Transforming Hierarchical Trees on Metric Spaces*

Mahmoodreza Jahanseir† Donald R. Sheehy‡

Abstract

We show how a simple hierarchical tree called a cover tree can be turned into an asymptotically more efficient one known as a net-tree in linear time. We also introduce two linear-time operations to manipulate cover trees called coarsening and refining. These operations make a trade-off between tree height and the node degree.

1 Introduction

There are many very similar data structures for searching and navigating n points in a metric space $\mathcal{M}$. Most such structures support range queries and (approximate) nearest neighbor search among others. For computational geometers, two of the most important such structures for general metric spaces are the cover tree [2] and the net-tree [8]. Cover trees, by virtue of their simplicity, have found wide adoption, especially for machine learning applications. Net-trees on the other hand, provide much stronger theoretical guarantees and can be used to solve a much wider class of problems, but they come at the cost of unrealistic constant factors and complex algorithms. In this paper, we generalize these two data structures and show how to convert a cover tree into a net tree in linear time. In fact, we show that a cover tree with the right parameters, satisfies the stronger conditions of a net tree, thus finding some middle ground between the two. In Section 5, we give efficient algorithms for modifying these parameters for an existing tree.

Related Work For Euclidean points, Quadtrees [5] and k-d trees [1] are perhaps the two famous data structures. Most data structures for general metric spaces are generalizations of these. Uhlmann [11] proposed ball trees to solve the proximity search on metric spaces. Ball trees are generalizations of k-d trees. Yianilos [12] proposed a structure similar to ball trees called a vp-tree that allows $O(\log n)$-time queries in expectation for restricted classes of inputs.

Clarkson [3] proposed two randomized data structures to answer nearest neighbor queries in metric spaces that satisfy a certain sphere packing property. These data structures assume that the input and query point are drawn from the same probability distribution. The nearest neighbor query time of these structures depends on the *spread* Δ of the input, which is the ratio of the diameter to the distance between the closest pair of points.

Karger & Ruhl [9] devised a dynamic data structure for nearest neighbor queries in growth restricted metrics. A *closed metric ball* centered at p with radius r is denoted $\mathrm{B}(p,r) := \{q \in P \mid \mathbf{d}(p,q) \leq r\}$. Karger & Ruhl defined the *expansion constant* as the minimum μ such that for all $p \in \mathcal{M}$ and $r > 0$, $|\mathrm{B}(p,2r)| \leq \mu\,|\mathrm{B}(p,r)|$. A growth restricted metric space has constant μ (independent of n). Karger & Ruhl proved that their data structure has size $\mu^{O(1)} n \log n$, and answers nearest neighbor queries in $\mu^{O(1)} \log n$.

Gupta et al. [7] defined the *doubling constant* ρ of a metric space as the minimum ρ such that every ball in $\mathcal{M}$ can be covered by ρ balls of half the radius. The *doubling dimension* is defined as $\gamma = \lg \rho$. A metric is called doubling when it has a constant doubling dimension.

Krauthgamer & Lee [10] proposed *navigating nets* to answer (approximate) nearest neighbor queries in $2^{O(\gamma)} \log \Delta + (1/\varepsilon)^{O(\gamma)}$-time for doubling metrics. Navigating nets require $2^{O(\gamma)} n$ space.

Gao et al. [6] proposed a $(1+\varepsilon)$-spanner with size $O(n/\varepsilon^d)$ for a set of n points in $\mathbb{R}^d$. Their data structure is similar to navigating nets and can be constructed in $O(n \log \Delta/\varepsilon^d)$ time and answers approximate nearest neighbor queries in $O(\log \Delta)$ time. They also maintained the spanner under dynamic updates and continuous motion of the points.

Har-Peled & Mendel [8] devised a data structure called a *net-tree* to address approximate nearest neighbor search and some other problems in doubling metrics. They proposed a linear-time algorithm to construct a net-tree of size $O(\rho^{O(1)} n)$ starting from a specific ordering of the points called an approximate greedy permutation. Constructing a greedy permutation requires $O(\rho^{O(1)} n \log(\Delta n))$ time. To beat the spread, they proposed a randomized algorithm to generate an approximate greedy permutation in $O(\rho^{O(1)} n \log n)$ time. Net-trees support approximate nearest neighbor search in $O(2^{O(\gamma)} \log n) + (1/\varepsilon)^{O(\gamma)}$ time.

Beygelzimer et al. [2] presented *cover trees* to solve the nearest neighbor problem in growth restricted metrics. Cover trees are a simplificiation of navigating nets and can be constructed incrementally in $O(\mu^6 n \log n)$

*Partially supported by the National Science Foundation under grant numbers CCF-1464379 and CCF-1525978

†University of Connecticut reza@engr.uconn.edu

‡University of Connecticut don.r.sheehy@gmail.com

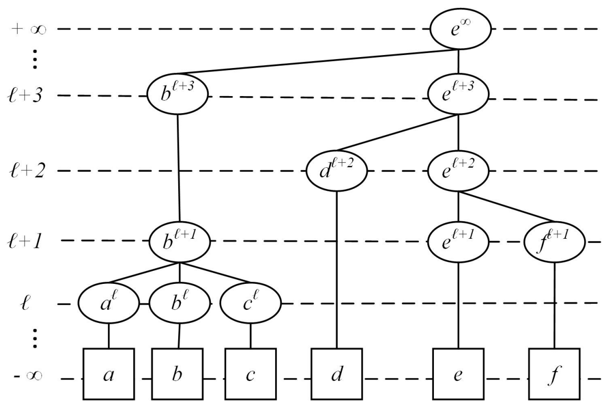

Figure 1: A hierarchical tree on $P = \{a, b, c, d, e, f\}$. Squares and ovals illustrate points and nodes respectively.

time. The space complexity of cover trees is $O(n)$ independent of doubling constant or expansion constant.

Cole & Gottlieb [4] extended the notion of navigating nets to construct a dynamic data structure to support approximate nearest neighbor search in $O(\log n) + (1/\varepsilon)^{O(1)}$ time for doubling metrics. Similar to net-trees, their data structure provides strong packing and covering properties. To insert a new point, they used biased skip lists to make the search process faster. They proved that the data structure requires $O(n)$ space independent of doubling dimension of the metric space.

2 Definitions

Hierarchical trees. Cover trees and net-trees are both examples of hierarchical trees. In these trees, the input points are leaves and each point p can be associated with many internal nodes. Each node is uniquely identified by its associated point and an integer called its *level*. Leaves are in level $-\infty$ and the root is in $+\infty$. The node in level ℓ associated with a point p is denoted p^{ℓ}. Let $\mathrm{par}(p^{\ell})$ be the parent of a node $p^{\ell} \in T$. Also, let $\mathrm{ch}(p^{\ell})$ be the children of p^{ℓ}. Each non-leaf has a child with the same associated point. Similar to compressed quadtrees, a node skips a level iff it is the only child of its parent and it has only one child. Let L_{ℓ} be the points associated with nodes in level at least ℓ. Let $P_{p^{\ell}}$ denote leaves of the subtree rooted at p^{ℓ}. The levels of the tree represent the metric space at different scales. The constant $\tau > 1$, called the *scale factor* of the tree determines the change in scale between levels. Fig 1 shows an example of hierarchical trees. Note that in this figure the tree is neither a cover tree nor a net-tree, because there are not any restrictions on the distance between points.

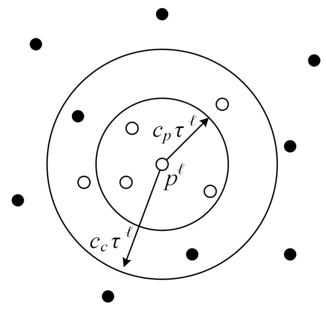

Figure 2: Packing and covering balls for a point p at level ℓ in a net-tree. White points belong to the subtree rooted at node p^{ℓ}.

Cover Trees. A *cover tree* T is a hierarchical tree with following properties.

- **Packing:** For all distinct $p, q \in L_{\ell}$, $\mathbf{d}(p, q) > c_p\tau^{\ell}$.
- **Covering:** For each $r^h \in \mathrm{ch}(p^{\ell})$, $\mathbf{d}(p, r) \leq c_c\tau^{\ell}$.

We call c_p and c_c the *packing constant* and the *covering constant*, respectively, and $c_c \geq c_p > 0$. We represent all cover trees with the same scale factor, packing constant, and covering constant with $\mathrm{CT}(\tau, c_p, c_c)$. Note that the cover tree definition by Beygelzimer et al. [2] results a tree in $\mathrm{CT}(2, 1, 1)$.

Net-trees. A *net-tree* is a hierarchical tree. For each node p^{ℓ} in a net-tree, the following invariants hold.

- **Packing:** $\mathrm{B}(p, c_p\tau^{\ell}) \bigcap P \subset P_{p^{\ell}}$.
- **Covering:** $P_{p^{\ell}} \subset \mathrm{B}(p, c_c\tau^{\ell})$. [1]

Here, c_p and c_c are defined similar to cover trees. Fig 2 illustrates both packing and covering balls for a point p at some level ℓ in a net-tree. Let $\mathrm{NT}(\tau, c_p, c_c)$ denote the set of net-trees. The algorithm in [8] constructs a tree in $\mathrm{NT}(11, \frac{\tau-5}{2(\tau-1)}, \frac{2\tau}{\tau-1})$.

The main difference in the definitions is in the packing conditions. The net-tree requires the packing to be consistent with the hierarchical structure of the tree, a property not necessarily satisfied by the cover trees. Also, Har-Peled and Mendel [8] set $\tau = 11$, whereas optimized cover tree code sets $\tau = 1.3$.

A net-tree can be augmented to maintain a list of nearby nodes called relatives defined for each node p^{ℓ} as follows.

$$\mathrm{Rel}(p^{\ell}) = \{x^f \in T \text{ with } y^g = \mathrm{par}(x^f) \mid f \leq \ell < g, \text{ and } \mathbf{d}(p, x) \leq c_r\tau^{\ell}\}$$

[1]The packing condition we give is slightly different from [8], but it is an easy exercise to prove this (more useful) version is equivalent.

We call c_r the *relative constant*, and Har-Peled and Mendel set $c_r = 13$.

In this paper, we add a new and easy to implement condition on cover trees. We require that children of a node p^ℓ are closer to p than to any other point in L_ℓ.

3 From cover trees to net-trees

In this section, first we show that for every node in a cover tree, the size of children and relatives of that node is constant. Then, we prove that a cover tree with a sufficiently large scale factor satisfies both stronger packing and covering properties of net-trees.

Lemma 1 *For each node p^ℓ in $T \in \mathrm{CT}(\tau, c_p, c_c)$, $|\mathrm{ch}(p^\ell)| = O(\rho^{\lg c_c\tau/c_p})$.*

Proof. When $|\mathrm{ch}(p^\ell)| > 1$, all children of p^ℓ are in level $\ell - 1$. From the packing property, the distance between every two nodes in this list is greater than $c_p\tau^{\ell-1}$. We know that all children of p^ℓ are within the distance $c_c\tau^\ell$ of p. By the definition of the doubling constant, the ball centered at p with radius $c_c\tau^\ell$ will be covered by $O(\rho^{\lg c_c\tau/c_p})$ balls of radius $c_p\tau^{\ell-1}$. □

Lemma 2 *Let $p^\ell \in T$ and $T \in \mathrm{CT}(\tau, c_p, c_c)$. For each two nodes $s^e, t^f \in \mathrm{Rel}(p^\ell)$, $\mathbf{d}(s,t) > c_p\tau^\ell$.*

Proof. Let $r^h = \mathrm{par}(s^e)$. By the definition of relatives, $e \le \ell < h$. If $e < \ell$, then $s = r$ and $s \in L_h$. Because $L_h \subset L_\ell$, the distance of s to all points in L_ℓ is greater than $c_p\tau^\ell$. Otherwise, s is in level ℓ. The same argument holds for t^f. Therefore, $s, t \in L_\ell$, and it implies $\mathbf{d}(s,t) > c_p\tau^\ell$. □

Lemma 3 *For each node p^ℓ in $T \in \mathrm{CT}(\tau, c_p, c_c)$, $|\mathrm{Rel}(p^\ell)| = O(\rho^{\lg c_r/c_p})$*

Proof. By the definition of relatives, all nodes in $\mathrm{Rel}(p^\ell)$ are within the distance $c_r\tau^\ell$ of point p. From Lemma 2, the distance between any two points in $\mathrm{Rel}(p^\ell)$ is greater than $c_p\tau^\ell$. Therefore, the total size of $\mathrm{Rel}(p^\ell)$ is $O(\rho^{\lg c_r/c_p})$. □

Lemma 4 *For each descendant x^f of p^ℓ in $T \in \mathrm{CT}(\tau, c_p, c_c)$, $\mathbf{d}(p,x) < \frac{c_c\tau}{\tau-1}\tau^\ell$*

Proof. The covering property and the triangle inequality imply that

$$\mathbf{d}(p,x) \le \sum_{i=f}^{\ell} c_c\tau^i < c_c\sum_{i=0}^{\infty}\tau^{\ell-i} = c_c\frac{\tau^{\ell+1}}{\tau-1}.$$

□

Theorem 5 *For all $\tau > \frac{2c_c}{c_p} + 1$, if $T \in \mathrm{CT}(\tau, c_p, c_c)$, then $T \in \mathrm{NT}(\tau, \frac{c_p(\tau-1)-2c_c}{2(\tau-1)}, \frac{c_c\tau}{\tau-1})$.*

Algorithm 1 Augmenting a given cover tree with relatives

```
1: procedure AUGMENT($T, c_r$)
2:     for all $p^\ell \in T$ in decreasing order of level $\ell$ do
3:         Rel($p^\ell$) ← $p^\ell$
4:         if $p^\ell$ is not the root then
5:             RELATIVES($p^\ell, c_r, true$)
```

Proof. From Lemma 4, for a node $p^\ell \in T$, $P_{p^\ell} \subset \mathrm{B}(p, \frac{c_c\tau}{\tau-1}\tau^\ell)$. Suppose for contradiction there exists a point $r \in \mathrm{B}(p, \frac{c_p(\tau-1)-2c_c}{2(\tau-1)}\tau^\ell)$ such that $r \notin P_{p^\ell}$. Then, there exists a node $x^f \in T$ which is the lowest node with $f \ge \ell$ and $r \in P_{x^f}$. Let y^g be the child of x^f such that $r \in P_{y^g}$. It is clear that $g < \ell$. First, Let $g < f - 1$. So, $x = y$ and $\mathbf{d}(p,x) = \mathbf{d}(p,y) > c_p\tau^\ell$. By the triangle inequality,

$$\mathbf{d}(y,r) \ge \mathbf{d}(y,p) - \mathbf{d}(p,r) > c_p\tau^\ell - \frac{c_p(\tau-1)-2c_c}{2(\tau-1)}\tau^\ell > \frac{c_p(\tau-1)+2c_c}{2(\tau-1)}\tau^\ell.$$

Also, $\mathbf{d}(y,r) \le \frac{c_c\tau}{\tau-1}\tau^g < \frac{c_c\tau^\ell}{\tau-1}$. Therefore,

$$\frac{c_p(\tau-1)+2c_c}{2(\tau-1)}\tau^\ell < \frac{c_c\tau^\ell}{\tau-1}.$$

This implies that $c_p(\tau-1) < 0$, which is a contradiction. Now, let $g = f-1$. In this case, we have $f = \ell$ and $g = \ell - 1$. By the parent property, $\mathbf{d}(y,p) > \mathbf{d}(y,x)$. So,

$$\mathbf{d}(y,p) \ge \mathbf{d}(p,x) - \mathbf{d}(x,y) > c_p\tau^\ell - \mathbf{d}(y,p) > c_p\tau^\ell/2.$$

Also, by the triangle inequality,

$$\mathbf{d}(y,r) \ge \mathbf{d}(y,p) - \mathbf{d}(p,r) > \frac{c_p}{2}\tau^\ell - \frac{c_p(\tau-1)-2c_c}{2(\tau-1)}\tau^\ell > \frac{c_c\tau^\ell}{\tau-1}.$$

We get a contradiction because $\mathbf{d}(y,r) \le \frac{c_c\tau^\ell}{\tau-1}$. Therefore, $r \in P_{p^\ell}$. □

4 Augment cover trees

Theorem 5 shows that for a sufficiently large scale factor packing and covering properties in a cover tree imply packing and covering properties of net-trees. However, net-tree nodes maintain a list of nearby nodes called *relatives*. Algorithm 1 is a procedure that adds a list of relatives to each node of a cover tree. Note that RELATIVES is similar to the find relative algorithm in [8], but it gives a smaller relative constant.

Algorithm 2 Finding relatives of a node p^ℓ

```
1: procedure RELATIVES($p^\ell, c_r, update$)
2:     Let $q^m = parent(p^\ell)$
3:     for all $x^f \in \mathrm{Rel}(q^m)$ do
4:         Let $y^g = \mathrm{par}(x^f)$
5:         if $\mathbf{d}(p,x) \leq c_r\tau^\ell$ and $f \leq \ell < g$ then
6:             Add $x^f$ to $\mathrm{Rel}(p^\ell)$
7:         else if $update = true$ and $\ell \leq f < m$ and $\mathbf{d}(p,x) \leq c_r\tau^f$ then
8:             Add $p^\ell$ to $\mathrm{Rel}(x^f)$
9:     $candidates \leftarrow \bigcup_{r^e \in \mathrm{Rel}(q^m)} \mathrm{ch}(r^e) \setminus \{p^\ell\}$
10:    for all $x^f \in candidates$ do
11:        Let $y^g = \mathrm{par}(x^f)$
12:        if $\mathbf{d}(p,x) \leq c_r\tau^\ell$ and $f \leq \ell < g$ then
13:            Add $x^f$ to $\mathrm{Rel}(p^\ell)$
14:        else if $\mathbf{d}(p,x) \leq c_r\tau^f$ and $\ell \leq f < m$ then
15:            if $update = true$ then
16:                Add $p^\ell$ to $\mathrm{Rel}(x^f)$
17:            $candidates \leftarrow candidates \cup \mathrm{ch}(x^f)$
```

Theorem 6 *For each node p^ℓ in $T \in \mathrm{CT}(\tau, c_p, c_c)$ and $c_r = \frac{c_c\tau^2}{(\tau-1)^2}$,* RELATIVES *correctly finds* $\mathrm{Rel}(p^\ell)$.

Proof. Suppose for contradiction there exists x^f with $y^g = \mathrm{par}(x^f)$ such that $x^f \in \mathrm{Rel}(p^\ell)$, and RELATIVES does not find it. Therefore, either $x^f \notin \mathrm{Rel}(q^m)$ or it has an ancestor s^h with $h \leq m-1$ such that $p^\ell \notin \mathrm{Rel}(s^h)$. We consider each case separately.

Case 1: $x^f \notin \mathrm{Rel}(q^m)$. In this case, at least one of the two conditions of relatives does not hold for x^f. If $\mathbf{d}(q,x) > \frac{c_c\tau^2}{(\tau-1)^2}\tau^m$, then by the triangle inequality,

$$\begin{aligned}\mathbf{d}(p,x) \geq \mathbf{d}(q,x) - \mathbf{d}(p,q) &> \frac{c_c\tau^2}{(\tau-1)^2}\tau^m - c_c\tau^m \\ &> c_c\frac{2\tau-1}{(\tau-1)^2}\tau^{\ell+1}.\end{aligned}$$

We assumed that $x^f \in \mathrm{Rel}(p^\ell)$, so $\mathbf{d}(p,x) \leq \frac{c_c\tau^2}{(\tau-1)^2}\tau^\ell$. These inequalities imply $\tau < 1$, a contradiction. If $\mathbf{d}(q,x) \leq \frac{c_c\tau^2}{(\tau-1)^2}\tau^m$ and $g > f > m$, then $f > \ell$ is also a contradiction. The last case $\mathbf{d}(q,x) \leq \frac{c_c\tau^2}{(\tau-1)^2}\tau^m$ and $f < g \leq m$ is a special case of $p^\ell \notin \mathrm{Rel}(s^h)$, which is described in the following.

Case 2: $p^\ell \notin \mathrm{Rel}(s^h)$. We know that $\ell < h < m$, so $\mathbf{d}(p,s) > \frac{c_c\tau^2}{(\tau-1)^2}\tau^h$. Also, $\tau^{h-1} \geq \tau^\ell$ because $h \geq \ell+1$. Using the triangle inequality and then applying Lemma 4,

$$\begin{aligned}\mathbf{d}(p,s) \leq \mathbf{d}(p,x) + \mathbf{d}(x,s) &< \frac{c_c\tau^2}{(\tau-1)^2}\tau^\ell + \frac{c_c\tau}{\tau-1}\tau^h \\ &< \frac{c_c\tau^2}{(\tau-1)^2}\tau^{h-1} + \frac{c_c\tau}{\tau-1}\tau^h = c_c(\frac{\tau^2}{(\tau-1)^2})\tau^h.\end{aligned}$$

This is a contradiction. □

Theorem 7 *Algorithm 1 has time complexity $O(\rho^{\lg(\frac{c_c\tau}{c_p(\tau-1)})^2\tau} n)$.*

Proof. We use an amortized analysis for the time complexity of RELATIVES. While finding relatives, if a node is inserted into the relative list of another node, we decrease one credit from the node whose relative list has been grown. Note that in Algorithm 2, for a node x^f in $\mathrm{Rel}(q^m)$ or children of $\mathrm{Rel}(q^m)$, when $x^f \notin \mathrm{Rel}(p^\ell)$ and $p^\ell \notin \mathrm{Rel}(x^f)$, p^ℓ is responsible for checking x^f. Also, a child of a node x^f is required to be checked against relative conditions if $p^\ell \in \mathrm{Rel}(x^f)$. In this case, we charge node x^f one credit. From Lemma 3, the relative list for each node has size $O(\rho^{\lg\frac{c_c\tau^2}{c_p(\tau-1)^2}})$. Also, Lemma 1 implies that each node has at most $O(\rho^{\lg\frac{c_c\tau}{c_p}})$ children. Therefore, the total required credit for each node of the tree is $O(\rho^{\lg\frac{c_c\tau^2}{c_p(\tau-1)^2}}) + 2 \cdot O(\rho^{\lg\frac{c_c\tau^2}{c_p(\tau-1)^2}}\rho^{\lg\frac{c_c\tau}{c_p}}) = O(\rho^{\lg(\frac{c_c\tau}{c_p(\tau-1)})^2\tau})$. So, the total total time complexity is $O(\rho^{\lg(\frac{c_c\tau}{c_p(\tau-1)})^2\tau} n)$. □

5 Transform cover trees

For a cover tree, there is a trade-off between the height of the tree and the scale factor. It is not hard to see that the height of a cover tree has upper bound $O(\log_\tau \Delta)$. So by increasing the scale factor, the height of the tree will be decreased. Also, from Lemma 1, increasing the scale factor results in more children for each node of a cover tree.

In this section, we define two operations to change scale factor of a given tree. A *coarsening* operation modifies the tree to increase the scale factor. Similarly, a refining operation results a tree with smaller scale factor. Note that in Theorem 5, we assumed that $\tau > \frac{2c_c}{c_p} + 1$. However, in many cases we may have $\tau \leq \frac{2c_c}{c_p} + 1$. For example, Beygelzimer et. al. [2] set $\tau = 2$, and they found $\tau = 1.3$ is even more efficient in practice. In these situations, we can use the coarsening operation to get a cover tree with the stronger packing and covering conditions of net-trees.

5.1 Coarsening

The coarsening operation can be seen as combining every k levels of T into one level in T'. We define a mapping between nodes of T and T'. In this mapping, each

Algorithm 3 Coarsening operation for a given cover tree

```
1: procedure COARSENING($T, k$)
2:     $T' \leftarrow \emptyset$
3:     AUGMENT($T, \frac{3c_c\tau^2}{(\tau-1)^2}$)
4:     for all $p^\ell \in T$ in increasing order of level $\ell$ do
5:         $q^m \leftarrow$ the lowest ancestor of $p^\ell$ with $m' > \ell'$
6:         if $p = q$ then
7:             $p^{\ell'} \leftarrow$ FINDNODE(high($p$), $\ell'$)
8:             $q^{\ell'+1} \leftarrow$ FINDNODE(high($q$), $\ell'+1$)
9:         else
10:            $par \leftarrow q^m$
11:            $relatives \leftarrow \mathrm{Rel}(q^m)$
12:            if $m' > \ell'+1$ then
13:                $h \leftarrow k(\lfloor \ell/k \rfloor + 2) - 1$
14:                RELATIVES($q^h, \frac{3c_c\tau^2}{(\tau-1)^2}, false$) $\cup \{q^h\}$
15:                $relatives \leftarrow \mathrm{Rel}(q^h)$
16:            for all $x^f \in relatives$ do
17:                if $f' = \ell'+1$ then
18:                    $x^f \leftarrow$ RESTRICTEDNN($p, x^f, k$)
19:                if $\mathbf{d}(p,x) < \mathbf{d}(p,par)$ then
20:                    $par \leftarrow x^f$
21:            $par^{\ell'} \leftarrow$ FINDNODE(high($par$), $\ell'$)
22:            $par^{\ell'+1} \leftarrow$ FINDNODE(high($par$), $\ell'+1$)
23:            $p^{\ell'} \leftarrow$ FINDNODE(high($p$), $\ell'$)
24:            Add $p^{\ell'}$ as a child of $par^{\ell'+1}$
```

node p^ℓ in T maps to a node $p^{\ell'} = p^{\lfloor \ell/k \rfloor}$ in T'. Here, we use prime as a function that indicates the level of the node in T' that corresponds to p^ℓ, i.e. $\ell' = \lfloor \ell/k \rfloor$. We also assume that each point p in T' maintains high(p), which is the highest node of T' associated to point p. Algorithm 3 describes the coarsening operation.

COARSENING uses three procedures RELATIVES, RESTRICTEDNN, and FINDNODE. The first procedure is described in Algorithm 2. Note that in Algorithm 3, T does not have node $q^{k(\lfloor \ell/k \rfloor+2)-1}$. This node is a dummy and we set q^m as its parent. The only reason to use this dummy node is to bound the running time of the algorithm. The next procedure is RESTRICTEDNN, and it returns the nearest neighbor to point p among those nodes of the subtree rooted at x^f such that their levels in T' are greater that ℓ'. Finally, FINDNODE receives a node $p^{\ell'}$ and a level m' in T', and it tries to find node $p^{m'}$. If it finds that node, the node is returned. Otherwise, $p^{m'}$ will be inserted in T' such that it satisfies all properties of a hierarchical tree. More specifically, if $p^{m'}$ has only one child $p^{e'}$ and $p^{e'}$ has only one child, then $p^{e'}$ will be removed from T' and the only child of $p^{e'}$ will be added as a child of $p^{m'}$. Then, this new node will be returned.

Theorem 8 *Algorithm 3 converts a $T \in \mathrm{CT}(\tau, c_p, c_c)$ to $T' \in \mathrm{CT}(\tau^k, c_p, \frac{c_c\tau}{\tau-1})$.*

Proof. The theorem requires showing three invariants holds: the covering property, the packing property, and the parent relation. First we prove that T' satisfies the covering property. If $r^e \in T$ is the descendant at most k levels down from some p^ℓ, then from Lemma 4,

$$\mathbf{d}(p,r) < \frac{c_c\tau}{\tau-1}\tau^\ell < \frac{c_c\tau}{\tau-1}(\tau')^{\ell'}.$$

Because we are combining sets of k consecutive levels, it follows that each node in T' will have a node in the level above whose distance is at most this amount. It follows that T' has a covering constant $\frac{c_c\tau}{\tau-1}$.

Next, we prove that the packing constant is correct. If $\ell = k\ell'$, then the minimum distance between points in level ℓ' of T' is equal to the minimum distance between points in level ℓ of T, which is at least $c_p\tau^\ell = c_p(\tau')^{\ell'}$. Thus, the points in level ℓ' of T' satisfy the packing condition and the packing constant is c_p.

Now, we prove that this algorithm correctly finds the parent of $p^{\ell'}$ in T'. Without loss of generality, let ℓ be divisible by k. Also, let s be the closest point to p among all points in $L_{\ell+k}$, and s has been appeared for the first time in level e such that $e' > \ell'$. So, s is the right parent for p^ℓ. For contradiction, assume that $s^e \notin \mathrm{Rel}(q^m)$ and s^e is not resulted from RESTRICTEDNN over all nodes in $\mathrm{Rel}(q^m)$. Let t^h be parent of s^e. From Lemma 4,

$$\mathbf{d}(p,s) < \mathbf{d}(p,q) \le \frac{c_c\tau}{\tau-1}\tau^m.$$

We have following cases:

Case 1: $\mathbf{d}(s,q) > \frac{3c_c\tau^2}{(\tau-1)^2}\tau^m$. By the triangle inequality,

$$\mathbf{d}(s,q) \le \mathbf{d}(p,s) + \mathbf{d}(p,q) < 2\mathbf{d}(p,q) \le \frac{2c_c\tau}{\tau-1}\tau^m,$$

which is a contradiction, because $\tau > 1$.

Case 2: $\mathbf{d}(s,q) \le \frac{3c_c\tau^2}{(\tau-1)^2}\tau^m$ and $e > m$. In this case, there exists a node s^g with $g \le m$, such that it satisfies both conditions of relatives. So, $s^g \in \mathrm{Rel}(q^m)$ and the algorithm correctly finds s^g. [2]

Case 3: $\mathbf{d}(s,q) \le \frac{3c_c\tau^2}{(\tau-1)^2}\tau^m$, $m \ge h$, and RESTRICTEDNN does not find s^e. Let x^f be the highest ancestor s^e such that $f \le m$. Then, Lemma 4 implies

$$\mathbf{d}(x,s) \le \frac{c_c\tau}{\tau-1}\tau^f < \frac{c_c\tau}{\tau-1}\tau^m.$$

[2] g can be equal to $-\infty$, in this case we have a long edge from a node s in a level greater than m to the point s in level $-\infty$.

Also, by the triangle inequality,

$$\begin{aligned}\mathbf{d}(x,q) &\leq \mathbf{d}(x,s)+\mathbf{d}(p,s)+\mathbf{d}(p,q)\\ &< \mathbf{d}(x,s)+2\mathbf{d}(p,q)\\ &< \frac{c_c\tau}{\tau-1}\tau^m+\frac{2c_c\tau}{\tau-1}\tau^m\\ &< \frac{3c_c\tau}{\tau-1}\tau^m<\frac{3c_c\tau^2}{(\tau-1)^2}\tau^m.\end{aligned}$$

Therefore, $x^f \in \mathrm{Rel}(q^m)$. Because $e' > \ell'$, RestrictedNN returns s^e as the nearest neighbor to p^ℓ, which is a contradiction. □

Theorem 9 *The time complexity of Algorithm 3 is* $O(\rho^{\lg(\frac{c_c\tau}{c_p(\tau-1)})^2\tau} n\lg k)$.

Proof. From Theorem 7, T can be augmented with relatives in $O(\rho^{\lg(\frac{c_c\tau}{c_p(\tau-1)})^2\tau} n)$ time. As a preprocessing step, we can maintain the lowest ancestor of all nodes in T in $O(n)$ time, which results a constant time access in the algorithm. By Lemma 3, the size of each list of relatives is $O(\rho^{\lg\frac{c_c\tau^2}{c_p(\tau-1)^2}})$. The time complexity of RestrictedNN is $O(\lg k)$, because the height of the subtree is $O(k)$. When Algorithm 3 is processing all nodes of level ℓ in T, for each point p in T', the level of high(p) is at most $\ell'+1$. So, FindNode requires $O(1)$ to return a node of T' in level ℓ' or $\ell'+1$. Since the number of edges in T is $O(n)$, finding relatives of dummy nodes will be done $O(n)$ times for the entire algorithm. Consequently, because $c_c \geq c_p > 0$, the total time complexity of the algorithm is $O(\rho^{\lg(\frac{c_c\tau}{c_p(\tau-1)})^2\tau} n\lg k)$. □

5.2 Refining

Decreasing the scale factor is another useful operation for cover trees, and we call this operation *refining.* To refine a given cover tree T, each level ℓ in T is split into at most k levels $k\ell,\dots,(k\ell+k-1)$ in T'. Note that by this division, a node p^ℓ in T may be appeared at most k times in levels $k\ell,\dots,(k\ell+k-1)$ of T'. Similar to the coarsening operation, suppose that each point p in T' maintains high(p) which is the highest node associated to point p. Algorithm 4 describes the refining operation.

Algorithm 4 Refining operation for a given cover tree

```
1: procedure Refining($T, k$)
2:     $T' \leftarrow \emptyset$
3:     Augment($T, \frac{3c_c\tau^2}{(\tau-1)^2}$)
4:     for all $p^\ell \in T$ in increasing order of level $\ell$ do
5:         Let $q^m = \mathrm{par}(p^\ell)$
6:         $p^{h'} \leftarrow \mathrm{high}(p)$
7:         if $p = q$ and $h' < k\ell$ then
8:             $p^{k\ell} \leftarrow$ FindNode(high$(p), k\ell$)
9:             $q^{k(\ell+1)} \leftarrow$ FindNode(high$(q), k(\ell+1)$)
10:        else if $p \neq q$ then
11:            $par \leftarrow q^m$
12:            $list \leftarrow \bigcup_{s^h\in\mathrm{Rel}(q^m)} \mathrm{ch}(s^h) \setminus \{q^\ell\}$
13:            for all $x^f \in list$ where $f = \ell$ do
14:                $x^{h'} \leftarrow \mathrm{high}(x)$
15:                if $\mathbf{d}(p,x) \leq c_c\tau^{h'/k}$ and $\mathbf{d}(p,x) < \mathbf{d}(p,par)$ then
16:                    $par \leftarrow x^f$
17:            Find an $i$ such that $c_p\tau^{\ell+i/k} < \mathbf{d}(p,par) \leq c_c\tau^{\ell+(i+1)/k}$
18:            $par^{k\ell+i+1} \leftarrow$ FindNode(high$(par), k\ell+i+1$)
19:            $par^{k\ell+i} \leftarrow$ FindNode(high$(par), k\ell+i$)
20:            $p^{k\ell+i} \leftarrow$ FindNode(high$(p), k\ell+i$)
21:            Add $p^{k\ell+i}$ as a child of $par^{k\ell+i+1}$
```

Theorem 10 *Algorithm 4 turns* $T \in \mathrm{CT}(\tau, c_p, c_c)$ *into* $T' \in \mathrm{CT}(\tau^{1/k}, c_p, c_c)$.

Proof. First, we prove that the algorithm correctly finds parent of each node in T'. Note that q as the current parent of p^ℓ in T may not be the right parent of it in T' because there may exist a node x^ℓ such that $\mathbf{d}(p,x) < \mathbf{d}(p,q)$ and the level of x in T' be greater than the level of p in T'. In this case, p should be inserted as a child of x. To find the right parent of p^ℓ, we search its nearby nodes and select the closest node that satisfies the covering property with constant c_c.

Now, we show that those nodes of T that have appeared for the first time in level ℓ only required to be checked. Let x have appeared for the first time in level $h > \ell$. By the parent property of T, $\mathbf{d}(p,q) < \mathbf{d}(p,x)$, otherwise p should have x as its parent. Therefore, p cannot be closer to x than q and we can ignore x in the search process.

We also show that the right parent of p in T' is in the set of children of relatives of $q^{\ell+1}$. Let x^ℓ serve as the parent of p in T'. So, $\mathbf{d}(p,x) < \mathbf{d}(p,q) \leq c_c\tau^{\ell+1}$. From the previous part, we know that x^ℓ has parent $y^{\ell+1}$ and $x \neq y$. By the triangle inequality,

$$\begin{aligned}\mathbf{d}(q,y) \leq \mathbf{d}(q,p)+\mathbf{d}(p,x)+\mathbf{d}(x,y) &< 3c_p\tau^{\ell+1}\\ &< \frac{3c_c\tau^2}{(\tau-1)^2}\tau^{\ell+1}.\end{aligned}$$

It implies that $y^{\ell+1} \in \mathrm{Rel}(q^{\ell+1})$.

Now, we should find the right level $k\ell+i$ such that insertion of p in that level and as a child of x satisfies both packing and covering conditions with constants c_p and c_c, respectively. So, in this way we guarantee that these constants in T' will be the same as T. □

Theorem 11 *Algorithm 4 has time complexity* $O((\rho^{\lg(\frac{c_c\tau}{c_p(\tau-1)})^2\tau}+k)n)$.

Proof. By Theorem 7 we can augment T in $O(\rho^{\lg(\frac{c_c\tau}{c_p(\tau-1)})^2\tau} n)$ time. From Lemma 3 and Lemma 1, number of nearby nodes to p^ℓ is $O(\rho^{\lg(\frac{c_c\tau}{c_p(\tau-1)})^2\tau})$. Note that for each node $p^\ell \in T$ in this algorithm, level of high(p) in T' is at most $k(\ell+1)$. Therefore, FindNode requires $O(k)$ to return a node which is in a level between $k(\ell+1)$ to $k\ell$ in T'. Also, finding the right interval i requires $O(\lg k)$. Therefore, the time complexity of the refining algorithm is $O((\rho^{\lg(\frac{c_c\tau}{c_p(\tau-1)})^2\tau} + k)n)$. $\square$

6 Conclusion

In this paper, we add an easy to implement condition to cover trees and we show that a cover tree with a large enough scale factor is a net-tree. We also proposed a linear time algorithm to augment nodes of a cover tree with relatives. Furthermore, we present two linear-time algorithms to transform a cover tree to a coarser or finer cover tree. In fact, these two operations are useful to trade-off between the depth and the degree of nodes in a cover tree.

References

[1] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, Sept. 1975.

[2] A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 97–104, 2006.

[3] L. K. Clarkson. Nearest neighbor queries in metric spaces. *Discrete & Computational Geometry*, 22(1):63–93, 1999.

[4] R. Cole and L.-A. 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.

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

[6] J. Gao, L. J. Guibas, and A. Nguyen. Deformable spanners and applications. *Comput. Geom. Theory Appl.*, 35(1-2):2–19, Aug. 2006.

[7] A. Gupta, R. Krauthgamer, and J. R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 534–, 2003.

[8] S. Har-Peled and M. Mendel. Fast construction of nets in low dimensional metrics, and their applications. *SIAM Journal on Computing*, 35(5):1148–1184, 2006.

[9] D. R. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, pages 741–750, 2002.

[10] R. Krauthgamer and J. 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.

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

[12] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 311–321, 1993.