Constructing Hierarchical Trees from Locally Greedy Permutations

Mahmoodreza Jahanseir* Donald R. Sheehy†

1 Introduction

Hierarchical trees are powerful data structures to solve classical computational geometry problems such as approximate nearest neighbor and range search. For Euclidean space, quadtrees and k-d trees are the most famous hierarchical trees, however, they are not applicable for general metric spaces. Cover trees [1] and net-trees [2] are two important data structures for general metric spaces. Cover trees are intended to be simple and practically efficient. On the other hand, net-trees are theoretically more powerful and can be applied to a wider range of problems. However, their construction algorithm has a prohibitively complex preprocessing step. Recently, we [3] showed that cover trees can be transformed to net-trees in linear time. In this paper, we define a new class of permutations called *locally greedy permutations*, and then present an incremental algorithm to construct cover trees from these permutations. Locally greedy permutations are more general than greedy permutations, which are used in [2] to construct net-trees. Our construction algorithm uses a bottom-up approach and it has two main steps of insertion and promotion. We also define semi-compressed cover trees as intermediate structures between compressed and uncompressed cover trees which greatly simplify the algorithm and the analysis.

2 Definitions

In this paper, we assume that the set P of n points is in a doubling metric space. A metric is called *doubling* when it has a constant doubling dimension. The *doubling dimension* is the minimum value of γ such that every ball of radius r can be covered with 2^γ balls of radius $r/2$. A *closed metric ball* centered at a point $p \in P$ with radius $r > 0$ is defined as $\mathrm{B}(p,r) = \{q \in P \mid \mathbf{d}(p,q) \leq r\}$. The *spread* Δ is the ratio of the distance of the farthest points to the closest points.

A *hierarchical tree* is a leveled tree on P such that points are leaves and each point can be associated with many internal nodes. We denote a node associated with a point p in level ℓ by p^ℓ. We use $\mathrm{par}(p^\ell)$ and $\mathrm{ch}(p^\ell)$ to show parent and children of a node p^ℓ, respectively. In these trees, each level is a subset of the level below and we assume that leaves are in level $-\infty$ and the root is in level $+\infty$. Also, each level requires to be an (α,β)-net of the level below or the entire P. An (α,β)-*net* of a set P is $Q \subseteq P$ such that for every two points $p,q \in Q$, $\mathbf{d}(p,q) > \alpha$ and $\mathbf{d}(p, P \setminus Q) \leq \beta$. We use a constant $\tau > 1$, called the *scale factor*, to show the changes of scales between levels. Removing each node that has only one child and it is the only child of its parent, and connecting its parent to its child via a long edge results a compressed hierarchical tree.

Cover trees are hierarchical trees with the following properties: (Packing) for all distinct p,q in a level ℓ, $\mathbf{d}(p,q) > c_p\tau^\ell$, (Covering) for each $r^h \in \mathrm{ch}(p^\ell)$, $\mathbf{d}(p,r) \leq c_c\tau^\ell$ and r^h is the closest node to p among all node with a level greater than ℓ. Here, c_p and c_c are packing and covering constant and we let $0 < c_p \leq c_c$. Similar to [3] we denote all cover trees with the same scale factor, packing constant and covering constant by $\mathrm{CT}(\tau, c_p, c_c)$. In [1], they set $\tau = 2$ and $c_p = c_c = 1$.

Net-trees are also hierarchical trees and for each node p^ℓ we have following properties : (Packing) $\mathrm{B}(p, c_p\tau^\ell) \bigcap P \subset P_{p^\ell}$, (Covering) $P_{p^\ell} \subset \mathrm{B}(p, c_c\tau^\ell)$. The two constants c_p and c_c are defined similarly. Har-peled & Mendel [2] set $c_p = (\tau-5)/2(\tau-1)$ and $c_c = 2\tau/(\tau-1)$. In net-trees, each node p^ℓ has a list of nearby nodes or *relatives*, which is $\mathrm{Rel}(p^\ell) = \{x^f \in T \text{ with } y^g = \mathrm{par}(x^f) \mid f \leq \ell < g, \text{ and } \mathbf{d}(p,x) \leq c_r\tau^\ell\}$. Here, c_r is relative constant and it is usually a function of τ and c_c [3].

A *greedy permutation* is an ordering of the points such that each point p_i is the farthest point in $P \setminus P_{i-1}$, where $P_{i-1} = \{p_1, \ldots, p_{i-1}\}$. The *predecessor* of a point p_i in a permutation is $\mathrm{pred}(p_i)$, the closest point to p_i in P_{i-1}. Given $Q \subseteq P$, the *aspect ratio* of a point $p \in Q$ with respect to Q is $\mathrm{aspect}_Q(p) = \max_{q \in \mathrm{Vor}_Q(p)} \mathbf{d}(p,q)/\mathbf{d}(p, Q \setminus \{p\})$, where $\mathrm{Vor}_Q(p) = \{x \in P \setminus Q \mid \mathbf{d}(p,x) = \mathbf{d}(x,Q)\}$. We define the aspect ratio of $Q \subseteq P$ to be the maximum aspect ratio among all points. Given a constant $\delta \geq 0$, a permutation is *δ-locally greedy* if it has aspect ratio at most δ. For example, a greedy permutation is 1-locally.

*University of Connecticut reza@engr.uconn.edu
†University of Connecticut don.r.sheehy@gmail.com

3 Construction

In this section, we propose a linear time, incremental algorithm to construct a cover tree from a locally greedy permutation. In our algorithm, T_i is obtained by insertion of the new point p_i as a child of its predecessor in the lowest level of $T_{i-1} \in \mathrm{CT}(\tau, c_p, c_c)$. This insertion could violate the covering property. In other words, for some level ℓ and $c'_c > c_c$, we may have $c_c\tau^{\ell+1} < \mathbf{d}(p_i^\ell, \mathrm{par}(p_i^\ell)) \le c'_c\tau^{\ell+1}$. To restore the covering property, we promote the new node to higher levels until we find a node that covers the promoted node with the original covering constant. Insertion of a new point may require $O(\log \Delta)$ time, and a crude analysis results $O(n \log \Delta)$ time in total for the entire construction. Later, we show that the insertion actually requires constant amortized time and the time complexity of construction is $O(n)$.

Before describing the algorithm to restore the covering property, we define a new type of cover tree called a *semi-compressed cover tree*. Semi-compressed cover trees are intermediate structures between uncompressed and compressed cover trees. In these trees, we have an additional condition for a node to be collapsed. Specifically, we collapse a node when it has only one child, it is the only child of its parent, and it has **no relatives besides itself**. We have the following theorem for the size of semi-compressed cover trees. We omit the proof due to lack of space. However, the proof is constructive and from a compressed cover tree.

Theorem 1. *A semi-compressed cover tree on a set of n points has $O(n)$ size.*

Now, we are ready to present the algorithm to restore the covering property with one violating node. Here, we output a tree which is something between a compressed and semi-compressed cover tree, and we call it *partially semi-compressed*. This tree has covering constant c'_c such that $c_c/(\tau-1) \le c'_c < c_c$. In this algorithm, iteration i indicates promotion of node p^ℓ to level $\ell + i$. Note that in each iteration i, we create a node $p^{\ell+i}$ no matter what happens in that iteration. At the beginning, $ancestor = p^\ell$. Here, we set $c_r = c_c(\frac{\tau}{\tau-1})^2\tau$. The i-th iteration of the algorithm is as follows.

1. Find $\mathrm{Rel}(p^{\ell+i})$ from relatives and children of relatives of $\mathrm{par}(\mathrm{par}(ancestor))$.

2. Among relatives and children of relatives of $p^{\ell+i}$ if there is a node in level $< \ell + i$ that is closer to p than its previous parent, then assign that node as a child of $p^{\ell+i}$.

3. Find a node r in $\mathrm{Rel}(\mathrm{par}(\mathrm{par}(ancestor)))$ such that $\mathbf{d}(p, r) < c'_c\tau^{\ell+i+1}$. If such r exists, set it as parent of $p^{\ell+i}$ and quit. Otherwise, set $\mathrm{par}(p^{\ell+i}) = p^{\ell+i+1}$.

4. $i = i + 1$, $ancestor = \mathrm{par}(ancestor)$.

Theorem 2. *Given a cover tree $T \in \mathrm{CT}(\tau \ge 2, c_p, c'_c)$, where $\max\{c_c/(\tau-1), c_p\} \le c'_c < c_c$. A cover tree $T' \in \mathrm{CT}(\tau \ge 2, c_p, c_c)$ can be constructed from T in $O(n)$ time.*

Proof Sketch. Using the previous algorithm, we restore the covering property of T for each violating node p^ℓ. It can be seen that in the i-th iteration of the algorithm, $\mathbf{d}(p, \mathrm{par}(ancestor)) < c_r\tau^{\ell+i}$, which implies p^ℓ has at least one relative besides itself, and it is $\mathrm{par}(ancestor)$. We make this relative responsible to pay the cost of promotion of node p^ℓ to level $\ell + i$. Also note that the final tree T' is a partially semi-compressed cover tree, and by Theorem 1 its size is $O(n)$. Using a charging scheme, we can restore the covering property in $O(n)$ time. We omit the details of the proof due to lack of space. □

Now, we return to the main construction algorithm. For each point p_i in the given δ-locally greedy permutation with $p_j = \mathrm{pred}(p_i)$ and $j < i$, $\mathbf{d}(p_i, p_j) \le \delta\mathbf{d}(p_j, P_{i-1})$. Thus, insertion of p_i in T_{i-1} as a child of p_j in the lowest level above $-\infty$ results $T_i \in \mathrm{CT}(\tau, c_p, \delta c_c)$. To return the covering property to the original covering constant c_c, we need to run the previous algorithm for $\log_\tau \delta$ times because by each run of the algorithm, the covering constant is decreased by a factor of τ. The following theorem summarizes all the results.

Theorem 3. *Given a set P of n points in a δ-locally greedy permutation, where $\delta > 1$. A cover tree $T \in \mathrm{CT}(\tau \ge 2, c_p, c_c)$ can be constructed from P in $O(n \log_\tau \delta)$ time, where $0 < c_p \le c_c$.*

References

[1] 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.

[2] 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.

[3] M. Jahanseir and D. R. Sheehy. Transforming hierarchical trees on metric spaces. In *Proceedings of the 28th Canadian Conference on Computational Geometry*, 2016.