our understanding of the data, for example, via visualization, such as using the new TDAView visualization technique [87] or TTK [85]. Or they can be combined with other features and used further in machine learning tasks, for example, a neural network can be built on persistence diagrams using a topological landscape layer [52] or Perslay-library [17] containing multiple customized permutation invariants.

1.2   Thesis structure

The central objective of this thesis is to describe new computational methods for the analysis of finite metric spaces, as well as parametrized near-linear time complexities for k-nearest neighbors search, construction of minimum spanning tree, as well as newly a developed object, mergegram. The research is carried out in the framework of space partitioning data structures and topological data analysis (TDA), for which we develop and effectively utilize novel computational geometry techniques. All worst-case time complexity estimates in this work are measured by the random-access machine model [42]. This work breaks into following four chapters:

- Chapter 2 introduces the basic concepts and definitions.

- Chapter 3 solves the k-nearest neighborhood problem.

- Chapter 4 is dedicated to the minimum spanning tree problem.

- Chapter 5 introduces a new invariant of finite metric spaces, mergegram.

To avoid confusion we formally define the meaning of *incorrectness of proof.*

Definition 1.2.1. We say that a result (Theorem, Proposition, Lemma ...) has an incorrect proof, if it contains a claim and there exists an example for which the claim can not be satisfied.

It should be noted that if a presented proof in the literature for some result is incorrect it does not necessary follow that the result is incorrect since there might exist another correct proof for it.

1.2.1   A new compressed cover tree for the k-nearest neighbor search

Chapter 3 is dedicated to the exact k-nearest neighbor problem in any metric space. In a modern formulation, the problem is to find all $k \geq 1$ nearest neighbors in a finite reference set R for all points from a finite query set Q. Both sets live in an ambient space X with a given 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.

The approach taken to resolve the k-nearest neighbors problem is inspired by [9]. Publication [9] introduced new concepts: cover tree, which stored sets of R into a leveled tree-type data structure and a parameter, expansion constant $c(R)$ that was defined as a growth-bound for any finite metric space R. The main contribution of [9] was an attempt to prove two new results regarding time upper bound time complexity estimates of the cover tree construction

algorithm and the nearest neighbors algorithm. Theorem 6 of [9] claimed a construction time of $O(c(R)^{O(1)} \cdot |R| \log |R|)$ for cover tree on any finite metric space R. Theorem 5 claimed that the first nearest neighbor of any query point $q \in Q$ can be found in time $O(c(R)^{O(1)} \cdot \log |R|)$ assuming that the cover tree data structure was built on the reference set R. However, in this Ph.D. thesis both of the results have been shown to have incorrect proof: Counterexample 3.4.5 shows that a step in the proof of Theorem 6 and Counterexample 3.4.7 shows a step in the proof of Theorem 5 were incorrect.

In this Ph.D. thesis Definition 3.2.1 introduces a new concept: compressed cover tree, which is a simplified version of a cover tree. For its construction, Algorithm 3.5.3 is employed, which is an adaption of [9, Algorithm 2] to compressed cover trees. Corollary 3.5.11 proves that the upper bound time complexity estimate of the new construction method is bounded by $O(c(R)^{O(1)} \cdot |R| \cdot \log |R|)$. This thesis introduces a new method for finding k-nearest neighbors Algorithm 3.7.2. Algorithm 3.7.2 is a modification of [9, Algorithm 1], where an additional stoppage is added to prove a near-logarithmic time complexity for a k-nearest neighbors search of a single query point $q \in Q$. The main result, Theorem 3.7.14 gives an upper bound time complexity of $O\big(c(R)^{O(1)} \cdot \log(k) \cdot (\log(|R|) + k)\big)$ for the newly introduced k-nearest neighbors method.

Other important results of this section are:

- Definition 3.1.4 introduces new dimensionality constant $c_m(R)$. Theorem 3.3.15 shows that if R is a finite subset of $\mathbb{R}^n$, then $c_m(R) \leq 2^n$.

- Theorem 3.5.7 and Corollary 3.7.9 give an upper bound for time complexity of the compressed cover tree construction algorithm and the k-nearest neighbors algorithm by using aspect ratio $\Delta(R)$ of Definition 3.2.14 and minimized expansion constant $c_m(R)$ of Definition 3.1.4 as parameters.

- Theorem 3.8.6 gives an upper bound for time complexity of $(1 + \epsilon)$-approximate k-nearest neighbors method, Algorithm 3.8.2.

- Counterexample 3.9.2 shows that the time complexity result [78, Theorem 3.1] of the dual-tree approach to the nearest neighbor problem had an incorrect proof as well.

1.2.2   Fast algorithm for MST based on the compressed cover tree

Chapter 4 is dedicated to metric minimum spanning tree problem. A minimum spanning tree on a finite metric space (R, d) is a tree MST(R) with vertex set R and a minimum total length of

Table 1.1: This table contains list of theorems that used the cover tree datastructure and were proven incorrectly, as well as new algorithms and new theorems that are intended to rectify the past mistakes.

Original result	Counterexample	New Algorithm	New Theorem
k-nn search [9, Theorem 5]	Counterexample 3.4.7	Algorithm 3.5.3	Corollary 3.5.11
Cover-tree construction [9, Theorem 6]	Counterexample 3.4.5	Algorithm 3.7.2	Theorem 3.7.14
Paired-tree k-nn search [78, Theorem 3.1]	Counterexample 3.9.5	Ryan Curtin [26, Algorithm 2,3]	Ryan Curtin Weaker result [26, Theorem 2]
Construction of MST [63, Theorem 5.4]	Counterexample 4.2.3, Counterexample 4.3.5	Single-tree Algorithm 4.4.3	Weaker result Corollary 4.4.15

edges among all possible spanning trees of R. In the metric minimum spanning tree problem the goal is to construct a near-linear time algorithm that constructs MST(R) on any finite metric space R in a near-linear time.

The approach taken to resolve the metric minimum spanning tree problem is inspired by [63]. In [63, Algorithm 1] there was an attempt to solve the metric MST problem by exploiting a cover tree datastructure and a dual-tree nearest neighbors algorithm [78, Algorirthm 1] in a near linear time. However, the proof of time complexity result [63, Theorem 5.1] was exhibited to be incorrect in Counterexample 4.2.3 and Counterexample 4.3.5 .

To resolve the metric minimum spanning tree problem we propose new method, [63, Algorithm 1], which is a single-tree version of [63, Algorithm 3]. Theorem 4.4.9 proves the correctness of Algorithm 4.4.4. Theorem 4.4.14 shows that the time complexity of Algorithm 4.4.4 is: $O\Big((c_m(R))^{4+\lceil \log_2(\rho(R))\rceil} \cdot \log_2(\Delta(R)) \cdot |R| \cdot \log(|R|) \cdot \alpha(|R|)\Big)$, where $\rho(R)$ is the ratio: maximal edge length divided by the minimal edge length of MST(R) and $\Delta(R)$ is the aspect ratio of Definition 3.2.14.

1.2.3   Mergegram extends the 0-dimensional persistence

Chapter 5 introduces a new isometry invariant of a point cloud, mergegram. Mergegram is an extension of the 0-dimensional persistence diagram, in such a way that it encodes a dendrogram built on X into multiset of $\mathbb{R}^2$ by projecting every link of the dendrogram into a pair (the link

start, the link end). It will be proved that the mergegram is stable under noise in the sense that a topological space and its noisy point sample have close persistence diagrams. The stability of mergegram shows that it is a suitable method for recognition of noisy point clouds.

One of the main aims of Chapter 5 is to solve an isometry recognition problem: given a set of different objects, the goal is to find out which of the objects have similar shapes. In the specific case, where objects are point clouds, the goal is to classify them as classes modulo isometries that preserve distances between points. In practice, we are given a dataset of n shapes, each of which belongs to one of m shape classes. The goal is to build an algorithm that takes as input a shape that is isometric to one of the m shape classes and outputs its correct class.

All the main results of Chapter 5 were published in: [32] and [33]. Here is the summary of new contributions to Topological Data Analysis:

- Definition 5.3.4 introduces the concept of a mergegram for any dendrogram of clustering.

- Theorem 5.4.1 justifies that the mergegram of a single-linkage dendrogram is strictly stronger than the 0D persistence of a distance-based filtration of sublevel sets.

- Theorem 5.4.2 showing how to reconstruct a dendrogram of a single-linkage clustering from a mergegram in general position.

- Theorem 5.5.4 proves bottleneck stability property of mergegram.

- Theorem 5.5.6 shows that the mergegram can be computed in a near-linear time.

- Sections 5.6 and 5.7 shows experimentally that mergegram performs better than 0-dimensional persistence diagram or a diagram made of distances to two closes neighbors NN(2) for the shape recognition problems.

1.3   Publications

Counterexamples of Chapter 3 are published in [31]. Rest of the chapter will be published later. Chapter 4 will be published later. Chapter 5 is published in [32] and [33].

Chapter 2

Basic definitions and past results

2.1 Metric spaces

Topological and geometric features can be usually found only in continuous structures and therefore as real data usually represent finite sets of observation it is impossible to directly inherit any topological information that the input contains. One of the natural ways to approach this problem is to connect data points that are "close" to each other. This unmasks global continuous shapes associated with the data. To specify the notion of closeness between data points we use the standard notion of metric introduced in Definition 2.1.1.

Definition 2.1.1 (Metric space)**.** Metric space (X, d) is a set X with a function $d : X \times X \to \mathbb{R}$, called a distance, such that for any triplet $x, y, z \in X$ the following properies of

1. Identity: $d(x, y) \geq 0$ and $d(x, y) = 0$ if and only if $x = y$

2. Symmetry: $d(x, y) = d(y, x)$ and,

3. Triangle inequality: $d(x, z) \leq d(x, y) + d(y, z)$

are satisfied.

Definition 2.1.2 (Hausdorff distance)**.** Let A, B be two non-empty subsets of a metric space (X, d). We define their Hausdorff distance $d_H(A, B)$ by

$$d_H(A, B) = \max\{\sup_{b \in B} d(b, A), \sup_{a \in A} d(a, B)\}$$

Definition 2.1.3 (δ-sparse subset). We say that a subset C of metric space (X, d) is δ-sparse if $d(a, b) \geq \delta$ for all $a, b \in C$.

From the TDA perspective, Hausdorff distance provides a helpful way to specify the distance between two sets which are subsets of the same metric space. However, in some cases, one has to compare data sets that belong to different metric spaces. In this case, we can use the Gromov-Hausdorff distance which generalizes Hausdorff distance in a way where it can be used for the comparison of pairs of compact metric spaces.

Definition 2.1.4 (Isometric embedding). Let (X, d) and (Y, u) be metric spaces. Injection $\phi : X \to Y$ is a *isometric embedding* if $d(a, b) = u(\phi(a), \phi(b))$ for any $a, b \in X$.

Definition 2.1.5 (Gromov-Hausdorff distance). The Gromov-Hausdorff distance $d_{GH}(X, Y)$ measured on two compact metric spaces X, Y is the infinium $d_H(f(X), f(Y))$ of all metric space M and isometric embeddings $f : X \to M$, $g : Y \to M$.
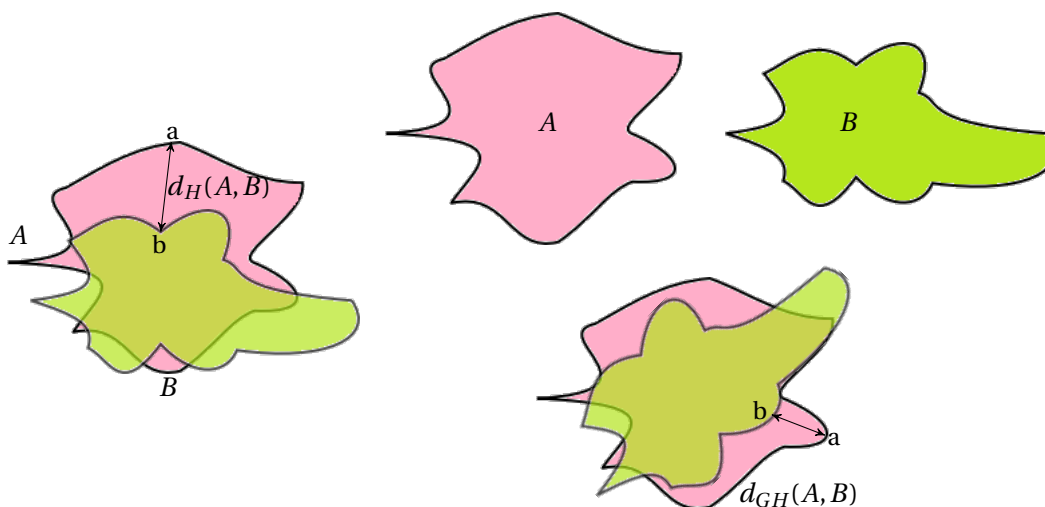
Figure 2.1: This figure illustrates Hausdorff distance between A and B on **left** side and their Gromov Hausdorff distance on **right** side.

Example 2.1.6 (Hausdorff distance and Gromov-Hausdorff distance). Let A, B of Figure 2.1 be sets located in the same plane of $\mathbb{R}^2$. In the picture the distance of points a, b equals $d_H(A, B)$, because $b \in B$ is the nearest neighbor of a which produces maximal distance to its nearest neighbor over all points in A. On the other hand, while computing the Gromov-Hausdorff

distance we assume that A and B might belong to distinct spaces. Thus A can be moved and rotated to reduce its distance to B. Because we are allowed to perform isometry operations, while computing Gromov-Hausdorff distance on both spaces, it is evident that $d_{GH}(A, B) \leq d_H(A, B)$.

2.2 Graphs

This section formally defines concept of graph.

Definition 2.2.1 (Graph)**.** An undirected graph G is a pair $(\mathbb{V}(G), \mathbb{E}(G))$, where:

- $\mathbb{V}(G)$ is a finite set of vertices.

- $\mathbb{E}(G)$ is a finite set of unordered pairs (a, b) of $\mathbb{V}(G) \times \mathbb{V}(G)$

We can extend any graph G into *a weighted graph* by introducing a function $w : \mathbb{E}(G) \to \mathbb{R}_+$, that measures the length of each edge.

Abstract graph defined above does not represent directly any geometrical shape. However, this notion can be developed into a *embedded* graph which contains the features of abstract graph, but which now exposes explicit geometrical info.

Definition 2.2.2 (Embedded graph)**.** Given a metric space (X, d) define embedded graph G to be a graph having

- Vertex set $\mathbb{V}(G)$ is a subset of X.

- For each edge $(u, v) \in \mathbb{E}(G)$ there exists a path in X which connects a to b.

In a special case where $X = \mathbb{R}^m$ we say that a graph is a *straight-line* graph if every path in the edge set E is a straight line.

2.3 Heap

Heap is a tree-based data structure which was designed to store elements with a priority in an effective manner. Although there are many implementation of heap data structure for simplicity in this section we focus only on binary heap data structure.

First introduced in 1964 binary heap [91] was first concrete implementation of heap data structure, that is effective at storing data with additional priority property. The binary heap data

structure allows adding elements, removing the minimal element, and decreasing the value of any given element in logarithmic time. To define a binary heap, we first define the notion of a complete binary tree.
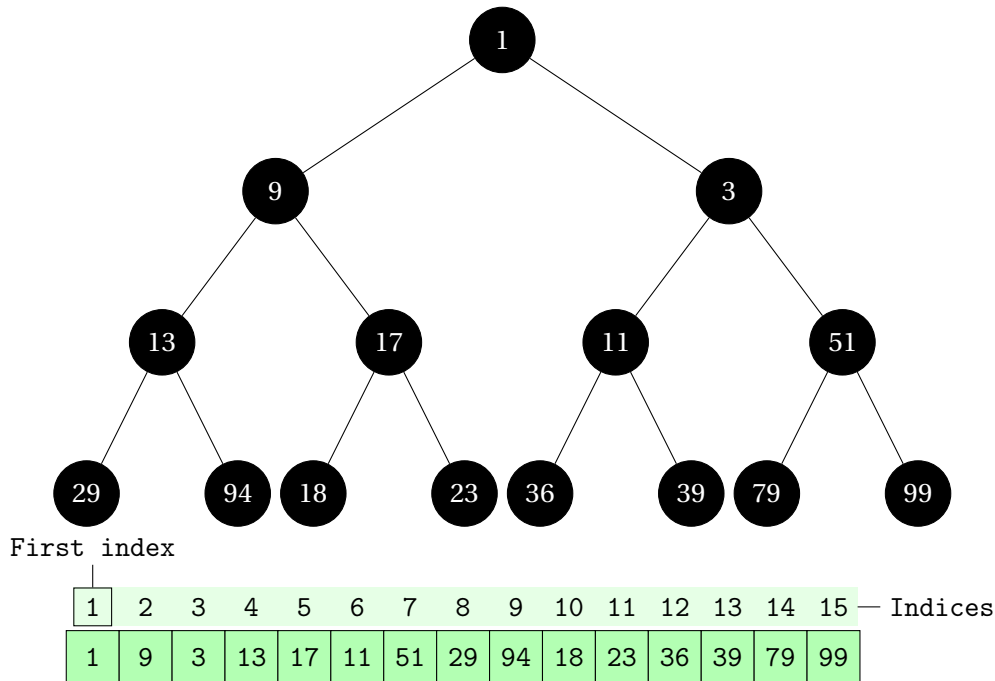
Figure 2.2: a binary heap built on a finite subset of integers; on the **top:** its binary tree representation; on the **bottom:** its array-representation.

Definition 2.3.1 (Binary search tree)**.** Given a binary tree, assign an index i to each of its nodes. To start, assign the root an index $i = 1$. Now suppose a node has been assigned an index i. Then assign its left child (if it exists) the index $2i$, and assign to its right child (if it exists) the value $2i + 1$. Then we say that the tree is *complete* iff the set of assigned indices is a *contiguous* set; meaning that, if i and k are assigned indices, with $i < k$, then so is j, for every $i < j < k$.

Definition 2.3.2. Let $(C, \leq)$ be a completely ordered set. A *min heap* (respectively, *max heap*) is a complete binary tree T on the set C, which satisfies the following condition: If $p \in T$, and $q \in T$ is a child of p, then the element stored at q is larger (respectively, is smaller) or equal to the element stored at p.

Lemma 2.3.3 (Binary heap complexities)**.** *A min binary heap T built on an ordered set $(C, \leq)$ has the following complexities:*

- *Adding any element $a \in C$ to a binary tree can be done in $\log(|C|)$ time.*

- *Finding the minimal of the set C can be done in $O(1)$ in time.*

- *Removing the minimal element of the set C can be performed in $O(\log(|C|))$ time.*

Proof. The proof is available in [25, Section 6.5]. $\qquad\qquad\qquad\qquad\qquad\qquad\square$

In the explicit implementation, a binary heap often has a tree-like structure, where the minimal element can always be found at the root of the tree.

2.4 Hierarchical clustering

This section formally defines concepts related to hierarchical clustering. Hierarchial clustering will be used in the definition of mergegram, Chapter 5.

Definition 2.4.1 (partition set $\mathbb{P}(A)$)**.** For any set A, a *partition* of A is a finite collection of non-empty disjoint subsets $A_1, \ldots, A_k \subset A$ whose union is A. The *single-block* partition of A consists of the set A itself. The *partition set* $\mathbb{P}(A)$ consists of all partitions of A. $\qquad\qquad\blacksquare$

The partition set $\mathbb{P}(A)$ of the abstract set $A = \{0, 1, 2\}$ consists of the five partitions

$$(\{0\}, \{1\}, \{2\}), \quad (\{0, 1\}, \{2\}), \quad (\{0, 2\}, \{1\}), \quad (\{1, 2\}, \{0\}), \quad (\{0, 1, 2\}).$$

For example, the collections $(\{0\}, \{1\})$ and $(\{0, 1\}, \{0, 2\})$ are not partitions of A.

The definition of dendrogram was first presented in [15, Section 3.1 (4)].

Definition 2.4.2 ([15, Section 3.1])**.** Let (A, θ) be a pair , where A is a finite set and $\theta : [0, +\infty) \to \mathbb{P}(X)$ is a function from positive real numbers to the power set of X. We say that (A, θ) is a dendrogram if it satisfies the following conditions:

1. $\theta(0) = \{\{x_0\}, \ldots, \{x_n\}\}$, where $x_i \in A$ for all $i \in [n]$.

2. There exists t_0 s.t $\theta(t)$ is the single block partition for all $t \geq t_0$ i.e. $\theta(t) = \{\{x \mid x \in A\}\}$.

3. If $r \leq s$ then $\theta(r)$ refines $\theta(t)$ i.e. for every $U \in \theta_r$ there exists $V \in \theta_s$ satisfying $U \subseteq V$.

4. Technical condition [15, Section 3.1 (4)] For all r there exists $\epsilon > 0$ s.t. $\theta(r) = \theta(t)$ for $t \in [r, r + \epsilon]$.

As an important type of dendrogram, we define a single linkage clustering dendrogram in Definition 2.4.4.

Definition 2.4.3 (single-linkage clustering)**.** Let A be a finite set in a metric space X with a distance $d : X \times X \to [0, +\infty)$ and let $s \in R_+$ be a scale. Define $\Delta_{SL}(A; s)$ to be single linkage clustering at scale s, in such a way that any points $a, b \in A$ belong to *SL cluster* if and only if there is a finite sequence $a = a_1, \ldots, a_m = b \in A$ such that any two successive points have a distance at most s, i.e. $d(a_i, a_{i+1}) \le s$ for $i = 1, \ldots, m-1$.   ∎

For $s = 0$, any point $a \in A$ forms a singleton cluster $\{a\}$.

Definition 2.4.4 (single-linkage dendrogram)**.** Representing each cluster from $\Delta_{SL}(A; s)$ over all $s \ge 0$ by one point, we get the *single-linkage dendrogram* $\Delta_{SL}(A)$ visualizing how clusters merge, see Figure. 2.3.   ∎
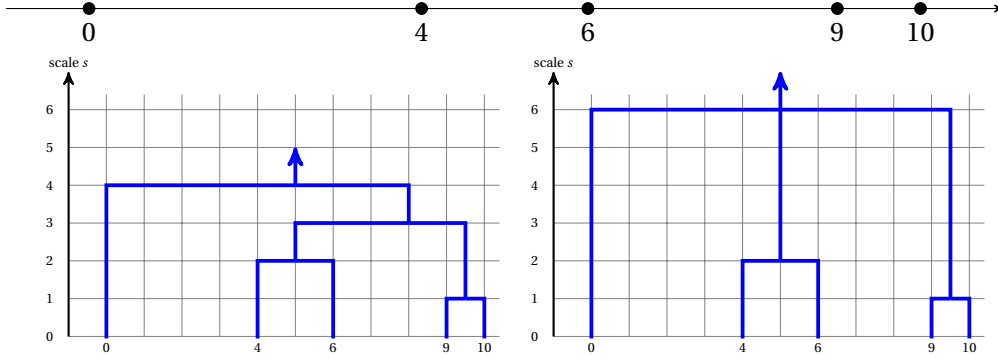
Figure 2.3: **Top**: the 5-point cloud $A = \{0, 4, 6, 9, 10\} \subset \mathbb{R}$. **Bottom left**: Single-linkage dendrogram $\Delta_{SL}(A)$ from Definition 2.4.4. **Bottom right**: Complete-linkage dendrogram $\Delta_{CL}(A)$ from Example 2.4.5.

Example 2.4.5 ([15, Section 3.2.2])**.** Let (A, d) be a finite metric space. Let $l : \mathbb{P}(A) \times \mathbb{P}(A) \to \mathbb{R}$ a function satisfying $l(U, V) = \max_{(p,q) \in U \times V} d(p, q)$ for all $U, V \in \mathbb{P}(A)$. For every $r \in \mathbb{R}$ consider equivalence relation $\frown_r$ on $\mathbb{P}(A)$, given by $B \frown B'$ if there exists a sequence in $B = B_1, \ldots, B_s = B'$ satisfying $l(B_k, B_{k+1}) \le r$ for all $k \in 1, \ldots, s-1$. Consider the sequences $r_1, r_2, \ldots \in [0, \infty)$ and $\gamma_1, \gamma_2, \ldots \in \mathbb{P}(A)$ given by $\gamma_1 = A$, and recursively for $i \ge 1$ by $\gamma_{i+1} = \gamma_i / \frown_{r_i}$, where

$$r_i = \min\{l(B, B') \mid B, B' \in \gamma_i, B \ne B'\}.$$

Finally, we define $\triangle_{CL}(A, r) = \gamma_{i(r)}$, where $i(r) = \max\{i \mid r_i \le r\}$.

Any dendrogram can be represented as an ultra-metric space. This representation will be used to prove the stability result of single-linkage dendrograms first presented in [15].

Definition 2.4.6. [15, Section 3.3.1] Let (A, θ) be a dendrogram over A. Let us define a symmetric map $u : A \times A \to \mathbb{R}_+$ given by $u(x, y) = \min\{r \in \mathbb{R}_+ \mid x, y$ belong to the same partition of $\theta(r)\}$. We say that (A, u) is the *ultra-metric space* corresponding to a dendrogram (A, θ).

Theorem 2.4.7 ([15, Proposition 2])**.** *Let (X, d_X) and (Y, d_Y) be two finite metric spaces and let (X, u_X) and (Y, u_Y) be the two finite ultrametric space corresponding to single linkage dendrograms built on (X, d_X) and (Y, d_Y), respectively. Then*

$$d_{GH}((X, u_X), (Y, u_Y)) \le d_{GH}((X, d_X), (Y, d_Y)).$$

Here, d_{GH} stands for the Gromov-Hausdorff distance of Definition 2.1.5.

2.5   Persistent homology

This section introduces the key concepts from the thorough review by Chazal et al. [19] that will be used in Chapter 5.

Definition 2.5.1 (persistence module $\mathbb{V}$)**.** A *persistence module* $\mathbb{V}$ over the real numbers $\mathbb{R}$ is a family of vector spaces V_t, $t \in \mathbb{R}$ with linear maps $v_s^t : V_s \to V_t$, $s \le t$ such that v_t^t is the identity map on V_t and the composition is respected: $v_s^t \circ v_r^s = v_r^t$ for any $r \le s \le t$. ∎

The set of real numbers can be considered as a category $\mathbb{R}$ in the following sense. The objects of $\mathbb{R}$ are all real numbers. Any two real numbers such that $a \le b$ define a single morphism $a \to b$. The composition of morphisms $a \to b$ and $b \to c$ is the morphism $a \le c$. In this language, a persistence module is a functor from $\mathbb{R}$ to the category of vector spaces.

A basic example of $\mathbb{V}$ is an interval module. An interval J between points $p < q$ in the line $\mathbb{R}$ can be one of the following types: closed $[p, q]$, open (p, q) and half-open or half-closed $[p, q)$ and $(p, q]$. It is convenient to encode types of endpoints by $\pm$ superscripts as follows:

$$[p^-, q^+] := [p, q], \quad [p^+, q^-] := (p, q), \quad [p^+, q^+] := (p, q], \quad [p^-, q^-] := [p, q).$$

The endpoints p, q can also take the infinite values $\pm\infty$, but without superscripts.

Example 2.5.2 (interval module $\mathbb{I}(J)$)**.** For any interval $J \subset \mathbb{R}$, the *interval module* $\mathbb{I}(J)$ is the persistence module defined by the following vector spaces I_s and linear maps $i_s^t : I_s \to I_t$

$$I_s = \begin{cases} \mathbb{Z}_2, & \text{for } s \in J, \\ 0, & \text{otherwise}; \end{cases} \qquad i_s^t = \begin{cases} \text{id}, & \text{for } s, t \in J, \\ 0, & \text{otherwise} \end{cases} \quad \text{for any } s \leq t.$$

The direct sum $\mathbb{W} = \mathbb{U} \oplus \mathbb{V}$ of persistence modules $\mathbb{U}, \mathbb{V}$ is defined as the persistence module with the vector spaces $W_s = U_s \oplus V_s$ and linear maps $w_s^t = u_s^t \oplus v_s^t$.

We illustrate the abstract concepts above using geometric constructions of Topological Data Analysis. Let $f : X \to \mathbb{R}$ be a continuous function on a topological space. Its *sublevel* sets $X_s^f = f^{-1}((-\infty, s])$ form nested subspaces $X_s^f \subset X_t^f$ for any $s \leq t$. The inclusions of the sublevel sets respect compositions similarly to a dendrogram Δ in Definition 5.3.2.

On a metric space X with with a distance function $d : X \times X \to [0, +\infty)$, a typical example of a function $f : X \to \mathbb{R}$ is the distance to a finite set of points $A \subset X$. More specifically, for any point $p \in X$, let $f(p)$ be the distance from p to (a closest point of) A. For any $r \geq 0$, the preimage $X_r^f = f^{-1}((-\infty, r]) = \{q \in X \mid d(q, A) \leq r\}$ is the union of closed balls that have the radius r and centers at all points $p \in A$. For example, $X_0^f = f^{-1}((-\infty, 0]) = A$ and $X_{+\infty}^f = f^{-1}(\mathbb{R}) = X$.

If we consider any continuous function $f : X \to \mathbb{R}$, we have the inclusion $X_s^f \subset X_r^f$ for any $s \leq r$. Hence all sublevel sets X_s^f form a nested sequence of subspaces within X. The above construction of a *filtration* $\{X_s^f\}$ can be considered as a functor from $\mathbb{R}$ to the category of topological spaces. Below we discuss the most practically used case of dimension 0.

Example 2.5.3 (persistent homology)**.** For any topological space X, the 0-dimensional *homology* $H_0(X)$ is the vector space (with coefficients $\mathbb{Z}_2$) generated by all connected components of X. Let $\{X_s\}$ be any *filtration* of nested spaces, e.g. sublevel sets X_s^f based on a continuous function $f : X \to \mathbb{R}$. The inclusions $X_s \subset X_r$ for $s \leq r$ induce the linear maps between homology groups $H_0(X_s) \to H_0(X_r)$ and define the *persistent homology* $\{H_0(X_s)\}$, which satisfies the conditions of a persistence module from Definition 2.5.1.   ∎

If X is a finite set of m points, then $H_0(X)$ is the direct sum $\mathbb{Z}_2^m$ of m copies of $\mathbb{Z}_2$. The persistence modules that can be decomposed as direct sums of interval modules can be described in a very simple combinatorial way by persistence diagrams of dots in $\mathbb{R}^2$.

Definition 2.5.4 (persistence diagram PD($\mathbb{V}$))**.** Let a persistence module $\mathbb{V}$ be decomposed as a direct sum of interval modules from Example 2.5.2 : $\mathbb{V} \cong \bigoplus_{l \in L} \mathbb{I}(p_l^*, q_l^*)$, where $*$ is $+$ or $-$. The *persistence diagram* PD($\mathbb{V}$) is the multiset PD($\mathbb{V}$) $= \{(p_l, q_l) \mid l \in L\} \setminus \{p = q\} \subset \mathbb{R}^2$. ∎

The 0-dimensional persistent homology of a space X with a continuous function $f : X \to \mathbb{R}$ will be denoted by PD$\{H_0(X_s^f)\}$.

Definition 2.5.5 (a homomorphism of a degree δ between persistence modules)**.** Let $\mathbb{U}$ and $\mathbb{V}$ be persistent modules over $\mathbb{R}$. A *homomorphism* $\mathbb{U} \to \mathbb{V}$ of *degree* $\delta \in \mathbb{R}$ is a collection of linear maps $\phi_t : U_t \to V_{t+\delta}$, $t \in \mathbb{R}$, such that the diagram commutes for all $s \le t$.

$$
\begin{array}{ccc}
U_s & \xrightarrow{\ u_s^t\ } & U_t \\
\phi_s \downarrow & & \downarrow \phi_t \\
V_{s+\delta} & \xrightarrow{\ v_{s+\delta}^{t+\delta}\ } & V_{t+\delta}
\end{array}
$$

Let Hom$^\delta(\mathbb{U}, \mathbb{V})$ be all homomorphisms $\mathbb{U} \to \mathbb{V}$ of degree δ. Persistence modules $\mathbb{U}, \mathbb{V}$ are *isomorphic* if they have inverse homomorphisms $\mathbb{U} \to \mathbb{V} \to \mathbb{U}$ of degree 0. ∎

For a persistence module $\mathbb{V}$ with maps $v_s^t : V_s \to V_t$, the simplest example of a homomorphism of a degree $\delta \ge 0$ is $1_{\mathbb{V}}^\delta : \mathbb{V} \to \mathbb{V}$ defined by the maps $v_s^{s+\delta}$, $t \in \mathbb{R}$. So v_s^t defining the structure of $\mathbb{V}$ shift all vector spaces V_s by the difference $\delta = t - s$.

The concept of interleaved modules below is an algebraic generalization of a geometric perturbation of a set X in terms of (the homology of) its sublevel sets X_s.

Definition 2.5.6 (interleaving distance ID)**.** Persistence modules $\mathbb{U}$ and $\mathbb{V}$ are called δ-interleaved if there are homomorphisms $\phi \in$ Hom$^\delta(\mathbb{U}, \mathbb{V})$ and $\psi \in$ Hom$^\delta(\mathbb{V}, \mathbb{U})$ such that $\phi \circ \psi = 1_{\mathbb{V}}^{2\delta}$ and $\psi \circ \phi = 1_{\mathbb{U}}^{2\delta}$. The *interleaving distance* between the persistence modules $\mathbb{U}$ and $\mathbb{V}$ is ID($\mathbb{U}, \mathbb{V}$) $= \inf\{\delta \ge 0 \mid \mathbb{U}$ and $\mathbb{V}$ are δ-interleaved$\}$. ∎

If $f, g : X \to \mathbb{R}$ are continuous functions such that $||f - g||_\infty \le \delta$ in the L_∞-distance, the modules $H_k\{f^{-1}(-\infty, s]\}$, $H_k\{g^{-1}(-\infty, s]\}$ are δ-interleaved for any k [23]. The last conclusion extends to persistence diagrams for the bottleneck distance below.

Definition 2.5.7 (bottleneck distance BD)**.** Let multisets C, D contain finitely many points $(p, q) \in \mathbb{R}^2$, $p < q$, of finite multiplicity and all diagonal points $(p, p) \in \mathbb{R}^2$ of infinite multiplicity. For

$\delta \geq 0$, a δ-matching is a bijection $h : C \to D$ such that $|h(a) - a|_\infty \leq \delta$ in the L_∞-distance for any point $a \in C$. The *bottleneck* distance between persistence modules $\mathbb{U}, \mathbb{V}$ is $\mathrm{BD}(\mathbb{U}, \mathbb{V}) = \inf\{\delta \mid$ there is a δ-matching between $\mathrm{PD}(\mathbb{U}), \mathrm{PD}(\mathbb{V})\}$.   ∎

The original stability of persistence for sequences of sublevel sets found by Herbert Edelsbrunner [23] was extended as Theorem 2.5.8 to q-tame persistence modules. A persistence module $\mathbb{V}$ is q-tame if any non-diagonal square in the persistence diagram $\mathrm{PD}(\mathbb{V})$ contains only finitely many of points, see [19, section 2.8]. Any finitely decomposable persistence module is q-tame.

Theorem 2.5.8 (stability of persistence modules)**.** *[19, isometry theorem 4.11] Let $\mathbb{U}$ and $\mathbb{V}$ be q-tame persistence modules. Then* $\mathrm{ID}(\mathbb{U}, \mathbb{V}) = \mathrm{BD}(\mathrm{PD}(\mathbb{U}), \mathrm{PD}(\mathbb{V}))$, *where* ID *is the interleaving distance,* BD *is the bottleneck distance between persistence modules.*   ∎

Chapter 3

New compressed cover tree for k-nearest neighbor search

Section 3.4 and Section 3.9 were accepted to Proceedings of TopoInVis 2022 [31].

3.1 The k-nearest neighbor search and overview of results

Given a query set Q and a reference set R, which are subsets of some ambient space X with a distance d satisfying all metric axioms the k-nearest neighbors problem intends to discover all $k \geq 1$ nearest neighbors in the reference set R for all points from the query set Q. The simplest example is X being $\mathbb{R}^n$ with the Euclidean metric and the query set Q can be a single point or a subset of a larger set R.

The *exact* k-nearest neighbor problem intends to find exact (true) k-nearest neighbors of every query point q. Another probabilistic version of the k-nearest neighbor search [43, 62] aims to find exact k-nearest neighbors with a given probability. It should be noted that the probabilistic k-nearest neighbor problem can be simplified to k instances of 1-nearest-neighbors problem by splitting R into k subsets $R_1, ..., R_k$ and searching for nearest neighbors in each of the subsets.

On the other hand, the approximate version [2, 54, 1, 88] looks for its ϵ-approximate neighbor $r \in R$ satisfying $d(q, r) \leq (1 + \epsilon) d(q, \mathrm{NN}(q))$ for every query point $q \in Q$, where $\epsilon > 0$ is fixed and $\mathrm{NN}(q)$ is the exact first nearest neighbor of q.

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 $|Q| \cdot |R|$
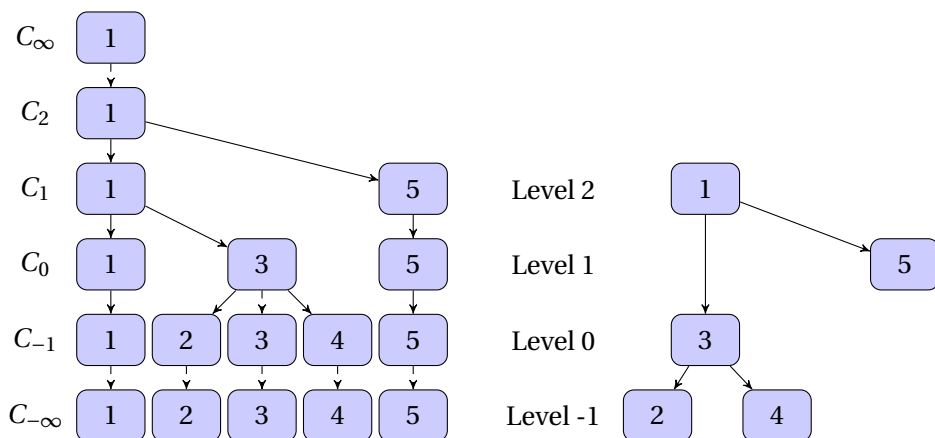
Figure 3.1: **Left:** an implicit cover tree from [9, Section 2] for a finite set of reference points $R = \{1, 2, 3, 4, 5\}$. **Right:** a new compressed cover tree in Definition 3.2.1 corrects the past worst-case complexity results for k-nearest neighbors search in R.

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. One of the first spacial data structures, a *quadtree* [36], 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. Since then more advanced new algorithms utilizing spatial data structures have positively impacted various related research areas such as a minimum spanning tree [6], range search [74], k-means clustering [74], and ray tracing [39]. The spacial data structures for finding nearest neighbors in the chronological order are k-means tree [38], R tree [5], ball tree [69], R^* tree [5], vantage-point tree [92], TV trees [59], X trees [8], principal axis tree [66], spill tree [60], cover tree [9], cosine tree [44], max-margin tree [77], cone tree [76] and others.

Definition 3.1.1 (k-nearest neighbor set NN_k)**.** For any fixed point $q \in Q$, let $d_1 \leq \cdots \leq d_{|R|}$ be ordered distances from q to all points of R, where $|R|$ is the number of points in R. For any $k \geq 1$, the k-nearest neighbor sets $\mathrm{NN}_k(q; R)$ consists of all points $u \in R$ with $d(q, u) \leq d_k$.   ■   ■

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

Expansion constants. Recall that $|\bar{B}(p,t)|$ is the number (if finite) of points in closed ball $\bar{B}(p,t)$.

Definition 3.1.3 (Locally finite space)**.** We say that a subset R of a metric space (X,d) is locally finite, if for all $p \in X$ and $t \in \mathbb{R}_+$ set $\bar{B}(p,t) \cap R$ is finite. ∎

The following definition of the expansion constant is borrowed from [9]. It also introduces the new minimized expansion constant. Minimized expansion constant is a discrete analog of doubling dimension.

Definition 3.1.4 (expansion constants c, c_m)**.** Let R be a locally 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 $p \in R$ and $t \geq 0$ [9]. The *minimized expansion constant* $c_m(R) = \inf\limits_{0<\xi} \inf\limits_{R \subseteq A \subseteq X} \sup\limits_{p \in A, t > \xi} \dfrac{|\bar{B}(p,2t) \cap A|}{|\bar{B}(p,t) \cap A|}$ where A is a locally finite set which covers R. ∎

Lemma 3.1.5 (properties of c_m)**.** *For any finite sets $R \subseteq A$ in a metric space, the following inequalities hold: $c_m(R) \leq c_m(A)$, $c_m(R) \leq c(R)$.* ∎

Note that both $c(R), c_m(R)$ is always defined when R is finite. We will show that a single outlier point can make the expansion constant $c(R)$ as large as $O(|R|)$. The set $R = \{1,2,\dots,n,2n+1\}$ of $|R| = n+1$ points has $c(R) = n+1$ because $\bar{B}(2n+1;n) = \{2n+1\}$ is a single point, while $\bar{B}(2n+1;2n) = R$ is the full set of $n+1$ points. On the other hand the same set R can be extended to a larger uniform set $A = \{1,2,\dots,2n-1,2n\}$ whose expansion constant $c(A) = 2$, therefore the minimized constant of the original set R becomes much smaller: $c_m(R) \leq c(A) = 2 < c(R) = n+1$.

The constant c from [9] equals to $2^{\dim_{KR}}$ from [54, Section 2.1]. In [54, Section 1.1] the doubling dimension $2^{\dim}$ is defined as a minimum value ρ such that any set X can be covered by 2^ρ sets

whose diameters are half of the diameter of X. The work [54] proves that $2^{\dim} \leq 2^n$ for any subset of $\mathbb{R}^n$. In Theorem 3.3.15 it is shown that $c_m(R) \leq 2^n$ for any a finite subset R of a normed vector space $\mathbb{R}^n$. It is not a surprise because $c_m(R)$ mimics $2^{\dim}$.

Navigating nets. In 2004, [54, Theorem 2.7] claimed that a navigating net can be constructed in time $O\big(2^{O(\dim_{KR}(R))}|R|(\log|R|)\log(\log|R|)\big)$ and all k-nearest neighbors of a query point q can be found in time $O(2^{O(\dim_{KR}(R \cup \{q\}))}(k + \log|R|)$, where $\dim_{KR}(R \cup \{q\})$ is the expansion constant defined above. All proofs and pseudo-codes were unfortunately omitted. The authors didn't reply to our request for details.

Modified navigating nets [24] were used in 2006 to claim the time $O(\log(n) + (1/\epsilon)^{O(1)})$ for the $(1 + \epsilon)$-approximate neighbors. All proofs and pseudo-codes were left out, also for the construction of the modified navigating net for the claimed time $O(|R| \cdot \log(|R|))$.

Cover trees. In 2006, [9] introduced a cover tree inspired by the navigating nets [54]. This cover tree was designed to prove a worst-case bound for the nearest neighbor search in terms of the size $|R|$ of a reference set R and the expansion constant $c(R)$ of Definition 3.1.4. Assume that a cover tree is already constructed on set R. Then [9, Theorem 5] claims that nearest neighbor of any query point $q \in Q$ could be found in time $O(c(R)^{12} \cdot \log|R|)$. In 2015, [27, Section 5.3] pointed out that the proof of [9, Theorem 5] contains a crucial gap, now have been confirmed by a specific dataset in Counterexample 3.4.7. The time complexity result of the cover tree construction algorithm [9, Theorem 6] had a similar issue, the gap of which is exposed rigorously in Counterexample 3.4.5.

Further studies in cover trees. A noteworthy paper on cover trees [53] introduced a new probabilistic algorithm for the nearest neighbor search, as well as corrected the pseudo-code of the cover tree construction algorithm of [9, Algorithm 2]. Later in 2015, a new, more efficient implementation of cover tree was introduced in [46]. However, no new time-complexity results were proven. A study [47] explored connections between modified navigating nets [24] and cover trees [9]. Multiple papers [10, 78, 26] studied possibility of solving k-nearest neighbor problem (Problem 3.1.2) by using cover tree on both, the query set and the reference set, for further details see Section 3.9.

Compressed cover tree. In this work we correct the past issues of the standard single-tree cover tree approach [9] by applying a new compressed cover tree $\mathcal{T}(R)$ of Definition 3.2.1, which can be constructed on any finite set R with a metric d. In Theorem 3.5.10 it will be shown that a compressed cover tree $\mathcal{T}(R)$ can be built in time $O(c_m(R)^8 \cdot c(R)^2 \cdot \log_2(|R|) \cdot |R|)$.

Table 3.1: Results for exact k-nearest neighbors of one query point $q \in X$ using hidden classic expansion constant $c(R)$ of Definition 3.1.4 or KR-type constant $2^{\dim_{KR}}$ [54, Section 2.1] and assuming that all data structures are already built. Note that $2^{\dim_{KR}}$ corresponds to $c(R)^{O(1)}$

Data structure, reference	time complexity	space	proofs
Navigating nets [54]	$O\big(2^{O(\dim_{KR})}(\log(|R|) + k)\big)$ for $k \geq 1$ [54, Theorem 2.7]	$O(2^{O(\dim)} \cdot |R|)$	Not available
Cover tree [9]	$O\big(c(R)^{O(1)}\log(|R|)\big)$ for $k = 1$, [9, Theorem 5]	$O(|R|)$	Counterexample 3.4.7 shows that the proof of Theorem 5 is incorrect
Compressed cover tree [Definition 3.2.1]	$O\big(c(R)^{O(1)} \cdot \log(k) \cdot (\log(|R|) + k)\big)$	$O(|R|)$, Lem 3.2.2	Theorem 3.7.14

Table 3.2: Results for building data structures with hidden classic expansion constant $c(R)$ of Definition 3.1.4 or KR-type constant $2^{\dim_{KR}}$ [54, Section 2.1]

Data structure, reference	time complexity	space	proofs
Navigating nets [54]	$O\big(2^{O(\dim_{KR})} \cdot |R|\log(|R|)\log(\log(|R|))\big)$, [54, Theorem 2.6]	$O(2^{O(\dim)}|R|)$	Not available
Cover tree [9]	$O(c(R)^{O(1)} \cdot |R| \cdot \log(|R|))$, [9, Theorem 6]	$O(|R|)$	Counterexample 3.4.5 shows that the proof of Theorem 6 is incorrect
Compressed cover tree [dfn 3.2.1]	$O\big(c(R)^{O(1)} \cdot |R| \cdot \log(R)\big)$	$O(|R|)$, Lem 3.2.2	Corollary 3.5.11

Table 3.3: Results for exact k-nearest neighbors of one point q using hidden $c_m(R)$ or dimensionality constant $2^{\dim}$ [54, Section 1.1] assuming that all structures are built.

Data structure, reference	time complexity	space	proofs
Navigating nets [54]	$O\big(2^{O(\dim)} \cdot \log(\Delta) + |\bar{B}(q, O(1) \cdot d(q,R))|\big)$, for $k = 1$, [54, Theorem 2.3]	$O(2^{O(\dim)} \cdot |R|)$	a proof outline in [54, Theorem 2.3]
Compressed cover tree [dfn 3.2.1]	$O\big(c_m(R)^{O(1)} \cdot \log(k) \cdot (\log(|\Delta|) + |\bar{B}(q, O(1) \cdot d_k(q,R))|)\big)$	$O(|R|)$, Lem 3.2.2	Corollary 3.7.9

Table 3.4: Building data structures with hidden $c_m(R)$ or dimensionality constant $2^{\dim}$ [54, Section 1.1]

Data structure, reference	time complexity	space	proofs
Navigating nets [54]	$O\big(2^{O(\dim)} \cdot |R| \cdot \log(\Delta) \cdot \log(\log((\Delta)))\big)$	$O(2^{O(\dim)} \cdot |R|)$	[54, Theorem 2.5]
Compressed cover tree [dfn 3.2.1]	$O\big(c_m(R)^{O(1)} \cdot |R| \log(\Delta(|R|))\big)$	$O(|R|)$, Lem 3.2.2	Theorem 3.5.7

The past gap in time complexity result [9, Theorem 1] of nearest neighborhood search is tackled by introducing a new method, Algorithm 3.7.2 which differs from the original method [9, Algorithm 1] by having an additional block of code. The extra block eliminates the issue of having too many successive iterations, in case where the query point q is disproportionately far away from the remaining candidate set R_i on some level i. By means of the new block, Lemma 3.7.13 guarantees that the number of iterations of Algorithm 3.7.2 is bounded by $O(c(R)^2 \log_2(|R|))$. This new lemma replaces the old result [9, Lemma 4.3], which had a similar bound for the number of explicit levels of a cover tree, see Definition 3.4.2, but couldn't be used to estimate the number of iterations of [9, Algorithm 1] due to Counterexample 3.4.7.

Assume that a compressed cover tree $\mathcal{T}(R)$ is already constructed on the reference set R. The main result of this work Theorem 3.7.14 shows that k-nearest neighbors of a query node q can be found in time of

$$O\Big(c(R \cup \{q\})^2 \cdot \log_2(k) \cdot \big((c_m(R))^{10} \cdot \log_2(|R|) + c(R \cup \{q\}) \cdot k\big)\Big).$$

Recall that $c(R)$ can potentially become as large as $O(|R|)$ when R is not uniformly distributed. Our second result Corollary 3.7.9 bounds the time complexity of the new k-nearest neighborhood algorithm through making use of only minimized expansion constant $c_m(R)$ of Definition 3.1.4 and the aspect ratio $\Delta(R)$ of Definition 3.2.14 as parameters. The advantage of this is that the parameters are less dependent on the noise of the data set, in most cases $\Delta(R)$ is relatively small and $c_m(R)$ depends mostly on the dimension of the ambient space X. It is shown that k-nearest neighbors of q in a reference set R can be computed in time of

$$O\Big((c_m(R))^{10} \cdot \log_2(k) \cdot \log_2(\Delta(R)) + |\bar{B}(q, 5d_k(q, R))| \cdot \log_2(k)\Big),$$

where $d_k(q, R)$ is the distance of q to its kth nearest neighbor and $|\bar{B}(q, 5d_k(q, R))|$ is the number of points in the ball $\bar{B}(q, 5d_k(q, R))$.

3.2 Compressed cover tree

This section introduces a new data structure, compressed cover tree in Definition 3.2.1 that will be used to solve Problem 3.1.2. This section also contains important results regarding expansion constants: Lemma 3.2.8 and Lemma 3.2.10. Given a δ-spare finite metric space R, Lemma 3.2.8 shows that the number of points of R in a closed ball $\bar{B}(p, t)$ can be bound by $c_m(S)^\mu$, where μ depends on ratio $\frac{t}{\delta}$. On the other hand Lemma 3.2.10 can be used to show that if there exists points p, q in a finite metric space R satisfying $2r < d(p, q) \geq 3r$ for some $r \in \mathbb{R}$, then $|bar B(q, 4r)| \geq (1 + \frac{1}{c(R)^2})|\bar{B}(q, r)|$.

Definition 3.2.1 (a compressed cover tree $\mathcal{T}(R)$). Let R be a finite set in a metric space (X, d). *A compressed cover tree $\mathcal{T}(R)$ has the vertex set R with a root $r \in R$ and a level function $l : R \to \mathbb{Z}$ satisfying the conditions below.*

(3.2.1a) *Root condition*: the level of the root node r is $l(r) \geq 1 + \max_{p \in R \setminus \{r\}} l(p)$.

(3.2.1b) *Covering condition*: for every non-root node $q \in R \setminus \{r\}$, we select a unique *parent* p and a level $l(q)$ such that $d(q, p) \leq 2^{l(q)+1}$ and $l(q) < l(p)$; this parent node p has a single link to its *child* node q in the tree $\mathcal{T}(R)$.

(3.2.1c) *Separation condition*: for $i \in \mathbb{Z}$, the *cover set* $C_i = \{p \in R \mid l(p) \geq i\}$ has $d_{\min}(C_i) = \min_{p \in C_i} \min_{q \in C_i \setminus \{p\}} d(p, q) > 2^i$.

Since there is a 1-1 map between all points of R and all nodes of $\mathcal{T}(R)$, the same notation p can refer to a point in the set R or to a node of the tree $\mathcal{T}(R)$. Set $l_{\max} = 1 + \max_{p \in R \setminus \{r\}} l(p)$ and $l_{\min} = \min_{p \in R} l(p)$. For any node $p \in \mathcal{T}(R)$, Children(p) denotes the set consisting of all children of p, including p itself. For any node $p \in \mathcal{T}(R)$, define the *node-to-root* path as a unique sequence of nodes $w_0, \ldots, w_m$ such that $w_0 = p$, w_m is the root and w_{j+1} is the parent of w_j for $j = 0, \ldots, m-1$. A node $q \in \mathcal{T}(R)$ 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) be the set of all descendants of p, including p. ∎

Lemma 3.2.2 (Linear space of $\mathcal{T}(R)$). *Let (R, d) be a finite metric space. Then any cover tree $\mathcal{T}(R)$ from Definition 3.2.1 takes $O(|R|)$ space.*

Proof. Since $\mathcal{T}(R)$ is a tree , both its vertex set and its edge set contain at most $|R|$ nodes. Therefore $\mathcal{T}(R)$ takes at most $O(|R|)$ space. $\qquad\square$
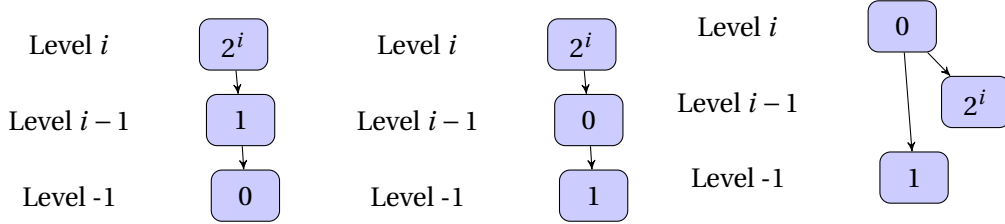
Figure 3.2: Compressed cover trees $\mathcal{T}(R)$ from Definition 3.2.1 for $R = \{0, 1, 2^i\}$.

Example 3.2.3 ($\mathcal{T}(R)$ in Fig. 3.3)**.** Let $(\mathbb{R}, d = |x - y|)$ be the real line with euclidean metric. Let $R = \{1, 2, 3, ..., 15\}$ be its finite subset. Fig. 3.3 shows a compressed cover tree on the set R with the root $r = 8$. The cover sets of $\mathcal{T}(R)$ are $C_{-1} = \{1, 2, 3, ..., 15\}$, $C_0 = \{2, 4, 6, 8, 10, 12, 14\}$, $C_1 = \{4, 8, 12\}$ and $C_2 = \{8\}$. We check the conditions of Definition 3.2.1.

- Root condition (3.2.1a): since $\max_{p \in R \setminus \{8\}} d(p, 8) = 7$ and $\lceil \log_2(7) \rceil - 1 = 2$, the root can have the level $l(8) = 2$.

- Covering condition (3.2.1b) : for any $i \in -1, 0, 1, 2$, let p_i be arbitrary point having $l(p_i) = i$. Then we have $d(p_{-1}, p_0) = 1 \le 2^0$, $d(p_0, p_1) = 2 \le 2^1$ and $d(p_1, p_2) = 4 \le 2^2$.

- Separation condition (3.2.1c) : $d_{\min}(C_{-1}) = 1 > \frac{1}{2} = 2^{-1}$, $d_{\min}(C_0) = 2 > 1 = 2^0$, $d_{\min}(C_1) = 4 > 2 = 2^1$. $\qquad\blacksquare$

A cover tree was defined in [9, Section 2] as a tree version of a navigating net from [54, Section 2.1]. For any index $i \in \mathbb{Z} \cup \{\pm\infty\}$, the level i set of this cover tree coincides with the cover set C_i
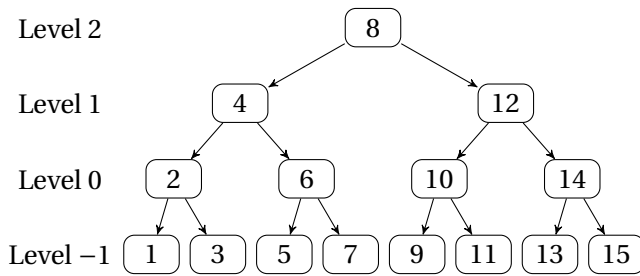
Figure 3.3: Compressed cover tree $\mathcal{T}(R)$ on the set R in Example 3.2.3 with root 16.
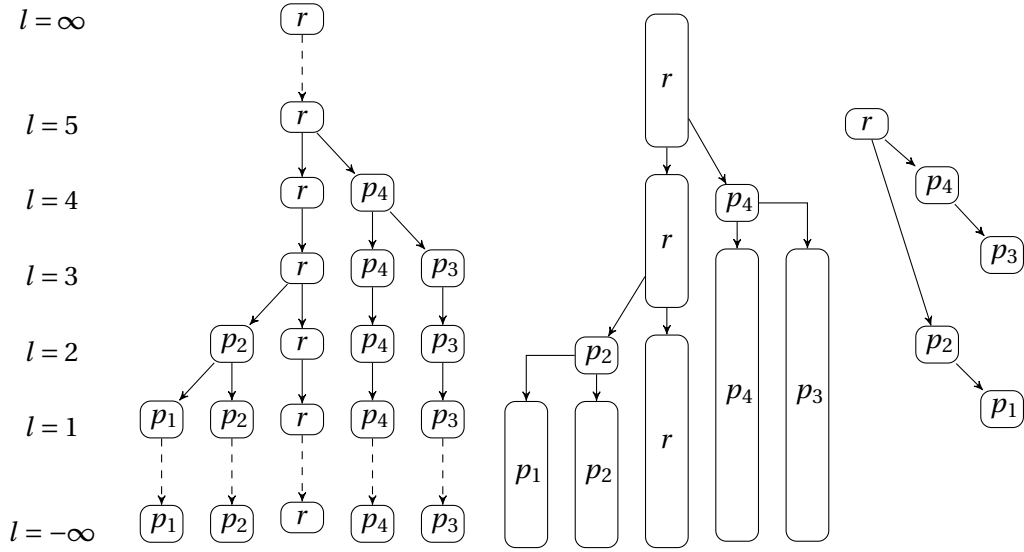
Figure 3.4: A comparison of past cover trees and a new tree in Example 3.2.4. **Left:** an implicit cover tree contains infinite repetitions. **Middle:** an explicit cover tree. **Right:** a compressed cover tree from Definition 3.2.1 includes each point once.

above, which can have nodes at different levels in Definition 3.2.1. Any point $p \in C_i$ has a single parent in the set C_{i+1}, which satisfied conditions (3.2.1b,c). [9, Section 2] referred to this original tree as an implicit representation of a cover tree. Such a tree in Figure 3.4 (left) contains infinitely many repetitions of every point $p \in R$ in long branches and will be called an *implicit cover tree*.

Since an implicit cover tree is formally infinite, for practical implementations, the authors of [9] had to use another version that they named an explicit representation of a cover tree. We call this version an *explicit cover tree*. Here is the full defining quote at the end of [9, Section 2]: "The explicit representation of the tree coalesces all nodes in which the only child is a self-child". In an explicit cover tree, if a subpath of every node-to-root path consists of all identical nodes without other children, all these identical nodes collapse to a single node, see Figure 3.4 (middle).

Since an explicit cover tree still contains repeated points, Definition 3.2.1 is well-motivated by the aim to include every point only once, which saves memory and simplifies all subsequent algorithms, see Fig. 3.4 (right).

Example 3.2.4 (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 a compressed cover tree $\mathcal{T}(R)$ by choosing the level $l(p_i) = i$ and by setting the root r to be the parent of both p_2 and p_4, p_4 to be the parent of p_3, and p_2 to be the parent of p_1. Then $\mathcal{T}(R)$ satisfies all the conditions of Definition 3.2.1, see a comparison of the three cover trees in Fig. 3.4. ∎
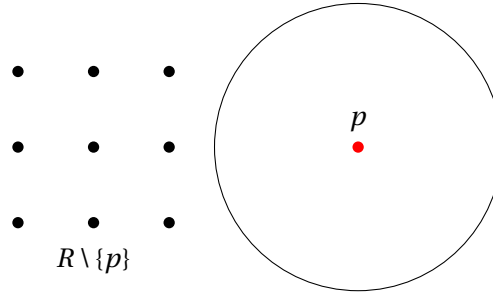
Figure 3.5: Example 3.2.5 describes a set R with a big expansion constant $c(R)$. Let $R \setminus \{p\}$ be a finite subset of a unit square lattice in $\mathbb{R}^2$, but a point p is located far away from $R \setminus \{p\}$ at a distance larger than $\text{diam}(R \setminus \{p\})$. Definition 3.1.4 implies that $c(R) = |R|$.

Even a single outlier point can make the expansion constant big. Consider set $R = \{1, 2, ..., n - 1, 2n\}$ for some $n \in \mathbb{Z}_+$. Since $|\bar{B}(2n, n)| = 1$ and $|\bar{B}(2n, n)| = |R|$, we have $c(R) = |R|$. Example 3.2.5 shows that expansion constant of a set R can be as big as $|R|$.

Example 3.2.5 (one outlier can make the expansion constant big)**.** Let R be a finite metric space and $p \in R$ satisfy $d(p, R \setminus \{t\}) > \text{diam}(R \setminus \{p\})$. Since $\bar{B}(p, 2d(p, R \setminus \{t\}) = R$, $\bar{B}(p, d(p, R \setminus \{t\}) = \{p\}$, we get $c(R) = N$, see Fig. 3.5. ∎

Example 3.2.6 shows that the minimized expansion can be significantly smaller than the original expansion constant.

Example 3.2.6 (minimized expansion constants)**.** Let $(\mathbb{R}, d)$ be the Euclidean line. For an integer $n > 10$, consider the finite sets $R = \{2^i \mid i \in [1, n]\}$ and let $Q = \{i \mid i \in [1, 2^n]\}$. If $0 < \epsilon < 10^{-9}$, then $\bar{B}(2^n, 2^{n-1} - \epsilon) = \{2^n\}$ and $\bar{B}(2^n, 2(2^{n-1} - \epsilon)) = R$, so $c(R) = n$. For any $q \in Q$ and any $t \in \mathbb{R}$, we have the balls $\bar{B}(q, t) = \mathbb{Z} \cap [q - t, q + t]$ and $\bar{B}(q, 2t) = \mathbb{Z} \cap [q - 2t, q + 2t]$, so $c(Q) \leq 4$. Lemma 3.1.5 implies that $c_m(R) \leq c_m(Q) \leq c(Q) \leq 4$. ∎

Lemma 3.2.7 provides an upper bound for a distance between a node and its descendants.
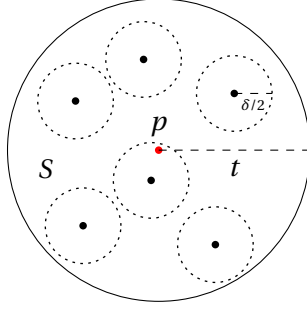
Figure 3.6: This volume argument proves Lemma 3.2.8. By using an expansion constant, we can find an upper bound for the number of smaller balls of radius $\frac{\delta}{2}$ that can fit inside a larger $\bar{B}(p,t)$.

Lemma 3.2.7 (a distance bound on descendants). *Let R be a finite subset of an ambient space X with a metric d. In a compressed cover tree $\mathcal{T}(R)$, let q be any descendant of a node p. Let the node-to-root path S of q contain a node u satisfying $u \in \text{Children}(p) \setminus \{p\}$. Then $d(p,q) \le 2^{l(u)+2} \le 2^{l(p)+1}$.* ∎

Proof. Let $(w_0, ..., w_m)$ be a subpath of the node-to-root path for $w_0 = q$, $w_{m-1} = u$, $w_m = p$. Then $d(w_i, w_{i+1}) \le 2^{l(w_i)+1}$ for any i. The first required inequality follows from the triangle inequality below:

$$d(p,q) \le \sum_{j=0}^{m-1} d(w_j, w_{j+1}) \le \sum_{j=0}^{m-1} 2^{l(w_j)+1} \le \sum_{t=l_{\min}}^{l(u)+1} 2^t \le 2^{l(u)+2}$$

Finally, $l(u) \le l(p) - 1$ implies that $d(p,q) \le 2^{l(p)+1}$. □

Lemma 3.2.8 uses the idea of [26, Lemma 1] to show that if S is a δ-sparse subset of a metric space X, then S has at most $(c_m(S))^\mu$ points in the ball $\bar{B}(p,r)$, where $c_m(S)$ is the minimized expansion constant of S, while μ depends on δ, r.

Lemma 3.2.8 (packing). *Let S be a finite δ-sparse set in a metric space (X,d), so $d(a,b) > \delta$ for all $a, b \in S$. Then, for any point $p \in X$ and any radius $t > \delta$, we have $|\bar{B}(p,t) \cap S| \le (c_m(S))^\mu$, where $\mu = \lceil \log_2(\frac{4t}{\delta} + 1) \rceil$.* ∎

Proof. Assume that $d(p,q) > t$ for any point $q \in S$. Then $\bar{B}(p,t) \cap S = \emptyset$ and the lemma holds trivially. Otherwise $\bar{B}(p,t) \cap S$ is non-empty. By Definition 3.1.4 of a minimized expansion constant, for any $\epsilon > 0$, we can always find a set A such that $S \subseteq A \subseteq X$, $\xi = \frac{\delta}{4}$ and

$$|B(q,2s) \cap A| \le (c_m(S) + \epsilon) \cdot |B(q,s) \cap A| \tag{3.2.8.1}$$

for any $q \in A$ and $s > \xi = \frac{\delta}{4}$. Note that for any $u \in \bar{B}(p,t) \cap S$ we have $\bar{B}(u, \frac{\delta}{2}) \subseteq \bar{B}(u, t + \frac{\delta}{2})$. Therefore, for any point $q \in \bar{B}(p,t) \cap S$, we get

$$\bigcup_{u \in \bar{B}(p,t) \cap S} \bar{B}(u, \frac{\delta}{2}) \subseteq \bar{B}(p, t + \frac{\delta}{2}) \subseteq \bar{B}(q, 2t + \frac{\delta}{2})$$

Since all the points of S were separated by δ, we have

$$|\bar{B}(p,t) \cap S| \cdot \min_{u \in \bar{B}(p,t) \cap S} |\bar{B}(u, \frac{\delta}{2}) \cap A| \leq \sum_{u \in \bar{B}(p,t) \cap S} |\bar{B}(u, \frac{\delta}{2}) \cap A| \leq |\bar{B}(q, 2t + \frac{\delta}{2}) \cap A|$$

In particular, by setting $q = \mathrm{argmin}_{a \in S \cap \bar{B}(p,t)} |\bar{B}(a, \frac{\delta}{2})|$, we get:

$$|\bar{B}(p,t) \cap S| \cdot |\bar{B}(q, \frac{\delta}{2}) \cap A| \leq |\bar{B}(q, 2t + \frac{\delta}{2}) \cap A| \tag{3.2.8.2}$$

Inequality (3.2.8.1) applied μ times for the radii $s_i = \dfrac{2t + \frac{\delta}{2}}{2^i}$, $i = 1, ..., \mu$, implies that:

$$|\bar{B}(q, 2t + \frac{\delta}{2}) \cap A| \leq (c_m(S) + \epsilon)^\mu |\bar{B}(q, \frac{2t + \frac{\delta}{2}}{2^\mu}) \cap A| \leq (c_m(S) + \epsilon)^\mu |\bar{B}(q, \frac{\delta}{2}) \cap A|. \tag{3.2.8.3}$$

By combining inequalities (3.2.8.2) and (3.2.8.3), we get

$$|\bar{B}(p,t) \cap S| \leq \frac{|\bar{B}(q, 2t + \frac{\delta}{2}) \cap A|}{|\bar{B}(q, \frac{\delta}{2}) \cap A|} \leq (c_m(S) + \epsilon)^\mu.$$

The required inequality is obtained by letting $\epsilon \to 0$. $\qquad\square$

[54, Section 1.1] defined $\dim(X)$ of a space (X, d) as the minimum number m such that every set $U \subseteq X$ can be covered by 2^m sets whose diameter is a half of the diameter of U. If U is finite, an easy application of Lemma 3.2.8 for $\delta = \frac{r}{2}$ shows that $\dim(X) \leq \sup_{A \subseteq X} (c_m(A))^4 \leq \sup_{A \subseteq X} \inf_{A \subseteq B \subseteq X} (c(B))^4$, where A and B are finite subsets of X.

Let $T(R)$ be an implicit cover tree of [9] on a finite set R. [9, Lemma 4.1] showed that the number of children of any node $p \in T(R)$ has the upper bound $(c(R))^4$. Lemma 3.2.9 generalizes [9, Lemma 4.1] for a compressed cover tree.

Lemma 3.2.9 (width bound). *Let R be a finite subset of a metric space (X, d). For any compressed*

cover tree $\mathcal{T}(R)$, any node p we have

$$\{q \in \text{Children}(p) \mid l(q) = i\} \cup \{p\} \le (c_m(R))^4,$$

where $c_m(R)$ is the minimized expansion constant of the set R. ∎

Proof. By the covering condition of $\mathcal{T}(R)$, any child q of p located on the level i has $d(q, p) \le 2^{i+1}$. Then the number of children of the node p at level i at most $|\bar{B}(p, 2^{i+1})|$. The separation condition in Definition 3.2.1 implies that the set C_i is a 2^i-sparse subset of X. We apply Lemma 3.2.8 for $t = 2^{i+1}$ and $\delta = 2^i$. Since $4 \cdot \frac{t}{\delta} + 1 \le 4 \cdot 2 + 1 \le 2^4$, we get $|\bar{B}(q, 2^{i+1}) \cap C_i| \le (c_m(C_i))^4$. Lemma 3.1.5 implies that $(c_m(C_i))^4 \le (c_m(R))^4$, so the upper bound is proved. □

Lemma 3.2.10 (growth bound)**.** *Let (A, d) be a finite metric space, let $q \in A$ be an arbitrary point and let $r \in \mathbb{R}$ be a real number. Let $c(A)$ be the expansion constant from Definition 3.1.4. If there exists a point $p \in A$ such that $2r < d(p, q) \le 3r$, then $|\bar{B}(q, 4r)| \ge (1 + \frac{1}{c(A)^2}) \cdot |\bar{B}(q, r)|$.*

Proof. Since $\bar{B}(q, r) \subset \bar{B}(p, 3r + r)$, we have $|\bar{B}(q, r)| \le |\bar{B}(q, 4r)| \le c(A)^2 \cdot |\bar{B}(p, r)|$. And since $\bar{B}(p, r)$ and $\bar{B}(q, r)$ are disjoint and are subsets of $\bar{B}(q, 4r)$, we have

$$|\bar{B}(q, 4r)| \ge |\bar{B}(q, r)| + |\bar{B}(p, r)| \ge |\bar{B}(q, r)| + \frac{|\bar{B}(q, r)|}{c(A)^2} \ge (1 + \frac{1}{c(A)^2}) \cdot |\bar{B}(q, r)|,$$

which proves the claim. □

Lemma 3.2.11 (extended growth bound)**.** *Let (A, d) be a finite metric space, let $q \in A$ be an arbitrary point. Let $p_1, ..., p_n$ be a sequence of distinct points in R, in such a way that for all $i \in \{2, ..., n\}$ we have $4 \cdot d(p_i, q) \le d(p_{i+1}, q)$. Then*

$$|\bar{B}(q, \frac{4}{3} \cdot d(q, p_n))| \ge (1 + \frac{1}{c(A)^2})^n \cdot |\bar{B}(q, \frac{1}{3} \cdot d(q, p_1))|.$$

Proof. Let us prove this by induction. In basecase $n = 1$ define $r = \frac{d(q, p_m)}{3}$. Now by Lemma 3.2.10 we have

$$|\bar{B}(q, \frac{4}{3}d(q, p_1))| = |\bar{B}(q, 4r)| \ge (1 + \frac{1}{c(A)^2}) \cdot |\bar{B}(q, r)| = |\bar{B}(q, \frac{1}{3}d(q, p_1))|.$$

Assume now that the claim holds for some $i = m$ and let $p_1, ..., p_{m+1}$ be a sequence satisfying the condition of Lemma 3.2.11. By induction assumption we have $|\bar{B}(q, \frac{4}{3}d(q, p_m))| \ge (1 + \frac{1}{c(A)^2})^m \cdot$

$|\bar{B}(q, \frac{1}{3}d(q, p_1))|$. Let us pick $r = \frac{d(q,p_{m+1})}{3}$. Then we have:

$$
\begin{aligned}
|\bar{B}(q, \frac{4}{3} \cdot d(q, p_{m+1}))| &\geq (1 + \frac{1}{c(A)^2}) \cdot |\bar{B}(q, \frac{1}{3} \cdot d(q, p_{m+1}))| \\
&\geq (1 + \frac{1}{c(A)^2}) \cdot |\bar{B}(q, \frac{4}{3} \cdot d(q, p_m))| \\
&\geq (1 + \frac{1}{c(A)^2}) \cdot (1 + \frac{1}{c(A)^2})^m \cdot |\bar{B}(q, \frac{1}{3} \cdot d(q, p_1))| \\
&\geq (1 + \frac{1}{c(A)^2})^{m+1} \cdot |\bar{B}(q, \frac{1}{3} \cdot d(q, p_1))|
\end{aligned}
$$

which proves the claim. $\qquad\square$

Lemma 3.2.12. *For every $x \in \mathbb{R}$ satisfying $x \geq 2$, the following inequality holds:*

$$
x^2 \geq \frac{1}{\log_2(1 + \frac{1}{x^2})}.
$$

Proof. Let ln be natural logarithm. Note first that for any $u > 0$ we have:

$$
\frac{u}{u+1} = \int_0^u \frac{dt}{u+1} \leq \int_0^u \frac{dt}{t+1} = \ln(u+1).
$$

By setting $u = \frac{1}{x^2} > 0$ we get:

$$
\log_2(1 + \frac{1}{x^2}) = \frac{\ln(\frac{1}{x^2})}{\ln(2)} \geq \frac{1}{\ln(2)} \cdot \frac{\frac{1}{x^2}}{\frac{1}{x^2}+1} = \frac{1}{\ln(2)} \cdot \frac{1}{x^2+1}.
$$

Let us now show that for $x \geq 2$ we have: $\frac{1}{\ln(2)} \cdot \frac{1}{x^2+1} \geq \frac{1}{x^2}$. Note first that $4 \geq \frac{\ln(2)}{1-\ln(2)}$. Since $x \geq 2$ we have $x^2 \geq \frac{\ln(2)}{1-\ln(2)}$. Therefore $x^2 - \ln(2) \cdot x^2 \geq \ln(2)$ and $x^2 \geq \ln(2) \cdot (1 + x^2)$. It follows that $\frac{1}{\ln(2)} \frac{1}{1+x^2} \geq \frac{1}{x^2}$, which proves the claim.

$\qquad\square$

Definition 3.2.13 (the height of a compressed cover tree)**.** For a compressed cover tree $\mathcal{T}(R)$ on a finite set R, the *height set* is $H(\mathcal{T}(R)) = \{i \mid C_{i-1} \neq C_i\} \cup \{l_{\max}, l_{\min}\}$. The size $|H(\mathcal{T}(R))|$ of this set is called the *height* of $\mathcal{T}(R)$. $\qquad\blacksquare$

The new concept of the height $|H(\mathcal{T})|$ will justify a near-linear parameterized worst-case complexity in Theorem 3.7.14. By condition (3.2.1b), the height $|H(\mathcal{T}(R))|$ counts the number of levels

i whose cover sets C_i include new points that were absent on higher levels. Then $|H(\mathcal{T})| \leq |R|$ as any point can be alone at its own level.

Definition 3.2.14. For any finite metric set R with a metric d, the *diameter* is

$$\text{diam}(R) = \max_{p \in R} \max_{q \in R} d(p, q).$$

The *aspect ratio* [54] is $\Delta(R) = \dfrac{\text{diam}(R)}{d_{\min}(R)}$.   ∎

Lemma 3.2.15. *Any finite set R has the bound $|H(\mathcal{T}(R))| \leq 1 + \log_2(\Delta(R))$.*   ∎

Proof of Lemma *3.2.15.* We have $|H(\mathcal{T}(R))| \leq l_{\max} - l_{\min} + 1$ by Definition 3.2.13. We estimate $l_{\max} - l_{\min}$ as follows.

Let $p \in R$ be a point such that $\text{diam}(R) = \max_{q \in R} d(p, q)$. Then R is covered by the closed ball $\bar{B}(p; \text{diam}(R))$. Hence the cover set C_i at the level $i = \log_2(\text{diam}(R))$ consists of a single point p. The separation condition in Definition 3.2.1 implies that $l_{\max} \leq \log_2(d_{\max}(R))$. Since any distinct points $p, q \in R$ have $d(p, q) \geq d_{\min}(R)$, the covering condition implies that no new points can enter the cover set C_i at the level $i = [\log_2(d_{\min}(R))]$, so $l_{\min} \geq \log_2(d_{\min}(R))$. Then $|H(\mathcal{T}(R))| \leq 1 + l_{\max} - l_{\min} \leq 1 + \log_2(\frac{\text{diam}(R)}{d_{\min}(R)})$.   □

If the aspect ratio $\Delta(R) = O(\text{Poly}(|R|))$ polynomially depends on the size $|R|$, then $|H(\mathcal{T}(R))| \leq O(\log(|R|))$. Lemma 3.2.10 corresponds [9, Lemma 4.2] with slightly modified notation.

3.3   Minimized expansion constant in Euclidean space $\mathbb{R}^n$

The main result of this section is Theorem 3.3.15, in which we will show that for any finite subset R of normed vector space $(\mathbb{R}^n, \|\cdot\|)$, the minimized expansion constant from Definition 3.1.4 satisfies

$$c_m(R) = \inf_{0 < \xi} \inf_{R \subseteq A \subseteq \mathbb{R}^n} \sup_{p \in A, t > \xi} \frac{|\bar{B}(p, 2t) \cap A|}{|\bar{B}(p, t) \cap A|} \leq 2^n.$$

The proof of Theorem 3.3.15 is based on the volume argument. We briefly explain the idea before giving the proof later. For this purpose, we recall the definition of the Lebesgue measure in Definition 3.3.2.

In Definition 3.3.5 we define concepts of grid $G(\delta) = \delta \cdot \mathbb{Z}^n$ and cubic regions $\bar{V}(p, \delta) = p + [-\frac{\delta}{2}, \frac{\delta}{2}]^n$. For every $\delta > 0$ we define grid extension $U(\delta)$ of R as set $U(\delta) = (G(\delta) \setminus f(R)) \cup R$, where $f : R \to G(\delta)$ is used to replace points of R with their nearest neighbors in grid $G(\delta)$.

Note that ξ in the definition of $c_m(R)$ acts as a low bound for radius $t > \xi$. Let $\gamma > 0$ be a constant, that depends on dimension n and norm $\|\cdot\|$. In Lemma 3.3.13 it is shown that if δ satisfies $0 < \delta < \frac{\xi}{\gamma}$, then for any $p \in U(\delta)$ and $t > \xi$ we can bound $|\bar{B}(p,t) \cap U(\delta)|$ as follows:

$$\frac{\mu(\bar{B}(p, t - \delta \cdot \gamma))}{\delta^n} \leq |\bar{B}(p,t) \cap U(\delta)| \leq \frac{\mu(\bar{B}(p, t + \delta \cdot \gamma))}{\delta^n}.$$

Therefore

$$\frac{|\bar{B}(p, 2t) \cap U(\delta)|}{|\bar{B}(p,t) \cap U(\delta)|} \leq \frac{\mu(\bar{B}(p, 2t + \delta \cdot \gamma))}{\mu(\bar{B}(p, t - \delta \cdot \gamma))}.$$

Now since this inequality is satisfied for any $\delta > 0$, we can choose arbitrary dense grid extension $U(\delta)$. It will be shown that when $\delta \to 0$, then

$$\frac{\mu(\bar{B}(p, 2t + \delta \cdot \gamma))}{\mu(\bar{B}(p, t - \delta \cdot \gamma))} \to 2^n,$$

we conclude that $c_m(R) \leq 2^n$.

Definition 3.3.1 (Normed vector space on $\mathbb{R}^n$, [80])**.** Let $\mathbb{R}^n$ be an n-dimensional space. A norm is a function $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ satisfying the following properties:

1. Non-negativity, $\|x\| \geq 0$.

2. It is positive on nonzero vectors, that is $\|x\| = 0$ implies $x = 0$.

3. For every vector $x \in \mathbb{R}^n$, and every scalar $a \in \mathbb{R}$: $\|a \cdot x\| = |a| \cdot \|x\|$.

4. The triangle inequality holds for every $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$, $\|x + y\| \leq \|x\| + \|y\|$.

A norm induces a metric by the formula $d(x,y) = \|x - y\|$. For every $i \in \{1, ..., n\}$ let e_i be a unit vector of $\mathbb{R}^n$ i.e. $e_i(i) = 1$ and $e_i(j) = 0$ for all $j \in \{1, ..., n\} \setminus \{i\}$. Define $\rho = \max_{i \in \{1,...,n\}} \|e_i\|$. ∎

Definition 3.3.2 (Lebesgue outer measure, [48, Section 2.A])**.** Let $\mathbb{R}^n$ be an n-dimensional space. Define n-dimensional interval as

$$I = \{x \in \mathbb{R}^n \mid a_i \leq x_i \leq b_i, i = 1, ..., n\} = [a_1, b_1] \times ... \times [a_n, b_n],$$

with sides parallel to the coordinate axes. Define Lebesgue outer measure $\mu^* : \{A \mid A \subseteq \mathbb{R}^n\} \to [0, \infty) \cup \{\infty\}$ of interval I as $\mu^*(I) = (b_1 - a_1) \cdot ... \cdot (b_n - a_n)$. The Lebesgue μ measure of a set $A \subseteq \mathbb{R}^n$ is defined as:

$$\mu^*(A) = \inf_A \{\sum_{i=0}^{\infty} \mu^*(I_i) \mid A \subseteq \cup_{i=0}^{\infty} I_i\},$$

where the infinium is taken over all covering of A by countably many intervals $I_i, i = 1, 2....$ If set $E \subseteq \mathbb{R}^n$ satisfies $\mu^*(A) = \mu^*(A \cap E) + \mu^*(A \setminus E)$ for all $A \subseteq \mathbb{R}^n$, then E is lebesgue-measurable and we set $\mu(E) = \mu^*(E)$. ∎

It should be noted that all open sets and closed sets , as well as compact sets are Lebesgue-measurable.

Lemma 3.3.3 (Basic properties of Lebesgue measure, [48, Section 2.A]). *A Lebesgue outer measure μ^* of Definition 3.3.2 satisfies the following conditions:*

1. $\mu^*(\emptyset) = 0$,

2. $\mu^*(A) \leq \mu^*(B)$ *whenever $A \subseteq B \subseteq \mathbb{R}^n$ and*

3. $\mu^*(\cup_{i=1}^{\infty} \mu^*(A_i)) \leq \sum_{i=1}^{\infty} \mu^*(A_i)$.

Lemma 3.3.4 (Lebesgue measure scale property, [48, Section 3.B]). *Let μ be Lebesgue measure on normed vector space $(\mathbb{R}^n, d)$, then for any $p \in \mathbb{R}^n$ and $t \in \mathbb{R}_+$ we have: $\mu(\bar{B}(p, t)) = t^n \cdot \mu(\bar{B}(p, 1))$.*

Definition 3.3.5 (Grid and Cubic region). Let $\mathbb{R}^n$ be a normed vector space and let $\delta \in \mathbb{R}$. Define δ-grid on $\mathbb{R}^n$ as the set $G(\delta) = \{t \cdot \delta \mid t \in \mathbb{Z}^n\}$. For any $p \in \mathbb{R}^n$ define its open cubic region $V(p, \delta) \subseteq \mathbb{R}^n$ as the set $\{p + u \mid u \in (-\frac{\delta}{2}, \frac{\delta}{2})^n\}$ and closed cubic region $\bar{V}(p, \delta) \subseteq \mathbb{R}^n$ as $\{p + u \mid u \in [-\frac{\delta}{2}, \frac{\delta}{2}]^n\}$.

Note that the union $\cup_{p \in G(\delta)} V(p, \delta)$ covers whole set $\mathbb{R}^n$.

Lemma 3.3.6 (Cubic regions are separate). *In conditions of Definition 3.3.5 let $p, q \in G(\delta)$ be distinct points. Then their cubic regions are separate i.e. $V(p, \delta) \cap V(q, \delta) = \emptyset$.*

Proof. Assume contrary that there exists $a \in V(p, \delta) \cap V(q, \delta)$, then $|a(i) - p(i)| < \frac{\delta}{2}$ and $|a(i) - q(i)| < \frac{\delta}{2}$ for all $i \in \{1, ..., n\}$. Since $p \neq q$, there exists index j, such that $p(j) \neq q(j)$. By definition of grid $G(\delta)$ it follows that $|p(j) - q(j)| \geq \delta$. However, by triangle inequality we have

$$|p(j) - q(j)| \leq |p(j) - a(j)| + |q(j) - a(j)| < \frac{\delta}{2} + \frac{\delta}{2} = \delta,$$

which is a contradiction. Therefore $V(p, \delta) \cap V(q, \delta) = \emptyset$.

□

Lemma 3.3.7. *Let $\mathbb{R}^n$ be a normed vector space of Definition 3.3.1. Let $\delta \in \mathbb{R}$ and let $G(\delta)$ be a grid of Definition 3.3.5. Let $p \in G(\delta)$ and let $q \in V(p, \delta)$, then $d(p, q) \leq \frac{n \cdot \delta \cdot \rho}{2}$*

Proof. Let $\gamma \in \mathbb{R}$ be such that $q = p + \gamma$. By condition (3) of Definition 3.3.1 we have $\|\gamma(i)\| \leq \|e_i\| \cdot \frac{\delta}{2} \leq \frac{\delta \cdot \rho}{2}$ for all $i \in \{1, ..., n\}$. By the definition of norm and triangle inequality we have:

$$d(p, q) = \|p - q\| = \|\gamma\| \leq \sum_{i=1}^{n} \|\gamma(i)\| \leq \frac{n \cdot \delta \cdot \rho}{2}.$$

$\square$

If $(\mathbb{R}^n, d)$ is a normed vector space , then its metric $d(\cdot, \cdot)$ satisfies $d(x, y) = \|x - y\|$.

Lemma 3.3.8 (Existence of covering grid). *Let R be a finite subset of normed vector space $(\mathbb{R}^n, d)$. Then for any $\delta \in \mathbb{R}$ having $\delta < \frac{d_{\min}(R)}{n \cdot \rho}$, then any map $f : R \to G(\delta)$ which maps $p \in R$ to one of its nearest neighbor in $G(\delta)$ is a well-defined injection.*

Proof. Let f be an arbitrary map $f : R \to G(\delta)$ mapping point $p \in R$ to one of its nearest neighbors. This map is clearly well-defined. Let us now show that it is injective. Assume that $x, y \in R$ are such that $f(x) = f(y)$. Then by triangle inequality and Lemma 3.3.7 we have:

$$d(x, y) \leq d(x, p) + d(p, y) \leq n \cdot \delta \cdot \rho < d_{\min}(R),$$

it follows that $x = y$. Therefore map f is injective.

$\square$

Lemma 3.3.9. *Let R be a finite subset of normed space $(\mathbb{R}^n, d)$, let ρ be as in Definition 3.3.1 and let $\delta \in \mathbb{R}$ be such that $0 < \delta < \frac{d_{\min}(R)}{n \cdot \rho}$. Let $p \in R$ be arbitrary point and let $t > \frac{n \cdot \delta \cdot \rho}{2}$ be a real number. Then there exists a set $U(\delta)$ satisfying $R \subseteq U(\delta)$ and*

$$|G(\delta) \cap \bar{B}(p, t - \frac{n \cdot \delta \cdot \rho}{2})| \leq |U(\delta) \cap \bar{B}(p, t)| \leq |G(\delta) \cap \bar{B}(p, t + \frac{n \cdot \delta \cdot \rho}{2})|$$

Proof. Let $f : R \to G(\delta)$ be an injection from Lemma 3.3.8, which maps every $q \in R$ to one of its nearest neighbors in $G(\delta)$. Define $U(\delta) = (G(\delta) \setminus f(R)) \cup R$. Let us first show that

$$g : U(\delta) \cap \bar{B}(p, t) \to G(\delta) \cap \bar{B}(p, t + \frac{n \cdot \delta \cdot \rho}{2}),$$

defined by $g(q) = f(q)$, if $q \in R$ and $g(q) = q$, if $q \notin R$, is an injection. Let us show first that the map g is well-defined, if $q \notin R$, the claim is trivial. Let $q \notin R$, then by triangle inequality $d(g(q), p) \leq d(q, p) + d(g(q), q) \leq t + \frac{n \cdot \delta \cdot \rho}{2}$. Assume now that $g(a) = g(b)$ for some $a, b \in U(\delta) \cap \bar{B}(p, t)$. Let us first show that either a, b both belong to R or neither of a, b belong to R. Assume contrary

that $a \in R$ and $b \notin R$. Since $b \notin R$ we have $b \in G(\delta) \setminus f(R)$. On the other hand since $h(a) = h(b)$ we have $f(a) = b$, therefore $b \in f(R)$, which is a contradiction. If both, a and b belong to R we have $a = b$, similarly if $a, b \notin R$ we have $a = b$ by injectivity of function f. Therefore we have now shown that g is well-defined injection. It follows $|U(\delta) \cap \bar{B}(p, t)| \leq |G(\delta) \cap \bar{B}(p, t + \frac{n \cdot \delta \cdot \rho}{2})|$. Let us now show that map

$$h : G(\delta) \cap \bar{B}(p, t - \frac{n \cdot \delta \cdot \rho}{2}) \rightarrow U(\delta) \cap \bar{B}(p, t),$$

defined by $h(q) = f^{-1}(q)$, if $q \in f(R)$ and $h(q) = q$, if $q \notin f(R)$ is well-defined injection. Let us first show that the map is well-defined. Let $q \in G(\delta) \cap \bar{B}(p, t - \frac{n \cdot \delta \cdot \rho}{2})$, if $q \notin f(R)$ the claim is satisfied trivially. If $q \in f(R)$, then by definition $d(h(q), q) \leq \frac{n \cdot \delta \cdot \rho}{2}$. By using triangle inequality we obtain:

$$d(p, h(q)) \leq d(p, q) + d(q, h(q)) \leq t - \frac{n \cdot \delta \cdot \rho}{2} + \frac{n \cdot \delta \cdot \rho}{2} \leq t.$$

Therefore $h(q) \in U(\delta) \cap \bar{B}(p, t)$.

Let us now show that h is an injection. Let $a, b \in G(\delta) \cap \bar{B}(p, t - \frac{n \cdot \delta}{2})$ assume that $h(a) = h(b)$, let us show that $a = b$. Let us first show that either $a, b \in f(R)$ or neither of a, b belong to $f(R)$. Assume contrary that $a \in f(R)$ and $b \notin f(R)$. Then $h(a) = h(b)$ implies that $f^{-1}(a) = b$. Since $f^{-1}(a) \in R$, we have $b \in R$. Since $b \in G(\delta)$, it follows that $f(b) = b$, which is a contradiction since $b \notin f(R)$. Assume now that $a, b \in f(R)$, then the claim follows by noting that f^{-1} is injection. If $a, b \notin f(R)$, then claim follows by noting that $h(a) = a$ and $h(b) = b$. Therefore map h is injection. It follows that $|G(\delta) \cap \bar{B}(p, t - \frac{n \cdot \delta \cdot \rho}{2})| \leq |U(\delta) \cap \bar{B}(p, t)|$.

$\square$

Lemma 3.3.10. *Let R be a finite subset of normed vector space $\mathbb{R}^n$ and let $\delta \in \mathbb{R}$. For any $p \in G(\delta)$ recall that $V(p, \delta)$ is Minkowski sum $p + (-\frac{\delta}{2}, \frac{\delta}{2})^n$. Define*

$$\bar{W}(p, t, \delta) = \bigcup_{q \in \bar{B}(p,t) \cap G(\delta)} \bar{V}(q, \delta).$$

Then for any $p \in R$ and $t > \frac{n \cdot \delta \cdot \rho}{2}$ we have:

$$\bar{B}(p, t - \frac{n \cdot \delta \cdot \rho}{2}) \subseteq \bar{W}(p, t, \delta) \subseteq \bar{B}(p, t + \frac{n \cdot \delta \cdot \rho}{2}).$$

Proof. Let $u \in \bar{B}(p, t - \frac{n \cdot \delta \cdot \rho}{2})$ be an arbitrary point. Since $\{\bar{V}(q, \delta) \mid q \in G(\delta)\}$ covers R it follows

that there exists $a \in G(\delta)$ such that $u \in \bar{V}(a,\delta)$. By triangle inequality we obtain:

$$d(a,p) \le d(a,u) + d(u,p) \le \frac{n \cdot \delta \cdot \rho}{n} + t - \frac{n \cdot \delta \cdot \rho}{n} \le t.$$

It follows that $\bar{V}(w,\delta) \in \bar{W}(p,t)$, therefore $p \in \bar{W}(p,t,\delta)$. We have $\bar{B}(p,t - \frac{n \cdot \delta \cdot \rho}{2}) \subseteq \bar{W}(p,t,\delta)$. Let $u \in \bar{W}(p,t,\delta)$, then there exists $a \in G(\delta)$ such that $u \in \bar{V}(a,\delta)$ and $\bar{V}(a,\delta) \in \bar{W}(p,t)$. By triangle inequality we obtain:

$$d(u,p) \le d(u,a) + d(a,p) \le \frac{n \cdot \delta \cdot \rho}{n} + t.$$

It follows that $u \in \bar{B}(p,t + \frac{n \cdot \delta \cdot \rho}{2})$. Therefore $\bar{W}(p,t,\delta) \subseteq \bar{B}(p,t + \frac{n \cdot \delta \cdot \rho}{2})$ which proves the claim.

$\square$

Lemma 3.3.11 (Countable additivity, [48, Section 2.A]). *Assume that $A_i \subseteq \mathbb{R}^n$, $i = 1,2,...$, are pairwise disjoint i.e. $A_i \cap A_j = \emptyset$ for all $i \ne j$ Lebesgue-measurable sets. Then*

$$\mu(\bigcup_{i=0}^{\infty} A_i) = \sum_{i=1}^{\infty} \mu(A_i).$$

Lemma 3.3.12 (Lebesgue measure of $\bar{W}(p,t,\delta)$). *In notations of Lemma 3.3.10 let μ be a Lebesgue measure on R from Definition 3.3.2, then $\mu(\bar{W}(p,t,\delta)) = \delta^n \cdot |\bar{B}(p,t) \cap G(\delta)|$.*

Proof. Define $W(p,t,\delta) = \bigcup_{q \in \bar{B}(p,t) \cap G(\delta)} V(q,\delta)$. Recall that for all $p \in \mathbb{R}^n$ and $\delta > 0$ set $\bar{V}(p,t)$ is a closed n–dimensional interval and $V(p,t)$ is an open n-dimensional interval. Therefore we have $\mu(\bar{V}(p,t)) = \mu(V(p,t))$. Since $\bar{V}(p,t)$ is a closed interval, it follows that $\mu(\bar{V}(p,t)) = \delta^n$. Since all the sets of W are separate we can use Lemma 3.3.11 to obtain:

$$\mu(W(p,t,\delta)) = \sum_{A \in W(p,t)} \mu(A) = \sum_{A \in \bar{W}(p,t)} \mu(A) = \delta^n \cdot |\bar{B}(p,t) \cap G(\delta)|$$

By Lemma 3.3.3 (2), since $W(p,t,\delta) \subseteq \bar{W}(p,t,\delta)$ we obtain $\mu(\cup \bar{W}(p,t)) \ge \delta^n \cdot |\bar{B}(p,t) \cap G(\delta)|$. On the other hand, by Lemma 3.3.3 (3) we obtain

$$\mu(\bar{W}(p,t,\delta)) \le \sum_{A \in \bar{W}(p,t)} \mu(A) = \delta^n \cdot |\bar{B}(p,t) \cap G(\delta)|$$

Therefore we have shown that $\mu(\bar{W}(p,t,\delta)) = \delta^n \cdot |\bar{B}(p,t) \cap G(\delta)|$.

$\square$

Lemma 3.3.13 (Set $U(\delta)$ bounds)**.** *Let $\mathbb{R}^n$ be a normed vector space. Let $R \subseteq \mathbb{R}^n$ be its finite subset. Then any set $U(\delta)$ of Lemma 3.3.9 satisfies the following inequalities:*

$$\frac{\mu(\bar{B}(p, t - \delta \cdot n \cdot \rho))}{\delta^n} \le |\bar{B}(p, t) \cap U(\delta)| \le \frac{\mu(\bar{B}(p, t + \delta \cdot \gamma \cdot n \cdot \rho))}{\delta^n},$$

for all $p \in R$ and $t > n \cdot \delta \cdot \rho$.

Proof. Let $p \in \mathbb{R}^n$ be an arbitrary point and let $t > n \cdot \delta \cdot \rho$ be an arbitrary real number. By Lemma 3.3.9 it follows:

$$|G(\delta) \cap \bar{B}(p, t + \frac{n \cdot \delta \cdot \rho}{2})| \le |\bar{B}(p, t) \cap U(\delta)| \le |G(\delta) \cap \bar{B}(p, t + \frac{n \cdot \delta \cdot \rho}{2})|.$$

Let $\bar{W}(p, t + \frac{n \cdot \delta \cdot \rho}{2}, \delta) = \cup_q \{\bar{V}(q, \delta) \mid q \in \bar{B}(p, t + \frac{n \cdot \delta \cdot \rho}{2})\}$. By Lemma 3.3.10 we have:

$$\bar{B}(p, t - n \cdot \delta \cdot \rho) \subseteq \bar{W}(p, t - \frac{n \cdot \delta \cdot \rho}{2}, \delta) \text{ and } \bar{W}(p, t + \frac{n \cdot \delta \cdot \rho}{2}, \delta) \subseteq \bar{B}(p, t + n \cdot \delta \cdot \rho)$$

By Lemma 3.3.3 we have $\mu(\bar{W}(p, t + \frac{n \cdot \delta \cdot \rho}{2}, \delta)) \le \mu(\bar{B}(p, t + n \cdot \delta \cdot \rho))$. By Lemma 3.3.12 we have:

$$\mu(\bar{W}(p, t + \frac{n \cdot \delta \cdot \rho}{2}, \delta)) = \delta^n \cdot |\bar{B}(p, t + \frac{n \cdot \delta \cdot \rho}{2}) \cap G(\delta)|.$$

By combining the facts we obtain:

$$|\bar{B}(p, t) \cap U(\delta)| \le |G(\delta) \cap \bar{B}(p, t + \frac{n \cdot \delta \cdot \rho}{2})| \le \frac{\mu(\bar{W}(p, t + \frac{n \cdot \delta \cdot \rho}{2}, \delta))}{\delta^n} \le \frac{\mu(\bar{B}(p, t + n \cdot \delta \cdot \rho))}{\delta^n}$$

$$|\bar{B}(p, t) \cap U(\delta)| \ge |G(\delta) \cap \bar{B}(p, t - \frac{n \cdot \delta \cdot \rho}{2})| \ge \frac{\mu(\bar{W}(p, t - \frac{n \cdot \delta \cdot \rho}{2}, \delta))}{\delta^n} \ge \frac{\mu(\bar{B}(p, t - n \cdot \delta \cdot \rho))}{\delta^n}$$

$\square$

Lemma 3.3.14 (Set $U(\delta)$ is locally finite)**.** *Let $\mathbb{R}^n$ be a normed vector space. Let $R \subseteq \mathbb{R}^n$ be its finite subset Then any set $U(\delta)$ of Lemma 3.3.9 is locally finite.*

Proof. With the exact same proof of Lemma 3.3.13 it can be shown that

$$|\bar{B}(p, t) \cap U(\delta)| \le \frac{\mu(\bar{B}(p, t + \delta \cdot n \cdot \rho))}{\delta^n}$$

is satisfied for all $p \in R$ and $t > 0$. Therefore $|\bar{B}(p, t) \cap U(\delta)|$ is finite as well.

$\square$

Recall that *minimized expansion constant* of Definition 3.1.4 of a finite subset R of a metric space (X, d) was defined as $c_m(R) = \inf\limits_{0<\xi} \inf\limits_{R\subseteq A\subseteq X} \sup\limits_{p\in A, t>\xi} \dfrac{|\bar{B}(p, 2t) \cap A|}{|\bar{B}(p, t) \cap A|}$ where A is a locally finite set which covers R.

Theorem 3.3.15 (The minimized expansion constant of a finite subset R of $\mathbb{R}^n$ is at most 2^n)**.** *Let R be a finite subset of a normed Euclidean space $\mathbb{R}^n$. Let $c_m(R)$ be the minimized expansion constant of Definition 3.1.4, then $c_m(R) \le 2^n$.*

Proof. Let $0 < \xi < \frac{d_{\min}(R)}{2}$ be an arbitrary real number. Let $0 < \delta < \frac{\xi}{n\cdot\rho}$ be a real number. Since $\delta < \frac{d_{\min}(R)}{2\cdot n\cdot\rho}$ by Lemma 3.3.13 we have:

$$\frac{\mu(\bar{B}(p, t - \delta\cdot n\cdot\rho))}{\delta^n} \le |\bar{B}(p, t) \cap U(\delta)| \le \frac{\mu(\bar{B}(p, t + \delta\cdot\gamma\cdot n\cdot\rho))}{\delta^n}$$

Note that by Lemma 3.3.4 we have: $\mu(\bar{B}(q, y)) = y^n\cdot\mu(\bar{B}(q, 1))$ for any $q \in \mathbb{R}^n$ and $y \in \mathbb{R}_+$. Therefore

$$\frac{|\bar{B}(p, 2t) \cap U(\delta)|}{|\bar{B}(p, t) \cap U(\delta)|} \le \frac{\mu(\bar{B}(p, 2t + n\delta\rho))\cdot\delta^2}{\mu(\bar{B}(p, t - n\delta\rho))\cdot\delta^2} = \frac{(2t + n\delta\rho)^n\cdot\mu(\bar{B}(p, 1))}{(t - n\delta\rho)^n\cdot\mu(\bar{B}(p, 1))} = \frac{(2t + n\delta\rho)^n}{(t - n\delta\rho)^n},$$

is satisfied for for all $t > \xi$. Since $0 < \xi < \frac{d_{\min}(R)}{2}$ was chosen arbitrarily, we conclude that:

$$c_m(R) = \inf\limits_{0<\xi} \inf\limits_{R\subseteq A\subseteq X} \sup\limits_{p\in A, t>\xi} \frac{|\bar{B}(p, 2t)| \cap A}{|\bar{B}(p, t)| \cap A} \le \lim\limits_{\delta\to 0} \frac{|\bar{B}(p, 2t) \cap U(\delta)|}{|\bar{B}(p, t) \cap U(\delta)|} = \lim\limits_{\delta\to 0} \frac{(2t + \delta\cdot n\rho)^n}{(t - \delta\cdot n\rho)^n} = \frac{2^n\cdot t^n}{t^n} = 2^n.$$

$\square$

3.4   Counterexamples for ICML 2006 complexity analysis

This section show that both proofs of the main theorems [9, Theorem 5] and [9, Theorem 6] are incorrect. Since the same false arguments were later repeated in the papers [63] and [78] we provide a detailed counterexamples that expose the contradiction within each of the proof of the theorems.

Counterexample 3.4.5 shows that the proof of worst-case time complexity theorem of the Insert algorithm for an implicit cover tree [9, Theorem 6] is incorrect.
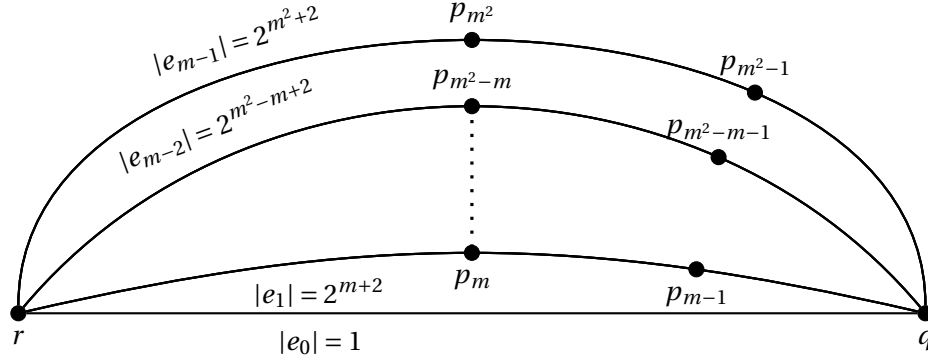
Figure 3.7: Illustration of a graph G and a point cloud R defined in Example 3.4.1

Counterexample 3.4.7 shows that the proof of [9, Theorem 5], which gives an upper bound for the complexity of Algorithm 3.4.6 is incorrect as well. Both counterexamples are based on Example 3.4.1, which extends Example 3.2.4.

Although given proofs are incorrect, it is not known if claimed worst-case time complexity estimates are true for [9, Theorem 5] and [9, Theorem 6] which claim a parametrized worst-case near-linear time complexities for algorithms [9, Algorithms 1] and [9, Algorithms 2], respectively. The issues are fixed by using similar techniques, but by taking more rigorous approach to the problem. New approach with new worst-case time complexity theorems for cover tree construction is explained in Section 3.5 and for k-nearest neighbor search in Section 3.7.

Example 3.4.1 (tall imbalanced tree)**.** Let (R, d) be a finite metric space. Consider a tree in Figure 3.8 containing $\sqrt{|R|}$ branches, each having a size of $\sqrt{|R|}$. Simulation of Algorithms 3.4.4 and Algorithm 3.4.6 will show that both algorithms iterate over all nodes of R. Original [9, Theorem 5] and [9, Theorem 6] claimed that the number of iterations is proportional to the length of a longest root-to-leaf path that has a size of $\sqrt{|R|}$, which will contradict our findings.

For any integer $m > 10$, let G be a metric graph pictured in Figure 3.7 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 $i \geq 1$. 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 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 a compressed cover tree $\mathcal{T}(R)$ by setting r to be the root node and $l(p_i) = i$ for all i. If i is divisible by m, we set r to be the parent of p_i. If i is not divisible by m, we set p_{i+1} to be the parent of p_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 $\mathcal{T}(R)$ satisfies covering condition (3.2.1b). 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 (3.2.1c). 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 (3.2.1c). Hence this condition holds for the whole tree $\mathcal{T}(R)$. ∎

Let us now define the concept of explicit depth, that corresponds to maximal root-to-node path of any cover tree. Recall that in [9, Section 2] the explicit representation of cover tree was defined as "the explicit representation of the tree coalesces all nodes in which the only child is a self-child". Simplest way to interpret this is to consider cover sets C_i and define $p \in C_i$ to be an explicit node, if p has child at level $i - 1$. By [9, Lemma 4.3] depth of any node p 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". Explicit depth of a node p in any compressed tree $\mathcal{T}$ will be defined in Definition 3.4.2 using the simplest interpretation of the aforementioned quotes.

Lemma 3.4.2 (Explicit depth for compressed cover tree)**.** *Let R be a finite subset of a metric space with a metric d. Let $\mathcal{T}(R)$ be a compressed cover tree on R. For any $p \in \mathcal{T}(R)$, let $s = (w_0, ..., w_m)$ be a node-to-root path of p. Then the explicit depth $D(p)$ of node p belonging to compressed cover*

tree can be interpreted as the sum

$$D(p) = \sum_{i=0}^{m-1} |\{q \in \text{Children}(w_{i+1}) \mid l(q) \in [l(w_i), l(w_{i+1}) - 1]\}|.$$

∎

Proof. Note that the node-to-root path of an implicit cover tree on R has $l(w_{j+1}) - l(w_j) - 1$ extra copies of w_{j+1} between every w_j and w_{j+1} for any index $j \in [0, m-1]$. Recall that a node is called explicit, if it has non-trivial children. Therefore there will be exactly

$$|\{q \in \text{Children}(w_{i+1}) \mid l(q) \in [l(w_i), l(w_{i+1}) - 1]\}|$$

explicit nodes between w_j and w_{j+1}. It remains to take the total sum. □

Lemma 3.4.3 shows that the cover tree of Example 3.4.1 the maximal node-to-root has $2 \cdot \sqrt{|R|}$ size, where $|R|$ is the size of dataset.

Lemma 3.4.3. *Let $\mathcal{T}(R)$ be a compressed cover tree on the set R from Example 3.4.1 for some $m \in \mathbb{Z}$. For any $p \in R$, the explicit depth $D(p)$ of Definition 3.4.2 has the upper bound $2m + 1$.* ∎

Proof. For any p_i, if i is divisible by m, then r is the parent of p_i. By definition, the explicit depth is $D(p_i) = |\{p \in \text{Children}(r) \mid l(p) \in [l(p_i), m^2]\}|$. Since r contains children on every level j, where j is divisible by m, we have $D(p_i) = m - \frac{i}{m} + 1$.

Let us now consider an index i that is not divisible by m. Note that p_{j+1} is the parent of p_j for all $j \in [i, m \cdot \lceil i/m \rceil - 1]$. Then the path consisting of all ancestors of p_i from p_i to the root node r has the form $(p_i, p_{i+1}, ..., p_{m \cdot \lceil i/m \rceil}, r)$. It follows that

$$D(p_i) = \sum_{j=i}^{m \cdot \lceil i/m \rceil - 1} |\{p \in \text{Children}(p_j) \mid l(p) \in [l(p_j), l(p_{j+1}) - 1]\}| + D(p_{m \cdot \lceil i/m \rceil}).$$

Since $i \geq m \cdot (\lceil i/m \rceil - 1) + 1$ and $\lceil i/m \rceil \geq 1$, we get the required upper bound:

$$D(p_i) = (m \cdot \lceil i/m \rceil - i) + (m - \lceil i/m \rceil + 1) \leq m + (m + 1) = 2m + 1.$$

□
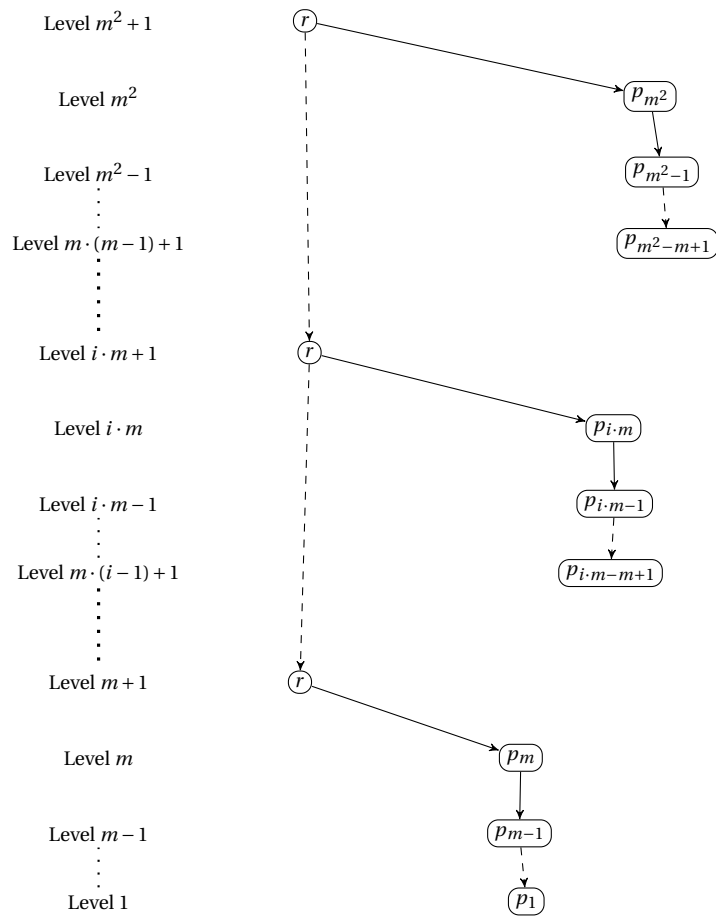
Figure 3.8: Illustration of explicit cover tree built on the dataset of Example 3.4.1

Algorithm 3.4.4 Copy-pasted Insert() algorithm for inserting 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.

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**

Counterexample 3.4.5 (Counterexample for a step in the proof of [9, Theorem 6])**.** Counterexample 3.4.5 shows that there is a mistake in proof of [9, Theorem 6]. The idea is based on adding a new point q of Figure 3.7 to the tree $\mathcal{T}(R)$ of Example 3.4.1 that lures the Algorithm 3.4.4 into using all branches of $\mathcal{T}(R)$. It follows that the Algorithm 3.4.4 is launched $O(|R|)$ times. However, in the proof of [9, Theorem 6] it was claimed that Algorithm 3.4.4 is launched at most $4 \cdot D(\mathcal{T}(R))$, where $D(\mathcal{T}(R))$ is the explicit depth of explicit cover tree $\mathcal{T}(R)$. This is a contradiction, since $D(\mathcal{T}(R)) \le 2\sqrt{(|R|)} + 1$.

For more detailed exhibition let us first 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 3.4.4 is $O(c^6 \log(n))$, where n is the number points of original dataset S on which tree T was constructed.]

[*Part of the 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 $4 \cdot \max_{p \in R} D(p)$, where $D(p)$ is the explicit depth of a point p, see Definition 3.4.2. In this Counterexample we will show that $\mathcal{T}(R)$ from Example 3.4.1 does not satisfy the claimed inequality.

Take the reference set R, the compressed cover tree $\mathcal{T}(R)$ and the point q from Example 3.4.1 for any parameter $m > 200$. Assume that we have already constructed tree $\mathcal{T}(R)$. Let us show that $\mathcal{T}(R \cup q)$ constructed by Algorithm 3.4.4 from the input $q, i = m^2 + 1, Q_i = \{r\}$ runs at least $m^2 - 2$ self-recursions. This will lead to a contradiction since by Lemma 3.4.3 any node $p \in \mathcal{T}(R)$ has $D(p) \le 2m + 1$.

We show by induction on m going down that, for every step $i \in [1, m^2]$, we have $Q_i = \{r, p_i\}$. 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} is a child of the root r. For all levels $i - 1$ not divisible by m, the node p_i is a child of p_i. Since $\mathcal{T}(R)$ contains exactly one node at each level, in both cases we have $Q = \{r, p_i, p_{i-1}\}$. Since $d(q, r) = 1$, $d(q, p_i) = 2^{i+1}$ and $d(q, p_{i-1}) = 2^i$ we have

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

The actual implementation of Algorithm 3.4.4 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 $Q_i = \{r, p_i\}$ 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:

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

where $m > 20$.   ∎

Counterexample 3.4.7 (for a step in the proof of [9, Theorem 5]). Counterexample 3.4.7 shows that there is a mistake in proof of [9, Theorem 6]. The counterexample is obtained by running Algorithm 3.4.6 for node q of Figure 3.7 and tree $\mathcal{T}(R)$ of Example 3.4.1. It it shown that Algorithm 3.4.6 iterates over all branches of $\mathcal{T}(R)$, 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 $\max_i |R_i| \cdot D(\mathcal{T}(R))$, where $D(\mathcal{T}(R))$ is the maximal path-to-root path that has an upper bound $2\sqrt{|R|}$. In this counterexample it will be also shown that $\max_i |R_i| \le 3$

Algorithm 3.4.6 Copy-pasted [9, Algorithm 1] based on an implicit cover tree T [9, Section 2] for nearest neighborhood search, which is used in Counterexample 3.4.7. 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). The level $+\infty$ can be replaced by $l_{\max}(T)$ and $-\infty$ can be replaced by $l_{\min}(T)$ in the code.

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** $\text{argmin}_{q \in Q_{-\infty}} d(p,q)$

---

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 3.4.6 (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 3.4.6 runs lines 3-6 has an upper bound $\xi \le \max_{p \in R} D(p) \cdot \max_i |Q_i|$. Contradiction will be obtained by showing that $\mathcal{T}(R)$ from Example 3.4.1 does not satisfy this inequality.

Take $R, \mathcal{T}(R)$ and q from Example 3.4.1. We will apply Algorithm 3.4.6 to the tree $\mathcal{T}(R)$ and query point q. By Lemma 3.4.3 the cover tree $\mathcal{T}(R)$ 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 3.4.5 we will show that, for every iteration (lines 3-5) $i \in [1, m^2]$ of Algorithm 3.4.6, we have $Q_i = \{r, p_i\}$. 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} is a child of the root r. For all levels $i-1$ not divisible by m, the node p_{i-1} is a child of p_i. Since $\mathcal{T}(R)$ contains exactly one node at each level, in both cases we have $Q = \{r, p_i.p_{i-1}\}$. Since $d(q,r) = 1$, $d(q,p_i) = 2^{i+1}$ and $d(q,p_{i-1}) = 2^i$, we have

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

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

The actual implementation of algorithm 3.4.6iterates 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 or p_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

$$m^2 - 2 \le \xi \le \max_{p \in R} D(p) \cdot \max_i |Q_i| \le (2m+1) \cdot 2 \le 4m + 2 \text{ for any } m > 20.$$

∎

3.5 Construction of a compressed cover tree

This section introduces a new method Algorithm 3.5.3 for construction of a compressed cover tree, which is based on Insert() method [9, Algorithm 2] that was specifically adapted for compressed cover tree. The proof of [9, Theorem 6], which estimated the time complexity of [9, Algorithm 2] was shown to be incorrect Counterexample 3.4.5. The main contribution of this section are two new time complexity results that bound the time complexity of Algorithm 3.5.3:

- Theorem 3.5.7 bounds the time complexity as $O(c_m(R)^{10} \cdot \log_2(\Delta(R)) \cdot |R|)$ by using minimized expansion constant $c_m(R)$ and aspect ratio $\Delta(R)$ as parameters.

- Theorem 3.5.10 bounds the time complexity as $O(c(R)^{12} \cdot \log_2 |R| \cdot |R|)$ by using expansion constant $c(R)$ as parameter.

Definition 3.5.1 explains the concrete implementation of compressed cover tree.

Definition 3.5.1 (Children(p,i) and Next$(p,i,\mathcal{T}(R))$)**.** In a compressed cover tree $\mathcal{T}(R)$ on a set R, for any level i and a node $p \in R$, set Children$(p,i) = \{a \in \text{Children}(p) \mid l(a) = i\}$. Let Next$(p,i,\mathcal{T}(R))$ be the maximal level j satisfying $j < i$ and Children$(p,i) \ne \emptyset$. If such level does not exist, we set $j = l_{\min}(\mathcal{T}(R)) - 1$. For every node p, we store its set of children in a linked hash map so that

(a) any key i gives access to Children(p, i),

(b) every Children(p, i) has access to Children$(p, \text{Next}(p, i, \mathcal{T}(R)))$,

(c) we can directly access $\max\{ j \mid \text{Children}(p, j) \neq \emptyset \}$. ∎

Definition 3.5.2 (construction iteration set $L(\mathcal{T}(W), p)$)**.** Let W be a finite subset of a metric space (X, d). Let $\mathcal{T}(W)$ be a cover tree of Definition 3.2.1 built on W and let $p \in X \setminus W$ be an arbitrary point. Let $L(\mathcal{T}(W), p) \subseteq H(\mathcal{T}(R))$ be the set of all levels i during iterations 5-14 of Algorithm 3.5.4 launched with inputs $\mathcal{T}(W), p$. Set $\eta(i) = \min_t \{ t \in L(\mathcal{T}(W), p) \mid t > i \}$. ∎

Algorithm 3.5.3 Building a compressed cover tree $\mathcal{T}(R)$ from Definition 3.2.1.

1: **Input** : a finite subset R of a metric space (X, d)
2: **Output** : a compressed cover tree $\mathcal{T}(R)$.
3: Choose a random point $r \in R$ to be a root of $\mathcal{T}(R)$
4: Build the initial compressed cover tree $\mathcal{T} = \mathcal{T}(\{r\})$ by making $l(r) = +\infty$.
5: **for** $p \in R \setminus \{r\}$ **do**
6:    $\mathcal{T} \leftarrow$ run AddPoint$(\mathcal{T}, p)$ described in Algorithm 3.5.4.
7: **end for**
8: For root r of $\mathcal{T}$ set $l(r) = 1 + \max_{p \in R \setminus \{r\}} l(p)$

Let R be a finite subset of a metric space (X, d). A compressed cover tree $\mathcal{T}(R)$ will be incrementally constructed by adding points one by one as summarized in Algorithm 3.5.3. First we select a root node $r \in R$ and form a tree $\mathcal{T}(\{r\})$ of a single node r at the level $l_{\max} = l_{\min} = +\infty$. Assume that we have a compressed cover tree $\mathcal{T}(W)$ for a subset $W \subset R$. For any point $p \in R \setminus W$, Algorithm 3.5.4 builds a larger compressed cover tree $\mathcal{T}(W \cup \{p\})$ from $\mathcal{T}(W)$.

Note that during the construction of the compressed cover tree in Algorithm 3.5.4 we write down additional information for every node p , which includes the number of descendants of node p and the maximal level of nodes in set Children(p).

Theorem 3.5.5 (correctness of Algorithm 3.5.3)**.** *Algorithm 3.5.3 builds a compressed cover tree satisfying Definition 3.2.1.* ∎

Proof. It suffices to prove that Algorithm 3.5.4 correctly extends a compressed cover tree $\mathcal{T}(W)$ for any finite subset $W \subseteq X$ by adding a point p. Let us prove that $\mathcal{T}(W \cup \{p\})$ satisfies Definition 3.2.1.

If parent of p is chosen on lines 8 - 11 , then $d(p, q) \leq 2^{\eta(i)+1}$, for any $q \in R_{\eta(i)}$. Therefore we have found a parent q for node p satisfying the covering condition. If p is chosen on lines 15 - 17 then by definition m is minimal satisfying $d(p, R_{l_{\min}}) > 2^m$. It follows that $d(p, R_{l_{\min}}) \leq 2^{m+1}$. Therefore

---

Algorithm 3.5.4 Building $\mathcal{T}(W \cup \{p\})$ in lines 5-7 of Algorithm 3.5.3.

1: **Function** AddPoint(a compressed cover tree $\mathcal{T}(W)$ with a root r for a finite subset $W \subseteq X$, a point $p \in X$)
2: **Output** : compressed cover tree $\mathcal{T}(W \cup \{p\})$.
3: Set $i \leftarrow l_{\max}(\mathcal{T}(R)) - 1$ and $\eta(l_{\max} - 1) = l_{\max}$ {If the root r has no children then $i \leftarrow -\infty$}
4: Set $R_{l_{\max}} \leftarrow \{r\}$, Set $m \leftarrow \max_{j \leq l_{\max}} \{j \mid d(p, R_{\eta(i)}) > 2^j\}$
5: **while** $l_{\min} \leq i$ **do**
6:     Assign $\mathcal{C}_i(R_{\eta(i)}) \leftarrow R_{\eta(i)} \cup \{a \in \text{Children}(p)$ for some $p \in R_{\eta(i)} \mid l(a) = i\}$
7:     Set $R_i = \{a \in \mathcal{C}_i(R_{\eta(i)}) \mid d(p, a) \leq 2^{i+1}\}$
8:     **if** R_i is emptyset **then**
9:         Pick $a \in R_{\eta(i)}$ minimizing $d(p, a)$.
10:        Set $l(p) = \eta(i)$ and define a to be the parent of p and **exit**.
11:    **end if**
12:    $t = \max_{a \in R_i} \text{Next}(a, i, \mathcal{T}(W))$ {If R_i has no children we set $t = l_{\min} - 1$}
13:    $\eta(i) \leftarrow i$ and $i \leftarrow t$ {if $t = -\infty$, then $t + 1 = -\infty$}
14: **end while**
15: Set $m \leftarrow \max_{z < l_{\min}} \{z \mid d(p, R_{\eta(i)}) > 2^z\}$
16: Pick $a \in R_{\eta(i)}$ minimizing $d(p, a)$.
17: Set $l(p) = m$ and define a to be the parent of p

---

there exists a node $q \in R_{l_{\min}}$, in such a way that $d(q, p) \leq 2^{m+1}$. We can choose q to be the parent of node p. Since the rest of tree is unchanged, condition (3.2.1b) holds.

To check (3.2.1c) let $h \leq l(p)$ be arbitrary level. Let $\theta \in C_h$ be any other node satisfying $l(\theta) \geq h$. Assume first that p is chosen on lines 15 - 17, then R_h will always consist of only one node p. Since $\theta \in C_h$, there exists a node q and index j, in such a way that q is ancestor θ and $q \in \mathcal{C}_j(R_{\eta(j)} \setminus R_j)$. Since q is an ancestor of θ there is a chain of nodes (a_z) satisfying $a_0 = \theta$ and $a_s = q$ so that for all $z \in [0, s-1]$ node a_{z+1} is parent of a_z. Since $l(\theta) \geq h$ we know that $l(a_1) \geq h+1$, generally $l(a_t) \geq h+z$ for any $z \in [0, s-1]$. Then

$$d(q, \theta) \leq \sum_{z=0}^{s-1} d(a_z, a_{z+1}) \leq \sum_{z=0}^{s-1} 2^{h+z} \leq \sum_{x=h+1}^{j} 2^x = (2^{j+1} - 2^{h+1}).$$

The triangle inequality gives $d(p, r) \geq d(p, q) - d(q, r) > 2^{j+1} - (2^{j+1} - 2^{h+1}) \geq 2^{h+1}$, then $d(p, \theta) > 2^h$. Since θ was chosen arbitrarily, we get $d(C_h, p) > 2^h$, therefore (3.2.1c) holds. $\qquad \square$

Lemma 3.5.6 (time complexity of a key step for $\mathcal{T}(R)$)**.** *Arbitrarily order all points of a finite reference set R in a metric space (X, d) starting from the root: $r = p_1, p_2, \ldots, p_{|R|}$. Set $W_1 = \{r\}$ and*

$W_{y+1} = W_y \cup \{p_y\}$ *for* $y = 1, ..., |R| - 1$. *Then Algorithm 3.5.3 builds a compressed cover tree* $\mathcal{T}(R)$ *in time*

$$O\Big((c_m(R))^8 \cdot \max_{y=1,...,|R|-1} L(\mathcal{T}(W_y), p_y) \cdot |R|\Big),$$

where $c_m(R)$ *is the minimized expansion constant from Definition 3.1.4.*   ∎

Proof. The worst-case time complexity of Algorithm 3.5.3 is dominated by lines $3 - 5$ which call Algorithm 3.5.4 $O(|R|)$ times in total.

Assume that we have already constructed a cover tree on set $\mathcal{T}(W_y)$, the goal Algorithm 3.5.4 is to construct tree $\mathcal{T}(W_y \cup \{p_{y+1}\})$. By Definition 3.5.2 loop on lines 5-14 is performed $L(\mathcal{T}(W_y), p_{y+1})$ times. Let R_* be the maximal size of set R_i during all iterations $i \in L(\mathcal{T}(W_y), p_{y+1})$. By Lemma 3.2.9 since $W_{y+1} \subseteq R \subseteq X$ we have

$$|\mathcal{C}_i(R_{\eta(i)})| \le c_m(W_{y+1})^4 \cdot |R_*| \le c_m(R)^4 \cdot |R_*|$$

nodes, where $\mathcal{C}_{\eta(i)}(R_{\eta(i)})$ is defined in line 6. Therefore both, lines 7 and 6 take at most $c_m(R)^4 |R_*|$ time. In line 12 we handle $|R_*|$ elements, for each of them we can retrieve index $\text{Next}(a, i, \mathcal{T}(W_y))$ in $O(1)$ time, since for every $a \in \mathcal{T}(R)$ we can update the last index j, when a had children on level j in line 6. Therefore line 12 takes at most $O(|R_*|)$ time. Lines 15-17 take at most $O(|R_*|)$ time. Let us now bound $|R_*|$ during the whole run-time of the algorithm.

Let i be an arbitrary level. Note that $R_i \subseteq B(p, 2^{i+1}) \cap C_i$ where C_i is a ith cover set of $\mathcal{T}(R)$. Since C_i is 2^i-spares subset of R we can apply packing Lemma 3.2.8 with $r = 2^{i+1}$ and $\delta = 2^i$ to obtain $|B(p, 2^{i+1}) \cap C_i| \le (c_m(W))^4$. Lemma 3.1.5 implies $(c_m(W))^4 \le (c_m(R))^4$, therefore $|B(p, 2^i) \cap C_i| \le (c_m(R))^4$.

The time complexity of loop 5 - 14 in Algorithm 3.5.4 is dominated by line 6 that has time $O(|C(R_i)|) \le O((c_m(R))^4 \cdot |R_*|) \le O((c_m(R))^8)$. Then the whole Algorithm 3.5.3 has time

$$O((c_m(R))^8 \cdot \max_{y=2,...,|R|} L(\mathcal{T}(W_{y-1}), p_y) \cdot |R|)$$

as desired.   □

Theorem 3.5.7 (time complexity of $\mathcal{T}(R)$ via aspect ratio). *Let R be a finite subset of a metric space* (X, d) *having aspect ratio* $\Delta(R)$. *Algorithm 3.5.3 builds a compressed cover tree* $\mathcal{T}(R)$ *in time* $O((c_m(R))^8 \cdot \log_2(\Delta(R)) \cdot |R|)$, *where* $c_m(R)$ *is the minimized expansion constant from Definition 3.1.4.*

Proof. In Lemma 3.5.6 use the upper bounds due to Lemma 3.2.15 as follows:

$$\max_{y \in 2,\dots,|R|} |L(\mathcal{T}(W_{y-1}), p_y)| \le H(\mathcal{T}(R)) \le 1 + \log_2(\Delta(R)).$$ □

Lemma 3.5.8. *Let (X, d) be a metric space and let $W \subseteq X$ be its finite subset. Let $q \in X \setminus W$ be an arbitrary point. Let $i \in L(\mathcal{T}(W), q)$ be arbitrarily iteration of Definition 3.5.2. Assume that $t = \eta(\eta(i+1))$ is defined. Then there exists $p \in R$ satisfying $2^{i+1} < d(p, q) \le 2^{t+1}$.*

Proof. Note first that since $\eta(i+3) \in L(\mathcal{T}(R), q)$, there exists distinct $u \in R_{\eta(\eta(i+3))}$ and $v \in \mathcal{C}_{\eta(i+1)}(R_{\eta(\eta(i+1))})$, in such a way that u is the parent of v. Let us show that both of u, v cant belong to set R_i. Assume contrary that both $u, v \in R_i$. Then by line 7 of Algorithm 3.5.4 we have $d(v, q) \le 2^{i+1}$ and $d(u, q) \le 2^{i+1}$. By triangle inequality $d(u, v) \le d(u, q) + d(q, v) \le 2^{i+2} \le 2^{\eta(i+1)}$. Recall that we denote a level of a node by l. On the other hand we have $l(u) \ge \eta(i+1)$ and $l(v) \ge \eta(i+1)$, by separation condition of Definition 3.2.1 we have $d(u, v) > 2^{\eta(i+1)}$, which is a contradiction. Therefore only one of $\{u, v\}$ can belong to R_i. It sufficies two consider the two cases below:

Assume that $v \notin R_i$. Since v is children of u we have $d(u, v) \le 2^{\eta(i+1)+1}$. By line 7 of Algorithm 3.5.4 we have $d(u, q) \le 2^{\eta(i+1)+1}$. By triangle inequality

$$d(v, q) \le d(v, u) + d(u, q) \le 2^{\eta(i+1)+1} + 2^{\eta(i+1)+1} \le 2^{\eta(i+1)+2} \le 2^{\eta(\eta(i+1))+1}$$

Since $v \notin R_i$ there exists level t having $\eta(i+1) \ge t \ge i$ and $v \in \mathcal{C}_t(R_{\eta(t)}) \setminus R_t$. Therefore by line 7 of Algorithm 3.5.4 we have $d(q, v) > 2^{t+1} \ge 2^{i+1}$. It follows that we have found point $v \in R$ satisfying $2^{i+1} < v \le 2^{\eta(\eta(i+1))+1}$. Therefore $p = v$, is the desired point.

Assume that $u \notin R_i$. Since $u \in R_{\eta(\eta(i+1))}$, by line 7 of Algorithm 3.5.4 we have $d(u, q) \le 2^{\eta(\eta(i+1))+1}$. On the other hand since $u \notin R_i$, there exists level t having $\eta(i+3) \ge t \ge i$ and $u \in \mathcal{C}_t(R_{\eta(t)}) \setminus R_t$. Therefore by line 7 of Algorithm 3.5.4 we have $d(q, u) > 2^{t+2} \ge 2^{i+2}$. It follows that we have found point $u \in R$ satisfying $2^{i+1} < u \le 2^{\eta(\eta(i+1))+1}$. Therefore $p = u$, is the desired point.

□

Lemma 3.5.9. *Let A, W be finite subsets of a metric space X satisfying $W \subseteq A \subseteq X$, take $q \in A \setminus W$. Given a compressed cover tree $\mathcal{T}(W)$ on W, Algorithm 3.5.4 runs lines 5-14 this number of times: $|L(\mathcal{T}(W), q)| = O(c(A)^2 \cdot \log_2(|A|))$.* ∎

Proof. Let $x \in L(\mathcal{T}(R), q)$ be the lowest level of $L(\mathcal{T}(R), q)$. Define $s_1 = \eta(\eta(x) + 1)$ and let $s_i = \eta(\eta(\eta(s_{i-1} + 1)) + 1)$, if it exists. Assume that s_{n+1} is the last sequence element for which

$\eta(\eta(\eta(s_{n-1}+1))+1)$ is defined. Define $S = \{s_1, ..., s_n\}$. For every $i \in \{1, ..., n\}$ let p_i be the point provided by Lemma 3.5.8 that satisfies

$$2^{s_i+1} < d(p_i, q) \leq 2^{\eta(\eta(s_i+1))+1}.$$

Let P be the sequence of points p_i. Denote $n = |P| = |S|$. Let us show that S satisfies the conditions of Lemma 3.2.11. Note that:

$$4 \cdot d(p_i, q) \leq 4 \cdot 2^{\eta(\eta(s_i+1))+1} \leq 2^{\eta(\eta(s_i+1))+3} \leq 2^{\eta(\eta(\eta(s_i+1))+1)+1} \leq 2^{s_{i+1}+1} \leq d(p_{i+1}, q)$$

By Lemma 3.2.11 applied for set A and sequence P we get:

$$|\bar{B}(q, \frac{4}{3}d(q, p_n))| \geq (1 + \frac{1}{c(R)^2})^n \cdot |\bar{B}(q, \frac{1}{3}d(q, p_1))|$$

Since $\eta(x) \in L(\mathcal{T}(R), q)$, there exists some point $u \in R_{\eta(x)}$. By definition of R_i we have $d(u, q) \leq 2^{\eta(x)+1}$. Also

$$2^{\eta(\eta(x)+1)-1} \leq \frac{2^{\eta(\eta(x)+1)+1}}{3} < \frac{d(q, p_1)}{3}$$

It follows that:

$$1 \leq |\bar{B}(q, 2^{\eta(x)+1})| \leq |\bar{B}(q, 2^{\eta(\eta(x)+1)-1})| \leq |\bar{B}(q, \frac{d(q, p_1)}{3})|$$

Therefore we have

$$|A| \geq \frac{|\bar{B}(q, \frac{4}{3} \cdot d(q, p_n))|}{|\bar{B}(q, \frac{1}{3} \cdot d(q, p_1))|} \geq (1 + \frac{1}{c(A)^2})^n$$

Note that $c(A) \geq 2$ by definition of expansion constant. Then by applying log and by using Lemma 3.2.12 we obtain: $c(A)^2 \log(A) \geq n = |S|$. Let x be minimal level of $L(\mathcal{T}(W), q)$ and let y be the maximal level of $L(\mathcal{T}(W), q)$ Note that S is a sub sequence of L in such a way that:

- $[x, s_1] \cap L(\mathcal{T}(R), q) \leq 3$,

- for all $i \in 1, ..., n$ we have $[s_i, s_{i+1}] \cap L(\mathcal{T}(R), q) \leq 6$

- $[s_n, y] \cap L(\mathcal{T}(R), q) < 12$

Since segments $[x, s_1], [s_1, s_2], ..., [s_2, s_n], [s_n, y]$ cover $|L(\mathcal{T}(R), q)|$, it follows that $|S| \geq \frac{|L(\mathcal{T}(R), q)|}{12}$. We obtain that

$$|L(\mathcal{T}(R), q)| \leq 12 \cdot c(A)^2 \cdot \log_2(|A|),$$

which proves the claim. □
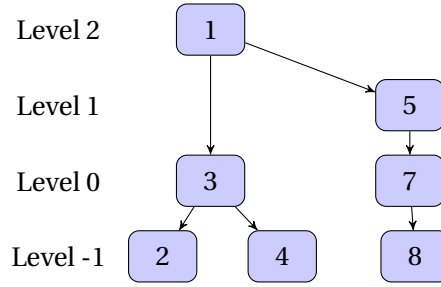
Figure 3.9: Consider a compressed cover tree $\mathcal{T}(R)$ that was built on set $R = \{1,2,3,4,5,7,8\}$. Let $\mathcal{S}_i(p, \mathcal{T}(R))$ be a distinctive descendant set of Definition 3.6.1. Then $V_2(1) = \varnothing$, $V_1(1) = \{5\}$ and $V_0(1) = \{3,5,7\}$. And also $\mathcal{S}_2(1, \mathcal{T}(R)) = \{1,2,3,4,5,7,8\}$, $\mathcal{S}_1(1, \mathcal{T}(R)) = \{1,2,3,4\}$ and $\mathcal{S}_0(1, \mathcal{T}(R)) = \{1\}$.

Theorem 3.5.10 (time complexity of $\mathcal{T}(R)$ via expansion constants)**.** *Let R be a finite subset of a metric space (X, d). Let A be a finite subset of X satisfying $R \subseteq A \subseteq X$. Then Algorithm 3.5.3 builds a compressed cover tree $\mathcal{T}(R)$ in time $O((c_m(R))^8 \cdot c(A)^2 \cdot \log_2(|A|) \cdot |R|)$, see $c(A), c_m(R)$ in Definition 3.1.4.* ∎

Proof. It follows from Lemmas 3.5.9 and 3.5.6. □

Corollary 3.5.11. *Let R be a finite subset of a metric space (X, d). Then Algorithm 3.5.3 builds a compressed cover tree $\mathcal{T}(R)$ in time $O((c_m(R))^8 \cdot c(R)^2 \cdot \log_2(|R|)) \cdot |R|)$, where $c(R), c_m(R)$ appeared in Definition 3.1.4.* ∎

Proof. The proof follows from Theorem 3.5.10 by setting $A = R$. □

3.6   Distinctive descendant sets

This section introduces auxiliary concepts for future proofs. The main concept is a distinctive descendant set in Definition 3.6.1. The distinctive descendant set at a level i of a node $p \in \mathcal{T}(R)$ in a compressed cover tree corresponds to the set of descendants of a copy of node p at level i in the original implicit cover tree $T(R)$. Other important concepts are λ-point of Definition 3.6.7 that is used in Algorithm 3.7.2 as an approximation for k-nearest neighboring point. The β-point property of λ-point defined in Lemma 3.6.17 plays a major role in the proof of the main worst-case time complexity result Theorem 3.7.14.

Definition 3.6.1 (Distinctive descendant sets)**.** Let $R \subseteq X$ be a finite reference set with a cover tree $\mathcal{T}(R)$. For any node $p \in \mathcal{T}(R)$ in a compressed cove tree on a finite set R and $i \le l(p) - 1$,

set $V_i(p) = \{u \in \text{Descendants}(p) \mid i \le l(u) \le l(p) - 1\}$. If $i \ge l(p)$, then set $V_i(p) = \emptyset$. For any level $i \le l(p)$, the *distinctive descendant set* is $\mathcal{S}_i(p, \mathcal{T}(R)) = \text{Descendants}(p) \setminus \bigcup_{u \in V_i(p)} \text{Descendants}(u)$ and has the size $|\mathcal{S}_i(p, \mathcal{T}(R))|$. ∎

Lemma 3.6.2 (Distinctive descendant set inclusion property)**.** *In conditions of Definition 3.6.1 let $p \in R$ and let i, j be integers satisfying $l_{\min}(\mathcal{T}(R)) \le i \le j \le l(p) - 1$. Then $\mathcal{S}_i(p, \mathcal{T}(R)) \subseteq \mathcal{S}_j(p, \mathcal{T}(R))$.*

Essential levels of a node $p \in \mathcal{T}(R)$ have 1-1 correspondence to the set consisting of all nodes containing p in the explicit representation of cover tree in [9], see Figure 3.4 middle.

Definition 3.6.3 (Essential levels of a node)**.** Let $R \subseteq X$ be a finite reference set with a cover tree $\mathcal{T}(R)$. Let $q \in \mathcal{T}(R)$ be a node. Let (t_i) for $i \in \{0, 1, ..., n\}$ be a sequence of $H(\mathcal{T}(R))$ in such a way that $t_0 = l(q)$, $t_n = l_{\min}(\mathcal{T}(R))$ and for all i we have $t_{i+1} = \text{Next}(q, t_i, \mathcal{T}(R))$. Define the set of essential indices $\mathcal{E}(q, \mathcal{T}(R)) = \{t_i \mid i \in \{0, ..., n\}\}$. ∎

Lemma 3.6.4 (Number of essential levels)**.** *Let $R \subseteq X$ be a finite reference set with a cover tree $\mathcal{T}(R)$. Then $\sum_{p \in R} |\mathcal{E}(p, \mathcal{T}(R))| \le 2 \cdot |R|$, where $\mathcal{E}(p, \mathcal{T}(R))$ appears in Definition 3.6.3.* ∎

Proof. Let us prove this claim by induction on size $|R|$. In basecase $R = \{r\}$ and therefore $|\mathcal{E}(r, \mathcal{T}(R))| = 1$. Assume now that the claim holds for any tree $\mathcal{T}(R)$, where $|R| = m$ and let us prove that if we add any node $v \in X \setminus R$ to tree $\mathcal{T}(R)$, then $\sum_{p \in R} |\mathcal{E}(p, \mathcal{T}(R \cup \{v\}))| \le 2 \cdot |R| + 2$. Assume that we have added u to $\mathcal{T}(R)$, in such a way that v is its new parent. Then $|\mathcal{E}(p, \mathcal{T}(R \cup \{v\}))| = |\mathcal{E}(p, \mathcal{T}(R))| + 1$ and $|\mathcal{E}(v, \mathcal{T}(R \cup \{v\}))| = 1$. We have:

$$\sum_{p \in R \cup \{u\}} |\mathcal{E}(p, \mathcal{T}(R))| = \sum_{p \in R} |\mathcal{E}(p, \mathcal{T}(R))| + 1 + |\mathcal{E}(v, \mathcal{T}(R \cup \{v\}))| \le 2 \cdot |R| + 2 \le 2(|R \cup \{v\}|)$$

which completes the induction step. □

Lemma 3.6.6. *Let R be a finite subset of a metric space. Let $\mathcal{T}(R)$ be a compressed cover tree on R. Then, Algorithm 3.6.5 computes the sizes $|\mathcal{S}_i(p, \mathcal{T}(R))|$ for all $p \in R$ and essential levels $i \in \mathcal{E}(p, \mathcal{T}(R))$ in time $O(|R|)$.* ∎

Proof. By Lemma 3.6.4 we have $\sum_{p \in R} |\mathcal{E}(p, \mathcal{T}(R))| \le 2 \cdot |R|$. Since CountDistinctiveDescendants is called once for every any combination $p \in R$ and $i \in \mathcal{E}(p, \mathcal{T}(R))$ it follows that the time complexity of Algorithm 3.6.5 is $O(R)$.

□

Algorithm 3.6.5 This algorithm returns sizes of distinctive descendant set $\mathcal{S}_i(p, \mathcal{T}(R))$ for all essential levels $i \in \mathcal{E}(p, \mathcal{T}(R))$

1: **Function** : CountDistinctiveDescendants(Node p, a level i of $\mathcal{T}(R)$)
2: **Output** : an integer
3: **if** $i > l_{\min}(\mathcal{T}(Q))$ **then**
4:     **for** $q \in$ Children(p) having $l(p) = i - 1$ or $q = p$ **do**
5:         Set $s = 0$
6:         $j \leftarrow 1 + \text{Next}(q, i - 1, \mathcal{T}(R))$
7:         $s \leftarrow s + \text{CountDistinctiveDescendants}(q, j)$
8:     **end for**
9: **else**
10:     Set $s = 1$
11: **end if**
12: Set $|\mathcal{S}_i(p)| = s$ and **return** s

Recall that the neighborhood $N(q; r) = \{p \in C \mid d(q, p) \leq d(q, r)\}$ was introduced in Definition 3.1.1.

Definition 3.6.7 (λ-point)**.** Fix a query point q in a metric space (X, d) and fix any level $i \in \mathbb{Z}$. Let $\mathcal{T}(R)$ be its compressed cover tree on a finite reference set $R \subseteq X$. Let C be a subset of a cover set C_i from Definition 3.2.1 satisfying $\sum_{p \in C} |\mathcal{S}_i(p, \mathcal{T}(R))| \geq k$, where $\mathcal{S}_i(p, \mathcal{T}(R))$ is the distinctive descendant set from Definition 3.6.1. For any $k \geq 1$, define $\lambda_k(q, C)$ as a point $\lambda \in C$ that minimizes $d(q, \lambda)$ subject to $\sum_{p \in N(q; \lambda)} |\mathcal{S}_i(p, \mathcal{T}(R))| \geq k$.   ∎

Algorithm 3.6.8 Finding k-lowest element of a finite subset $A \subseteq R$ with priority function $f : A \to \mathbb{R}$

1: **Input:** Ordered subset $A \subseteq R$, priority function $f : A \to \mathbb{R}$, an integer $k \in \mathbb{Z}$
2: Initialize an empty max-binary heap B and an empty array D on points A.
3: **for** $p \in A$ **do**
4:     add p to B with priority $f(p)$
5:     **if** $|H| \geq k$ **then**
6:         remove the point with a maximal value from B
7:     **end if**
8: **end for**
9: Transfer points from the binary heap B to the array D in reverse order.
10: **return** D.

Lemma 3.6.10. *Let $A \subseteq R$ be a finite subset and let $f : A \to \mathbb{R}$ be a priority function and let $k \in \mathbb{Z}_+$. Then Algorithm 3.6.8 finds k-smallest elements of A in time $|A| \cdot \log_2(k)$*

Algorithm 3.6.9 Computation of a λ-point of Definition 3.6.7 in line 7 of Algorithm 3.7.2

1: **Input:** A point $q \in X$, a subset C of a level set C_i of a compressed cover tree $\mathcal{T}(R)$, an integer $k \in \mathbb{Z}$
2: Define $f : C \to \mathbb{R}$ by setting $f(p) = d(p,q)$.
3: Run Algorithm 3.6.8 on inputs (C, f, k) and retrieve array D.
4: Find the smallest index j such that $\sum_{t=0}^{j} |\mathcal{S}_i(D[t], \mathcal{T}(R))| \geq k$.
5: **return** $\lambda = D[j]$.

Proof. Adding and removing element from binary heap data structure [25, section 6.5] takes at most $O(\log(n))$ time, where n is the size of binary heap. Since the size of our binary heap is capped at k and we add/remove at most $|A|$ elements, the total time complexity is $O(|A| \cdot \log_2(k))$. $\qquad\square$

Lemma 3.6.11 (time complexity of a λ-point)**.** *In the conditions of Definition 3.6.7, the time complexity of Algorithm 3.6.9 is $O(|C| \cdot \log_2(k))$.*

Proof. Note that in line 4 we have $|\mathcal{S}_i(D[t], \mathcal{T}(R))| \geq 1$ for all $t = 0, ..., j$. Therefore the time complexity of line 4 is $O(k)$. By Lemma 3.6.10 The time complexity of line 3 is $O(|C| \cdot \log_2(k))$, which proves the claim. $\qquad\square$

Lemma 3.6.12 (separation)**.** *In the conditions of Definition 3.6.1, let $p \neq q$ be nodes of $\mathcal{T}(R)$ with $l(p) \geq i$, $l(q) \geq i$. Then $\mathcal{S}_i(p, \mathcal{T}(R)) \cap \mathcal{S}_i(q, \mathcal{T}(R)) = \emptyset$.* $\qquad\blacksquare$

Proof. Without loss of generality assume that $l(p) \geq l(q)$. If q is not a descendant of p, the lemma holds trivially due to Descendants$(q) \cap$ Descendants$(p) = \emptyset$. If q is a descendant of p, then $l(q) \leq l(p) - 1$ and therefore $q \in V_i(p)$. It follows that $\mathcal{S}_i(p, \mathcal{T}(R)) \cap$ Descendants$(q) = \emptyset$ and therefore

$$\mathcal{S}_i(p, \mathcal{T}(R)) \cap \mathcal{S}_i(q, \mathcal{T}(R)) \subseteq \mathcal{S}_i(p, \mathcal{T}(R)) \cap \text{Descendants}(q) = \emptyset.$$

$\qquad\square$

Lemma 3.6.13 (Sum lemma)**.** *In the notations of Definition 3.6.1 assume that i is arbitrarily index and a subset $V \subseteq R$ satisfies $l(p) \geq i$ for all $p \in V$. Then*

$$\left| \bigcup_{p \in V} \mathcal{S}_i(p, \mathcal{T}(R)) \right| = \sum_{p \in V} |\mathcal{S}_i(p, \mathcal{T}(R))|.$$

Proof. Proof follows from Lemma 3.6.12. $\qquad\square$

By Lemma 3.6.13 in Definition 3.6.7 one can assume that $|\bigcup_{p \in C} \mathcal{S}_i(p, \mathcal{T}(R))| \geq k$.

Lemma 3.6.14. *In the notations of Definition 3.6.1, let $p \in \mathcal{T}(R)$ be any node. If $w \in \mathcal{S}_i(p, \mathcal{T}(R))$ then either $w = p$ or there exists $a \in \text{Children}(p) \setminus \{p\}$ such that $l(a) < i$ and $w \in \text{Descendants}(a)$.* ∎

Proof. Let $w \in \mathcal{S}_i(p)$ be an arbitrary node satisfying $w \neq p$. Let s be the node-to-root path of w. The inclusion $\mathcal{S}_i(p) \subseteq \text{Descendants}(p)$ implies that $w \in \text{Descendants}(p)$. Let $a \in \text{Children}(p) \setminus \{p\}$ be a child on the path s. If $l(a) \geq i$ then $a \in V_i(p)$. Note that $w \in \text{Descendants}(a)$. Therefore $w \notin \mathcal{S}_i(p)$, which is a contradiction. Hence $l(a) < i$. □

Lemma 3.6.15. *In the notations of Definition 3.6.1, let $p \in \mathcal{T}(R)$ be any node. If $w \in \mathcal{S}_i(p, \mathcal{T}(R))$ then $d(w, p) \leq 2^{i+1}$.* ∎

Proof. By Lemma 3.6.14 either $w = \gamma$ or $w \in \text{Descendants}(a)$ for some $a \in \text{Children}(\gamma) \setminus \{\gamma\}$ for which $l(a) < i$. If $w = \gamma$, then trivially $d(\gamma, w) \leq 2^i$. Else w is a descendant of a, which is a child of node γ on level $i - 1$ or below, therefore by Lemma 3.2.7 we have $d(\gamma, w) \leq 2^i$ anyway. □

Lemma 3.6.16. *Let R be a finite subset of a metric pace. Let $\mathcal{T}(R)$ be a compressed cover tree on R. Let $R_j \subseteq C_j$, where C_j is the ith cover set of $\mathcal{T}(R)$. Let $i = \max_{p \in R_j} \text{Next}(p, j, \mathcal{T}(R))$. Set $\mathcal{C}_j(R_j) = R_j \cup \{a \in \text{Children}(p) \text{ for some } p \in R_i \mid l(a) = i\}$. Then*

$$\bigcup_{p \in \mathcal{C}_j(R_j)} \mathcal{S}_i(p, \mathcal{T}(R)) = \bigcup_{p \in R_j} \mathcal{S}_j(p, \mathcal{T}(R)).$$

Proof. Let $a \in \bigcup_{p \in \mathcal{C}_j(R_j)} \mathcal{S}_i(p, \mathcal{T}(R))$ be an arbitrary node. Then there exits $u \in \mathcal{C}_j(R_j)$ having $a \in \mathcal{S}_i(u, \mathcal{T}(R))$. By definition of index i, either $u \in R_j$ or u has a parent in R_j. If $u \in R_j$ then we note that $V_j(u) \subseteq V_i(u)$. Since $a \notin V_i(u)$, we also have $a \notin V_j(u)$.

Otherwise let w be a parent of u. Therefore there are no descendants of w in having level in interval $[l(u) + 1, l(p) - 1]$. Since $l(u) = i$ and $j > i$ it follows that $V_j(w) = \emptyset$. Denote w to be the lowest level ancestor of u on level j. By cases above we have $a \notin V_j(w)$. Therefore it follows that

$$a \in \mathcal{S}_j(w, \mathcal{T}(R)) \subseteq \bigcup_{p \in R_j} \mathcal{S}_j(p, \mathcal{T}(R)).$$

To prove the converse inclusion assume now that $a \in \bigcup_{p \in R_j} \mathcal{S}_j(p, \mathcal{T}(R))$. Then $a \in \mathcal{S}_j(v, \mathcal{T}(R))$ for some $w \in R_j$. Assume that w has no children at the level i. Then $V_j(w) = V_i(w)$ and

$$a \in \mathcal{S}_i(w, \mathcal{T}(R)) \subseteq \bigcup_{p \in \mathcal{C}_j(R_j)} \mathcal{S}_i(p, \mathcal{T}(R)).$$

Assume now that w has children at the level i. If there exists $b \in \text{Children}(w)$ for which $a \in \text{Descendants}(b)$. Since $V_i(b) = \emptyset$, we conclude that

$$a \in \mathcal{S}_i(b, \mathcal{T}(R)) \subseteq \bigcup_{p \in \mathcal{C}_j(R_j)} \mathcal{S}_i(p, \mathcal{T}(R)).$$

Assume that $a \notin \text{Descendants}(b)$ for all $b \in \text{Children}(w)$ with $l(b) = i$. Then $a \in \text{Descendants}(w)$ and $a \notin \text{Descendants}(b')$ for any $b' \in V_j(w)$. Then $a \in \mathcal{S}_i(w, \mathcal{T}(R))$ and the proof finishes:

$$\bigcup_{p \in R_j} \mathcal{S}_j(p, \mathcal{T}(R)) \subseteq \bigcup_{p \in \mathcal{C}_j(R_j)} \mathcal{S}_i(p, \mathcal{T}(R)).$$

$\square$

Lemma 3.6.17 (β-point)**.** *In the notations of Definition 3.6.7, let $C \subseteq C_i$ so that $\cup_{p \in C} \mathcal{S}_i(p, \mathcal{T}(R))$ contains all k-nearest neighbors of q. Set $\lambda = \lambda_k(q, C)$. Then R has a point β among the first k nearest neighbors of q such that $d(q, \lambda) \leq d(q, \beta) + 2^{i+1}$.* ∎

Proof. We show that R has a point β among the first k nearest neighbors of q such that

$$\beta \in \bigcup_{p \in C} \mathcal{S}_i(p, \mathcal{T}(R)) \setminus \bigcup_{p \in N(q, \lambda) \setminus \{\lambda\}} \mathcal{S}_i(p, \mathcal{T}(R)).$$

Lemma 3.6.13 and Definition 3.6.7 imply that

$$\left| \bigcup_{p \in N(q, \lambda) \setminus \{\lambda\}} \mathcal{S}_i(p, \mathcal{T}(R)) \right| = \sum_{p \in N(q, \lambda) \setminus \{\lambda\}} |\mathcal{S}_i(p, \mathcal{T}(R))| < k.$$

Since $\cup_{p \in C} \mathcal{S}_i(p, \mathcal{T}(R))$ contains all k-nearest neighbors of q, a required point $\beta \in R$ exists.

Let us now show that β satisfies $d(q, \lambda) \leq d(q, \beta) + 2^{i+1}$. Let $\gamma \in C \setminus N(q, \lambda) \cup \{\lambda\}$ be such that $\beta \in \mathcal{S}_i(\gamma, \mathcal{T}(R))$. Since $\gamma \notin N(q, \lambda) \setminus \{\lambda\}$, we get $d(\gamma, q) \geq d(q, \lambda)$. The triangle inequality says that $d(q, \gamma) \leq d(q, \beta) + d(\gamma, \beta)$. Finally Lemma 3.6.15 implies that $d(\gamma, \beta) \leq 2^{i+1}$. Then

$$d(q, \lambda) \leq d(q, \gamma) \leq d(q, \beta) + d(\gamma, \beta) \leq d(q, \beta) + 2^{i+1}$$

So β is a desired k-nearest neighbor satisfying $d(q, \lambda) \leq d(q, \beta) + 2^{i+1}$. $\square$

3.7 k-nearest neighbor search algorithm

This section is motivated by Counterexample 3.4.7 which showed that the proof of past time complexity result [9, Theorem 5] of nearest neighborhood search algorithm contained mistakes. The two main results of this sections are Corollary 3.7.9 and Theorem 3.7.14 which provide new time complexity results for k-nearest neighborhood problem, assuming that a compressed cover tree was already constructed for the reference set R. For the construction algorithm of compressed cover tree and its time complexity, we refer to Section 3.5.

The past mistakes are resolved by introducing a new Algorithm 3.7.2 for finding k-nearest neighbors that generalize and improves the original method for finding nearest neighbors using an implicit cover. [9, Algorithm 1]. The first improvement is λ-point of line 7 which allows us to search for all k-nearest neighbors of a given query point for any $k \geq 1$. The second improvement is a new loop break condition on line 9. The new loop break condition is utilized in the proof of Lemma 3.7.13 to conclude that the total number of performed iterations is bounded by $O(c(R)^2 \log(|R|))$ during whole run-time of the algorithm. The latter improvement closes the past gap in proof of [9, Theorem 5] by bounding the number of iterations independently from the explicit depth $D(\mathcal{T}(R))$ of Definition 3.4.2, that generated the past confusion.

Recall from Definition 3.6.3 that an essential set $\mathcal{E}(p, \mathcal{T}(R)) \subseteq H(\mathcal{T}(R)$ consists of all levels $i \in H(\mathcal{T}(R))$ for which p has non-trivial children in $\mathcal{T}(R)$ at level i. By Lemma 3.6.6 the sizes of distinctive descendants $|\mathcal{S}_i(p, \mathcal{T}(R))|$ can be precomputed in a linear time $O(|R|)$ for all $p \in R$ and $i \in \mathcal{E}(p, \mathcal{T}(R))$. Since the size of distinctive descendant set $|\mathcal{S}_i(p, \mathcal{T}(R))|$ can only change at indices $i \in \mathcal{E}(p, \mathcal{T}(R))$, we assume that the sizes of $|\mathcal{S}_i(p, \mathcal{T}(R))|$ can be retrieved in a constant time $O(1)$ for any $p \in R$ and $i \in H(\mathcal{T}(R))$ during the run-time of Algorithm 3.7.2.

Definition 3.7.1. Let R be a finite subset of a metric space (X, d). Let $\mathcal{T}(R)$ be a cover tree of Definition 3.2.1 built on R and let $q \in X$ be arbitrary point. Let $L(\mathcal{T}(R), q) \subseteq H(\mathcal{T}(R))$ be the set of all levels i during iterations of lines 4-18 of Algorithm 3.7.2 launched with inputs $\mathcal{T}(R), q$. If Algorithm 3.7.2 reaches line 14 at level $\varrho \in L(\mathcal{T}(R), q)$, then we say that is *special*. We denote $\eta(i) = \min_t \{t \in L(\mathcal{T}(R), q) \mid t > i\}$. ∎

Note that $\eta(i)$ of Definition 3.7.1 may be undefined. If $\eta(i)$ is defined, then by definition we have $\eta(i) \geq i + 1$. Let $d_k(q, R)$ be the distance of q to its kth nearest neighbor in R.

Example 3.7.4 (Simulated run of Algorithm 3.7.2)**.** Let R and $\mathcal{T}(R)$ be as in Example 3.2.3. Let $q = 0$ and $k = 5$. Figures 3.10, 3.11, 3.12 and 3.13 illustrate simulated run of Algorithm 3.7.2 on

---

Algorithm 3.7.2 k-nearest neighbor search by a compressed cover tree

---

1: **Input** : compressed cover tree $\mathcal{T}(R)$, a query point $q \in X$, an integer $k \in \mathbb{Z}_+$

2: Set $i \leftarrow l_{\max}(\mathcal{T}(R)) - 1$ and $\eta(l_{\max} - 1) = l_{\max}$

3: Let r be the root node of $\mathcal{T}(R)$. Set $R_{l_{\max}} = \{r\}$.

4: **while** $i \geq l_{\min}$ **do**

5:     Assign $\mathcal{C}_i(R_{\eta(i)}) \leftarrow R_{\eta(i)} \cup \{a \in \text{Children}(p) \text{ for some } p \in R_{\eta(i)} \mid l(a) = i\}$

6: {Recall that $\text{Children}(p)$ contains node p }

7:     Compute $\lambda = \lambda_k(q, \mathcal{C}_i(R_{\eta(i)}))$ from Definition 3.6.7 by Algorithm 3.6.9.

8:     Find $R_i = \{p \in \mathcal{C}_i(R_{\eta(i)}) \mid d(q, p) \leq d(q, \lambda) + 2^{i+2}\}$

9:     **if** $d(q, \lambda) > 2^{i+2}$ **then**

10:         Define list $S = \emptyset$

11:         **for** $p \in R_i$ **do**

12:             Update S by running Algorithm 3.7.3 on (p, i)

13:         **end for**

14:         Compute and **output** k-nearest neighbors of the query point q from set S.

15:     **end if**

16:     Set $j \leftarrow \max_{a \in R_i} \text{Next}(a, i, \mathcal{T}(R))$ {If such j is undefined, we set $j = l_{\min} - 1$}

17:     Set $\eta(j) \leftarrow i$ and $i \leftarrow j$.

18: **end while**

19: Compute and **output** k-nearest neighbors of query point q from the set $R_{l_{\min}}$.

Algorithm 3.7.3 The node collector called in line 12 of Algorithm 3.7.2.

1: **Input:** $p \in R$, index i.

2: **Output:** a list $S \subseteq R$ containing all nodes of $\mathcal{S}_i(p, \mathcal{T}(R))$.

3: Add p to list S.

4: **if** $i > l_{\min}(\mathcal{T}(R))$ **then**

5:     Set $j = \text{Next}(p, i, \mathcal{T}(R))$

6:     Set $C = \{a \in \text{Children}(p) \mid l(a) = j\}$

7:     **for** $u \in C$ **do**

8:         Call Algorithm 3.7.3 with (u, j).

9:     **end for**

10: **end if**

input $(\mathcal{T}(R), q, k)$. Recall that $l_{\max} = 2$ and $l_{\min} = -1$. During the iteration i of Algorithm 3.7.2 we maintain the following coloring: Points in R_i are colored orange. Points $\mathcal{C}_{\eta(i)}(R_{\eta(i)})$ (of line 5) that are not contained in R_i are colored yellow. The λ-point of line 7 is denoted by using purple color. All the nodes that were present in $R_{\eta(i)}$, but are no longer included in R_i will be colored red. Finally all the points that are selected as k-nearest neighbors of q are colored green in the final iteration. Nodes that haven't been yet visited or that will never be visited are colored white. Let $R_2 = \{8\}$. Consider the following steps:

Iteration $i = 1$: Figure 3.10 illustrates iteration $i = 1$ of the Algorithm 3.7.2. In line 6 we find $\mathcal{C}_2(R_2) = \{4, 8, 12\}$. Since node 4 minimizes distance $d(\mathcal{C}_2(R_2), 0)$ and distinctive descendant set $\mathcal{S}_2(4, \mathcal{T}(R))$ consists of 7 elements we get $\lambda = 4$ and therefore $d(q, \lambda) = 4 \le 2^{i+2} = 8$. In line 8 we find $R_1 = \{r \in C \mid d(0, r) \le d(q, \lambda) + 2^3 = 12\} = \{4, 8, 12\}$.

Iteration $i = 0$: Figure 3.11 illustrates iteration $i = 0$ of the Algorithm 3.7.2. In line 6 we find $\mathcal{C}_1(R_1) = \{2, 4, 6, 8, 10, 12, 14\}$. Since $|\mathcal{S}_1(2, \mathcal{T}(R))| = 3$, $|\mathcal{S}_1(4, \mathcal{T}(R))| = 1$ and $|\mathcal{T}_1(6)| = 3$ and 6 is the node with smallest to distance 0 satisfying $\sum_{p \in N(0,6)=\{2,4,6\}} |\mathcal{S}_1(p, \mathcal{T}(R))| \ge 5 = k$. It follows that $\lambda = 6$. In line 8 we find $R_0 = \{r \in \mathcal{C}(R_1) \mid d(0, r) \le d(q, \lambda) + 2^2 = 10\} = \{2, 4, 6, 8, 10\}$. Since $d(q, \lambda) > 2^{i+2} = 4$. We proceed into lines 9 - 15

Final block lines 9 - 15 for $i = 0$: Figure 3.12 marks all the points S discovered by line 12 as orange. Figure 3.13 illustrates the final selection of k points from set S that are selected as the final output $\{1, 2, 3, 4, 5\}$.
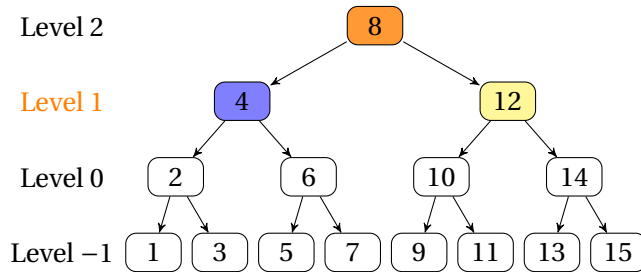
Figure 3.10: Iteration $i = 1$ of simulation in Example 3.7.4 of Algorithm 3.7.2
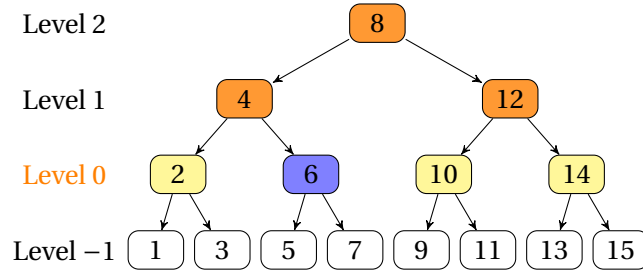
Figure 3.11: Iteration $i = 0$ of simulation in Example 3.7.4 of Algorithm 3.7.2
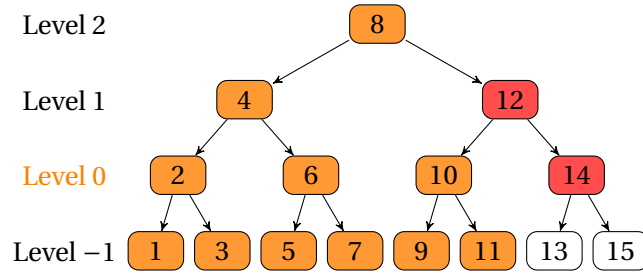
Figure 3.12: Line 12 of Iteration $i = 0$ of simulation in Example 3.7.4 of Algorithm 3.7.2
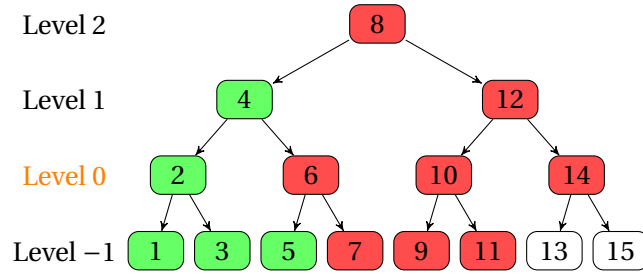
Figure 3.13: Line 14 of iteration $i = 0$ of simulation in Example 3.7.4 of Algorithm 3.7.2

Note that $\bigcup_{p \in R_i} \mathcal{S}_i(p, \mathcal{T}(R))$ is decreasing set for which $\bigcup_{p \in R_{l_{\max}}} \mathcal{S}_{l_{\max}}(p, \mathcal{T}(R)) = R$ and

$$\bigcup_{p \in R_{l_{\min}}} \mathcal{S}_{l_{\min}}(p, \mathcal{T}(R)) = R_{l_{\min}}.$$

Lemma 3.7.5 (k-nearest neighbors in the candidate set for all i)**.** *Let R be a finite subset of an ambient metric space (X, d), let $q \in X$ be a query point and let $k \in \mathbb{Z} \cap [1, \infty)$ be a parameter. Let $\mathcal{T}(R)$ be a compressed cover tree of R. Assume that $|R| \geq k$. Then for any iteration $i \in L(\mathcal{T}(R), q)$ of Definition 3.7.1 the candidate set $\bigcup_{p \in R_i} \mathcal{S}_i(p, \mathcal{T}(R))$ contains all k-nearest neighbors of q.* ∎

Proof. Since $R_{l_{\max}} = \{r\}$, where r is the root $\mathcal{T}(R)$ we have $\mathcal{S}_{l_{\max}}(r, \mathcal{T}(R)) = R$ and therefore any point among k-nearest neighbor of q is contained in $R_{l_{\max}}$. Let i be the largest index for which there exists a point among k-nearest neighbor of q that doesn't belong to $\bigcup_{p \in R_i} \mathcal{S}_i(p, \mathcal{T}(R))$. Let us denote such point by β, then:

$$\beta \in \bigcup_{p \in R_{\eta(i)}} \mathcal{S}_{\eta(i)}(p, \mathcal{T}(R)) \setminus \bigcup_{p \in R_i} \mathcal{S}_i(p, \mathcal{T}(R)).$$

By Lemma 3.6.16 we have

$$\bigcup_{p \in \mathcal{C}_{\eta(i)}(R_{\eta(i)})} \mathcal{S}_i(p, \mathcal{T}(R)) = \bigcup_{p \in R_{\eta(i)}} \mathcal{S}_{\eta(i)}(p, \mathcal{T}(R)) \tag{3.7.5.1}$$

Let λ be as in line 7 of Algorithm 3.7.2. By Equation (3.7.5.1) we have

$$\Big| \bigcup_{p \in \mathcal{C}_{\eta(i)}(R_{\eta(i)})} \mathcal{S}_i(p, \mathcal{T}(R)) \Big| \geq k,$$

therefore by Definition 3.6.7 such λ exists. Since $\beta \in \bigcup_{p \in \mathcal{C}_{\eta(i)}(R_{\eta(i)})} \mathcal{S}_i(p, \mathcal{T}(R))$, there exists $\alpha \in \mathcal{C}_{\eta(i)}(R_{\eta(i)})$ satisfying $\beta \in \mathcal{S}_i(\alpha, \mathcal{T}(R))$. By assumption it follows $\alpha \notin R_i$. By line 8 of the algorithm we have

$$d(\alpha, q) > d(q, \lambda) + 2^{i+2}. \tag{3.7.5.2}$$

Let w be arbitrary point in set $\bigcup_{p \in N(q;\lambda)} \mathcal{S}_i(p, \mathcal{T}(R))$. Therefore $w \in \mathcal{S}_i(\gamma, \mathcal{T}(R))$ for some $\gamma \in N(q;\lambda)$. By Lemma 3.6.15 applied on i we have $d(\gamma, w) \leq 2^{i+1}$. By Definition 3.6.7 since $\gamma \in N(q;\lambda)$ we have $d(q, \gamma) \leq d(q, \lambda)$. By (3.7.5.2) and the triangle inequality we obtain:

$$d(q, w) \leq d(q, \gamma) + d(\gamma, w) \leq d(q, \lambda) + 2^{i+1} < d(\alpha, q) - 2^{i+1} \tag{3.7.5.3}$$

On the other hand β is a descendant of α thus we can estimate:

$$d(q, \beta) \geq d(q, \alpha) - d(\alpha, \beta) \geq d(\alpha, q) - 2^{i+1} \tag{3.7.5.4}$$

By combining Inequality (3.7.5.3) with Inequality (3.7.5.4) we obtain $d(q, w) < d(q, \beta)$. Since w was arbitrary point from $\bigcup_{p \in N(q;\lambda)} \mathcal{S}_i(p, \mathcal{T}(R))$, that contains at least k points, β cannot be any k-nearest neighbor of q, which is a contradiction. $\qquad\square$

Theorem 3.7.6 (correctness of Algorithm 3.7.2)**.** *Algorithm 3.7.2 correctly finds all k nearest neighbors of query point q within the reference set R.* $\qquad\blacksquare$

Proof. Note that Algorithm 3.7.2 is terminated by either reaching line 19 or by going inside block 11 - 13.

Assume first that Algorithm 3.7.2 is terminated by reaching line 19. Claim follows directly from Lemma 3.7.5 by noting that since $i = l_{\min}$ all the nodes $p \in R_{l_{\min}}$ do not have any children. Therefore it follows $\bigcup_{p \in R_{l_{\min}}} S_i(p, \mathcal{T}(R)) = R_{l_{\min}}$. Thus all the k-nearest neighbors of q are contained in the set $R_{l_{\min}}$.

Assume then that block 11 - 13 is reached during some iteration $i \in L(\mathcal{T}(R), q)$. By Lemma 3.7.5 set $\bigcup_{p \in R_i} S_i(p, \mathcal{T}(R))$ contains all k-nearest neighbors of q. Note that in line 12 we collect all nodes of $\bigcup_{p \in R_i} S_i(p, \mathcal{T}(R))$ into single array S. Therefore in line 14 we correctly select k nearest neighbors of q from array S, which proves the claim. $\square$

Lemma 3.7.7. *Algorithm 3.7.2 has the following time complexities of its lines*

(a) $\max\{\#\text{Line}[4-9], \#\text{Line}[15-18], \#\text{Line}[19]\} = O\big(c_m(R)^{10} \cdot \log_2(k)\big)$;

(b) $\#\text{Line}[9-15] = O\big(|\bar{B}(q, 5d_k(q,R))| \cdot \log_2(k)\big)$.   ∎

Proof of Lemma *3.7.7.* **(a)** Let $\varrho \in L(\mathcal{T}(R), q)$ be as in Definition 3.7.1. Note that if iteration ϱ is encountered, it becomes the last iteration of $L(\mathcal{T}(R), q)$. The total number of children encountered in line 6 during single iteration (4-18) is at most is at most $(c_m(R))^4 \cdot \max\limits_{i \in L(\mathcal{T}(R), q) \setminus \varrho} |R_i|$ by Lemma 3.2.9. From Lemma 3.6.11 we obtain that line 7, which launches Algorithm 3.6.9 takes at most

$$|\mathcal{C}(R_i)| \cdot \log_2(k) = (c_m(R))^4 \cdot \max\limits_{L(q, \mathcal{T}(R)) \setminus \varrho} |R_i| \cdot \log_2(k)$$

time. Line 8 never does more work than line 6, since in the worst case scenario $R_{\eta(i)}$ is copied to R_i in its current form. Line 16 handles $|R_i|$ nodes, since we can keep track of value of $\text{Next}(a, i, \mathcal{T}(R))$ of Definition 3.5.1 by updating it when necessary in line 6 we can retrieve its value in $O(1)$ time. Therefore maximal run-time of line 16 is $\max\limits_{i \in L(q, \mathcal{T}(R)) \setminus \varrho} |R_i|$. Final line 19 picks lowest k-elements $R_{\eta(i)}$ ranked by function $f(p) = d(p, q)$. By Lemma 3.6.10 it can be computed in time $O(\log_2(k) \cdot \max\limits_{L(q, \mathcal{T}(R)) \setminus \varrho} |R_i|)$. It follows that

$$\max(\#\text{Line}[4,9], \#\text{Line}[15,18], \#\text{Line}[19]) = O\Big(c_m(R)^4 \cdot \max\limits_{i \in L(q, \mathcal{T}(R)) \setminus \varrho} |R_i| \cdot \log_2(k)\Big) \qquad (3.7.7.1)$$

Let us now bound $\max_{i \in L(q, \mathcal{T}(R)) \setminus \varrho} |R_i|$, by showing $|R_i| \le c_m(R)^6$. Let C_i be the ith level of $\mathcal{T}(R)$

as in Definition 3.2.1. For all $i \in L(\mathcal{T}(R), q) \setminus \varrho$ we have:

$$R_i = \{r \in \mathcal{C}_i(R_{\eta(i)}) \mid d(p,q) \le d(q,\lambda) + 2^{i+2}\} \tag{3.7.7.2}$$

$$= B(q, d(q,\lambda) + 2^{i+2}) \cap \mathcal{C}_i(R_i) \tag{3.7.7.3}$$

$$\subseteq B(q, 2^{i+3}) \cap C_i \tag{3.7.7.4}$$

From cover-tree condition we know that all the points in C_i are separated by 2^i. We will now apply Lemma 3.2.8 with $t = 2^{i+3}$ and $\delta = 2^i$. Since $4\frac{t}{\delta} + 1 = 2^5 + 1 \le 2^6$ we obtain $\max\limits_{i \in L(q, \mathcal{T}(R)) \setminus \varrho} |R_i| \le |B(q, 2^{i+2}) \cap C_i| \le c_m(R)^6$. The claim follows by replacing $\max\limits_{i \in L(q, \mathcal{T}(R)) \setminus \varrho} |R_i|$ with $c_m(R)^6$ in (3.7.7.1).

(b) Let us now bound the run-time of #Line[9, 18]. which runs Algorithm 3.7.3 for all (p, i), where $p \in R_i$. Let $\mathcal{S}$ be a distinctive descendant set from Definition 3.6.1. Algorithm 3.7.3 visits every node $u \in \cup_{p \in R_i} \mathcal{S}_i(p, \mathcal{T}(R))$ once, therefore its running time is $O(\cup_{p \in R_i} |\mathcal{S}_i(p, \mathcal{T}(R))|)$. Let us now show that

$$\cup_{p \in R_i} \mathcal{S}_i(p, \mathcal{T}(R)) \subseteq \bar{B}(q, 5 d_k(q, R))$$

Note first that by Lemma 3.7.5 set $\cup_{p \in R_i} \mathcal{S}_i(p, \mathcal{T}(R))$ contains all k-nearest neighbors of q. Using Lemma 3.6.17 we find β among k-nearest neighbors of q satisfying $d(q,\lambda) \le d(q,\beta) + 2^{i+1}$. From assumption It follows $2^{i+1} \le d(q,\beta)$.

By line 9 we have $d(q,\lambda) \le 2^{i+1}$. By line 14 we perform depth-first traversal on

$$A = \cup_{p \in R_i} \mathcal{S}_i(p, \mathcal{T}(R)).$$

Let $u \in \cup_{p \in R_i} \mathcal{S}_i(p, \mathcal{T}(R))$ be arbitrary node and let $v \in R_i$ be such that $u \in \mathcal{S}_i(v, \mathcal{T}(R))$. By Lemma 3.6.15 we have $d(u,v) \le 2^{i+1}$. Since $v \in R_i$ we have $d(q,v) \le d(\lambda,q) + 2^{i+2}$. By triangle inequality

$$d(u,q) \le d(u,v) + d(v,q) \le 2^{i+1} + d(\lambda,v) + 2^{i+2} \le 2^{i+1} + 2^{i+1} + d(q,\beta) + 2^{i+2} \le 5 \cdot d(q,\beta)$$

It follows that $\cup_{p \in R_i} \mathcal{S}_i(p, \mathcal{T}(R)) \subseteq \bar{B}(q, 5 \cdot d(q,\beta))$. Let us now bound the time complexity of line 14. By Lemma 3.6.10 for any set A is takes $\log(k) \cdot |A|$ time to select k-lowest elements. We have:

$$\#\text{Line}[9, 18] = O(|\bar{B}(q, 5 \cdot d_k(q, R))| \cdot \log(k)).$$

$\square$

Theorem 3.7.8. *Let R be a finite set in a metric space (X, d), $c_m(R)$ be the minimized constant*

from Definition 3.1.4. Given a compressed cover tree $\mathcal{T}(R)$, Algorithm 3.7.2 finds all k nearest neighbors of a query point $q \in X$ in time

$$O\Big(\log_2(k) \cdot ((c_m(R))^{10} \cdot |L(q, \mathcal{T}(R))| + |\bar{B}(q, 5d_k(q, R))|)\Big), \text{ where}$$

$L(\mathcal{T}(R), q) \subseteq H(\mathcal{T}(R))$ *is the set of all levels i in lines 4-18 of Algorithm 3.7.2.*

Proof. Apply Lemma 3.7.7 to estimate the time complexity of Algorithm 3.7.2:
$O\big(|L(\mathcal{T}(R), q)| \cdot (\#\text{Line}[4-9] + \#\text{Line}[15-18] + \#\text{Line}[19]) + \#\text{Line}[9-15]\big).$ □

Corollary 3.7.9 gives a run-time bound using only minimized expansion constant $c_m(R)$, where if $R \subset \mathbb{R}^m$, then $c_m(R) \leq 2^m$. Recall that $\Delta(R)$ is aspect ratio of R introduced in Definition 3.2.14.

Corollary 3.7.9. *Let R be a finite set in a metric space (X, d). Given a compressed cover tree $\mathcal{T}(R)$, Algorithm 3.7.2 finds all k nearest neighbors of q in time*

$$O\Big((c_m(R))^{10} \cdot \log_2(k) \cdot \log_2(\Delta(R)) + |\bar{B}(q, 5d_k(q, R))| \cdot \log_2(k)\Big).$$

Proof. Replace $|L(q, \mathcal{T}(R))|$ in the time complexity of Theorem 3.7.8 by its upper bound from Lemma 3.2.15: $|L(q, \mathcal{T}(R))| \leq |H(\mathcal{T}(R))| \leq \log_2(\Delta(R))$. □

If we are allowed to use the standard expansion constant, that corresponds to KR-dimension of [54], then we obtain a stronger result, Theorem 3.7.14.

Lemma 3.7.10. *Let R be a finite reference set in a metric space (X, d) and let $q \in X$ be a query point. Let ϱ be the special level of $L(\mathcal{T}(R), q)$. Let $i \in L(\mathcal{T}(R), q) \setminus \varrho$ be any level. Then if $p \in R_i$ we have $d(p, q) \leq 2^{i+3}$.*

Proof. By assumption in this part of the algorithm we have $d(q, \lambda) \leq 2^{i+2}$. By line 8 of Algorithm 3.7.2, since $p \in R_i$ we have $d(p, q) \leq d(q, \lambda) + 2^{i+2} \leq 2^{i+2} + 2^{i+2} \leq 2^{i+3}$, which proves the claim. □

Lemma 3.7.11. *Let R be a finite reference set in a metric space (X, d) and let $q \in X$ be a query point. Let ϱ be the special level of $L(\mathcal{T}(R), q)$. Let $i \in L(\mathcal{T}(R), q) \setminus \varrho$ be any level. Then if $p \in \mathcal{C}_i(R_{\eta(i)}) \setminus R_i$, we have $d(p, q) > 2^{i+2}$.*

Proof. By assumption $p \in \mathcal{C}_i(R_{\eta(i)}) \setminus R_i$. By line 8 of Algorithm 3.7.2 it follows that $d(q, p) > 2^{i+2} + d(q, \lambda) \geq 2^{i+2}$. Therefore $d(q, p) > 2^{i+2}$, which proves the claim. □

Lemma 3.7.12. *Let i be a non-minimal level of $L(\mathcal{T}(R), q)$ of Definition 3.7.1. Assume that $t = \eta(\eta(i+3))$ is defined. Then there exists $p \in R$ satisfying $2^{i+2} < d(p, q) \leq 2^{t+4}$.*

Proof. Note first that since $\eta(i+3) \in L(\mathcal{T}(R), q)$, there exists distinct $u \in R_{\eta(\eta(i+3))}$ and $v \in \mathcal{C}_{\eta(i+3)}(R_{\eta(\eta(i+3))})$, in such a way that u is the parent of v. Let us show that both of u, v cant belong to set R_i. Assume contrary that both $u, v \in R_i$. Then by Lemma 3.7.10 we have $d(v, q) \leq 2^{i+3}$ and $d(u, q) \leq 2^{i+3}$. By triangle inequality $d(u, v) \leq d(u, q) + d(q, v) \leq 2^{i+4} \leq 2^{\eta(i+3)}$. Recall that we denote a level of a node by l. On the other hand we have $l(u) \geq \eta(i+3)$ and $l(v) \geq \eta(i+3)$, by separation condition of Definition 3.2.1 we have $d(u, v) > 2^{\eta(i+3)}$, which is a contradiction. Therefore only one of $\{u, v\}$ can belong to R_i. It sufficies two consider the two cases below:

Assume that $v \notin R_i$. Since v is children of u we have $d(u, v) \leq 2^{\eta(i+3)+1}$. By Lemma 3.7.10 we have $d(u, q) \leq 2^{\eta(\eta(i+3))+3}$. By triangle inequality

$$d(v, q) \leq d(v, u) + d(u, q) \leq 2^{\eta(\eta(i+3))+3} + 2^{\eta(i+3)+1} \leq 2^{\eta(\eta(i+3))+4}$$

Since $v \notin R_i$ there exists level t having $\eta(i+3) \geq t \geq i$ and $v \in \mathcal{C}_t(R_{\eta(t)}) \setminus R_t$. Therefore by Lemma 3.7.11 we have $d(q, v) > 2^{t+2} \geq 2^{i+2}$. It follows that we have found point $v \in R$ satisfying $2^{i+2} < v \leq 2^{\eta(\eta(i+3))+4}$. Therefore $p = v$, is the desired point.

Assume that $u \notin R_i$. Since $u \in R_{\eta(\eta(i+3))}$, by Lemma 3.7.10 we have $d(u, q) \leq 2^{\eta(\eta(i+3))+3}$. On the other hand since $u \notin R_i$, there exists level t having $\eta(i+3) \geq t \geq i$ and $u \in \mathcal{C}_t(R_{\eta(t)}) \setminus R_t$. Therefore by Lemma 3.7.11 we have $d(q, u) > 2^{t+2} \geq 2^{i+2}$. It follows that we have found point $u \in R$ satisfying $2^{i+2} < u \leq 2^{\eta(\eta(i+3))+4}$. Therefore $p = u$, is the desired point.

$\square$

Lemma 3.7.13. *Algorithm 3.7.2 executes lines 4-18 the following number of times: $|L(\mathcal{T}(R), q)| = O(c(R \cup \{q\})^2 \cdot \log_2(|R|))$.* ∎

Proof of Lemma *3.7.13.* Let $x \in L(\mathcal{T}(R), q)$ be the lowest level of $L(\mathcal{T}(R), q)$. Define $s_1 = \eta(\eta(x) + 1)$ and let $s_i = \eta(\eta(\eta(s_{i-1} + 3)) + 3)$, if it exists. Assume that s_{n+1} is the last sequence element for which $\eta(\eta(\eta(s_{n-1} + 3)) + 3)$ is defined. Define $S = \{s_1, ..., s_n\}$. For every $i \in \{1, ..., n\}$ let p_i be the point provided by Lemma 3.7.12 that satisfies

$$2^{s_i+2} < d(p_i, q) \leq 2^{\eta(\eta(s_i+3))+4}.$$

Let P be the sequence of points p_i. Denote $n = |P| = |S|$. Let us show that S satisfies the conditions

of Lemma 3.2.11. Note that:

$$4 \cdot d(p_i, q) \leq 4 \cdot 2^{\eta(\eta(s_i+3))+4} \leq 2^{\eta(\eta(s_i+3))+6} \leq 2^{\eta(\eta(\eta(s_i+3))+3)+2} \leq 2^{s_{i+1}+2} \leq d(p_{i+1}, q)$$

By Lemma 3.2.11 applied for $A = R \cup q$ and sequence P we get:

$$|\bar{B}(q, \frac{4}{3}d(q, p_n))| \geq (1 + \frac{1}{c(R)^2})^n \cdot |\bar{B}(q, \frac{1}{3}d(q, p_1))|$$

Since $\eta(x) \in L(\mathcal{T}(R), q)$, there exists some point $u \in R_{\eta(x)}$. By Lemma 3.7.10 we have $d(u, q) \leq 2^{\eta(x)+3}$. Also $2^{\eta(\eta(x)+1)+1} \leq \frac{2^{\eta(\eta(x)+1)+2}}{3} < \frac{d(q, p_1)}{3}$ It follows that:

$$1 \leq |\bar{B}(q, 2^{\eta(x)+3})| \leq |\bar{B}(q, 2^{\eta(\eta(x)+1)} + 1)| \leq |\bar{B}(q, \frac{d(q, p_1)}{3})|$$

Therefore we have
$$|R| \geq \frac{|\bar{B}(q, \frac{4}{3} \cdot d(q, p_n))|}{|\bar{B}(q, \frac{1}{3} \cdot d(q, p_1))|} \geq (1 + \frac{1}{c(R \cup \{q\})^2})^n$$

Note that $c(R \cup \{q\}) \geq 2$ by definition of expansion constant. Then by applying log and by using Lemma 3.2.12 we obtain: $c(R \cup \{q\})^2 \log(|R|) \geq n = |S|$. Let x be minimal level of $L(\mathcal{T}(R), q)$ and let y be the maximal level of $L(\mathcal{T}(R), q)$ Note that S is a sub sequence of L in such a way that:

- $[x, s_1] \cap L(\mathcal{T}(R), q) \leq 3$,

- for all $i \in 1, ..., n$ we have $[s_i, s_{i+1}] \cap L(\mathcal{T}(R), q) \leq 10$

- $[s_n, y] \cap L(\mathcal{T}(R), q) < 20$

Since segments $[x, s_1], [s_1, s_2], ..., [s_2, s_n], [s_n, y]$ cover $|L(\mathcal{T}(R), q)|$, it follows that $|S| \geq \frac{|L(\mathcal{T}(R), q)|}{20}$. We obtain that

$$|L(\mathcal{T}(R), q)| \leq 20 \cdot c(R \cup \{q\})^2 \cdot \log_2(|R|),$$

which proves the claim. $\square$

Theorem 3.7.14. *Let R be a finite reference set in a metric space (X, d). Let $q \in X$ be a query point, $c(R \cup \{q\})$ be the expansion constant of $R \cup \{q\}$ and $c_m(R)$ be the minimized expansion constant from Definition 3.1.4. Given a compressed cover tree $\mathcal{T}(R)$, Algorithm 3.7.2 finds all k nearest neighbors of q in time*

$$O\Big(c(R \cup \{q\})^2 \cdot \log_2(k) \cdot \big((c_m(R))^{10} \cdot \log_2(|R|) + c(R \cup \{q\}) \cdot k\big)\Big).$$

Proof. By Theorem 3.7.8 the required time complexity is

$$O\Big((c_m(R))^{10} \cdot \log_2(k) \cdot |L(q, \mathcal{T}(R))| + |\bar{B}(q, 5d(q, \beta))| \cdot \log_2(k)\Big)$$

for some point β among the first k-nearest neighbors of q. Apply Definition 3.1.4:

$$|B(q, 5d(q, \beta))| \le (c(R \cup \{q\}))^3 \cdot |B(q, \frac{5}{8} d(q, \beta))| \qquad (3.7.14.1)$$

Since $|B(q, \frac{5}{8} d(q, \beta))| \le k$, we have $|B(q, 5d(q, \beta))| \le (c(R \cup \{q\}))^3 \cdot k$. It remains to apply Lemma 3.7.13: $|L(q, \mathcal{T}(R))| = O(c(R \cup \{q\})^2 \cdot \log_2(|R|))$. $\qquad \square$

Corollary 3.7.15 combines Theorem 3.5.10 with Theorem 3.7.14, to show that Problem 3.1.2 can be resolved in $O(c^{O(1)} \cdot \log(k) \cdot \max\{|Q|, |R|\} \cdot (\log(|R|)) + k)$ time.

Corollary 3.7.15 (solution to Problem 3.1.2)**.** *In the notations of Theorem 3.7.14, set* $c = \max\limits_{q \in Q} c(R \cup \{q\})$. *Algorithms 3.5.3 and 3.7.2 solve Problem 3.1.2 in time*

$$O\Big(\max(|Q|, |R|) \cdot c^2 \cdot \log_2(k) \cdot \big((c_m(R))^{10} \cdot \log_2(|R|) + c \cdot k\big)\Big).$$

Proof. For any $q \in Q$, since $\log_2(R \cup \{q\}) \le 2 \log_2(R)$, a tree $\mathcal{T}(R)$ can be built in time

$$O(c^2 \cdot c_m(R)^8 \cdot \log(|R|))$$

by Theorem 3.5.10. Therefore the time complexity is dominated by running Algorithm 3.7.2 on all points $q \in Q$. The final complexity is obtained by multiplying the time from Theorem 3.7.14 by $|Q|$. $\qquad \square$

3.8   Approximate k-nearest neighbor search

The original navigating nets and cover trees were used in [54, Theorem 2.2] and [9, Section 3.2] to solve the $(1 + \epsilon)$-approximate nearest neighbor problem for $k = 1$. The main result, Theorem 3.8.6 justifies a near linear parameterized complexity to find approximate a k-nearest neighbor set $\mathcal{P}$ formalized in Definition 3.8.1.

Definition 3.8.1 (approximate k-nearest neighbor set $\mathcal{P}$)**.** Let R be a finite reference set and let Q be a finite query set of a metric space (X, d). Let $q \in Q \subseteq X$ be a query point, $k \ge 1$ be an

integer and $\epsilon > 0$ be a real number. Let $\mathcal{N}_k = \cup_{i=1}^k \mathrm{NN}_i(q)$ be the union of neighbor sets from Definition 3.1.1. A set $\mathcal{P} \subseteq R$ is called an *approximate k-nearest neighbors set*, if $|\mathcal{P}| = k$ and there is an injection $f : \mathcal{P} \to \mathcal{N}_k$ satisfying $d(q,p) \leq (1+\epsilon) \cdot d(q, f(p))$ for all $p \in \mathcal{P}$. ∎

Algorithm 3.8.2 This algorithm finds approximate k-nearest neighbor of Definition 3.8.1.

1: **Input** : compressed cover tree $\mathcal{T}(R)$, a query point $q \in X$, an integer $k \in \mathbb{Z}_+$, real $\epsilon \in \mathbb{R}_+$.
2: Set $i \leftarrow l_{\max}(\mathcal{T}(R)) - 1$ and $\eta(l_{\max} - 1) = l_{\max}$. Set $R_{l_{\max}} = \{\mathrm{root}(\mathcal{T}(R))\}$.
3: **while** $i \geq l_{\min}$ **do**
4:     Assign $\mathcal{C}_i(R_{\eta(i)}) \leftarrow R_{\eta(i)} \cup \{a \in \mathrm{Children}(p) \text{ for some } p \in R_{\eta(i)} \mid l(a) = i\}$.
5:     Compute $\lambda = \lambda_k(q, \mathcal{C}_i(R_{\eta(i)}))$ from Definition 3.6.7 by Algorithm 3.6.9.
6:     Find $R_i = \{p \in \mathcal{C}_i(R_{\eta(i)}) \mid d(q,p) \leq d(q,\lambda) + 2^{i+2}\}$.
7:     **if** $\frac{2^{i+2}}{\epsilon} + 2^{i+1} \leq d(q,\lambda)$ **then**
8:         Let $\mathcal{P} = \emptyset$.
9:         **for** $p \in \mathcal{C}_i(R_{\eta(i)})$ **do**
10:           **if** $d(p,q) < d(q,\lambda)$ **then**
11:             $\mathcal{P} = \mathcal{P} \cup \mathcal{S}_i(p, \mathcal{T}(R))$
12:           **end if**
13:         **end for**
14:         Fill $\mathcal{P}$ until it has k points by adding points from sets $\mathcal{S}_i(p, \mathcal{T}(R))$, where $d(p,q) = d(q,\lambda)$.
15:         **return** $\mathcal{P}$.
16:     **end if**
17:     Set $j \leftarrow \max_{a \in R_i} \mathrm{Next}(a, i, \mathcal{T}(R))$. {If such j is undefined, we set $j = l_{\min} - 1$}
18:     Set $\eta(j) \leftarrow i$ and $i \leftarrow j$.
19: **end while**
20: Compute and **output** k-nearest neighbors of query point q from the set $R_{l_{\min}}$.

Definition 3.8.3 is analog of Definition 3.7.1 for $(1+\epsilon)$-approximate k-nearest neighbor search.

Definition 3.8.3 (Iteration set of approximate k-nearest neighborhood search)**.** Let R be a finite subset of a metric space (X,d). Let $\mathcal{T}(R)$ be a cover tree of Definition 3.2.1 built on R and let $q \in X$ be an arbitrary point. Let $L(\mathcal{T}(R), q) \subseteq H(\mathcal{T}(R))$ be the set of all levels i during iterations of lines 3-19 of Algorithm 3.8.2 launched with inputs $(\mathcal{T}(R), q)$. We denote $\eta(i) = \min_t \{t \in L(\mathcal{T}(R), q) \mid t > i\}$. ∎

Lemma 3.8.4 (k-nearest neighbors in the candidate set for all i)**.** *Let R be a finite subset of an ambient metric space (X,d), let $q \in X$ be a query point , let $k \in \mathbb{Z} \cap [1, \infty)$ and $\epsilon \in \mathbb{R}_+$ be parameters. Let $\mathcal{T}(R)$ be a compressed cover tree of R. Assume that $|R| \geq k$. Then for any iteration $i \in L(\mathcal{T}(R), q)$ of Algorithm 3.8.2 the candidate set $\bigcup_{p \in R_i} \mathcal{S}_i(p, \mathcal{T}(R))$ contains all k-nearest neighbors of q.* ∎

Proof. Proof of this lemma is similar to Lemma 3.8.4 and is therefore omitted.   □

Lemma 3.8.5 shows that Algorithm 3.8.2 correctly returns an Approximate k-nearest neighbor set of Definition 3.8.1.

Lemma 3.8.5 (Correctness of Algorithm 3.8.2). *Algorithm 3.8.2 finds an approximate k-nearest neighbors set of any query point $q \in X$.*   ■

Proof. Assume first that condition on line 7 of Algorithm 3.8.2 is satisfied during some iteration $i \in H(\mathcal{T}(R))$ of Algorithm 3.8.2. Let us denote

$$\mathcal{A} = \bigcup_{p \in \mathcal{C}_i(R_{\eta(i)})} \{\mathcal{S}_i(p, \mathcal{T}(R)) \mid d(p,q) < d(q,\lambda)\}, \mathcal{B} = \bigcup_{p \in \mathcal{C}_i(R_{\eta(i)})} \{\mathcal{S}_i(p, \mathcal{T}(R)) \mid d(p,q) = d(q,\lambda)\}.$$

By Algorithm 3.8.2 set $\mathcal{P}$ contains all points of $\mathcal{A}$ and rest of the points are filled form $\mathcal{B}$. We will now form $f : \mathcal{P} \to \mathcal{N}_k$ by mapping every point $p \in \mathcal{A} \cap \mathcal{P}$ into itself and then by extending f to be injective map on whole set $\mathcal{P}$. The claim holds trivially for all points $p \in \mathcal{A} \cap \mathcal{P}$. Let us now consider points $p \in \mathcal{P} \setminus \mathcal{A}$. Let $\gamma \in \mathcal{C}_i(R_{\eta(i)})$ be such that $p \in \mathcal{S}_i(\gamma, \mathcal{T}(R))$ and let $\psi \in \mathcal{C}_i(R_{\eta(i)})$ be such that $f(p) \in \mathcal{S}_i(\psi, \mathcal{T}(R))$. By using triangle inequality, Lemma 3.2.7 and the fact that $p \in \mathcal{A} \cup \mathcal{B}$ we obtain:

$$d(q,p) \le d(q,\gamma) + d(\gamma,p) \le d(q,\lambda) + 2^{i+1} \tag{3.8.5.2}$$

On the other hand since $f(p) \notin \mathcal{A}$ we have

$$(1+\epsilon) \cdot d(q, f(p)) \ge (1+\epsilon) \cdot (d(q,\psi) - d(\psi, f(p))) \ge (1+\epsilon) \cdot (d(q,\lambda) - 2^{i+1}) \tag{3.8.5.3}$$

Note that by line 7 we have $\frac{2^{i+2}}{\epsilon} + 2^{i+1} \le d(q,\lambda)$. It follows that $2^{i+2} \le \epsilon \cdot d(q,\lambda) - \epsilon \cdot 2^{i+1}$. Therefore we have:

$$d(q,\lambda) + 2^{i+1} \le d(q,\lambda) + 2^{i+2} - 2^{i+1} \le (1+\epsilon) \cdot (d(q,\lambda) - 2^{i+1}) \tag{3.8.5.4}$$

By combining Equations (3.8.5.2) - (3.8.5.4) we obtain $d(q,p) \le (1+\epsilon) \cdot d(q, f(p))$. If the condition on line 7 of Algorithm 3.8.2 is never satisfied, then the Algorithm finds real k-nearest neighbors of point q in the end of the algorithm and therefore the claim holds.

                                                                                                  □

Theorem 3.8.6 (correctness of modified Algorithm 3.7.2). *In the notations of Definition 3.8.1, the complexity of Algorithm 3.8.2 is $O\Big((c_m(R))^{8 + \lceil \log(2 + \frac{1}{\epsilon}) \rceil} \cdot \log_2(k) \cdot \log_2(\Delta(R)) + k\Big)$.*   ■

Proof. Similarly to Lemma 3.7.7 it can be shown that Algorithm 3.8.2 is bounded by:

$$O((c_m(R))^4 \cdot \log_2(k) \cdot \max_i |R_i| \cdot |H(\mathcal{T}(R))| + \#\text{Line}[7-16]) \qquad (3.8.6.1)$$

Note first that in lines 7 - 16 we loop over set $\mathcal{C}_i(R_{\eta(i)})$ and select k points from it. Therefore $\#\text{Line}[7-16] = k + |\mathcal{C}_i(R_{\eta(i)})|$.

Let us now bound the size of R_i. By line 7 of Algorithm 3.8.2 either Algorithm 3.8.2 is launched that terminates the program or $\frac{2^{i+2}}{\epsilon} + 2^{i+1} > d(q, \lambda)$. Let C_i be the ith cover set of $\mathcal{T}(R)$. To bound $|R_i|$ we can assume the latter. Similarly to Theorem 3.7.14 we have:

$$R_i = \{r \in \mathcal{C}_i(R_{\eta(i)}) \mid d(p, q) \le d(q, \lambda) + 2^{i+2}\} \qquad (3.8.6.2)$$

$$= \bar{B}(q, d(q, \lambda) + 2^{i+2}) \cap \mathcal{C}_i(R_{\eta(i)}) \qquad (3.8.6.3)$$

$$\subseteq \bar{B}(q, d(q, \lambda) + 2^{i+2}) \cap C_i \qquad (3.8.6.4)$$

$$\subseteq \bar{B}(q, 2^{i+2}(\frac{3}{2} + \frac{1}{\epsilon})) \cap C_i \qquad (3.8.6.5)$$

Since the cover set C_i is a 2^i-sparse subset of the ambient metric space X, we can apply Lemma 3.2.8 with $t = 2^{i+2}(\frac{3}{2} + \frac{1}{\epsilon})$ and $\delta = 2^i$. Since $4\frac{t}{\delta} + 1 = 2^4(\frac{3}{2} + \frac{1}{\epsilon}) + 1 \le 2^4(2 + \frac{1}{\epsilon})$, we get $\max|R_i| \le (c_m(R))^{4 + \lceil \log_2(2 + \frac{1}{\epsilon}) \rceil}$. The final complexity is obtained by plugging the upper bound of $|R_i|$ above into (3.8.6.1). $\qquad \square$

Corollary 3.8.7 (complexity for approximate k-nearest neighbors set $\mathcal{P}$)**.** *In the notations of Definition 3.8.1, an approximate k-nearest neighbors set is found for all $q \in Q$ in time* $O\Big(|Q| \cdot (c_m(R))^{8 + \lceil \log(2 + \frac{1}{\epsilon}) \rceil} \cdot \log(k) \cdot \log_2(\Delta(R)) + |Q| \cdot k\Big)$. $\qquad \blacksquare$

Proof. This corollary follows directly from Theorem 3.8.6 . $\qquad \square$

3.9 Counterexamples for NIPS 2009 dual-tree complexity analysis

This section contains counterexamples for [78]. In 2009 [78, 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 [78, Definition 3.1]. We have found the following issues.

First, Counterexample 3.9.2 shows that [78, Algorithm 1] when $Q = R$ returns the same point for all $q \in Q$, which shows that [78, Algorithm 1] cannot be used to find non-trivial neighbors in this case. Second, Remark 3.9.4 explains several possible interpretations of [78, Definition 3.1] for the parameter κ. Third, [78, Theorem 3.1] similarly to [9, Theorem 5] relied on the same estimate of recursions in the proof of [9, Lemma 4.3]. Counterexample 3.9.5 explains step-by-step why the proof of the time complexity result of [78, Algorithm 1] is incorrect and requires a clearer definition of κ.

In 2015 Curtin [26] made an attempt to fix the issues of [78] by introducing other parameters: the imbalance I_t in [26, Definition 3] and θ in [26, 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. [26, Theorem 2] shows that, for constructed cover trees on a query set Q and a reference set R, Problem 3.1.2 for $k = 1$ (only 1st nearest neighbors) can be solved in time

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

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.

Algorithm 3.9.1 Original [78, Algorithm 1] is analyzed in Counterexamples 3.9.2 and 3.9.5.

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:     $\mathcal{C}(R_i) = \{\text{Children}(r) \mid r \in R_i\}$ {in original pseudo-code the notation is $R = \mathcal{C}(R_i)$}
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**

The step-by-step execution of Algorithm 3.9.1 will show that the number of reference expansions has a lower bound $O(\max\{|Q|, |R|\}^2)$. Recall that [78, End of Section 1] defined the all-nearest-neighbor problem as follows. "**All Nearest-neighbors:** For all queries $q \in Q$ 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 3.9.1, the node q_j has a level j, a reference subset $R_i \subset R$ is a subset of C_i for an implicit cover tree $T(R)$. The algorithm is called for a pair $q_j, R_i = \{r\}$, where q_j is the root of the query tree at the maximal level $j = l_{\max}(T(Q))$, and r is the root of the reference tree at the maximal level $i = l_{\max}(T(R))$. Split Algorithm 3.9.1 into these blocks:

lines 2-4 : FinalCandidates,

lines 5-9 : reference expansion,

lines 9-11 : query expansion.

Counterexample 3.9.2. In the notations of Example 3.4.1, m is a parameter of R. Build a compressed cover tree $\mathcal{T}(R)$ as in Figure 3.8. Set $Q = R$. First we show that Algorithm 3.9.1 returns the trivial neighbor for every point for $\mathcal{T}(Q) = \mathcal{T}(R)$.

We start the simulation with the query node r on the level $m^2 + 1$, which has the reference subset $R_{m^2+1} = \{r\}$. 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 has two children at the level m^2, so $\mathcal{C}(R_i) = \{p_{m^2}, r\}$. Since $d(p_{m^2}, p_{m^2}) = 0$ and $d(p_{m^2}, r) = 2^{m^2+1}$ on line 7, we have:

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

Similarly, for $i = m^2, j = m^2 - 1, q_j = p_{m^2}$, we have $\mathcal{C}(R_i) = \{p_{m^2}, p_{m^2-1}, r\}$ 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:

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

For $i = m^2 - 1, j = m^2 - 2, q_j = p_{m^2}$, we have $\mathcal{C}(R_i) = \{p_{m^2}, p_{m^2-1}, p_{m^2-2}\}$. 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:

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

Finally, for $i = m^2 - 2, j = m^2 - 3, q_j = p_{m^2}$, we have $\mathcal{C}(R_i) = \{p_{m^2}, p_{m^2-1}, p_{m^2-2}, p_{m^3-3}\}$ 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

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

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 3.9.1 finds trivial nearest neighbor for every point $p_t \in R$.   ■

Example 3.9.3. To avoid the issue of finding trivial nearest neighbors in Counterexample 3.9.2, we will modify Example 3.4.1. 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 $\mathcal{T}(Q), \mathcal{T}(R)$ have the same structure as the compressed cover tree $\mathcal{T}(R)$ in Example 3.4.1.   ■

Remark 3.9.4. [78, 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.

[26, 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 3.9.2.

Counterexample 3.9.5 (Counterexample to [78, Theorem 3.1])**.** We will now show that in addition to the problems in the pseudocode the proof of [78, 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.*

[Part of the proof] Since at any level of recursion, the size of R [Corresponding to $\mathcal{C}(R_i)$ in Algorithm 3.9.1] 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 3.9.1] 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:

$$\#(\text{Line 8}) \le \max_{p \in R} D(p) \cdot \max_i \mathcal{C}(R_i) \cdot (\text{number of duplications}). \qquad (3.9.5.6)$$

It will be shown that cover tree $\mathcal{T}(R)$ from Example 3.9.3 does not satisfy the inequality above.

Let $X, \mathcal{T}(R), \mathcal{T}(Q), R, Q$ be as in Example 3.9.3 for some parameter m. We will consider the simulation of Algorithm 3.9.1 on pair $(\mathcal{T}(Q), \mathcal{T}(R))$. We note first Lemma 3.4.3 applied on $\mathcal{T}(R)$ provides $\max_{p \in R} D(p) \le 2m + 1$ Similarly, as in Counterexample 3.4.7, a contradiction will be achieved by showing that R_i and a set of its children $\mathcal{C}(R_i)$ will have a constant size bound on any recursion (i, j) of Algorithm 3.9.1.

Since $\mathcal{T}(R)$ contains at most one children on every level i we have $|\mathcal{C}(R_i)| \le |R_i| + 1$ 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 $t \ge i + 1$ 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 $t \ge i + 1$ 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 3.9.1 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 $r_{2m+1} = r$.

Assume now that the claim holds 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 $\mathcal{C}(R_i) = \{r, r_{i+1}, r_i, r_{i-1}\}$. 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

$$R_{i-1} = \{r' \in \mathcal{C}(R_i) \mid d(r, r') \leq 2^i + 2^{i+1}\} = \{r, r_i, r_{i-1}\}.$$

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 $R_{i-1} = \{r, r_i, r_{i-1}\}$. 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 3.9.1 ($q, R_i = \{r\}$) 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 3.9.1 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 $\mathcal{T}(Q)$ has exactly one new child on every level and that trees $\mathcal{T}(Q)$ and $\mathcal{T}(R)$ 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 (3.9.5.6), we obtain the following contradiction:

$$O(m^4) = \xi \le \max_{p \in R} D(p) \cdot (\max_i \mathcal{C}(R_i)) \cdot (\text{number of duplications}) \le (2m+1) \cdot 4 \cdot m^2 \le O(m^3).$$

3.10 Discussions: current contributions and future steps

This chapter rigorously proved the time complexity of the exact k-nearest neighbor search. The motivations were the past challenges in the proofs of time complexities in [54, Theorem 2.7], [9, Theorem 5], [78, Theorem 3.1], [63, Theorem 5.1]. Though [27, Section 5.3] pointed out some difficulties, no corrections were published. Main Theorem 3.7.14 and Corollary 3.7.15 fill the above gaps.

First, Definition 3.1.1 and Problem 3.1.2 rigorously deal with a potential ambiguity of k-nearest neighbors at equal distances, which wasn't discussed in the past work. The main new data structure of a compressed cover tree in Definition 3.2.1 substantially simplifies the navigating nets [54] and original cover trees [9] by avoiding any repetitions of given data points. This compression has substantially clarified the construction and search Algorithms 3.5.3 and 3.7.2.

Second, In Section 3.3 we showed that the newly defined minimized expansion constant c_m of any finite subset R of normed vector space on $\mathbb{R}^n$ has upper-bound 2^m. In the future, it can be similarly shown that if R is uniformly distributed then classical expansion constant $c(R)$ is 2^m as well.

Third, the claims of original work [9] regarding near-linear time complexities of cover tree construction algorithm, as well as nearest neighbor search had incorrect proofs. Counterexamples 3.4.5 and 3.4.7. provide concrete examples of datasets R that break claimed proofs. Additionally, in 2009 [78, Theorem 3.1], there was an attempt to show that after the construction of all the relevant data structures, such as the implicit cover tree, Problem 3.1.2 can be solved in a linear time $c(R)^{O(1)} \cdot |R|$. In Section 3.9 Counterexample 3.9.5 shows that the number of total reference recursions was estimated similarly wrong as in the proof of [9, Theorem 5].

Fourth, the approach of [9] is corrected in Section 3.5 and 3.5.11. Assuming that expansion constants and aspect ratio of a reference set R are fixed, Corollary 3.5.11 and Corollary 3.7.15 rigorously show that the time complexities are linear in the maximum size of R and a query set Q and near-linear $O(k \log k)$ in the number k of neighbors.

Finally, In future, we wish to improve the time complexity result of k-nearest neighbor search to a purely linear-time $O(c(R)^{O(1)} \cdot |R|$ by constructing cover trees on both, query and reference sets.

Since a similar approach of [78] was shown to have incorrect proof and [26] contain additional parameters I, θ, it requires significantly more effort to understand if $O(c(R)^{O(1)} \cdot |R|)$ is archivable by means of a compressed cover tree data structure.

Chapter 4

Fast algorithm for minimum spanning tree based on compressed cover tree.

4.1 Minimum spanning tree

Recall that a tree T is any connected graph without cycles. A minimum spanning tree on a finite metric space (R, d) is a tree $\text{MST}(R)$ with vertex R that has a minimal total length of edges. A minimum spanning tree, $\text{MST}(R)$ and is a fundamental object of computational geometry that finds its applications in various data structures, such as topomap [28] and accelerated hierarchical density based clustering [65], mergegram [33]. This chapter corrects the work of [63] and introduces a new algorithm that finds $\text{MST}(R)$ in a parametrized near-linear time in the size of $|R|$.

Definition 4.1.1 (Minimum spanning tree of a metric space)**.** Let (R, d) be a finite metric space. We say that a graph T is a spanning tree of X, if T has the points of X as its vertex set and if T is connected. A graph T is a minimum spanning tree of graph G, if the total length of all the edges $w(T)$ in the graph T is minimal among all the spanning trees of X. A minimum spanning tree of (R, d) will be denoted as $\text{MST}(R)$. ∎

The *exact* metric spanning tree problem asks for exact $\text{MST}(R)$ [63, 18]. For any graph G let us denote wt to be the sum of all edge lengths in G. In *approximate* the goal is to find a spanning tree T on R, in such a way that $wt(T) \leq (1 + \epsilon) \cdot wt(\text{MST}(R))$, where $\epsilon > 0$ is fixed and $\text{MST}(R)$ is the exact minimum spanning tree [3, 89]. The most recent approaches [90] focus on developing parallel algorithms for the construction of $\text{MST}(R)$.

Problem 4.1.2. Given a finite subset R of a metric space (X, d) find a near-linear in the size $|R|$ algorithm to compute minimum spanning tree MST(R).

Note that MST(R) is not generally uniquely defined. However, if R is in general position i.e. all pairwise distances of (R, d) have distinct values, then MST(R) is unique.

Problem 4.1.3. Given a finite subset R of a metric space (X, d) and a real number $\epsilon > 0$ find a near-linear in the size $|R|$ algorithm to compute $(1 + \epsilon)$-approximate minimum spanning tree MST(R) i.e. find spanning tree T on R in such a way that $wt(T) \leq (1 + \epsilon) \cdot wt(\text{MST}(R))$.

In the naive approach for finding a minimum spanning tree, first, a complete graph G is built on R, in such a way that the edge set of G contains all combinations of $R \times R$. Then any classical method, such as Borůvka's algorithm [11] or Prim's algorithm [75] to compute MST(R). Since there are $O(|R|^2)$ edges, the total run-time of the algorithm $|R|^2 \cdot \log(|R|)$. The obtained time worst-case time complexity estimate is too slow for most practical applications. In recent years more sophisticated techniques have been developed.

One of the main aims of this Chapter is to show that the original approach [63] based on dual-tree strategy and cover tree data structure failed to give a worst-case time complexity estimate for their proposed algorithm [63, Algorithm]. In [63, Theorem 5.1] it was claimed that Problem 4.1.2 can be resolved in

$$O(\max\{c(R)^6, c_p^2 c_l^2\} \cdot c(R)^{10} \cdot |R| \cdot \log(|R|) \cdot \alpha(|R|)$$

time, where $c(R)$ is expansion constant of Definition 3.1.4, $\alpha(|R|)$ is slowly growing Ackermann function and new expansion constants c_p, c_l see Remark 4.3.3. However, the provided proof of Theorem 5.1 [63] was incorrect. Counterexample 4.2.3 gives an example, which produces $O(|R| \cdot \sqrt{|R|})$ iterations in its first Borůvka step of [63, Algorithm]. However, the proof claimed that the number of iterations is bounded by depth multiplied by the number of duplications. The contradiction will be obtained by showing that the given proof implies that the number of iterations should be bounded by $40 \cdot \sqrt{|R|}$, which contradicts previously found $O(|R| \cdot \sqrt{|R|})$ low-bound for the number of iterations. Section 4.3 will highlight further challenges in the definition of the linkage expansion constant c_l, which is used in another part of the proof of Theorem 5.1.

Shortcomings of [63] are resolved in Section 4.4. The main method of the section is Algorithm 4.4.4, which is an analog of the minimum spanning tree algorithm [63, Algorithm 1] to
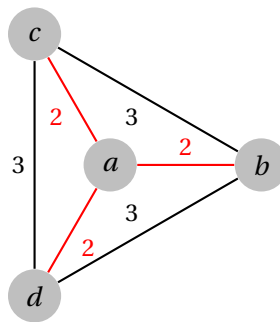
Figure 4.1: MST drawn with red color on a data set arranged triangularly.

compressed cover trees. In Theorem 4.4.9 we prove correctness of Algorithm 4.4.4. In Theorem 4.4.14 it is shown that the time complexity of Algorithm 4.4.4 is:

$$O\Big((c_m(R))^{4+\lceil \log_2(\rho(R))\rceil} \cdot |H(\mathcal{T}(R))| \cdot |R| \cdot \log(|R|) \cdot \alpha(|R|)\Big), \tag{4.1.3.1}$$

where $\rho(R)$ is introduced in Definition 4.4.11 and $H(\mathcal{T}(R))$ is the height of Definition 3.2.13.

4.2   Counterexample: An incorrect estimate of reference recursions

In this section, we assume that (R,d) is a finite metric space. In this section we show that [63, Theorem 5.1] was proven incorrectly. Counterexample 4.2.3 gives a specific example, which contradicts a step in a proof of [63, Theorem 5.1], which similarly to [78, Theorem 3.1] fails to estimate correctly the number of recursions during the whole execution of Algorithm 4.2.2.

In original Algorithm 4.2.2, the node q_j has a level j, a reference subset $R_i \subseteq R$ is a subset of C_i for an implicit cover tree $T(R)$. The algorithm is called for a pair $r, R_i = \{r\}$, where r is the root of the reference tree at the maximal level $j = l_{\max}(T(R))$. Split Algorithm 4.2.2 into these blocks: lines 2-8 : FinalCandidates, lines 8-13: reference expansion, and lines 13-27: query expansion.

Note that DualTreeBorůvka algorithm of [63, Algorithm 1] designed on implicit cover trees corresponds Algorithm 4.2.1 that uses compressed cover tree. Algorithm 4.2.2 ([63, Algorithm 3]) is launched from Line 4 of Algorithm 4.2.1.

Counterexample 4.2.3 shows that [63, Theorem 5.4] was proven incorrectly by using slightly modified space R from Example 3.4.1.

Algorithm 4.2.1 Copy-pasted [63, Algorithm 1], algorithm for dual-tree MST algorithm. Here E is a set of edges of graph G having the whole reference set R as its vertex set, N is the size of set R, and UpdateTree(q) is a technical method for keeping track of necessary properties of the implicit tree $T(Q)$.

1: **Dual-tree Borůvka**(Implicit cover tree node q)
2: $E = \emptyset$
3: **while** $|E| < N - 1$ **do**
4:     call Algorithm 4.2.2 with parameters (q, q, e).
5:     $E = E \cup e$
6:     UpdateTree(q)
7: **end while**

Algorithm 4.2.2 Copy-pasted [63, Algorithm 3], now deprecated method for finding nearest neighboring components during the iteration of Algorithm 4.2.1.

1: **Function** Fcn($T(Q)$-tree node q_j, Subset R_i of cover set C_i of $T(R)$, Edge set e).
2: **if** $i = -\infty$ **then**
3:     **for** q that are descendants of q_j and $r \in R_i$ with $r \nsim q$ **do**
4:         **if** $d(q, r) < d(C_q)$ **then**
5:             $d(C_q) = d(q, r)$, $e(C_q) = (q, r)$.
6:         **end if**
7:     **end for**
8: **else if** $j < i$ **then**
9:     $\mathcal{C}(R_i) = \{r \in \text{Children}(r') \mid r' \in R_i \text{ and } r \nvDash q_j\}$ {$r \vDash q_j$ means that all the descendants of r and the descendants of q_j belong to the same component}
10:     $d = \min\{d(C_q), \min_{r \in C, r \curvearrowright q}\{d(q, r) + 2^i\}, \min_{r \in \mathcal{C}(R_i), r \nsim q}\{d(q, r)\}\}$
11:     $R_{i-1} = \{r \in R \mid d(q_j, r) \leq d + 2^i + 2^{j+2}\}$
12:     **return** Fcn(q_{j-1}, R_i, e) {q_{j-1} is the same point as q_j on one level below}
13: **else**
14:     **for** $p_{j-1} \in \text{Children}(q_j)$ **do**
15:         **return** Fcn(p_{j-1}, R_i, e)
16:     **end for**
17: **end if**

Counterexample 4.2.3 (Modification of Example 3.4.1 fails proof of [63, Theorem 5.4])**.** In this Counterexample we will point out that [63, Theorem 5.4] was incorrectly proven by using Example 3.4.1. Let us consider the following quote from its proof:

*Page 608: "**Theorem 5.4**. Under the assumptions of Theorem. 5.1, the FindComponentNeighbors algorithm on a cover tree [Algorithm 4.2.1] finds the nearest neighbor of each component in time bounded by:*

$$O(N + c^4 N + \max\{c^6, c_p^2 c_l^2\} \cdot N\alpha(N) + \max\{c^6, c_p^2 c_l^2\} \cdot c^{10} \cdot \log(N) \cdot \alpha(N)).\text{"}$$

Consider the following part from proof of Theorem 5.4:

" At each level, $|R| \leq c^4 \max_i |R_i|$. Since the maximum depth of a node is $O(c^2 \log(N))$ (depth bound), the number of nodes considered in Line 14 is at most $O(c^6 \max_i |R_i| \log(N)$. Considering possible duplication across queries the total number of calls to Line 14 [Our Line 12] is at most $O(c^{10} \max_i |R_i| \log(N))$. "

Our interpretation: Let $D(p)$ be the explicit depth of a point p of Definition 3.4.2. Denote the number of duplications occurring in Algorithm 4.2.2 as χ. The above arguments claimed the algorithm runs Line 12 at most this number of times:

$$\#(\text{Line 12}) \leq \max_{p \in R} D(p) \cdot (\max_i \mathcal{C}(R_i)) \cdot \chi \tag{4.2.3.2}$$

$$\leq c^2 \log(N) \cdot c^4 \max_i |R_i| \cdot c^4 \leq c^{10} \max_i |R_i| \log(N). \tag{4.2.3.3}$$

Using a cover tree $\mathcal{T}(R)$ based on Example 3.4.1 it will be shown that $\mathcal{T}(R)$ does not satisfy (4.2.3.3), which ultimately crashes the proof for this specific dataset.

Let $X, R, \mathcal{T}(R)$ be as in Example 3.4.1 for some $m > 5$. Let us modify X by collapsing edge (q, r) and by identifying q with r. It can be shown that cover tree $\mathcal{T}(R)$ will have exactly same structure as in Example 3.4.1. Let us run Algorithm 4.2.1 on the dataset R and the cover tree $\mathcal{T}(R)$. We will focus on the first iteration of the loop $3 - 7$ for $E = \emptyset$. In this case we launch Algorithm 4.2.2 with parameters $(q_j = r, R_m = r, E = \emptyset)$.

Note first that $\mathcal{T}(R)$ contains at most one children on every level. Since c^4 corresponds to the number of children on level below according to the source, the number of duplications χ will be at most 2. Similarly to Lemma 3.4.3 we obtain $\max_{p \in R} D(p) \leq 2m + 1$. Let us now bound the maximal size of set $\mathcal{C}(R_i)$. For clarity $\text{Fcn}(q_j, R_i, E = \emptyset)$ will be denoted as $\text{Fcn}(j, i, q_j, R_i)$ in this counterexample. Since $\mathcal{T}(R)$ contains at most one children on every level i, we have

$|\mathcal{C}(R_i)| \le |R_i| + 1$ for any recursion of FindComponentNeighbors algorithm. For any $i > m^2$ denote r_i to be r.

Note first that since $l(q_t) = t$ for any $t \in [1, m^2]$, then r_t is recursed into from $\text{Fcn}(t+1, t+1, p, R_i)$, where p is parent node of r_t. Let us fix query node $q = r$ and show by induction that for any recursion $(i, j = i-1, q = r, R_i)$ of FindComponentNeighbors we have $R_i = \{r_{i+2}, r_{i+1}, r_i, r\}$. The base case of the induction holds trivially since $R_{m^2+1} = \{r\}$. Assume now that $R_i = \{r_{i+2}, r_{i+1}, r_i, r\}$ for some i. Then $\mathcal{C}(R_i) = \{r_{i+2}, r_{i+1}, r_i, r, r_{i-1}\}$. Recall that $d(r_t, r) = 2^{t+1}$ for all t. Since there are no nodes in the same component with r, by line 10 we have $d = 2^i$. It follows that $R_{i-1} = \{a \in \mathcal{C}(R_i) \mid d(a, r) \le d + 2^i + 2^{i+1}\} = \{r_{i+1}, r_i, r_{i-1}, r\}$. Since R_i is not modified during query-expansion (lines 13-17) for node r we have proved the induction claim for $\text{Fcn}(i-1, j = i-2, q = r, R_i)$. Note that if $i-1$ is divisible by m, then r_{i-1} is a child of r on level $i-1$. Therefore the claim is also satisfied for all $\text{Fcn}(i-1, j = i-2, q = r_{i-1}, R_i)$, where $i-1$ is divisible by m.

Let us now show by induction that for parameters $(i, j = i-1, q = r_i, R_i)$ we have $\{r, r_i\} \subseteq R_i \subseteq \{r_{i+3}, r_{i+2}, r_{i+1}, r_i, r\}$. From the previous case $q = r$ we obtain the this claim is satisfied in the base case and when i is divisible by m. Assume now that the claim holds for some i, because the claim is satisfied for all i that are divisible by m, it is sufficient to prove that the claim holds for $i-1$, when $i-1$ is not divisible by m. By induction assumption we have $\{r_i, r_{i-1}, r\} \subseteq \mathcal{C}(R_i) = \{r_{i+3}, r_{i+2}, r_{i+1}, r_i, r, r_{i-1}\}$. By line 10: $d = d(r_i, r_{i-1}) = 2^i$. Since $d(r_{i-1}, r) = 2^i$ and $d(r_{i+3}, r_i) \ge 2^{i+2} + 2^{i+1} + 2^i$ we have

$$\{r, r_{i-1}\} \subseteq R_{i-1} = \{r_t \in \mathcal{C}(R_i) \mid d(r_{i-1}, r_t) \le 2^{i+2}\} \subseteq \{r_{i+2}, r_{i+1}, r_i, r_{i-1}, r\}.$$

At line 12 we recurse into $\text{Fcn}(i-1, j = i-1, q = r_i, R_{i-1})$. Note that $i = j$ in this iteration, therefore we proceed into query expansion (lines 13-17). Note that since $i-1$ is not divisible by m, r_{i-1} is a child of r_i at level $i-1$ and therefore FindComponentNeighbors is launched with parameters $(i-1, j = i-2, q = r_{i-1}, R_{i-1})$, which proves the claim.

Let t be such that $t \equiv 1 \mod m$. Let us now show that for any $i \le t$ in recursion $\text{Fcn}(i, j = i-1, q = r_t, R_i)$ we have

$$\{r, r_t\} \subseteq R_i \subseteq \{r_{i+2}, r_{i+1}, r_i, r_t, r\}.$$

BaseCases $i = t$ follows from the previous paragraph. Assume now that the claim holds for i and let us show that it holds for $i-1$. Again $\{r_i, r_{i-1}, r\} \subseteq \mathcal{C}(R_i) = \{r_{i+3}, r_{i+2}, r_{i+1}, r_i, r, r_{i-1}\}$. By line 10

$d = d(r_t, r) = 2^t$, $d(r_t, r_{i-1}) = 2^i + 2^t$ and $d(r_{i+3}, r_t) \geq 2^t + 2^{i+2}$ we have

$$\{r, r_{i-1}\} \subseteq R_{i-1} = \{r_t \in \mathcal{C}(R_i) \mid d(r_{i-1}, r_t) \leq 2^{i+1} + 2^i + 2^t\} \subseteq \{r_{i+1}, r_i, r_{i-1}, r\}.$$

As in previous paragraph we proceed to $(i-1, j = i-1, q = r_t, R_{i-1})$, where $(i-1, j = i-2, q = r_t, R_{i-1})$ is launched. Therefore the claim holds.

It remains to show that Algorithm 4.2.2 that is launched from $(r, R_i = \{r\}, E = \emptyset)$ has $O(m^3)$ low bound on number of times a reference expansions (lines 9-15) was performed. Let ξ be the number of times Algorithm 4.2.2 performs reference expansions. For every $r' \in R$ denote $\xi(r')$ to be the number of reference expansions that were performed for $q_j = r'$. We note that any node r_u is introduced the in a query expansion (lines 9-11) of $(i = u + 1, j = u + 1)$. Since for all u satisfying $i \equiv 1 \mod m$ set R_i is non empty for all the levels $[1, u]$, we have $\xi(r_u) \geq u - 1$ for such u. Thus

$$\xi \geq \sum_{i \equiv 1 \mod m} \xi(r_i) \geq \sum_{u=1}^{m+1} u^2 - 2 = O(m^3).$$

By using Inequality 4.2.3.3 we obtain a contradiction

$$O(m^3) = \xi \leq \max_{p \in R} D(p) \cdot (\max_i \mathcal{C}(R_i)) \cdot \chi \leq (2m+1) \cdot 6 \cdot 2 \leq O(m).$$

4.3   Counterexample: linkage expansion constant c_l

The purpose of this section is to show linkage expansion constant c_l of [63, Definition 4] cannot be used to estimate maximal size of reference set R_i in proof of [63, Theorem 5.4]. Remark 4.3.3 shows that linkage expansion constant c_l was defined incompletely. Counterexample 4.3.5 uses Example 4.3.1 to show c_l was incorrectly used in an important step in a proof of [63, Theorem 5.4].

Example 4.3.1 (two separated sets example)**.** Let (R, d) be a finite metric space consisting of 2^{k+1} points divided into two subsets A, B having size 2^k. Let $A = \{p_i \mid i \in 0, ..., 2^k - 1\}$ and let $B = \{p_i \mid i \in 0, ..., 2^k - 1\}$. Metric $d(p_i, p_j)$ is defined as follows: For any $1 \leq i < j \leq 2^k$, let $(a_1, ..., a_k)$ be the binary representation of i. Define $J(p_i, p_j) = \min_t \{t \in [1, k] \cap \mathbb{Z} \mid a_t \neq b_t\}$ and $d(p_i, p_j) = 1 + \frac{k+1-J(p_i, p_j)}{k+1}$. It follows that for all $p_i, p_j \in A$ we have $1 < d(p_i, p_j) < 2$. Similarly for $q_i, q_j \in B$ we define $d(q_i, q_j) = d(p_i, p_j)$ for all i, j. For any i, j we define $d(q_i, p_j) = 2^{10}$.

Let us now prove that (X, d) is a metric space. Notice that for any indices $i, j, t \in [1, 2^k]$ we have $d(p_i, p_j) + d(p_j, p_t) > 2$ and $d(p_i, p_t) < 2$. Therefore the triangle inequality is satisfied for all
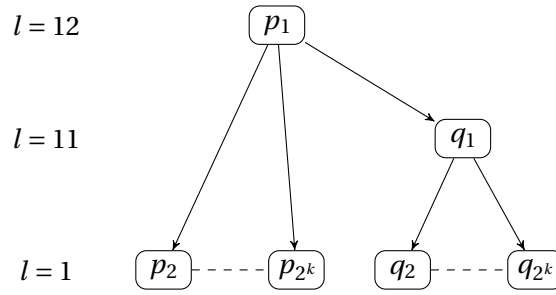
Figure 4.2: Compressed cover tree $\mathcal{T}(R)$ of Example 4.3.1. Level $l = 1$ contains points $p_2, p_3, ..., p_{2^k}$ that are all connected to p_1, as well as points $q_2, q_3, ..., q_{2^k}$ that are all connected to q_1.
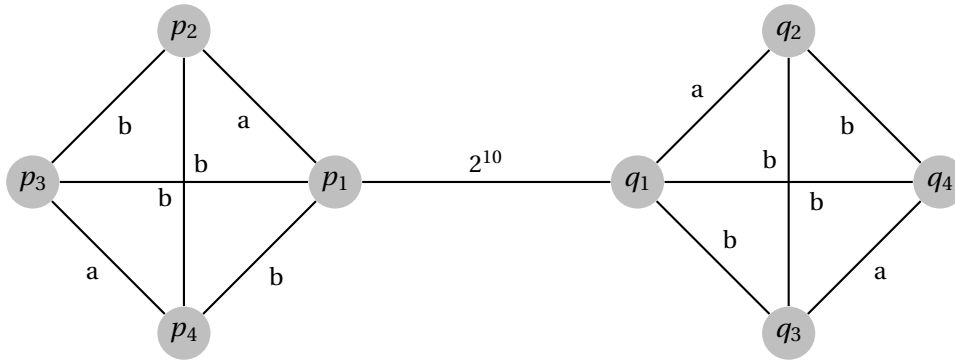
Figure 4.3: This figure illustrates clusters $A = \{p_1, p_2, p_3, p_4\}, B = \{q_1, q_2, q_3, q_4\}$ of Example 4.3.1 for $k = 2$. In this picture we have $a = 1\frac{1}{3}$ and $b = 1\frac{2}{3}$.

triplets inside A, and by the same argument for B as well. Let $p_i \in A$ and $q_j, q_t \in B$ be arbitrary nodes . Then $d(p_i, q_j) = 2^{10} \leq 2^{10} + d(q_t, q_j) \leq d(p_i, q_t) + d(q_t, q_j)$. We have now shown that (R, d) is indeed a metric space. Let us now construct compressed cover tree $\mathcal{T}(R)$.

Let p_1 be the root of $\mathcal{T}(R)$. Set $l(q_1) = 2$ and let q_1 be a child of p_1. Set $l(p_i) = 1$ for all $p_i \in A \setminus \{p_1\}$ and set their parent to be p_1. Similarly for all $q_i \in B \setminus \{q_1\}$ set $l(q_i) = 1$ and let q_1 be their parent. We have $C_{12} = \{p_1\}$, $C_{11} = \{p_1, q_1\}$, $C_{10} = \{p_1, q_1\}$, ... , $C_2 = \{p_1, q_1\}$, $C_1 = R$. Since for all i both inequalities $d(p_1, p_i) \leq 2$ and $d(q_1, q_i) \leq 2$ are satisfied we note that $\mathcal{T}(R)$ satisfies the covering condition 3.2.1(b). Note that $d(a, b) > 1$ for all $a, b \in R$, therefore the separation condition 3.2.1(c) holds as well.   ∎

Definition 4.3.2 (Borůvka clustering, [63, Definition 2])**.** Given a finite metric space (R, d), *Borůvka clustering* is an indexed collection of partitions $\mathcal{F}_i$. Partition $\mathcal{F}_0$ is the partition into isolated points $\{\{p\} \mid p \in R\}$. For any i recall from Definition 4.4.6 that $\mathrm{NN}(U, \mathcal{F}_i)$ is the set of

nearest neighboring components of $U \in \mathcal{F}_i$. Partition $\mathcal{F}_i$ is obtained from $\mathcal{F}_{i-1}$ by applying *Borůvka step* on $\mathcal{F}_{i-1}$: In every step for every $V \in \mathcal{F}_{i-1}$ we select $U \in \text{NN}(V, \mathcal{F}_{i-1})$ and merge V with chosen subset U. This process, illustrated in Figure 4.4 is repeated until $\mathcal{F}_i$ becomes a partition consisting of one element $\{R\}$. $\blacksquare$

Remark 4.3.3 (Cases where linkage expansion constant c_l is undefined). Recall that linkage expansion constant of [63, Page 605] was defined as: " [63, Definition 4]: [r is replaced with t, Borůvka clustering D replaced is replaced with $\mathcal{F}$]. Linkage Expansion Constant. Let C_1 and C_2 be two clusters in the Borůvka clustering at level i and let $S_1 \subseteq C_1$ and $S_2 \subseteq C_2$. Let $B_i^l(S_1, S_2, t)$ be the set of all pairs (p, q) such that $p \in S_1$, $q \in S_2$ and $d(p, q) \leq t$.

Definition 4: The linkage expansion constant is the smallest real number c_l such that

$$B_i^l(S_1, S_2, 2t) \leq c_l |B_i^l(S_1, S_2, t)|,$$

for all levels of the Borůvka clustering $\mathcal{F}_i$, clusters C_1 and C_2 at level i, subsets $S_1 \subseteq C_1$ and $S_2 \subseteq C_2$, and distances $t > 0$. "

Borůvka clustering $\mathcal{F}$ was originally introduced in [63, Definition 1] and restated as in Defintion 4.3.2 in this chapter. Let R, A, B be as in Example 4.3.1. It will be shown in Counterexample 4.3.5, that in the previous to last Borůvka step has $\mathcal{F} = \{A, B\}$. Let i be the index of this step. Then for $t = 2^9$ set $B_i^l(A, B, 2t)$ all edges between A and B, therefore $|B_i^l(A, B, 2t)| = 2^{2k}$. However, by definition $|B_i^l(A, B, t)| = 0$. Therefore there does not exist real number c_l satisfying $B_i^l(A, B, 2t) \leq c_l |B_i^l(A, B, t)|$. We can conclude that Definition 4 is undefined for small values t. $\blacksquare$

Remark 4.3.4 (Cluster expansion constant). In this remark, we cite the Cluster expansion constant that will be used in Counterexample 4.3.5

[63, Definition 3]: [r is replaced with t, set S replaced with set R, Borůvka clustering D replaced is replaced with $\mathcal{F}$]." Given the Borůvka clustering, we define the new expansion constants. Let $B_i^c(q, t)$ be the set of all components C_p with a point $p \in C_p$ such that $d(q, p) \leq t$. Using this component-wise ball, we define the cluster expansion constant.

Definition 3: The cluster expansion constant is the smallest real number c_p such that

$$|B_i^c(q, 2t) \leq c_p |B_i^c(q, t)|,$$

for all points $q \in R$, distances $t > 0$, and each level of the Borůvka clustering $\mathcal{F}_i$

In Counterexample 4.3.5 it will be shown that there were challenges with using linkage expansion constant in the proof of [63, Theorem 5.4]

Counterexample 4.3.5 (Linkage expansion constant c_l cannot be used to estimate the size of reference set $\max_i R_i$). **The Buildup:** Let $R, d, \mathcal{T}(R), k, A, B$ be as in Example 4.3.1. Let $\mathcal{F}_i$ be the Borůvka clustering introduced in Definition 4.3.2. Let us first show that after k-iterations of the while-loop in Algorithm 4.2.1 on an input $\mathcal{T}(R)$ we have $\mathcal{F}_k = A, B$.

For nodes $p_i, p_j \in A$ define relation $p_i \frown_t^A p_j$ if indices i and j have exactly the same first $k - t$ numbers in their binary representations. Define relation $\frown_i^B$ similarly for nodes in B. Let us prove by induction that for all $t \in [0, k]$ we have $F_t = [\frown_t^A] \cup [\frown_t^B]$, where $[\frown_t^A]$ and $[\frown_t^B]$ are collections of classes spanned by relation $\frown_t^A$ and $\frown_t^B$, respectively. The basecase of induction follows by noting that F_0 has every node in its own separate component.

Let us now prove the induction step. We will focus only on elements $p_i \in A$, since case for $q_i \in B$ can be proved similarly. Assume that the claim holds for some $u \in [0, k-1]$, let us show that it holds for $u + 1$. By definition of metric $d(p_i, p_j) = 1 + \frac{k+1-J(p_i,p_j)}{k+1}$. By induction assumption any pair of nodes that share u common numbers in the binary representations of their indices are already connected. Therefore for any p_i we have $\min_{w \sim_u p_i} d(p_i, w) > 1 + \frac{k+1-u}{k+1}$. Since A contains all indices of the interval $[1, 2^k]$ for every p_i we can find $w \in A$ in such a way that p and w have $k - t - 1$ same number in their binary representations , but the number at $k - t$ differs. Since there are no integers between $k - 1$ and $k - t - 1$ we have $\min_{w \sim_u p_i} d(p_i, w) = 1 + \frac{k+1-u+1}{k+1}$ for all p_i. It follows that in F_{t+1} all the nodes that have $k - t - 1$ same numbers in their binary representations will be merged. Note that $[\frown_k^A] = \{A\}$ and $[\frown_k^A] = \{B\}$. Therefore it follows that $\mathcal{F}_k = \{A, B\}$. Consider now the following quote:

Exact quote of *[63, Theorem 5.4]:[d' replaced with t] " Consider the other case when $d > 2^{i+2}$ ...*

... We now bound the number of points within a component that q_j may have to consider. Let C_r be a component distinct from C_q . Let $L(q_j)$ denote the set of all leaves that are descendants of q_j . Let $t = \min_{q \in L(q_j), r \in C_r} d(q, r)$. Then,

$$|B_k^l(C_q, C_r, d + 2^{i+1} + 2^i)| \leq |B_k^l(C_q, C_r, 4(d - 2^{i+1}))|$$
$$\leq c_l^2 |B_k^l(C_q, C_r, d - 2^{i+1})|$$
$$\leq c_l^2 |B_k^l(C_q, C_r, t)|.$$

By the above argument, there can be at most one pair in $B_k^l(C_q \cap L(q_j), C_r, t)$. Therefore, there are

at most c_l^2 points in C_r contained in $B(q_j, d + 2^{i+1} + 2^i)$. In the worst case, each of the point is at level C_{i-1} of the tree and must be considered in R_{i-1}. There are at most c_p^2 components C_r that can contribute points, so the maximum number of points in R_{i-1} is $c_p^2 c_l^2$. "

Note that set of leaves $L(q_j)$ corresponds to nodes $\mathcal{S}_j(q_j, \mathcal{T}(R))$ of compressed cover tree. We denote the component of q in $\mathcal{F}_k$ by C_q. Variable d was introduced in line 10 of Algorithm 4.2.2 and constant c_p, c_l were defined in Remark 4.3.3. The quote above states that if $d > 2^{i+2}$ number of the points is bounded $c_p^2 \cdot c_l^2$, where c_p^2 can be replaced by the number of compoonents at the current stage of the algorithm. Therefore for $k + 1$th iteration of lines 3-6 of Algorithm 4.2.1 the claim is

$$|R_{i-1}| \le c_l^2 \cdot (\text{Number of components in } \mathcal{F}_k).$$

The contradiction: In this counterexample it will be shown that $\mathcal{T}(R)$ from Example 4.3.1 does not satisfy the above inequality in Algorithm 4.2.2 launched with inputs $\mathcal{T}(R)$ and $\mathcal{F}_k$.

By Remark 4.3.3 $|B_k^l(C_q, C_r, t')|$ was indefined for all $t' < d(C_q, c_r)$. Let us change the definition by requiring $t' \ge d(C_q, C_r)$. In [63] constant c_l was defined as a maximum of constants over all partitions F_n. However, nothing prevents us by using the same argument of authors and focusing only on a single partition $\mathcal{F}_k$. In this case $\mathcal{F}_k$ consists of two components A, B which satisfy: for all $a \in A$ and $b \in B$ we have $d(a, b) = 2^{10}$. Therefore for any $S_1 \subseteq A$ and $S_2 \subseteq B$ we have $B(S_1, S_2, x) = S_1 \times S_2$ when $x \ge 2^{10}$ and indefined else. It follows that $c_l = 1$ in this case.

Let us now perform simulation of $k + 1$th iteration of Algorithm 4.2.1. We are especially interested in iteration $(p_1, R_2, j = 1, i = 2)$ of Algorithm 4.2.2. Let us start by running Algorithm 4.2.2 with using partition $\mathcal{F}_k$ and parameters $(p_1 R_{12} = \{p_1\}, i = 12, j = 12)$. Since $i = j = 12$ and q_1 is children of p_1 at level 11 we proceed into query expansion (lines 13-17), where we launch two separate instances $(p_1, R_{12} = \{p_1\}, i = 12, j = 11)$ and $(q_1, R_{12} = \{p_1\}, i = 12, j = 11)$. Let us focus only on iterations having p_1 as the query node. In $(p_1, R_{12} = \{p_1\}, i = 12, j = 11)$ we proceed into reference expansion (lines 8 -13) , where $\mathcal{C}(R_{12}) = \{p_1, q_1\}$, since q_1 is the only node that contains descendants that are not already connected to p_1 we get $R_{11} = \{q_1\}$. Note that both q_1 and p_1 do not contain any children on levels $[3, 11]$. Therefore we can skip to the case $i = 2$. Let us continue with iteration $p_1, R_2 = \{q_1\}, i = 2, j = 1)$, we proceed into reference expansion (lines 8-13). We have $\mathcal{C}(R_2) = B$. Now since all $b \in B$ are not connected to p_1 and $d(p_1, b) = 2^{10}$ for all b, we have $d = 2^{10}$ and $R_1 = B$. Since $d \ge 2^4 = 2^{i+2}$ we can use the arguments of the quote above. However, now we have:

$$|R_1| \le c_l^2 \cdot (\text{Number of components in } \mathcal{F}_k) \le 1 \cdot 2 \le 2.$$
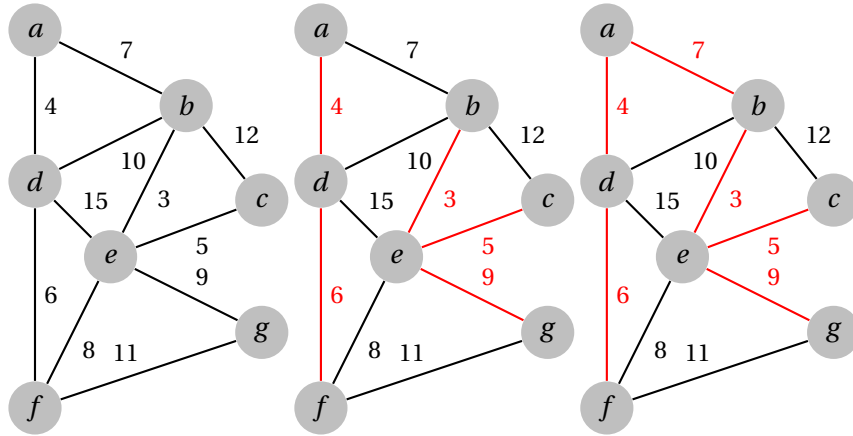
Figure 4.4: Borůvka's clustering on an example graph. **Left:** In $\mathcal{F}_0$ every point exists in its own separate cluster , **Middle:** $\mathcal{F}_1$ consists of two clusters $\{a, d, f\}$ and $\{b, c, e, g\}$, **Right:** In $\mathcal{F}_2$ all the points are in the same cluster.

Since $|R_1| = |B| = \frac{|R|}{2}$, this is a contradiction.   ∎

4.4   Time complexity proof for Singletree Borůvka MST

The main result of this section is Theorem 4.4.14 that resolves issues presented in Section 4.2 and Section 4.3 by giving weaker complexity result for Problem 4.1.2:

$$O(c_m(R)^{O(\log_2(\rho(R)))} \cdot |R| \cdot \log_2(|R|) \cdot \log_2(\Delta(R))),$$

where $\rho(R)$ is the maximal edge length of $\text{MST}(R)$ divided by its minimum edge length and $\Delta(R)$ is aspect ratio of R from Definition 3.2.14. Algorithm 4.4.3, which solves the problem is a single-tree analog of the original dual-tree metric minimum spanning tree [63, Algorithm 3].

Algorithm 4.4.3, which resolves Problem 4.1.2 is based on the idea of Borůvka algorithm. The computation is started by placing all points of R in their own components. During every Borůvka step every component is merged with the component that minimizes the Hausdorff distance. It will be shown that there will be at most $O(\log(|R|))$ Borůvka steps. Theorem 4.4.14 shows that in every Borůvka step for every component in $\mathcal{F}$ we can find its nearest components in a near-linear time.

Definition 4.4.1 (Partition $\mathcal{F}$ of a finite metric space (R, d))**.** Let (R, d) be a finite metrics space.

A *a partition* $\mathcal{F}$ of (R, d) is a splitting of R into disjoint subsets called *clusters* if $R = \sqcup_{U \in F} U$ and $U \cap V = \emptyset$, for all distinct $U, V \in \mathcal{F}$. ∎

Recall that distinctive descendant set $\mathcal{S}_j(q, \mathcal{T}(R))$ was introduced in Definition 3.6.1 and that for any partition $\mathcal{F}$ and node $p \in R$ we denote $\mathcal{F}(p)$ to be the cluster of p in the partition $\mathcal{F}$.

Definition 4.4.2 (Cluster of descendants)**.** Let R be a finite subset of a metric space (X, d). Let $\mathcal{T}(R)$ be a compressed cover and let $\mathcal{F}$ be arbitrary partition of the set R. Let $p \in \mathcal{T}(R)$ be arbitrary node , let $\mathcal{F}(p)$ be its cluster and let i be arbitrary level of $\mathcal{T}(R)$. Define *cluster of descendants* $\tau_i(p) : R \to \mathcal{F} \times R$ in the following way:

$$\tau_i(p) = \begin{cases} (\mathcal{F}(p), \emptyset) & \text{If } \mathcal{S}_i(p, \mathcal{T}(R)) \subseteq \mathcal{F}(p) \\ (\emptyset, q) & \text{Else choose } q \in \mathcal{S}_i(p, \mathcal{T}(R)) \setminus \mathcal{F}(p) \end{cases}$$

Note that if $\mathcal{S}_i(p, \mathcal{T}(R))$ is not completely contained into some cluster, then $\tau_i(p)$ maps into $(\emptyset, q)$ where $q \in \mathcal{S}_i(p, \mathcal{T}(R))$ is arbitrary node that doesn't belong to $\mathcal{F}(p)$. ∎

In Figure 3.9 assume that $\mathcal{F} = \{U, V\}$, where $U = \{6, 8, 9\}$ and $V = \{2, 3, 4, 5\}$. Then $\tau_2(2) = 6$, $\tau_1(6) = V$, $\tau_1(2) = U$. Let $\mathcal{F}$ be an arbitrary partition of R. For every $p \in R$ we denote $p \in \mathcal{F}(p)$ to be the cluster of p in $\mathcal{F}$.

Definition 4.4.6 (Nearest neighboring component)**.** Let $\mathcal{F}$ be an arbitrary partition of a finite metric space (R, d). For any two sets $U, V \in \mathcal{F}$ define the Hausdorff distance as $d_H(U, V) = \min\{d(x, y) \mid x \in U \text{ and } y \in V\}$. For every component $U \in F$ the nearest neighbor set $\text{NN}(U, \mathcal{F})$ consists of all the components $V \in \mathcal{F}$ satisfying $d_H(U, V) \leq \min_{W \in \mathcal{F}} d_H(U, W)$. ∎

Lemma 4.4.7 (True nearest components are always descendants of R_i for all levels i)**.** *Let R be a finite subset of an ambient metric space (X, d). Let $\mathcal{T}(R)$ be a compressed cover tree of R. Let $\mathcal{F}$ be arbitrary partition of R and let $U \in \mathcal{F}$ be an arbitrary cluster. Then for any iteration $i \in H(\mathcal{T}(R))$ of lines 4-17 of Algorithm 4.4.5 there exists $V \in \text{NN}(U, \mathcal{F})$ and node $\beta \in \bigcup_{p \in R_i} \mathcal{S}_i(p, \mathcal{T}(R)) \cap V$ for which $d(U, \beta) = d(U, V)$ for any $V' \in \text{NN}(U, \mathcal{F})$.* ∎

Proof. Assume contrary. Since $R_{l_{\max}} = \{r\}$, where r is the root $\mathcal{T}(R)$ we have $S_{l_{\max}}(r, \mathcal{T}(R)) = R$ and therefore we always have $\bigcup_{p \in R_{l_{\max}}} \mathcal{S}_i(p, \mathcal{T}(R)) \cap V \neq \emptyset$. Let i be largest index for which $\bigcup_{p \in R_{i-1}} \mathcal{S}_i(p, \mathcal{T}(R)) \cap \text{NN}(U, \mathcal{F}) = \emptyset$. Let β be arbitrary point satisfying $\beta \in \bigcup_{p \in R_i} \mathcal{S}_i(p, \mathcal{T}(R))$ and $d(U, \beta) = d(U, V)$ for any $V \in \text{NN}(U, \mathcal{F})$. By assumption we have:

$$\beta \in \bigcup_{p \in R_i} \mathcal{S}_i(p, \mathcal{T}(R)) \setminus \bigcup_{p \in R_{i-1}} \mathcal{S}_{i-1}(p, \mathcal{T}(R)).$$

Algorithm 4.4.3 Singletree Borůvka MST based on compressed cover trees.

1: **Input:** A compressed cover tree $\mathcal{T}(R)$ with a root r

2: **Output:** Minimum spanning tree MST(R)

3: Let G be a graph with vertex set R and an empty edge set.

4: Let $\mathcal{F}$ be a partition of R into isolated points.

5: **while** $|\mathcal{F}| > 1$ **do**

6:     Precompute τ of Definition 4.4.2 by running Algorithm 4.4.4 on $(l_{\max}(\mathcal{T}(R)), r)$.

7:     **for** $U \in \mathcal{F}$ **do**

8:         Set $\mathcal{C}(U) =$ Algorithm 4.4.5 executed on $(\mathcal{T}(R), U)$.

9:     **end for**

10:    **for** $U \in \mathcal{F}$ **do**

11:        Let $(q, p) = \mathcal{C}(U)$.

12:        **if** $\mathcal{F}(q) \neq \mathcal{F}(p)$ **then**

13:            Add edge (q, p) to G.

14:            Merge components $\mathcal{F}(q)$ and $\mathcal{F}(p)$ in $\mathcal{F}$.

15:        **end if**

16:    **end for**

17: **end while**

18: **return** graph G

Algorithm 4.4.4 This algorithm runs a depth-first traversal for a compressed cover tree $\mathcal{T}(R)$ to find all subtrees, where all the nodes of the subtree belong to the same component.

1: **Function** : FindClusters(a level i of $\mathcal{T}(R)$, a node p of $\mathcal{T}(R)$)

2: **Output** : A connected component $U \in \mathcal{F}$ or $\emptyset$.

3: Set $U = F(p)$ {U is the cluster containing p}

4: **if** $i > l_{\min(\mathcal{T}(R))}$ **then**

5:     Set $\mathcal{A} = \{p\} \cup \{q \in \text{Children}(p) | l(q) = i - 1\}$

6:     $j \leftarrow 1 + \text{Next}(p, i - 1, \mathcal{T}(R))$.

7:     **for** $a \in \mathcal{A}$ **do**

8:         $(V, b) = \text{FindClusters}(a, j)$.

9:         **if** $U \neq V$ **then**

10:            Set $\tau_i(p) = (\emptyset, b)$ and **return** $(\emptyset, b)$.

11:        **end if**

12:    **end for**

13: **end if**

14: Set $\tau_i(p) = (U, \emptyset)$ and **return** $(U, \emptyset)$.

Algorithm 4.4.5 Single tree Borůvka step

1: **Input** : a compressed cover tree $\mathcal{T}(R)$, a cluster $U \in \mathcal{F}(R)$
2: Set $i \leftarrow l_{\max}(\mathcal{T}(R))$
3: Let r be the root node of $\mathcal{T}(R)$. Set $R_i = \{r\}$.
4: **while** $i > l_{\min}$ **do**
5: $\quad$ Assign $\mathcal{C}(R_i) \leftarrow \{a \in \text{Children}(p) \text{ for some } p \in R_i \mid l(a) \geq i - 1\}$
6: $\quad$ {**Recall** that Children(p) contains node p }
7: $\quad$ Set $\mathcal{C}^*(R_i) = \{p \in \mathcal{C}(R_i) \mid \tau_i(p) \neq U\}$ {**Note:** τ appears in Definition 4.4.2. }
8: $\quad$ Set $l = \infty$.
9: $\quad$ **for** $q \in U$ **do**
10: $\quad\quad$ $l_1 \leftarrow \min\limits_{p \in \mathcal{C}^*(R_i)} \{d(q,p) + 2^i \mid \mathcal{F}(p) = U\}$.
11: $\quad\quad$ $l_2 \leftarrow \min\limits_{p \in \mathcal{C}^*(R_i)} \{d(q,p) \mid \mathcal{F}(p) \neq U\}$.
12: $\quad\quad$ $l \leftarrow \min\{l, l_1, l_2\}$.
13: $\quad$ **end for**
14: $\quad$ Find $R_{i-1} = \{p \in \mathcal{C}^*(R_i) \mid d(U,p) \leq l + 2^i\}$
15: $\quad$ Set $j \leftarrow \max_{a \in R_{i-1}} \text{Next}(a, i - 1, \mathcal{T}(R))$ {If such j is undefined, we set $j = l_{\min}$}
16: $\quad$ Set $R_j = R_{i-1}$ and $i = j$
17: **end while**
18: **returns** pair $(q, p) \in U \times R_{l_{\min}} \setminus U$ minimizing $d(q, p)$

By Lemma 3.6.16 we have:

$$\bigcup_{p \in \mathcal{C}(R_i)} \mathcal{S}_{i-1}(p, \mathcal{T}(R)) = \bigcup_{p \in R_i} \mathcal{S}_i(p, \mathcal{T}(R)) \tag{4.4.7.4}$$

It follows that $\beta \in \bigcup_{p \in \mathcal{C}(R_i)} \mathcal{S}_{i-1}(p, \mathcal{T}(R))$. We denote $\alpha \in \mathcal{C}(R_i)$ to be the node that satisfies $\beta \in \mathcal{S}_{i-1}(\alpha, \mathcal{T}(R))$. By counter assumption $d(U, \alpha) > l + 2^i$. By using triangle inequality we have:

$$d(U, \beta) \geq d(U, \alpha) - d(\alpha, \beta) \geq d(U, \alpha) - 2^i > l \tag{4.4.7.5}$$

Note that $\beta \in \mathcal{C}^*(R_i)$ by line 7 of Algorithm 4.4.5 , therefore $\mathcal{C}^*(R_i)$ is non-empty. By definition $l = \min\{l_1, l_2\}$. Let us assume first that

$$l = l_1 = \min_{q \in U} \min_{p \in \mathcal{C}^*(R_i)} \{d(q, p) + 2^i \mid \mathcal{F}(p) = U\}, \tag{4.4.7.6}$$

Let $q \in U$ and $\gamma \in \mathcal{C}^*(R_i)$ be the points minimizing the distance of (4.4.7.6). Let $w \in \mathcal{S}_{i-1}(\gamma, \mathcal{T}(R))$ be an arbitrary point for which $w \notin U$. By Lemma 3.6.15 applied on $i - 1$ we have $d(\gamma, w) \leq 2^i$. It

follows:

$$d(w,q) \leq d(q,\gamma) + d(\gamma,w) \leq d(q,\gamma) + 2^i = l \qquad (4.4.7.7)$$

By combining inequalities (4.4.7.5) and (4.4.7.7) and since $q \in U$ we have:

$$d(U, \mathcal{F}(w)) \leq d(U, w) \leq d(q, w) < d(U, \beta).$$

It follows that there exists $V \in \mathcal{F}$ for which $d(\beta, U) > d(V, U)$, which is a contradiction. The case where $l = l_2 = \min\limits_{p \in \mathcal{C}^*(R_i)} \{d(U, p) \mid \mathcal{F}(p) \neq U\}$ can be proven similarly. $\qquad \square$

Lemma 4.4.8 (the lowest cost edge of cluster is always contained in MST(R))**.** *Let (R, d) be a finite metric space and let* MST(R) *be a minimum spanning tree of R. Let $U \subseteq R$ be a set for which graph* MST(R) *restricted to vertex set U is connected. Then for any $(a, b) \in$ MST(R) having $a \in U$ and $b \in R \setminus U$ minimizing the distance $d(U, R \setminus U)$, there exists possibly another minimum spanning tree P of R having $P|U = R|U$ and $(a, b) \in P$.*

Proof. Since R is finite, we can pick $a \in U$ and $b \in R \setminus U$ minimizing the distance $d(U, R \setminus U)$ Let C be a component of MST$(R) \setminus U$ containing b. Assume that there is no edges between C to U in MST(R). Then since MST$(R) \setminus U$ differs from MST(R) only by edges having an end point in U it follows that MST(R) is not connected, which is a contradiction. Therefore there exists some $(f, g) \in$ MST(R) connecting U to C. Note that $P =$ MST$(R) \setminus (f, g) \cup (a, b)$ is a spanning tree of R. By assumption $d(a, b) \leq d(f, g)$. Denote the sum of edge lengths by $w(\cdot)$, then $w(P) \leq w($MST$(R))$. It follows that P is a minimum spanning tree of R. $\qquad \square$

Theorem 4.4.9 (Correctness of Algorithm 4.4.3)**.** *Given any compressed cover tree $\mathcal{T}(R)$ on a reference set R Algorithm 4.4.3 solves Problem 4.1.2 by finding some Minimum Spanning Tree* MST(R).

Proof. Since $l_{\min}$ is the minimal level of $\mathcal{T}(R)$ we have $\bigcup_{p \in R_{l_{\min}}} \mathcal{S}_i(p, \mathcal{T}(R)) = R_{l_{\min}}$. By Lemma 4.4.7 for any component $U \in \mathcal{F}$ we can find $\beta \in R_{l_{\min}}$ satisfying $d(U, \beta) = d(U, R \setminus U)$. Therefore line 18 of Algorithm 4.4.5 returns correctly a pair (q, β) that minimizes distance between U and $R \setminus U$.

Let us now show by induction on Borůvka step (lines 5-17) that graph G is always a subset of some MST(R). In base case G is a graph with empty edge set and therefore the claim holds trivially. Assume now that G satisfies the assumption at the beginning of some execution of Borůvka step. During every execution of lines 10 - 16 for every $U \in \mathcal{F}$ we add an edge (a, b) if

$\mathcal{F}(a) \neq \mathcal{F}(b)$, where $a \in U$ and $b \in R \setminus U$ minimizing distance $d(U, R \setminus U)$. By using Lemma 4.4.8 repetitively we obtain that the resulting graph G is a minimum spanning tree of R at the end of each Borůvka step. Since the algorithm is terminated when $|\mathcal{F}| = 1$, using the induction step we conclude that the final output G of the Algorithm 4.4.3 is a minimum spanning tree of R.

□

Recall that the set of essential levels $\mathcal{E}(p, \mathcal{T}(R))$ was introduced in Definition 3.6.3. Lemma 4.4.10 shows that for any compressed cover tree $\mathcal{T}(R)$ and any partition $\mathcal{F}$ of R it is possible to precompute $\tau_i(p, \mathcal{T}(R), F)$ for all $p \in \mathcal{T}(R)$ and $i \in \mathcal{E}(p, \mathcal{T}(R))$ in $O(|R|)$ time.

Lemma 4.4.10. *Let R be a finite subset of a metric space. Let $\mathcal{T}(R)$ be a compressed cover tree on R. Let F be an arbitrary partition of set R. Recall that $\mathcal{E}(\mathcal{T}(R), p)$ was introduced in Definition 3.6.3. Then Algorithm 4.4.4 launched for $(r, l_{\max}(\mathcal{T}(R)))$ finds $\tau(p, i, \mathcal{T}(R), F)$ of Definition 4.4.2 for all $p \in \mathcal{T}(R)$ and $i \in \mathcal{E}(p, \mathcal{T}(R))$ in $O(|R| \cdot \alpha(|R|))$.* ∎

Proof. Let us prove the time complexity claim of Algorithm 4.4.4. By Lemma 3.6.4 we have $|\sum_{p \in R} \mathcal{E}(\mathcal{T}(R), p)| \leq 2 \cdot |R|$. Since FindComponents() is launched once for every (p, i), where $p \in Q$ and $i \in \mathcal{E}(\mathcal{T}(R), p)|$. The total number of iterations is bounded by $2 \cdot |R|$. In every iteration we search for a component of p. This is done using the union-find data structure that performs the operation in $O(\alpha(|R|))$. Therefore it follows that the total time complexity is: $O(|R| \cdot \alpha(|R|))$. □

Definition 4.4.11. Let MST(R) be an arbitrary minimum spanning tree of a finite metric space (R, d). Define $d_{\min}(R)$ to be the minimal edge length of MST(R) and $d_{\max}(R)$ to be the maximal edge length of MST(R). Define $\rho(R) = 17 + 8 \cdot \frac{d_{\max}(R)}{d_{\min}(R)}$. ∎

Note that $d_{\min}$ was originally defined in Definition 3.2.14. Edge lengths $d_{\max}(R)$, $d_{\min}(R)$ do not depend on the choice of minimum spanning tree MST(R) and are therefore well-defined. Let $\alpha(|R|)$ be the inverse Ackermann function [82], which grows very slowly e.g. $\alpha(10^{80}) \leq 4$.

Lemma 4.4.12 (Run-time of Algorithm 4.4.5)**.** *Let R be a finite reference set in a metric space (X, d). Given a compressed cover tree $\mathcal{T}(R)$ and any cluster $U \subseteq R$, the run-time of Algorithm 4.4.5 is*

$$O(c_m(R)^{4 + \lceil \log_2(\rho(R)) \rceil} \cdot |H(\mathcal{T}(R))| \cdot |U| \cdot \alpha(|R|)),$$

where all parameters were introduced in Definitions 3.1.4, 3.2.13, 4.4.11.

Proof. In while loop (lines 4-17) the algorithms encounters all levels $H(\mathcal{T}(R))$. Therefore the number of iterations is bounded by the height $|H(\mathcal{T}(R))|$. The total number of nodes in line 6

during a single iteration by Lemma 3.2.9 is at most $(c_m(R))^4 \cdot \max_i |R_i|$. Since τ_i of Definition 4.4.2 was precomputed in Algorithm 4.4.4, the time complexity of line 7 is the same as line 6. Lines 9 - 13 takes at most $O(|U| \cdot |\mathcal{C}^*(R)|) \leq O(|U| \cdot (c_m(R))^4 \cdot \max_i |R_i|)$ time. Line 14 never does more work that line 6. In line 15 we keep track of $\text{Next}(a, i, \mathcal{T}(R))$, therefore its time complexity is the same as line 6. It follows that the time complexity of lines 4 - 17 is bounded by $O(|U| \cdot (c_m(R))^4 \cdot \max_i |R_i|)$. Line 18 takes at most $\max_i |R_i| \cdot \alpha(|R|)$ time. Consequently, the running time of the whole algorithm is bounded by:

$$O((c_m(R))^4 \cdot |U| \cdot \max_i |R_i| \cdot D(\mathcal{T}(R)) \cdot \alpha(|R|)). \tag{4.4.12.1}$$

To finish the proof we will show that $\max_i |R_i| \leq c_m(R)^{\lceil \log_2(\rho(R)) \rceil}$. Since $l \leq d_{\max}(R) + 2^i$, by line 14 of Algorithm 4.4.5 we have the following inclusion of subsets:

$$R_{i-1} = \{r \in \mathcal{C}^*(R_i) \mid d(q, r) \leq l + 2^i\} \subseteq \{r \in \mathcal{C}^*(R_i) \mid d(q, r) \leq d_{\max}(R) + 2^{i+1}\}$$

Note that during any iteration we have $i - 1 \geq l_{\min}(R)$ and $2^{l_{\min}(R)+1} \geq d_{\min}(R)$. It follows that $2^{i-1} \geq 2^{l_{\min}(R)} \geq \frac{d_{\min}(R)}{2}$. Therefore $\frac{1}{2^{i-1}} \leq \frac{2}{d_{\min}(R)}$. By Lemma 3.2.8 with $\delta = 2^{i-1}$ and $t = d_{\max}(R) + 2^{i+1}$ we have

$$\mu = 4 \frac{t}{\delta} + 1 = 4 \cdot \frac{2^{i+1} + d_{\max}(R)}{2^{i-1}} + 1 \leq 17 + 8 \cdot \frac{d_{\max}(R)}{d_{\min}(R)} = \rho(R).$$

It follows that $\max_i |R_i| \leq c_m(R)^{\lceil \log_2(\mu) \rceil} \leq c_m(R)^{\lceil \log_2(\rho(R)) \rceil}$. The by substituting $\max_i |R_i|$ with $c_m(R)^{\lceil \rho(R) \rceil}$ in (4.4.12.1) we obtain the final bound. $\square$

Lemma 4.4.13 is used to show that Algorithm 4.4.3 performs at most $O(\log(|R|)$ borůvka steps, lines 7- 9. This lemma was skipped in the original work [63].

Lemma 4.4.13 (Borůvka step halves the number of components)**.** *Let A be a finite set and let $f : A \rightarrow A$ be a function satisfying $f(x) \neq x$ for all $x \in A$. For any $k \in \mathbb{N}$ define f^k to be k-compositions of function f. Define relation $\backsim$ on A, in such a way that for any $y, z \in A$ we have $y \backsim z$, if there exists $k \in \mathbb{N}$ having $f^k(y) = z$ or $f^k(z) = y$. Then $|A/ \backsim_f| \leq \frac{|A|}{2}$.*

Proof. Assume that $A = \{a_1, ..., a_n\}$ Let us prove this lemma by induction. Base case $|A| = 2$ is trivial, since $f(a_1) = a_2$ and $f(a_2) = a_1$ is forced. Therefore $a_1 \backsim a_2$ and $|A/ \backsim_f| = 1 \leq \frac{|A|}{2}$. For base case $|A| = 3$ consider all possible functions f. Since all the cases are symmetrical lets us consider only one variation $f(a_1) = a_2$, $f(a_2) = a_3$ and $f(a_3) = a_2$. By definition we have $a_1 \backsim a_2 \backsim a_3$ and therefore $|A/ \backsim_f| = 1 \leq \frac{3}{2} = \frac{A}{2}$.

Let us now prove the induction step. Assume that the claim holds for all $|A| \leq n$ and let us

prove that it holds for $n+1$. Let $|A| = n+1$ and let $f : A \to A$ be a function having $f(x) \neq x$ for all $x \in A$. Assume first that $f^2(a_{n+1}) = a_{n+1}$. Denote $B = A \setminus \{a_{n+1}, f(a_{n+1})\}$. Define new function $g(x) = f(x)$ if $f(x) \neq a_{n+1}$ and $g(x) = f^2(x)$. Since $|B| \leq n-1 \leq n$, we can use induction assumption to obtain $|B/ \frown_g | \leq \frac{|B|}{2} \leq \frac{n-1}{2}$.

Let us show that if $u, v \in B$ are such that $u \frown_f v$, then $u \frown_g v$. Without loss of generality assume that $u = f^n(v)$ for some $n \in N$. By definition of g for any u we have $v = f^n(u) = g_m(u)$ for some $m \leq n$. Therefore we have $u \frown_g v$. Since $f^2(a_{n+1}) = a_{n+1}$ pair $\{a_{n+1}, f(a_{n+1})\}$ is a component in $A/ \frown_f$. We conclude that $A/ \frown_f = \{a_{n+1}, f(a_{n+1})\} \cup B \frown_f \subseteq \{a_{n+1}, f(a_{n+1})\} \cup B \frown_g$. Therefore it follows that:

$$|A/ \frown_f | \leq |B/ \frown_g | + 1 \leq \frac{n-1}{2} + 1 \leq \frac{n+1}{2} \leq \frac{|A|}{2}.$$

Assume then that $f^2(a_{n+1}) \neq a_{n+1}$. Define similarly $g(x) = f(x)$ if $f(x) \neq a_{n+1}$ and $g(x) = f^2(x)$ else. Let $B = A \setminus \{a_{n+1}\}$. Since $|B| \leq n$, by induction assumption we have $|B/ \frown_g | \leq \frac{|B|}{2} \leq \frac{n}{2}$. Let us show that $|A/ \frown_f | \leq |B/ \frown_g |$. Let $u, v \in B$ be such that $u \frown_f v$. Without loss of generality assume that $u = f^n(v)$. By definition of g we have $u = f^n(v) = g^m(u)$ for some $m \leq n$. Therefore $u \frown_g v$, which shows that $|A/ \frown_f | \leq |B/ \frown_g |$. The claim follows by noting that:

$$|A/ \frown_f | \leq |B/ \frown_g | \leq \frac{n}{2} \leq \frac{n+1}{2} \leq \frac{|A|}{2}.$$

$\square$

Theorem 4.4.14 (complexity for exact single tree MST search)**.** *Let R be a finite reference set in a metric space* (X, d). *Given a compressed cover tree* $\mathcal{T}(R)$ *Algorithm 4.4.3 find* $\mathrm{MST}(R)$ *in time:*

$$O\Big((c_m(R))^{4 + \lceil \log_2(\rho(R)) \rceil} \cdot |H(\mathcal{T}(R))| \cdot |R| \cdot \log_2(|R|) \cdot \alpha(|R|)\Big),$$

where all parameters were introduced in Definitions 3.1.4, 3.2.13, 4.4.11. ∎

Proof. Note first that during an iteration of while loop lines 5-17 of Algorithm 4.4.3 we merge every $U \in \mathcal{F}$ with some cluster $V \in \mathrm{NN}(U, \mathcal{F})$. Define function $f : \mathcal{F} \to \mathcal{F}$ by mapping $f(U) = V$ for all $U \in \mathcal{F}$, where V is some cluster in $\mathrm{NN}(U, \mathcal{F})$. By assumptions we have $f(U) \neq U$ for all $U \in \mathcal{F}$, therefore Lemma 4.4.13 can be applied on $\mathcal{F}$ and function f. We note that relation $\frown_f$ corresponds to the merging operation of lines 10 - 16. Let $\mathcal{R}$ be the partition $\mathcal{F}$ after executions of lines lines 10 - 16. Then by Lemma 4.4.13

$$|\mathcal{R}| = |\mathcal{F}/ \frown_f | \leq \frac{|\mathcal{F}|}{2}$$

Denote by k the total number of times lines 5-17 were executed. Since at the start of execution of Algorithm 4.4.3 we have $|\mathcal{F}| = |R|$ and the algorithm is terminated when $|\mathcal{F}| = 1$, we have $\frac{|R|}{2^{k-1}} \geq 1$. It follows that $k \leq \log(|R|) + 1$.

By Lemma 4.4.10 the time complexity of line 6 is $O(|R| \cdot \alpha(|R|))$. To prove the complexity we need to estimate the runtime of lines 7 - 9 that launch Algorithm 4.4.5. By Lemma 4.4.12 time complexity of line 8 is $O(c_m(R)^{4+\lceil \log_2(\rho(R)) \rceil} \cdot |H(\mathcal{T}(R))| \cdot |U| \cdot \alpha(|R|))$. Since $\mathcal{F}$ is partition of R, we have $\sum_{U \in \mathcal{F}} |U| = |R|$. It follows that runtime of lines 7 - 9 is $O\Big((c_m(R))^{4+\lceil \log_2(\rho(R)) \rceil} \cdot |H(\mathcal{T}(R))| \cdot |R| \cdot \alpha(|R|)\Big)$. Note that the time complexity of lines 10-16 is $O(|\mathcal{F}| \cdot \alpha(|\mathcal{F}|)) \leq O(|R| \cdot \alpha(|R|))$. Therefore the time complexity inside while-loop , lines 5-17 is dominated by lines 7 - 9. The claim of the theorem follows by recalling that lines 5-17 were executed $O(\log(|R|))$ times.

$\square$

Corollary 4.4.15. *Let R be a finite reference set in a metric space* (X, d)*. Given a compressed cover tree* $\mathcal{T}(R)$ *Algorithm 4.4.3 find* $\mathrm{MST}(R)$ *in time:*

$$O\Big((c_m(R))^{4+\lceil \log_2(\rho(R)) \rceil} \cdot \log_2(\Delta(R)) \cdot |R| \cdot \log_2(|R|) \cdot \alpha(|R|)\Big),$$

where all parameters were introduced in Definitions 3.1.4, 3.2.14, 4.4.11. ■

Proof. Follows from Theorem 4.4.14 by noting that $|H(\mathcal{T}(R))| \leq \log_2(\Delta(|R|))$. $\square$

4.5 Discussions: past challenges, contributions, and future work

The motivations were the past challenges in the proof of time complexities in [63, Theorem 5.1], which attempted to solve metric minimum spanning tree problem in $O(|R| \cdot \log_2(|R|))$ time. Section 4.2 and 4.3 show that the provided proof was incorrect. In Section 4.2 Counterexample 4.2.3 provides a concrete example, where the step in the proof of [63, Theorem 5.1] claims incorrect bound for the number of total recursions during a single Borůvka. Section 4.3 deals with problematic definition of linkage expansion constant c_l, which is undefined for small radius t. In Counterexample 4.3.5 we provide a concrete example, where the reference set size $|R_i|$ exceeds the claimed upper bound $c_p^2 \cdot c_l^2$, where c_p is cluster expansion constant. This ultimately shows that the proof of [63, Theorem 5.1] is incorrect and there is no easy way to fix it.

In Section 4.4 we use a different approach to prove the time complexity of the construction of the exact minimum spanning tree on any finite metric space. To resolve Problem 3.1.2 we use

the idea of Borůvka algorithm. However, to overcome past challenges the search algorithm is replaced with the modification of a single tree search for k-nearest neighbors. To avoid using multiple expansion constants, especially the undefined linkage expansion constant c_l in the statement and proof of the main Theorem 4.4.14 we use only a single, minimized expansion constant $c_m(R)$ with possibly big exponent $O(\rho(R)$, which depends on the ratio of maximal edge length and minimal edge length of MST(R). The main result, Theorem 4.4.14 gives

$$O(c_m(R)^{\log_2(\rho(R))} \cdot |R| \cdot \log_2(|R|) \cdot \log_2(\Delta(R)))$$

time complexity for finding MST(R), where $\Delta(R)$ is the aspect ratio of R.

In the future we wish to improve the time complexity to $O(c(R)^{O(1)} \cdot |R| \log(|R|))$ by utilizing $\mathcal{T}(R)$ as both, query and reference tree, where $c(R)$ is expansion constant of R. However, for such time complexity estimates it would be required to show that a complete paired-tree traversal of monochromatic pair $(\mathcal{T}(R), \mathcal{T}(R))$ could be done in time $c(R)^{O(1)} \cdot |R|$. Another approach for obtaining such time complexity could be by using Kruskal's algorithm as the base algorithm. Assuming that $\mathcal{T}(R)$ has been already constructed. It would be enough to show that any minimal non-zero sized edge in any Borůvka clustering $R/\frown$ can be found in $O(c(R)^{O(1)} \log(|R|))$ time. This is plausible since to find a non-trivial pair $a \in R$, $b \in R$ that minimizes $d(a, b)$ we can use $\mathcal{T}(R)$ as both, query and reference tree, and unlike in the standard dual-tree traversal, we can eliminate nodes that are far-away, from both: the query and the reference tree.

Chapter 5

Mergegram, extends 0D persistence

5.1 Introduction

Real-life objects are often represented by unstructured point clouds obtained by laser range scanning or by selecting salient or feature points in images [73]. Point clouds are easy to store and can be used as primitives for visualization [95]. The arguments above give a strong motivation for comparing and classifying unstructured point clouds.

Rigid objects are naturally studied up to rigid motion or isometry (including reflections), which is any map that preserves inter-point distances. The recognition of point clouds with fixed number of points is practically solved by the histogram of all pairwise distances, which is a complete isometry invariant in general position [12].

Real shapes are often given in a distorted form because of noisy measurements, when points are perturbed, missed or accidentally added. One of the first approaches to recognize nearly identical point clouds A, B of different sizes in the same metric space, for example in $\mathbb{R}^m$, is to use the *Hausdorff* distance Definition 2.1.2: $\mathrm{HD}(A, B) = \min \epsilon \geq 0$ such that the first cloud A is covered by ϵ-balls centered at all points of B and vice versa.

However, we also need to take into account infinitely many potential isometries of the ambient space $\mathbb{R}^m$. The exact computation of $\inf_f \mathrm{HD}(f(A), B)$ minimized over isometries f of $\mathbb{R}^m$ has a high polynomial complexity already for dimension $m = 2$ [20]. An approximate algorithm is cubic in the number of points for $m = 3$ [41].

This chapter extends the 12-page conference version [32], which introduced the new invariant mergegram but didn't prove its continuity under perturbations. In addition to the proof of

continuity, another contribution is Theorem 5.4.2 showing how to reconstruct a dendrogram of single-linkage clustering from a mergegram in general position.

The practical novelty is the harder recognition problem including perturbations of isometries within wider classes of affine and projective maps motivated by Computer Vision applications. Indeed different positions of cameras produce projectively equivalent images of the same rigid shape. The new experiments in section 5.7 extensively compared several approaches on 15000 clouds obtained from real images, see examples in Fig. 5.1.

Figure 5.1: Example images from the dataset of mythical creatures, which was introduced in [13].

Problem 5.1.1 (isometry shape recognition under noise)**.** Find an isometry invariant of point clouds in $\mathbb{R}^m$ that is

  (a) independent of a cloud size

  (b) provably continuous under perturbations of a cloud

  (c) computable in a near linear time in the size of a cloud

  (d) more efficient for recognizing isometry classes of clouds than past invariants.

∎

The key contributions are proofs of Theorems 5.4.2, 2.5.8 and Fig. 5.11, 5.12, 5.13 showing that the mergegram achieves a state-of-the-art recognition on substantially distorted images.

5.2 Past work

For the isometry classification of clouds consisting of the same number of points, the easiest invariant is the distribution of all pairwise distances, whose completeness (or injectivity) was proved for all point clouds in general (non-singular) position in $\mathbb{R}^m$ [12].

Fixed point clouds $A, B \subset \mathbb{R}^m$ of different sizes can be pairwisely compared by the *Hausdorff* distance [45] $\mathrm{HD}(A, B) = \max\{\sup_{p \in A} d_B(p), \sup_{q \in B} d_A(q)\}$, where $d_B(p) = \inf_{q \in B} d(p, q)$ is the (Euclidean or another) distance from a point $p \in A$ to the cloud B.

The rigid shape recognition problem for non-fixed clouds A, B is harder because of infinitely many potential isometries that can match A, B exactly or approximately. Partial cases of this problem were studied for clouds representing surfaces [30] and when two clouds have a given isometric matching of one pair of points [71]. Shape Google by Ovsjanikov et al. practically extends these ideas to non-rigid shape recognition [70].

The most general framework for the isometry shape recognition of point cloud data was proposed by Mémoi and Sapiro [67]. They studied the Gromov-Hausdorff distance $d_{GH}(A, B) = \inf_{f,g,M} \mathrm{HD}(f(A), g(B))$ minimized over all isometric embeddings $f : A \to M$ and $g : B \to M$ of given point clouds into a metric space M. Since the above definitions involve even more minimizations over infinitely many maps and spaces, GH can be only approximated.

The proposed invariant mergegram extends the 0-dimensional persistence in the area of Topological Data Analysis (TDA), which grew from the theory of size functions [86]. TDA views a point cloud $A \subset \mathbb{R}^m$ not by fixing any distance threshold but across all scales s, for example by blurring given points to balls of a variable radius s. The resulting evolution of topological shapes is summarized by a persistence diagram, which is invariant under isometries of $\mathbb{R}^m$. TDA can be combined with machine learning and statistical tools due to stability under noise, which was first proved by Cohen-Steiner et al. [23] and then extended to a very general form by Chazal et al. [19].

In dimension 0 the *persistence diagram* PD(A) for distance-based filtrations of a point cloud A consists of the pairs $(0, s) \in \mathbb{R}^2$, where values of s are distance scales at which subsets of A merge by the single-linkage clustering. These scales equal half-lengths of edges in a *Minimum Spanning Tree* MST(A). If distances between all points of A are known, MST(A) is a connected graph with the vertex set A and a minimal total length.

Representing a point cloud A by PD(A) loses a lot of geometry of A, but gains stability under perturbations, which can be expressed in the case of point clouds as BD(PD(A), PD(B)) $\leq$ HD(A, B). Here the *bottleneck distance* BD between diagrams is defined as a minimum $\epsilon \geq 0$ such that all pairs of PD(A) can be bijectively matched to ϵ-close points of PD(B) or to diagonal pairs (s, s), and vice versa. Here ϵ-closeness of pairs (a, c) and (b, d) in $\mathbb{R}^2$ is measured in the distance $L_\infty = \max\{|a - b|, |c - d|\}$.

The *mergegram* extends PD(A) to a stronger invariant whose stability under perturbations in the above sense is proved in section 5.5 for the first time. The idea of a mergegram is related to the Reeb graph [72] or the merge tree [68] for the sublevel set filtration of a scalar function. The

mergegram MG is defined at a more abstract level for any clustering dendrogram, which can be reconstructed from MG in general position.

Since any persistence diagram and a mergegram are unordered collections of pairs, the experiments in section 5.7 will use the neural network PersLay [17] whose output is invariant under permutations of input points by design. PersLay extends the neural network DeepSet [93] for unordered sets and introduces new layers to specifically handle persistence diagrams, as well as a new form of representing such permutation-invariant layers. In other related work deep learning was recently applied to outputs of hierarchical clustering [37], [21], [50] and to 0-dimensional persistence [22], [40].

5.3   Single-linkage clustering and mergegram

This section introduces dendrogram in Definition 5.3.2 and mergegram in Definition 5.3.4.

Example 5.3.1. Fig. 5.2 illustrates the key concepts before formal Definitions 2.4.4, 5.3.2, 5.3.4 for the point cloud $A = \{0, 1, 3, 7, 10\}$ in the real line $\mathbb{R}$. Imagine that we gradually blur original data points by growing disks of the same radius s around the given points.
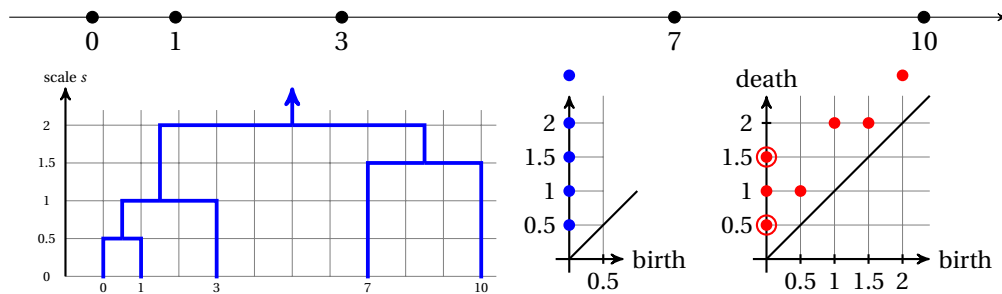

Figure 5.2: **Top**: the 5-point cloud $A = \{0, 1, 3, 7, 10\} \subset \mathbb{R}$. **Bottom** from left to right: single-linkage dendrogram $\Delta_{SL}(A)$ from Definition 2.4.4, the 0D persistence diagram PD from Definition 2.5.4 and the new mergegram MG from Definition 5.3.4, where double circles show pairs of multiplicity 2.

The disks of the closest points $0, 1$ start overlapping at the scale $s = 0.5$ when these points merge into one cluster $\{0, 1\}$. This merger is shown by blue arcs joining at the node at $s = 0.5$ in the single-linkage dendrogram, see the bottom left picture in Fig. 5.2. The persistence diagram PD in the bottom middle picture of Fig. 5.2 represents this merger by the pair $(0, 0.5)$ meaning that

a singleton cluster of (say) point 1 was born at the scale $s = 0$ and then died later at $s = 0.5$ by merging into another cluster of point 0.

When clusters $\{0, 1, 3\}$ and $\{7, 10\}$ merge at $s = 2$, this event was previously encoded in the persistence diagram by the single pair $(0, 2)$ meaning that one cluster inherited from (say) point 7 was born at $s = 0$ and died at $s = 2$. The new mergegram in the bottom right picture of Fig. 5.2 represents the above merger by the following two pairs. The pair $(1, 2)$ means that the cluster $\{0, 1, 3\}$ is merging at the current scale $s = 2$ and was previously formed at the smaller scale $s = 1$. The pair $(1.5, 2)$ means that another cluster $\{7, 10\}$ is merging at the scale $s = 2$ and was previously formed at $s = 1.5$.

The 0D persistence diagram represents the cluster of the whole cloud A by the pair $(0, +\infty)$, because A was inherited from a singleton cluster starting from $s = 0$. The mergegram represents the same cluster A by the pair $(2, +\infty)$, because A was formed during the last merger of $\{0, 1, 3\}$ and $\{7, 10\}$ at $s = 2$ and continues to live as $s \to +\infty$.

Definition 5.3.2 below extends a dendrogram Definition 2.4.2 to arbitrary (possibly, infinite) sets A. Since every partition of A is finite by Definition 2.4.1, we don't need to add that an initial partition of A is finite. Non-singleton sets are now allowed.

Definition 5.3.2 (dendrogram Δ of merge sets)**.** A *dendrogram* Δ over any set A is a function $\Delta : [0, +\infty) \to \mathbb{P}(A)$ of a scale $s \geq 0$ satisfying the following conditions.

(5.3.2a) There exists a scale $r \geq 0$ such that $\Delta(A; s)$ is the single block partition for $s \geq r$.

(5.3.2b) If $s \leq t$, then $\Delta(A; s)$ *refines* $\Delta(A; t)$, so any set from $\Delta(t)$ is a subset of some set from $\Delta(A; t)$. These inclusions of subsets of X induce the natural map $\Delta_s^t : \Delta(s) \to \Delta(t)$.

(5.3.2c) There are finitely many *merge scales* s_i such that

$$s_0 = 0 \text{ and } s_{i+1} = \sup\{s \mid \text{ the map } \Delta_s^t \text{ is identity for } s' \in [s_i, s)\}, i = 0, \ldots, m - 1.$$

Since $\Delta(A; s_i) \to \Delta(A; s_{i+1})$ is not an identity map, there is a subset $B \in \Delta(s_{i+1})$ whose preimage consists of at least two subsets from $\Delta(s_i)$. This subset $B \subset X$ is called a *merge* set and its *birth* scale is s_i. All sets of $\Delta(A; 0)$ are merge sets at the birth scale 0. The life(B) is the interval $[s_i, t)$ from its birth scale s_i to its *death* scale $t = \sup\{s \mid \Delta_{s_i}^s(B) = B\}$.   ∎

Dendrograms are usually drawn as trees whose nodes represent all sets from the partitions $\Delta(A; s_i)$ at merge scales. Edges of such a tree connect any set $B \in \Delta(A; s_i)$ with its preimages under $\Delta(A; s_i) \to \Delta(A; s_{i+1})$. Fig. 5.3 shows Δ for $A = \{0, 1, 2\}$.
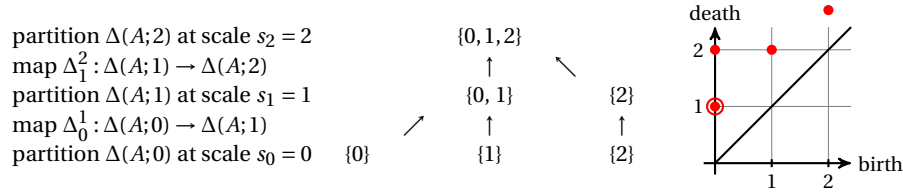
partition $\Delta(A;2)$ at scale $s_2 = 2$		$\{0,1,2\}$	
map $\Delta_1^2 : \Delta(A;1) \to \Delta(A;2)$		$\uparrow$	$\searrow$
partition $\Delta(A;1)$ at scale $s_1 = 1$		$\{0,1\}$	$\{2\}$
map $\Delta_0^1 : \Delta(A;0) \to \Delta(A;1)$	$\nearrow$	$\uparrow$	$\uparrow$
partition $\Delta(A;0)$ at scale $s_0 = 0$	$\{0\}$	$\{1\}$	$\{2\}$

Figure 5.3: The dendrogram Δ on $A = \{0,1,2\}$ and its mergegram MG(Δ) from Definition 5.3.4.

In Fig. 5.3 the partition $\Delta(A;1)$ consists of $\{0,1\}$ and $\{2\}$. The maps Δ_s^t induced by inclusions respect the compositions in the sense that $\Delta_s^t \circ \Delta_r^s = \Delta_r^t$ for any $r \le s \le t$. For example, $\Delta_0^1(\{0\}) = \{0,1\} = \Delta_0^1(\{1\})$ and $\Delta_0^1(\{2\}) = \{2\}$, so Δ_0^1 is a well-defined map from the partition $\Delta(A;0)$ of 3 singleton sets to $\Delta(A;1)$, but isn't an identity.

At the scale $s_0 = 0$ the merge sets $\{0\},\{1\}$ have life $= [0,1)$, while the merge set $\{2\}$ has life $= [0,2)$. At the scale $s_1 = 1$ the only merge set $\{0,1\}$ has life $= [1,2)$. At the scale $s_2 = 2$ the only merge set $\{0,1,2\}$ has life $= [2,+\infty)$. The notation Δ is motivated as the first (Greek) letter in the word dendrogram and by a Δ-shape of a typical tree.

Condition (5.3.2a) says that a partition of a set X is trivial for all large scales. Condition (5.3.2b) means that if the scale s is increasing, then sets from a partition $\Delta(s)$ can only merge but can not split. Condition (5.3.2c) implies that there are only finitely many mergers, when two or more subsets of X merge into a larger merge set.

Lemma 5.3.3. *[32, Lemma 3.3] Given a metric space (X,d) and a finite set $A \subset X$, the single-linkage dendrogram $\Delta_{SL}(X)$ from Definition 2.4.4 satisfies Definition 5.3.2.* ∎

A *mergegram* represents life spans of merge sets by pairs (birth, death) $\in \mathbb{R}^2$.

Definition 5.3.4 (mergegram MG(Δ))**.** The *mergegram* of a dendrogram Δ has the pair (birth, death) $\in \mathbb{R}^2$ for each merge set B of Δ with life(B) = [birth, death). If any life interval appears k times, the pair (birth,death) has the multiplicity k in MG(Δ). ∎

If our input is a point cloud A in a metric space, then the mergegram MG($\Delta_{SL}(A)$) is an isometry invariant of A, because $\Delta_{SL}(A)$ depends only on inter-point distances. Though $\Delta_{SL}(A)$ as any dendrogram is unstable under perturbations of points, the key advantage of MG($\Delta_{SL}(A)$) is its stability, which will be proved in Theorem 5.5.4.

The dendrogram Δ in the first picture of Fig. 5.5 generates the mergegram as follows:

- each of the singleton sets $\{b\}$, $\{c\}$, $\{p\}$, $\{q\}$ has pair $(0,1)$, so its multiplicity is 4;

- each of the merge sets $\{b,c\}$ and $\{p,q\}$ has the pair $(1,2)$, so its multiplicity is 2;

- the singleton set $\{a\}$ has the pair $(0,3)$; the merge set $\{b,c,p,q\}$ has the pair $(2,3)$;

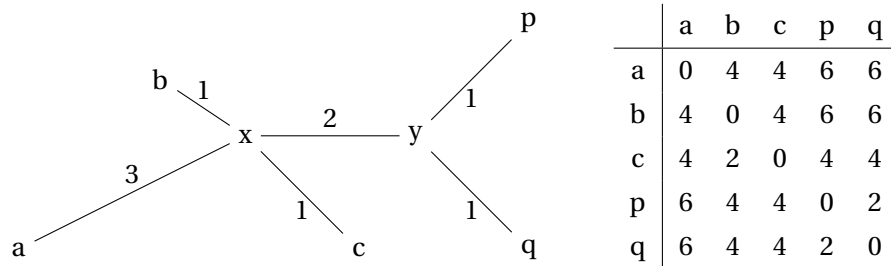- the full set $\{a,b,c,p,q\}$ continues to leave up to $s = 3$, hence has the pair $(3, +\infty)$.

	a	b	c	p	q
a	0	4	4	6	6
b	4	0	4	6	6
c	4	2	0	4	4
p	6	4	4	0	2
q	6	4	4	2	0

Figure 5.4: The set $X = \{a,b,c,d,e\}$ has the distance matrix defined by the shortest path metric.
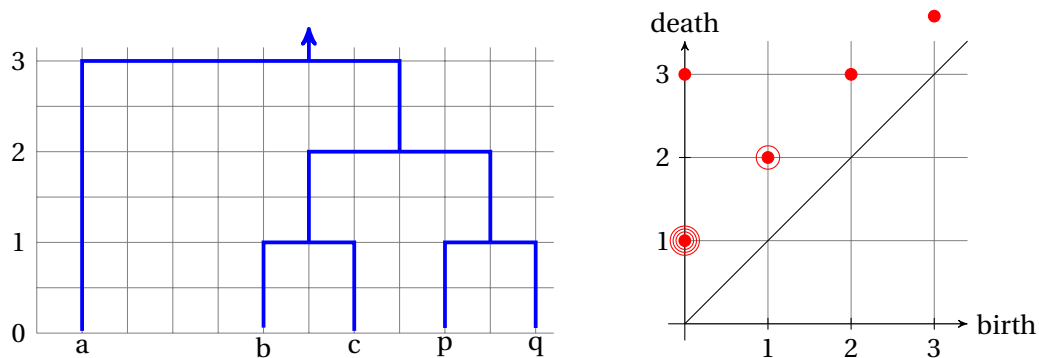
Figure 5.5: **Left**: the dendrogram Δ for the single linkage clustering of the set 5-point set $X = \{a,b,c,d,e\}$ in Fig. 5.4. **Right**: the mergegram $\mathrm{MG}(\Delta)$ with one pair $(0,1)$ of multiplicity 4.

5.4   Relations between 0-dim persistence and mergegram

This section recalls the concept of persistence and then shows how any 0D persistence and dendrogram in general position can be reconstructed from a mergegram.

Key results of this section are Theorem 5.4.1 which describes how the persistence diagram PD of the distance-based filtration of any point cloud A can be obtained from the mergegram

MG($\Delta_{\text{SL}}(S)$) and Theorem 5.4.2 which shows that in some cases dendrogram $\Delta_{\text{SL}}(S)$ can be reconstructed from mergegram MG($\Delta_{\text{SL}}(S)$).

Theorem 5.4.1. *[32, Theorem 5.3] For a finite set A in a metric space (X, d), let $d_A : X \to \mathbb{R}$ be the distance to A. Let the mergegram MG($\Delta_{SL}(A)$) be a multiset $\{(b_i, d_i)\}_{i=1}^{k}$, where some pairs can be repeated. Then the persistence diagram PD$\{H_0(X_s^{d_A})\}$ is the difference of the multisets $\{(0, d_i)\}_{i=1}^{k} - \{(0, b_i)\}_{i=1}^{k}$ containing each pair $(0, s)$ exactly $\#b - \#d$ times, where $\#b$ is the number of births $b_i = s$ and $\#d$ is the number of deaths $d_i = s$. All trivial pairs $(0, 0)$ are ignored, alternatively we take $\{(0, d_i)\}_{i=1}^{k}$ only with $d_i > 0$.* ∎

Theorem 5.4.1 is illustrated by Example 5.3.1, where $A = \{0, 1, 3, 7, 10\}$ has the diagram PD(A) = $\{(0, 0.5), (0, 1), (0, 1.5), (0, 2), (0, +\infty)\}$ obtained from the mergegram

$$\text{MG}(\Delta_{\text{SL}}(A)) = \{(0, 0.5), (0, 0.5), (0, 1), (0, 1.5), (0, 1.5), (0.5, 1), (1, 2), (1.5, 2), (2, +\infty)\}$$

as follows: one pair $(0, 0.5) \in$ PD(A) comes from two deaths and one birth $s = 0.5$ in MG($\Delta_{\text{SL}}(A)$). Similarly each of the pairs $(0, 1), (0, 1.5), (0, 2) \in$ PD(A) comes from two deaths and one birth equal to the same scale s. The cloud $B = \{0, 4, 6, 9, 10\} \subset \mathbb{R}$ in [32, Example 1.1] has exactly the same PD(B) = PD(A) but different MG($\Delta_{\text{SL}}(B)$) $\neq$ MG($\Delta_{\text{SL}}(A)$). This example together with Theorem 5.4.1 justify that the mergegram is strictly stronger than 0D persistence as an isometry invariant of a point cloud.

New Reconstruction Theorem 5.4.2 below can be contrasted with the weakness of 0D persistence PD$\{H_0(X_s^{d_A})\}$ consisting of only pairs $(0, s)$ whose finite deaths are half-lengths of edges in a Minimum Spanning Tree MST(A). In Example 5.3.1 these scales $s = 0.5, 1, 1.5, 2$ are insufficient to reconstruct the SL dendrogram in Fig. 5.2. Such a unique reconstruction is possible by using the richer invariant mergegram as follows.

Theorem 5.4.2 (from a mergegram to a dendrogram)**.** *Let A be a finite point cloud in* general position *in the sense that all merge scales of A in a dendrogram Δ from Definition 5.3.2 are different. Then the dendrogram Δ can be reconstructed from its mergegram MG(Δ), uniquely up to a permutation of nodes in Δ at scale $s = 0$.* ∎

Proof. Consider all merge scales one by one in the increasing order starting from the smallest. The general position implies that only two clusters merge at any merge scale. For any current scale s, the mergegram contains exactly two pairs (b_1, s) and (b_2, s).

For a smallest merge scale $s > 0$, the births should be $b_1 = b_2 = 0$. We start drawing a dendrogram Δ by merging any two points of A at this smallest scale s. To realize a merger at any larger s, we should select two clusters representing (b_1, s) and (b_2, s).

If $b_i = 0$ then we take any of the unmerged points of A. If $b_i > 0$ then the already constructed dendrogram should contain a unique non-singleton cluster determined by the scale $b_i \in (0, s)$. Hence at any merge scale s we know how to select two clusters to merge. The only choice comes from choosing points of A or permuting notes of Δ.   □

Following the above proof of Theorem 5.4.2 for the cloud $A = \{0, 1, 3, 7, 10\}$ in Example 5.3.1, the first two pairs $(0, 0.5) \in \mathrm{MG}(\Delta_{\mathrm{SL}}(A))$ indicate that we should merge two points of A at $s = 0.5$. The scale $s = 0.5$ uniquely determines this 2-point cluster.

The next two pairs $(0, 1), (0.5, 1)$ mean that the above cluster born at $s = 0.5$ should merge at $s = 1$ with a singleton cluster (any free point of A). The resulting 3-point cluster is uniquely determined by its merge scale $s = 1$. The further two pairs $(0, 1.5), (0, 1.5)$ say that a new 2-point cluster is formed at $s = 1.5$ by the two remaining points of A.

The final pairs $(1, 2), (1.5, 2)$ tell us to merge at $s = 2$ the two clusters formed earlier at $s = 1$ and $s = 1.5$. The resulting dendrogram Δ has the expected combinatorial structure as in Fig. 5.2, though we can draw Δ in another way by permuting points of A.

5.5   Stability of the mergegram

This section fully proves the stability of a mergegram in Theorem 5.5.4 with help of Lemmas 5.5.2 and 5.5.3. For simplicity, we consider vector spaces with coefficients only in $\mathbb{Z}_2 = \{0, 1\}$, which can be replaced by any field.

Definition 2.5.5 introduces homomorphisms between persistence modules, which are needed to state the stability of persistence diagrams $\mathrm{PD}\{H_0(X_s^f)\}$ under perturbations of a function $f : X \to \mathbb{R}$. This result will imply a stability for the mergegram $\mathrm{MG}(\Delta_{SL}(A))$ for the dendrogram $\Delta_{SL}(A)$ of the single-linkage clustering of a set $A \subset X$.

Definition 5.5.1 (merge module $M(\Delta)$)**.** For any dendrogam Δ on a finite set X, the *merge module* $M(\Delta)$ consists of the vector spaces $M_s(\Delta)$, $s \in \mathbb{R}$, and linear maps $m_s^t : M_s(\Delta) \to M_t(\Delta)$, $s \le t$. For any $s \in \mathbb{R}$ and $A \in \Delta(s)$, the space $M_s(\Delta)$ has the generator or a basis vector $[A] \in M_s(\Delta)$. For $s < t$ and any set $A \in \Delta(s)$, if the image of A under Δ_s^t coincides with $A \subset X$, so $\Delta_s^t(A) = A$, then $m_s^t([A]) = [A]$, else $m_s^t([A]) = 0$.   ∎

scale $s_3 = +\infty$	0					0
map $m_2^{+\infty}$	$\uparrow$					$\uparrow$
scale $s_2 = 2$	$\mathbb{Z}_2$			0	0	[{0,1,2}]
map m_1^2	$\uparrow$			$\uparrow$	$\uparrow$	
scale $s_1 = 1$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2$	0	0	[{2}]	[{0,1}]	
map m_0^1	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$		
scale $s_0 = 0$	$\mathbb{Z}_2 \oplus \mathbb{Z}_2 \oplus \mathbb{Z}_2$	[{0}]	[{1}]	[{2}]		

Figure 5.6: The merge module $M(\Delta)$ of the dendrogram Δ on the set $X = \{0,1,2\}$ in Fig. 5.3.

In a dendrogram Δ from Definition 5.3.2, any merge set A of Δ has life$(A) = [b,d)$ from its birth scale b to its death scale d. Lemmas 5.5.2 and 5.5.3 are proved for the first time.

Lemma 5.5.2 (merge module decomposition). *For any dendrogram Δ from Definition 5.3.2, the merge module $M(\Delta) \cong \bigoplus_A \mathbb{I}(\mathrm{life}(A))$ decomposes over all merge sets A.* ∎

Proof of Lemma 5.5.2. The goal is to prove that $M(\Delta) \cong \bigoplus_A \mathbb{I}(\mathrm{life}(A))$. Recall that the interval module $\mathbb{I}(\mathrm{life}(A))$ consists of only vector spaces 0 and $\mathbb{Z}_2$. For a scale r, let $\mathbb{I}_r(\mathrm{life}(A))$ be its vector space, whose generator is denoted by $[\mathbb{I}_r(\mathrm{life}(A))]$. Define

$$\psi_r : M_r(\Delta) \to \bigoplus_A \mathbb{I}_r(\mathrm{life}(A)) \text{ such that } [A] \to [\mathbb{I}_r(\mathrm{life}(A))] \text{ for all } A \in \Delta(r),$$

$$\phi_r : \bigoplus_A \mathbb{I}_r(\mathrm{life}(A)) \to M_r(\Delta) \text{ such that } [\mathbb{I}_r(\mathrm{life}(A))] \to [A] \text{ for all } \mathrm{life}(A) \text{ containing } r.$$

We will first prove that ϕ_r is well-defined. If $r \in \mathrm{life}(A)$ then $A \in M_r(\Delta)$. We know that $M_r(\Delta)$ is generated by elements $A \in \Delta(r)$ for which $r \in \mathrm{life}(A)$. Thus the compositions satisfy $\phi_r \circ \psi_r = \mathrm{id}_r$ and $\psi_r \circ \phi_r = \mathrm{id}_r$. It remains to prove that morphisms correctly behave under the functors ψ, ϕ. The proofs for both cases are essentially the same, thus we will prove it only for ψ. The goal is to prove that the following diagram commutes:

$$
\begin{array}{ccc}
M_s(\Delta) & \xrightarrow{\ m_s^t\ } & M_t(\Delta) \\
\psi_s \downarrow & & \downarrow \psi_t \\
\bigoplus_A \mathbb{I}_s(\mathrm{life}(A)) & \xrightarrow{\ i_s^t\ } & \bigoplus_A \mathbb{I}_t(\mathrm{life}(A))
\end{array}
$$

Here i_s^t is the direct sum of the corresponding maps of interval modules $\bigoplus_A (i_s^t)^A$. Let $[A]$ be an arbitrary generator of $M_r(\Delta)$. There are two possibilities how m_s^t can map $[A]$. If $t \in \mathrm{life}(A)$, then

$m_s^t([A]) = [A] \in M_t(\triangle)$ and by definition

$$\phi_t \circ m_s^t([A]) = [\mathbb{I}_t(\text{life}(A))].$$

Since both $s, t \in \text{life}(A)$, we also have that

$$m_s^t \circ \phi_t([A]) = [\mathbb{I}_t(\text{life}(A))] = \phi_t \circ m_s^t([A]).$$

Assume now that $t \notin \text{life}(A)$. Then $m_s^t([A]) = 0$ and thus $\phi_t(m_s^t([A])) = 0$. On the other hand $i_s^t \circ \phi_s([A]) = [\mathbb{I}_t(\text{life}(A))] = \mathbb{Z}_2$. Since $t \notin \text{life}(A)$, we get $i_s^t \circ \phi_s([A]) = 0$. □

Lemma 5.5.3 (merge modules interleaved). *If subsets A, B of a metric space (X, d) have $\text{HD}(A, B) = \delta$, then the merge modules $M(\Delta_{SL}(A)), M(\Delta_{SL}(B))$ are δ-interleaved.* ∎

Proof of Lemma 5.5.3. The equality $\text{HD}(A, B) = \delta$ means that A is covered by the union of closed balls that have the radius δ and centers at all points $b \in B$. This union is the preimage is $d_B^{-1}([0, \delta])$, i.e. $A \subset d_B^{-1}([0, \delta])$. Extending the distance values by $s \geq 0$, we get $d_A^{-1}([0, s]) \subset d_B^{-1}([0, s + \delta])$ and similarly $d_B^{-1}([0, s]) \subset d_A^{-1}([0, s + \delta])$.

Let U be an arbitrary set in $\Delta_{SL}(A)$. Define map $\phi_r : M(A; r) \to M(B; r + \delta)$

$$\phi_r([U]) = \begin{cases} [U], & \text{if } r + \delta \in \text{life}(\Delta_{SL}(B), U), \\ 0, & \text{otherwise}. \end{cases}$$

Symmetrically for any $V \in \Delta_{SL}(B)$ we define $\psi_r : M(B; r) \to M(A; r + \delta)$

$$\psi_r([V]) = \begin{cases} [V], & \text{if } r + \delta \in \text{life}(\Delta_{SL}(A), V), \\ 0, & \text{otherwise}. \end{cases}$$

In the notation above, $\text{life}(\Delta_{SL}(B), U)$ is the $\text{life}(U)$ in the dendrogram $\Delta_{SL}(B)$. If $U \notin \Delta_{SL}(B)(t)$ for all values t, then $\text{life}(U) = \emptyset$. By symmetry it is enough to prove that the following diagrams commute:

$$
\begin{array}{ccc}
M_s(\Delta_{SL}(A)) & \xrightarrow{\;\; m_s^t \;\;} & M_t(\Delta_{SL}(A)) \\
\phi_s \downarrow & & \downarrow \phi_t \\
M_{s+\delta}(\Delta_{SL}(B)) & \xrightarrow{\;\; m_{s+\delta}^{t+\delta} \;\;} & M_{t+\delta}(\Delta_{SL}(B))
\end{array}
$$

$$M_s(\Delta_{SL}(B))$$

$$\phi_{s-\delta} \nearrow \qquad \searrow \psi_s$$

$$M_{s-\delta}(\Delta_{SL}(A)) \xrightarrow{\quad m_{s-\delta}^{s+\delta} \quad} M_{s+\delta}(\Delta_{SL}(A))$$

We note first that if $[a,b) = (\text{life}(\Delta_{SL}(A),U)$, then $(\text{life}(\Delta_{SL}(B),U) \subseteq [a-\epsilon,b+\epsilon)$

We begin by proving commutativity of the first diagram. Let U be arbitrary element of $\Delta_{SL}(A)(s)$. If $t \notin \text{life}(\Delta_{SL}(A),U)$ then $\phi_t \circ m_s^t = 0$. If $s+\delta \notin \text{life}(\Delta_{SL}(B),U)$ or $t+\delta \notin \text{life}(\Delta_{SL}(B),U)$ then we are done. Since $t \notin \text{life}(\Delta_{SL}(A),U)$, it follows that $t+\delta \notin \text{life}(\Delta_{SL}(B),U)$. And thus with given assumptions the diagram commutes.

Assume now that $t+\delta \notin \text{life}(\Delta_{SL}(A),U)$. Then both $\phi_t(m_s^t(U)) = 0 = m_{s+\delta}^{t+\delta}(\phi_s(U))$. In the last case we assume that $t \in \text{life}(\Delta_{SL}(A),U)$ and $t+\delta \in \text{life}(\Delta_{SL}(B),U)$. In this case obviously $s+\delta \in \text{life}(\Delta_{SL}(B),U)$ and thus $\phi_t(m_s^t([U])) = [U] = m_{s+\delta}^{t+\delta}(\phi_s([U]))$.

For the second diagram assume now that $U \in M(\Delta_{SL}(A))(s-\delta)$. Assume first that $s \notin \text{life}(\Delta_{SL}(B),U)$, then $s+\delta \notin \text{life}(\Delta_{SL}(B),U)$ and $m_{s-\delta}^{s+\delta}([U]) = 0 = \psi_s(\phi_{s-\delta}[U])$.

Assume then that $s \in \text{life}(\Delta_{SL}(B),U)$. Now outcome of both maps ψ_s and $m_{s-\delta}^{s+\delta}$ depend on if $s+\delta \in \text{life}(\Delta_{SL}(A),U)$ and thus $m_{s-\delta}^{s+\delta}([U]) = \psi_s(\phi_{s-\delta}[U])$. Since all the diagrams commute, the required conclusion follows. $\qquad\square$

Theorem 5.5.4 (stability of a mergegram). *The mergegrams of any finite point clouds A,B in a metric space (X,d) satisfy* $\text{BD}(\text{MG}(\Delta_{SL}(A)),\text{MG}(\Delta_{SL}(B)) \leq \text{HD}(A,B)$. *Hence any small perturbation of a cloud A in the Hausdorff distance yields a similarly small perturbation in the bottleneck distance for its mergegram* $\text{MG}(\Delta_{SL}(A))$. $\qquad\blacksquare$

Proof. The given clouds $A,B \subset X$ with $\text{HD}(A,B) = \delta$ have δ-interleaved merge modules by Lemma 5.5.3, so $\text{ID}(\text{MG}(\Delta_{SL}(A)),\text{MG}(\Delta_{SL}(B)) \leq \delta$. Since any merge module $M(\Delta)$ is finitely decomposable, hence is q-tame by Lemma 5.5.2. The corresponding mergegram $\text{MG}(M(\Delta))$ satisfies Theorem 2.5.8, so $\text{BD}(\text{MG}(\Delta_{SL}(A)),\text{MG}(\Delta_{SL}(B)) \leq \delta$. $\qquad\square$

Fig. 5.7 illustrates Theorem 5.5.4 on a cloud and its perturbation by showing their close mergegrams. The more extensive experiment on 100 clouds in [32, Fig. 8] similarly confirms that the mergegram is perturbed within expected bounds. The computational complexity of the mergegram $\text{MG}(\Delta_{SL}(A))$ was proved to be near linear in the number n of points in a cloud $A \subset \mathbb{R}^m$, see [32, Theorem 8.2]. The results above justify that the invariant mergegram satisfies conditions (a,b,c) of Isometry Recognition Problem 5.1.1.
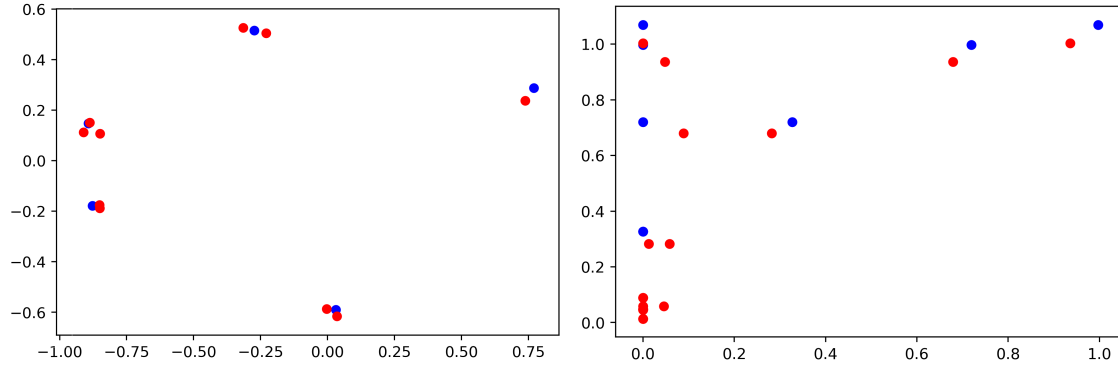
Figure 5.7: **Left**: the cloud C of 5 blue points is close to the cloud C' of 10 red points in the Hausdorff distance. **Right**: the mergegrams are close in the bottleneck distance as predicted by Theorem 5.5.4.

Algorithm 5.5.5 computes the mergegram of the SL dendrogram for any finite set $R \subset \mathbb{R}^m$. Let $\alpha(|R|)$ be the inverse Ackermann function. Other constants below are defined in [63].

Algorithm 5.5.5 This algorithm takes a finite metric space (R, d) as input and computes its mergegram, which is returned as the output.

1: **Input** : a finite point cloud $R \subset \mathbb{R}^m$
2: Compute MST(R) and sort all edges of MST(R) in increasing order of length
3: Initialize Union-Find structure U over R. Set all points of R to be their components.
4: Initialize the function prev: Components$[U] \to \mathbb{R}$ by setting prev$(t) = 0$ for all t
5: Initialize the vector Output that will consists of pairs in $\mathbb{R} \times \mathbb{R}$
6: **for** Edge $e = (a, b)$ in the set of edges (increasing order) **do**
7:     Find components c_1 and c_2 of a and b respectively in Union-Find U
8:     Add pairs (prev$[c_1]$,length(e)), (prev$[c_2]$,length(e)) $\in \mathbb{R}^2$ to Output
9:     Merge components c_1 and c_2 in Union-Find U and denote the component by t
10:    Set prev$[t] = $ length(e)
11: **end for**
12: **return** Output

Theorem 5.5.6 (a fast mergegram computation). *For any cloud $R \subset \mathbb{R}^m$, the mergegram* $\mathrm{MG}(\Delta_{SL}(R))$ *can be computed in time*

$$O\Big((c_m(R))^{4+\lceil \log_2(\rho(R)) \rceil} \cdot \log_2(\Delta(|R|)) \cdot |R| \cdot \log_2(|R|) \cdot \alpha(|R|)\Big),$$

where all parameters were introduced in Definitions 3.1.4, 3.2.14, 4.4.11.

Proof. By Theorem 4.4.15 the Minimum Spanning Tree MST(R) needs

$$O\Big((c_m(R))^{4+\lceil\log_2(\rho(R))\rceil}\cdot\log_2(\Delta(|R|))\cdot|R|\cdot\log_2(|R|)\cdot\alpha(|R|)\Big).$$

The rest of Algorithm 5.5.5 is dominated by $O(|R|\alpha(|R|))$ Union-Find operations. Hence the full algorithm has the same computational complexity as the MST.   □

5.6 Experiments on a classification of point sets

The experiments summarized in Fig. 5.9 show that the mergegram curve in blue outperforms other isometry invariants on the isometry classification by the state-of-the-art PersLay. We generated 10 classes of 100-point clouds within the unit ball $\mathbb{R}^m$ for $m = 2, 3, 4, 5$. For each class, we made 100 copies of each cloud and perturbed every point by a uniform random shift in a cube of the size $2 \times \epsilon$, where ϵ is called a *noise bound*. For each of 100 perturbed clouds, we added 25 points such that every new point is ϵ-close to an original point. Within each of 10 classes all 100 clouds were randomly rotated within the unit ball around the origin, see Fig. 5.8. For each of the resulting 1000 clouds, we computed the mergegram, 0D persistence diagram and the diagram of pairs of distances to two nearest neighbors for every point.
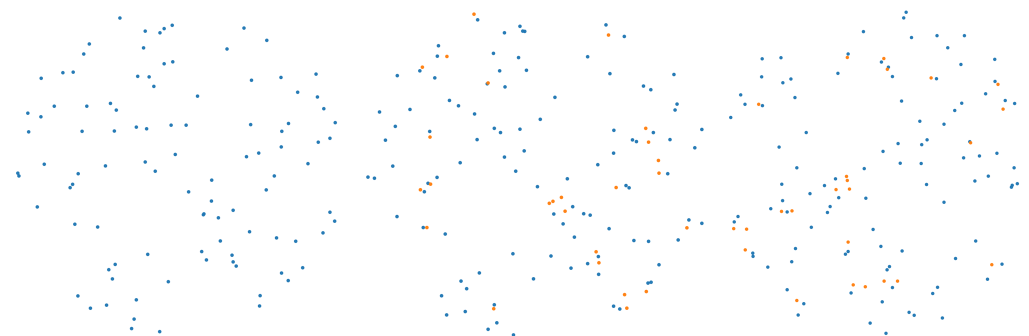

Figure 5.8: **Left**: an initial random cloud with 100 blue points. **Middle**: all blue points are perturbed, 25 extra orange points are added. **Right**: a cloud is rotated through a random angle. Can we recognize that the initial and final clouds are in the same isometry class modulo small noise?

The machine learning part has used the obtained diagrams as the input-data for the Perslay [17]. Each dataset was split into learning and test subsets in ratio 4:1. The learning loops ran by iterating over mini-batches consisting of 128 elements and going through the full dataset for a

given number of epochs. The success rate was measured on the test subset.

The original Perslay module was rewritten in Tensorflow v2 and RTX 2080 graphics card was used to run the experiments. The technical concepts of PersLay are explained in [17]:

- Adam(Epochs = 300, Learning rate = 0.01)

- Coefficents = Linear coefficents

- Functional layer = [PeL(dim=50), PeL(dim=50, operationalLayer=PermutationMaxLayer)].

- Operation layer = TopK(50)

The PersLay training has used the following invariants compared in Fig. 5.9:

- cloud : the initial cloud A of points corresponds to the baseline curve in black;

- PD0: the 0D persistence diagram PD for distance-based filtrations of sublevel sets in red;

- NN(2) brown curve: for each point $a \in A$ includes distances to two nearest neighbors;

- the mergegram $MG(\Delta_{SL}(A))$ of the SL dendrogram has the blue curve above others.

Fig. 5.9 shows that the new mergegram has outperformed all other invariants on the isometry classification problem. The 0D persistence turned out to be weaker than the pairs of distances to two neighbors. The topological persistence has found applications in data skeletonization with theoretical guarantees [55, 49]. We are planning to extend the experiments in section 5.7 for classifying rigid shapes by comining the new mergegram with the 1D persistence, which has the fast $O(n \log n)$ time for any 2D cloud of n points [57, 56].
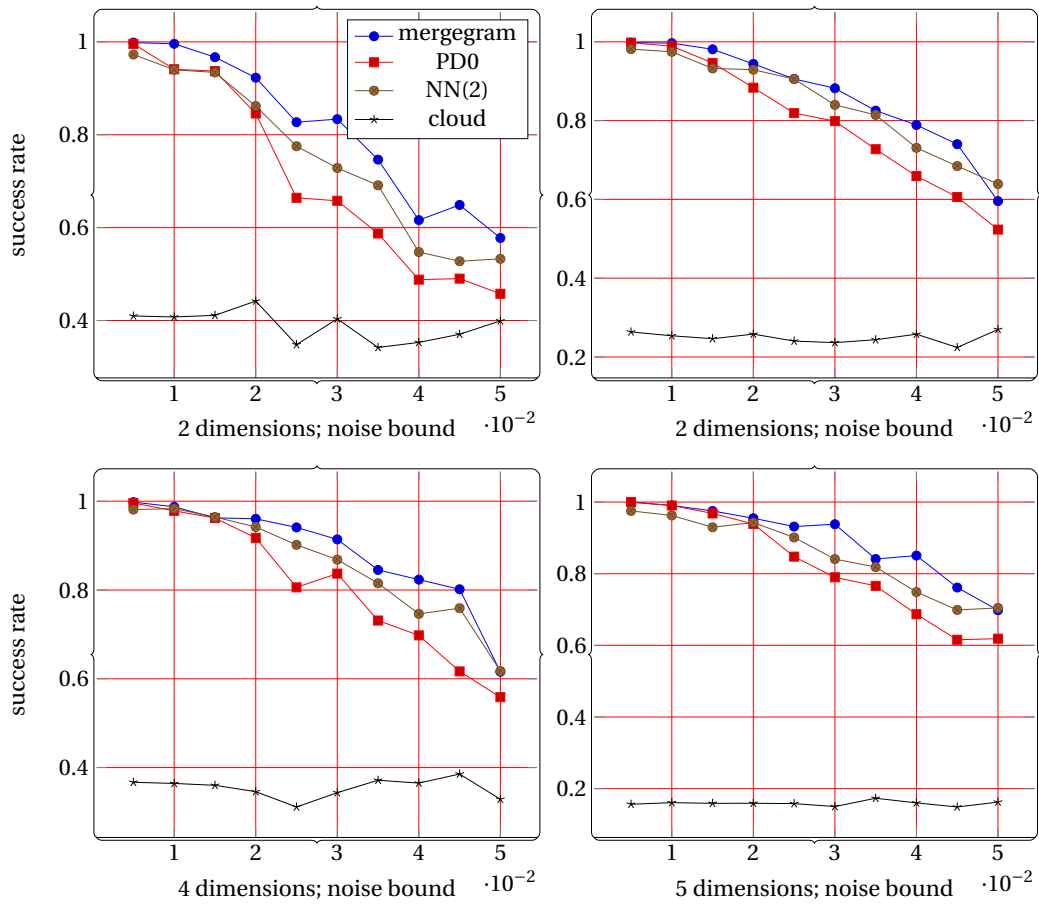
Figure 5.9: Success rates of PersLay in identifying isometry classes of 100-point clouds uniformly sampled in a unit ball, averaged over 5 different clouds and 5 cross-validations with 20/80 splits.

5.7   New experiments on isometry recognition of distorted shapes

This section fulfills final condition (d) of Problem 5.1.1 by experimentally comparing the merge-gram with 0D persistence and distributions of distances to neighbors on 15000 clouds. Section 5.6 did experiments only on randomly generated clouds.

We considered 15 classes of shapes represented by black and white images of mythical creatures [13], see Fig. 5.1. These shapes were chosen to make the shape recognition problem really challenging. Indeed, similar creatures from this dataset are represented by slightly different shapes, which can be hard to isometrically distinguish from each other. For example, several images of a horse include only minor differentiating features such as a saddle or a different tails, which makes horses nearly identical.

Shape generation. For each image, we generated 1000 perturbed images by affine and projective transformations to get 15000 distorted shapes split into 15 classes.

First we rotated each image around its central point by an angle generated uniformly in the interval $[0, 2\pi)$ using the function cv::rotate from the OpenCV library. If needed, we extended the resulting image to fit all black pixels of the rotated shape into a bounding box. Then both affine and projective transformations distort each image by using a noise parameter δ such that the value $\delta = 0$ represents the identity transformation.

Fig. 5.10 illustrates how an original image is randomly rotated, then randomly distorted by affine or projective transformations depending on the noise parameter δ.

Affine transformations are implemented as compositions of the already applied rotations above and the function cv::resize() from the OpenCV library. This function scales an image of size $w \times h$ by horizontal and vertical factors a, b sampled as follows.

• **Uniform noise**: $a \in [1 - \delta w, 1 + \delta w]$, $b \in [1 - \delta h, 1 + \delta h]$ have uniform distributions.

• **Gaussian noise**: $a \in \mathcal{N}(1, \delta h) \cap \mathbb{R}_+$ and $b \in \mathcal{N}(1, \delta w) \cap \mathbb{R}_+$ have Gaussian distributions with mean 1 and standard variance $\delta h, \delta v$, truncated to positive numbers.

Projective transformation are implemented as compositions of the already applied rotations above and the OpenCV function cv::getPerspectiveTransform() function, which is parametrized by 4-dimensional array $v = (a_0, a_1, a_2, a_3)$ consisting of points $a_i \in \mathbb{Z}^2$, $i = 0, 1, 2, 3$. This function maps the corners of the image as follows:

$$(0,0) \mapsto a_0, (0, h) \mapsto a_1, (w, 0) \mapsto a_2 \text{ and } (w, h) \mapsto a_3.$$

(3a) Affine transformation

(1) Original centaur   (2) Rotated centaur

(3b) Projective transformation

(3a) Affine transformation

(1) Original man   (2) Rotated man

(3b) Projective transformation

(3a) Affine transformation

(1) Original horse   (2) Rotated horse
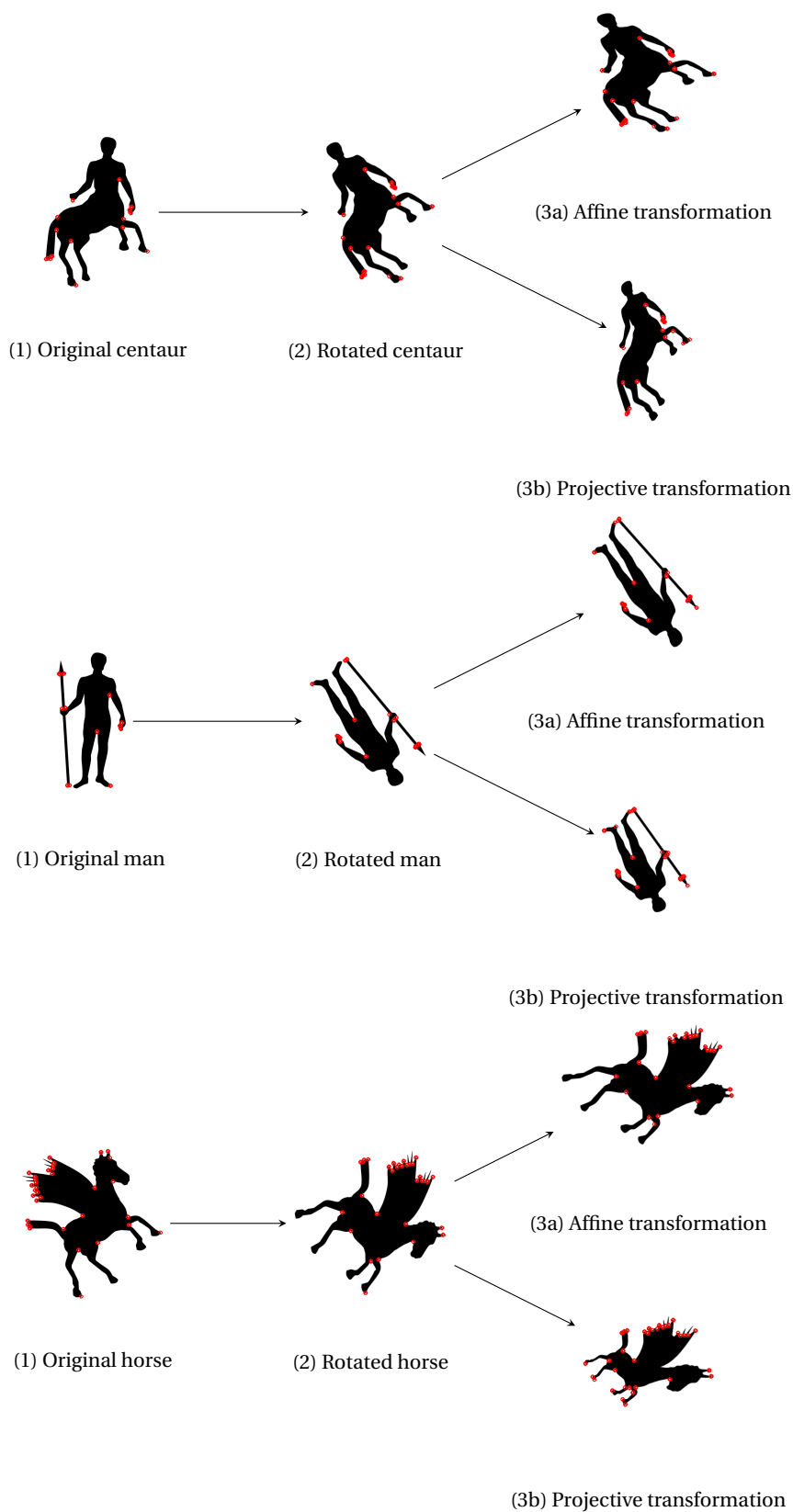
(3b) Projective transformation

Figure 5.10: Generating distorted shapes by applying random rotations, affine and projective transformations, which substantially affect the extracted clouds of Harris corner points [81] in red.

Then the projective transformation of the rectangle $w \times h$ is uniquely determined by the above corners. The above points a_i are randomly sampled by using a noise parameter δ.

• **Uniform noise**: each coordinate has a uniform distribution with a noise parameter δ

$$a_0 \in [0, \delta w] \times [0, \delta h], \qquad a_1 \in [0, \delta w] \times [h - \delta h, h],$$

$$a_2 \in [w - \delta w, w] \times [0, \delta h], \qquad a_3 \in [w - \delta w, w] \times [h - \delta h, h].$$

• **Gaussian noise**: each coordinate has a Gaussian distribution truncated to the image

$$a_0 \in (\mathcal{N}(0, \delta w) \cap [0, w]) \times (\mathcal{N}(0, \delta h) \cap [0, h]),$$

$$a_1 \in (\mathcal{N}(0, \delta w) \cap [0, w]) \times (\mathcal{N}(h, \delta h) \cap [0, h]),$$

$$a_2 \in (\mathcal{N}(w, \delta w) \cap [0, w]) \times (\mathcal{N}(0, \delta h) \cap [0, h]),$$

$$a_3 \in (\mathcal{N}(w, \delta w) \cap [0, w]) \times (\mathcal{N}(h, \delta h) \cap [0, h]).$$

Point cloud extraction. For each distorted image, we extract classical Harris point corners [81] due to their simplicity, see the red points in Fig. 5.10. For detecting corner points, the OpenCV function cv::cornerHarris was used with the parameters blockSize = 3, apertureSize = 5, k = 0.04, thresh = 120. However one can use any reliable algorithms such as FAST [79] or scale-invariant feature transform (SIFT) [61].

After describing the available point cloud data above, we specify condition (5.1.1d) of Isometry Recognition Problem 5.1.1 in the context of supervised machine learning.

Proposition 5.7.1 (experimental recognition)**.** *Given a labeled dataset split into classes of similar but projectively distorted shapes, develop a supervised learning tool to recognize a class of distorted shapes with a high accuracy despite substantial noise.*

Since all isometry invariants are independent of point ordering, the most suitable neural network is PersLay [17] whose output is invariant under permutations by design. Each layer is a combination of a coefficient layer $\omega(p) : \mathbb{R}^m \to \mathbb{R}$, a transformation $\phi(p) : \mathbb{R}^m \to \mathbb{R}^q$ and a permutation invariant layer op combined as follows

$$\text{PersLay}(D) = \text{op}(\{w(p)\phi(p)\}_{p \in D}), \text{ where } D \text{ is a diagram or multiset of points in } \mathbb{R}^m.$$

Coordinates of all input points are linearly normalized to $[0,1]$. We have used the following parameters of the PersLay network for all experiments below.

The max layer $\mathrm{MAX}(q)$ consists of the following functions.

- The coefficient layer $w : \mathbb{R}^m \to \mathbb{R}$ is the weight $w(x_1, \ldots, x_m) = k|x_1 - x_2|$, where k is a trainable scalar and the dimension is typically $m = 2$.

- The transformation layer $\phi : \{\text{diagrams of points in } \mathbb{R}^m\} \to \mathbb{R}^q$ is the function $\phi(D) = \sum_{p \in D} \lambda p + \gamma \mathrm{maxpool}(D) + \beta$, where λ, γ are $\mathbb{R}^{m \times q}$ trainable matrices, β is a $\mathbb{R}^q$ trainable vector and maxpool returns a maximum value for every $i = 1, \ldots, m$.

- The operational layer $\mathrm{op} : \mathbb{R}^q \to \mathbb{R}^t$ puts all coordinates in increasing order and composes the result with standard densely connected layer [35] $\mathrm{Dense} : \mathbb{R}^q \to \mathbb{R}^t$.

An output is a vector in $\mathbb{R}^t$ for $t = 15$ of image classes. A final prediction is obtained by choosing a class with a largest coordinate in the output vector.

The image layer $\mathrm{Im}[x, y]$ for integer parameters x, y and a multiset of points in the unit square $[0,1]^2$ consists of the following functions.

- The coefficient layer $w : \mathbb{R}^2 \to \mathbb{R}$ is a piecewise constant function trained on $x \cdot y$ parameters, defined on the unit square partition

$$\mathcal{P}(x, y) = \left\{ \left[\frac{i}{x}, \frac{i+1}{x} \right] \times \left[\frac{j}{y}, \frac{j+1}{y} \right] \mid i = 0, \ldots, x-1 \text{ and } j = 0, \ldots, y-1 \right\}.$$

- Let $\phi_p : \mathbb{R}^2 \to \mathbb{R}$ be the Gaussian distribution centered at $p \in D$ with a trainable standard deviation σ. The transformation layer $\phi : \mathbb{R} \to \mathbb{R}^{xy}$ consists of xy functions ϕ_p, where p runs over all centroids of the partition $\mathcal{P}(x, y)$.

- The operation layer op takes the sum over the given point cloud. A final prediction is made by composing the operation layer with the Dense layer.

Finally, the PersLay network used the optimizer tf.keras.adam with the standard learning rate 0.01 and 150 epochs, the loss function SparseCategoricalCrossEntropy, the 80:20 of training and testing, a 5-fold Monte Carlo cross validation for each run.

Fig. 5.11, 5.12, 5.13 show that the mergegram MG consistently outperforms two other isometry invariants: 0D persistence and the multiset $NN(4)$ consisting of 4-tuple distances to neighbors

per given point. The simpler multiset $NN(2)$ performed worse. A given cloud $C \subset \mathbb{R}^2$ was considered as a baseline input. The noise factors δ reached 25%, which means that original images were distorted up to a quarter of image sizes.

5.8 A discussion of novel contributions and further open problems

This chapter has further demonstrated that the provably stable-under-noise invariant merge-gram of a dendrogram is a fast and efficient tool in the challenging problem of isometry shape recognition, especially for substantially distorted images.

In comparison with the conference version [32], section 5.4 proved new Theorem 5.4.2 describing how to reconstruct a single-linkage dendrogram in general position from its much simpler mergegram. It is hard to define a continuous metric between dendrograms, especially because they can be unstable under perturbations. Theorem 5.4.2 allows us to measure a continuous similarity between dendrograms in general position as the bottleneck distance between their unique mergegrams. This distance can be computed in time $O(n^{1.5} \log n)$ [51] for diagrams consisting of at most n points.

Section 5.5 provided a full proof of stability of the mergegram under perturbations of points, while the earlier paper [32] only announced this result without proving highly non-trivial Lemmas 5.5.2 and 5.5.3, which required a heavy algebraic machinery.
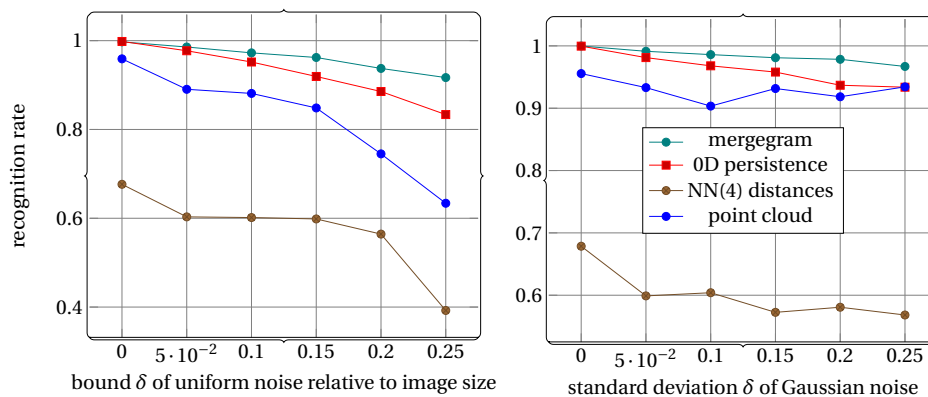

Figure 5.11: Recognition rates are obtained by training the max layer MAX(75) of PersLay on three isometry invariants and a cloud of corner points extracted from 15000 affinely distorted images.
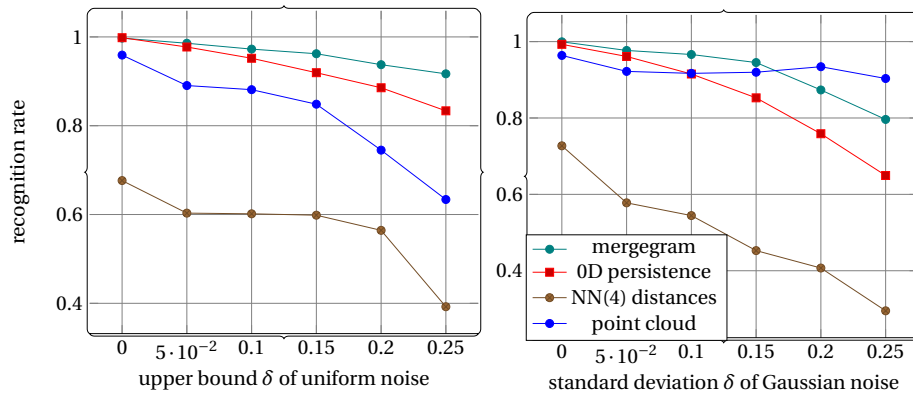
Figure 5.12: Recognition rates are obtained by training the max layer MAX(75) of PersLay on isometry invariants and corner points extracted from 15000 projectively distorted images.
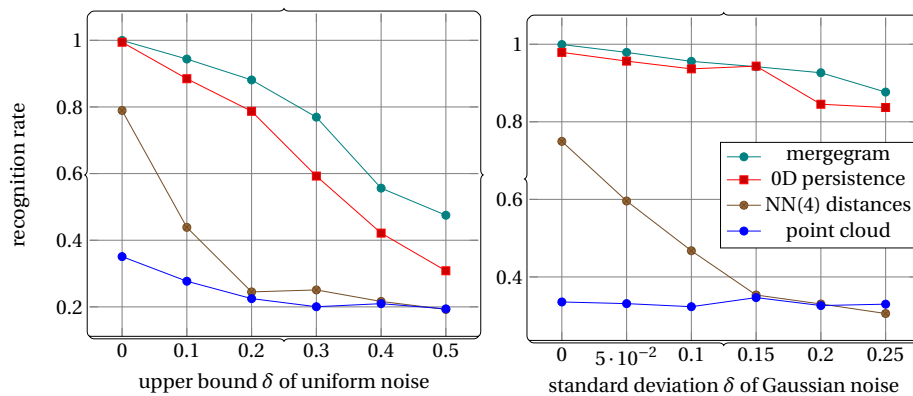

Figure 5.13: Recognition rates are obtained by training the image layer IM[20,20] of PersLay on isometry invariants and a cloud of corner points extracted from 15000 affinely distorted images.

Example 5.3.1 and the discussion following Theorem 5.4.1 justify that the invariant mergegram is strictly stronger than the 0D persistence. This theoretical fact is now confirmed by the new experiments on 15000 point clouds extracted from substantially distorted real shapes. In Fig. 5.11, 5.12, 5.13 the mergegram outperformed other isometry invariants. Since the distribution $NN(2)$ of distances to two closest neighbors per point performed badly, we have strengthened this invariant to $NN(4)$ of distances to four nearest neighbors. However, even $NN(4)$ performed always always worse than the original point cloud, which can not be considered as an isometry invariant.

For very high levels of 20% and 25% distortions in projective transformations, the PersLay network

trained on a point cloud achieved high recognition rates, because we have extensively tried many parameters in the layers MAX(75) and Im[20,20] for a best trade-off between accuracy and speed. The C++ code for the mergegram is at [32].

Chapter 6

Conclusion and future research work

The main accomplishments of this thesis are the following.

Chapter 3 is dedicated to the k-nearest neighborhood problem, Problem 3.1.2. In this chapter, we provide a concrete set of counterexamples that illustrates that the time complexity estimates of the past approaches based on cover tree data structure [9, 78] were proven incorrectly. To overcome past challenges we introduce a new data structure, a compressed cover tree, as well as a new algorithm for the k-nearest neighborhood search. At the end of the chapter, we rigorously demonstrate that the new approach resolved Problem 3.1.2 in $O(k \cdot \log(k) \cdot \log(R) \cdot |R|)$ time, parametrized by expansion constant $c(R)$ of Definition 3.1.4.

Chapter 4 is dedicated to a minimum spanning tree problem of any finite metric space. It is shown that in the past approach [63] the time complexity result that claimed the parametrized $O(|R| \cdot \log(|R|)$ time complexity was proven incorrectly. In this chapter, we present a new simpler algorithm based on a compressed cover tree. In the end the problem gets resolved with a weaker $O(|R| \cdot \log(|R|) \cdot \log(\Delta(R)))$ parametrized time complexity estimate.

Chapter 5 is dedicated to a new object, mergegram. It is confirmed that mergegram is stable under perturbations of data set R and contains more information than standard 0-dimensional persistence. In the experimental part of the chapter, we apply mergegram to the recognition of distorted 2d shapes - problem. It is experimentally verified that mergegram produces excellent results in 2d shape recognition when it is combined with Perslay-neural network architecture.

As no work is ever complete it is important to highlight a few perspective research directions:

Paired-trees: Worst-time complexities for k-nearest neighbors and the minimum spanning tree problem can be potentially improved by utilizing cover trees of both query and reference sets.

Given query and reference sets Q, R we wish to develop an algorithm that scans through all relevant pairs of nodes in $\mathcal{T}(Q) \times \mathcal{T}(R)$ in time $O(c^{O(1)} \cdot \max\{|Q|, |R|\})$, where c is an expansion constant that depends on sets Q, R. In [78, Algorithm 1] such algorithm was developed, but Counterexample 3.9.5 exposes its time complexity estimate was incorrect. In [26] a more general approach was adopted to obtain a time complexity bound for the paired traversal of $(\mathcal{T}(R), \mathcal{T}(Q))$. However the time complexity estimate of the provided algorithm depended on two extra parameters: imbalance I and extra-reference recursions θ. No proper time complexity estimate was provided for the imbalance I. This imbalance I can be trivially bounded by $O(|R| \cdot |Q|)$. However this is not sufficient to achieve parametrized worst-time complexity $O(|R|)$. In the future we wish to apply ideas from past approaches [78, 26] to generate a new algorithm guaranteeing $O(c^{O(1)} \cdot \max\{|Q|, |R|\})$ complexity for the paired-tree traversal.

Improving the minimum spanning tree algorithm: To improve minimum spanning tree algorithm we can first consider the following smallest-pair problem: given a quotient space $R/\sim$ and compressed cover tree $\mathcal{T}(R)$, find pair (a, b) such that a, b belong to distinct components of $R/\sim$ and minimize the distance $d(a, b)$. This could be potentially done by augmented paired tree traversal of $\mathcal{T}(R) \times \mathcal{T}(R)$. It should be noted that this task is significantly easier than finding all non-trivial nearest neighbors of the set R. Using results, similar to Lemma 3.5.9 and Lemma 3.7.13 it might be realistic to prove that such a task could be completed in $O(c(R)^{O(1)} \cdot \log_2(|R|))$ complexity. We can then exploit the idea of Kruskal's algorithm i.e. we complete $|R|$ iteration, in each of which we modify quotient space $R/\sim$ by merging two closest pairs belonging to distinct components. This algorithm could potentially achieve $O(c(R)^{O(1)} \cdot |R| \cdot \log_2(|R|) \cdot \alpha(|R|))$ worst-case time complexity bound for the construction of a minimum spanning tree on a finite metric space (R, d).

Multiparameter mergegrams By taking advantage of multiparameter dendrograms [16] it is possible to consider multiparameter mergegrams. It will be beneficial to explore different definitions of multiparameter mergegrams and metrics between them. One of the most interesting theoretical problems in multiparamter dendrograms in our view is to find out under which metrics and conditions multiparameter dendrograms would obtain a stability property similar to Theorem 5.5.4.

Bibliography

[1] Alexandr Andoni, Piotr Indyk, and Ilya Razenshteyn. Approximate nearest neighbor search in high dimensions. In *Proceedings of the International Congress of Mathematicians: Rio de Janeiro 2018*, pages 3287–3318. World Scientific, 2018.

[2] Sunil Arya and David M Mount. Approximate nearest neighbor queries in fixed dimensions. In *SODA*, volume 93, pages 271–280, 1993.

[3] Sunil Arya and David M Mount. A fast and simple algorithm for computing approximate euclidean minimum spanning trees. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1220–1233. SIAM, 2016.

[4] Ulrich Bauer. Ripser: efficient computation of vietoris–rips persistence barcodes. *Journal of Applied and Computational Topology*, 5(3):391–423, 2021.

[5] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, pages 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] Jon Louis 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*, pages 28–39, 1996.

[9] Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *Proceedings of ICML*, pages 97–104, 2006.

[10] Alina Beygelzimer, Sham Kakade, and John Langford. Extended version of "cover trees for nearest neighbor". 2006. URL: https://hunch.net/~jl/projects/cover_tree/paper/paper.pdf.

[11] Otakar Boruvka. O jistém problému minimálním. *Mor. přírodovědecká společnost*, 1926.

[12] Mireille Boutin and Gregor Kemper. On reconstructing n-point configurations from the distribution of distances or areas. *Advances in Applied Mathematics*, 32(4):709–735, 2004.

[13] Alexander M Bronstein, Michael M Bronstein, Alfred M Bruckstein, and Ron Kimmel. Analysis of two-dimensional non-rigid shapes. *International Journal of Computer Vision*, 78(1):67–88, 2008.

[14] Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009.

[15] Gunnar Carlsson and Facundo Memoli. Characterization, stability and convergence of hierarchical clustering methods. *Journal of machine learning research*, 11:1425–1470, 2010.

[16] Gunnar Carlsson and Facundo Mémoli. Multiparameter hierarchical clustering methods. In *Classification as a Tool for Research*, pages 63–70. Springer, 2010.

[17] Mathieu Carriere, Frederic Chazal, Yuichi Ike, Theo Lacombe, Martin Royer, and Yuhei Umeda. Perslay: A neural network layer for persistence diagrams and new graph topological signatures. *AISTATS, arXiv:1904.09378*, 2020.

[18] Samidh Chatterjee, Michael Connor, and Piyush Kumar. Geometric minimum spanning trees with geofilterkruskal. In *International Symposium on Experimental Algorithms*, pages 486–500. Springer, 2010.

[19] Frédéric Chazal, Vin De Silva, Marc Glisse, and Steve Oudot. *The structure and stability of persistence modules*. Springer, 2016.

[20] Paul Chew, Michael Goodrich, Daniel Huttenlocher, Klara Kedem, Jon Kleinberg, and Dina Kravets. Geometric pattern matching under euclidean motion. *Computational Geometry*, 7:113–124, 1997.

[21] Giansalvo Cirrincione, Gabriele Ciravegna, Pietro Barbiero, Vincenzo Randazzo, and Eros Pasero. The gh-exin neural network for hierarchical clustering. *Neural Networks*, 121:57–73, 2020.

[22] James Clough, Nicholas Byrne, Ilkay Oksuz, Veronika A Zimmer, Julia A Schnabel, and Andrew King. A topological loss function for deep-learning based image segmentation using persistent homology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[23] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, 2007.

[24] Richard Cole and Lee-Ad Gottlieb. Searching dynamic point sets in spaces with bounded doubling dimension. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 574–583, 2006.

[25] Thomas Cormen. *Introduction to algorithms*. MIT Press McGraw-Hill, Cambridge, Mass. New York, 1990.

[26] Ryan R Curtin, Dongryeol Lee, William B March, and Parikshit Ram. Plug-and-play dual-tree algorithm runtime analysis. *J. Mach. Learn. Res.*, 16:3269–3297, 2015.

[27] Ryan Ross Curtin. *Improving dual-tree algorithms*. PhD thesis, Georgia Institute of Technology, 2015.

[28] Harish Doraiswamy, Julien Tierny, Paulo JS Silva, Luis Gustavo Nonato, and Claudio Silva. Topomap: A 0-dimensional homology preserving projection of high-dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):561–571, 2020.

[29] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. In *Proceedings 41st annual symposium on foundations of computer science*, pages 454–463. IEEE, 2000.

[30] Asi Elad and Ron Kimmel. Bending invariant representations for surfaces. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I. IEEE, 2001.

[31] Yury Elkin and Vitaliy Kurlin. Counterexamples expose gaps in the proof of time complexity for cover trees introduced in 2006. In *IEEE Workshop on Topological Data Analysis and Visualization (TopoInVis)*. URL: https://arxiv.org/abs/2208.09447.

[32] Yury Elkin and Vitaliy Kurlin. The mergegram of a dendrogram and its stability. In *Mathematical Foundations of Computer Science (MFCS)*, volume 170 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 32:1–32:13, 2020. `doi:10.4230/LIPIcs.MFCS.2020.32`.

[33] Yury Elkin and Vitaliy Kurlin. Isometry invariant shape recognition of projectively perturbed point clouds by the mergegram extending 0d persistence. *Mathematics*, 9(17):2121, 2021.

[34] Yury Elkin, Di Liu, and Vitaliy Kurlin. A fast approximate skeleton with guarantees for any cloud of points in a euclidean space. In *Topological Methods in Data Analysis and Visualization VI*, pages 245–269. Springer, 2021.

[35] Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL: `https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense`, `doi:10.5281/zenodo.4724125`.

[36] Raphael A Finkel and Jon Louis Bentley. Quad trees a data structure for retrieval on composite keys. *Acta informatica*, 4(1):1–9, 1974.

[37] Guoji Fu, Chengbin Hou, and Xin Yao. Learning topological representation for networks via hierarchical sampling. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2019.

[38] Keinosuke Fukunaga and Patrenahalli M. Narendra. A branch and bound algorithm for computing k-nearest neighbors. *IEEE transactions on computers*, 100(7):750–753, 1975.

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

[40] Rickard Brüel Gabrielsson, Bradley J Nelson, Anjan Dwaraknath, and Primoz Skraba. A topology layer for machine learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1553–1563. PMLR, 2020.

[41] Michael T Goodrich, Joseph SB Mitchell, and Mark W Orletsky. Approximate geometric pattern matching under rigid motions. *Trans. on Pattern Analysis and Machine Intelligence*, 21(4):371–379, 1999.

[42] C Gordon Bell and Allen Newell. *Computer Structures*. McGraw-Hill, New York, NY, January 1971.

[43] Sariel Har-Peled and Manor Mendel. Fast construction of nets in low-dimensional metrics and their applications. *SIAM Journal on Computing*, 35(5):1148–1184, 2006.

[44] Michael P Holmes, Alexander G Gray, and Charles Lee Isbell Jr. Quic-svd: Fast svd using cosine trees. In *NIPS*, pages 673–680, 2008.

[45] Daniel P Huttenlocher, Gregory A. Klanderman, and William J Rucklidge. Comparing images using the hausdorff distance. *Transactions on pattern analysis and machine intelligence*, 15(9):850–863, 1993.

[46] Mike Izbicki and Christian Shelton. Faster cover trees. In *International Conference on Machine Learning*, pages 1162–1170. PMLR, 2015.

[47] Mahmoodreza Jahanseir and Don Sheehy. Transforming hierarchical trees on metric spaces. In *Canadian Conference on Computational Geometry*, pages 107–113, 2016.

[48] Frank Jones. *Lebesgue integration on euclidean space*. Jones and Bartlett, Sudbury, MA, November 2000.

[49] S. Kalisnik, V. Kurlin, and D. Lesnik. A higher-dimensional homologically persistent skeleton. *Advances in Applied Mathematics*, 102:113–142, 2019.

[50] Md Rezaul Karim, Oya Beyan, Achille Zappa, Ivan G Costa, Dietrich Rebholz-Schuhmann, Michael Cochez, and Stefan Decker. Deep learning-based clustering approaches for bioinformatics. *Briefings in bioinformatics*, 22(1):393–415, 2021.

[51] Michael Kerber, Dmitriy Morozov, and Arnur Nigmetov. Geometry helps to compare persistence diagrams. In *Proceedings of ALENEX*, pages 103–112, 2016.

[52] Kwangho Kim, Jisu Kim, Joon Sik Kim, Frederic Chazal, and Larry Wasserman. Efficient topological layer based on persistent landscapes. *arXiv preprint arXiv:2002.02778*, 2020.

[53] Thomas Kollar. *Fast nearest neighbors*. Technical report, Computer Science and Artificial Intelligence Laboratory, 2006.

[54] Robert Krauthgamer and James R Lee. Navigating nets: Simple algorithms for proximity search. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 798–807, 2004.

[55] V. Kurlin. A homologically persistent skeleton is a fast and robust descriptor of interest points in 2d images. In *Proceedings of CAIP*, volume 9256, pages 606–617, 2015.

[56] Vitaliy Kurlin. Auto-completion of contours in sketches, maps and sparse 2d images based on topological persistence. In *Proceedings of SYNASC 2014 workshop CTIC: Computational Topology in Image Context*, pages 594–601. IEEE, 2014.

[57] Vitaliy Kurlin. A fast and robust algorithm to count topologically persistent holes in noisy clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1458–1463, 2014.

[58] Vitaliy Kurlin. A one-dimensional homologically persistent skeleton of an unstructured point cloud in any metric space. *Computer Graphics Forum*, 34(5):253–262, 2015.

[59] 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.

[60] Ting Liu, Andrew W Moore, Alexander G Gray, and Ke Yang. An investigation of practical approximate nearest neighbor algorithms. In *NIPS*, volume 12, page 2004, 2004.

[61] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[62] S Manocha and Mark A Girolami. An empirical analysis of the probabilistic k-nearest neighbour classifier. *Pattern Recognition Letters*, 28(13):1818–1824, 2007.

[63] William B March, Parikshit Ram, and Alexander G Gray. Fast euclidean minimum spanning tree: algorithm, analysis, and applications. In *Proceedings of SIG KDD: Knowledge discovery and data mining*, pages 603–612, 2010.

[64] Clément Maria, Jean-Daniel Boissonnat, Marc Glisse, and Mariette Yvinec. The gudhi library: Simplicial complexes and persistent homology. In *International Congress on Mathematical Software*, pages 167–174. Springer, 2014.

[65] Leland McInnes and John Healy. Accelerated hierarchical density based clustering. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 33–42. IEEE, 2017.

[66] James 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.

[67] Facundo Mémoli and Guillermo Sapiro. A theoretical and computational framework for isometry invariant recognition of point cloud data. *Foundations of Computational Mathematics*, 5(3):313–347, 2005.

[68] Dmitriy Morozov, Kenes Beketayev, and Gunther Weber. Interleaving distance between merge trees. *Discrete and Computational Geometry*, 49(22-45):52, 2013.

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

[70] Maks Ovsjanikov, Alexander M Bronstein, Michael M Bronstein, and Leonidas J Guibas. Shape google: a computer vision approach to isometry invariant shape retrieval. In *International Conference on Computer Vision workshops*, pages 320–327, 2009.

[71] Maks Ovsjanikov, Quentin Mérigot, Facundo Mémoli, and Leonidas Guibas. One point isometric matching with the heat kernel. In *Computer Graphics Forum*, volume 29, pages 1555–1564, 2010.

[72] Salman Parsa. A deterministic o(m log m) time algorithm for the reeb graph. *Discrete & Computational Geometry*, 49(4):864–878, 2013.

[73] Mark Pauly, Markus Gross, and Leif P Kobbelt. Efficient simplification of point-sampled surfaces. In *IEEE Visualization*, pages 163–170, 2002.

[74] Dan Pelleg and Andrew Moore. Accelerating exact k-means algorithms with geometric reasoning. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 277–281, 1999.

[75] Robert Clay Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401, 1957.

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

[77] Parikshit Ram, Dongryeol Lee, and Alexander G Gray. Nearest-neighbor search on a time budget via max-margin trees. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 1011–1022. SIAM, 2012.

[78] Parikshit Ram, Dongryeol Lee, William March, and Alexander Gray. Linear-time algorithms for pairwise statistical problems. *Advances in Neural Information Processing Systems*, 22:1527–1535, 2009.

[79] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *IEEE transactions on pattern analysis and machine intelligence*, 32(1):105–119, 2008.

[80] Walter Rudin. *Functional Analysis*. International Series in Pure & Applied Mathematics. McGraw Hill Higher Education, Maidenhead, England, 2 edition, October 1990.

[81] Javier Sánchez, Nelson Monzón, and Agustín Salgado De La Nuez. An analysis and implementation of the harris corner detector. *Image Processing On Line*, 2018.

[82] Raimund Seidel. Understanding the inverse ackermann function. In *Twenty-second European Workshop on Computational Geometry Delphi, Greece March 27–29, 2006*, page 37, 2006.

[83] Gurjeet Singh, Facundo Mémoli, and Gunnar E Carlsson. Topological methods for the analysis of high dimensional data sets and 3d object recognition. *SPBG*, 91:100, 2007.

[84] Philip Smith and Vitaliy Kurlin. Skeletonisation algorithms with theoretical guarantees for unorganised point clouds with high levels of noise. *Pattern Recognition*, 115:107902, 2021.

[85] Julien Tierny, Guillaume Favelier, Joshua A Levine, Charles Gueunet, and Michael Michaux. The topology toolkit. *IEEE transactions on visualization and computer graphics*, 24(1):832–842, 2017.

[86] Alessandro Verri, Claudio Uras, Patrizio Frosini, and Massimo Ferri. On the use of size functions for shape analysis. *Biological cybernetics*, 70(2):99–107, 1993.

[87] Kieran Walsh, Mircea A Voineagu, Fatemeh Vafaee, and Irina Voineagu. Tdaview: an online visualization tool for topological data analysis. *Bioinformatics*, 36(18):4805–4809, 2020.

[88] Mengzhao Wang, Xiaoliang Xu, Qiang Yue, and Yuxiang Wang. A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. *arXiv preprint arXiv:2101.12631*, 2021.

[89] Xia Li Wang, Xiaochun Wang, and Xiaqiong Li. A fast two-level approximate euclidean minimum spanning tree algorithm for high-dimensional data. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 273–287. Springer, 2018.

[90] Yiqiu Wang, Shangdi Yu, Yan Gu, and Julian Shun. Fast parallel algorithms for euclidean minimum spanning tree and hierarchical spatial clustering. In *Proceedings of the 2021 International Conference on Management of Data*, pages 1982–1995, 2021.

[91] John William Joseph Williams. Algorithm 232: heapsort. *Commun. ACM*, 7:347–348, 1964.

[92] Peter N Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Symposium on Discrete Algorithms*, volume 93, pages 311–21, 1993.

[93] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017.

[94] Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33(2):249–274, 2005.

[95] Matthias Zwicker, Mark Pauly, Oliver Knoll, and Markus Gross. Pointshop 3d: An interactive system for point-based surface editing. *ACM Transactions on Graphics (TOG)*, 21(3):322–329, 2002.