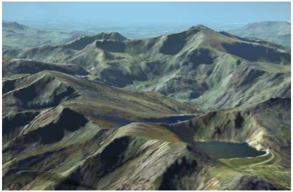
Delaunay Triangulations

Computational Geometry

Lecture 12: Delaunay Triangulations

Motivation: Terrains by interpolation

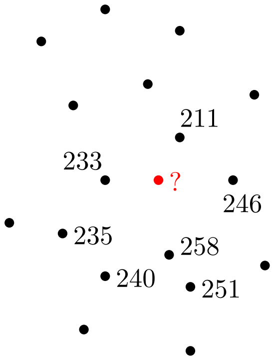
To build a model of the terrain surface, we can start with a number of sample points where we know the height.



Motivation: Terrains

How do we interpolate the height at other points?

- Nearest neighbor interpolation
- Piecewise linear interpolation by a triangulation
- Moving windows interpolation
- Natural neighbor interpolation
- ...



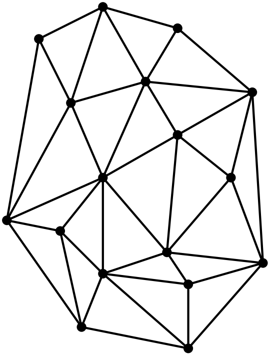
Triangulation

Let $P = \{p_1, \ldots, p_n\}$ be a point set.
A triangulation of P is a maximal planar subdivision with vertex set P.

Complexity:

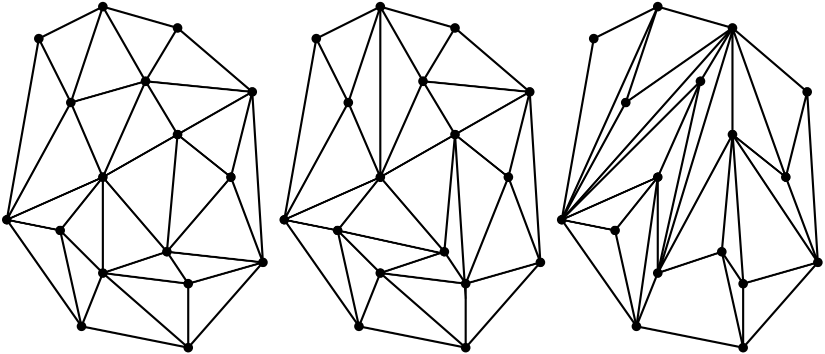
- $2n-2-k$ triangles
- $3n-3-k$ edges

where k is the number of points in P on the convex hull of P



Triangulation

But which triangulation?



Triangulation

But which triangulation?

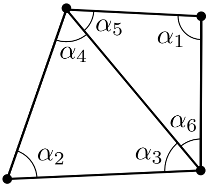
For interpolation, it is good if triangles are not long and skinny. We will try to use large angles in our triangulation.

Angle Vector of a Triangulation

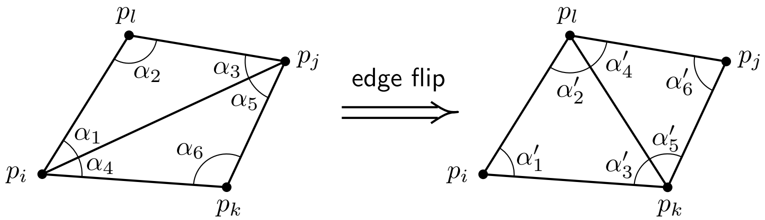
- Let $\mathcal{T}$ be a triangulation of P with m triangles. Its angle vector is $A(\mathcal{T}) = (\alpha_1, \ldots, \alpha_{3m})$ where $\alpha_1, \ldots, \alpha_{3m}$ are the angles of $\mathcal{T}$ sorted by increasing value.

- Let $\mathcal{T}'$ be another triangulation of P. We define $A(\mathcal{T}) > A(\mathcal{T}')$ if $A(\mathcal{T})$ is lexicographically larger than $A(\mathcal{T}')$

- $\mathcal{T}$ is angle optimal if $A(\mathcal{T}) \geq A(\mathcal{T}')$ for all triangulations $\mathcal{T}'$ of P



Edge Flipping

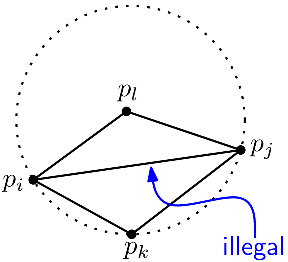


- Change in angle vector:
  $\alpha_1, \ldots, \alpha_6$ are replaced by $\alpha'_1, \ldots, \alpha'_6$
- The edge $e = \overline{p_i p_j}$ is illegal if $\min_{1 \le i \le 6} \alpha_i < \min_{1 \le i \le 6} \alpha'_i$
- Flipping an illegal edge increases the angle vector

Characterisation of Illegal Edges

How do we determine if an edge is illegal?

Lemma: The edge $\overline{p_i p_j}$ is illegal if and only if p_l lies in the interior of the circle C.

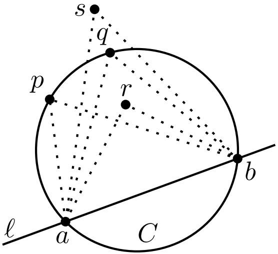


Thales Theorem

Theorem: Let C be a circle, ℓ a line intersecting C in points a and b, and p, q, r, s points lying on the same side of ℓ. Suppose that p, q lie on C, r lies inside C, and s lies outside C. Then

$$\measuredangle arb > \measuredangle apb = \measuredangle aqb > \measuredangle asb,$$

where $\measuredangle abc$ denotes the smaller angle defined by three points a, b, c.



Legal Triangulations

A legal triangulation is a triangulation that does not contain any illegal edge.

Algorithm LEGALTRIANGULATION($\mathcal{T}$)
Input. A triangulation $\mathcal{T}$ of a point set P.
Output. A legal triangulation of P.

1. **while** $\mathcal{T}$ contains an illegal edge $\overline{p_i p_j}$
2. **do** (∗ Flip $\overline{p_i p_j}$ ∗)
3. Let $p_i p_j p_k$ and $p_i p_j p_l$ be the two triangles adjacent to $\overline{p_i p_j}$.
4. Remove $\overline{p_i p_j}$ from $\mathcal{T}$, and add $\overline{p_k p_l}$ instead.
5. **return** $\mathcal{T}$

Question: Why does this algorithm terminate?

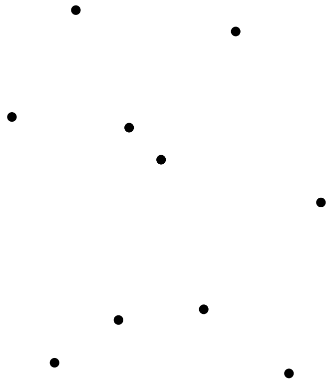
Voronoi Diagram and Delaunay Graph

Let P be a set of n points in the plane

The Voronoi diagram $\mathrm{Vor}(P)$ is the subdivision of the plane into Voronoi cells $\mathcal{V}(p)$ for all $p \in P$

Let $\mathcal{G}$ be the *dual graph* of $\mathrm{Vor}(P)$

The Delaunay graph $\mathcal{DG}(P)$ is the *straight line embedding* of $\mathcal{G}$



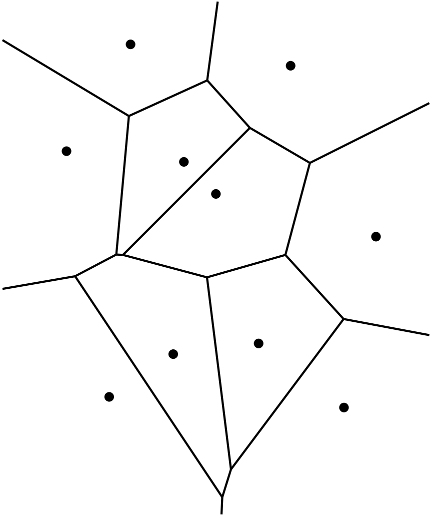
Voronoi Diagram and Delaunay Graph

Let P be a set of n points in the plane

The Voronoi diagram $\mathrm{Vor}(P)$ is the subdivision of the plane into Voronoi cells $\mathcal{V}(p)$ for all $p \in P$

Let $\mathcal{G}$ be the *dual graph* of $\mathrm{Vor}(P)$

The Delaunay graph $\mathcal{DG}(P)$ is the *straight line embedding* of $\mathcal{G}$



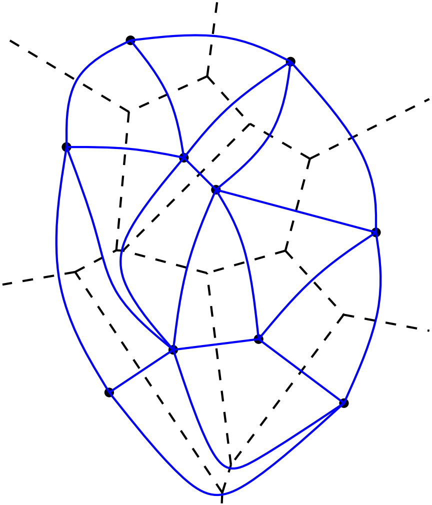
Voronoi Diagram and Delaunay Graph

Let P be a set of n points in the plane

The Voronoi diagram $\text{Vor}(P)$ is the subdivision of the plane into Voronoi cells $\mathcal{V}(p)$ for all $p \in P$

Let $\mathcal{G}$ be the *dual graph* of $\text{Vor}(P)$

The Delaunay graph $\mathcal{DG}(P)$ is the *straight line embedding* of $\mathcal{G}$



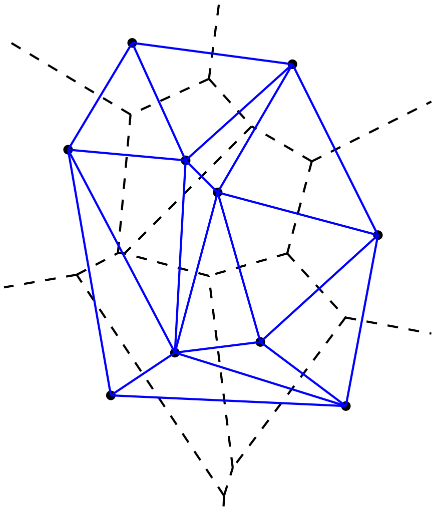
Voronoi Diagram and Delaunay Graph

Let P be a set of n points in the plane

The Voronoi diagram $\text{Vor}(P)$ is the subdivision of the plane into Voronoi cells $\mathcal{V}(p)$ for all $p \in P$

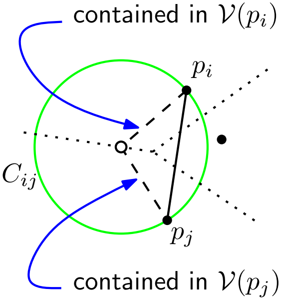
Let $\mathcal{G}$ be the *dual graph* of $\text{Vor}(P)$

The Delaunay graph $\mathcal{DG}(P)$ is the *straight line embedding* of $\mathcal{G}$



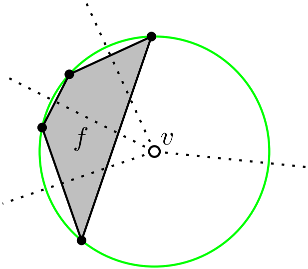
Planarity of the Delaunay Graph

Theorem: The Delaunay graph of a planar point set is a plane graph.



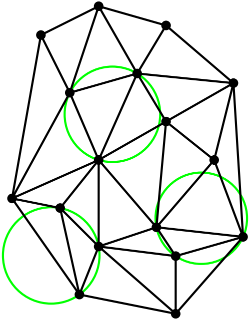
Delaunay Triangulation

If the point set P is in *general position* then the Delaunay graph is a triangulation.



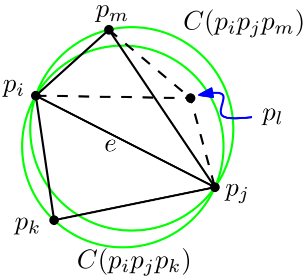
Empty Circle Property

Theorem: Let P be a set of points in the plane, and let $\mathcal{T}$ be a triangulation of P. Then $\mathcal{T}$ is a Delaunay triangulation of P if and only if the circumcircle of any triangle of $\mathcal{T}$ does not contain a point of P in its interior.



Delaunay Triangulations and Legal Triangulations

Theorem: Let P be a set of points in the plane. A triangulation $\mathcal{T}$ of P is legal if and only if $\mathcal{T}$ is a Delaunay triangulation.



Angle Optimality and Delaunay Triangulations

Theorem: Let P be a set of points in the plane. Any angle-optimal triangulation of P is a Delaunay triangulation of P. Furthermore, any Delaunay triangulation of P maximizes the minimum angle over all triangulations of P.

Computing Delaunay Triangulations

There are several ways to compute the Delaunay triangulation:

- By iterative flipping from any triangulation
- By plane sweep
- By randomized incremental construction
- By conversion from the Voronoi diagram

The last three run in $O(n \log n)$ time [expected] for n points in the plane

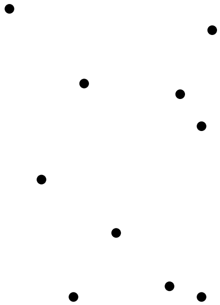
Using Delaunay Triangulations

Delaunay triangulations help in constructing various things:

- Euclidean Minimum Spanning Trees
- Approximations to the Euclidean Traveling Salesperson Problem
- α-Hulls

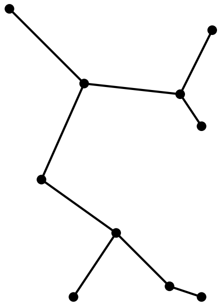
Euclidean Minimum Spanning Tree

For a set P of n points in the plane, the Euclidean Minimum Spanning Tree is the graph with minimum summed edge length that connects all points in P and has only the points of P as vertices



Euclidean Minimum Spanning Tree

For a set P of n points in the plane, the Euclidean Minimum Spanning Tree is the graph with minimum summed edge length that connects all points in P and has only the points of P as vertices



Euclidean Minimum Spanning Tree

Lemma: The Euclidean Minimum Spanning Tree does not have cycles (it really is a tree)

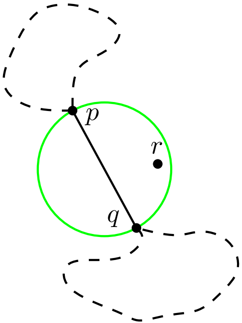
Proof: Suppose G is the shortest connected graph and it has a cycle. Removing one edge from the cycle makes a new graph G' that is still connected but which is shorter. Contradiction

Euclidean Minimum Spanning Tree

Lemma: Every edge of the Euclidean Minimum Spanning Tree is an edge in the Delaunay graph

Proof: Suppose T is an EMST with an edge $e = \overline{pq}$ that is not Delaunay

Consider the circle C that has e as its diameter. Since e is not Delaunay, C must contain another point r in P (different from p and q)

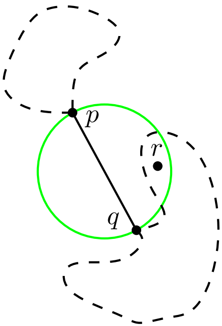


Euclidean Minimum Spanning Tree

Lemma: Every edge of the Euclidean Minimum Spanning Tree is an edge in the Delaunay graph

Proof: (continued)

Either the path in T from r to p passes through q, or vice versa. The cases are symmetric, so we can assume the former case



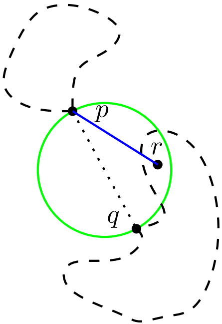
Euclidean Minimum Spanning Tree

Lemma: Every edge of the Euclidean Minimum Spanning Tree is an edge in the Delaunay graph

Proof: (continued)

Then removing e and inserting $\overline{pr}$ instead will give a connected graph again (in fact, a tree)

Since q was the furthest point from p inside C, r is closer to q, so T was not a *minimum* spanning tree. Contradiction



Euclidean Minimum Spanning Tree

How can we compute a Euclidean Minimum Spanning Tree efficiently?

From your Data Structures course: A data structure exists that maintains disjoint sets and allows the following two operations:

- **Union**: Takes two sets and makes one new set that is the union (destroys the two given sets)
- **Find**: Takes one element and returns the name of the set that contains it

If there are n elements in total, then all **Union**s together take $O(n \log n)$ time and each **Find** operation takes $O(1)$ time

Euclidean Minimum Spanning Tree

Let P be a set of n points in the plane for which we want to compute the EMST

1. Make a Union-Find structure where every point of P is in a separate set
2. Construct the Delaunay triangulation DT of P
3. Take all edges of DT and sort them by length
4. For all edges e from short to long:
   - Let the endpoints of e be p and q
   - If **Find**$(p) \neq$ **Find**(q), then put e in the EMST, and **Union**(**Find**(p),**Find**(q))

Euclidean Minimum Spanning Tree

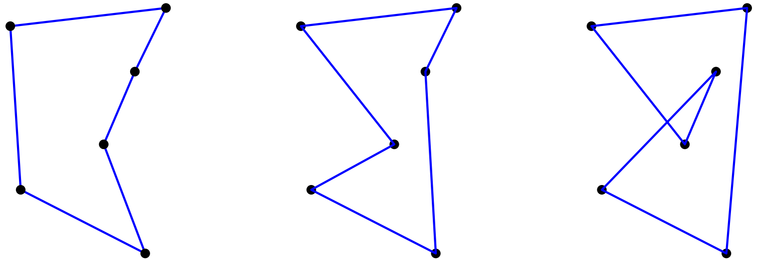
Step 1 takes linear time, the other three steps take $O(n \log n)$ time

Theorem: Let P be a set of n points in the plane. The Euclidean Minimum Spanning Tree of P can be computed in $O(n \log n)$ time

The traveling salesperson problem

Given a set P of n points in the plane, the Euclidean Traveling Salesperson Problem is to compute a tour (cycle) that visits all points of P and has minimum length

A tour is an *order* on the points of P (more precisely: a cyclic order). A set of n points has $(n-1)!$ different tours



The traveling salesperson problem

We can determine the length of each tour in $O(n)$ time: a brute-force algorithm to solve the Euclidean Traveling Salesperson Problem (ETSP) takes $O(n) \cdot O((n-1)!) = O(n!)$ time

How bad is $n!$?

Efficiency

n	n^2	2^n	$n!$
6	36	64	720
7	49	128	5040
8	64	256	40K
9	81	512	360K
10	100	1024	3.5M
15	225	32K	2,000,000T
20	400	1M	
30	900	1G	

Clever algorithms can solve instances in $O(n^2 \cdot 2^n)$ time

Approximation algorithms

If an algorithm A solves an optimization problem always within a factor k of the optimum, then A is called an k-approximation algorithm

If an instance I of ETSP has an optimal solution of length L, then a k-approximation algorithm will find a tour of length $\leq k \cdot L$

Approximation algorithms

Consider the diameter problem of a set of n points. We can compute the real value of the diameter in $O(n \log n)$ time

Suppose we take any point p, determine its furthest point q, and return their distance. This takes only $O(n)$ time

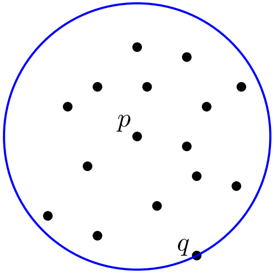
Question: Is this an approximation algorithm?

Approximation algorithms

Consider the diameter problem of a set of n points. We can compute the real value of the diameter in $O(n \log n)$ time

Suppose we take any point p, determine its furthest point q, and return their distance. This takes only $O(n)$ time

Question: Is this an approximation algorithm?



Approximation algorithms

Suppose we determine the point with minimum x-coordinate p and the point with maximum x-coordinate q, and return their distance. This takes only $O(n)$ time

Question: Is this an approximation algorithm?

Approximation algorithms

Suppose we determine the point with minimum x-coordinate p and the point with maximum x-coordinate q, and return their distance. This takes only $O(n)$ time

Question: Is this an approximation algorithm?

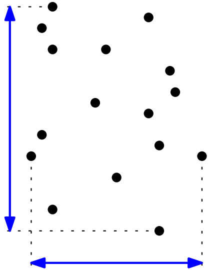
Approximation algorithms

Suppose we determine the point with minimum x-coordinate p and the point with maximum x-coordinate q.
Then we determine the point with minimum y-coordinate r and the point with maximum y-coordinate s.
We return $\max(d(p,q),\, d(r,s))$.
This takes only $O(n)$ time

Question: Is this an approximation algorithm?

Approximation algorithms

Suppose we determine the point with minimum x-coordinate p and the point with maximum x-coordinate q.
Then we determine the point with minimum y-coordinate r and the point with maximum y-coordinate s.
We return $\max(d(p,q),\, d(r,s))$.
This takes only $O(n)$ time

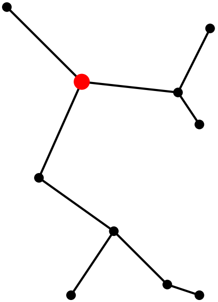


Question: Is this an approximation algorithm?

Approximation algorithms

Back to Euclidean Traveling Salesperson:

We will use the EMST to approximate the ETSP

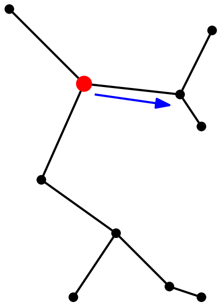


start at any vertex

Approximation algorithms

Back to Euclidean Traveling Salesperson:

We will use the EMST to approximate the ETSP

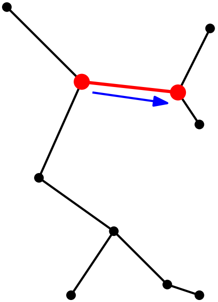


follow an edge on one side

Approximation algorithms

Back to Euclidean Traveling Salesperson:

We will use the EMST to approximate the ETSP

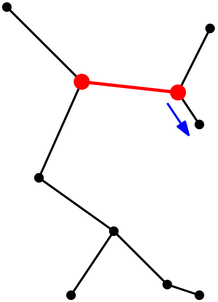


. . . to get to another vertex

Approximation algorithms

Back to Euclidean Traveling Salesperson:

We will use the EMST to approximate the ETSP

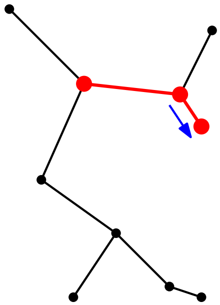


proceed this way

Approximation algorithms

Back to Euclidean Traveling Salesperson:

We will use the EMST to approximate the ETSP

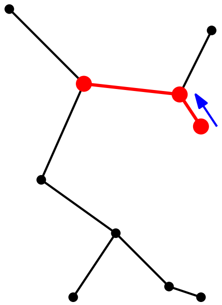


proceed this way

Approximation algorithms

Back to Euclidean Traveling Salesperson:

We will use the EMST to approximate the ETSP

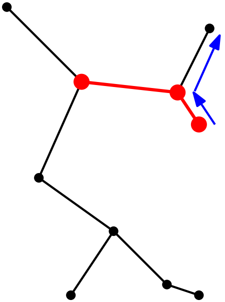


proceed this way

Approximation algorithms

Back to Euclidean Traveling Salesperson:

We will use the EMST to approximate the ETSP

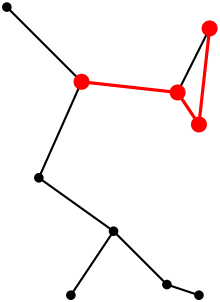


skipping visited vertices

Approximation algorithms

Back to Euclidean Traveling Salesperson:

We will use the EMST to approximate the ETSP

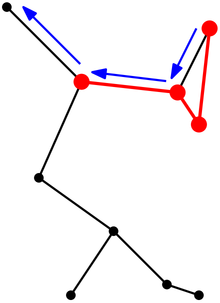


skipping visited vertices

Approximation algorithms

Back to Euclidean Traveling Salesperson:

We will use the EMST to approximate the ETSP

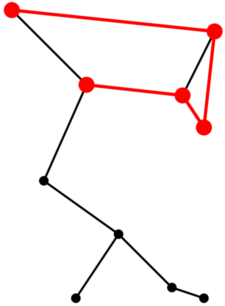


skipping visited vertices

Approximation algorithms

Back to Euclidean Traveling Salesperson:

We will use the EMST to approximate the ETSP

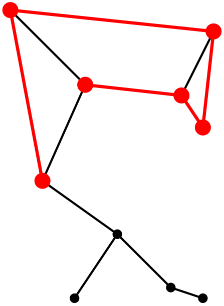


skipping visited vertices

Approximation algorithms

Back to Euclidean Traveling Salesperson:

We will use the EMST to approximate the ETSP

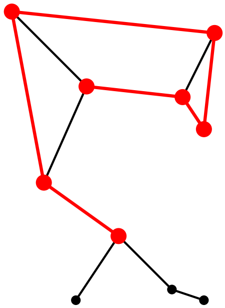


skipping visited vertices

Approximation algorithms

Back to Euclidean Traveling Salesperson:

We will use the EMST to approximate the ETSP

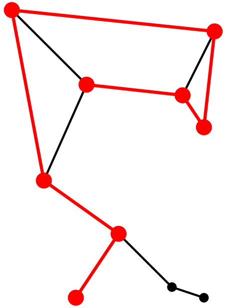


skipping visited vertices

Approximation algorithms

Back to Euclidean Traveling Salesperson:

We will use the EMST to approximate the ETSP

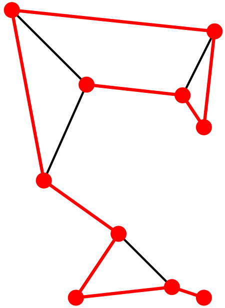


skipping visited vertices

Approximation algorithms

Back to Euclidean Traveling Salesperson:

We will use the EMST to approximate the ETSP

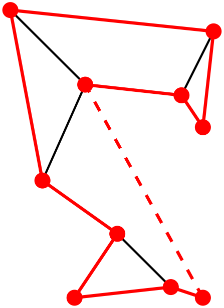


skipping visited vertices

Approximation algorithms

Back to Euclidean Traveling Salesperson:

We will use the EMST to approximate the ETSP

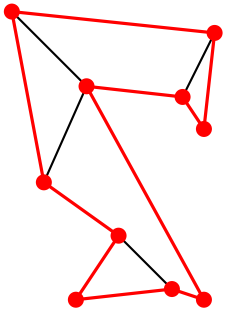


and close the tour

Approximation algorithms

Back to Euclidean Traveling Salesperson:

We will use the EMST to approximate the ETSP

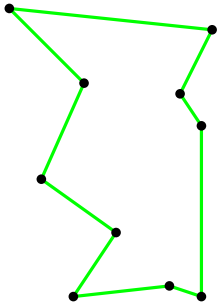


and close the tour

Approximation algorithms

Why is this tour an approximation?

- The walk visits every edge twice, so it has length $2 \cdot |EMST|$
- The tour skips vertices, which means the tour has length $\leq 2 \cdot |EMST|$
- The optimal ETSP-tour is a spanning tree if you remove any edge!!!
  So $|EMST| < |ETSP|$



optimal ETSP-tour

Approximation algorithms

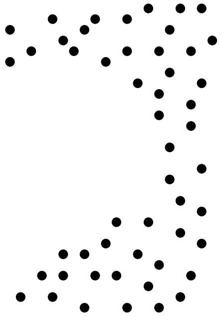
Theorem: Given a set of n points in the plane, a tour visiting all points whose length is at most twice the minimum possible can be computed in $O(n \log n)$ time

In other words: an $O(n \log n)$ time, 2-approximation for ETSP exists

α-Shapes

Suppose that you have a set of points in the plane that were sampled from a shape

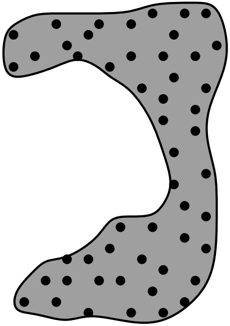
We would like to reconstruct the shape



α-Shapes

Suppose that you have a set of points in the plane that were sampled from a shape

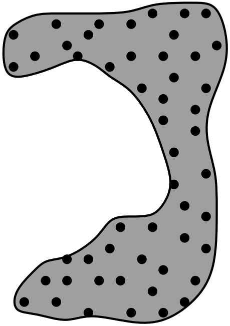
We would like to reconstruct the shape



α-Shapes

An α-disk is a disk of radius α

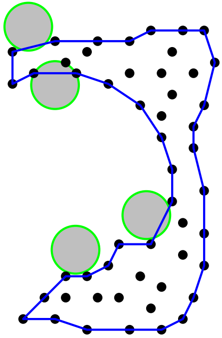
The α-shape of a point set P is the graph with the points of P as the vertices, and two vertices p, q are connected by an edge if there exists an α-disk with p and q on the boundary but no other points if P inside or on the boundary



α-Shapes

An α-disk is a disk of radius α

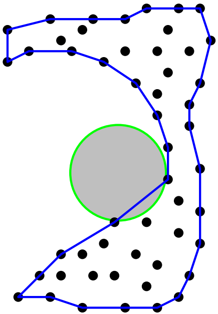
The α-shape of a point set P is the graph with the points of P as the vertices, and two vertices p, q are connected by an edge if there exists an α-disk with p and q on the boundary but no other points if P inside or on the boundary



α-Shapes

An α-disk is a disk of radius α

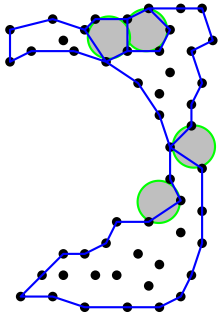
The α-shape of a point set P is the graph with the points of P as the vertices, and two vertices p, q are connected by an edge if there exists an α-disk with p and q on the boundary but no other points if P inside or on the boundary



α-Shapes

An α-disk is a disk of radius α

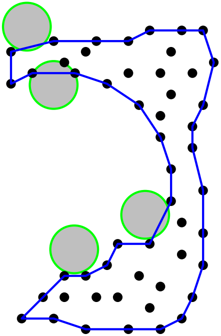
The α-shape of a point set P is the graph with the points of P as the vertices, and two vertices p, q are connected by an edge if there exists an α-disk with p and q on the boundary but no other points if P inside or on the boundary



α-Shapes

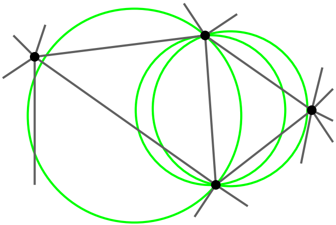
Because of the empty disk property of Delaunay triangulations (each Delaunay edge has an empty disk through its endpoints), every α-shape edge is also a Delaunay edge

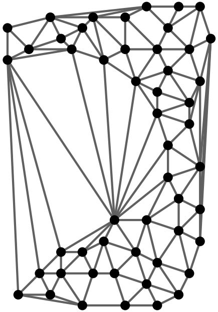
Hence: there are $O(n)$ α-shape edges, and they cannot properly intersect



α-Shapes

Given the Delaunay triangulation, we can determine for any edge all sizes of empty disks through the endpoints in $O(1)$ time





So the α-shape can be computed in $O(n \log n)$ time

Conclusions

The Delaunay triangulation is a versatile structure that can be computed in $O(n \log n)$ time for a set of n points in the plane

Approximation algorithms are like heuristics, but they come with a guarantee on the quality of the approximation. They are useful when an optimal solution is too time-consuming to compute