

Notes on the Net–Tree Construction Algorithm

Anthony Man–Cho So*

June 12, 2006

The purpose of this note is to fill in some of the details of the net–tree construction algorithm of Har–Peled and Mendel [2]. We begin with some definitions.

Definition 1 *Let \mathcal{M} be a metric space, and let $P \subset \mathcal{M}$ be a finite subset of points. Clearly, P is also a metric space.*

- (a) *The **ball** of radius $r \geq 0$ around $p \in \mathcal{M}$ is given by $B(p, r) = \{q \in \mathcal{M} : d_{\mathcal{M}}(p, q) \leq r\}$.*
- (b) *The **spread** $\Phi(P)$ of P is given by $\Phi(P) = \frac{\max_{p \neq q \in P} d_{\mathcal{M}}(p, q)}{\min_{p \neq q \in P} d_{\mathcal{M}}(p, q)}$.*
- (c) *The **doubling constant** $\lambda(\mathcal{M})$ of a metric space \mathcal{M} is defined as the minimum number $m \in \mathbb{N}$ such that every ball in \mathcal{M} can be covered by at most m balls of at most half the radius.*
- (d) *The **doubling dimension** $\dim(\mathcal{M})$ of \mathcal{M} is given by $\dim(\mathcal{M}) = \log_2 \lambda(\mathcal{M})$.*

It is clear that $\lambda(P) \leq \lambda(\mathcal{M})$ for any finite subset $P \subset \mathcal{M}$. The following proposition is also immediate from the definitions.

Proposition 1 *Let \mathcal{M} and P be as above.*

- (a) *Suppose that P has spread at most Φ . Then, we have $|P| \leq \lambda^{O(\log \Phi)}$.*
- (b) *Let $r^*(P, m)$ be the radius of the smallest ball in P (whose center is also in P) containing m points, and suppose that \mathcal{M} has doubling constant λ . Then, any ball of radius $2r$, where $r \leq 2r^*(P, m)$, contains at most $\lambda^2 m$ points.*

Proof

- (a) Let $r_{max} = \max_{p \neq q \in P} d_P(p, q)$ and $r_{min} = \min_{p \neq q \in P} d_P(p, q)$. Let $p^* \in P$ be arbitrary. Clearly, we have $P \subset B(p^*, r_{max})$. Let i be the smallest integer such that $r_{max} \leq 2^i r_{min}$. Then, we have $\log_2(r_{max}/r_{min}) = \log_2(\Phi(P)) \leq i < \log_2(\Phi(P)) + 1$. By definition of the doubling constant, the ball $B(p^*, r_{max})$ can be covered by at most $\lambda^{i+1} < \lambda^{\log_2(\Phi(P))+2}$ balls of radius at most $r_{max}/2^{i+1}$. However, for any $p \in P$, we have $B(p, r_{max}/2^{i+1}) \cap P = \{p\}$. It follows that $|P| \leq \lambda^{O(\log \Phi)}$, as desired.
- (b) By definition of the doubling constant, every ball of radius $2r$ can be covered by λ^2 balls of radius $r^*(P, m)$. However, each of the latter balls contains at most m points, and the statement follows.

*Dept. of Computer Science, Stanford University, Stanford, CA 94305. E-mail: manchoso@cs.stanford.edu

□

An important algorithmic tool for problems with metric structures is the hierarchical nets. Roughly speaking, hierarchical nets are sequences of larger and larger subsets of some underlying set P , such that in any given resolution, there is a subset in this sequence that represents well the structure of P in that resolution. Formally, an (r, α) -net in a metric space \mathcal{M} is a subset $\mathcal{N} \subset \mathcal{M}$ of points such that it satisfies (i) a covering property, i.e. $\sup_{x \in \mathcal{M}} d_{\mathcal{M}}(x, \mathcal{N}) \leq r$; and (ii) a packing property, i.e. $\inf_{x, y \in \mathcal{N}; x \neq y} d_{\mathcal{M}}(x, y) \geq r/\alpha$ for some $\alpha \geq 1$. We are interested in nets in a finite metric space with a hierarchical structure. This motivates the following definition.

Definition 2 *Let $P \subset \mathcal{M}$ be a finite subset, and let $\tau \geq 11$ be a packing constant. A net-tree of P is a tree T with the following properties:*

- (a) *The set of leaves is P .*
- (b) *Let $P_v \subset P$ be the set of leaves in the subtree rooted at $v \in T$. We associate with each vertex $v \in T$ a point $rep_v \in P_v$.*
- (c) *Each internal vertex of T has at least two children.*
- (d) *Each vertex $v \in T$ has a level $l(v) \in \mathbb{Z} \cup \{-\infty\}$. The levels satisfy $l(v) < l(\bar{p}(v))$, where $\bar{p}(v)$ is the parent of v in T . If $v \in T$ is a leaf, then $l(v) = -\infty$.*
- (e) **(Covering Property)** *For every vertex $v \in T$, we have:*

$$B\left(rep_v, \frac{2\tau}{\tau-1}\tau^{l(v)}\right) \supset P_v$$

- (f) **(Packing Property)** *For every non-root vertex $v \in T$, we have:*

$$B\left(rep_v, \frac{\tau-5}{2(\tau-1)}\tau^{l(\bar{p}(v))-1}\right) \cap P \subset P_v$$

- (g) **(Inheritance Property)** *For every non-leaf vertex $u \in T$, there exists a child $v \in T$ of u such that $rep_u = rep_v$.*

The reason for the choices of parameters above will become clear in the proof of Theorem 6. For now, we can think of the net-tree as a representation of nets from all scales in the following sense.

Theorem 1 *Given a net-tree T and a real number $t \in \mathbb{R}$, let $\mathcal{N}(t) = \{rep_u : l(u) \leq t < l(\bar{p}(u))\}$. Then, for any distinct $p, q \in \mathcal{N}(t)$, we have $d_P(p, q) \geq 3\tau^{t-1}/10$. Moreover, we have $P \subset \cup_{p \in \mathcal{N}(t)} B(p, 3\tau^t)$. In other words, $\mathcal{N}(t)$ is an $(3\tau^t, 10\tau)$ -net.*

Proof Let $t \in \mathbb{R}$ be fixed, and consider distinct $p, q \in \mathcal{N}(t)$. Let u and v be the corresponding vertices in T , respectively. Consider the balls $B_p \equiv B(p, r_p)$ and $B_q \equiv B(q, r_q)$, where:

$$r_p = \frac{\tau-5}{2(\tau-1)}\tau^{l(\bar{p}(u))-1}, \quad r_q = \frac{\tau-5}{2(\tau-1)}\tau^{l(\bar{p}(v))-1}$$

By definition of the net-tree, we have $B_p \cap P \subset P_u$ and $B_q \cap P \subset P_v$. Moreover, the vertices u and v must be on different branches of T , for otherwise we would have $lca(u, v) \in \{u, v\}$, i.e. one

is the ancestor of the other, which contradicts the definition of $\mathcal{N}(t)$. Hence, we conclude that $P_u \cap P_v = \emptyset$, which in turn implies that:

$$d_P(p, q) \geq \max\{r_p, r_q\} \geq \frac{\tau - 5}{2(\tau - 1)} \tau^{t-1} \geq \frac{3}{10} \tau^{t-1}$$

On the other hand, consider the set $V(t) = \{u \in T : l(u) \leq t < l(\bar{p}(u))\}$. For any $v \in V(t)$, we have:

$$P_v \subset B\left(\text{rep}_v, \frac{2\tau}{\tau - 1} \tau^{l(v)}\right) \subset B(\text{rep}_v, 3\tau^t)$$

since $\tau \geq 11$. It follows that $P \subset \cup_{v \in V(t)} B(\text{rep}_v, 3\tau^t) = \cup_{p \in \mathcal{N}(t)} B(p, 3\tau^t)$ as required. \square

Remark: A similar statement holds for the set $\mathcal{N}'(t) = \{\text{rep}_u : l(u) < t \leq l(\bar{p}(u))\}$.

The following property is a direct consequence of the covering and packing properties of a net-tree.

Proposition 2 *Suppose that the n -point metric space P has doubling constant λ . Then, each vertex of the net-tree has at most $\lambda^{O(1)}$ children.*

Proof Let $u \in T$ be a vertex of some net-tree T , and let v_1, \dots, v_k be its children. We would like to bound k . Clearly, we have $P_u = \cup_{i=1}^k P_{v_i}$. By the covering and packing properties, we have:

$$B\left(\text{rep}_u, \frac{2\tau}{\tau - 1} \tau^{l(u)}\right) \supset P_u = \bigcup_{i=1}^k P_{v_i} \supset \biguplus_{i=1}^k B\left(\text{rep}_{v_i}, \frac{\tau - 5}{2(\tau - 1)} \tau^{l(\bar{p}(v_i)) - 1}\right) \cap P$$

Since $l(u) = l(\bar{p}(v_i))$, it follows that $k \leq \lambda^{O(\log(4\tau^2/(\tau-5)))} = \lambda^{O(1)}$, as desired. \square

Our goal is to design an algorithm for constructing a net-tree whose running time does not depend on the spread. The following lemma is a first step in this direction. It allows us to find a sparse separating ring fast.

Proposition 3 *Given an n -point metric space P with doubling constant λ , there exists a randomized algorithm that computes a point $p \in P$ and a radius $r > 0$ such that:*

$$|P \cap B(p, r)| \geq \frac{n}{2\lambda^3} \quad \text{and} \quad |P \cap B(p, 2r)| \leq \frac{n}{2}$$

Moreover, the expected running time of the algorithm is $O(\lambda^3 n)$.

Proof The algorithm is extremely simple:

Algorithm 1 Procedure SEPRING(P)

Input: An n -point metric space P .

Output: A point $p \in P$ and a radius $r > 0$ such that $|P \cap B(p, r)| \geq \frac{n}{2\lambda^3}$ and $|P \cap B(p, 2r)| \leq \frac{n}{2}$.

- 1: pick randomly a point $p \in P$ and compute the ball $B(p, r)$ of smallest radius around p containing at least $n/(2\lambda^3)$ points
 - 2: **if** $|P \cap B(p, 2r)| \leq n/2$ **then**
 - 3: return
 - 4: **else**
 - 5: repeat
 - 6: **end if**
-

We claim that the algorithm succeeds with constant probability in each iteration. Indeed, consider the smallest ball $Q = P \cap B(q, r^*(P, m))$ that contains at least $m \equiv n/(2\lambda^3)$ points of P . By definition, any ball of radius $r^*(P, m)/2$ contains less than m points. Now, with probability $1/(2\lambda^3)$, our sample p is from Q . However, if $p \in Q$, then we have $r \leq 2r^*(P, m)$, and the ball $B(p, 4r^*(P, m))$ can be covered by at most λ^3 balls of radius $r^*(P, m)/2$ by the doubling property. Hence, by Proposition 1(c), we have $|P \cap B(p, 2r)| < \lambda^3 m \leq n/2$. In particular, this implies that the algorithm succeeds with probability $1/(2\lambda^3)$ in each iteration, and with constant probability after $O(\lambda^3)$ iterations. Since each iteration takes $O(n)$ time, the proposition follows. \square

Our main theorem is the following:

Theorem 2 ([2]) *Given an n -point metric space P , one can construct a net-tree for P in $2^{O(\dim(P))} \cdot n \log n$ expected time.*

The proof of Theorem 2 consists of four steps.

Step 1: Compute a greedy clustering for a point set with bounded spread.

We use Gonzalez's greedy algorithm [1], denoted by GREEDYCLUSTER, to compute a permutation $\Pi = \{p_1, \dots, p_n\}$ of P and numbers r_1, \dots, r_n with $r_n = 0$ such that for any $k \geq 1$, we have $P \subset \bigcup_{j=1}^k B(p_j, r_k)$ and $\min_{1 \leq i < j \leq k} d_P(p_i, p_j) = r_{k-1}$. Specifically, the algorithm GREEDYCLUSTER is given as follows:

Algorithm 2 Procedure GREEDYCLUSTER(P, T)

Input: An n -point metric space P , a parameter T such that $2 \leq T \leq n$.

Output: A sequence of distinct points $\{p_1, \dots, p_T\}$ in P and numbers r_1, \dots, r_T such that for any $k \in \{1, \dots, T\}$, we have $P \subset \bigcup_{j=1}^k B(p_j, r_k)$ and $\min_{1 \leq i < j \leq k} d_P(p_i, p_j) = r_{k-1}$.

```

1: pick an arbitrary point in  $P$  to be  $p_1$ , set  $r_1 \leftarrow \max_{q \in P} d_P(q, p_1)$ , and set  $c_{q,1} \leftarrow p_1$  for all
    $q \in P$  //  $c_{q,k}$  is the center of  $q \in P$  at the end of the  $k$ -th iteration
2: for each  $k = 2, \dots, T$  do
3:   // beginning of the  $k$ -th iteration
4:   for each  $q \in P$  do
5:     let  $\alpha_q^k \leftarrow d_P(q, c_{q,k-1})$ 
6:   end for
7:   set  $p_k \leftarrow \arg \max_{q \in P} \alpha_q^k$  // add a new center
8:   for each  $q \in P$  do
9:     if  $d_P(q, p_k) \leq \alpha_q^k$  then
10:      set  $c_{q,k} \leftarrow p_k$  // update the center of  $q \in P$ 
11:    end if
12:  end for
13:  set  $r_k \leftarrow \max_{q \in P} \min_{1 \leq i \leq k} d_P(q, p_i)$  // current maximum radius of a cluster
14:  // end of the  $k$ -th iteration
15: end for

```

It is routine to show that the algorithm is correct. To facilitate the analysis, we divide the execution of the algorithm into phases. A phase starting at the beginning of the i -th iteration ($i \geq 1$) terminates at the end of the j -th iteration, where j is the integer such that $r_j > r_i/2$ and $r_{j+1} \leq r_i/2$. We now make two easy observations:

- At the end of the k -th iteration ($k = 1, \dots, T$), the center $c_{q,k}$ of $q \in P$ is given by p_{i^*} , where $i^* = \arg \min_{1 \leq i \leq k} d_P(q, p_i)$. In particular, we have $d_P(q, c_{q,k}) \leq r_k$.

- By definition, we have $r_{k-1} \geq \alpha_q^k = \min_{1 \leq i \leq k-1} d_P(q, p_i)$ for all $q \in P$. In particular, if $d_P(q, p_k) > r_{k-1}$, then we must have $\alpha_q^{k+1} = \alpha_q^k$. This shows that we only need to consider those points $q \in P$ that are at a distance of at most r_{k-1} from p_k when we update α_q^{k+1} in line 5.

These observations suggest the following approach for implementing GREEDYCLUSTER:

Data Structures:

- Associate with each center p_i the set of points in P that it serves. Such a set will be called the *cluster* of p_i . Note that p_i is in the cluster of p_i .
- Maintain a max-heap in which every center p_i maintains the point $p'_i \in P$ furthest away from p_i in its cluster along with its current $\alpha_{p'_i} = d_P(p_i, p'_i)$ value.
- For each center p_i , maintain a friends list that contains all the *centers* that are at a distance of at most $6r_k$ from it at the end of the k -th iteration, where $k \geq i$.
- For each center p_i , maintain also its serving center p''_i two phases ago (at the end of that phase) and the friends list of p''_i at that time.

Update Rules:

At the k -th iteration, do the following:

- (**Find the New Center**) Extract the maximum value from the heap, and set p_k to be the new center. Let c_{p_k} be the closest point among $\{p_1, \dots, p_{k-1}\}$ to p_k .
- (**Update α Values**) Scan all the points currently served by the center c_{p_k} or by the centers in the friends list of c_{p_k} and update their α values. In addition, update the current clustering radius r_k .
- (**Update Clusters**) Move the relevant points in P to the newly formed cluster (i.e. the cluster whose center is p_k) and update the points p'_i (of maximum distance from p_i in its cluster) for all p_i in the friends list of c_{p_k} .
- (**Update Friends Lists**) Scan the old friends list of p''_k (i.e. the list that is two phases ago) and do the following:
 - If a scanned point p_j is at a distance of at most $6r_k$ from p_k , then add p_j to the friends list of p_k and p_k to the friends list of p_j .
 - Now, for each of the points p_j in the old friends list of p''_k , scan the *current* friends list of p_j . If a scanned point $p_{j,l}$ is at a distance of at most $6r_k$ from p_k , then add $p_{j,l}$ to the friends list of p_k and p_k to the friends list of $p_{j,l}$.

For the first two phases, we simply perform a brute-force calculation.

We now establish some properties of the above algorithm.

Proposition 4 *For all $i \geq 1$, the size of the friends list of p_i is at most $2\lambda^3$.*

Proof Consider a fixed $i \geq 1$, and let $k > i$ be such that $r_k \leq r_i/2$. Clearly, the metric space $P'_k \equiv \{p_1, \dots, p_k\}$ has doubling constant at most λ . Now, define $B'_k(p_j, r) = \{p \in P'_k : d_P(p, p_j) \leq r\}$. The size of p_i 's friends list at the end of the j -th iteration ($j \geq i$) is then $|B'_j(p_i, 6r_j)|$. Clearly, we have $B'_k(p_i, 6r_k) \subset B'_k(p_i, 6r_i) \subset \cup_{j=1}^k B'_k(p_j, r_k)$. Since the r_i 's are non-increasing, it follows that $|B'_k(p_j, r_k)| \leq 2$ for $j = 1, \dots, k$. In particular, this implies that we have $|B'_k(p_i, 6r_k)| \leq 2\lambda^{\log_2 12} \leq 2\lambda^4$ by definition of the doubling constant. Moreover, observe that for any $j = i, \dots, k-1$, we have $|B'_j(p_i, 6r_j)| \leq |B'_k(p_i, 6r_i)| \leq 2\lambda^4$, since we have $B'_j(p_i, 6r_j) \subset B'_j(p_i, 6r_i) \subset B'_k(p_i, 6r_i)$. Thus, the claim follows. \square

Proposition 5 *The values $\{\alpha_q\}_{q \in P}$ are updated correctly at each iteration.*

Proof Suppose that we have just obtained a new center p_k , where $k \geq 2$. We need to show that all points $q \in P$ with $d_P(q, p_k) \leq r_{k-1}$ are scanned by the algorithm. Let $q \in P$ be such a point. If q is served by c_{p_k} , then we are done. Otherwise, let $c_q \in \{p_1, \dots, p_{k-1}\}$ be the center of the cluster serving q just before p_k is chosen. By definition of the algorithm, we have $d_P(c_q, q) \leq r_{k-1}$ and $d_P(c_{p_k}, p_k) = r_{k-1}$. Thus, by the triangle inequality, we have:

$$d_P(c_q, c_{p_k}) \leq d_P(c_q, q) + d_P(q, p_k) + d_P(p_k, c_{p_k}) \leq 3r_{k-1}$$

In other words, c_q is in the friends list of c_{p_k} , and hence q is scanned by the algorithm. \square

Corollary 1 *The points p'_i (of maximum distance from p_i in its cluster) are updated correctly.*

Proof We need to show that if a center p_i ($i < k$) does not belong to the friends list of c_{p_k} , then the furthest point p'_i from p_i that is served by p_i remains unchanged. Suppose that p'_i is no longer served by p_i . Then, it has to be served by p_k . As such, we must have $d_P(p'_i, p_k) \leq r_k$. In addition, we have $d_P(p_k, c_{p_k}) = r_{k-1}$ and $d_P(p'_i, p_i) \leq r_{k-1}$ (since p'_i is served by p_i just before p_k is chosen). Thus, by the triangle inequality, we have $d_P(p_i, c_{p_k}) \leq 2r_{k-1} + r_k \leq 3r_{k-1}$, i.e. p_i belongs to the friends list of c_{p_k} , which is a contradiction. \square

Proposition 6 *The friends lists are correctly updated at each iteration.*

Proof We proceed by induction on the number of iterations k . For the base case, if k is within the first two phases, then the statement trivially holds by our update rule. To carry out the inductive step, it suffices to check that the set $\{p_i : i < k, d_P(p_i, p_k) \leq 6r_k\}$ is scanned by the algorithm. Indeed, consider a fixed p_i with $i < k$ such that $d_P(p_i, p_k) \leq 6r_k$. Let p''_k be the center of p_k two phases ago with radius r''_k . Then, by definition, we have $r_k \leq r''_k/2$ and $r_k > r''_k/4$, or equivalently, $2r_k \leq r''_k < 4r_k$. Now, suppose that p_i is added before p''_k . Since we have:

$$d_P(p_i, p''_k) \leq d_P(p_i, p_k) + d_P(p_k, p''_k) \leq 6r_k + r''_k \leq 6r''_k$$

it follows that p_i is on the old friends list of p''_k . Henceforth, we shall assume that p_i is added after p''_k . Let S be the set of centers added up to (and including) p''_k , and let $S' = \{p_1, \dots, p_{k-1}\} \setminus S$. We claim that $d_P(q, S) < 4r_k$ for all $q \in S'$. Indeed, consider the first center $q' \in S'$ added after p''_k . By definition of the algorithm, we have $r''_k = d_P(q', S) \geq d_P(q, S)$ for all $q \in S'$. This yields $d_P(q, S) \leq r''_k < 4r_k$ for all $q \in S'$ as required.

Now, the claim implies that there exists an $p_j \in S$ such that $d_P(p_i, p_j) < 4r_k$. By the inductive hypothesis, the *current* friends list of p_j contains p_i . Note that p_j is either p''_k or is added before

p_k'' . If it is the latter, then it remains to show that p_j is in the old friends list of p_k'' . Indeed, we have:

$$d_P(p_j, p_k'') \leq d_P(p_j, p_i) + d_P(p_i, p_k) + d_P(p_k, p_k'') < 4r_k + 6r_k + r_k'' \leq 6r_k''$$

This establishes the inductive step and hence completes the proof. \square

We are now ready to establish the following theorem and finish Step 1.

Theorem 3 *Let P be an n -point metric space with doubling constant λ and spread Φ . Then, the greedy permutation for P can be computed in $O(\lambda^{O(1)}n \log(\Phi n))$ time and $O(\lambda^{O(1)})$ space.*

Proof Consider a phase starting at the i -th iteration and ending at the j -th iteration. Let us first focus on Step (b) of the update rules. Note that a point $q \in P$ can be scanned if there is a center located at a distance of at most $7r_i$ from q . Now, by the argument in Proposition 4, a ball of radius $7r_i$ around each point $q \in P$ contains at most $2\lambda^4$ of the centers p_1, \dots, p_j . Thus, every point in P is being scanned at most $2\lambda^4$ times within each phase of the algorithm. Since there are at most $O(\log \Phi)$ phases, it follows that Step (b) takes $O(\lambda^{O(1)}n \log \Phi)$ time overall.

Steps (a) and (c) involve maintaining the max-heap. The total time required is $O(\lambda^{O(1)}n \log n)$, since only $\lambda^{O(1)}$ values in the head are changed in each iteration.

Lastly, for Step (d), observe that for each center p_l , we need to scan at most $\lambda^{O(1)}$ centers in the relevant friends lists. Hence, Step (d) takes $O(\lambda^{O(1)}n)$ time overall.

The time bound in the theorem now follows immediately from the above discussion. The space bound also follows easily from the construction. This completes the proof. \square

Step 2: Removing the dependence on spread: the case of finite ultrametrics.

One way to remove the spread from the running time of the clustering algorithm is to restrict the type of input metric spaces.

Definition 3 *An ultrametric is a metric space $\mathcal{M} = (X, d)$ such that for every $x, y, z \in X$, we have $d(x, z) \leq \max\{d(x, y), d(y, z)\}$.*

Definition 4 *A Hierarchically Well-Separated Tree (HST) is a metric space defined on the leaves of a rooted tree T . Each vertex $u \in T$ is associated with a label $\Delta_u \geq 0$ such that $\Delta_u = 0$ iff u is a leaf of T , and a representative rep_u which is an arbitrary leaf of the subtree rooted at u . The labels are such that if a vertex u is a child of another vertex v , then $\Delta_u \leq \Delta_v$. Moreover, we have $\text{rep}_u \in \{\text{rep}_v : v \text{ is a child of } u\}$. The distance between two leaves $u, v \in T$ is defined as $\Delta_{\text{lca}(u, v)}$, where $\text{lca}(u, v)$ is the least common ancestor of u and v in T .*

Without loss of generality, we shall assume that the HST is binary.

Proposition 7 *The class of HSTs coincides with the class of finite ultrametrics.*

Proof Folklore. \square

Given an HST T , let P be the set of leaves of T , and let d'_P be the metric induced by the labels $\{\Delta_u\}_{u \in T}$. It turns out that we can remove the dependence on spread by using the HST to guide our choice of centers in the clustering algorithm. In particular, we will apply the clustering algorithm to a dynamic set of points (more precisely, to a set of points that will grow as the algorithm progresses) that will correspond to a level of the HST. Specifically, the new clustering algorithm, which we shall call HSTCLUSTER, works as follows. To initialize, let u_r be the root of the HST. We set $p_1 = \text{rep}_{u_r}$ to be the first center. The initial point set will consist of the point

$\{\text{rep}_{u_r}\}$, and we set $r_{curr} = 0$. The parameter r_{curr} keeps track of the maximum cluster radius of the *current* point set. We use a max-heap \mathcal{H}_1 to maintain the α values of each point (recall that α_p is the distance of p to its center) in the *current* point set. We also use a max-heap \mathcal{H}_2 to maintain the nodes of the HST sorted by their diameters Δ .

In a general step of the algorithm, we extract from both heaps. Let q be the result of the extraction from \mathcal{H}_1 with corresponding radius r_{curr} , u be the node in the HST that is extracted from \mathcal{H}_2 , and $\bar{p}(u)$ be the parent of u in the HST. For some fixed constant $\alpha > 0$, consider the following two possibilities:

- (a) $\Delta_u > r_{curr}/n^\alpha$. We replace u with its two children v and w . Specifically, if rep_u belongs to the cluster of p_i , then we remove rep_u from and add $\text{rep}_v, \text{rep}_w$ to the list of points associated with p_i . Next, we compute the nearest center of the new point, i.e. compute α_{rep_v} and α_{rep_w} , by scanning the friends list of p_i . Lastly, we insert $\{\text{rep}_l\}$ ($l \in \{v, w\}$) into \mathcal{H}_1 if $\alpha_{\text{rep}_l} > 0$, and remove $\{\text{rep}_u\}$ from \mathcal{H}_1 . Note that the current point set is now changed and consists of the points in the updated heap \mathcal{H}_1 , together with the current centers.
- (b) $\Delta_u \leq r_{curr}/n^\alpha \leq \Delta_{\bar{p}(u)}$. Set $p_{k+1} = q$ to be the next center. Then, update the clusters, the α values and the friends lists as before.

We need to show that Step (a) above correctly computes the values α_{rep_v} and α_{rep_w} . This is achieved by the following proposition.

Proposition 8 *Let $\Pi = \{p_1, \dots, p_n\}$ be the permutation of P generated by HSTCLUSTER, and let $r_k = \min_{i=1}^k d'_P(p_{k+1}, p_i)$ for $k = 1, \dots, n$. Then, for all $k = 1, \dots, n$, we have $P \subset \cup_{i=1}^k B(p_i, (1 + n^{-\alpha})r_k)$, and that for any $u, v \in \{p_1, \dots, p_k\}$, we have $d'_P(u, v) \geq (1 + n^{-\alpha})r_k$.*

Proof Consider a fixed k . It is clear that balls of radius r_k around p_1, \dots, p_k cover all the points in the *current* point set. Now, observe that every point q in the current point set is the representative of some node $u(q)$ in the HST. Thus, the distance from q to (the representative of) any leaf in the subtree rooted at $u(q)$ is at most $\Delta_{u(q)} \leq r_k/n^\alpha$. It follows that $P \subset \cup_{i=1}^k B(p_i, (1 + n^{-\alpha})r_k)$. In particular, this implies that in order to compute the values α_{rep_v} and α_{rep_w} in Step (a), it suffices to scan the friends list of p_i .

Now, observe that the above argument also implies that for any $1 \leq i < j \leq n$, we have $(1 + n^{-\alpha})r_i \geq r_j$. In particular, let $i \leq k$ be the minimum number such that $u, v \in \{p_1, \dots, p_i\}$. Then, we have $d'_P(u, v) \geq r_{i-1} \geq (1 + n^{-\alpha})^{-1}r_k \geq (1 - n^{-\alpha})r_k$ as desired. \square

We now proceed to establish the running time of HSTCLUSTER for the case of finite ultrametrics.

Theorem 4 *The running time of HSTCLUSTER is $O(\lambda^{O(1)}n \log n)$.*

Proof As before, we divide the execution of the algorithm into phases. In the i -th phase, the algorithm handles new clusters with radii in the range $(\text{diam}(P) \cdot 2^{-i}, \text{diam}(P) \cdot 2^{-(i-1)})$. Now, consider a point $p \in P$, and suppose that it is inserted into our dynamic point set when a node v in the HST is split in phase i (in particular, p is the representative of one of the children of v). Let p, q be the two representatives of the two children of v . We shall charge v for any work done on p and q for the next $L = \beta \log n$ phases, where $\beta > 2\alpha$ is some fixed constant.

To determine the amount of work charged to v , consider first the work done on p before it undergoes another split event. Using the same arguments as in Step 1, one can show that within

each phase, the amount of work done on p is $\lambda^{O(1)}$. Thus, if p is at most L phases away from the split event of v , then the node v is charged at most $\lambda^{O(1)}L = \beta\lambda^{O(1)}\log n$ amount of work.

Now, suppose that p is at $j > L$ phases away from the split event of v . Let r_{curr} be the current cluster radius, and let r_{old} be the cluster radius when the node v is split. Since p is not yet split by the algorithm, we see that p represents a set of points with diameter at most r_{curr}/n^α . Also, since v is split, we have $r_{old} < n^\alpha\Delta_v$. Since p is now at $j > L$ phases away, we conclude that $r_{curr} < r_{old} \cdot 2^{-L} < n^{\alpha-\beta}\Delta_v$. In particular, this implies that:

$$P \cap B(p, r_{curr} \cdot n^\alpha) \subset P \cap B(p, n^{2\alpha-\beta}\Delta_v) \subset P \cap B(p, r_{curr}/n^\alpha) \quad (1)$$

where the second inclusion follows from the following observation: if p' is a leaf in the subtree rooted at p and q' is a leaf of the HST that is not in the subtree rooted at p , then $d'_P(p', q') \geq \Delta_v$, which, by our choice of α and β , implies that the set $P \cap B(n^{2\alpha-\beta}\Delta_v)$ contains only those leaves that are in the subtree rooted at p . Now, observe that (1) implies that all the points that p represents are polynomially far (in terms of r_{curr}) from the rest of the points of P . In particular, the cluster that p represents cannot be in any updated friends list in the current phase. It follows that p does not require any work from the algorithm until it undergoes a split event.

Thus, we have shown that every node in the HST is charged at most $O(\lambda^{O(1)}\log n)$ amount of work. Since there are $O(n)$ nodes in the HST, it follows that the overall running time of HSTCLUSTER is $O(\lambda^{O(1)}n\log n)$. \square

Step 3: Removing the dependence on spread: the general case.

For a general n -point metric space P with doubling constant λ , we show that it can be embedded into an HST with certain distortion. This will then allow us to apply the algorithm in Step 2 to the resulting HST. We first state a definition.

Definition 5 *Given two metric spaces \mathcal{M} and \mathcal{N} , we say that \mathcal{N} is an t -approximation of \mathcal{M} if for all $u, v \in \mathcal{M}$, we have $d_{\mathcal{M}}(u, v) \leq d_{\mathcal{N}}(u, v) \leq t \cdot d_{\mathcal{M}}(u, v)$.*

Theorem 5 ([2]) *For an n -point metric space P with doubling constant λ , one can construct in $O(\lambda^6 n \log n)$ expected time an HST which is an $5n^2$ -approximation of P . In other words, for any $p, q \in P$, we have $d_P(p, q) \leq d'_P(p, q) \leq 5n^2 \cdot d_P(p, q)$, where d'_P is the metric on the HST.*

Before we prove Theorem 5, let us first point out what needs to be modified in the algorithm given in Step 2. The parameter r_{curr} now refers to the cluster radius in the d_P metric. We then have the following version of Proposition 8:

Proposition 8' *Let $\Pi = \{p_1, \dots, p_n\}$ be the permutation of P generated by HSTCLUSTER, and let $r_k = \min_{i=1}^k d_P(p_{k+1}, p_i)$ for $k = 1, \dots, n$. Then, for all $k = 1, \dots, n$, we have $P \subset \cup_{i=1}^k B(p_i, (1 + n^{-\alpha})r_k)$, and that for any $u, v \in \{p_1, \dots, p_k\}$, we have $d_P(u, v) \geq (1 + n^{-\alpha})r_k$.*

The proof is almost identical to that of Proposition 8 and hence will be omitted. It remains to establish the running time of the algorithm.

Theorem 4' *The expected running time of HSTCLUSTER is $O(\lambda^{O(1)}n\log n)$.*

Proof By Theorem 5, we can embed P into the desired HST in $O(\lambda^{O(1)}n\log n)$ expected time. Now, let $\beta \geq 2\alpha + 3$ be a fixed constant. The rest of the proof is similar to that of Theorem 4, except for the part where p is at $j > L$ phases away from the split event of v . As before, p represents a set of points with diameter at most r_{curr}/n^α , and we have $r_{old} < n^\alpha\Delta_v$ and $r_{curr} < n^{\alpha-\beta}\Delta_v$. Thus, it is immediate that $P \cap B(p, r_{curr} \cdot n^\alpha) \subset P \cap B(p, n^{2\alpha-\beta}\Delta_v)$. Now, in

order to show that $P \cap B(p, n^{2\alpha-\beta}\Delta_v) \subset P \cap B(p, r_{curr}/n^\alpha)$, we proceed as follows. Let p' be a leaf in the subtree rooted at p , and let q' be a leaf of the HST that is not in the subtree rooted at p . Then, by the property of the HST, we have $5n^2 \cdot d_P(p', q') \geq d'_P(p', q') \geq \Delta_v$. In other words, we have $\Delta_v/5n^2 \leq d_P(p', q')$. Hence, by our choice of β , we have $P \cap B(p, n^{2\alpha-\beta}\Delta_v) \subset P \cap B(p, \Delta_v/n^3) \subset P \cap B(p, r_{curr}/n^\alpha)$ as desired. \square

To complete Step 3, it remains to prove Theorem 5. We begin with the following proposition.

Proposition 9 *Given a weighted connected graph G on n vertices and m edges, one can construct in $O(n \log n + m)$ time an HST that $(n-1)$ -approximates the shortest path metric on G .*

Proof We sort the edges of T (according to their weights) in non-decreasing order, and add them to the graph one-by-one. The HST is built bottom-up. At each point we have a collection of HSTs, each of which corresponds to a connected component of the current graph. Initially, we have n HSTs, each of which corresponds to a vertex of the graph. When an added edge e merges two connected components, we merge the two corresponding HSTs by adding a new common root r for the two HSTs and labelling this root with $\Delta_r = (n-1)w(e)$, where $w(e)$ is the weight of e . Note that this algorithm is just a slight variation of Kruskal's algorithm for computing the minimum spanning tree of G . As such, it has the same running time, i.e. $O(n \log n + m)$.

Now, let H be the resulting HST. It remains to establish the approximation guarantee of H . Let u, v be two vertices of G , and let e be the first edge that merges the connected components containing u and v into a single connected component C . Note that at that point in time, the edge e is the heaviest edge in C , and hence we have $w(e) \leq d_G(u, v) \leq (|C|-1)w(e) \leq (n-1)w(e)$. Since $d_H(u, v) = (n-1)w(e)$, the proof is completed. \square

At this point one may be tempted to embed the n -point metric P into an HST directly using Proposition 9. However, this will result in an $O(n^2)$ time algorithm. To circumvent this problem, one can first find a good, sparse (i.e. with $o(n^2)$ edges) spanner for P , and then embed this spanner into an HST. The main point here is that there is a tradeoff between the quality of the embedding (as measured by the distortion) and the time required to embed P into an HST.

Proposition 10 *Given an n -point metric space P with doubling constant λ , one can compute a weighted graph G that $5n$ -approximates P in $O(\lambda^6 n \log n)$ time. Moreover, the graph G has $O(\lambda^3 n \log n)$ edges.*

Proof See [2], Lemma 3.5. The proof relies on Proposition 3 in a crucial manner. \square

Theorem 5 now follows easily from Propositions 9 and 10.

Step 4: Constructing the Net-Tree.

The construction of the net-tree T is done by adding the points of P according to the permutation generated by HSTCLUSTER. Let $T^{(k)}$ be the tree constructed for p_1, \dots, p_k . Our goal is to construct $T \equiv T^{(n)}$ by inductively constructing the trees $T^{(1)}, T^{(2)}, \dots$. During the construction, we maintain, for every vertex $u \in T^{(k)}$, a set of “close-by” vertices $\text{Rel}(u)$. The set $\text{Rel}(u)$ is defined implicitly by the algorithm, but as we will show later, it will coincide with the set:

$$\overline{\text{Rel}}(u) = \left\{ v \in T^{(k)} : l(v) \leq l(u) < l(\bar{p}(v)), d_P(\text{rep}_u, \text{rep}_v) \leq 3(1 + n^{-\alpha})^2 \tau^{l(u)} \right\}$$

where $\alpha \geq 2$ is the constant that appears in Steps 2 and 3 above. Let $\bar{r}_i = \min_{1 \leq j \leq i} r_j$. Clearly, the sequence $\{\bar{r}_i\}_i$ is monotonically decreasing. Moreover, by Proposition 8', we have

$r_i \geq \bar{r}_i \geq (1 + n^{-\alpha})^{-1} r_i$. Let $\{p_1, \dots, p_n\}$ be the permutation generated by HSTCLUSTER. The algorithm proceeds by inserting the points p_1, \dots, p_n one-by-one. To facilitate the analysis, we say that a point p_i is inserted in the l -th phase if $\lceil \log_\tau \bar{r}_{i-1} \rceil = l$. At the k -th iteration, the point p_k will be added as a leaf to the tree $T^{(k-1)}$ to form $T^{(k)}$. As such, we set $l(p_k) = -\infty$ and $\text{rep}_{p_k} = p_k$. Let $l = \lceil \log_\tau \bar{r}_{k-1} \rceil$, and let h be the largest index such that $\log_\tau \bar{r}_{h-1} > l$. In other words, the point p_h is the last added center in the previous phase. Let $q \in \{p_1, \dots, p_h\}$ be the closest point to p_k among p_1, \dots, p_h . Upon identifying q with the unique leaf of $T^{(k-1)}$ whose representative is q , we set $u = \bar{p}(q)$ and obtain $T^{(k)}$ as follows:

- (i) If $l(u) > l$, then create a new vertex v and set $l(v) = l$, $\text{rep}_v = q$. Connect q and p_k as children of v and make v a child of u .
- (ii) Otherwise, connect p_k as another child of u .

To initialize, we set $T^{(1)} = \{p_1\}$ with $l(p_1) = -\infty$, $\text{rep}_{p_1} = p_1$ and $\text{Rel}(p_1) = \emptyset$. In addition, $T^{(2)}$ is obtained as follows. First, we create a new vertex v with $l(v) = \lceil \log_\tau \bar{r}_1 \rceil$, $\text{rep}_v = p_1$, and $\text{Rel}(v) = \emptyset$. Then, we connect p_1, p_2 as children of v .

To complete the description of the algorithm, we need to specify how to find q and how to update the sets $\text{Rel}(\cdot)$. We treat each of these two items in turn.

- (iii) (**Find q**) Let c_{p_k} be the closest point among $\{p_1, \dots, p_{k-1}\}$ to p_k , and set $\hat{u} = \bar{p}(c_{p_k})$. We consider the following two cases:

Case 1: If $l(\hat{u}) > l$, then $q = \hat{u}$.

Case 2: Otherwise, we have $l(\hat{u}) = l$. In this case, q must be in the set $\Gamma = \{\text{rep}_w : w \in \text{Rel}(\hat{u})\}$. Thus, we simply pick q to be the nearest neighbor to p_k in Γ .

- (iv) (**Update $\text{Rel}(\cdot)$**) For each newly added vertex x we do the following. Let $y = \bar{p}(x)$. For each $z \in \text{Rel}(y)$ and for each child z' of z , we traverse part of the tree rooted at z' in the following way. When we visit a vertex u , we check whether u should be added to $\text{Rel}(x)$ and whether x should be added to $\text{Rel}(u)$ according to the definition of $\bar{\text{Rel}}(\cdot)$. If x has been added to $\text{Rel}(u)$, then we continue by traversing the children of u . Otherwise, we skip them.

Before we prove the correctness of (iii) and (iv) above, let us first make some easy observations about the algorithm.

Proposition 11 *Suppose that we are at the end of the k -th iteration, where $k \geq 1$. Then, the following hold:*

- (a) *Every point inserted during the l -th phase has its current parent (in $T^{(k)}$) at level l .*
- (b) *Let $q \in \{p_1, \dots, p_h\}$ be the closest point to p_k , where h is the largest index such that $\log_\tau \bar{r}_{h-1} > l \equiv \lceil \log_\tau \bar{r}_{k-1} \rceil$, and let $v = \bar{p}(q)$ (note that v is the parent of q at the end of the k -th iteration). If $l(v) = l$, then $\text{rep}_v = q$.*

Proof We proceed by induction on k . The base case is trivial. For the inductive step, we first observe that (a) follows directly from the update rules. Now, to prove (b), consider the iteration $k' < k$ at which q is inserted. By the result of (a), the parent of q in $T^{(k')}$ has level larger than l . Now, since $l(v) = l$, there must exist a smallest index $j \in \{h+1, \dots, k\}$ such that $\lceil \log_\tau \bar{r}_{j-1} \rceil = l$, and that q is the closest point to p_j among p_1, \dots, p_h . If $j = k$, then by (i) we

have $\text{rep}_v = q$. Otherwise, by (i) again, the parent of q at the end of the j -th iteration has level l , and its representative is q . It then follows from (ii) that the parent of q is unchanged when p_k is inserted. This completes the proof. \square

The correctness of the algorithm can now be established via the following theorem.

Theorem 6 *For $k = 1, 2, \dots, n$, the tree $T^{(k)}$ has the following properties:*

- (a) *If w' is a child of w , then we have $d_P(\text{rep}_w, \text{rep}_{w'}) \leq (1 + n^{-\alpha})^2 \tau^{l(w)}$.*
- (b) *The procedure for finding q is correct.*
- (c) *For every $t \in \mathbb{R}$, every pair of points in $\mathcal{N}(t) = \{\text{rep}_u : l(u) \leq t < l(\bar{p}(u))\}$ is at a distance of at least τ^{t-1} apart.*
- (d) *$T^{(k)}$ is a net-tree of the points $\{p_1, \dots, p_k\}$.*
- (e) *For any $u \in T^{(k)}$, we have $\text{Rel}(u) = \overline{\text{Rel}}(u)$.*

Proof We shall prove all five assertions together by induction on k . For $k = 1$, the assertions are either vacuous or trivial. For $k = 2$, (a) follows from the fact that $d_P(\text{rep}_u, \text{rep}_v) \leq d_P(p_1, p_2) = \bar{r}_1 \leq \tau^{\lceil \log_\tau \bar{r}_1 \rceil} = \tau^{l(u)}$; (b) is vacuous; (c) follows from the fact that for $-\infty < t < \lceil \log_\tau \bar{r}_1 \rceil$, we have $d_P(p_1, p_2) = \bar{r}_1 \geq \tau^{\lceil \log_\tau \bar{r}_1 \rceil - 1} \geq \tau^{t-1}$; (d) follows directly from the definition; and (e) is obvious. To carry out the inductive step, we proceed as follows:

- (a) Consider first the construction in (i). The new vertex v is made a child of $u = \bar{p}(q)$, and the vertices q and p_k are now children of v . Thus, we need to establish the bound for $d_P(\text{rep}_u, \text{rep}_v)$ and $d_P(\text{rep}_v, \text{rep}_{p_k})$. By the inductive hypothesis, we have $d_P(\text{rep}_u, \text{rep}_v) = d_P(\text{rep}_u, q) \leq (1 + n^{-\alpha})^2 \tau^{l(u)}$. To establish the bound for $d_P(\text{rep}_v, p_k) = d_P(q, p_k)$, recall that by Proposition 8', we have $P \subset \cup_{i=1}^h B(p_i, (1 + n^{-\alpha})r_h)$. In particular, we have:

$$d_P(q, p_k) \leq (1 + n^{-\alpha})r_h \leq (1 + n^{-\alpha})^2 \tau^{\log_\tau \bar{r}_h} \leq (1 + n^{-\alpha})^2 \tau^{l(v)}$$

since $\bar{r}_h \geq (1 + n^{-\alpha})^{-1}r_h$ and $\log_\tau \bar{r}_h \leq l = l(v)$ by our choice of h . It follows that $d_P(q, p_k) \leq (1 + n^{-\alpha})^2 \tau^{l(v)}$, as desired.

Now, consider the construction in (ii). Since $\text{rep}_u = q$, we have $d_P(\text{rep}_u, p_k) = d_P(q, p_k) \leq (1 + n^{-\alpha})^2 \tau^{l(u)}$ by the result of the preceding paragraph.

- (b) Suppose first that $l(\hat{u}) > l$. By Proposition 11(a), we conclude that c_{p_k} is inserted before the current phase, which implies that it is indeed the closest point to p_k among $\{p_1, \dots, p_h\}$. Now, suppose that $l(\hat{u}) = l$. By the inductive hypothesis and the result of (a), we have $d_P(\hat{u}, c_{p_k}) \leq (1 + n^{-\alpha})^2 \tau^l$. Furthermore, since $P \subset \cup_{i=1}^{k-1} B(p_i, (1 + n^{-\alpha})r_{k-1})$ and $r_{k-1} \leq r_h \leq (1 + n^{-\alpha})\bar{r}_h$, we have $d_P(c_{p_k}, p_k) \leq (1 + n^{-\alpha})^2 \tau^l$. Hence, we conclude that:

$$d_P(\hat{u}, q) \leq d_P(\hat{u}, c_{p_k}) + d_P(c_{p_k}, p_k) + d_P(p_k, q) \leq 3(1 + n^{-\alpha})^2 \tau^l \quad (2)$$

Now, since q appears before the l -th phase, we either have $l(\bar{p}(q)) > l$, in which case we have $l(q) < l(\hat{u}) = l < l(\bar{p}(q))$, and together with (2) this implies that $q \in \overline{\text{Rel}}(\hat{u})$; or we have $l(\bar{p}(q)) = l$, in which case Proposition 11(b) gives $\text{rep}_{\bar{p}(q)} = q$, and together with (2) and the fact that $T^{(k-1)}$ is a net-tree (so that $l(\bar{p}(q)) < l(\bar{p}(\bar{p}(q)))$) this implies that $\bar{p}(q) \in \overline{\text{Rel}}(\hat{u})$. In either case, we see that q is a representative of a vertex in $\overline{\text{Rel}}(\hat{u})$, and by the inductive hypothesis we have $\overline{\text{Rel}}(\hat{u}) = \text{Rel}(\hat{u})$ in $T^{(k-1)}$. Thus, the procedure for finding q is correct.

- (c) Let $t \in \mathbb{R}$ be fixed, and consider any two vertices u, v for which $\max\{l(u), l(v)\} \leq t < \min\{l(\bar{p}(u)), l(\bar{p}(v))\}$. Clearly, we have $u, v \in \mathcal{N}(t)$. If neither u nor v is p_k , then the claim follows from the inductive hypothesis (even for the newly formed internal vertex, since it inherits its parent and representative from a previously established vertex). Otherwise, suppose without loss of generality that $u = p_k$. Since p_k is the latest leaf added to form $T^{(k)}$, we have $d_P(p_k, \text{rep}_v) \geq \bar{r}_{k-1} \geq \tau^{l-1}$ (recall that $l = \lceil \log_\tau \bar{r}_{k-1} \rceil$). Also, note that $l(\bar{p}(p_k)) = l$ by the construction in (i), which implies that $t < l$. Thus, we conclude that $d_P(\text{rep}_u, \text{rep}_v) = d_P(p_k, \text{rep}_v) \geq \tau^{t-1}$, as desired.

We remark that the same argument can be used to show that for every $t \in \mathbb{R}$, every pair of points in $\mathcal{N}'(t) = \{\text{rep}_u : l(u) < t \leq l(\bar{p}(u))\}$ is at a distance of at least τ^{t-1} apart.

- (d) It suffices to establish the covering and packing properties of $T^{(k)}$. To establish the covering property, we use the result of (a). Let $u = u_1$ be a vertex, $v = u_m$ be a descendant, and $\langle u_1, \dots, u_m \rangle$ be a path between them in $T^{(k)}$. Then, we have:

$$\begin{aligned}
d_P(\text{rep}_u, \text{rep}_v) &\leq \sum_{i=1}^{k-1} d_P(\text{rep}_{u_i}, \text{rep}_{u_{i+1}}) \\
&\leq (1 + n^{-\alpha})^2 \sum_{i=1}^{k-1} \tau^{l(u_i)} && \text{(by the result of (a))} \\
&\leq (1 + n^{-\alpha})^2 \tau^{l(u_1)} \sum_{i=1}^{k-1} \tau^{-(i-1)} && \text{(since } l(u_i) \in \mathbb{Z} \text{ and } l(u_i) > l(u_{i+1})) \\
&\leq \frac{(1 + n^{-\alpha})^2 \tau}{\tau - 1} \tau^{l(u)}
\end{aligned}$$

Since $\alpha \geq 2$, we have $(1 + n^{-\alpha})^2 \leq 2$ for $n \geq 2$, whence $d_P(\text{rep}_u, \text{rep}_v) \leq \frac{2\tau}{\tau-1} \tau^{l(u)}$ as desired.

We now proceed to establish the packing property. Let u be an arbitrary internal vertex of $T^{(k)}$, and let $q \notin P_u$ be a point in P . We need to show that $d_P(q, \text{rep}_u) \geq \frac{\tau-5}{2(\tau-1)} \tau^{\bar{p}(u)-1}$. Let $\hat{q} \in T^{(k)}$ be an ancestor of q such that $l(\hat{q}) < l(\bar{p}(u)) \leq l(\bar{p}(\hat{q}))$. By taking $t = l(\bar{p}(u))$ and applying the result of (c), we conclude that $d_P(\text{rep}_{\hat{q}}, \text{rep}_u) \geq \tau^{l(\bar{p}(u))-1}$. We now consider the following cases:

Case 1: $q = \hat{q}$. Then, our claim trivially holds.

Case 2: $l(\hat{q}) < l(\bar{p}(u)) - 1$. Then, we have:

$$\begin{aligned}
d_P(q, \text{rep}_u) &\geq d_P(\text{rep}_{\hat{q}}, \text{rep}_u) - d_P(\text{rep}_{\hat{q}}, q) \\
&\geq \tau^{l(\bar{p}(u))-1} - \frac{(1 + n^{-\alpha})^2 \tau}{\tau - 1} \tau^{l(\bar{p}(u))-2} && \text{(by the result of (a))} \\
&= \left(1 - \frac{(1 + n^{-\alpha})^2}{\tau - 1}\right) \tau^{l(\bar{p}(u))-1} \\
&\geq \frac{\tau - 3}{\tau - 1} \tau^{l(\bar{p}(u))-1} && \text{(since } (1 + n^{-\alpha})^2 \leq 2)
\end{aligned}$$

which establishes our claim.

Now, let $q' \in T^{(k)}$ be an ancestor of q such that $\hat{q} = \bar{p}(q')$ (i.e. q' is a child of \hat{q}). Note that it is possible that $q = q'$.

Case 3: $\text{rep}_{q'} = \text{rep}_{\hat{q}}$. Then, we have $d_P(q, \text{rep}_u) \geq d_P(\text{rep}_{q'}, \text{rep}_u) - d_P(\text{rep}_{q'}, q)$, and the argument in Case 2 applies here.

Case 4: We have the following situation: $l(\hat{q}) = l(\bar{p}(q')) = l(\bar{p}(u)) - 1$ and $\text{rep}_{q'} \neq \text{rep}_{\bar{p}(q')}$. By the constructions in (i) and (ii), this can only happen if $\text{rep}_{q'}$ is inserted in the $(l(\bar{p}(u)) - 1)$ -st phase. Now, recall that the algorithm connects $\text{rep}_{q'}$ as a child of a vertex in level $l(\bar{p}(u)) - 1$ whose representative is the closest point to $\text{rep}_{q'}$ among those appearing in phases earlier than the $(l(\bar{p}(u)) - 1)$ -st phase. In particular, this implies that $\text{rep}_{\bar{p}(q')}$ is inserted in a phase earlier than the $(l(\bar{p}(u)) - 1)$ -st phase. Moreover, since the level of $\bar{p}(u)$ is $l(\bar{p}(u))$, we see that rep_u is also inserted in a phase earlier than the $(l(\bar{p}(u)) - 1)$ -st phase. Thus, we conclude that $d_P(\text{rep}_{q'}, \text{rep}_{\bar{p}(q')}) \leq d_P(\text{rep}_{q'}, \text{rep}_u)$. Now, we compute:

$$\begin{aligned} d_P(\text{rep}_{q'}, \text{rep}_u) &\geq \max \left\{ d_P(\text{rep}_{q'}, \text{rep}_{\bar{p}(q')}), d_P(\text{rep}_{\bar{p}(q')}, \text{rep}_u) - d_P(\text{rep}_{q'}, \text{rep}_{\bar{p}(q')}) \right\} \\ &\geq \frac{1}{2} d_P(\text{rep}_{\bar{p}(q')}, \text{rep}_u) \\ &\geq \frac{1}{2} \tau^{l(\bar{p}(u))-1} \end{aligned}$$

It follows that:

$$\begin{aligned} d_P(q, \text{rep}_u) &\geq d_P(\text{rep}_{q'}, \text{rep}_u) - d_P(q, \text{rep}_{q'}) \\ &\geq \frac{1}{2} \tau^{l(\bar{p}(u))-1} - \frac{(1 + n^{-\alpha})^2 \tau}{\tau - 1} \tau^{l(\bar{p}(u))-2} \\ &= \left(\frac{1}{2} - \frac{(1 + n^{-\alpha})^2}{\tau - 1} \right) \tau^{l(\bar{p}(u))-1} \\ &\geq \frac{\tau - 5}{2(\tau - 1)} \tau^{l(\bar{p}(u))-1} \end{aligned}$$

as required.

- (e) Suppose that a new vertex x is attached as a child to a vertex y . We need to show that our algorithm scans all vertices w for which either $x \in \overline{\text{Rel}}(w)$ or $w \in \overline{\text{Rel}}(x)$. Suppose first that $x \in \overline{\text{Rel}}(w)$. Then, by definition, we have $l(w) < l(\bar{p}(x)) = l(y)$. Now, let z be an ancestor of w for which $l(z) \leq l(y) < l(\bar{p}(z))$, and let $\langle z_1 = z, \dots, z_m = w \rangle$ be the path between them in $T^{(k)}$. Observe that for $i = 1, \dots, m - 1$, we have:

$$\begin{aligned} d_P(\text{rep}_x, \text{rep}_{z_i}) &\leq d_P(\text{rep}_x, \text{rep}_w) + d_P(\text{rep}_w, \text{rep}_{z_i}) \\ &\leq 3(1 + n^{-\alpha})^2 \tau^{l(w)} + \frac{(1 + n^{-\alpha})^2 \tau}{\tau - 1} \tau^{l(z_i)} \end{aligned} \tag{3}$$

$$\leq 3(1 + n^{-\alpha})^2 \tau^{l(z_i)} \tag{4}$$

where the first term in (3) follows from the fact that $x \in \overline{\text{Rel}}(w)$, the second term in (3) follows from the result of (d), and (4) follows from the facts that $l(w) \leq l(z_i) - 1$ and $\tau \geq 11$

(see Definition 2). Since we also have $l(x) \leq l(w) < l(z_{m-1}) \leq l(z) \leq l(\bar{p}(x)) = l(y)$, we conclude that $x \in \overline{\text{Rel}}(z_i)$ for $i = 2, \dots, m$ (note that we do not necessarily have $x \in \overline{\text{Rel}}(z_1)$, but this is not important). Now, it remains to show that $z \in \text{Rel}(y)$. We compute:

$$\begin{aligned} d_P(\text{rep}_y, \text{rep}_z) &\leq d_P(\text{rep}_y, \text{rep}_x) + d_P(\text{rep}_x, \text{rep}_w) + d_P(\text{rep}_w, \text{rep}_z) \\ &\leq (1 + n^{-\alpha})^2 \tau^{l(y)} + 3(1 + n^{-\alpha})^2 \tau^{l(w)} + \frac{(1 + n^{-\alpha})^2 \tau}{\tau - 1} \tau^{l(z)} \\ &\leq (1 + n^{-\alpha})^2 \tau^{l(y)} + 3(1 + n^{-\alpha})^2 \tau^{l(y)-1} + \frac{(1 + n^{-\alpha})^2 \tau}{\tau - 1} \tau^{l(y)} \\ &\leq 3(1 + n^{-\alpha})^2 \tau^{l(y)} \end{aligned}$$

Since $l(z) \leq l(y) < l(\bar{p}(z))$ by definition, we see that $z \in \overline{\text{Rel}}(y)$. By the inductive hypothesis, we conclude that $z \in \text{Rel}(y)$ as desired.

Now, consider the case where $w \in \overline{\text{Rel}}(x)$. Then, we have $l(w) \leq l(x) < l(\bar{p}(w))$, and

$$d_P(\text{rep}_w, \text{rep}_y) \leq d_P(\text{rep}_w, \text{rep}_x) + d_P(\text{rep}_x, \text{rep}_y) \leq 3(1 + n^{-\alpha})^2 \tau^{l(y)}$$

We consider three possibilities:

Case 1: $l(y) < l(\bar{p}(w))$. Then, we have $w \in \overline{\text{Rel}}(y)$, which, by the inductive hypothesis, implies that $w \in \text{Rel}(y)$. Thus, the vertex w will be scanned by the algorithm.

Case 2: $l(y) = l(\bar{p}(w))$. We compute:

$$\begin{aligned} d_P(\text{rep}_{\bar{p}(w)}, \text{rep}_y) &\leq d_P(\text{rep}_{\bar{p}(w)}, \text{rep}_w) + d_P(\text{rep}_w, \text{rep}_x) + d_P(\text{rep}_x, \text{rep}_y) \\ &\leq \frac{(1 + n^{-\alpha})^2 \tau}{\tau - 1} \tau^{l(\bar{p}(w))} + 3(1 + n^{-\alpha})^2 \tau^{l(x)} + \frac{(1 + n^{-\alpha})^2 \tau}{\tau - 1} \tau^{l(y)} \\ &\leq 3(1 + n^{-\alpha})^2 \tau^{l(y)} \end{aligned}$$

This implies that $\bar{p}(w) \in \overline{\text{Rel}}(y)$, which, by the inductive hypothesis, yields $\bar{p}(w) \in \text{Rel}(y)$. Hence, the vertex w will be scanned by the algorithm.

Case 3: $l(y) > l(\bar{p}(w))$. If $l(\bar{p}(\bar{p}(w))) > l(y)$, then Case 2 above would yield the desired conclusion. However, if this is not the case, then we consider:

$$d_P(\text{rep}_{\bar{p}(w)}, \text{rep}_x) \leq d_P(\text{rep}_{\bar{p}(w)}, \text{rep}_w) + d_P(\text{rep}_w, \text{rep}_x) \leq 3(1 + n^{-\alpha})^2 \tau^{l(\bar{p}(w))}$$

Moreover, note that $l(x) < l(\bar{p}(w)) < l(\bar{p}(x)) = l(y)$. Hence, we see that $x \in \overline{\text{Rel}}(\bar{p}(w))$. By our result above, the vertex $\bar{p}(w)$ will be scanned by the algorithm, and as a result x will be added to $\text{Rel}(\bar{p}(w))$. These imply that the algorithm will also scanned the children of $\bar{p}(w)$. In particular, the vertex w will be scanned.

This completes the proof of Theorem 6. □

In order to establish the running time of the net-tree construction algorithm, we need the following proposition.

Proposition 12 *For each $k = 1, \dots, n$ and for each $u \in T^{(k)}$, the size of the set $\overline{\text{Rel}}(u)$ is $\lambda^{O(1)}$.*

Proof By Theorem 6(c), every pair of points in $\overline{\text{Rel}}(u)$ is at a distance of at least $\tau^{l(u)-1}$ apart. Hence, for each $v \in \overline{\text{Rel}}(u)$, we have $B(v, \tau^{l(u)-1}/2) \cap \overline{\text{Rel}}(u) = \{v\}$. Thus, by the doubling property, we have $|\overline{\text{Rel}}(u)| \leq \lambda^{\log_2(6(1+n^{-\alpha})^2)} = \lambda^{O(1)}$, as desired. \square

We are now ready to finish the proof of Theorem 2.

Proposition 13 *Given a greedy permutation $\Pi = \{p_1, \dots, p_n\}$ generated by HSTCLUSTER, the algorithm for constructing the net-tree runs in $O(\lambda^{O(1)}n)$ time.*

Proof The tree construction takes $\lambda^{O(1)}$ time per point. To estimate the time required to construct the $\text{Rel}(\cdot)$ sets, consider a newly added vertex x . By Proposition 12, we can charge $\lambda^{O(1)}$ amount of work to $\bar{p}(x)$ for visiting the children of $\text{Rel}(\bar{p}(x))$. Next, we charge all other visits to the parent of the visited vertex. Now, note that each vertex v in the tree has $\lambda^{O(1)}$ children, and the children of v are visited only if a new entry is inserted into $\text{Rel}(v)$. Since the total size of the $\text{Rel}(\cdot)$ lists is at most $\lambda^{O(1)}n$, we conclude that the number of visited vertices during the update process of $\text{Rel}(\cdot)$ is at most $\lambda^{O(1)}n$. It follows that the total time required is at most $\lambda^{O(1)}n$. \square

References

- [1] Teofilo F. Gonzalez, *Clustering to Minimize the Maximum Intercluster Distance*, Theoretical Computer Science 38:293–306, 1985.
- [2] Sariel Har-Peled, Manor Mendel, *Fast Construction of Nets in Low-Dimensional Metrics and Their Applications*, SIAM Journal on Computing 35(5):1148–1184, 2006.