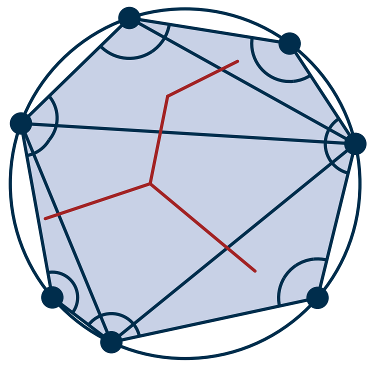
Computational Geometry

Exercise 6 *Assignment 5, 6 and Greedy Routing in Hyperbolic Geometry*

Jean-Pierre, Marcus, Wendy

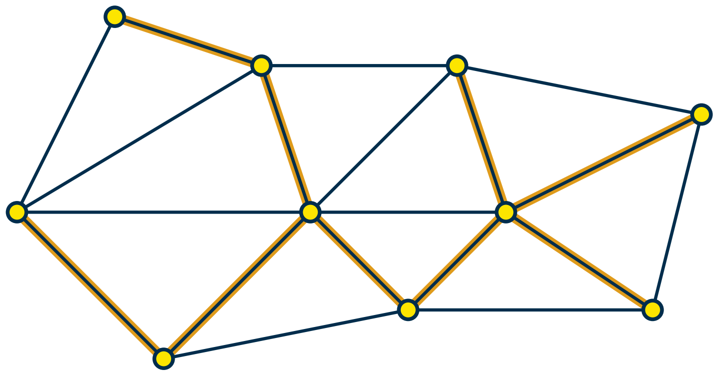
Assignment 5

Triangulation of Co-Circular Points



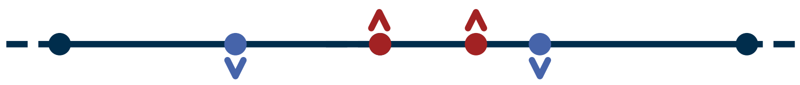
- Find optimal triangulation (smallest angle vector)

MST ⊆ Delaunay



- Prove that an MST is a subset of the Delaunay triangulation

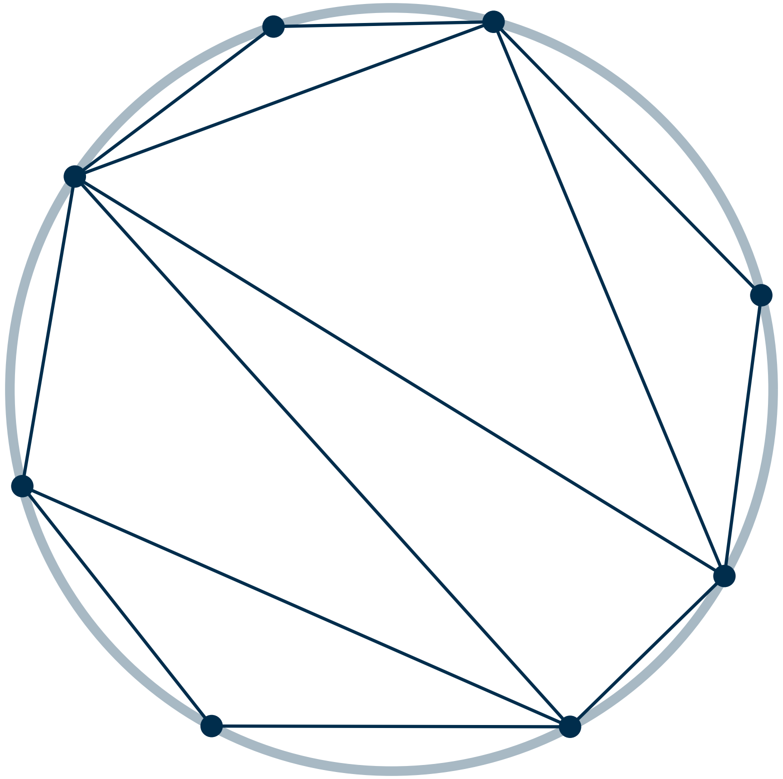
Foldability of Mountain/Valley Patterns



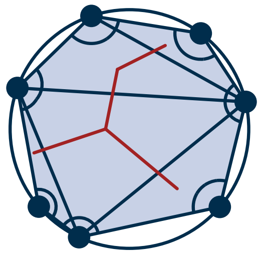
- test foldability in $O(n)$

Triangulation of Co-Circular Points

Vector of angles is optimized ⇔ Vector of lengths is optimized

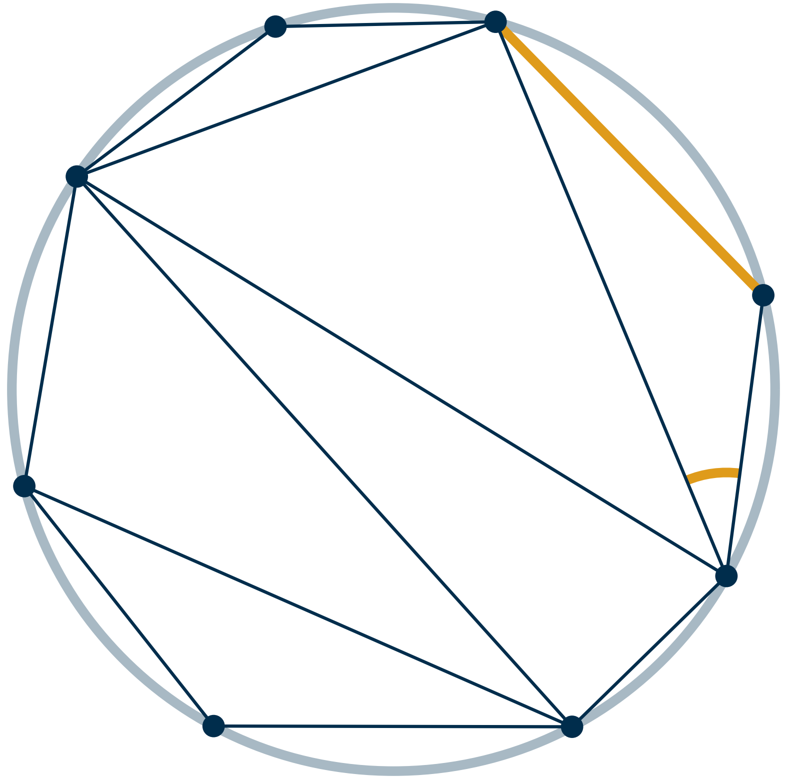


Triangulation of Co-Circular Points

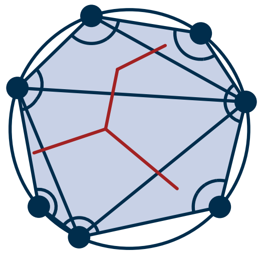


Vector of angles is optimized ⇔ Vector of lengths is optimized

- Every polygon edge has one opposite angle

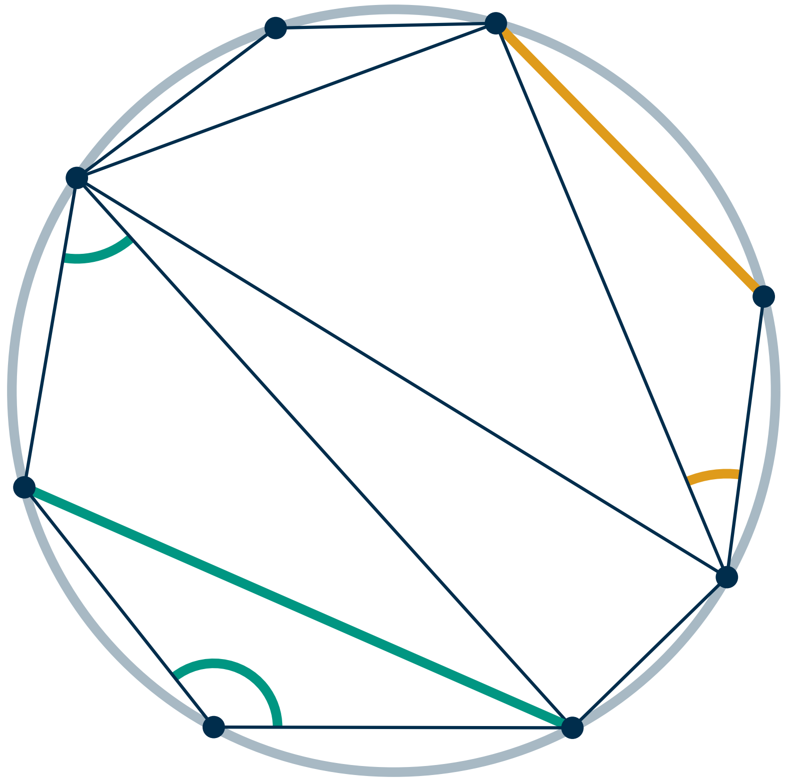


Triangulation of Co-Circular Points

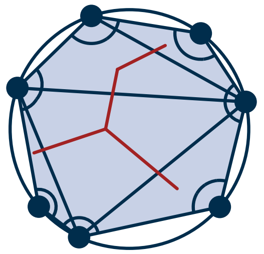


Vector of angles is optimized ⇔ Vector of lengths is optimized

- Every polygon edge has one opposite angle
- Every internal edge has a small and large opposing angle

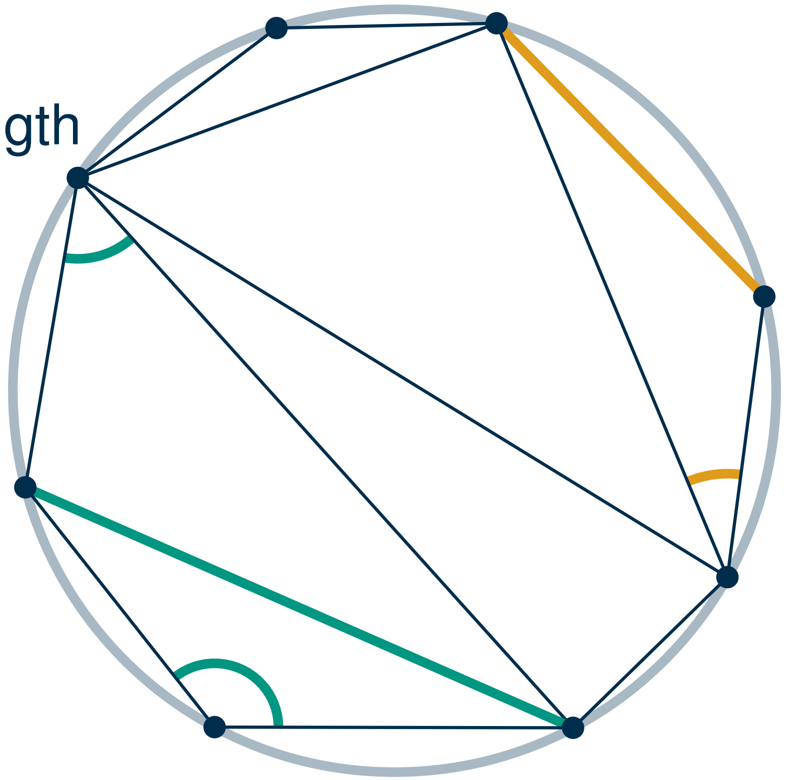


Triangulation of Co-Circular Points

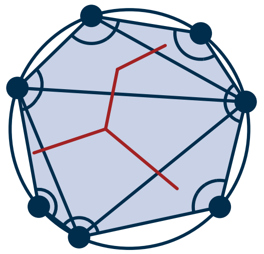


Vector of angles is optimized ⇔ Vector of lengths is optimized

- Every polygon edge has one opposite angle
- Every internal edge has a small and large opposing angle
- Generalized Thales: angle is only dependent on the edge length

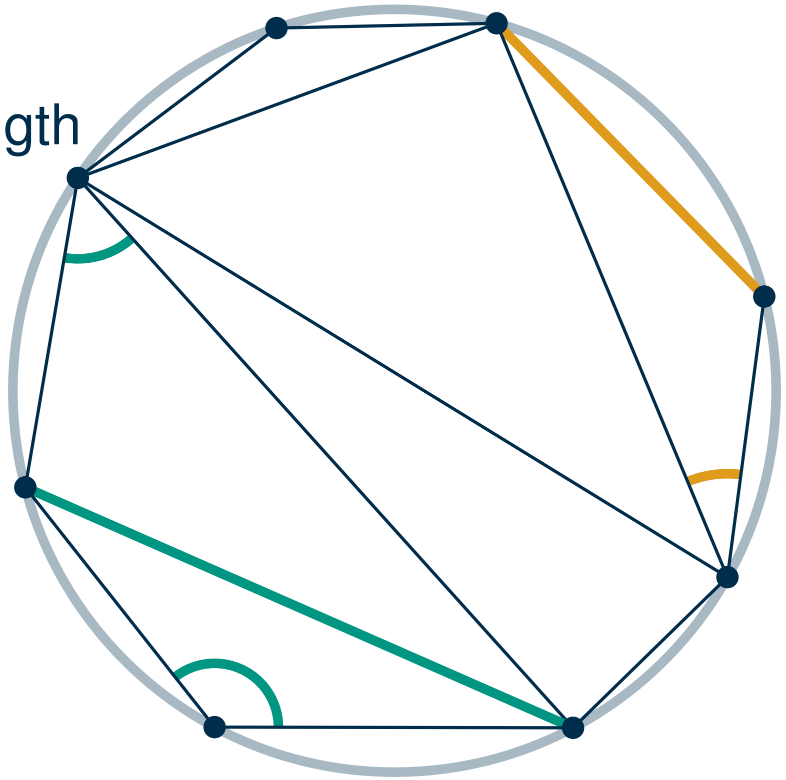


Triangulation of Co-Circular Points

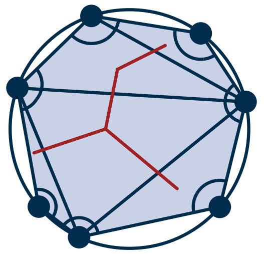


Vector of angles is optimized ⇔ Vector of lengths is optimized

- Every polygon edge has one opposite angle
- Every internal edge has a small and large opposing angle
- Generalized Thales: angle is only dependent on the edge length
- The smaller angle is monotone in the internal edge length

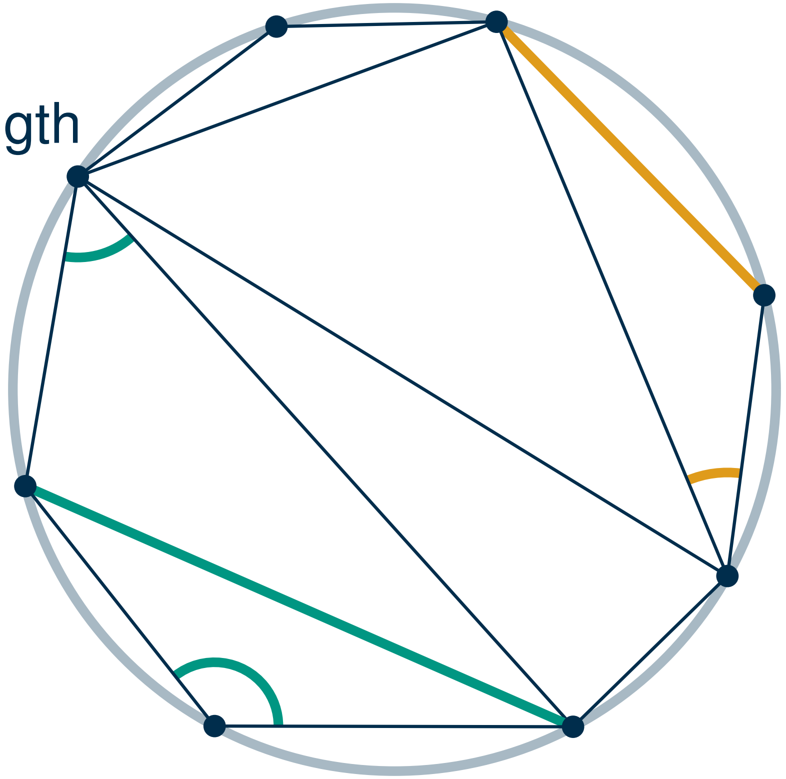


Triangulation of Co-Circular Points

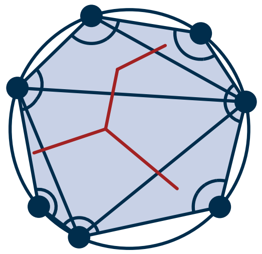


Vector of angles is optimized ⇔ Vector of lengths is optimized

- Every polygon edge has one opposite angle
- Every internal edge has a small and large opposing angle
- Generalized Thales: angle is only dependent on the edge length
- The smaller angle is monotone in the internal edge length
- Polygon edges are irrelevant.
  Consider first entry where two length vectors differ

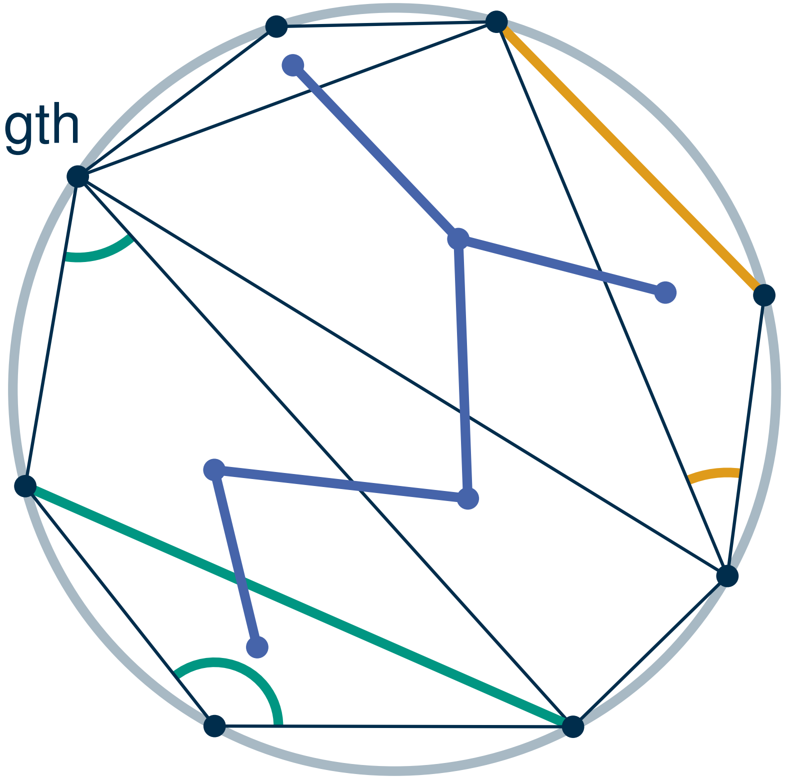


Triangulation of Co-Circular Points



Vector of angles is optimized ⇔ Vector of lengths is optimized

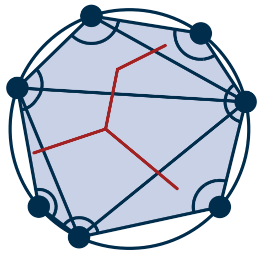
- Every polygon edge has one opposite angle
- Every internal edge has a small and large opposing angle
- Generalized Thales: angle is only dependent on the edge length
- The smaller angle is monotone in the internal edge length
- Polygon edges are irrelevant.
  Consider first entry where two length vectors differ



Weak dual is path

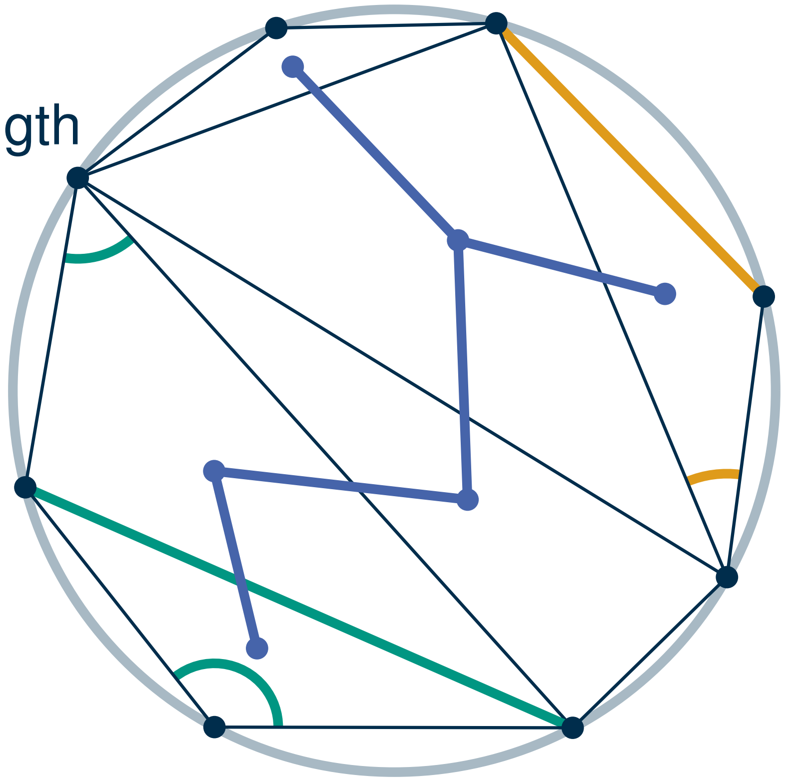
- Triangluation has $n-2$ triangles and $n-3$ cords $\Rightarrow$ Tree

Triangulation of Co-Circular Points



Vector of angles is optimized ⇔ Vector of lengths is optimized

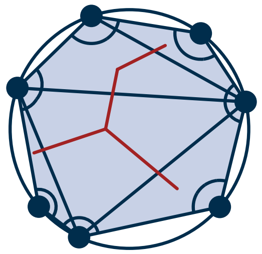
- Every polygon edge has one opposite angle
- Every internal edge has a small and large opposing angle
- Generalized Thales: angle is only dependent on the edge length
- The smaller angle is monotone in the internal edge length
- Polygon edges are irrelevant.
  Consider first entry where two length vectors differ



Weak dual is path

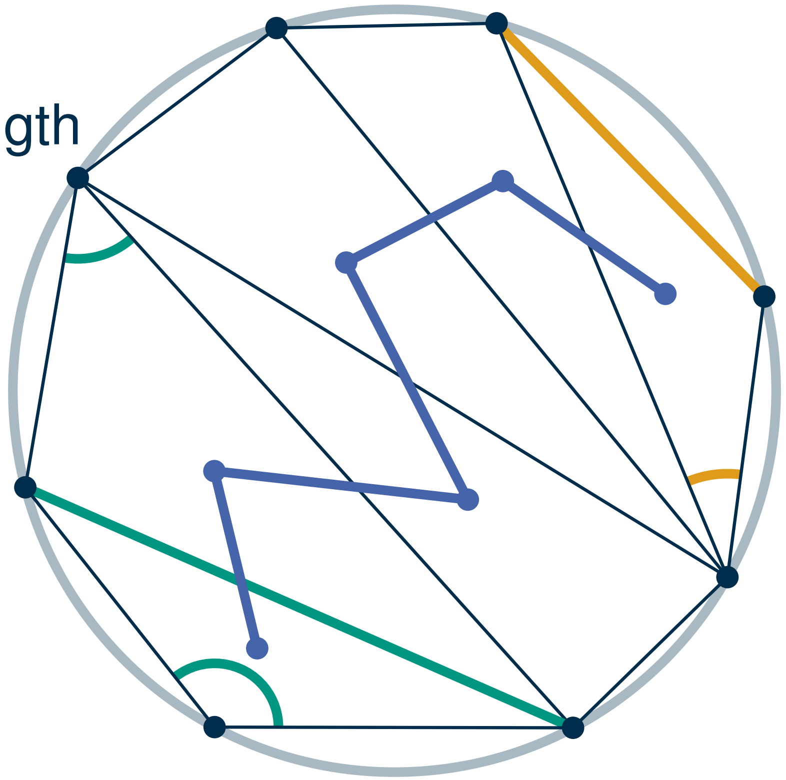
- Triangluation has $n-2$ triangles and $n-3$ cords $\Rightarrow$ Tree
- Vertex with degree 3 $\Rightarrow$ find larger internal edges

Triangulation of Co-Circular Points



Vector of angles is optimized ⇔ Vector of lengths is optimized

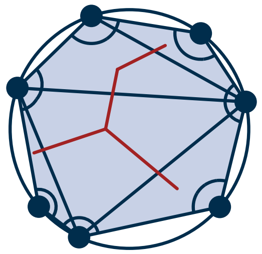
- Every polygon edge has one opposite angle
- Every internal edge has a small and large opposing angle
- Generalized Thales: angle is only dependent on the edge length
- The smaller angle is monotone in the internal edge length
- Polygon edges are irrelevant.
  Consider first entry where two length vectors differ



Weak dual is path

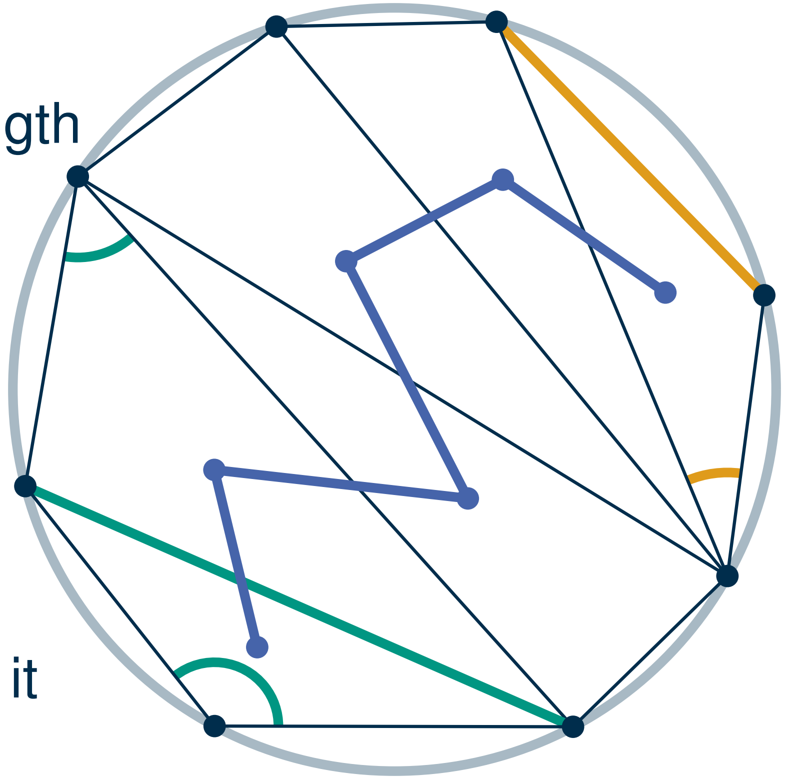
- Triangluation has $n-2$ triangles and $n-3$ cords $\Rightarrow$ Tree
- Vertex with degree 3 $\Rightarrow$ find larger internal edges

Triangulation of Co-Circular Points



Vector of angles is optimized ⇔ Vector of lengths is optimized

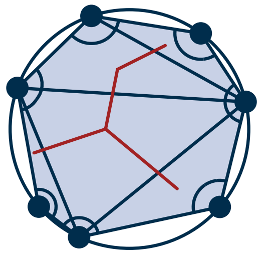
- Every polygon edge has one opposite angle
- Every internal edge has a small and large opposing angle
- Generalized Thales: angle is only dependent on the edge length
- The smaller angle is monotone in the internal edge length
- Polygon edges are irrelevant.
  Consider first entry where two length vectors differ



Weak dual is path

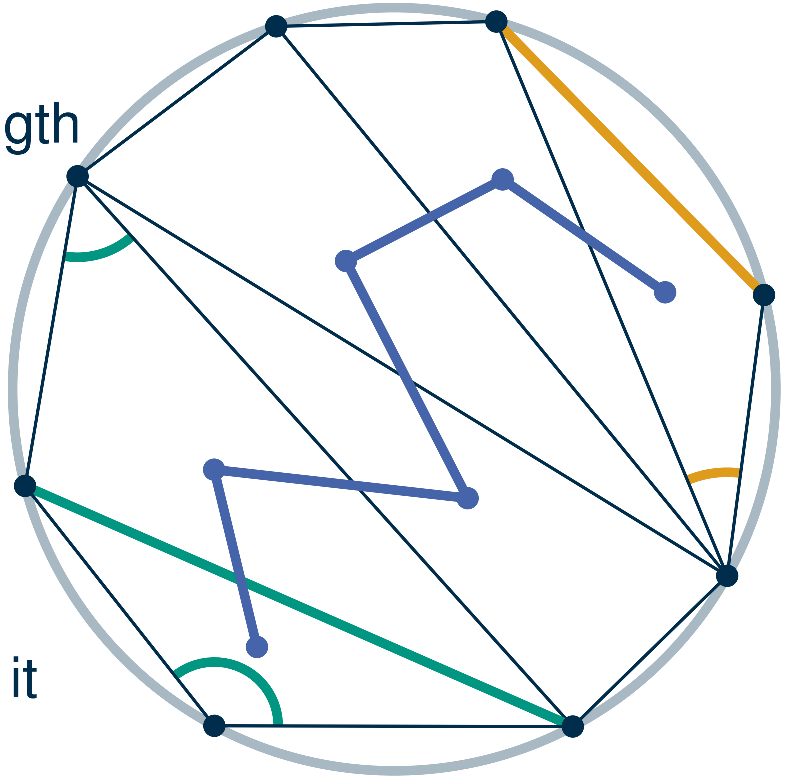
- Triangluation has $n-2$ triangles and $n-3$ cords $\Rightarrow$ Tree
- Vertex with degree 3 $\Rightarrow$ find larger internal edges
- If the starting points of the path are know, it is easy to extend it

Triangulation of Co-Circular Points



Vector of angles is optimized $\Leftrightarrow$ Vector of lengths is optimized

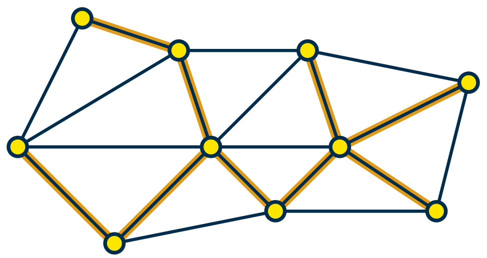
- Every polygon edge has one opposite angle
- Every internal edge has a small and large opposing angle
- Generalized Thales: angle is only dependent on the edge length
- The smaller angle is monotone in the internal edge length
- Polygon edges are irrelevant.
  Consider first entry where two length vectors differ



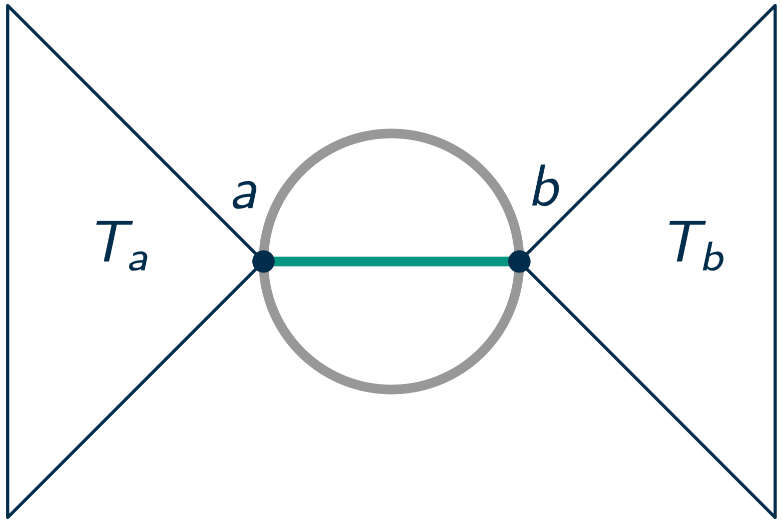
Weak dual is path

- Triangluation has $n - 2$ triangles and $n - 3$ cords $\Rightarrow$ Tree
- Vertex with degree 3 $\Rightarrow$ find larger internal edges
- If the starting points of the path are know, it is easy to extend it
- The starting points will be two out of the three largest ears
  - an *ear* is an internal edge, that touches two polygon edges

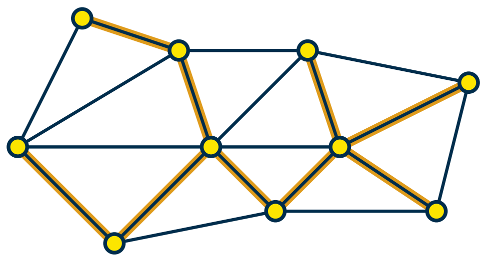
MST $\subseteq$ Delaunay



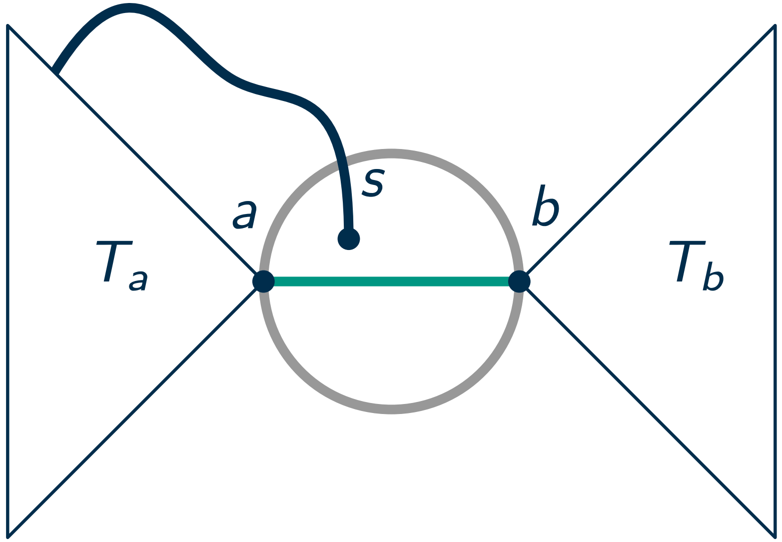
- Let ab be an edge of the MST, consider the smallest circle touching a and b



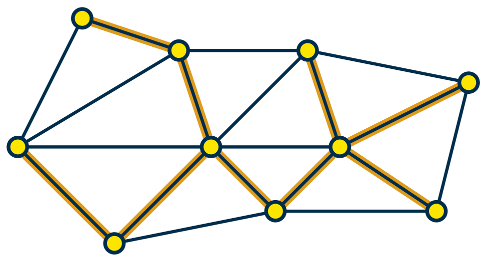
MST $\subseteq$ Delaunay



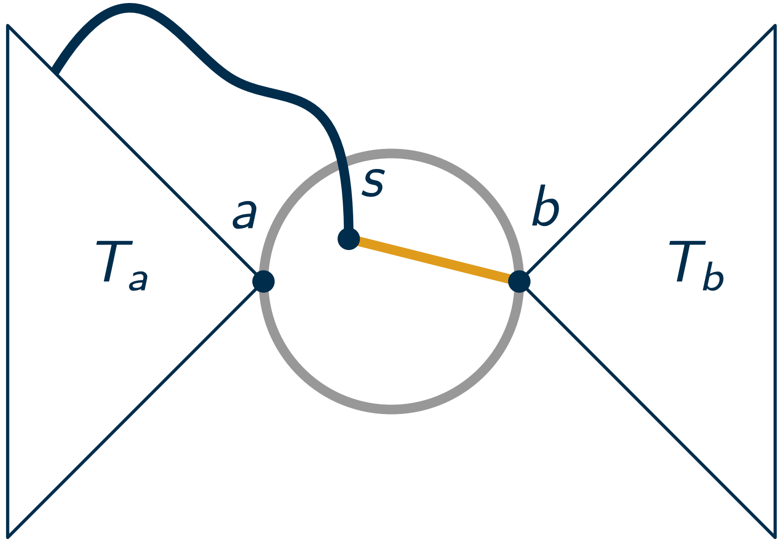
- Let ab be an edge of the MST, consider the smallest circle touching a and b
- The circle is empty, otherwise we can find a smaller MST
  - assume s is in the circle and $s \in T_a$



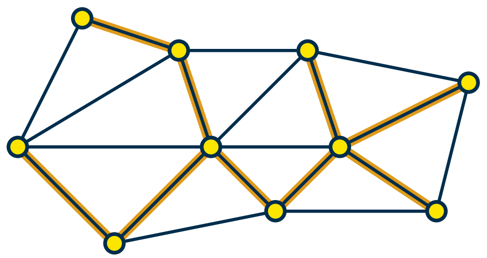
MST $\subseteq$ Delaunay



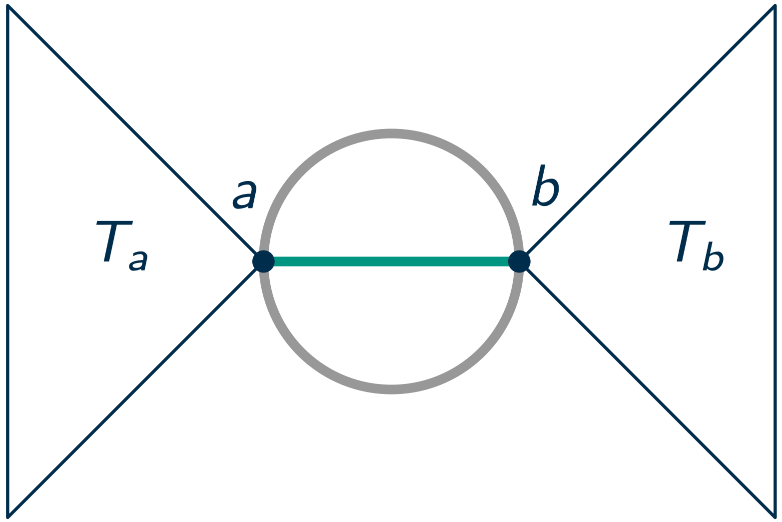
- Let ab be an edge of the MST, consider the smallest circle touching a and b
- The circle is empty, otherwise we can find a smaller MST
  - assume s is in the circle and $s \in T_a$
  - connect sb and remove ab



MST $\subseteq$ Delaunay



- Let ab be an edge of the MST, consider the smallest circle touching a and b
- The circle is empty, otherwise we can find a smaller MST
  - assume s is in the circle and $s \in T_a$
  - connect sb and remove ab
- Empty circle $\Rightarrow$ ab is in delauney triangulation
  - blow up circle until third point is hit



Your Submissions

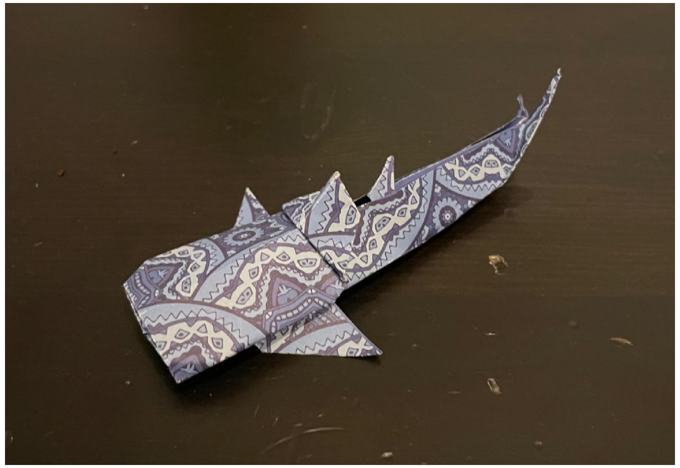
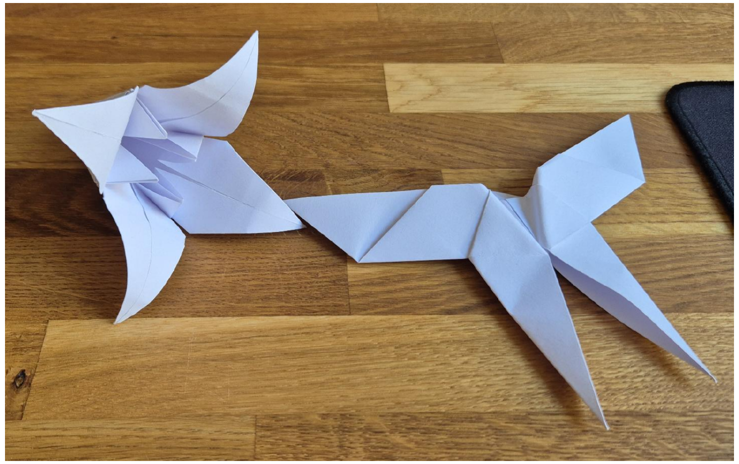
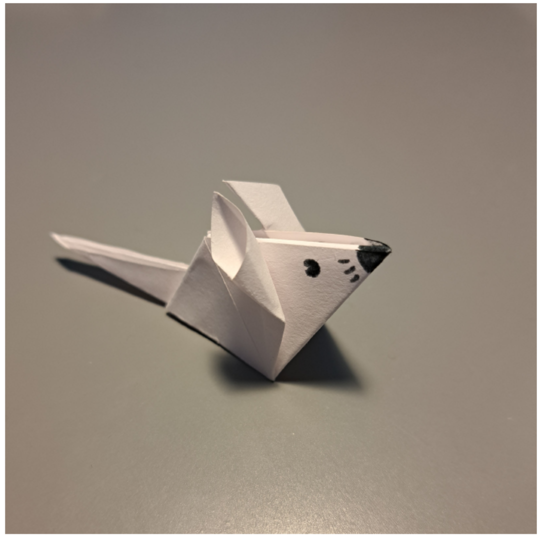


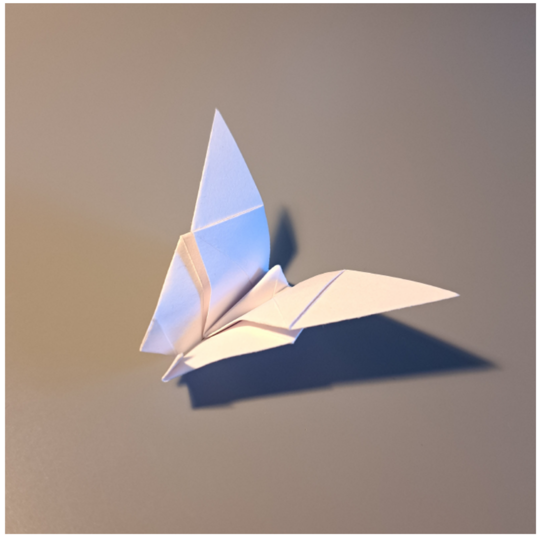
Figure 4: Even the paper pelican can hold things in its beak.

Your Submissions





(a) Mouse

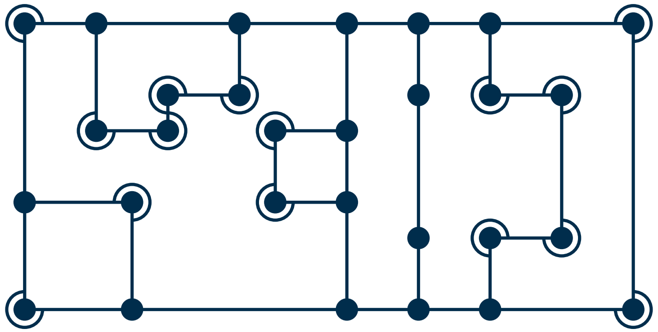


(b) Butterfly

Figure 3: Origami eines Otters

Assignment 6

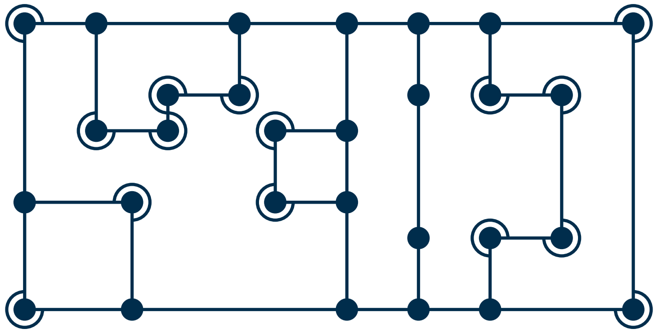
Bend and Behold



- Show that it is NP-hard to decide, whether a drawing exists with few $\frac{3}{2}\pi$ angles

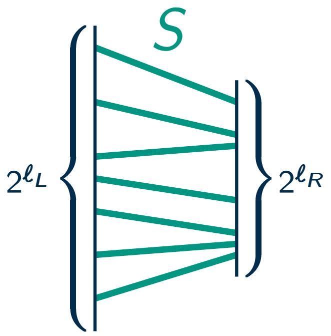
Assignment 6

Bend and Behold



- Show that it is NP-hard to decide, whether a drawing exists with few $\frac{3}{2}\pi$ angles

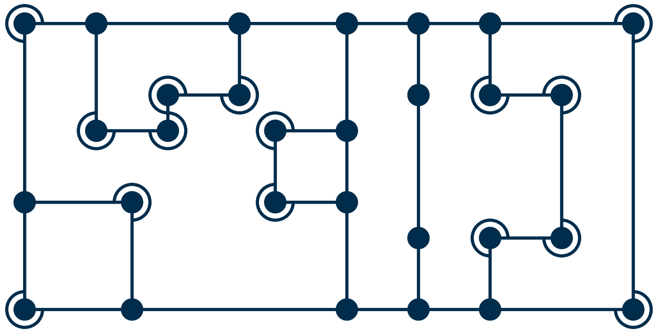
Nice (h, b)-Decomposition



- Find $B \subseteq S$ of size $\mathcal{O}(b)$
- Endpoints from elements in B are close or few segments inbetween
- *Discrete* version $\tilde{B}$ (enpoints are multiples of $2^{\ell_L - h}$)

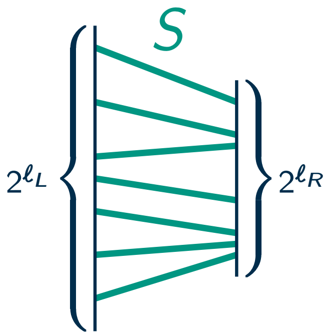
Assignment 6

Bend and Behold



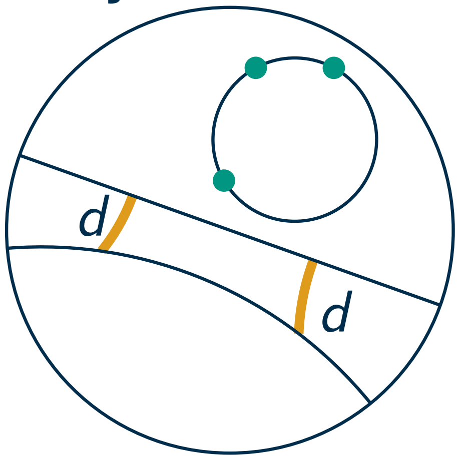
- Show that it is NP-hard to decide, whether a drawing exists with few $\frac{3}{2}\pi$ angles

Nice (h, b)-Decomposition



- Find $B \subseteq S$ of size $\mathcal{O}(b)$
- Endpoints from elements in B are close or few segments inbetween
- *Discrete* version $\tilde{B}$ (enpoints are multiples of $2^{\ell_L - h}$)

Geometry



- Draw line with distance d to other line
- How many circles lie on 3 points?
- For points A, B, C there is no point D in ABC^+ with same distance to A and B

Graph Embeddings

Many graphs have some form of hidden geometry

- Embedding: Input is a Graph; output is a position for every vertex
- Finding this geometry, is useful for various scenarios

Graph Embeddings

Many graphs have some form of hidden geometry

- Embedding: Input is a Graph; output is a position for every vertex
- Finding this geometry, is useful for various scenarios
  - We can draw the graph

Road network

Graph Embeddings

Many graphs have some form of hidden geometry

- Embedding: Input is a Graph; output is a position for every vertex
- Finding this geometry, is useful for various scenarios
  - We can draw the graph

Road network **obvious geometry**

Graph Embeddings

Many graphs have some form of hidden geometry

- Embedding: Input is a Graph; output is a position for every vertex
- Finding this geometry, is useful for various scenarios
  - We can draw the graph — Road network — **obvious geometry**
  - We can try to predict future edges — Social network — **hidden geometry**

Graph Embeddings

Many graphs have some form of hidden geometry

- Embedding: Input is a Graph; output is a position for every vertex
- Finding this geometry, is useful for various scenarios
  - We can draw the graph — Road network — **obvious geometry**
  - We can try to predict future edges — Social network — **hidden geometry**
  - Classify nodes by clustering them — Bots in social interaction graph

Graph Embeddings

Many graphs have some form of hidden geometry

- Embedding: Input is a Graph; output is a position for every vertex
- Finding this geometry, is useful for various scenarios
  - We can draw the graph — Road network — **obvious geometry**
  - We can try to predict future edges — Social network — **hidden geometry**
  - Classify nodes by clustering them — Bots in social interaction graph
  - Greedy Routing — Internet graph

Graph Embeddings

Many graphs have some form of hidden geometry

- Embedding: Input is a Graph; output is a position for every vertex
- Finding this geometry, is useful for various scenarios
  - We can draw the graph — Road network — **obvious geometry**
  - We can try to predict future edges — Social network — **hidden geometry**
  - Classify nodes by clustering them — Bots in social interaction graph
  - Greedy Routing — Internet graph
  - Natural Language processing — Word graph

Graph Embeddings

Many graphs have some form of hidden geometry

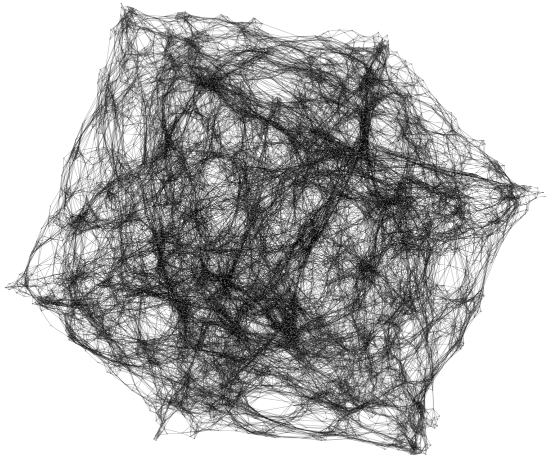
- Embedding: Input is a Graph; output is a position for every vertex
- Finding this geometry, is useful for various scenarios
  - We can draw the graph — Road network — **obvious geometry**
  - We can try to predict future edges — Social network — **hidden geometry**
  - Classify nodes by clustering them — Bots in social interaction graph
  - Greedy Routing — Internet graph
  - Natural Language processing — Word graph

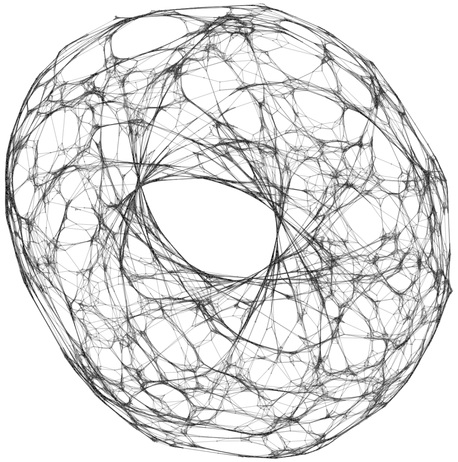
Many important questions

- What is the right embedding space/dimension?
- How can we measure the quality of an embedding?
- What problems can our embedding solve?

Some Examples

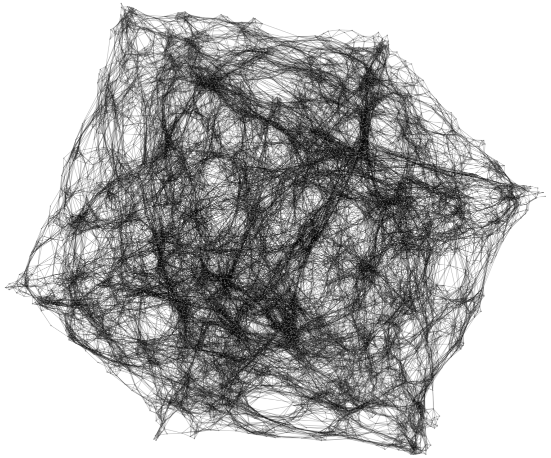
Generated Graphs

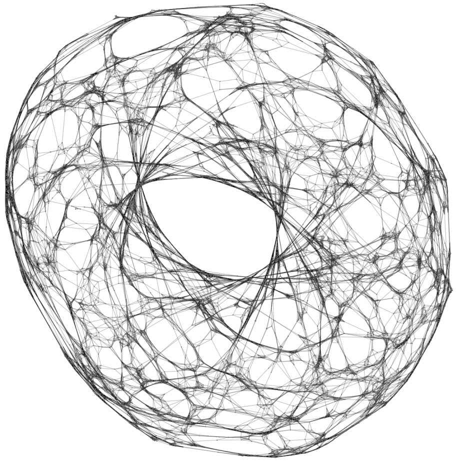




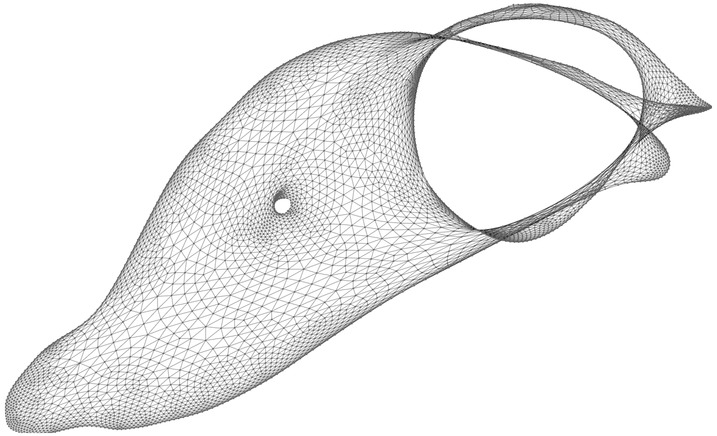
Some Examples

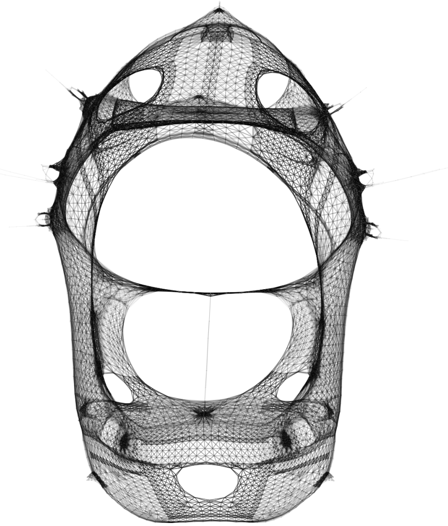
Generated Graphs





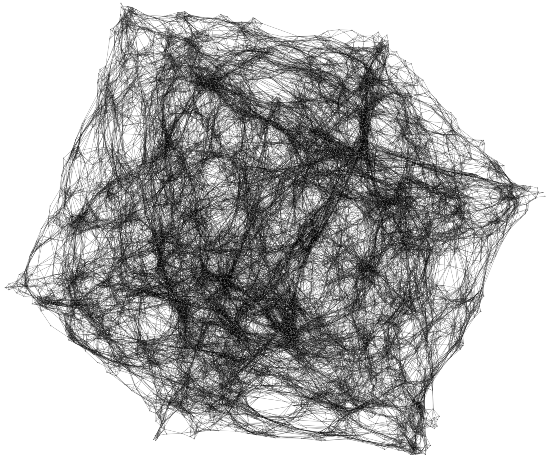
Real World Graphs

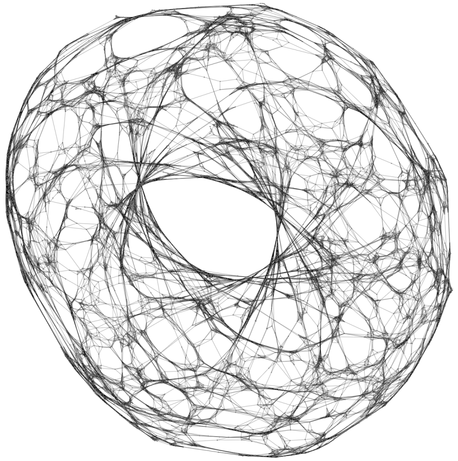


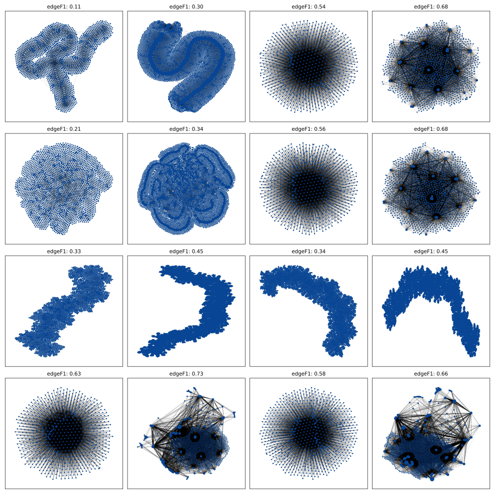


Some Examples

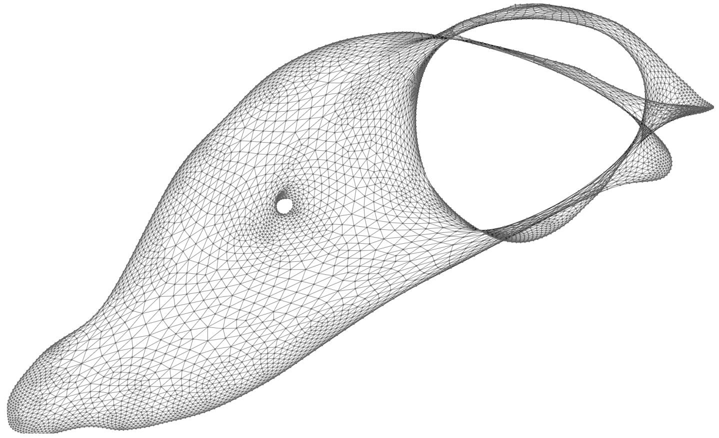
Generated Graphs

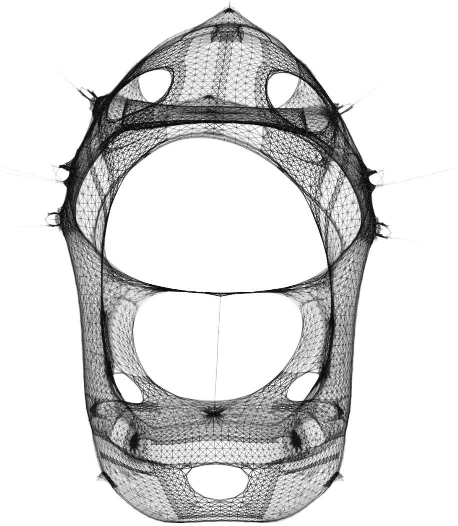




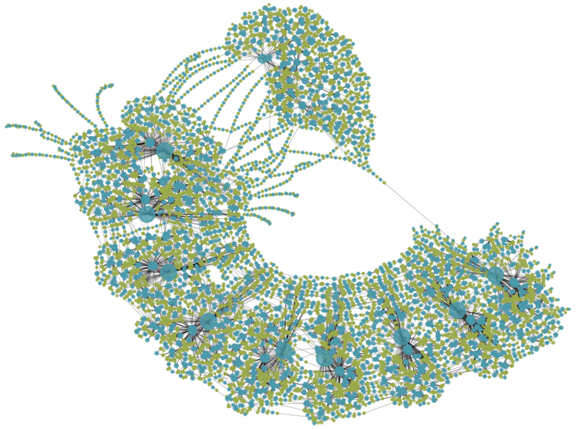


Real World Graphs



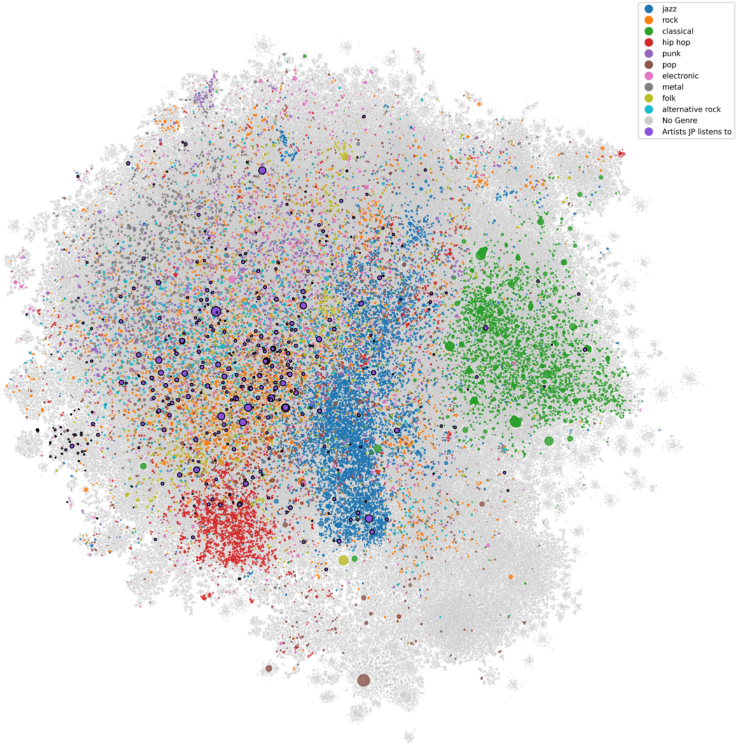


SAT-Instances



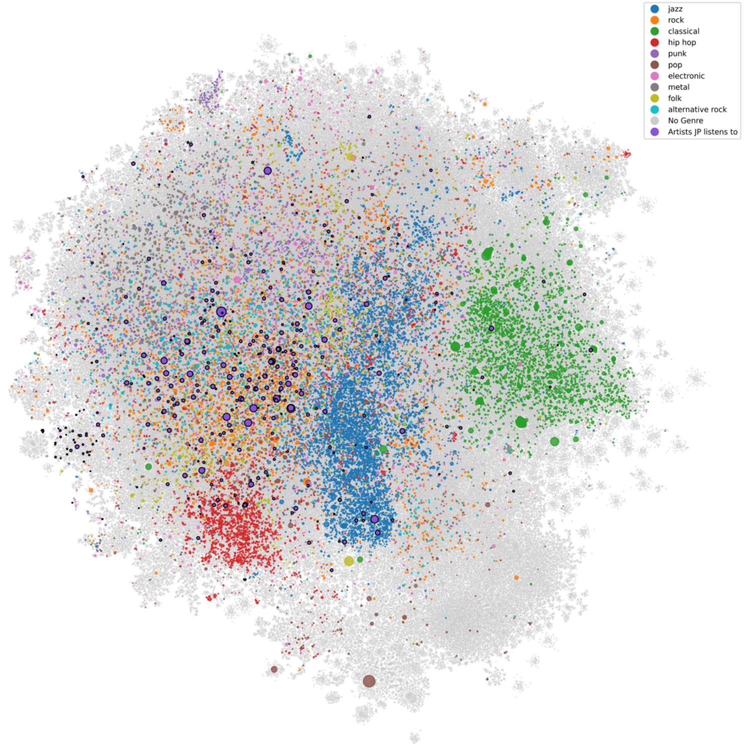
Some Examples

Music Artists

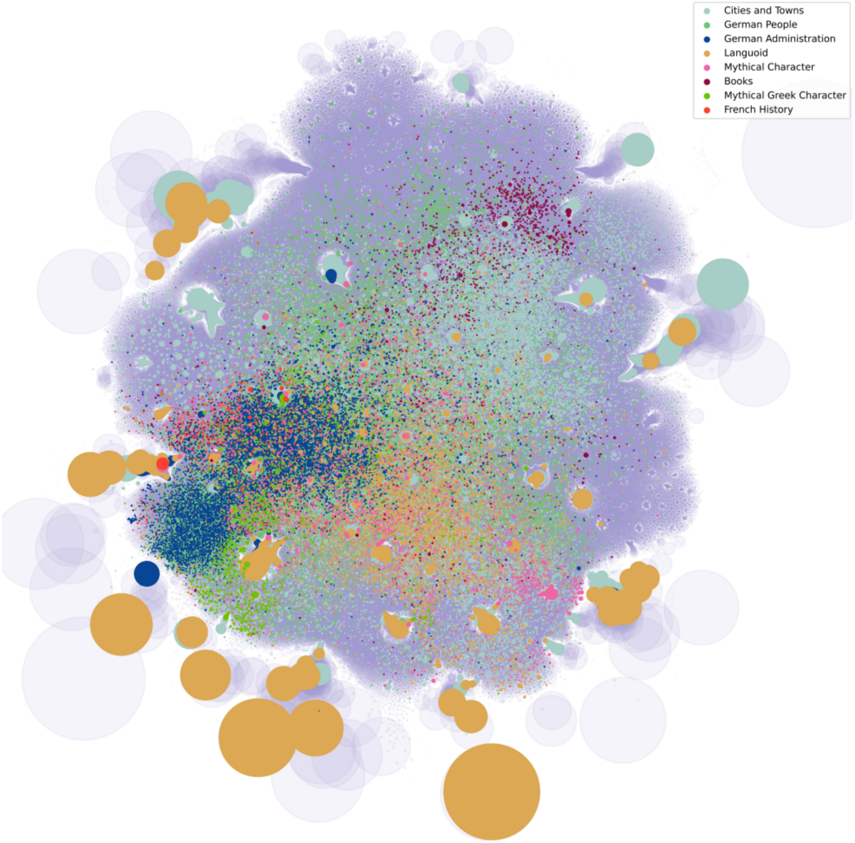


Some Examples

Music Artists

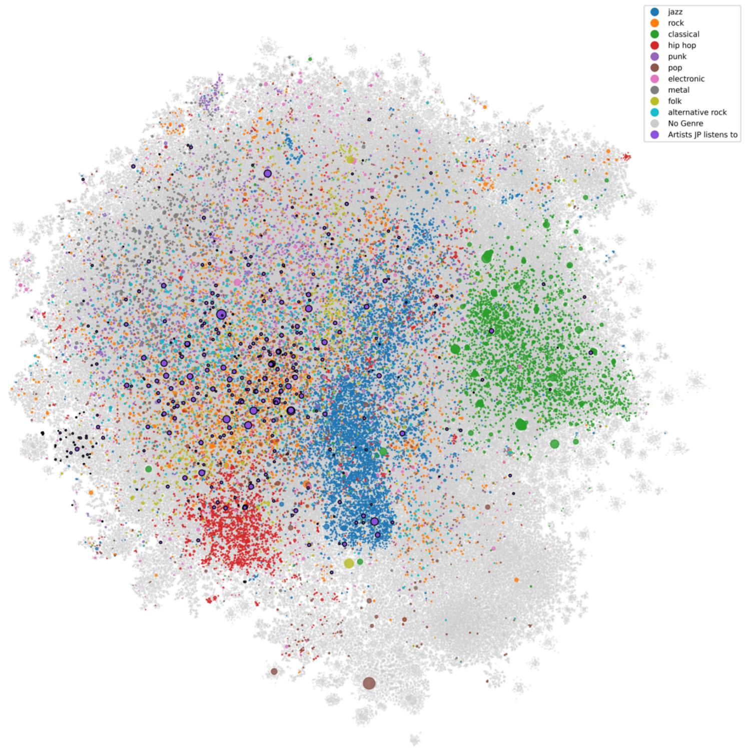


Wikipedia

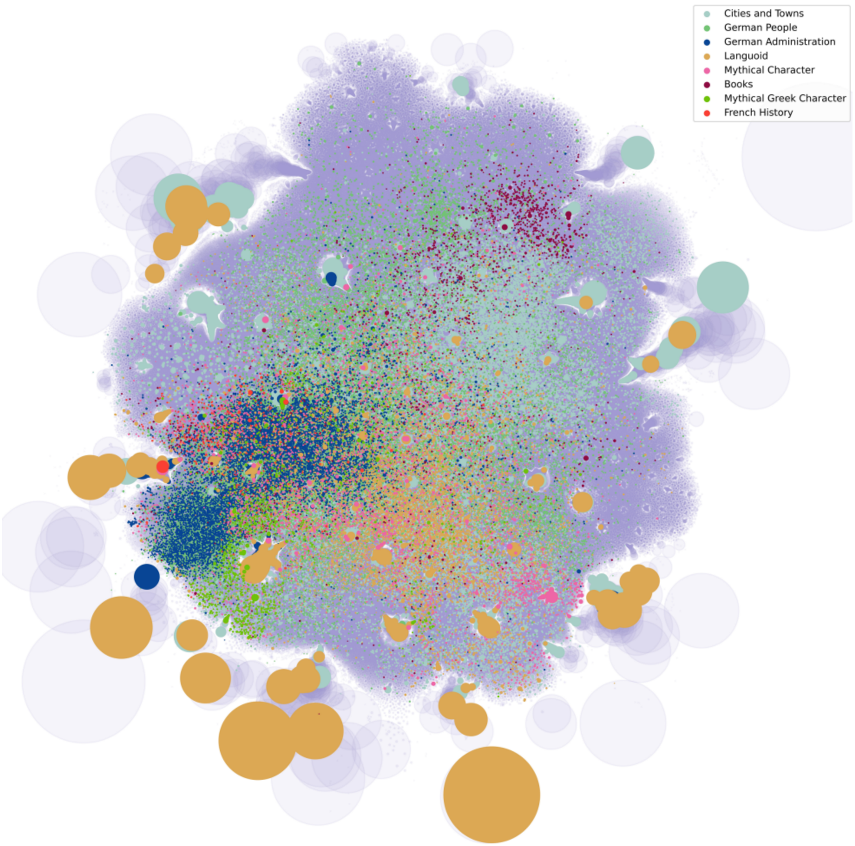


Some Examples

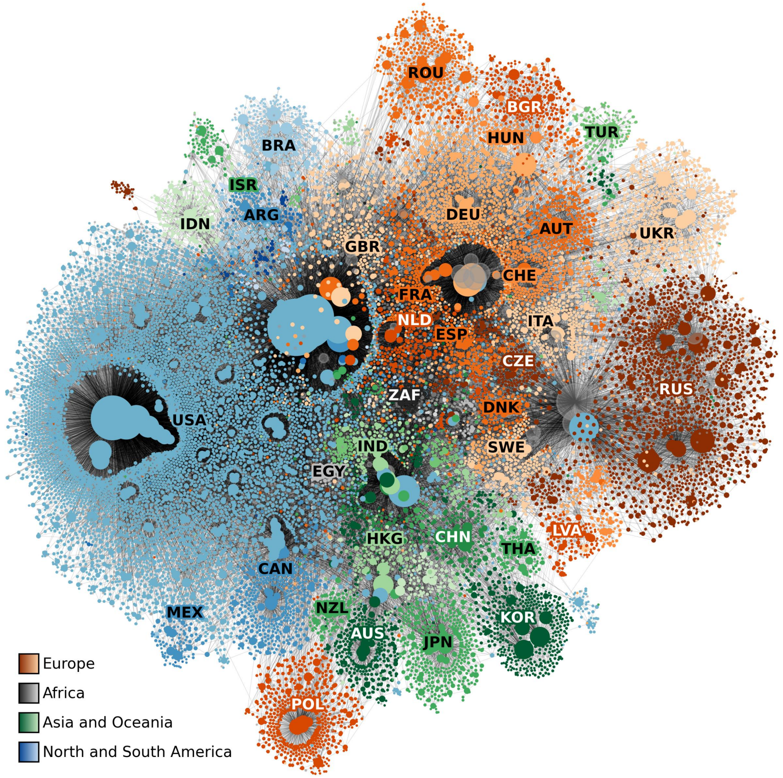
Music Artists



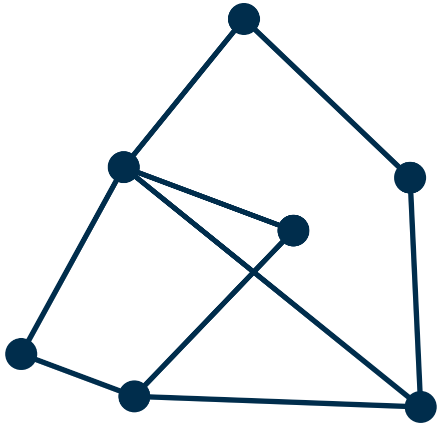
Wikipedia



Internet Graph



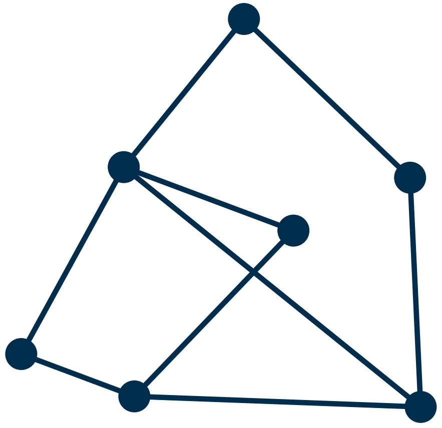
Greedy Routing



Greedy Routing

- **Given**: $2d$ drawing of G, **Goal**: move from s to t

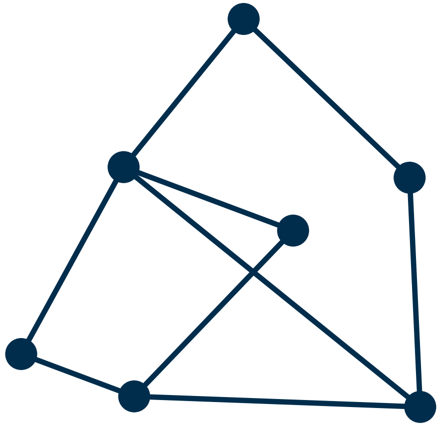
Greedy Routing



Greedy Routing

- **Given**: $2d$ drawing of G, **Goal**: move from s to t
- Strategy: at every step, select neighbour that minimizes the Euclidean distance to t the most
- successful, if we eventually reacht t
- unsuccessful, if we get stuck in a dead end

Greedy Routing



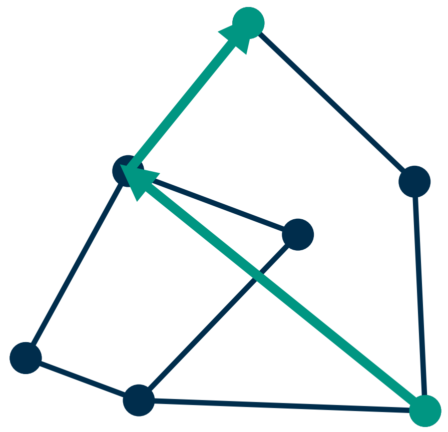
Greedy Routing

- **Given**: $2d$ drawing of G, **Goal**: move from s to t
- Strategy: at every step, select neighbour that minimizes the Euclidean distance to t the most
- successful, if we eventually reacht t
- unsuccessful, if we get stuck in a dead end

Greedy Embedding

- $2d$ drawing of a graph
- for every pair of vertices, greedy routing is successful

Greedy Routing



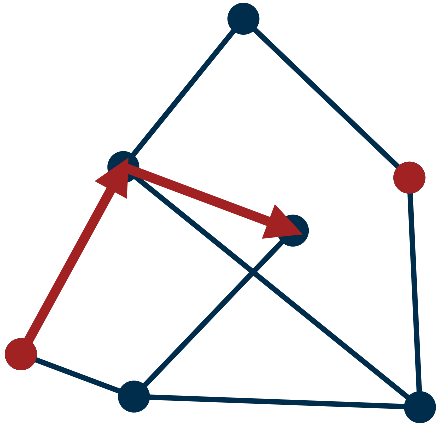
Greedy Routing

- **Given**: $2d$ drawing of G, **Goal**: move from s to t
- Strategy: at every step, select neighbour that minimizes the Euclidean distance to t the most
- successful, if we eventually reacht t
- unsuccessful, if we get stuck in a dead end

Greedy Embedding

- $2d$ drawing of a graph
- for every pair of vertices, greedy routing is successful

Greedy Routing



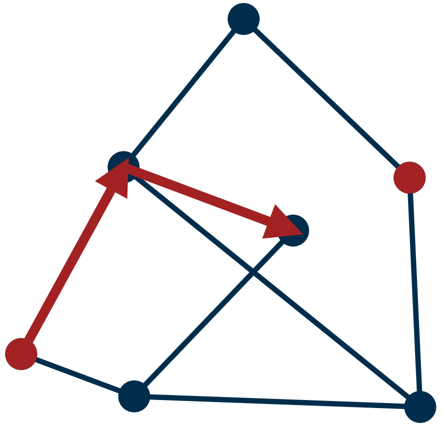
Greedy Routing

- **Given**: $2d$ drawing of G, **Goal**: move from s to t
- Strategy: at every step, select neighbour that minimizes the Euclidean distance to t the most
- successful, if we eventually reacht t
- unsuccessful, if we get stuck in a dead end

Greedy Embedding

- $2d$ drawing of a graph
- for every pair of vertices, greedy routing is successful

Greedy Routing



Greedy Routing

- **Given**: $2d$ drawing of G, **Goal**: move from s to t
- Strategy: at every step, select neighbour that minimizes the Euclidean distance to t the most
- successful, if we eventually reacht t
- unsuccessful, if we get stuck in a dead end

Greedy Embedding

- $2d$ drawing of a graph
- for every pair of vertices, greedy routing is successful

Does every graph have a greedy embedding in the Euclidean plane?

How about the hyperbolic plane?

Can you find counterexamples?

Greedy Routing

A greedy embedding is permissable if and only if for every $t \neq s$, there is a neighbor of s that is closer to t then s.

Greedy Routing

A greedy embedding is permissable if and only if for every $t \neq s$, there is a neighbor of s that is closer to t then s.

An embedding of a tree is permissible if and only if the perpendicular bisector of every edge separates the tree into two components.

Greedy Routing

A greedy embedding is permissable if and only if for every $t \neq s$, there is a neighbor of s that is closer to t then s.

An embedding of a tree is permissible if and only if the perpendicular bisector of every edge separates the tree into two components.

A 7 star does not have a $2d$-Euclidean greedy embedding

Greedy Routing

A greedy embedding is permissable if and only if for every $t \neq s$, there is a neighbor of s that is closer to t then s.

An embedding of a tree is permissible if and only if the perpendicular bisector of every edge separates the tree into two components.

A 7 star does not have a $2d$-Euclidean greedy embedding

Every tree has a $2d$-hyperbolic greedy embedding

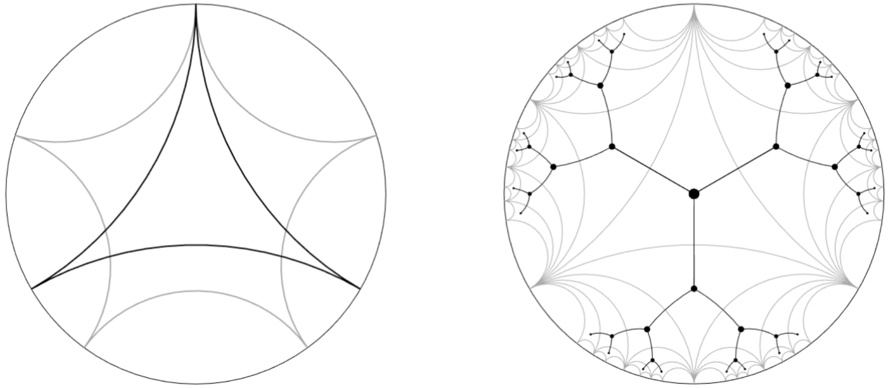
Greedy Routing

A greedy embedding is permissable if and only if for every $t \neq s$, there is a neighbor of s that is closer to t then s.

An embedding of a tree is permissible if and only if the perpendicular bisector of every edge separates the tree into two components.

A 7 star does not have a $2d$-Euclidean greedy embedding

Every tree has a $2d$-hyperbolic greedy embedding



Hyperbolic Embeddings for Near-Optimal Greedy Routing

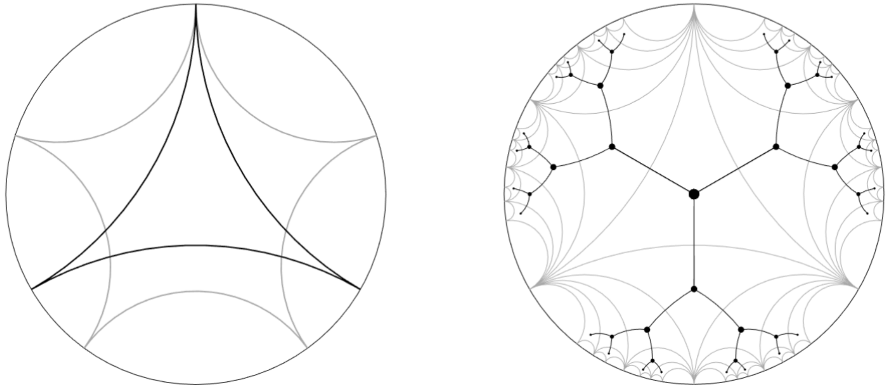
Greedy Routing

A greedy embedding is permissable if and only if for every $t \neq s$, there is a neighbor of s that is closer to t then s.

An embedding of a tree is permissible if and only if the perpendicular bisector of every edge separates the tree into two components.

A 7 star does not have a $2d$-Euclidean greedy embedding

Every tree has a $2d$-hyperbolic greedy embedding



Hyperbolic Embeddings for Near-Optimal Greedy Routing

Is every graph embeddable in hyperbolic space?