Chapter 12

Random Partition via Shifting

By Sariel Har-Peled, April 14, 2025[1]

Associations of mutual interest between the university and the corporations were natural, inevitable, and widely accepted. According to the state legislature, they were to be actively pursued. The legislature, in fact, was already counting the "resources" that could be "allocated" elsewhere in state government when corporations began picking up more of the tab for higher education, so success in finding this money would certainly convince them that further experiments in driving the university into the arms of the private sector would be warranted, that actually paying for the university out of state funds was irresponsible, or even immoral, or even criminal (robbing widows and children, etc., to fatten sleek professors who couldn't find real employment, etc.).

Moo, Jane Smiley

In this chapter, we investigate a rather simple technique for partitioning a geometric domain. We randomly shift a grid in $\mathbb{R}^{\mathsf{d}}$ and consider each grid cell resulting from this shift. The shifting is done by adding a random vector, chosen uniformly from $[0,1]^{\mathsf{d}}$, so that the new grid has this vector as its origin.

Points that are close together have a good probability of falling into the same cell in the new grid. This idea can be extended to shifting multi-resolution grids over $\mathbb{R}^{\mathsf{d}}$. That is, we randomly shift a quadtree over a region of interest. This yields some simple algorithms for clustering and nearest neighbor search.

12.1. Partition via shifting

12.1.1. Shifted partition of the real line

Consider a real number $\Delta > 0$, and let b be a uniformly distributed number in the interval $[0, \Delta]$. This induces a natural partition of the real line into intervals, by the function

$$h_{\mathsf{b},\Delta}(x) = \left\lfloor \frac{x - \mathsf{b}}{\Delta} \right\rfloor .$$

Hence each interval has size Δ, and the origin is shifted to the right by an amount b.

Remark 12.1.1. Note that $h_{\mathsf{b},\Delta}(x)$ induces the same partition of the real line as $h_{\mathsf{b}',\Delta}(x)$ (but it is not the same function) if $|\mathsf{b} - \mathsf{b}'| = i\Delta$, where i is an integer. In particular, this implies that we can pick b uniformly from any interval of length Δ and get the same distribution on the partitions of the real line.

Specifically, for our purposes, it is enough if b is distributed uniformly in an interval of the form $[y + i\Delta, y + j\Delta]$, for two integers i and j such that $i < j$ and y is some real number. It is easy to verify that we still get the same distribution on the partitions of the real line.

Lemma 12.1.2. *For any $x, y \in \mathbb{R}$, we have* $\mathbb{P}\Big[h_{\mathsf{b},\Delta}(x) \neq h_{\mathsf{b},\Delta}(y)\Big] = \min\Big(\frac{|x-y|}{\Delta}, 1\Big)$.

Proof: Assume $x < y$. Clearly, the claim trivially holds if $|y - x| > \Delta$, since then x and y are always assigned different values of $h_{\mathsf{b},\Delta}(\cdot)$ and the required probability is 1. Now, imagine we pick b uniformly in the range $[x, x + \Delta]$. Clearly, the probability of the event we are interested in remains the same. But then, $h_{\mathsf{b},\Delta}(x) \neq h_{\mathsf{b},\Delta}(y)$ if and only if $\mathsf{b} \in [x, y]$, which implies the claim.

[1]This work is licensed under the Creative Commons Attribution-Noncommercial 3.0 License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc/3.0/ or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

12.1.2. Shifted partition of space

Let P be a point set in $\mathbb{R}^{\mathsf{d}}$, and consider a point $\mathsf{b} = (\mathsf{b}_1, \ldots, \mathsf{b}_{\mathsf{d}}) \in \mathbb{R}^{\mathsf{d}}$ randomly and uniformly chosen from the hypercube $[0, \Delta]^{\mathsf{d}}$, and consider the grid $\mathsf{G}^{\mathsf{d}}(\mathsf{b}, \Delta)$ that has its origin at b and with sidelength Δ. For a point $p \in \mathbb{R}^{\mathsf{d}}$ the ID of the grid cell containing it is

$$\mathrm{id}(p) = h_{\mathsf{b},\Delta}(p) = \left(h_{\mathsf{b}_1,\Delta}(p_1), \ldots, h_{\mathsf{b}_{\mathsf{d}},\Delta}(p_{\mathsf{d}})\right).$$

(We used a similar concept when solving the closest pair problem; see Theorem 12.6.5$_{\text{p179}}$.)

Lemma 12.1.3. *Given a ball B of radius r in $\mathbb{R}^{\mathsf{d}}$ (or an axis parallel hypercube with sidelength $2r$). The probability that B is not contained in a single cell of $\mathsf{G}^{\mathsf{d}}(\mathsf{b}, \Delta)$ is bounded by* $\min(\, 2\mathsf{d}r/\Delta\,,\, 1)$*, where $\mathsf{G}^{\mathsf{d}}(\mathsf{b}, \Delta)$ is a randomly shifted grid.*

Proof: Project B into the ith coordinate. It becomes an interval B_i of length $2r$, and $\mathsf{G}^{\mathsf{d}}(\mathsf{b}, \Delta)$ becomes the one-dimensional shifted grid $\mathsf{G}^1(b_i, \Delta)$. Clearly, B is contained in a single cell of $\mathsf{G}^{\mathsf{d}}(\mathsf{b}, \Delta)$ if and only if B_i is contained in a single cell of $\mathsf{G}^1(\mathsf{b}_i, \Delta)$, for $i = 1, \ldots, \mathsf{d}$.

Now, B_i is not contained in an interval of $\mathsf{G}^1(\mathsf{b}_i, \Delta)$ if and only if its endpoints are in different cells. Let $\mathcal{E}_i$ denote this event. By Lemma 12.1.2, the probability of $\mathcal{E}_i$ is $\leq 2r/\Delta$. As such, the probability of B not being contained in a single grid is $\mathbb{P}[\bigcup_i \mathcal{E}_i] \leq \sum_i \mathbb{P}[\mathcal{E}_i] \leq 2\mathsf{d}r/\Delta$. Since a probability is always bounded by 1, we have that this probability is bounded by $\min(2\mathsf{d}r/\Delta, 1)$.

12.1.2.1. Improved bound

If we care only if two specific points get separated, then a better bound is possible.

Lemma 12.1.4. *Consider two points p, q in $\mathbb{R}^{\mathsf{d}}$. The probability that the segment pq is not contained in a single cell of $\mathsf{G}^{\mathsf{d}}(\mathsf{b}, \Delta)$ is bounded by* $\min\left(\sqrt{\mathsf{d}}\,\|p - q\| / \Delta\,,\, 1\right)$*, where $\mathsf{G}^{\mathsf{d}}(\mathsf{b}, \Delta)$ is a randomly shifted grid.*

Proof: Let R be the axis parallel bounding box of p and q. The ith coordinate of R is an interval R_i of length r_i, and $\mathsf{G}^{\mathsf{d}}(\mathsf{b}, \Delta)$ becomes the one-dimensional shifted grid $\mathsf{G}^1(\mathsf{b}_i, \Delta)$. Clearly, R is contained in a single cell of $\mathsf{G}^{\mathsf{d}}(\mathsf{b}, \Delta)$ if and only if R_i is contained in a single cell of $\mathsf{G}^1(b_i, \Delta)$, for $i = 1, \ldots, \mathsf{d}$.

Now, R_i is not contained in an interval of $\mathsf{G}^1(\mathsf{b}_i, \Delta)$ if and only if its endpoints are in different cells. Let $\mathcal{E}_i$ denote this event. Arguing as in Lemma 12.1.2, the probability of $\mathcal{E}_i$ is $\leq \mathsf{R}_i/\Delta$. As such, the probability of R not being contained in a single grid is

$$\mathbb{P}\left[\bigcup_i \mathcal{E}_i\right] \leq \sum_i \mathbb{P}[\mathcal{E}_i] = \sum_{i=1}^{\mathsf{d}} \frac{\mathsf{R}_i}{\Delta} \leq \sqrt{\sum_{i=1}^{\mathsf{d}} \mathsf{R}_i^2}\sqrt{\sum_{i=1}^{\mathsf{d}} \frac{1}{\Delta^2}} = \sqrt{\mathsf{d}}\,\frac{\|p - q\|}{\Delta},$$

by the Cauchy-Schwarz inequality.

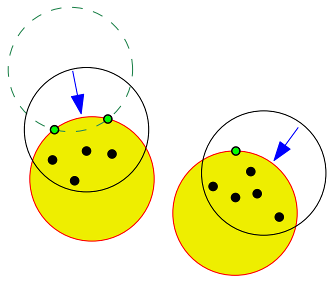
In particular, for two points in distance r from each other, we need a grid of length $\Delta = 2\sqrt{\mathsf{d}}r$ to have a probability of half to fall into the same cell. Observe, that a cell in this grid has diameter $\sqrt{\mathsf{d}}\Delta = 2\mathsf{d}r$; that is, the diameter of the cell is by a factor of $2\mathsf{d}$ bigger than the distance between the points. A better bound is known by using a different random partition, see bibliographical notes for details.

12.1.2.2. Application – covering by disks

Given a set P of n points in the plane, we would like to cover them by a minimal number of unit disks. Here, a ***unit disk*** is a disk of radius 1. Observe that if the cover requires k disks, then we can compute it in $O\left(kn^{2k+1}\right)$ time.

Indeed, consider a unit disk d that covers a subset $Q \subseteq P$. We can translate d such that it still covers Q, and either its boundary circle contains two points of Q or the top point of this circle is on an input point.

Observe the following: (I) Every pair of such input points defines two possible disks; see the figure on the right. (II) The same subset covered by a single disk might be coverable by several different such ***canonical*** disks. (III) If a pair of input points is at distance larger than 2, then the canonical disk they define is invalid.



As such, there are

$$2\binom{n}{2} + n \leq n^2$$

such canonical disks. We can assume the cover uses only such disks.

Thus, we exhaustively check all such possible covers formed by k canonical disks. Overall, there are $\leq n^{2k}$ different covers to consider, and each such candidate cover can be verified in $O(nk)$ time. We thus get the following easy result.

Lemma 12.1.5. *Given a set P of n points in the plane, one can compute, in $O\left(kn^{2k+1}\right)$ time, a cover of P by at most k unit disks, if such a cover exists.*

Proof: We use the above algorithm trying all covers of size i, for $i = 1, \ldots, k$. The algorithm returns the first cover found. Clearly, the running time is dominated by the last iteration of this algorithm.

One can improve the running time of Lemma 12.1.5 to $n^{O\left(\sqrt{k}\right)}$; see Exercise 12.5.2.

The problem with this algorithm is that k might be quite large (say $n/4$). Fortunately, the shifting grid saves the day.

Theorem 12.1.6. *Given a set P of n points in the plane and a parameter $\varepsilon > 0$, one can compute using a randomized algorithm, in $n^{O(1/\varepsilon^2)}$ time, a cover of P by X unit disks, where $\mathbb{E}[X] \leq (1+\varepsilon)\mathrm{opt}$, where opt is the minimum number of unit disks required to cover P.*

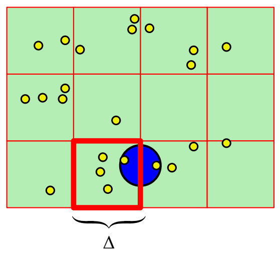
Proof: Let $\Delta = 12/\varepsilon$, and consider a randomly shifted grid $\mathsf{G}^2(\mathsf{b}, \Delta)$. Compute all the grid cells that contain points of P. This is done by computing for each point $p \in P$ its $\mathrm{id}(p)$ and storing it in a hash table. This clearly can be done in linear time.

Now, for each such grid cell $\square$ we now have the points of P falling into this grid cell (they are stored with the same id in the hash table), and let $P_\square$ denote this set of points.

Observe that any grid cell of $\mathsf{G}^2(\mathsf{b}, \Delta)$ can be covered by $M = (\Delta+1)^2$ unit disks. Indeed, each unit disk contains a unit square, and clearly a grid cell of sidelength Δ can be covered using at most M unit squares.

Thus, for each such grid cell $\square$, compute the minimum number of unit disks required to cover $P_\square$. Since this number is at most M, we can compute this minimum cover in $O\left(Mn^{2M+1}\right)$ time, by Lemma 12.1.5. There are at most n non-empty grid cells, so the total running time of this stage is $O\left(Mn^{2M+2}\right) = n^{O(1/\varepsilon^2)}$. Finally, merge together the covers from each grid cell, and return this as the overall cover.

We are left with the task of bounding the expectation of X. So, consider the optimal solution $\mathcal{F} = \{\mathsf{d}_1, \ldots, \mathsf{d}_{\mathrm{opt}}\}$. We generate a feasible solution $\mathcal{G}$ from $\mathcal{F}$. Furthermore, the solution $\mathcal{G}$ is one of the possible solutions considered by the algorithm. Specifically, for each grid cell $\square$, let $\mathcal{F}_\square$ denote the set of disks of the optimal solution that intersect $\square$. Consider the multi-set $\mathcal{G} = \bigcup_\square \mathcal{F}_\square$. Clearly, the algorithm returns for each grid cell $\square$ a cover that is of size at most $|\mathcal{F}_\square|$ (indeed, it returns the smallest possible cover for $P_\square$, and $\mathcal{F}_\square$ is one such possible cover). As such, the cover returned by the algorithm is of size at most $|\mathcal{G}|$.



Clearly, a disk of the optimal solution can intersect at most four cells of the grid, and as such it can appear in $\mathcal{G}$ at most four times. In fact, a disk $\mathsf{d}_i \in \mathcal{F}$ will appear in $\mathcal{G}$ more than once if and only if it is not fully contained in a grid cell of $\mathsf{G}^2(\mathsf{b}, \Delta)$. By Lemma 12.1.3 the probability for that is bounded by $4/\Delta$ (as $r = 1$ and $\mathsf{d} = 2$).

Specifically, let X_i be an indicator variable that is 1 if and only if d_i is not fully contained in a single cell of $\mathsf{G}^2(\mathsf{b}, \Delta)$. We have that

$$\mathbb{E}\Big[|\mathcal{G}|\Big] \leq \mathbb{E}\left[\mathrm{opt} + \sum_{i=1}^{\mathrm{opt}} 3X_i\right] = \mathrm{opt} + \sum_{i=1}^{\mathrm{opt}} 3\,\mathbb{E}\Big[X_i\Big] = \mathrm{opt} + \sum_{i=1}^{\mathrm{opt}} 3\,\mathbb{P}\Big[X_i = 1\Big] \leq \mathrm{opt} + \sum_{i=1}^{\mathrm{opt}} 3\frac{4}{\Delta}$$
$$= \left(1 + \frac{12}{\Delta}\right)\mathrm{opt} = (1+\varepsilon)\mathrm{opt},$$

since $\Delta = 12/\varepsilon$. As such, in expectation, the solution returned by the algorithm is of size at most $(1+\varepsilon)\mathrm{opt}$.

The running time of Theorem 12.1.6 can be improved to $n^{O(1/\varepsilon)}$; see Exercise 12.5.2.

12.1.3. Shifting quadtrees

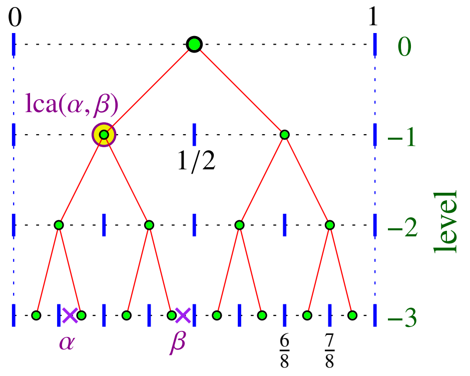
12.1.3.1. One-dimensional quadtrees

Assume that we are given a set P of n numbers contained in the interval $[1/2, 3/4]$. Randomly, and uniformly, choose a number $\mathsf{b} \in [0, 1/2]$, and consider the (one-dimensional) quadtree $\mathcal{T}$ of P, using the interval $\mathsf{b} + [0, 1]$ for the root cell. Now, for two numbers $\alpha, \beta \in P$ let

$$\mathbb{L}_{\mathsf{b}}(\alpha, \beta) = 1 - \mathrm{bit}_{\Delta}(\alpha - \mathsf{b}, \beta - \mathsf{b});$$

see Definition 12.6.3$_{\mathrm{p179}}$.

This is the last ***level*** of the (one-dimensional) shifted canonical grid (i.e., one-dimensional quadtree) that contains α and β in the same interval. Namely, this is the level of the node of $\mathcal{T}$ that is the least common ancestor containing both numbers; that is, the level of $\mathrm{lca}(\alpha, \beta)$ is $\mathbb{L}_{\mathsf{b}}(\alpha, \beta)$. We remind the reader that a node in this quadtree that corresponds to an interval of length 2^{-i} has level $-i$. These definitions are demonstrated in the figure on the right (without shifting).



In the following, we will assume that one can compute $\mathbb{L}_{\mathsf{b}}(\alpha, \beta)$ in constant time.

Remark 12.1.7. Interestingly, the value of $\mathbb{L}_{\mathsf{b}}(\alpha, \beta)$ depends only on α, β, and b but is independent of the other points of P.

The following lemma bounds the probability that the least common ancestor of two numbers is in charge of an interval that is considerably longer than the difference between the two numbers.

Lemma 12.1.8. *Let $\alpha, \beta \in [1/2, 3/4]$ be two numbers, and consider a random number $\mathsf{b} \in [0, 1/2]$. Then, for any positive integer t, we have that $\mathbb{P}[\mathbb{L}_{\mathsf{b}}(\alpha, \beta) > \lg |\alpha - \beta| + t] \le 4/2^t$[2], where $\lg x = \log_2 x$.*

Proof: Let $M = \lfloor \lg |\alpha - \beta| \rfloor$ and consider the shifted partition of the real line by intervals of length $\Delta_{M+i} = 2^{M+i}$ and a shift b. Let X_{M+i} be an indicator variable that is 1 if and only if α and β are in different intervals of this shifted partition. Formally, $X_{M+i} = 1$ if and only if $h_{\mathsf{b},\Delta_{M+i}}(\alpha) \neq h_{\mathsf{b},\Delta_{M+i}}(\beta)$.

Now, if $\mathbb{L}_{\mathsf{b}}(\alpha, \beta) = M + i$, then the highest level such that both α and β lie in the same shifted interval is $M + i$. As such, going one level down, these two numbers are in different intervals, and $h_{\mathsf{b},2^{M+i-1}}(\alpha) \neq h_{\mathsf{b},2^{M+i-1}}(\beta)$. Namely, we have that $X_{M+i-1} = 1$. By Lemma 12.1.2, we have that $\mathbb{P}[X_{M+i} = 1] \le |\alpha - \beta| / \Delta_{M+i}$. As such, the probability we are interested in is

$$\mathbb{P}[\mathbb{L}_{\mathsf{b}}(\alpha, \beta) > \lg |\alpha - \beta| + t] \le \sum_{i=1+t}^{\infty} \mathbb{P}[\mathbb{L}_{\mathsf{b}}(\alpha, \beta) = M + i] \le \sum_{i=t}^{\infty} \mathbb{P}[X_{M+i} = 1]$$
$$\le \sum_{i=t}^{\infty} \frac{|\alpha - \beta|}{\Delta_{M+i}} = \sum_{i=t}^{\infty} \frac{|\alpha - \beta|}{2^M \cdot 2^i} \le \sum_{i=t}^{\infty} \frac{|\alpha - \beta|}{(|\alpha - \beta|/2)2^i} \le \sum_{i=t}^{\infty} 2^{1-i} = 2^{2-t}.$$

Corollary 12.1.9. *Let $\alpha, \beta \in P \subseteq [1/2, 3/4]$ be two numbers, and consider a random number $\mathsf{b} \in [0, 1/2]$. Then, for any parameters $c > 1$, we have $\mathbb{P}[\mathbb{L}_{\mathsf{b}}(\alpha, \beta) > \lg |\alpha - \beta| + c \lg n] \le 4/n^c$, where $n = |P|$.*

12.1.3.2. Higher-dimensional quadtrees

Let P be a set of n points in $[1/2, 3/4]^{\mathsf{d}}$, pick uniformly and randomly a point $\mathsf{b} \in [0, 1/2]^{\mathsf{d}}$, and consider the shifted compressed quadtree $\mathcal{T}$ of P having $\mathsf{b} + [0, 1]^{\mathsf{d}}$ as the root cell.

Consider any two points $p, q \in P$, their $\mathrm{lca}(p, q)$, and the one-dimensional quadtrees $\mathcal{T}_1, \ldots, \mathcal{T}_{\mathsf{d}}$ built on each of the d coordinates of the point set. Since the quadtree $\mathcal{T}$ is the combination of these one-dimensional quadtrees $\mathcal{T}_1, \ldots, \mathcal{T}_{\mathsf{d}}$, the level where p and q get separated in $\mathcal{T}$ is the first level in any of the quadtrees $\mathcal{T}_1, \ldots, \mathcal{T}_{\mathsf{d}}$ where p and q are separated. In particular, the ***level of the least common ancestor*** (i.e., the level of $\mathrm{lca}(p, q)$) is

$$\mathbb{L}_{\mathsf{b}}(p, q) = \max_{i=1}^{\mathsf{d}} \mathbb{L}_{\mathsf{b}_i}(p_i, q_i). \tag{12.1}$$

As in the one-dimensional case (see Remark 12.1.7) the value of $\mathbb{L}_{\mathsf{b}}(p, q)$ is independent of the other points of P.

Intuitively, $\mathbb{L}_{\mathsf{b}}(p, q)$ is a well-behaved random variable, as testified by the following lemma. Since we do not use this lemma anywhere directly, we leave its proof as an exercise to the reader (see Exercise 12.5.1).

Lemma 12.1.10. *For any two fixed points $p, q \in P$, the following properties hold.*
(A) For any integer $t > 0$, we have that $\mathbb{P}\Big[\mathbb{L}_{\mathsf{b}}(p, q) > \lg \|p - q\| + t\Big] \le 4\mathsf{d}/2^t$.
(B) $\mathbb{E}[\mathbb{L}_{\mathsf{b}}(p, q)] \le \lg \|p - q\| + \lg \mathsf{d} + 6$.
(C) $\mathbb{L}_{\mathsf{b}}(p, q) \ge \lg \|p - q\| - \lg \mathsf{d} - 3$.

[2]We remind the reader that $\lg x = \log_2 x$.

12.2. Hierarchical representation of a point set

In the following, it will be convenient to carry out our discussion in a more general setting than in low-dimensional Euclidean space. In particular, we will use the notion of metric space; see Definition 12.6.2$_{\text{p179}}$. Specifically, we are given a metric space $\mathcal{M}$ with a metric d. We will slightly abuse notation by using $\mathcal{M}$ to refer to the underlying set of points.

12.2.1. Low quality approximation by HST

We will use the following special type of a metric space.

Definition 12.2.1. Let P be a set of elements, and let $\mathtt{H}$ be a tree having the elements of P as leaves. The tree $\mathtt{H}$ defines a ***hierarchically well-separated tree*** (HST) over the points of P if for each vertex $u \in \mathtt{H}$ there is associated a label $\Delta(u) \geq 0$, such that $\Delta(u) = 0$ if and only if u is a leaf of $\mathtt{H}$. Furthermore, the labels are such that if a vertex u is a child of a vertex v, then $\Delta(u) \leq \Delta(v)$. The distance between two leaves $x, y \in \mathtt{H}$ is defined as $\Delta(\mathrm{lca}(x, y))$, where $\mathrm{lca}(x, y)$ is the least common ancestor of x and y in $\mathtt{H}$.

If every internal node of $\mathtt{H}$ has exactly two children, we will refer to it as being a ***binary HST*** (BHST).

It is easy to verify that the distances defined by an HST comply with the triangle inequality, and as such the HST defines a metric. The usefulness of an HST is that the metric it defines has a very simple structure, and it can be easily manipulated algorithmically.

Example 12.2.2. Consider a point set $P \subseteq \mathbb{R}^{\mathsf{d}}$ and a compressed quadtree $\mathcal{T}$ storing P, where for each node $v \in \mathcal{T}$, we set the diameter of $\square_v$ to be its label. It is easy to verify that this is an HST.

For convenience, from now on, we will work with BHSTs, since any HST can be converted into a binary HST in linear time while retaining the underlying distances. We will also associate with every vertex $u \in \mathtt{H}$ an arbitrary representative point $\mathrm{rep}_u \in P_u$ (i.e., a point stored in the subtree rooted at u). We also require that $\mathrm{rep}_u \in \left\{ \mathrm{rep}_v \,\middle|\, v \text{ is a child of } u \right\}$.

Definition 12.2.3. A metric space $\mathcal{N}$ is said to ***t-approximate*** the metric $\mathcal{M}$ if they are defined over the same set of points P and $\mathrm{d}_{\mathcal{M}}(u, v) \leq \mathrm{d}_{\mathcal{N}}(u, v) \leq t \cdot \mathrm{d}_{\mathcal{M}}(u, v)$, for any $u, v \in P$.

It is not hard to see that any n-point metric is $(n-1)$-approximated by some HST.

Lemma 12.2.4. *Given a weighted connected graph G on n vertices and m edges, it is possible to construct, in $O(n \log n + m)$ time, a binary HST $\mathtt{H}$ that $(n-1)$-approximates the shortest path metric of G.*

Proof: Compute the minimum spanning tree $\mathcal{T}$ of G in $O(n \log n + m)$ time[③].

Sort the $n-1$ edges of $\mathcal{T}$ in non-decreasing order, and add them to the graph one by one, starting with an empty graph on $V(G)$. The HST is built bottom up. At each stage, we have a collection of HSTs, each corresponding to a connected component of the current graph. Each added edge merges two connected components, and we merge the two corresponding HSTs into a single HST by adding a new common root v for the two HSTs and labeling this root with the edge's weight times $|P_v| - 1$, where P_v

[③]Using, say, Prim's algorithm implemented using a Fibonacci heap.

is the set of points stored in this subtree. This algorithm is only a slight variation on Kruskal algorithm and hence has the same running time.

Let $\mathtt{H}$ denote the resulting HST. As for the approximation factor, let x, y be any two vertices of G, and let e be the first edge added such that x and y are in the same connected component C, created by merging two connected components C_x and C_y. Observe that e must be the lightest edge in the cut between C_x and the rest of the vertices of G. As such, any path between x and y in G must contain an edge of weight at least $\omega(\mathsf{e})$. Now, e is the heaviest edge in C and

$$\mathrm{d}_G(x,y) \le (|C|-1)\,\omega(\mathsf{e}) \le (n-1)\omega(\mathsf{e}) \le (n-1)\mathrm{d}_G(x,y),$$

since $\omega(\mathsf{e}) \le \mathrm{d}_G(x,y)$. Now, $\mathrm{d}_{\mathtt{H}}(x,y) = (|C|-1)\,\omega(\mathsf{e})$, and the claim follows.

Since any n-point metric $P \subseteq \mathcal{M}$ can be represented using the complete graph over n vertices (with edge weights $\omega(xy) = \mathrm{d}_{\mathcal{M}}(x,y)$ for all $x, y \in P$), we get the following.

Corollary 12.2.5. *For a set P of n points in a metric space $\mathcal{M}$, one can compute, in $O(n^2)$ time, an HST $\mathtt{H}$ that $(n-1)$-approximates the metric $\mathrm{d}_{\mathcal{M}}$.*

One can improve the running time in low-dimensional Euclidean space (the approximation factor deteriorates slightly).

Corollary 12.2.6. *For a set P of n points in $\mathbb{R}^{\mathsf{d}}$, one can construct, in $O(n \log n)$ time (the constant in the $O(\cdot)$ depends exponentially on the dimension), a BHST $\mathtt{H}$ that $(2n-2)$-approximates the distances of points in P. That is, for any $p, q \in P$, we have $\mathrm{d}_{\mathtt{H}}(p,q)/(2n-2) \le \|p-q\| \le \mathrm{d}_{\mathtt{H}}(p,q)$.*

Proof: We remind the reader, that in $\mathbb{R}^{\mathsf{d}}$ one can compute a 2-spanner for P of size $O(n)$, in $O(n \log n)$ time (see Theorem 12.6.4$_{\text{p179}}$). Let G be this spanner, and we apply Lemma 12.2.4 to this spanner. Let $\mathtt{H}$ be the resulting HST metric. For any $p, q \in P$, we have $\|p-q\| \le \mathrm{d}_{\mathtt{H}}(p,q) \le (n-1)\mathrm{d}_G(p,q) \le 2(n-1)\|p-q\|$.

Corollary 12.2.6 is unique to $\mathbb{R}^{\mathsf{d}}$ since for general metric spaces no HST can be computed in subquadratic time; see Exercise 12.5.3.

12.2.2. Fast and dirty HST in high dimensions

The above construction of an HST has exponential dependency on the dimension. We next show how one can get an approximate HST of low quality but in polynomial time in the dimension.

Lemma 12.2.7. *Let P be a set of n points in $[1/2, 3/4]^{\mathsf{d}}$, pick uniformly and randomly a point $\mathsf{b} \in [0, 1/2]^{\mathsf{d}}$, and consider the shifted compressed quadtree $\mathcal{T}$ of P having $\mathsf{b} + [0,1]^{\mathsf{d}}$ as the root cell. Then, for any constant $c > 1$, with probability $\ge 1 - 4\mathsf{d}/n^{c-2}$, for all $i = 1, \ldots, \mathsf{d}$ and for all pairs of points of P, we have*

$$\mathbb{L}_{\mathsf{b}_i}(p_i, q_i) \le \lg |p_i - q_i| + c \lg n. \tag{12.2}$$

In particular, this property implies that the compressed quadtree $\mathcal{T}$ is a $2\sqrt{\mathsf{d}}n^c$-approximate HST for P.

Proof: Consider two points $p, q \in P$ and a coordinate i. By Corollary 12.1.9, we have that

$$\mathbb{P}[\mathbb{L}_{\mathsf{b}_i}(p_i, q_i) > \lg |p_i - q_i| + c \lg n] \le 4/n^c.$$

There are d possible coordinates and $\binom{n}{2}$ possible pairs, and as such, by the union bound, this does not happen for any pair of points of P, for any coordinate, with probability $\geq 1 - \mathsf{d}\binom{n}{2}\frac{4}{n^c} \geq 1 - 2\mathsf{d}/n^{c-2}$.

As such, with probability $\geq 1 - 2\mathsf{d}/n^c$, the level of the lca of any two points $p, q \in P$ is at most

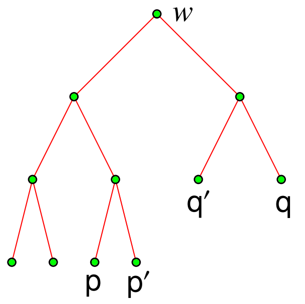
$$U = \mathbb{L}_{\mathsf{b}}(p,q) = \max_{i=1}^{\mathsf{d}} \mathbb{L}_{\mathsf{b}_i}(p_i,q_i) \leq \max(\lg|p_i - q_i| + c\lg n) \leq \left\lceil \lg\|p-q\| + c\lg n \right\rceil,$$

by Eq. (12.1). The diameter of a cell at level $U(p,q)$ is at most $\sqrt{\mathsf{d}}2^U \leq 2\sqrt{\mathsf{d}}\,\|p-q\|\,n^c$. Namely, $\mathcal{T}$ is a $2\sqrt{\mathsf{d}}n^c$-approximate HST.

Verifying *quickly* that $\mathcal{T}$ is an acceptable HST (as far as the quality of approximation goes) is quite challenging in general. Fortunately, one can easily check that Eq. (12.2) holds.

Claim 12.2.8. *Eq. (12.2) holds for the quadtree $\mathcal{T}$ computed by the algorithm of Lemma 12.2.7 in $O(\mathsf{d}n\log n)$ time.*

Proof: If Eq. (12.2) fails, then it must be that there are two points $p, q \in P$ and a coordinate j such that $\mathbb{L}_{\mathsf{b}_j}(p_j,q_j) > \lg|p_j - q_j| + t$, for $t = c\lg n$.



Now, observe that if there are two such bad points for the jth coordinate, then there are two bad points that are consecutive in the order along the jth coordinate. Indeed, consider $w = \mathrm{lca}(p_j,q_j)$ in the one-dimensional compressed quadtree $\mathcal{T}_j$ of P on the jth coordinate. Let p' (resp. q') be the point with maximal (resp. minimal) value in the jth coordinate that is still in the left (resp. right) subtree of w; see the figure on the right. Clearly, (i) $\mathrm{lca}(p',q') = w$, (ii) $\left|p'_j - q'_j\right| \leq |p_j - q_j|$, and (iii) p' and q' are consecutive in the ordering of P according to the values in the jth coordinate. As such, $\mathbb{L}_{\mathsf{b}_j}\left(p'_j, q'_j\right) = \mathbb{L}_{\mathsf{b}_j}(p_j,q_j) > \lg|p_j - q_j| + t \geq \lg\left|p'_j - q'_j\right| + t$. Namely, Eq. (12.2) fails for p' and q' on the jth coordinate.

As such, to verify Eq. (12.2) on the jth coordinate, we sort the points according to their order on the jth coordinate and verify Eq. (12.2) for each consecutive pair. This takes $O(n\log n)$ time per coordinate, since computing $\mathbb{L}_{\mathsf{b}_j}(\cdot,\cdot)$ takes constant time. Doing this for all d coordinates takes $O(\mathsf{d}n\log n)$ time overall.

Theorem 12.2.9. *Given a set P of n points in $\mathbb{R}^{\mathsf{d}}$, for $\mathsf{d} \leq n$, one can compute a $2\sqrt{\mathsf{d}}n^5$-approximate HST of P in $O(\mathsf{d}n\log n)$ expected time.*

Proof: Set $c = 5$, use the algorithm of Lemma 12.2.7, and verify it, as described in Claim 12.2.8. If the compressed quadtree fails, we repeat the construction until succeeding. Computing and verifying the compressed quadtree takes $O(\mathsf{d}n\log n)$ time, by Theorem 12.6.1$_{\mathrm{p179}}$ and Claim 12.2.8. Now, since the probability of success is $\geq 1 - 4\mathsf{d}/n^{c-2} \geq 1 - 1/n$ (assuming $n \geq 3$), it follows that the algorithm would have to perform, in expectation, $1/(1-1/n) \leq 2$ iterations till it succeeds.

12.2.2.1. An alternative deterministic construction of HST

Our construction is based on a recursive decomposition of the point set. In each stage, we split the point set into two subsets. We recursively compute an $n\mathsf{d}$-HST for each point set, and we merge the two trees into a single tree, by creating a new vertex, assigning it an appropriate value, and hang the two subtrees from this node. To carry this out, we try to separate the set into two subsets that are furthest away from each other.

Lemma 12.2.10. *Let P be a set of n points in $\mathbb{R}^{\mathsf{d}}$. One can compute an $n\mathsf{d}$-HST of P in $O(n\mathsf{d}\log^2 n)$ time (note that the constant hidden by the O notation does not depend on d).*

Proof: Let $R = R(P)$ be the minimum axis parallel box containing P, and let $\nu = \sum_{i=1}^{\mathsf{d}} \|I_i(R)\|$, where $I_i(R)$ is the projection of R to the ith dimension.

Clearly, one can find an axis parallel strip H of width $\geq \nu/((n-1)\mathsf{d})$, such that there is at least one point of P on each of its sides and there is no point of P inside H. Indeed, to find this strip, project the point set into the ith dimension, and find the longest interval between two consecutive points. Repeat this process for $i = 1, \dots, \mathsf{d}$, and use the longest interval encountered. Clearly, the strip H corresponding to this interval is of width $\geq \nu/((n-1)\mathsf{d})$. On the other hand, $\mathrm{diam}(P) \leq \nu$.

Now recursively continue the construction of two trees $\mathcal{T}^+, \mathcal{T}^-$, for P^+, P^-, respectively, where P^+, P^- is the splitting of P into two sets by H. We hung $\mathcal{T}^+$ and $\mathcal{T}^-$ on the root node v and set $\Delta(v) = \nu$. We claim that the resulting tree $\mathcal{T}$ is an $n\mathsf{d}$-HST. To this end, observe that $\mathrm{diam}(P) \leq \Delta(v)$, and for a point $p \in P^-$ and a point $q \in P^+$, we have $\|p - q\| \geq \nu/((n-1)\mathsf{d})$, which implies the claim.

To construct this efficiently, we use an efficient search trees to store the points according to their order in each coordinate. Let $\mathcal{D}_1, \dots, \mathcal{D}_{\mathsf{d}}$ be those trees, where $\mathcal{D}_i$ stores the points of P in ascending order according to the ith axis, for $i = 1, \dots, \mathsf{d}$. We modify them, such that for every node $v \in \mathcal{D}_i$, we know what the largest empty interval along the ith axis is for the points P_v (i.e., the points stored in the subtree of v in $\mathcal{D}_i$). Thus, finding the largest strip to split along can be done in $O(\mathsf{d}\log n)$ time. Now, we need to split the d trees into two families of d trees. Assume we split according to the first axis. We can split $\mathcal{D}_1$ in $O(\log n)$ time using the splitting operation provided by the search tree (Treaps [SA96] for example can do this split in $O(\log n)$ time). Let us assume that this splits P into two sets L and R, where $|L| \leq |R|$.

We still need to split the other $\mathsf{d} - 1$ search trees. This is going to be done by deleting all the points of L from those trees and building $\mathsf{d} - 1$ new search trees for L. This takes $O(|L|\,\mathsf{d}\log n)$ time. We charge this work to the points of L.

Since in every split only the points in the smaller portion of the split get charged, it follows that every point can be charged at most $O(\log n)$ time during this construction algorithm. Thus, the overall construction time is $O(\mathsf{d}n\log^2 n)$.

12.3. Low quality ANN search

We are interested in answering ***approximate nearest neighbor*** (***ANN***) queries in $\mathbb{R}^{\mathsf{d}}$. Namely, given a set P of n points in $\mathbb{R}^{\mathsf{d}}$ and a parameter $\tau > 1$, we want to preprocess P, such that given a query point q, we can compute (quickly) a point $p \in P$, such that p is a τ-approximate nearest neighbor to q in P. Formally, $\|\mathsf{q} - p\| \leq \tau \mathrm{d}(\mathsf{q}, P)$, where $\mathrm{d}(\mathsf{q}, P) = \min_{p\in P} \|\mathsf{q} - p\|$.

Here, τ is going to be relatively large (i.e., $> \mathsf{d}^{3/2}$), and as such the quality of approximation is quite low.

12.3.1. The data-structure and search procedure.

Let P be a set of n points in $\mathbb{R}^{\mathsf{d}}$, contained inside the cube $[1/2, 3/4]^{\mathsf{d}}$. Let b be a random vector in the cube $[0, 1/2]^{\mathsf{d}}$, and consider the compressed shifted quadtree $\mathcal{T}$ having the hypercube $\mathsf{b} + [0, 1]^{\mathsf{d}}$ as its root. We choose for each node v of the quadtree a representative point $\mathrm{rep}_v \in P_v$.

Given a query point $\mathsf{q} \in [1/2, 3/4]^{\mathsf{d}}$, let v be the lowest node of $\mathcal{T}$ whose region rg_v contains q.

(A) If rep_v is defined, then we return it as the ANN.

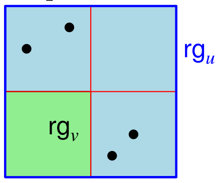
(B) If rep_v is not defined, then it must be that v is an empty leaf, so let u be its parent, and return rep_u as q's ANN.

12.3.2. Analysis

Let us consider the above search procedure. When answering a query, there are several possibilities:

(A) If rg_v is a cube (i.e., v is a leaf) and v stores a point $p \in Q$ inside it, then we return p as the ANN.

(B) v is a leaf but there is no point associated with it.



In this case rep_v is not defined, and we return rep_u, where $u = \overline{\mathrm{p}}(v)$. Observe that $\|\mathsf{q} - \text{rep}_u\| \leq 2\text{diam}(\mathsf{rg}_v)$. This case happens if the parent u of v has two children that contain points of P and v is one of the empty children. This situation is depicted in the figure on the right.

(C) If rg_v is an annulus, then v is a compressed node. In this case, we return rep_v as the ANN. Observe that $\mathrm{d}(\mathsf{q}, \text{rep}_v) \leq \text{diam}(\mathsf{rg}_v)$.

In all these cases, the distance to the ANN found is at most $2\text{diam}(\square_v)$.

Lemma 12.3.1. *For any $\tau > 1$ and a query point q, the above data-structure returns a τ-approximation to the distance to the nearest neighbor of q in P, with probability $\geq 1 - 4\mathsf{d}^{3/2}/\tau$.*

Proof: Let p be q's nearest neighbor in P (i.e., $= \|\mathsf{q} - p\| = \mathrm{d}(\mathsf{q}, P)$). Let b be the ball with diameter $\|\mathsf{q} - p\|$ that contains q and p (i.e., this is the diametrical ball of $\mathsf{q}p$). Consider the lowest node u in the compressed quadtree that contains b completely. By construction, the node v fetched by the point-location query for q must be either u or one of its descendants. As such, the ANN returned is of distance $\leq 2\text{diam}(\square_v) \leq 2\text{diam}(\square_u)$.

Let $\ell = \|\mathsf{q} - p\|$. By Lemma 12.1.3, the probability that b is fully contained inside a single cell in the ith level of $\mathcal{T}$ is at least $1 - \mathsf{d}\ell/2^i$ (i is a non-positive integer). In such a case, let $\square$ be this grid cell, and observe that the distance to the ANN returned is at most $2\text{diam}(\square) \leq 2\sqrt{\mathsf{d}}2^i$. As such, the quality of the ANN returned in such a case is bounded by $2\sqrt{\mathsf{d}}2^i/\ell$. If we want this ANN to be of quality τ, then we require that

$$2\sqrt{\mathsf{d}}2^i/\ell \leq \tau \quad \Longrightarrow \quad 2^i \leq \frac{\ell\tau}{2\sqrt{\mathsf{d}}} \quad \Longrightarrow \quad i \leq \lg \frac{\ell\tau}{2\sqrt{\mathsf{d}}}.$$

In particular, setting $i = \left\lfloor \lg\left(\ell\tau/2\sqrt{\mathsf{d}}\right)\right\rfloor$, we get that the returned point is a τ-ANN with probability at least

$$1 - \frac{\mathsf{d}\ell}{2^i} \geq 1 - \frac{4\mathsf{d}^{3/2}\ell}{\ell\tau} = 1 - \frac{4\mathsf{d}^{3/2}}{\tau},$$

implying the claim.

One thing we glossed over in the above is how to handle queries outside the square $[1/2, 3/4]^{\mathsf{d}}$. This can be handled by scaling the input point set P to lie inside a hypercube of diameter (say) $1/n^2$ centered at the middle of the domain $[1/2, 3/4]^{\mathsf{d}}$. Let T be the affine transformation (i.e., it is scaling and translation) realizing this. Clearly, the ANN to q in P is the point corresponding to the ANN to $T(\mathsf{q})$ in $T(P)$. In particular, given a query point q, we answer the ANN query on the transformed point set $T(P)$ using the query point $T(\mathsf{q})$.

Now, if the query point $T(\mathsf{q})$ is outside $[1/2, 3/4]^{\mathsf{d}}$, then any point of $T(P)$ is $(1 + 1/n)$-ANN to the query point as can be easily verified.

Combining Lemma 12.3.1 with Theorem 12.6.1$_{\text{p179}}$, we get the following result.

Theorem 12.3.2. *For a point set $P \subseteq [1/2, 3/4]^{\mathsf{d}}$, a randomly shifted compressed quadtree $\mathcal{T}$ of P can answer* ANN *queries in $O(\mathsf{d} \log n)$ time. The time to build this data-structure is $O(\mathsf{d}n \log n)$. Furthermore, for any $\tau > 1$, the returned point is a τ-*ANN *with probability $\geq 1 - 4\mathsf{d}^{3/2}/\tau$.*

Remark 12.3.3. 1. Theorem 12.3.2 is usually interesting for τ being polynomially large in n, where the probability of success is quite high. 2. Note that even for a large τ, this data-structure does not necessarily return the guaranteed quality of approximation for all the points in space. One can prove that this data-structure works with high probability for all the points in the space (but the guarantee of approximation deteriorates, naturally). See Exercise 12.5.4.

12.4. Bibliographical notes

The approximation algorithm for covering by unit disks is due to Hochbaum and Maas [HM85], and our presentation is a variant of their algorithm. Exercise 12.5.2 is similar in spirit to the work of Agarwal and Procopiuc [AP02]. The idea of HSTs was used by Bartal [Bar98] to get a similar result for more general settings.

The idea of shifted quadtrees can be derandomized [Cha98] and still yield interesting results. This is done by picking the shift carefully and inspecting the resulting $\mathcal{Q}$-order. The idea of doing point-location queries in a compressed quadtree to answer ANN queries is also from [Cha98] but it is probably found in earlier work.

The low-quality high-dimensional HST construction of Section 12.2.2.1 is taken from [Har01]. The running time of this lemma can be further improved to $O(\mathsf{d}n \log n)$ by more careful and involved implementation; see [CK95] for details.

Partition via random shifting. The improved partitions that have lower probability to separate close-by points in Euclidean space is described by Charikar et al. [CCG+98]. The idea is to randomly scope balls of radius Δ from space, till all the points in space are scooped out. Each cluster is the maximum set of points in a scooped points that are not contained in an earlier ball. This construction is not easily constructable (although this can be overcome with a bit of care), and the analysis relies on careful arguments about the volumes of balls and the volume of the intersection of two balls.

More recently, another random partition scheme was suggested that works reasonably well in practice: Randomly rotate and shift a grid. They also compute, more carefully, the probability of two points to be separated by a shifted grid. See the work by Aiger et al. [AKS12].

12.5. Exercises

Exercise 12.5.1 (My level it is nothing[④]). Prove Lemma 12.1.10.

Exercise 12.5.2 (Faster exact cover by unit disks). Let P be a set of n points in the plane that can be covered by k unit disks, and let $\mathcal{F}$ be this set of disks. Furthermore, assume that P cannot be covered by fewer unit disks.

(A) Prove that there exists an axis parallel line that passes through one of the points of P that intersects $O\left(\sqrt{k}\right)$ disks of $\mathcal{F}$.

(B) Provide an $n^{O\left(\sqrt{k}\right)}$ time algorithm for computing a cover of P by k unit disks.

(C) Given a set P of n points in the plane, show how to $(1+\varepsilon)$-approximate the minimum cover of P by a set of k disks, in $n^{O(1/\varepsilon)}$ time.

Exercise 12.5.3 (Lower bound on computing an HST). Show, by adversarial argument, that for any $t > 1$, we have the following: Any algorithm computing an HST $\mathtt{H}$ for n points in a metric space $\mathcal{M}$ that t-approximates $d_{\mathcal{M}}$ must in the worst case inspect all $\binom{n}{2}$ distances in the metric. Thus, computing an HST requires quadratic time in the worst case.

Exercise 12.5.4 (ANN for all). (Hard) Prove that the data-structure of Theorem 12.3.2 answers n^{10}-ANN queries for all the points in the space with high probability.

12.6. From previous lectures

Theorem 12.6.1. *Assuming one can compute the $\mathcal{Q}$-order in $O(\mathsf{d})$ time for two points $\mathbb{R}^{\mathsf{d}}$, then one can maintain a compressed quadtree of a set of points in $\mathbb{R}^{\mathsf{d}}$, in $O(\mathsf{d}\log n)$ time per operation. The operations of insertion, deletion, and point-location query are supported. Furthermore, this can be implemented using any data-structure for ordered-set that supports insertion/deletion/predecessor operations in logarithmic time.*

In particular, one can construct a compressed quadtree of a set of n points in $\mathbb{R}^{\mathsf{d}}$ in $O(\mathsf{d}n\log n)$ time.

Definition 12.6.2. A ***metric space*** is a pair $(\mathcal{X}, \mathrm{d})$ where $\mathcal{X}$ is a set and $\mathrm{d} : \mathcal{X} \times \mathcal{X} \to [0, \infty)$ is a ***metric*** satisfying the following axioms: (i) $d_{\mathcal{M}}(x, y) = 0$ if and only if $x = y$, (ii) $d_{\mathcal{M}}(x, y) = d_{\mathcal{M}}(y, x)$, and (iii) $d_{\mathcal{M}}(x, y) + d_{\mathcal{M}}(y, z) \geq d_{\mathcal{M}}(x, z)$ (triangle inequality).

Definition 12.6.3 (Bit index). Let $\alpha, \beta \in [0, 1)$ be two real numbers. Assume these numbers in base two are written as $\alpha = 0.\alpha_1\alpha_2\ldots$ and $\beta = 0.\beta_1\beta_2\ldots$. Let $\mathrm{bit}_{\Delta}(\alpha, \beta)$ be the index of the first bit after the period in which they differ.

Theorem 12.6.4. *Given a set P of n points in $\mathbb{R}^{\mathsf{d}}$ and a parameter $1 \geq \varepsilon > 0$, one can compute a $(1+\varepsilon)$-spanner of P with $O\left(n\varepsilon^{-\mathsf{d}}\right)$ edges, in $O\left(n\log n + n\varepsilon^{-\mathsf{d}}\right)$ time.*

Theorem 12.6.5. *For set P of n points in the plane, one can compute the closest pair of P in expected linear time.*

References

[AKS12] D. Aiger, H. Kaplan, and M. Sharir. *Reporting Neighbors in High-Dimensional Euclidean Space.* manuscript. 2012.

[AP02] P. K. Agarwal and C. M. Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33(2): 201–226, 2002.

[Bar98] Y. Bartal. On approximating arbitrary metrics by tree metrics. *Proc. 30th Annu. ACM Sympos. Theory Comput.* (STOC), 161–168, 1998.

[CCG+98] M. Charikar, C. Chekuri, A. Goel, S. Guha, and S. A. Plotkin. Approximating a finite metric by a small number of tree metrics. *Proc. 39th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), 379–388, 1998.

[Cha98] T. M. Chan. Approximate nearest neighbor queries revisited. *Discrete Comput. Geom.*, 20(3): 359–373, 1998.

[CK95] P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *J. Assoc. Comput. Mach.*, 42(1): 67–90, 1995.

[Har01] S. Har-Peled. A replacement for Voronoi diagrams of near linear size. *Proc. 42nd Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), 94–103, 2001.

[HM85] D. S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *J. Assoc. Comput. Mach.*, 32(1): 130–136, 1985.

[SA96] R. Seidel and C. R. Aragon. Randomized search trees. *Algorithmica*, 16: 464–497, 1996.