Euclidean Minimum Spanning Trees

- We need to examine only the edges of the Delaunay triangulation.
- We must compute the minimum spanning tree of a planar graph.
- MST in planar graphs can be computed in $O(n)$ time [Cheriton & Tarjan '76]

A set of points, its Voronoi diagram, and its EMST.

**Theorem 6.1.** An EMST of a set $S$ of $N$ points in the plane can be computed from the Delaunay Triangulation of $S$ in optimal time $\theta(N)$.

Combining this result with the fact that the Delaunay triangulation is computable in time $\theta(N \log N)$ we have

**Corollary 6.1.** An EMST of a set $S$ of $N$ points in the plane can be computed in optimal time $\theta(N \log N)$. 
Problem P.9 (Euclidean Traveling Salesman). Find a shortest closed path through $N$ given points in the plane.

A traveling salesman tour.

Theorem 6.2 [Rosenkrantz–Stearns–Lewis (1974)]. A minimum spanning tree can be used to obtain an approximate TSP tour whose length is less than twice the length of a shortest tour.

Let $T^*$ be optimal EMST
\[ E(T^*) \leq E(\phi_{\text{path}}), \]
where $\phi_{\text{path}}$ is $\phi$ \& edge
\[ E(W) \leq 2E(\phi_{\text{path}}) < 2E(\phi). \]

Doubling the EMST results in an Euler tour of all vertices of the set.

"Short-cuts" on the Euler tour ensure that each vertex is visited exactly once.
Problem P.10 (Minimum Euclidean Matching). Given $2N$ points in the plane, join them in pairs by line segments whose total length is a minimum.

- Minimum weighted matching in an arbitrary graph can be found in $O(n^3)$ time. \[\text{[Edmonds' 65, Gabow' 72]}\]

Can you get a faster algo?

- $2$-appr. in $O(n \log n)$
- $\frac{3}{2}$-appr. in $O(n^3)$
- Can you do better? \[\text{OPEN}\]

A minimum Euclidean matching.

**Theorem 6.3.** An approximation to the traveling salesman problem whose length is within $3/2$ of optimal can be obtained in $O(N^3)$ time if the interpoint distances obey the triangle inequality. \[\text{[Christofides' 76]}\]

**Proof.** The following algorithm achieves the desired result on the given set $S$:

1. Find a minimum spanning tree $T^*$ of $S$.
2. Find a minimum Euclidean matching $M^*$ on the set $X \subseteq S$ of vertices of odd degree in $T^*$. ($X$ has always even cardinality in any graph.)
3. The graph $T^* \cup M^*$ is an Eulerian graph, since all of its vertices have even degree. Let $\Phi_e$ be an Eulerian circuit of it.
4. Traverse $\Phi_e$ edge by edge and bypass each previously visited vertex. $\Phi_{\text{appr}}$ is the resulting tour.

- $\ell(T^*) < \ell(\Phi)$ (like before)
- $\ell(M^*) \leq \frac{1}{2} \ell(\Phi)$ (take every second edge in $\Phi$)
- $\ell(\Phi_{\text{appr}}) \leq \ell(\Phi_e)$ (from $\Delta$ inequality)
- Hence $\text{length}(\Phi_{\text{appr}}) \leq \text{length}(\Phi_e)$

\[= \text{length}(T^*) + \text{length}(M^*)\]

\[< \text{length}(\Phi) + \frac{1}{2} \text{length}(\Phi)\]

\[= \frac{3}{2} \text{length}(\Phi)\].
Smallest Circle

- **Given:** \( n \) points in the plane
- **Find:** Smallest circle containing all points.
  \[
  \min_{\rho_0} \max_i (x_i - x_0)^2 / (y_i - y_0).
  \]
- There is a unique solution. It goes through three points or has two diameter defining points.
- Trivial algorithm \( O(n^4) \).

- Determine the diameter of the point set. STOP if the circle with this diameter contains all points. Requires \( O(n \log n) \).
- Determine \( VD_{n-1}(S) \). Find circle with Voronoi point as origo and through three most remote points. Requires \( O(n \log n) \).

- \( O(n^n) \) [Rademacher, Toeplitz '57]
- \( O(n^2) \) [Elzinga, Hearn '72]
- \( O(n \log n) \) [Shamos '78]
- \( \Theta(n) \) [Megiddo '83]
Voronoi Diagrams of (n-1)-st Order
Smallest Circle

Can be solved in $\Theta(n)$ time using Megiddo’s prune and search technique for linear programming with 3 variables (PS 297-299).
Largest Empty Circle

- **Given:** \( n \) points in the plane
- **Find:** Largest empty circle with center in the convex hull of the point-set.

A largest empty circle whose center is internal to the hull.

\[
\max_{p_o \in CH(S)} \min_{i} \left( (x_i - x_o)^2 + (y_i - y_o)^2 \right)
\]

- Francis, White (1974) in \( O(n^3) \) time
- Shamos (1978) in \( O(n \log n) \) time
Largest Empty Circle

- The center is either a Voronoi point or occurs at an intersection of a Voronoi edge with the boundary of the convex hull.

- How to find the intersection?

- Voronoi edges intersect at most 2 boundary edges of the convex hull.

- Each side of the convex hull intersects at least one Voronoi edge.

- $f(x,y)$ is the distance of point $p=(x,y)$ from the nearest point in $S$.

- $f(x,y)$ is a downward-convex function of both $x$ and $y$ within a Voronoi polygon.

- Hence, $f(x,y)$ gets its maximum at a vertex of one such polygon.
Corollary 6.2. In the algebraic computation tree model, any algorithm for the MAXIMUM GAP problem on a set of \( N \) real numbers requires \( \Omega(N \log N) \) time.

In a modified computation model, however, Gonzalez (1975) has obtained the most surprising result that the problem can be actually solved in linear time. The modification consists of adding the (nonanalytic) floor function \( \lfloor \cdot \rfloor \) to the usual repertoire. Here is Gonzalez’s remarkable algorithm:

```
procedure MAX GAP
Input: \( N \) real numbers \( X[1:N] \) (unsorted)
Output: MAXGAP, the length of the largest gap between consecutive numbers in sorted order.
begin
MIN = \( \min X[i] \);
MAX = \( \max X[i] \);
create \( N-1 \) buckets by dividing the interval from MIN to MAX with \( N-2 \) equally-spaced points. In each bucket we will retain \( \text{HIGH}[i] \) and \( \text{LOW}[i] \), the largest and smallest values in bucket \( i \).
for \( i := 1 \) until \( N-1 \) do
   begin
      COUNT[\( i \)] := 0;
      LOW[\( i \)] := \text{HIGH}[\( i \)] := \( A \)
   end; (the buckets are set up*)
   (hash into buckets*)
for \( i := 1 \) until \( N-1 \) do
   begin
      BUCKET := \( 1 + \lfloor (N-1) \times (X[i] - MIN) / (MAX - MIN) \rfloor \);
      COUNT[BUCKET] := COUNT[BUCKET] + 1;
      LOW[BUCKET] := \( \min (X[i], \text{LOW}[\text{BUCKET}]) \);\(^{11}\)
      HIGH[BUCKET] := \( \max (X[i], \text{HIGH}[\text{BUCKET}]) \);\(^{11}\)
   end.
   (Note that \( N-2 \) points have been placed in \( N-1 \) buckets, so by the pigeonhole principle some bucket must be empty. This means that the largest gap cannot occur between two points in the same bucket. Now we make a single pass through the buckets*)
MAXGAP := 0;
LEFT := \text{HIGH}[1];
for \( i := 2 \) until \( N-1 \) do
   if (COUNT[i] \( = 0 \)) then
      begin
         THISGAP := \( \text{LOW}[i] \)-LEFT;
         MAXGAP := \( \max (\text{THISGAP}, \text{MAXGAP}) \);
         LEFT := \text{HIGH}[i]
      end
end.
```

This algorithm sheds some light on the computational power of the “floor” function.

\(^{11}\) Here, by convention, \( \min(x, A) = \max(x, A) = x \).