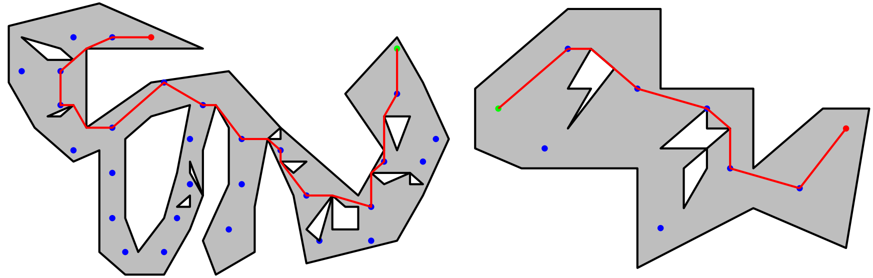
Karlsruher Institut für Technologie

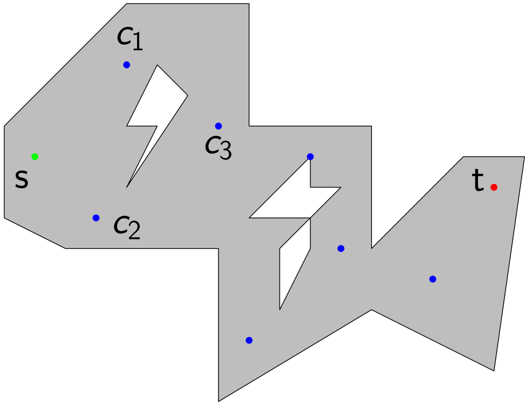
Save the Robot – Computational Geometry

February 3rd, 2016

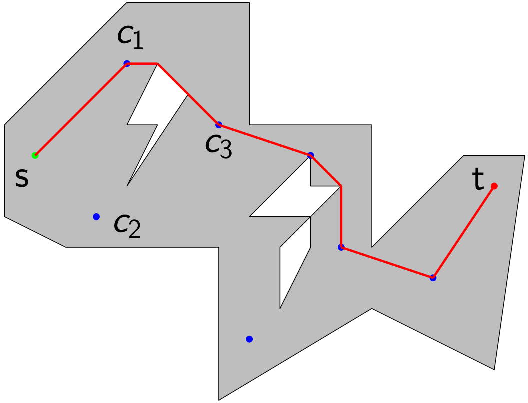
Niklas Baumstark, Samuel Groß

Institute of Theoretical Informatics





- Electric robot is in a warehouse polygon P, starting at green location s
- Range of r, recharge at blue charging stations $c \in C$
- What is the shortest path to red exit location t



- Electric robot is in a warehouse polygon P, starting at green location s
- Range of r, recharge at blue charging stations $c \in C$
- What is the shortest path to red exit location t

Overview

Computing visibility graphs

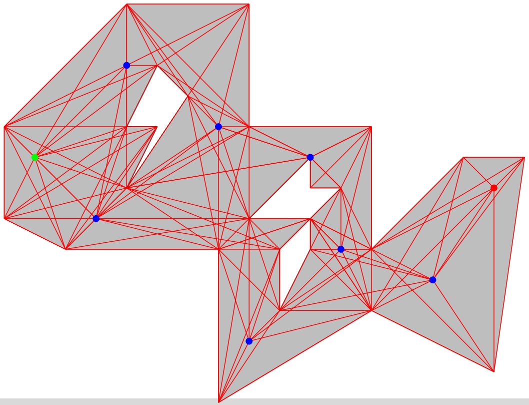
Computing shortest route

Possible optimizations

Visibility Graph

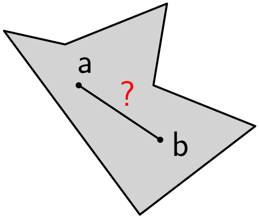
Given a set of points S inside P, the **visibility graph** of S is the undirected graph $G_S = (S, E)$ with edges $E = \{\{a, b\} \mid \overline{ab} \subseteq P\}$

We are interested in the point set $S = P \cup C \cup \{s, t\}$



Computing G_S – First Attempt

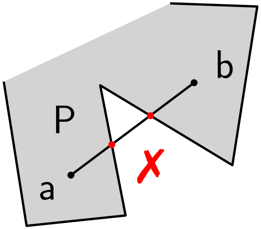
- For each pair $a, b \in S$, check if $\overline{ab}$ is inside P
- $\Theta(|S|^2)$ segment-in-polygon tests
- How to check if segment ab is inside P?



Computing G_S – First Attempt

- For each pair $a, b \in S$, check if $\overline{ab}$ is inside P
- $\Theta(|S|^2)$ segment-in-polygon tests
- How to check if segment ab is inside P?

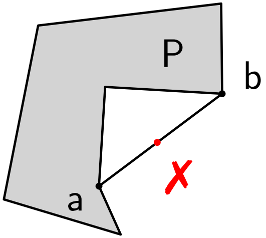
1. Is there an edge $\overline{pq}$ of P that intersects $\overline{ab}$ in on the interior? $\rightarrow$ output **NO**



Computing G_S – First Attempt

- For each pair $a, b \in S$, check if $\overline{ab}$ is inside P
- $\Theta(|S|^2)$ segment-in-polygon tests
- How to check if segment ab is inside P?

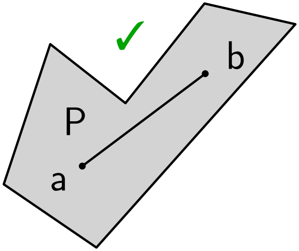
1. Is there an edge $\overline{pq}$ of P that intersects $\overline{ab}$ in on the interior?
   $\rightarrow$ output **NO**
2. Is the midpoint $(a + b)/2$ outside of P?
   $\rightarrow$ output **NO**



Computing G_S – First Attempt

- For each pair $a, b \in S$, check if $\overline{ab}$ is inside P
- $\Theta(|S|^2)$ segment-in-polygon tests
- How to check if segment ab is inside P?

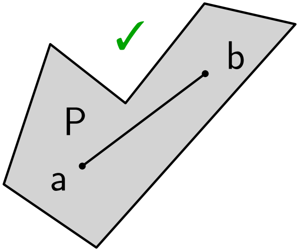
1. Is there an edge $\overline{pq}$ of P that intersects $\overline{ab}$ in on the interior? $\rightarrow$ output **NO**
2. Is the midpoint $(a+b)/2$ outside of P? $\rightarrow$ output **NO**
3. Otherwise, output **YES**



Computing G_S – First Attempt

- For each pair $a, b \in S$, check if $\overline{ab}$ is inside P
- $\Theta(|S|^2)$ segment-in-polygon tests
- How to check if segment ab is inside P?

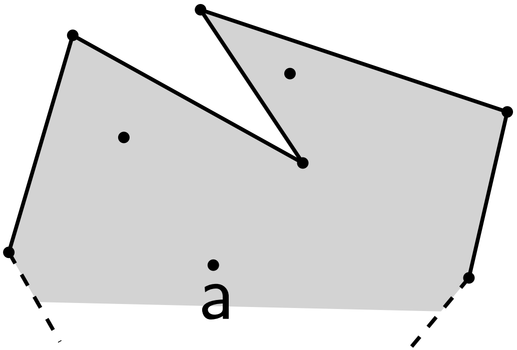
1. Is there an edge $\overline{pq}$ of P that intersects $\overline{ab}$ in on the interior?
   $\rightarrow$ output **NO**
2. Is the midpoint $(a + b)/2$ outside of P?
   $\rightarrow$ output **NO**
3. Otherwise, output **YES**



Runtime: $\Theta(|S|^2 \cdot |P|)$, dominates runtime of the algorithm

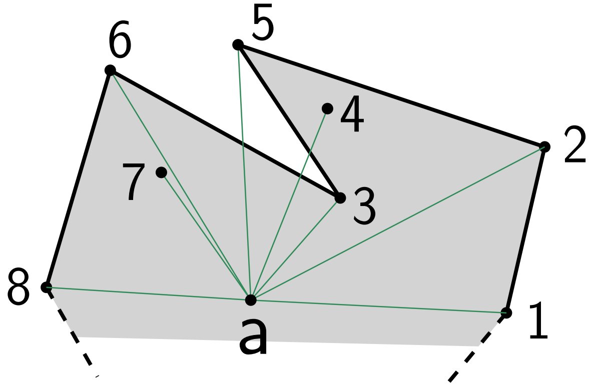
Computing G_S – Second Attempt

- For each $a \in S$, compute points b with $\overline{ab}$ inside P in one pass
- Use **Circle-Sweep**: Sort all points/vertices by their angle around a
- Maintain edges of P intersecting the sweep line, sorted by distance



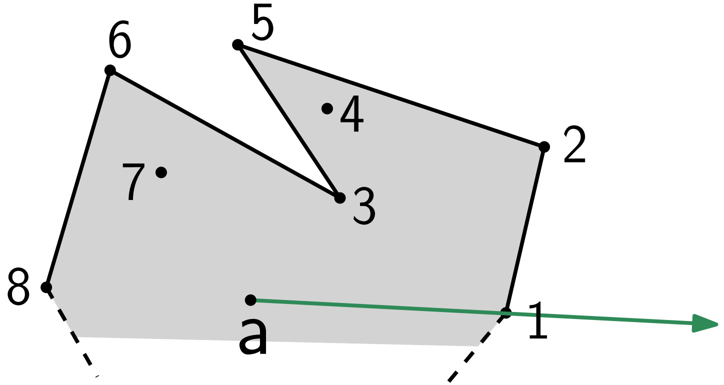
Computing G_S – Second Attempt

- For each $a \in S$, compute points b with $\overline{ab}$ inside P in one pass
- Use **Circle-Sweep**: Sort all points/vertices by their angle around a
- Maintain edges of P intersecting the sweep line, sorted by distance



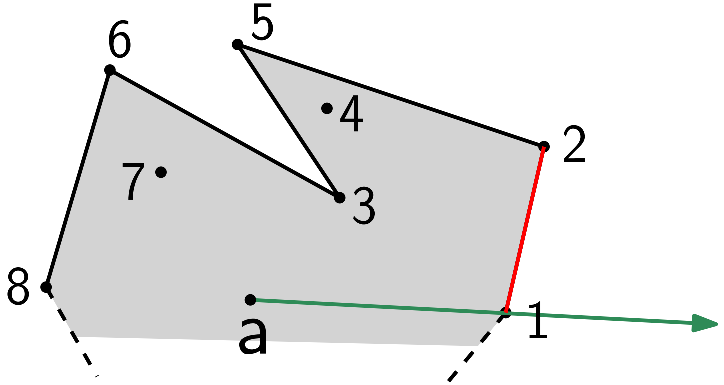
Computing G_S – Second Attempt

- For each $a \in S$, compute points b with $\overline{ab}$ inside P in one pass
- Use **Circle-Sweep**: Sort all points/vertices by their angle around a
- Maintain edges of P intersecting the sweep line, sorted by distance



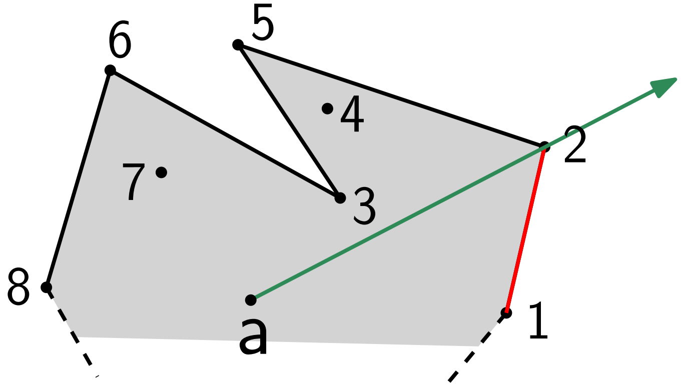
Computing G_S – Second Attempt

- For each $a \in S$, compute points b with $\overline{ab}$ inside P in one pass
- Use **Circle-Sweep**: Sort all points/vertices by their angle around a
- Maintain edges of P intersecting the sweep line, sorted by distance



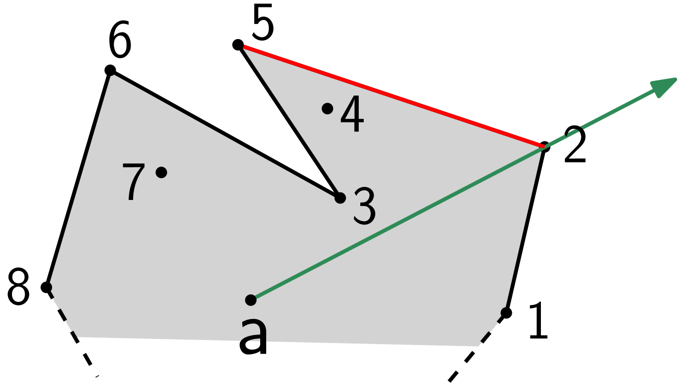
Computing G_S – Second Attempt

- For each $a \in S$, compute points b with $\overline{ab}$ inside P in one pass
- Use **Circle-Sweep**: Sort all points/vertices by their angle around a
- Maintain edges of P intersecting the sweep line, sorted by distance



Computing G_S – Second Attempt

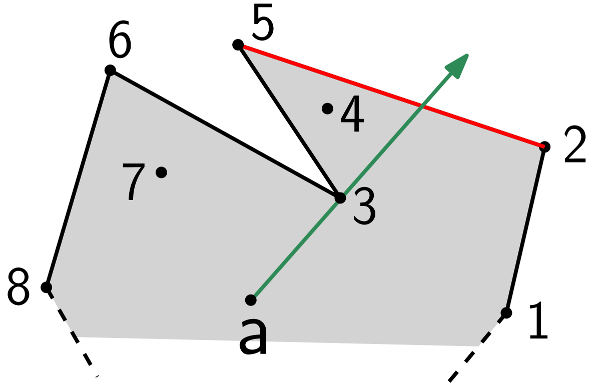
- For each $a \in S$, compute points b with $\overline{ab}$ inside P in one pass
- Use **Circle-Sweep**: Sort all points/vertices by their angle around a
- Maintain edges of P intersecting the sweep line, sorted by distance



Prof. Dr. Dorothea Wagner
Institute of Theoretical Informatics

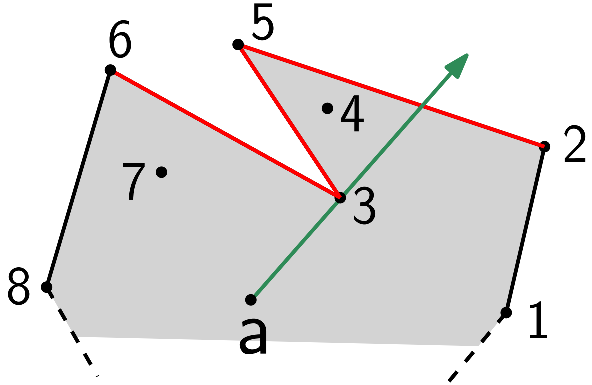
Computing G_S – Second Attempt

- For each $a \in S$, compute points b with $\overline{ab}$ inside P in one pass
- Use **Circle-Sweep**: Sort all points/vertices by their angle around a
- Maintain edges of P intersecting the sweep line, sorted by distance



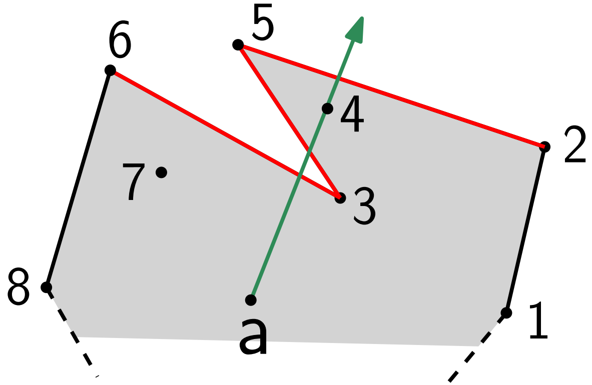
Computing G_S – Second Attempt

- For each $a \in S$, compute points b with $\overline{ab}$ inside P in one pass
- Use **Circle-Sweep**: Sort all points/vertices by their angle around a
- Maintain edges of P intersecting the sweep line, sorted by distance



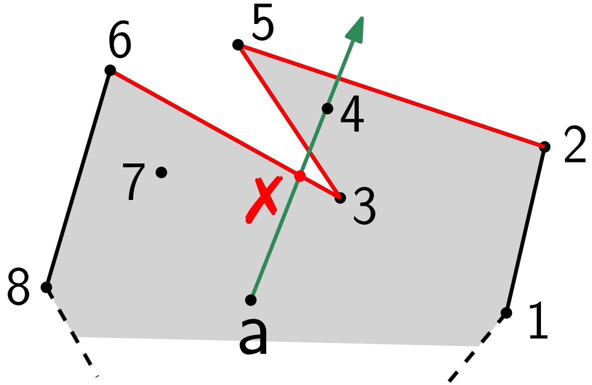
Computing G_S – Second Attempt

- For each $a \in S$, compute points b with $\overline{ab}$ inside P in one pass
- Use **Circle-Sweep**: Sort all points/vertices by their angle around a
- Maintain edges of P intersecting the sweep line, sorted by distance



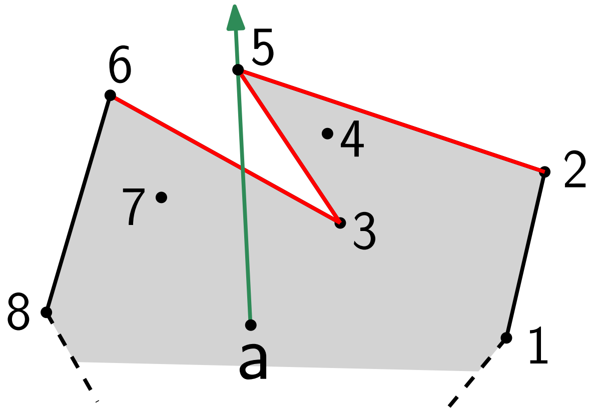
Computing G_S – Second Attempt

- For each $a \in S$, compute points b with $\overline{ab}$ inside P in one pass
- Use **Circle-Sweep**: Sort all points/vertices by their angle around a
- Maintain edges of P intersecting the sweep line, sorted by distance



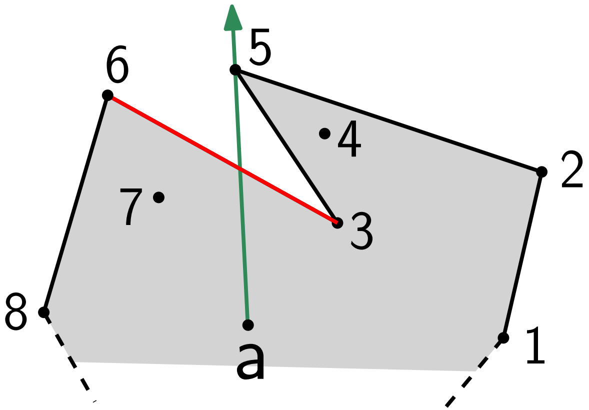
Computing G_S – Second Attempt

- For each $a \in S$, compute points b with $\overline{ab}$ inside P in one pass
- Use **Circle-Sweep**: Sort all points/vertices by their angle around a
- Maintain edges of P intersecting the sweep line, sorted by distance



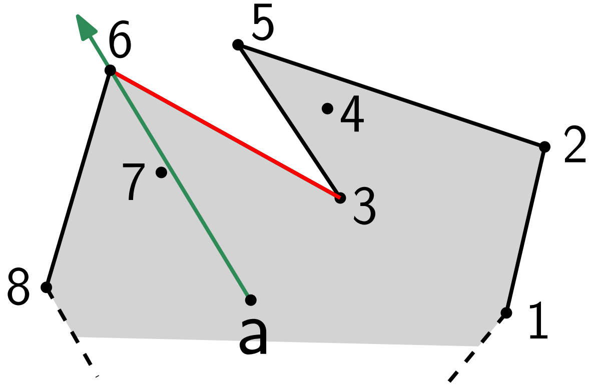
Computing G_S – Second Attempt

- For each $a \in S$, compute points b with $\overline{ab}$ inside P in one pass
- Use **Circle-Sweep**: Sort all points/vertices by their angle around a
- Maintain edges of P intersecting the sweep line, sorted by distance



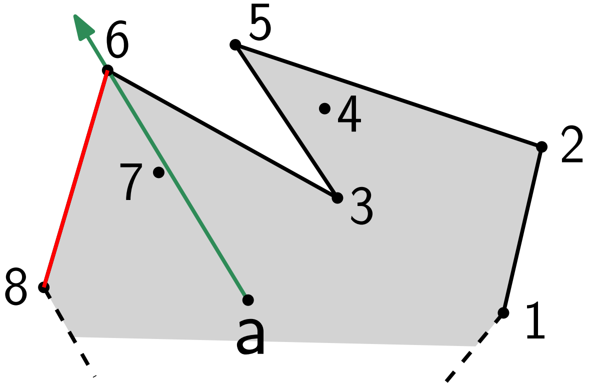
Computing G_S – Second Attempt

- For each $a \in S$, compute points b with $\overline{ab}$ inside P in one pass
- Use **Circle-Sweep**: Sort all points/vertices by their angle around a
- Maintain edges of P intersecting the sweep line, sorted by distance



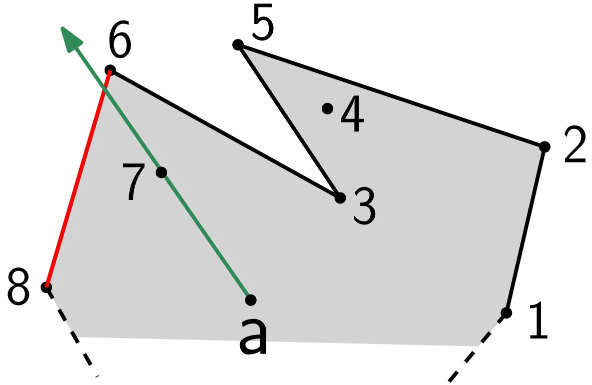
Computing G_S – Second Attempt

- For each $a \in S$, compute points b with $\overline{ab}$ inside P in one pass
- Use **Circle-Sweep**: Sort all points/vertices by their angle around a
- Maintain edges of P intersecting the sweep line, sorted by distance



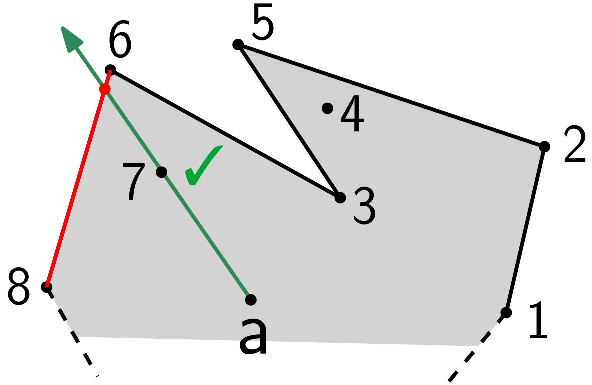
Computing G_S – Second Attempt

- For each $a \in S$, compute points b with $\overline{ab}$ inside P in one pass
- Use **Circle-Sweep**: Sort all points/vertices by their angle around a
- Maintain edges of P intersecting the sweep line, sorted by distance



Computing G_S – Second Attempt

- For each $a \in S$, compute points b with $\overline{ab}$ inside P in one pass
- Use **Circle-Sweep**: Sort all points/vertices by their angle around a
- Maintain edges of P intersecting the sweep line, sorted by distance



Computing G_S – Second Attempt

- For each $a \in S$, compute points b with $\overline{ab}$ inside P in one pass
- Use **Circle-Sweep**: Sort all points/vertices by their angle around a
- Maintain edges of P intersecting the sweep line, sorted by distance
- Runtime: $\Theta(|S| \cdot n \cdot \log n)$ where $n = |S| + |P|$
- Now that we have G_S, how to solve the problem on a general graph?

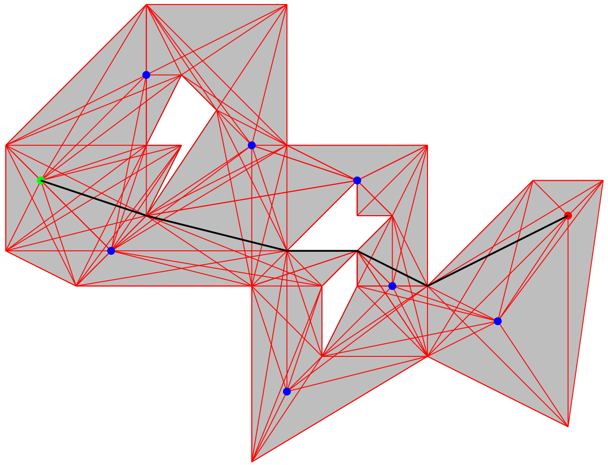
Status

~~Computing visibility graphs~~

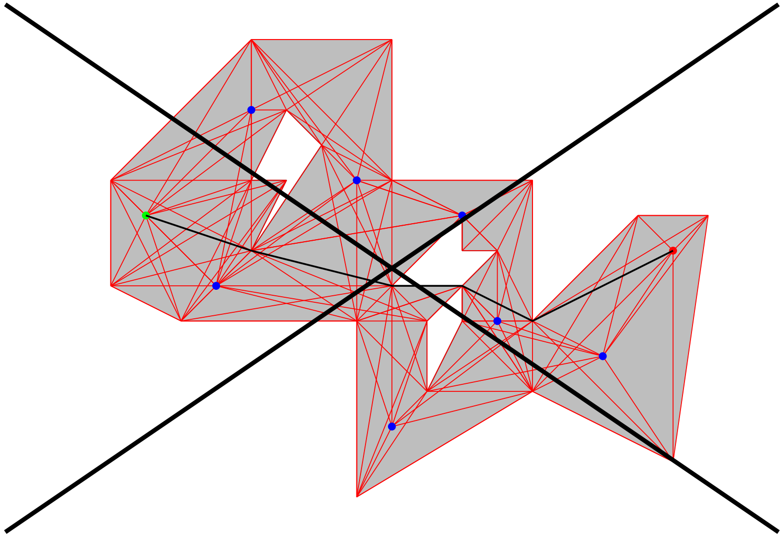
Computing shortest route

Possible optimizations

Computing shortest route



Computing shortest route



Computing shortest route

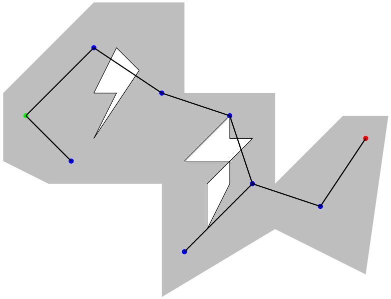
- Problem: How to guarantee that path includes charging stations?

Computing shortest route

- Problem: How to guarantee that path includes charging stations?
- Idea: Perform search on a reachability graph:
  - Edges only between charging stations (plus start/end node)
  - There's an edge between two nodes only if there's a path between them in the visibility graph that has a length of at most r

Computing shortest route

- Problem: How to guarantee that path includes charging stations?
- Idea: Perform search on a reachability graph:
  - Edges only between charging stations (plus start/end node)
  - There's an edge between two nodes only if there's a path between them in the visibility graph that has a length of at most r



Status

~~Computing visibility graphs~~

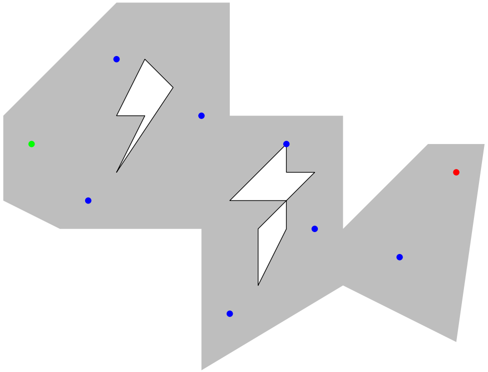
Computing shortest route

- Computing reachability graph
- Computing final path on reachability graph

Possible optimizations

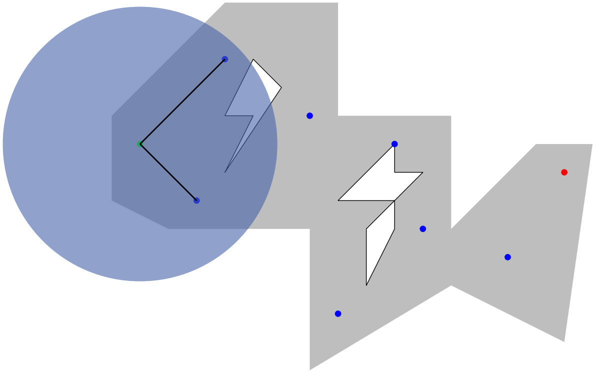
Computing reachability graph

- Compute shortest path from every charging station to all reachable charging stations



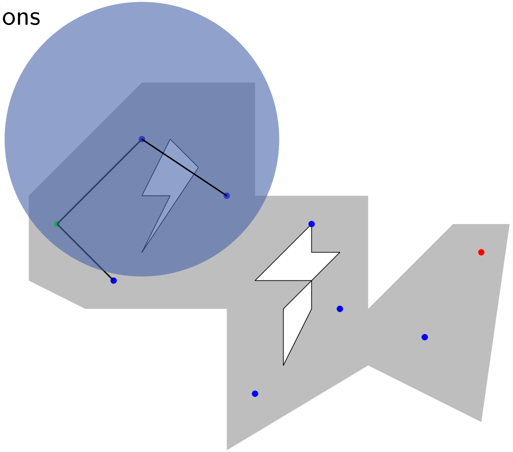
Computing reachability graph

- Compute shortest path from every charging station to all reachable charging stations



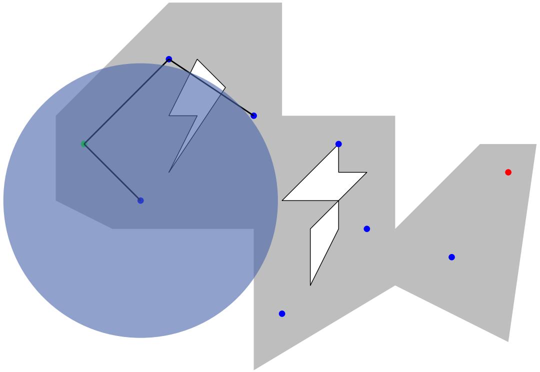
Computing reachability graph

- Compute shortest path from every charging station to all reachable charging stations



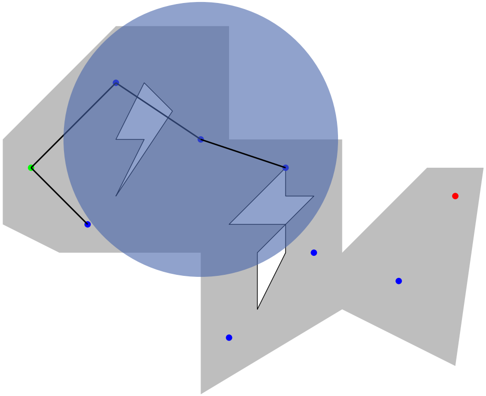
Computing reachability graph

- Compute shortest path from every charging station to all reachable charging stations



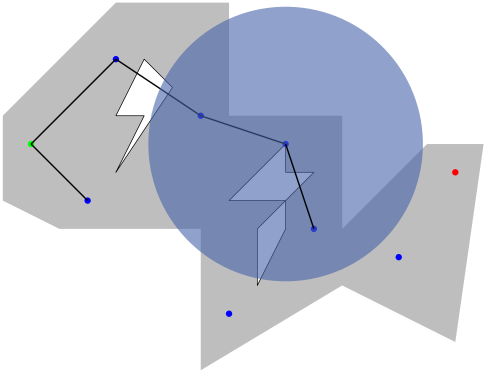
Computing reachability graph

- Compute shortest path from every charging station to all reachable charging stations



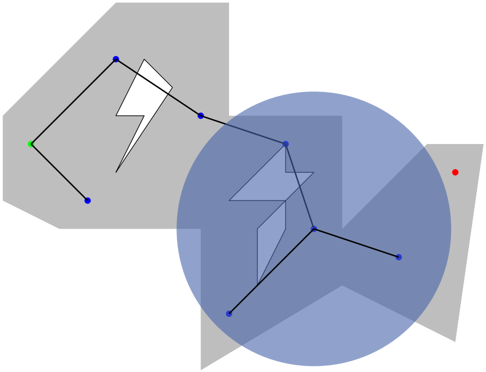
Computing reachability graph

- Compute shortest path from every charging station to all reachable charging stations



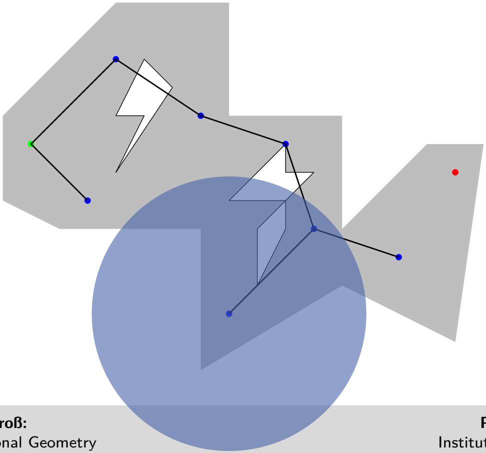
Computing reachability graph

- Compute shortest path from every charging station to all reachable charging stations



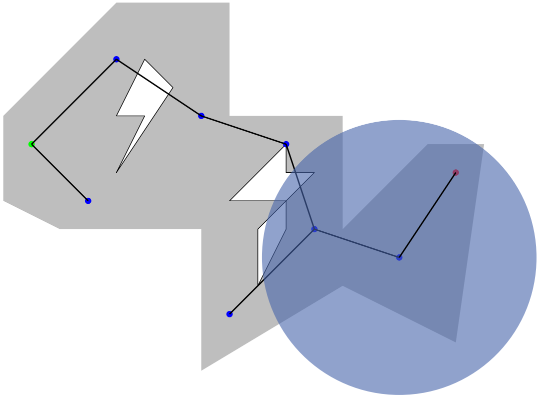
Computing reachability graph

- Compute shortest path from every charging station to all reachable charging stations



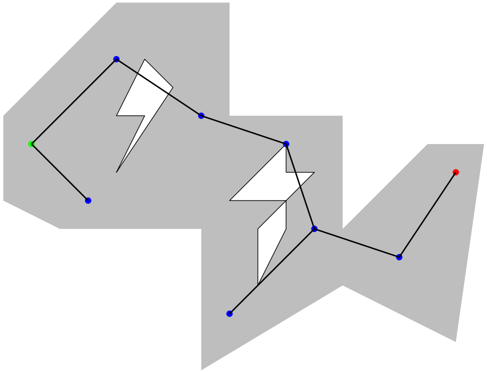
Computing reachability graph

- Compute shortest path from every charging station to all reachable charging stations



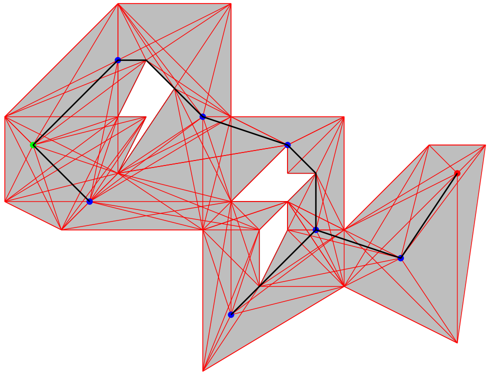
Computing reachability graph

- Compute shortest path from every charging station to all reachable charging stations



Computing reachability graph

- Compute shortest path from every charging station to all reachable charging stations



Status

~~Computing visibility graphs~~

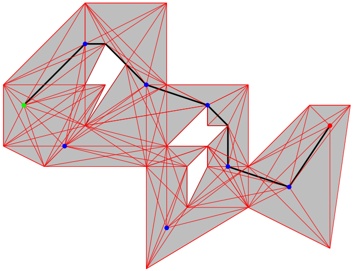
Computing shortest route

- ~~Computing reachability graph~~
- Computing final path on reachability graph

Possible optimizations

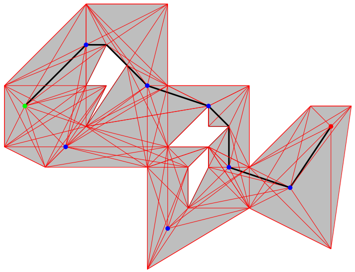
Computing final route

- Need one final (shortest) path computation from start to end node



Computing final route

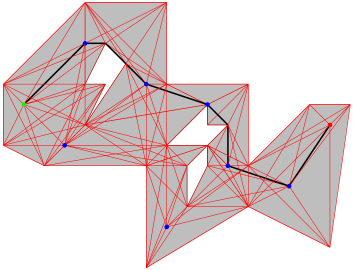
- Need one final (shortest) path computation from start to end node
- Runtime: Visibility graph + $(|C| + 3)$ shortest path calculations



Computing final route

- Need one final (shortest) path computation from start to end node
- Runtime: Visibility graph + $(|C| + 3)$ shortest path calculations
- Runtime:

$$\mathcal{O}(((|C| + |P|)^2 + |C| \cdot |E|) \cdot \log(|C| + |P|))$$



Status

~~Computing visibility graphs~~

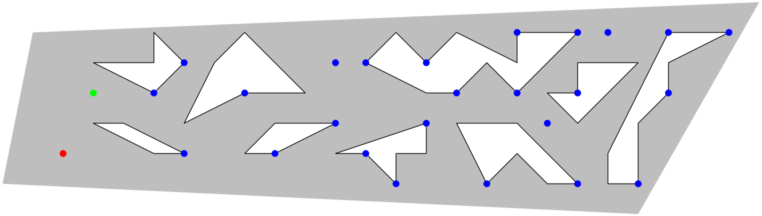
~~Computing shortest route~~

- ~~Computing reachability graph~~
- ~~Computing final path on reachability graph~~

Possible optimizations

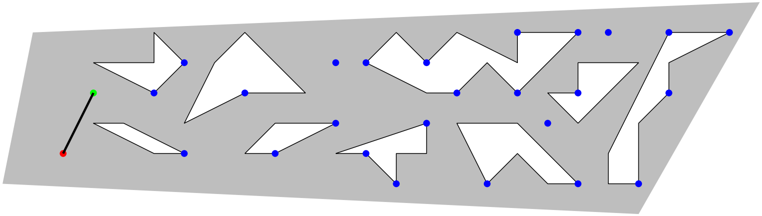
Further Optimization: Problem

- Huge input set, but start and end are close together (compared to size of the input)
- Need to compute visibility graph for whole input set, expensive!



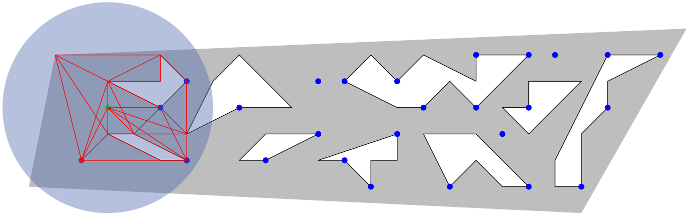
Further Optimization: Problem

- Huge input set, but start and end are close together (compared to size of the input)
- Need to compute visibility graph for whole input set, expensive!



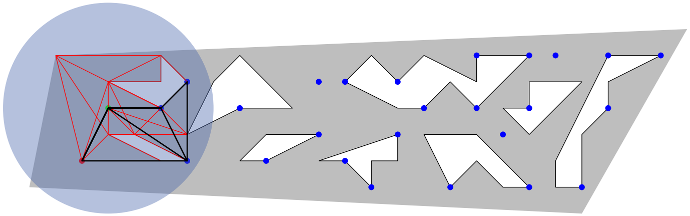
Further Optimization: Idea

- Compute visibility graph and shortest paths "on-demand":
  1. Start at s
  2. Compute visibility graph for all nodes in radius r around current node
  3. Perform shortest path search for each charging station in current radius
  4. Pick charging station with lowest distance to s as next node
  5. Go back to 2. until end node t found



Further Optimization: Idea

- Compute visibility graph and shortest paths "on-demand":
  1. Start at s
  2. Compute visibility graph for all nodes in radius r around current node
  3. Perform shortest path search for each charging station in current radius
  4. Pick charging station with lowest distance to s as next node
  5. Go back to 2. until end node t found



Questions?