Main topics are well-separated-pair-decompositions.

Well Separated Pair Decomposition (WSPD)

- Today, we’re going to discuss some applications of well separated pair decompositions (WSPDs).
- Let’s start with a review. Given a separation parameter $s > 0$, point sets $A$ and $B$ are $s$-well separated if $A$ and $B$ can be enclosed in sphere of radius $r$ that are distance at least $sr$ apart.
- An $s$-well separated pair decomposition ($s$-WSPD) of point set $P$ is a collection of pairs of subsets $\{\{A_1, B_1\}, \{A_2, B_2\}, \ldots, \{A_m, B_m\}\}$ such that
  1. $A_i, B_i \subseteq P$ for all $1 \leq i \leq m$
  2. $A_i \cap B_i = \emptyset$ for all $1 \leq i \leq m$
  3. $\bigcup_{i=1}^{m} A_i \otimes B_i = P \otimes P$
  4. $A_i$ and $B_i$ are $s$-well separated for all $1 \leq i \leq m$

  where $A \otimes B$ is the set of unordered pairs from $A$ and $B$.
- Last time, we saw how (for any $s \geq 2$), there exists an $s$-WSPD of size $O(s^d n)$ which can be constructed in $O(n \log n + s^d n)$ time.
- The WSPD can be represented as a set of unordered pairs of nodes from a compressed quadtree of $P$.
- For any node $u$, we’ll let $P_u$ be the points in $u$’s cell and let $\text{rep}(u)$ denote an arbitrary representative point in from $P_u$. We can compute these in representatives in $O(n)$ time given the compressed quadtree.
- Lemma: (WSPD Utility Lemma) If the pair $\{P_{-u}, P_{-v}\}$ is $s$-well separated and $x, x’$ in $P_{-u}$ and $y, y’$ in $P_{-v}$ then:
  1. $||x x’|| \leq 2/s \cdot ||x y||$
  2. $||x’ y’|| \leq (1 + 4/s) \cdot ||x y||$

  In other words, points within a subset are much closer than points between subsets.
- Proof:
  1. We can enclose $P_{-u}$ and $P_{-v}$ in balls of radius $r$ that are $sr$ distance apart.
  2. Therefore, $||x x’|| \leq 2r = (2r / sr) \cdot sr \leq (2r / sr) \cdot ||x y|| = 2 / s \cdot ||x y||$.
  3. And between triangle inequality and claim i., $||x’ y’|| \leq ||x’ x|| + ||x y|| + ||y y’|| \leq 2 / s \cdot ||x y|| + ||x y|| + 2 / s \cdot ||x y|| = (1 + 4/s) \cdot ||x y||$.
- Now we can look at some applications.

Approximating the Diameter
The diameter of a point set is the maximum distance between any pair of points in the set. We could compute it exactly in \( O(n^2) \) time by trying all pairs of points, and there’s an \( O(n \log n) \) time algorithm for points in the plane, but let’s find a fast \((1 + \varepsilon)\)-approximation algorithm for point sets in any constant dimensional Euclidean space.

Given \( \varepsilon \), let \( s = 4/\varepsilon \) and construct an \( s \)-WSPD. Let \( p_u = \text{rep}(u) \) and \( p_v = \text{rep}(v) \) for any pair of quadtree nodes \( u \) and \( v \). For every well-separated pair \( \{P_u, P_v\} \), compute \( ||p_u p_v|| \) and output the largest distance computed.

There are \( O(s^d n) \) distances computed, so the whole thing takes \( O(n \log n + s^d n) = O(n \log n + n/\varepsilon^d) \), which is \( O(n \log n) \) if \( \varepsilon \) is a constant.

To prove correctness, let \( x \) and \( y \) be the points realizing the diameter and let \( \{P_u, P_v\} \) be the well-separated pair containing \( x \) and \( y \) respectively.

By the Utility Lemma, \( ||x y|| \leq (1 + 4/s) ||p_u p_v|| = (1 + \varepsilon) ||p_u p_v|| \).

\( \{x, y\} \) is the diametrical pair, so \( ||x y|| / (1 + \varepsilon) \leq ||p_u p_v|| \leq ||x y|| \). We have a \((1 + \varepsilon)\)-approximation.

**Closest Pair**

- We can try the same algorithm for solving the closest pair problem: for every well-separated pair \( \{P_u, P_v\} \), compute \( ||p_u p_v|| \) and output the smallest distance computed.
- The surprising thing is that this algorithm actually find the exact closest pair as long as \( s \) is large enough.
- Say \( \{x, y\} \) is the closest pair and that \( s > 2 \). Again let \( \{P_u, P_v\} \) be the well-separated pair containing \( x \) and \( y \) respectively.
- \( P_u \) and \( P_v \) lie in balls of radius \( r \) at distance at least \( sr > 2r \) apart, so \( ||p_u x|| \leq 2r < sr \leq ||x y|| \).
- If \( p_u \neq x \), then this contradicts \( x \) and \( y \) being the closest pair. \( p_v = y \) for the same reason. So the representatives we tested must actually be the closest pair!
- We can set \( s \) arbitrarily close to 2 so the running time of the algorithm is \( O(n \log n + 2^d n) = O(n \log n) \), assuming \( d \) is a constant.

**Spanner Graphs**
• Recall we can express all pairwise distances using between points in P using the Euclidean graph, the complete graph with edge weights equal to the distance between its endpoints.
• Unfortunately, it is a dense graph with Theta(n^2) edges. It would be nice to find a sparse graph with far fewer edges.
• Given a stretch factor $t \geq 1$, a subgraph G of the Euclidean graph is called a t-spanner if for any pair of points $x, y$ in P we have $||x y|| \leq \delta_G(x, y) \leq t * ||x y||$ where $\delta_G(x, y)$ is the shortest path distance between $x$ and $y$ in G.
• I claimed in an earlier lecture that the Delaunay triangulation is a t-spanner for some $1.5846 \leq t \leq 2.418$. This observation does not generalize to higher dimensions, and maybe we want better approximations of distance anyway.
• So here’s what we’ll do. Pick some $s \geq 2$. We’ll make a more concrete choice later.
• Compute an s-WSPD, and for each well-separated pair $\{P_u, P_v\}$, with representatives $p_u = \text{rep}(u)$ and $p_v = \text{rep}(v)$, add edge $p_u p_v$ to the graph.
• G has $O(s^d n)$ edges and takes $O(n \log n + s^d)$ time to construct.

But is it a spanner? We need to prove for any $x, y$ in P, $||x y|| \leq \delta_G(x, y) \leq t * ||x y||$.

• The first inequality is true, because G is a subgraph of the Euclidean graph.
• We’ll prove the second inequality by induction on the Euclidean distance between two points.
• First, if $x$ and $y$ are joined by an edge in G, then $\delta_G(x, y) = ||x y|| \leq t * ||x y||$.
• Now suppose otherwise. Again let $\{P_u, P_v\}$ be the well-separated pair containing $x$ and $y$ respectively.
• By the triangle inequality,

$$
\delta_G(x, y) \leq \delta_G(x, p_u) + \delta_G(p_u, p_v) + \delta_G(p_v, y) \\
\leq \delta_G(x, p_u) + ||p_u p_v|| + \delta_G(p_v, y).
$$

• By the Utility Lemma, $\max(||x p_u||, ||p_v y||) \leq 2 / s * ||x y|| \leq ||x y||$, and $||p_u p_v|| \leq (1 + 4 / s) ||x y||$.
• We can apply induction to say

$$
\delta_G(x, y) \leq t \left(2 \cdot \frac{2}{s} ||x y|| + \left(1 + \frac{4}{s}\right) ||x y||\right) = \left(1 + \frac{4(t + 1)}{s}\right) ||x y||.
$$

So now to make the inequality work out, we just need $1 + 4(t + 1) / s \leq t$. So, set $s := 4(t + 1) / (t - 1)$. 

![WSPD and Spanner Diagram]
Now

$$\delta_G(x, y) \leq \left(1 + \frac{4(t + 1)}{4(t + 1)/(t - 1)}\right) \|xy\| = (1 + (t - 1))\|xy\| = t \cdot \|xy\|,$$

Spanners are most interesting for small stretch factors, so let’s assume $t = 1 + \varepsilon$ for some $0 < \varepsilon \leq 1$.

The size of the spanner is

$$O(s^d n) = O\left(\left(\frac{4(1 + \varepsilon)}{(1 + \varepsilon) - 1}\right)^d n\right) \leq O\left(\frac{12}{\varepsilon}^d n\right) = O\left(\frac{n}{\varepsilon^d}\right).$$

And it takes $O(n \log n + n / \varepsilon^d)$ time to build the thing.

**Euclidean MST**

Let’s finish up with a fast approximation algorithm for Euclidean Minimum Spanning Tree (MST).

Computing the MST directly from the Euclidean graph takes $\Theta(n^2)$ time.

Earlier, we discussed how the Delaunay triangulation actually contains the MST, giving us an $O(n \log n)$ time exact algorithm for the plane. What we’ll do now works as an approximation for any constant dimension.

First, we construct a $(1 + \varepsilon)$-spanner $G = (V, E)$ using the algorithm we just discussed.

Then, we compute the MST of $G$ in $O(V \log V + E)$ time using a variant of Prim’s algorithm with Fibonacci heaps. The total running time is $O(n \log n + n / \varepsilon^d)$.

To see why it works, let $w(x, y) = \|x y\|$. For any subgraph $H$ of the Euclidean graph, let $w(H)$ be the total weight of its edges. Finally, let $\pi_G(x, y)$ denote the shortest path from $x$ to $y$ in $G$ so that $w(\pi_G(x, y)) = \delta_G(x, y) \leq (1 + \varepsilon)\|x y\|$.

Let $T$ be the minimum spanning tree. Form $G'$ subset $G$ by taking the union of edges of $\pi_G(x, y)$ for all $xy$ in $T$. In other words, each edge of $T$ is replaced by its shortest path in the spanner.

$G'$ must be connected, but it may not be a tree.

We have $w(G')$

- $= \sum_{xy \in T} w(\pi_G(x, y))$
- $\leq \sum_{xy \in T} (1 + \varepsilon)\|x y\|$
- $= (1 + \varepsilon) \sum_{xy \in T} \|x y\|$
- $= (1 + \varepsilon) w(T)$

On the other hand, we have less options when building the MST of $G'$ than the MST of $G$, so the MST of $G$ must weigh less than the MST of $G'$.

We conclude $w(MST(G)) \leq w(MST(G')) \leq w(G') \leq (1 + \varepsilon) w(T)$. 