Freie Universität Berlin

Approximation of Minimum Convex Partitioning

software project and competition 2019/20

Agenda

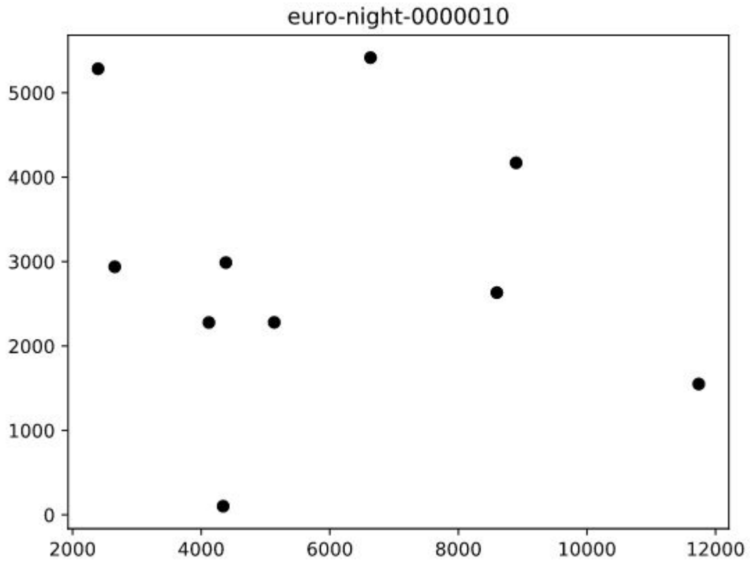
1. Introduction and Overview
2. DCEL
3. Nested Hulls
4. Single Convex Waves
5. Merged Convex Waves
6. Pass based
7. Start points
8. Solutions

1. Introduction and Overview

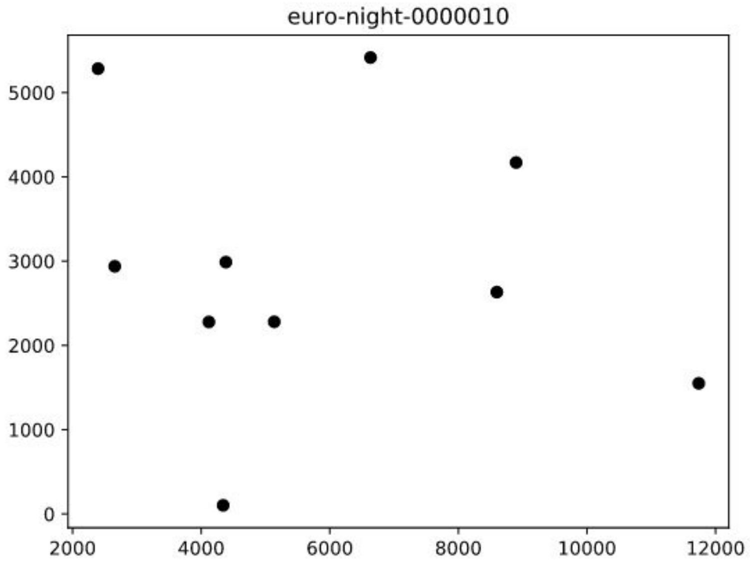
1. The CG Challenge 2020

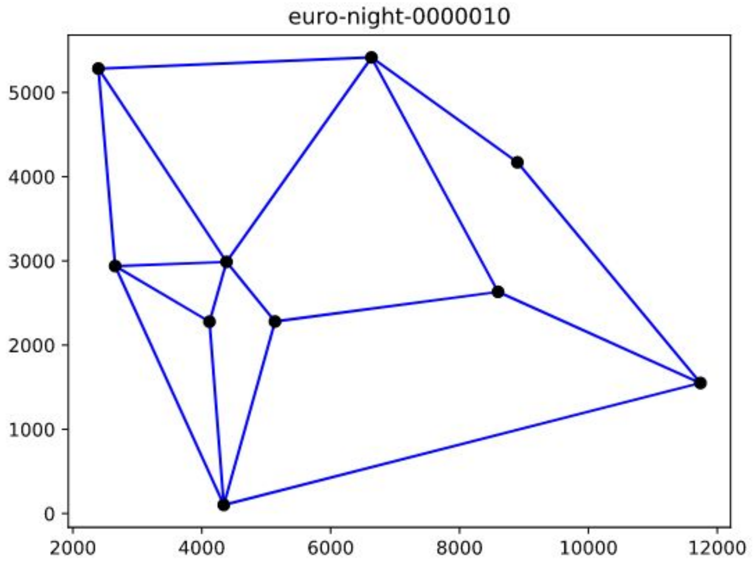
- CG:SHOP = Computational Geometry - Solving Hard Optimization Problems
- Part of the CG Week in Zurich (June 22-26)
- Open Class contest
- Opened: September 30
- Closes: February 14

1. The Minimum Convex Partition Problem



1. The Minimum Convex Partition Problem





Freie Universität Berlin

1. The Minimum Convex Partition Problem

- Complexity unknown
- At start: 247 instances
- Jan 21: 99 additional instances
- 4 types:
  - uniform
  - edge
  - illumination
  - orthogonally collinear points
- Tiebreaker: Time

1. Workflow

- Language: Python
- Communication: Slack
- Repository: GitHub
- Team meetings every Wednesday

1. Project Roadmap

23.10.19 Algorithm conception and proof of concept

1. Project Roadmap

23.10.19 Algorithm conception and proof of concept

1. Project Roadmap

23.10.19 Algorithm conception and proof of concept

6.11.19 Initial prototype

1. Project Roadmap

23.10.19 **Algorithm conception and proof of concept**

6.11.19 **Initial prototype**

1. Project Roadmap

23.10.19 Algorithm conception and proof of concept

6.11.19 Initial prototype

Common interface specification

1. Project Roadmap

23.10.19	Algorithm conception and proof of concept
6.11.19	Initial prototype
	Common interface specification

1. Project Roadmap

23.10.19 Algorithm conception and proof of concept

6.11.19 Initial prototype

Common interface specification

13.11.19 Multiple program runs

1. Project Roadmap

23.10.19	Algorithm conception and proof of concept
6.11.19	Initial prototype
	Common interface specification
13.11.19	Multiple program runs

1. Project Roadmap

23.10.19 Algorithm conception and proof of concept

6.11.19 Initial prototype

Common interface specification

13.11.19 Multiple program runs

20.11.19 Baseline results

1. Project Roadmap

23.10.19	Algorithm conception and proof of concept
6.11.19	Initial prototype
	Common interface specification
13.11.19	Multiple program runs
20.11.19	Baseline results

1. Project Roadmap

8.12.19 Alternative algorithms

1. Project Roadmap

8.12.19 Alternative algorithms

- Nested convex hulls

1. Project Roadmap

8.12.19 Alternative algorithms

- **Nested convex hulls**

1. Project Roadmap

8.12.19 Alternative algorithms

- **Nested convex hulls**
- Removing edges from triangulation

1. Project Roadmap

8.12.19 Alternative algorithms

- **Nested convex hulls**
- **Removing edges from triangulation**

1. Project Roadmap

8.12.19 Alternative algorithms

- **Nested convex hulls**
- **Removing edges from triangulation**
- Linear integer programming

1. Project Roadmap

8.12.19 Alternative algorithms

- Nested convex hulls
- Removing edges from triangulation
- Linear integer programming

1. Project Roadmap

8.12.19 Alternative algorithms

- **Nested convex hulls**
- **Removing edges from triangulation**
- **Linear integer programming**

25.12.19 Result comparison

1. Project Roadmap

8.12.19 Alternative algorithms

- Nested convex hulls
- Removing edges from triangulation
- Linear integer programming

25.12.19 Result comparison

1. Project Roadmap

8.12.19 Alternative algorithms

- **Nested convex hulls**
- **Removing edges from triangulation**
- **Linear integer programming**

25.12.19 Result comparison

8.1.20 User Interface

1. Project Roadmap

8.12.19 Alternative algorithms

- Nested convex hulls
- Removing edges from triangulation
- Linear integer programming

25.12.19 Result comparison

8.1.20 User Interface

1. Project Roadmap

8.12.19 Alternative algorithms

- **Nested convex hulls**
- **Removing edges from triangulation**
- **Linear integer programming**

25.12.19 Result comparison

8.1.20 User Interface

31.1.20 Miscellaneous improvements / alternatives

1. Project Roadmap

8.12.19 Alternative algorithms

- Nested convex hulls
- Removing edges from triangulation
- Linear integer programming

25.12.19 Result comparison

8.1.20 User Interface

31.1.20 Miscellaneous improvements / alternatives

1. Project Roadmap

8.12.19 Alternative algorithms

- Nested convex hulls
- Removing edges from triangulation
- Linear integer programming

25.12.19 Result comparison

8.1.20 User Interface

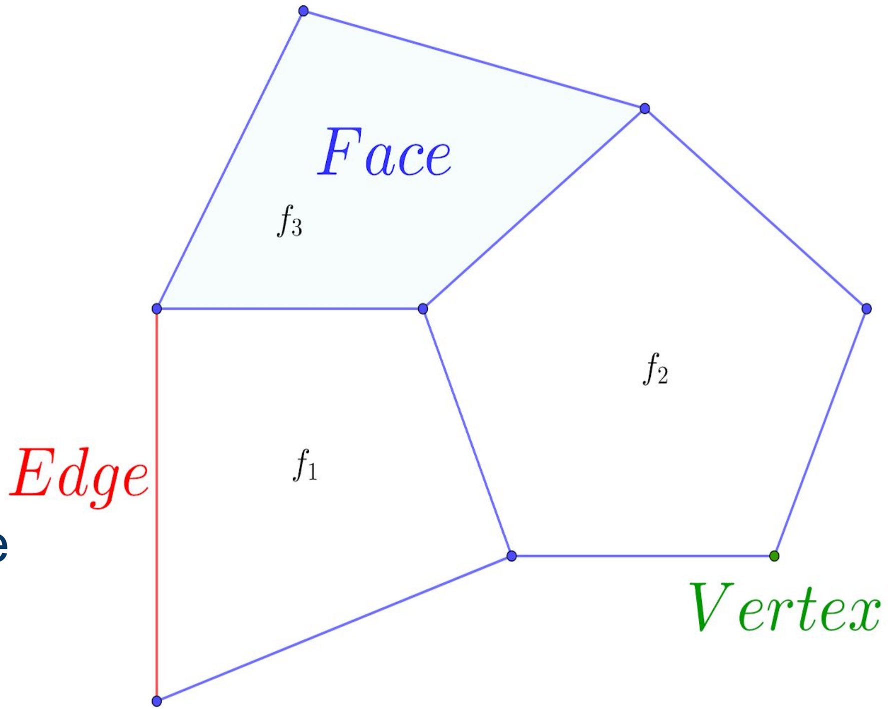
31.1.20 Miscellaneous improvements / alternatives

15.2.20 Contingency buffer

2. Doubly Connected Edge List (DCEL)

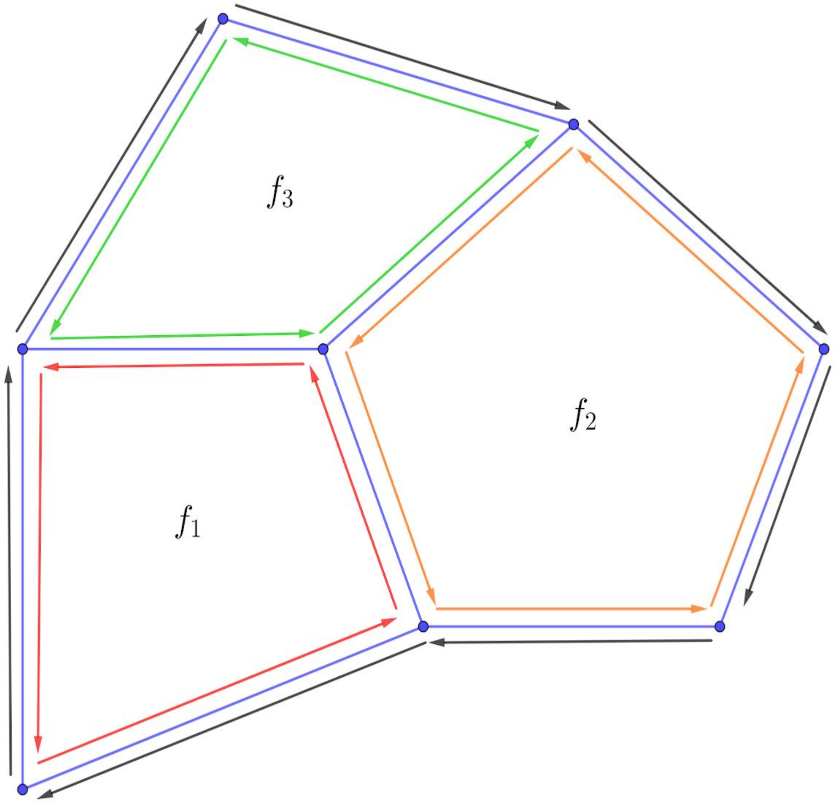
2. Doubly Connected Edge List (DCEL)

- Most commonly used representations for planar subdivisions
- It links together three sets of records:
  - ➢ Vertex
  - ➢ Edge
  - ➢ Face
- It provides the ability of traversing the faces of planar subdivision, visiting all the edges around a given vertex



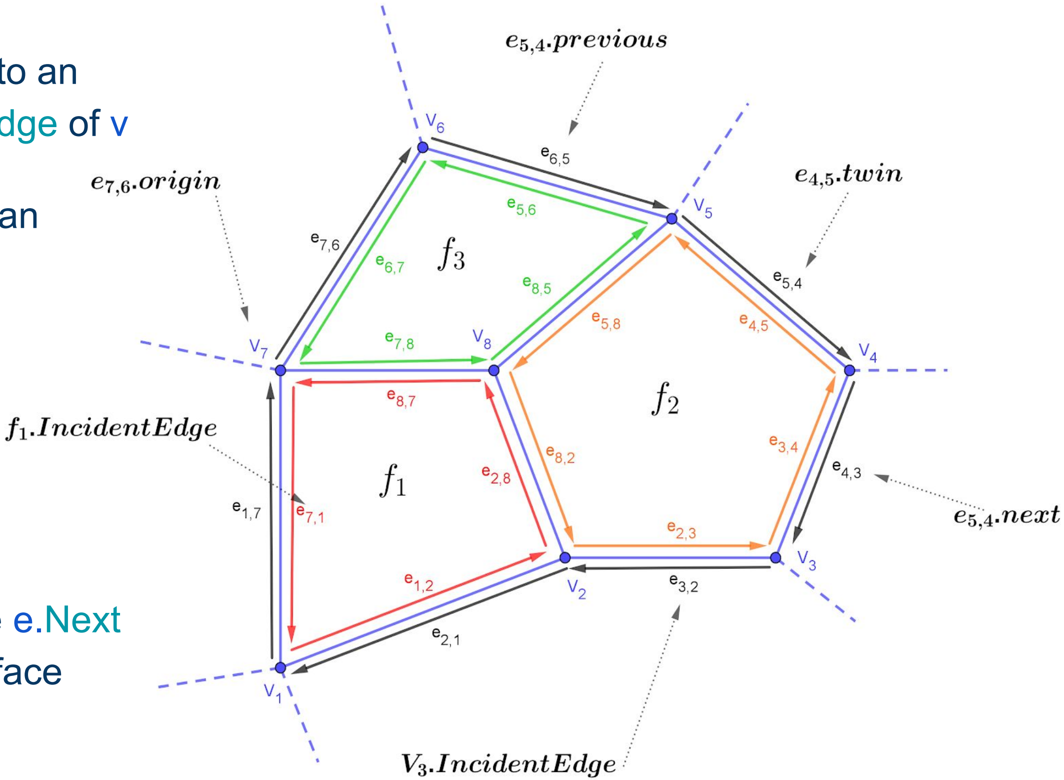
2. Doubly Connected Edge List (DCEL)

- Edges are oriented counterclockwise inside each face
- Each edge is a border between two faces, and is therefore represented by two half-edges, one for each face



2. Doubly Connected Edge List (DCEL)

- Each vertex entry v has a pointer that point to an arbitrary outgoing edge called the IncidentEdge of v
- Each face entry *f* has a pointer that point to an arbitrary half-edge on its border called the IncidentEdge of *f*
- Each half-edge entry e stores pointers to:
  - ➢ Its origin e.Origin
  - ➢ Its twin half-edge e.Twin
  - ➢ The face on its left e.IncidentFace
  - ➢ The next half-edge on its incident face e.Next
  - ➢ The previous hal-edge on its incident face e.Previous

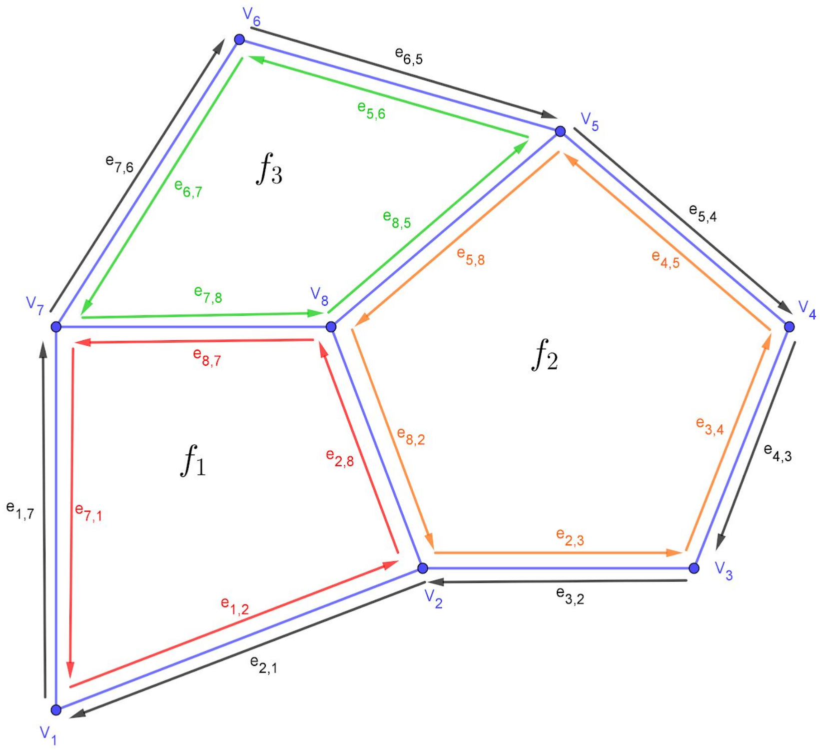


2. Doubly Connected Edge List (DCEL)

Vertex	Coordinates	IncidentEdge
v_1	(x_1,y_1)	$e_{1,2}$
v_2	(x_2,y_2)	$e_{2,8}$
...	...	...

Face	Edge
f_1	$e_{8,7}$
f_2	$e_{4,5}$
...	...

Half-edge	Origin	Twin	IncidentFace	Next	Previous
$e_{6,7}$	v_6	$e_{7,6}$	f_3	$e_{7,8}$	$e_{5,6}$
$e_{5,8}$	v_5	$e_{8,5}$	f_2	$e_{8,2}$	$e_{4,5}$
...	...	...	...	...	...



*In our implementation of DCEL we excluded the faces table, as we did not need it.

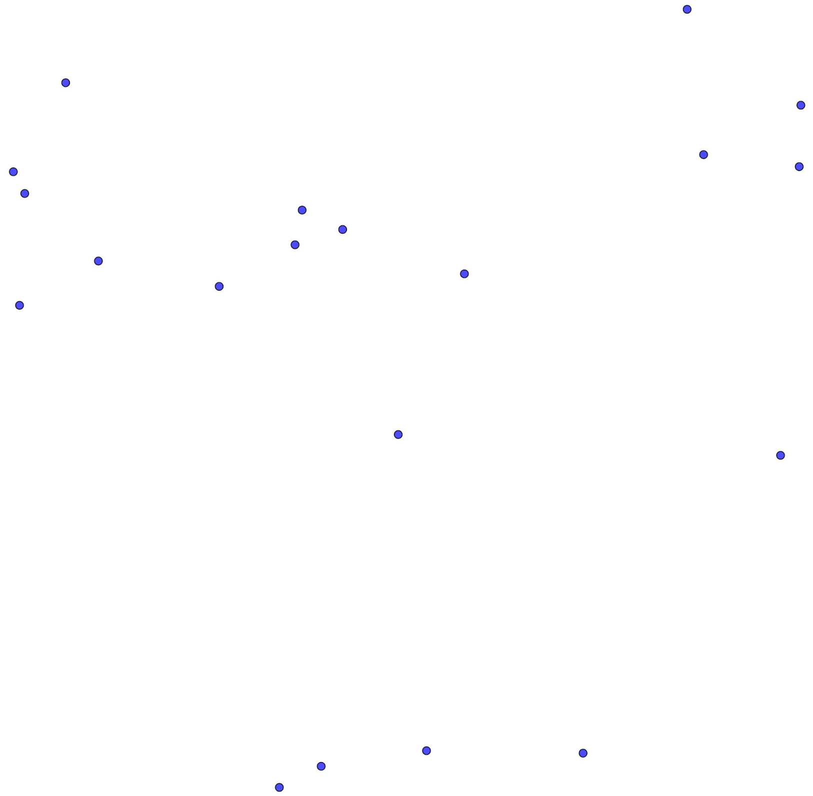
3. Nested Convex-Hulls Approach

3. Nested Convex-Hulls Approach

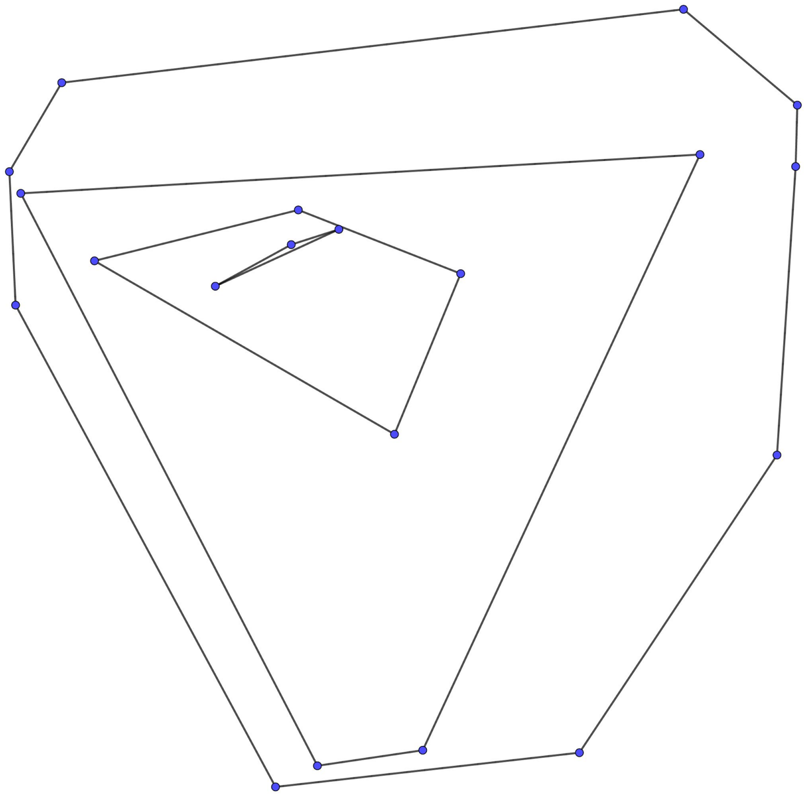
1. Iteratively keep computing c-hulls:
    1. Compute the c-hull of all points in the data set
    2. Subtract data points of the computed c-hull from the data set
    3. Repeat 1.1 & 1.2 until we get an empty data set
2. Connect each two sequential c-hulls in such a way that none of the added edges can be removed, unless we violate the convexity conditions
3. Except for the most outer c-hull, for each c-hull check for each edge if it can be removed

3. Nested Convex-Hulls, An Example

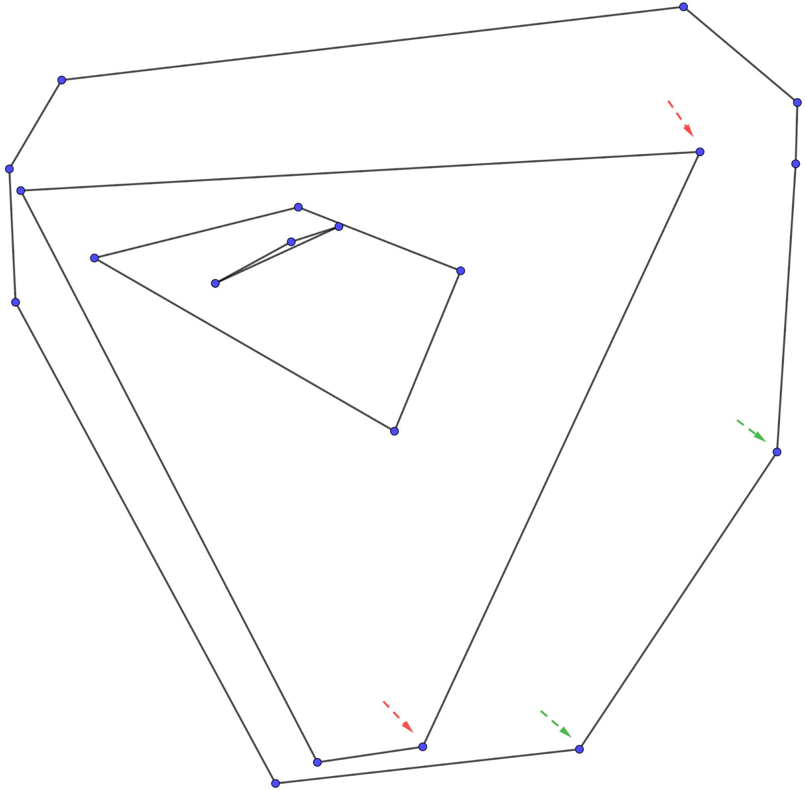
“stars-0000020”



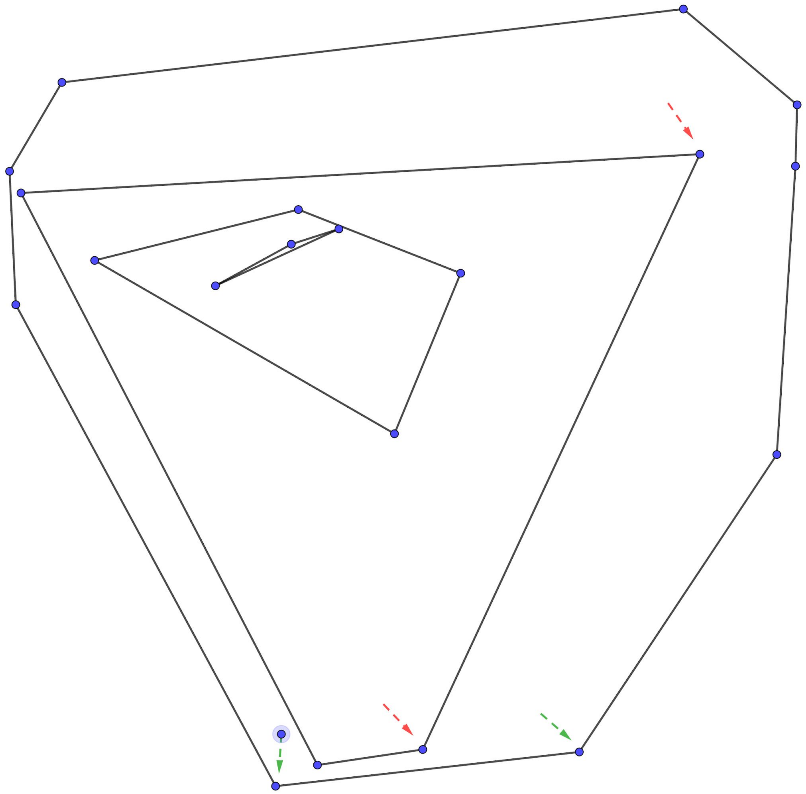
3. Example: Constructing C-Hulls



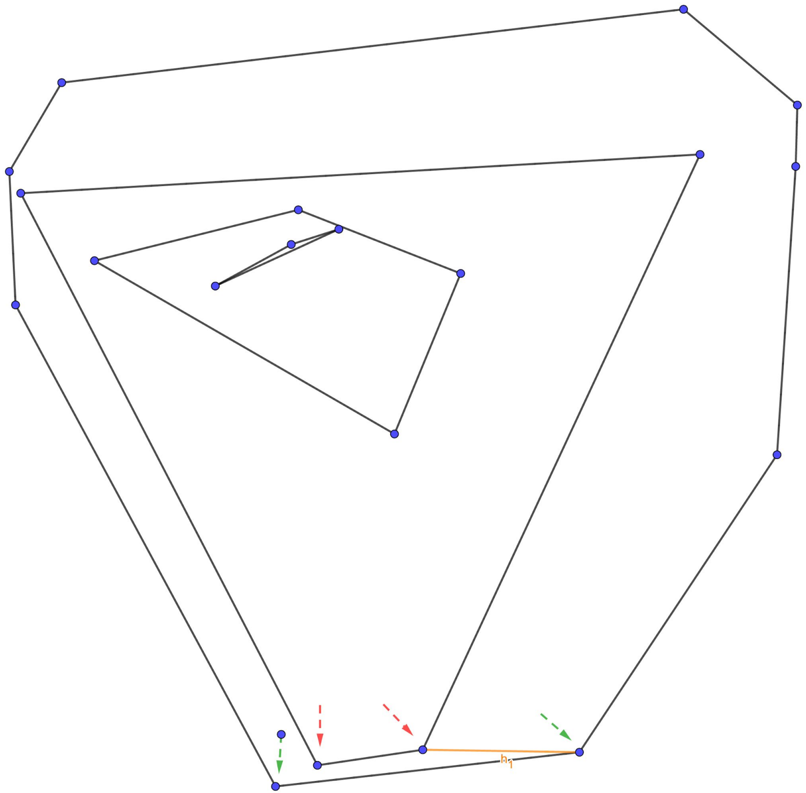
3. Example: Connecting Sequential Hulls



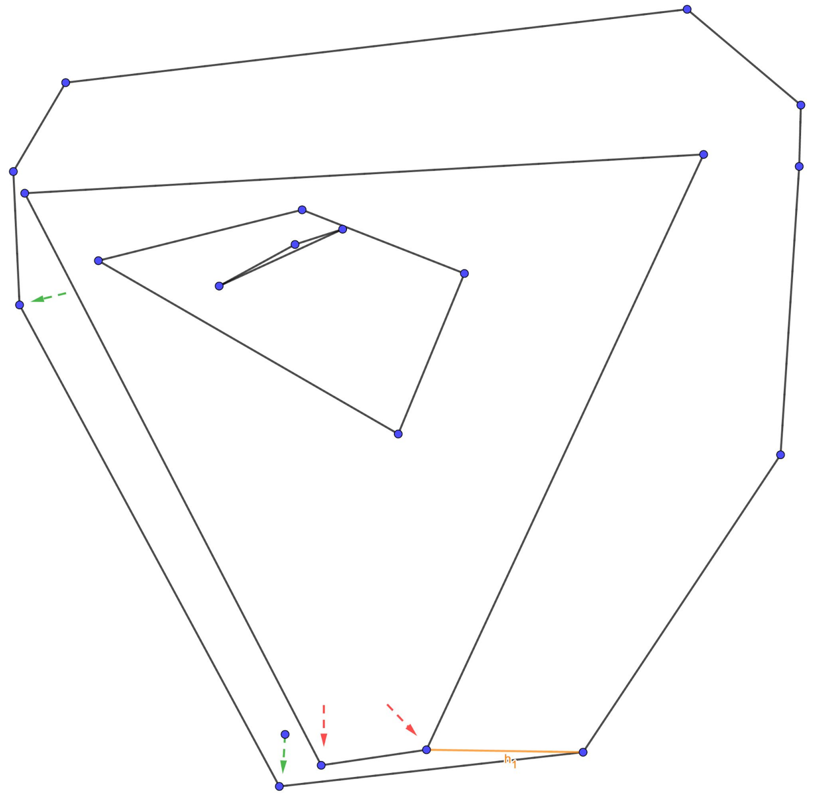
3. Example: Connecting Sequential Hulls



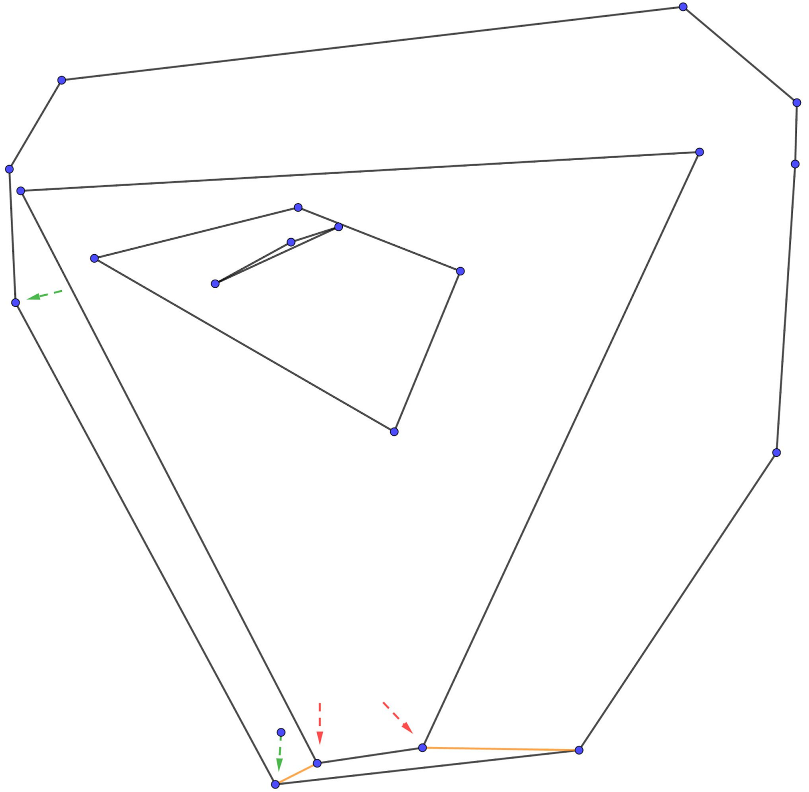
3. Example: Connecting Sequential Hulls



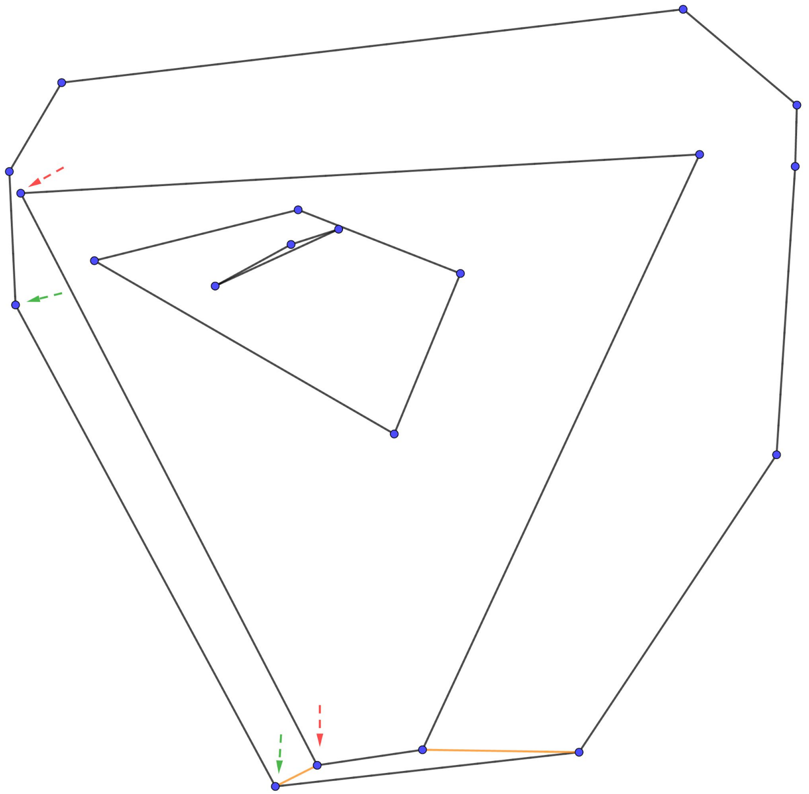
3. Example: Connecting Sequential Hulls



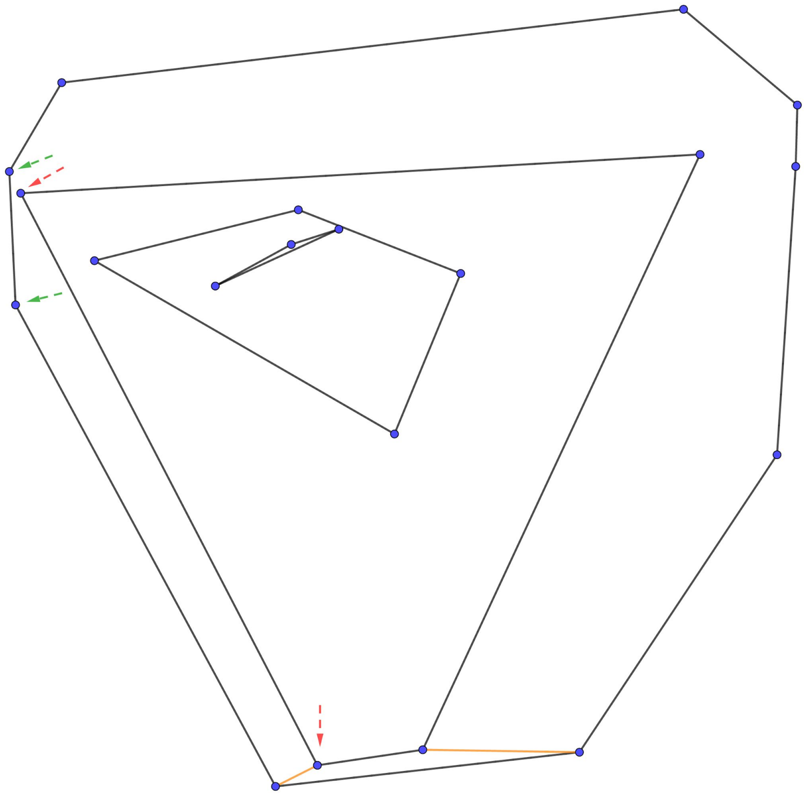
3. Example: Connecting Sequential Hulls



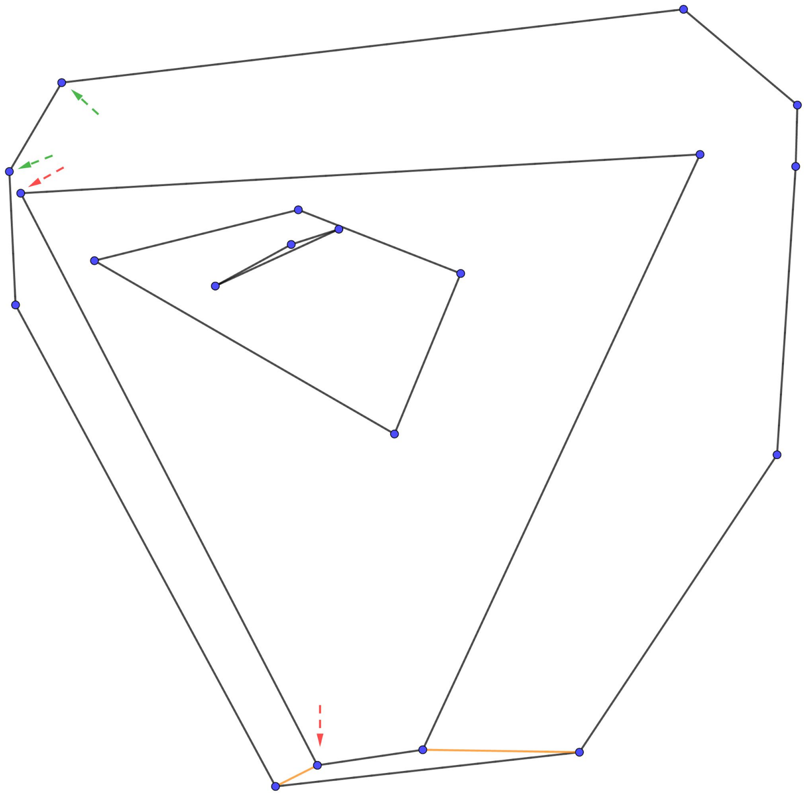
3. Example: Connecting Sequential Hulls



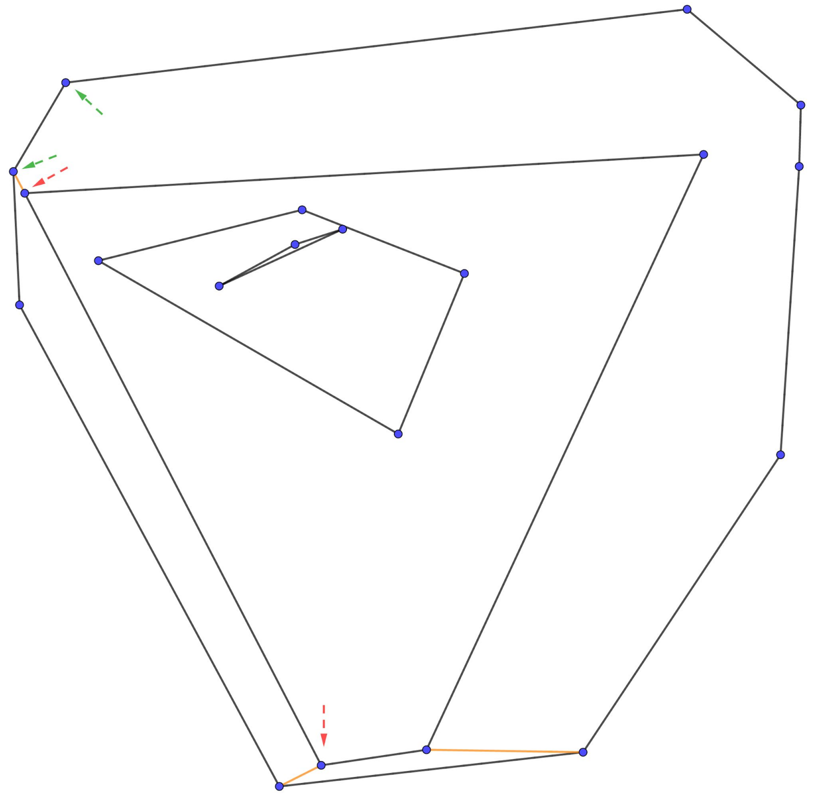
3. Example: Connecting Sequential Hulls



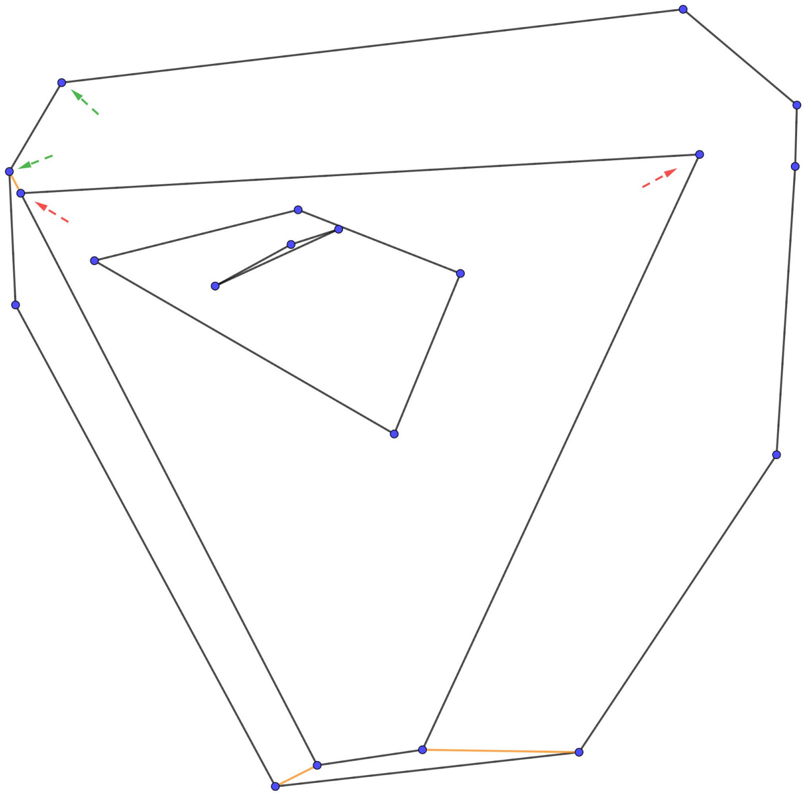
3. Example: Connecting Sequential Hulls



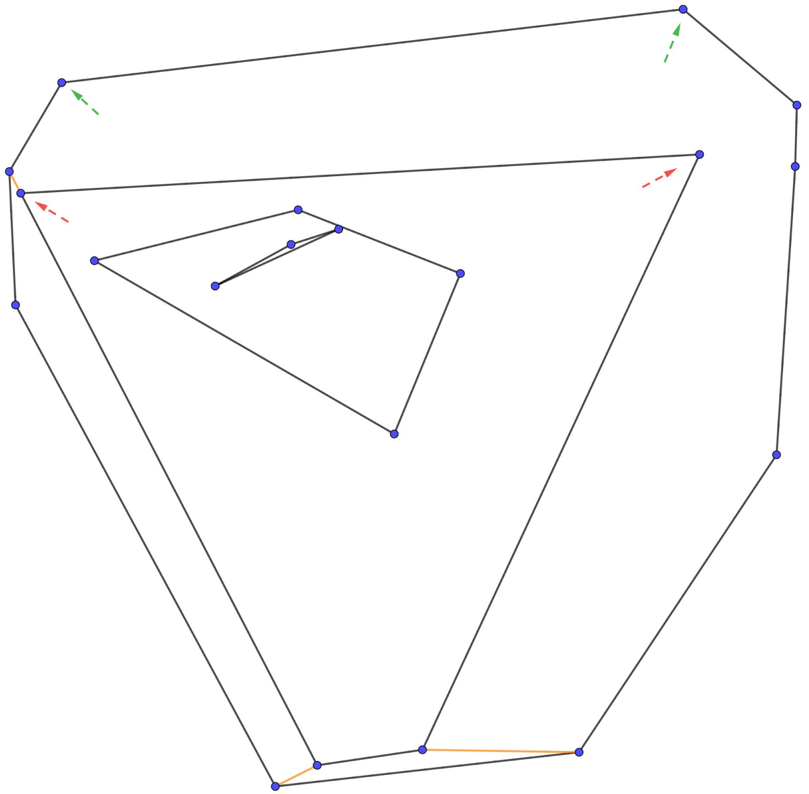
3. Example: Connecting Sequential Hulls



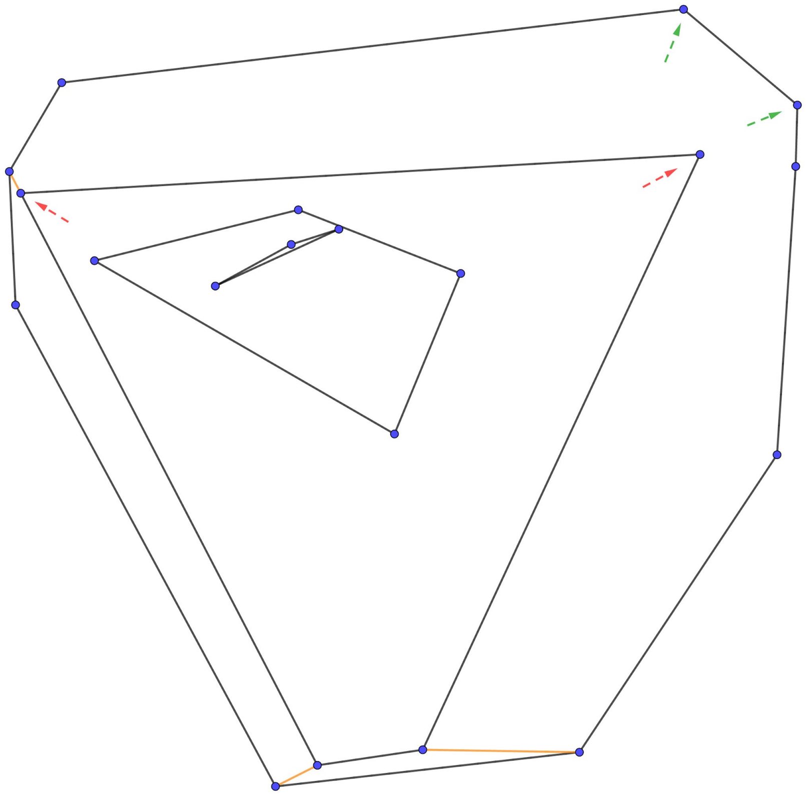
3. Example: Connecting Sequential Hulls



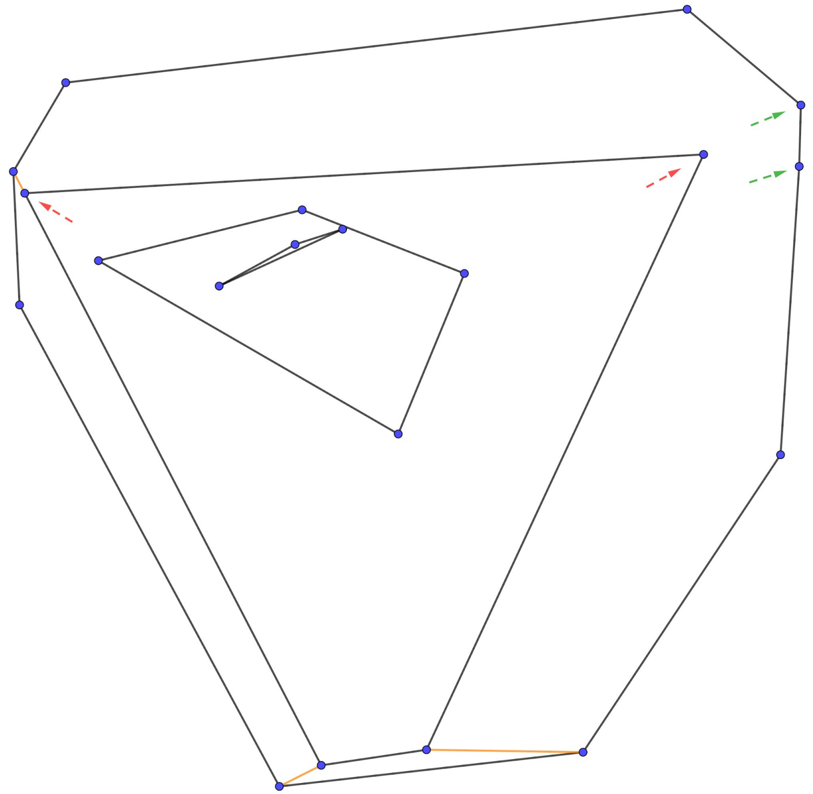
3. Example: Connecting Sequential Hulls



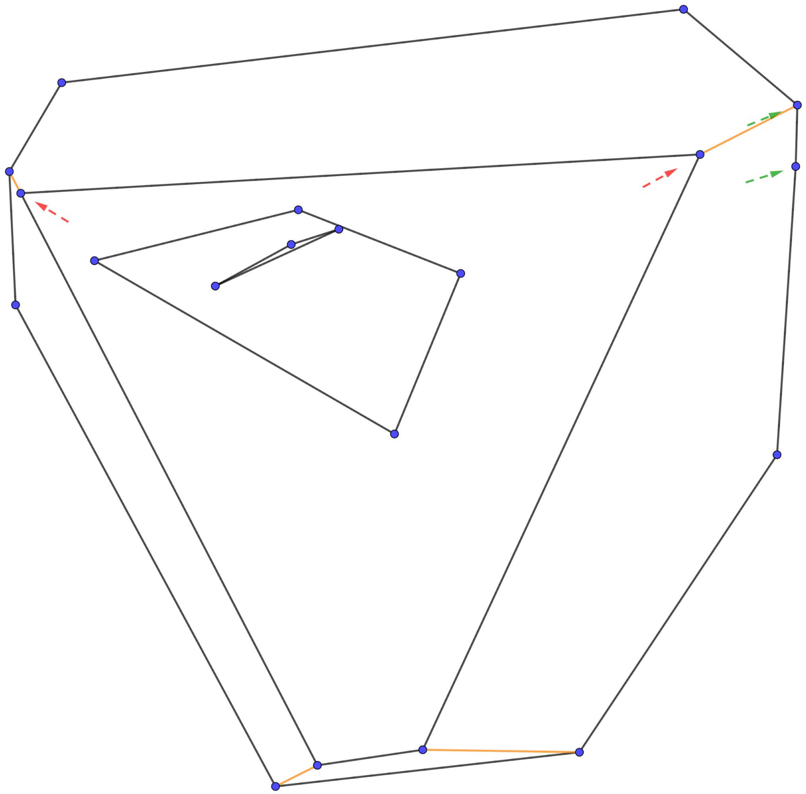
3. Example: Connecting Sequential Hulls



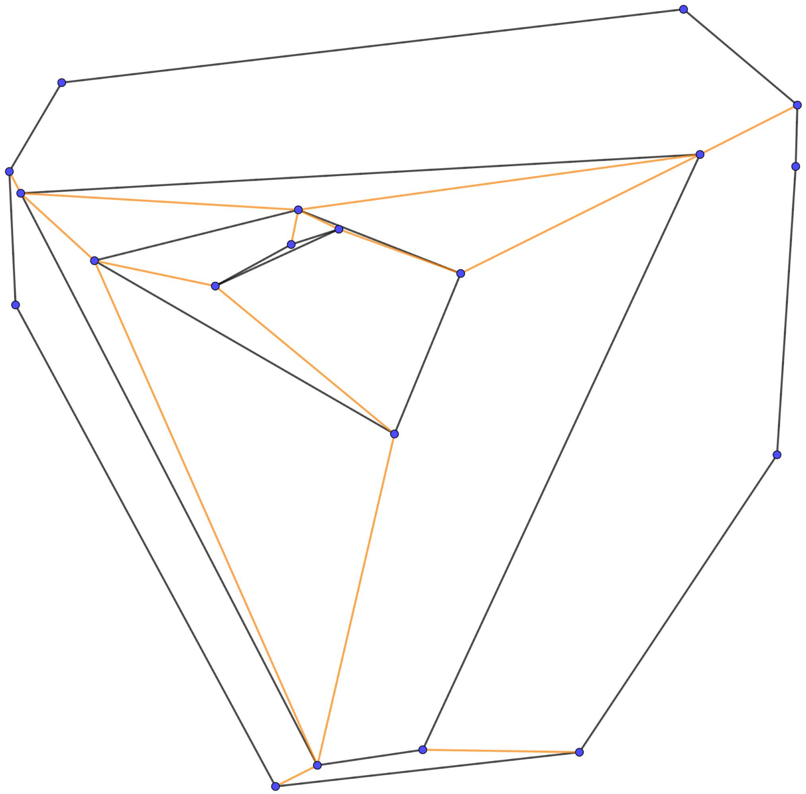
3. Example: Connecting Sequential Hulls



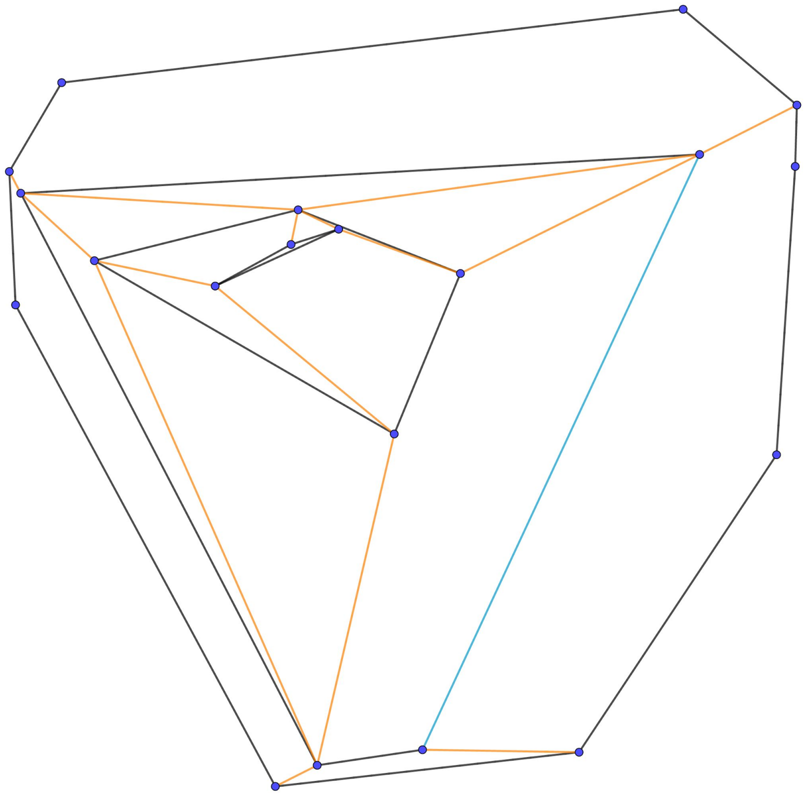
3. Example: Connecting Sequential Hulls



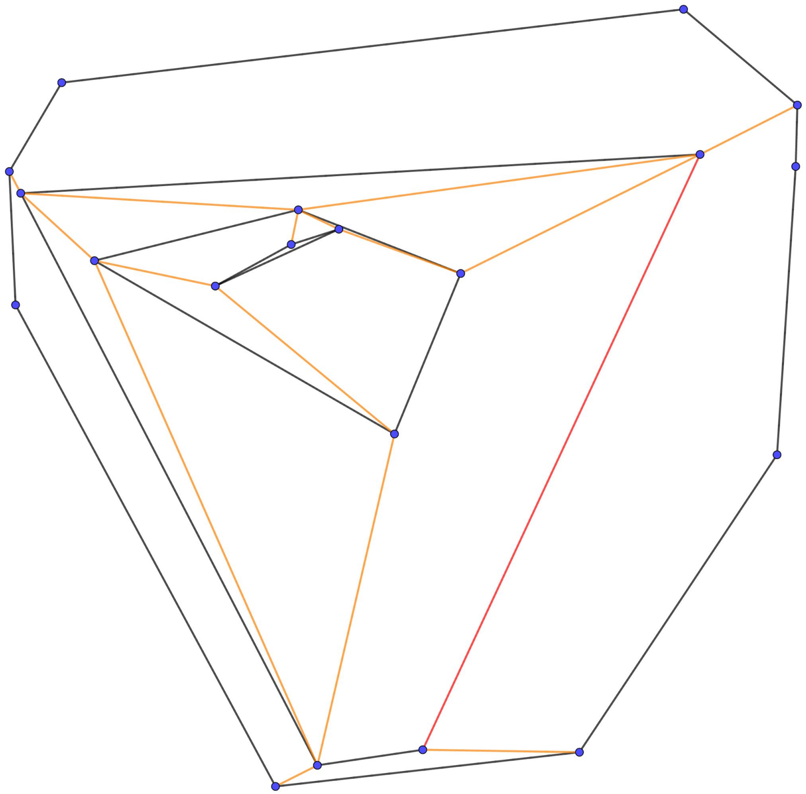
3. Example: Connecting Sequential Hulls



3. Example: Removing Unneeded Edges

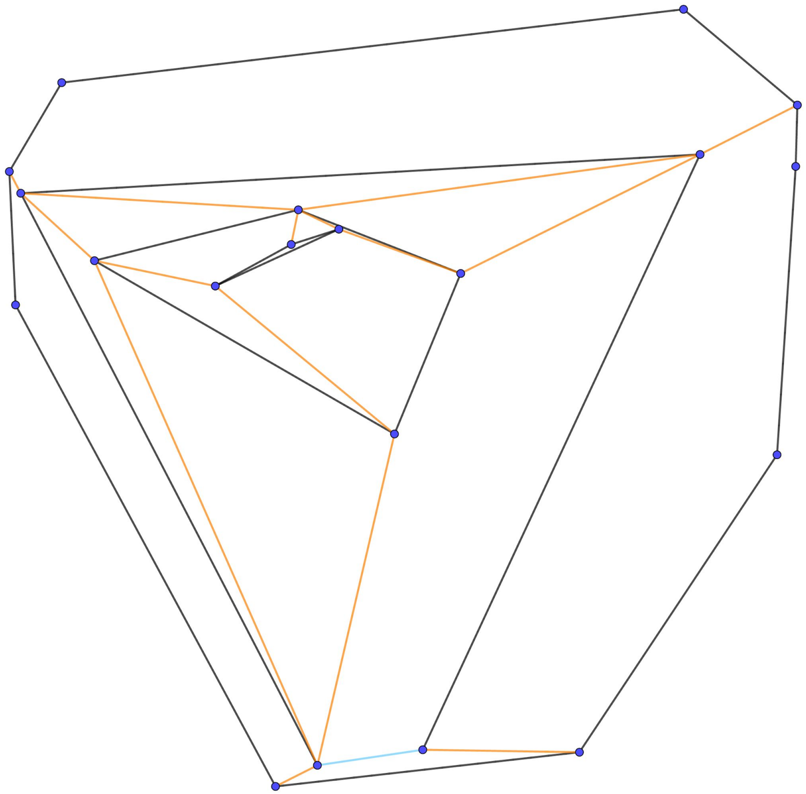


3. Example: Removing Unneeded Edges

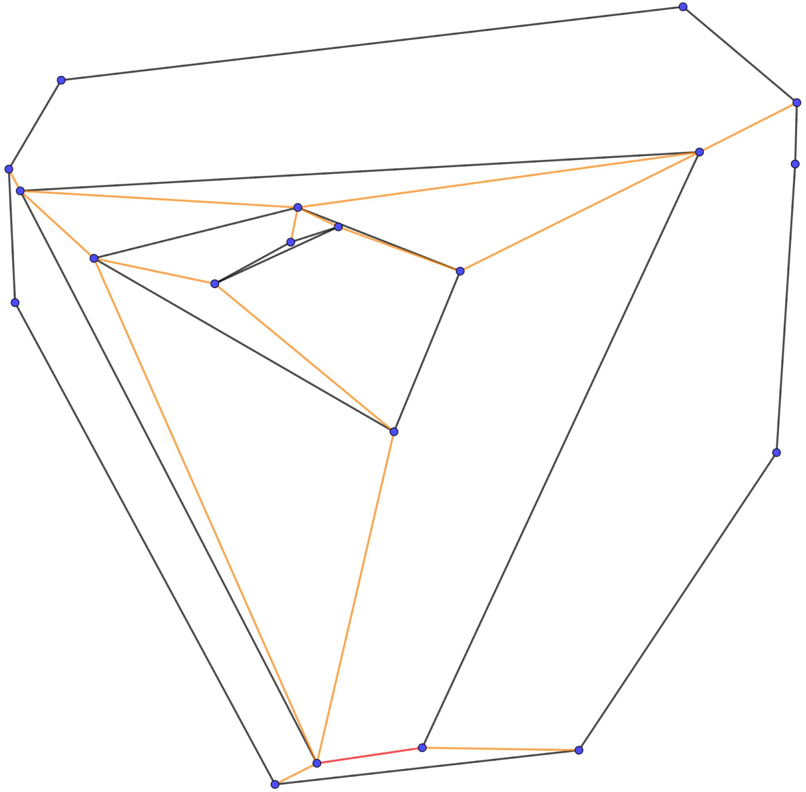


Freie Universität Berlin

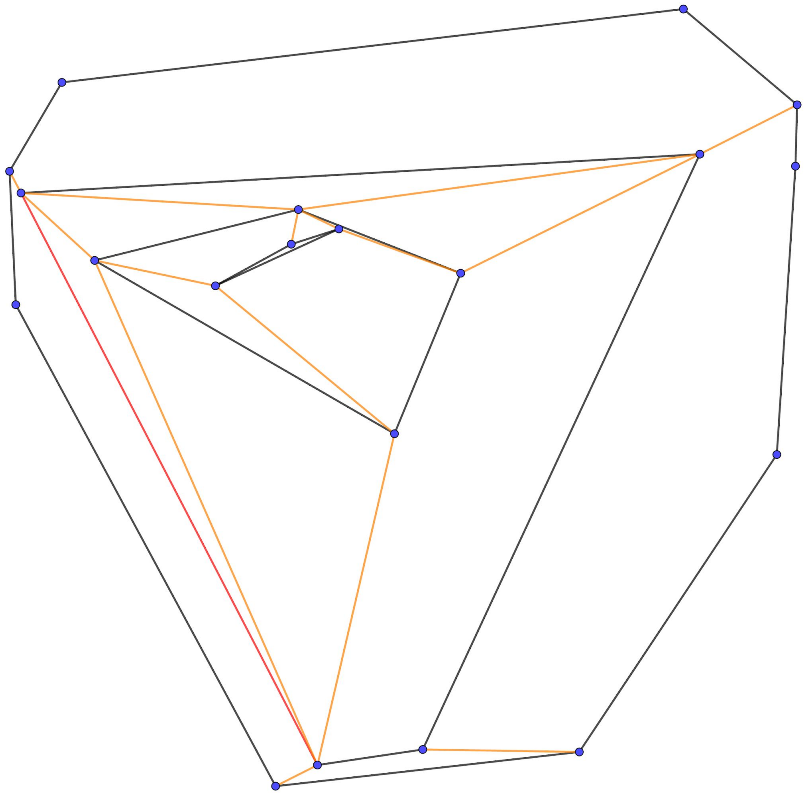
3. Example: Removing Unneeded Edges



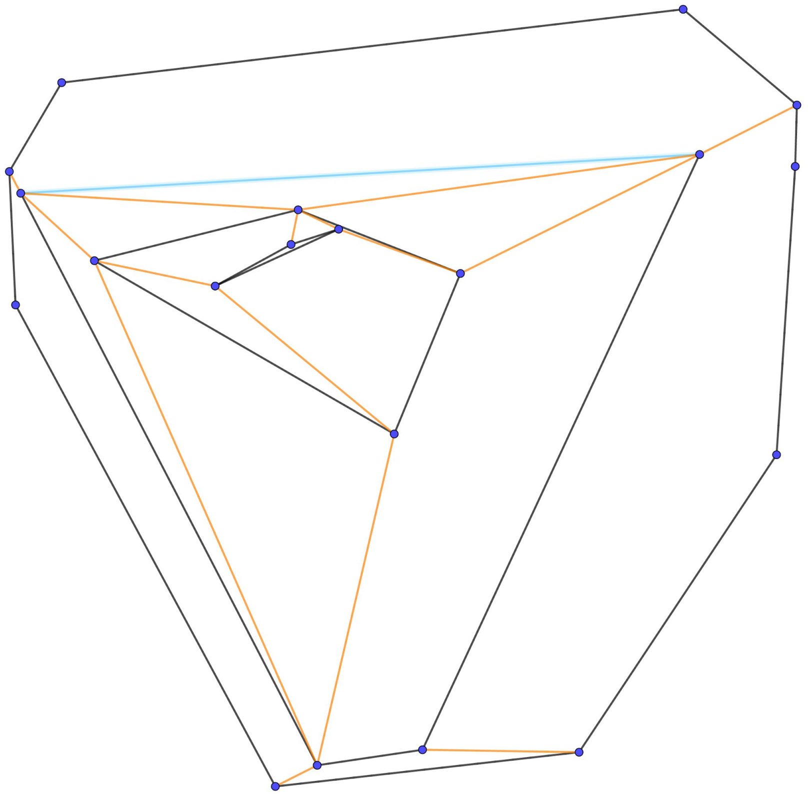
3. Example: Removing Unneeded Edges



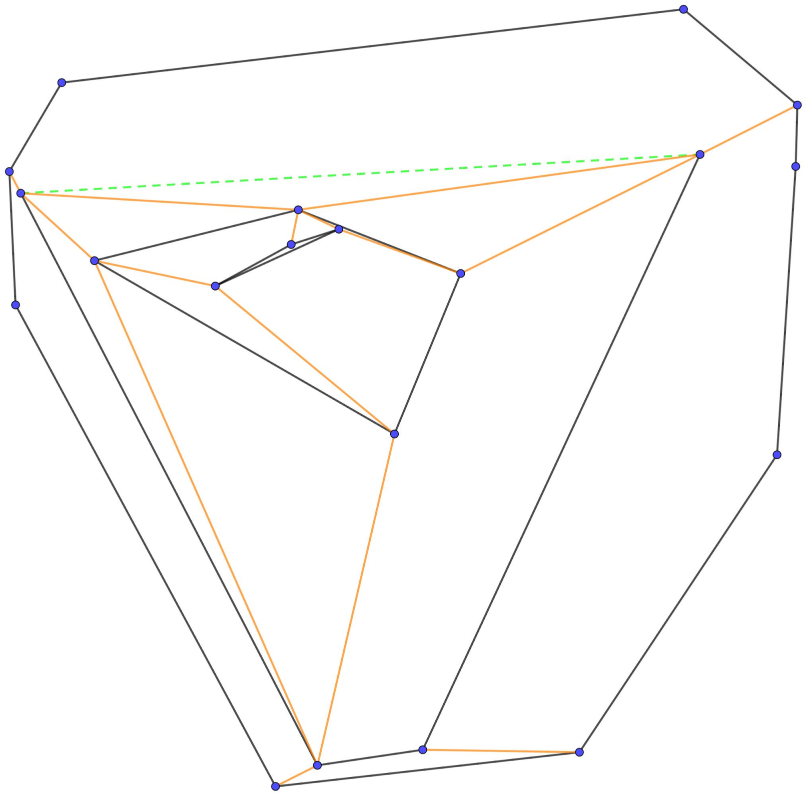
3. Example: Removing Unneeded Edges



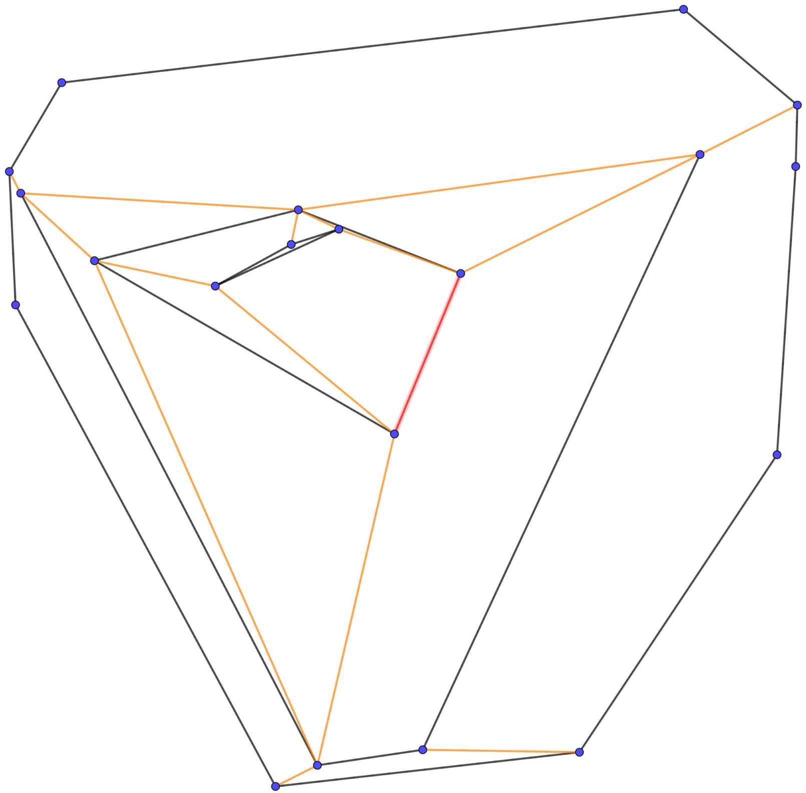
3. Example: Removing Unneeded Edges



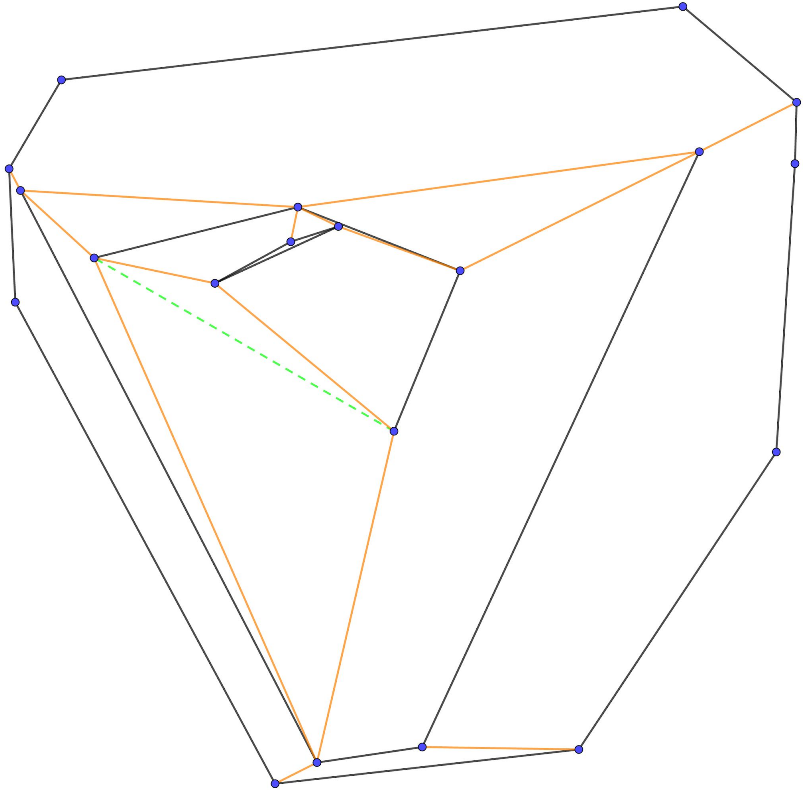
3. Example: Removing Unneeded Edges



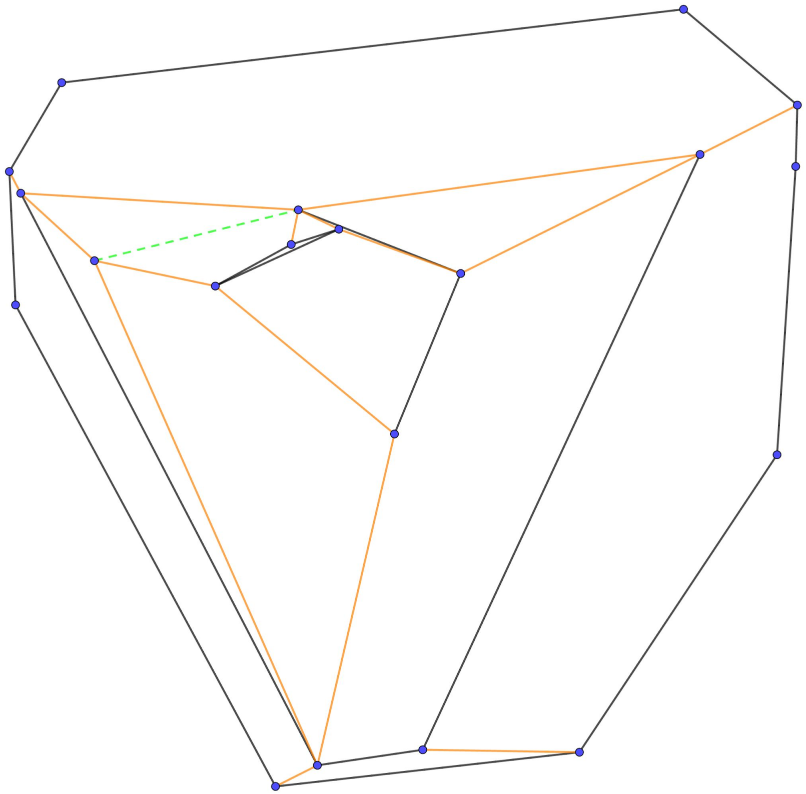
3. Example: Removing Unneeded Edges



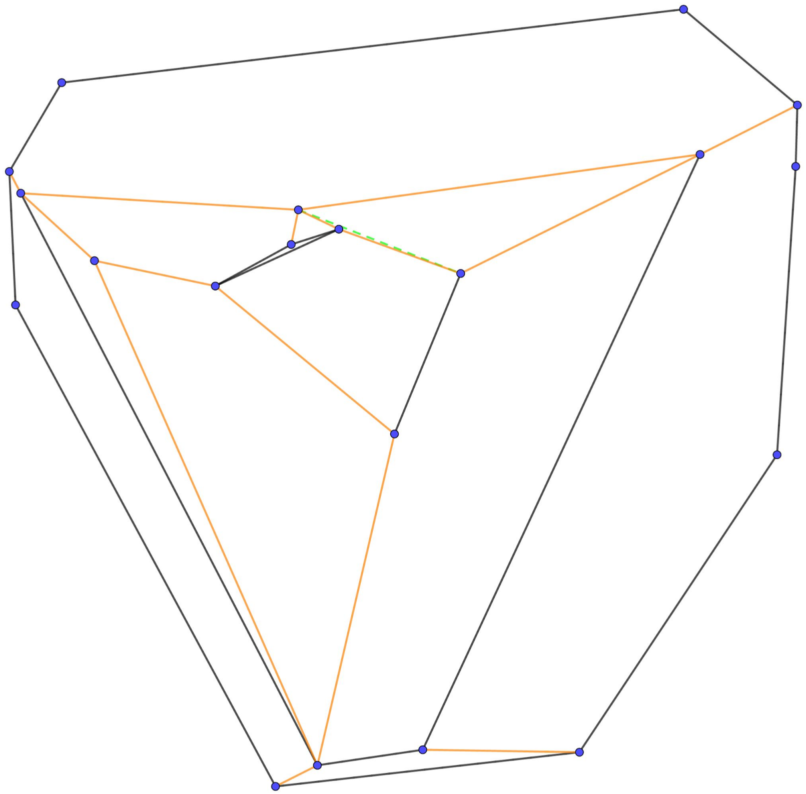
3. Example: Removing Unneeded Edges



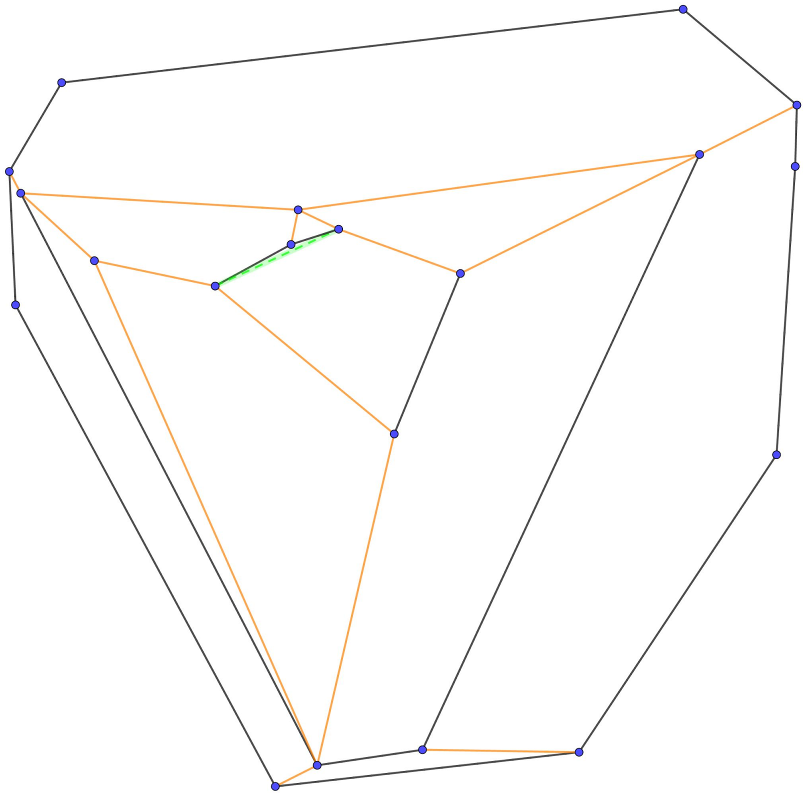
3. Example: Removing Unneeded Edges



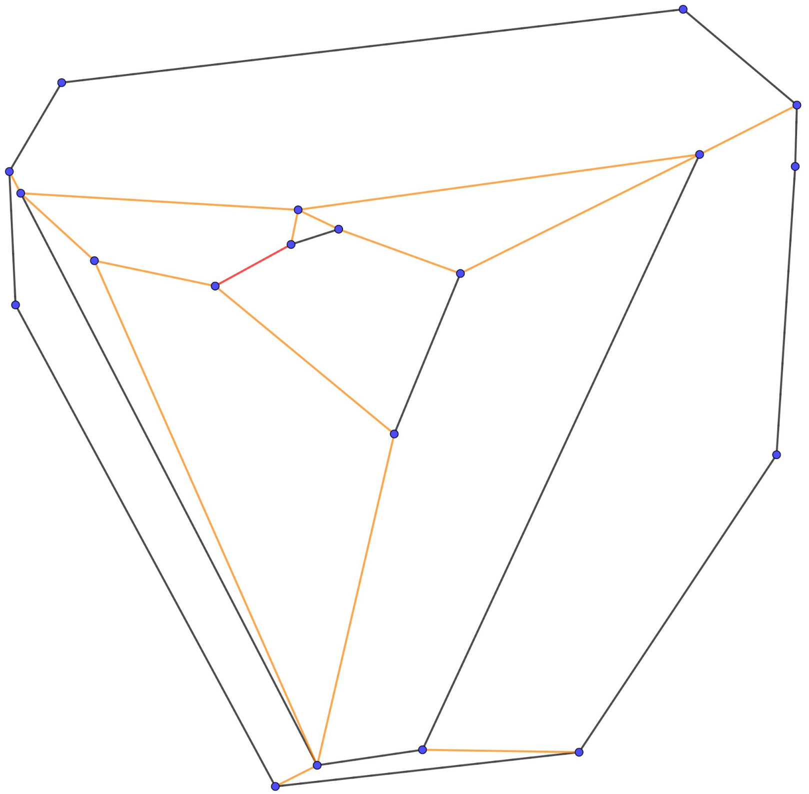
3. Example: Removing Unneeded Edges



3. Example: Removing Unneeded Edges

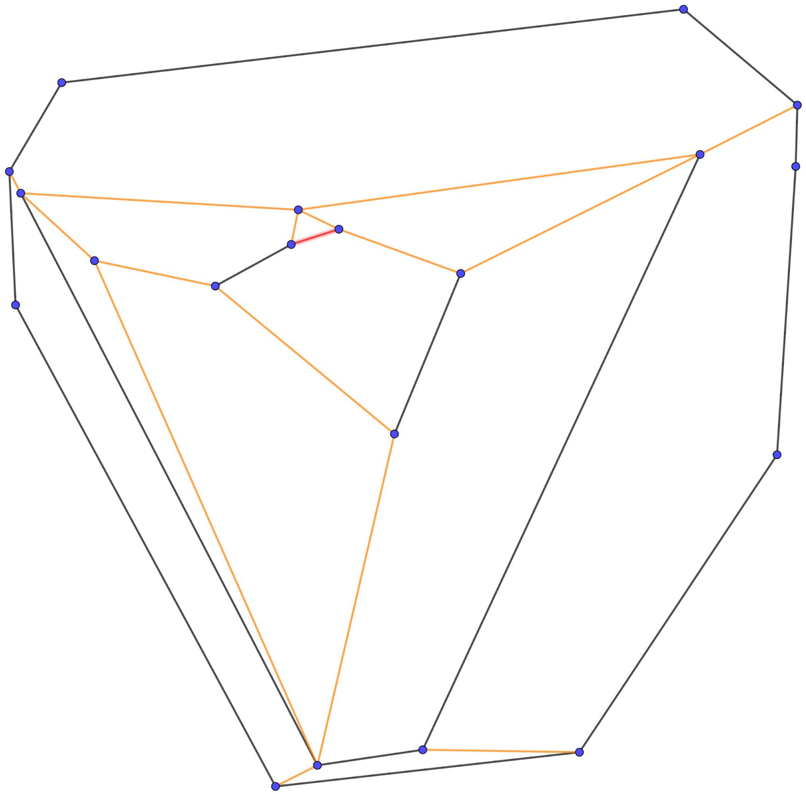


3. Example: Removing Unneeded Edges

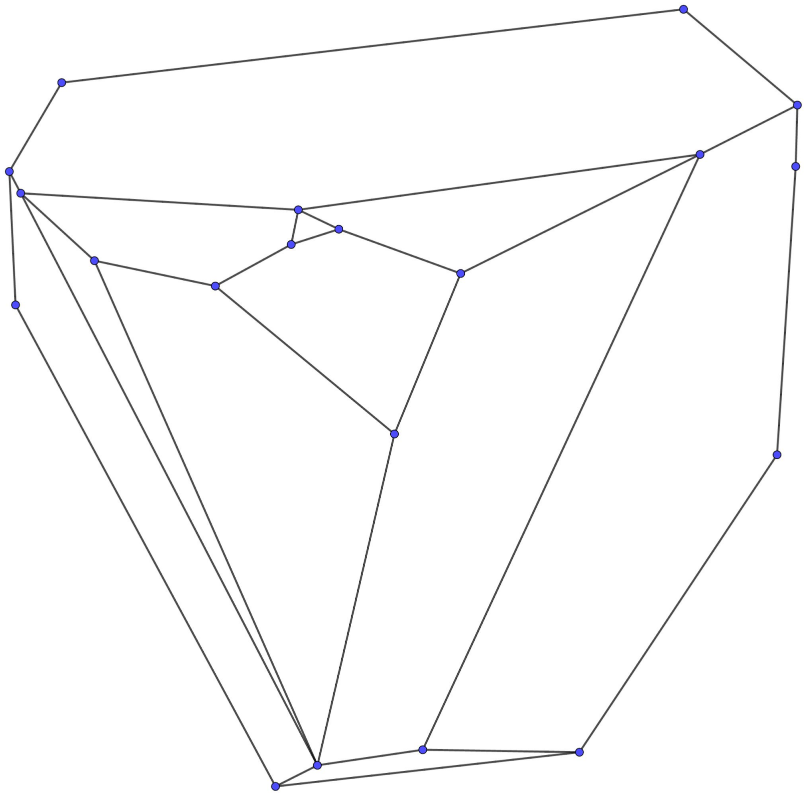


Freie Universität Berlin

3. Example: Removing Unneeded Edges



3. Example: Final Result



20 Vertices, 30 Edges & 11 Faces

3. Nested Convex-Hulls, An Upper Bound

For V to be the number of vertices, we have:

- When constructing nested c-hulls, we add at most V edges
- When connecting two c-hulls, we connect each vertex of the inner hull to at most 2 vertices of the outer hull, except when only one vertex left as last c-hull, which need to be connected to at most 3 vertices of the outer hull, so the worst case would be:
    - If the most outer hull is of size 3 and the most inner hull is of size 1, then for connecting hulls we add at most 2*(V-3) +1 edges
- Suppose in the deletion step no edge was eligible to be deleted

Then the max number of edges that can be added is V + 2*(V-3)+1 = 3V-5 edges

*In practice: 2V - ~ 20% “ 20% of 2V”

Freie Universität Berlin

3. Nested Convex-Hulls, Run-Time

Sorting: $O(nlog(n))$

Constructing Convex-Hulls : $O(n^2)$

Connecting Convex-Hulls : $O(n)$

Deleting Edges: $O(n)$

Overall Run-Time: $O(n^2)$

4. Convex Waves

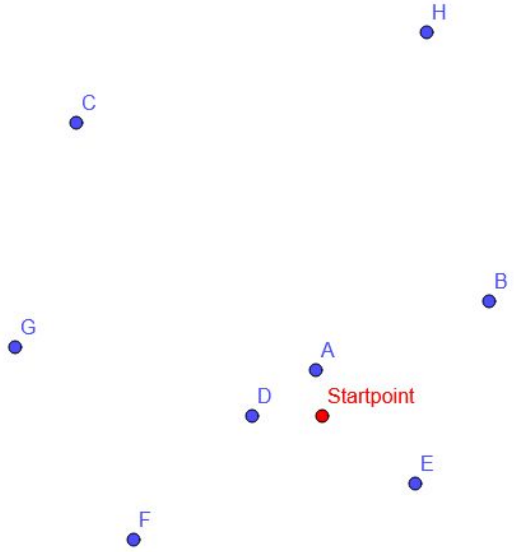
4. Convex Waves

Sort by distance to *startpoint* $\rightarrow Q$

First three points in $Q \rightarrow H$

For each point p in Q:

- Get visible bounds of p to H
- Connect p to all points in-between
- Remove redundant edges
- Update H to the convex hull



[A,D,E,B,F,G,C,H]

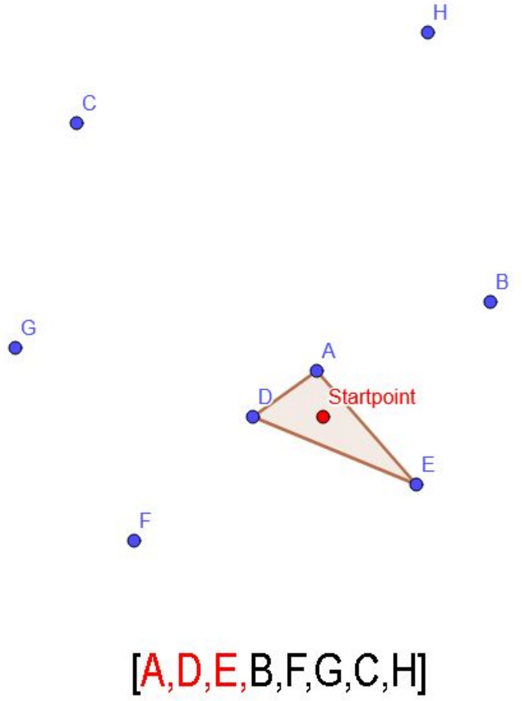
4. Convex Waves

Sort by distance to *startpoint* $\rightarrow Q$

First three points in $Q \rightarrow H$

For each point p in Q:

- Get visible bounds of p to H
- Connect p to all points in-between
- Remove redundant edges
- Update H to the convex hull



Freie Universität Berlin

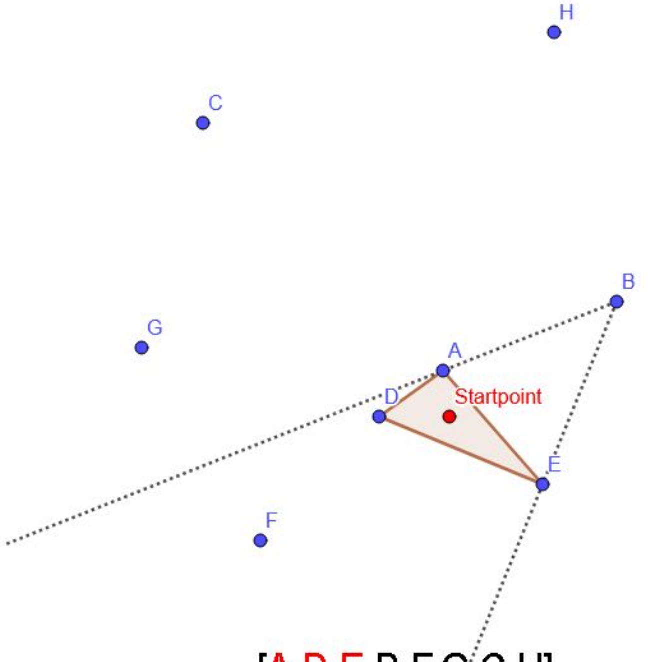
4. Convex Waves

Sort by distance to *startpoint* $\rightarrow Q$

First three points in $Q \rightarrow H$

For each point p in Q:

- Get visible bounds of p to H
- Connect p to all points in-between
- Remove redundant edges
- Update H to the convex hull



[A,D,E,B,F,G,C,H]

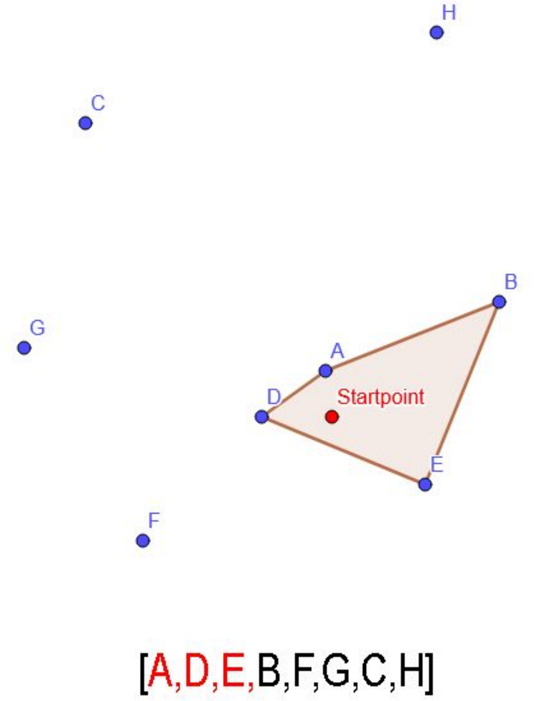
4. Convex Waves

Sort by distance to *startpoint* $\rightarrow Q$

First three points in $Q \rightarrow H$

For each point p in Q:

- Get visible bounds of p to H
- Connect p to all points in-between
- Remove redundant edges
- Update H to the convex hull



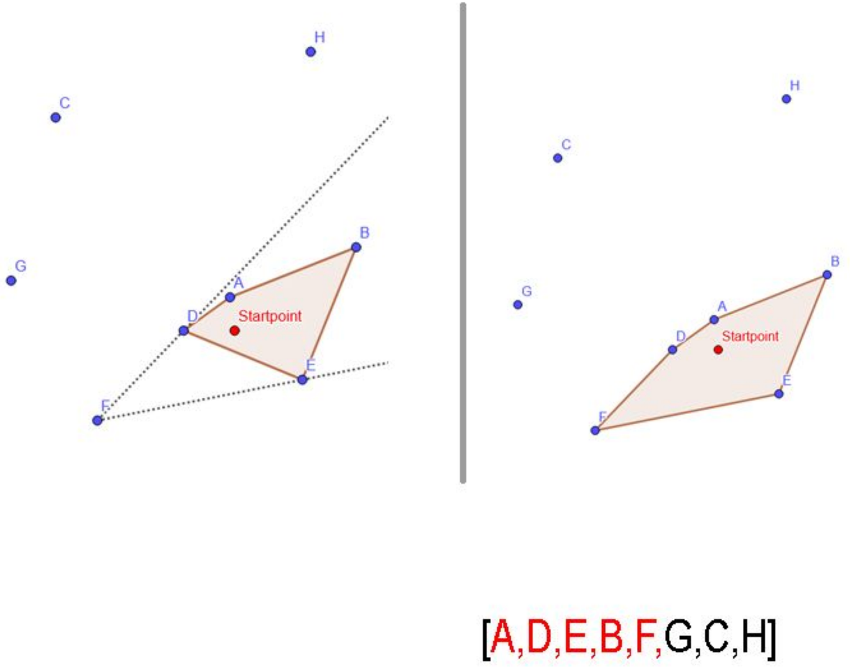
4. Convex Waves

Sort by distance to *startpoint* $\rightarrow Q$

First three points in $Q \rightarrow H$

For each point p in Q:

- Get visible bounds of p to H
- Connect p to all points in-between
- Remove redundant edges
- Update H to the convex hull



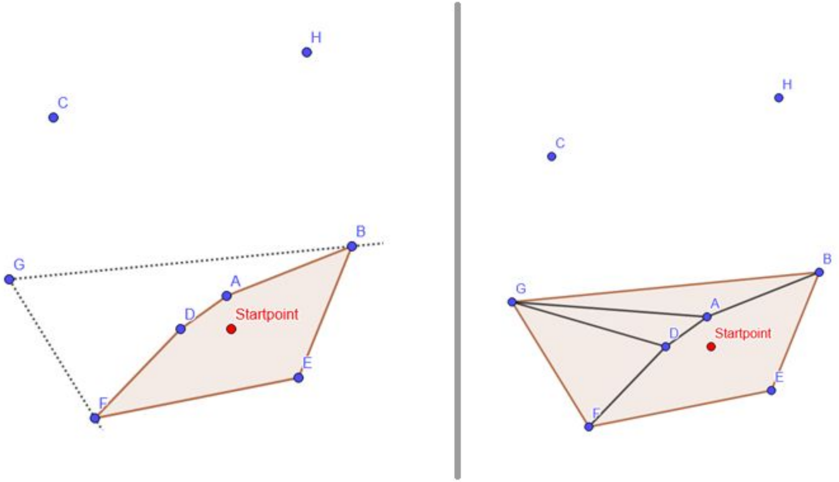
4. Convex Waves

Sort by distance to *startpoint* $\rightarrow Q$

First three points in $Q \rightarrow H$

For each point p in Q:

- Get visible bounds of p to H
- Connect p to all points in-between
- Remove redundant edges
- Update H to the convex hull



[A,D,E,B,F,G,C,H]

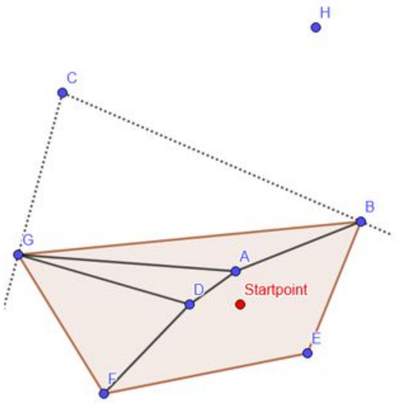
4. Convex Waves

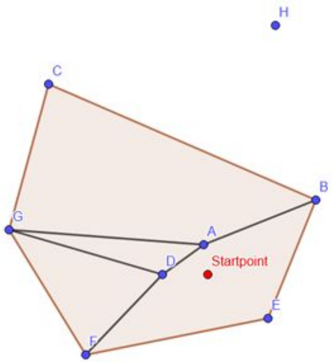
Sort by distance to *startpoint* $\rightarrow Q$

First three points in $Q \rightarrow H$

For each point p in Q:

- Get visible bounds of p to H
- Connect p to all points in-between
- Remove redundant edges
- Update H to the convex hull





[A,D,E,B,F,G,C,H]

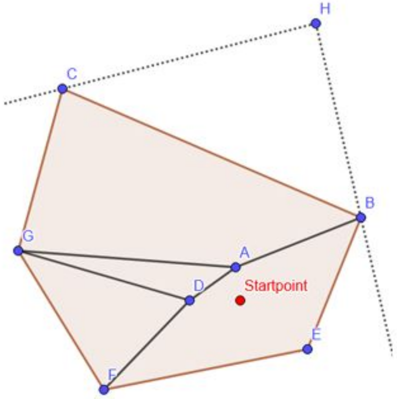
4. Convex Waves

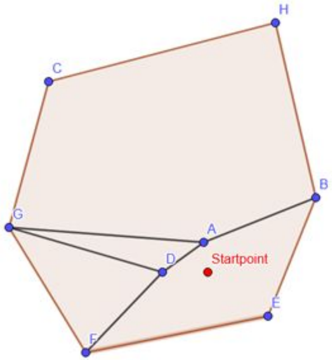
Sort by distance to *startpoint* $\rightarrow Q$

First three points in $Q \rightarrow H$

For each point p in Q:

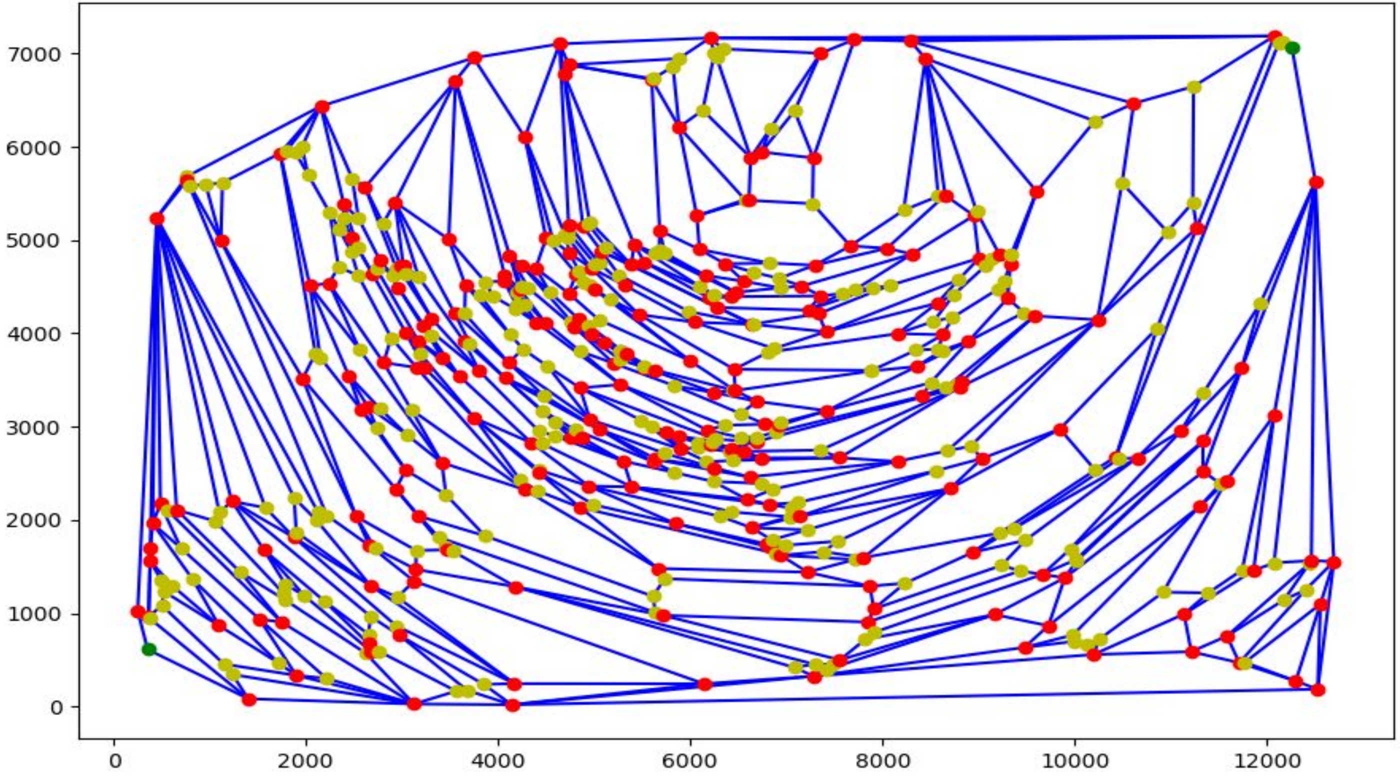
- Get visible bounds of p to H
- Connect p to all points in-between
- Remove redundant edges
- Update H to the convex hull





[A,D,E,B,F,G,C,H]

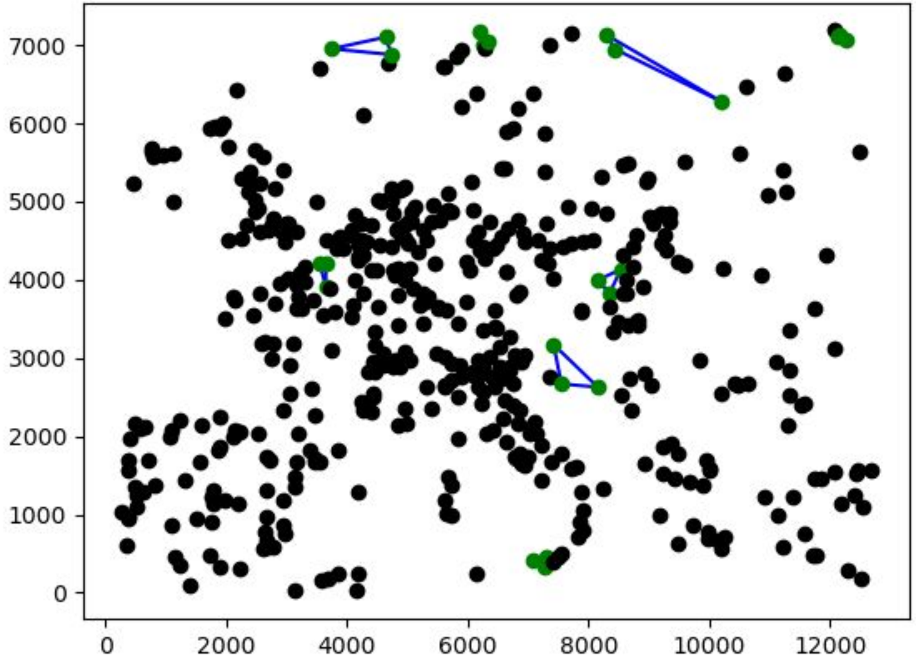
4. Convex Waves



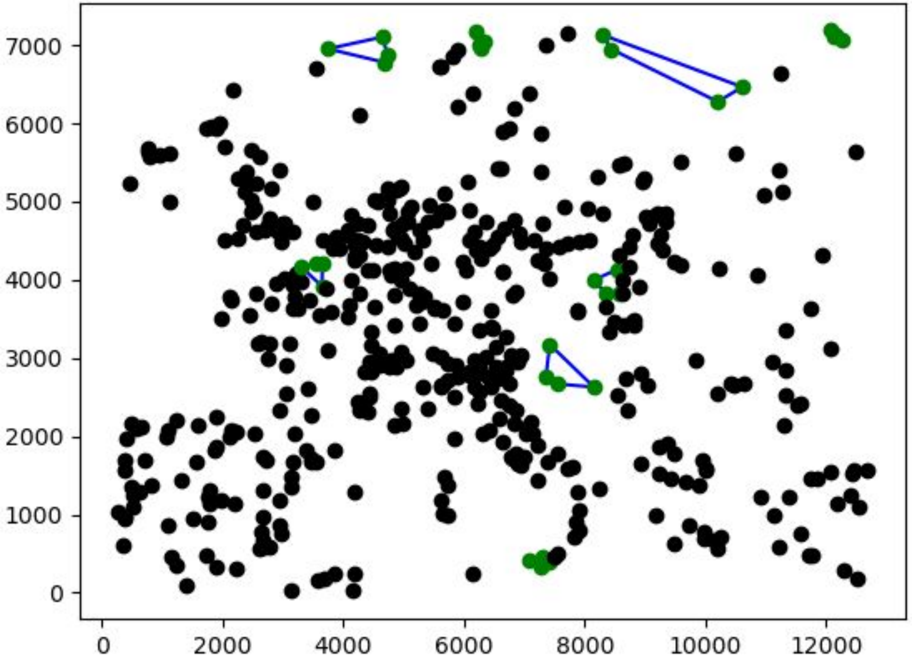
5. Merged Convex Waves

- Perform a convex wave for each startpoint
- Merge two waves on collision

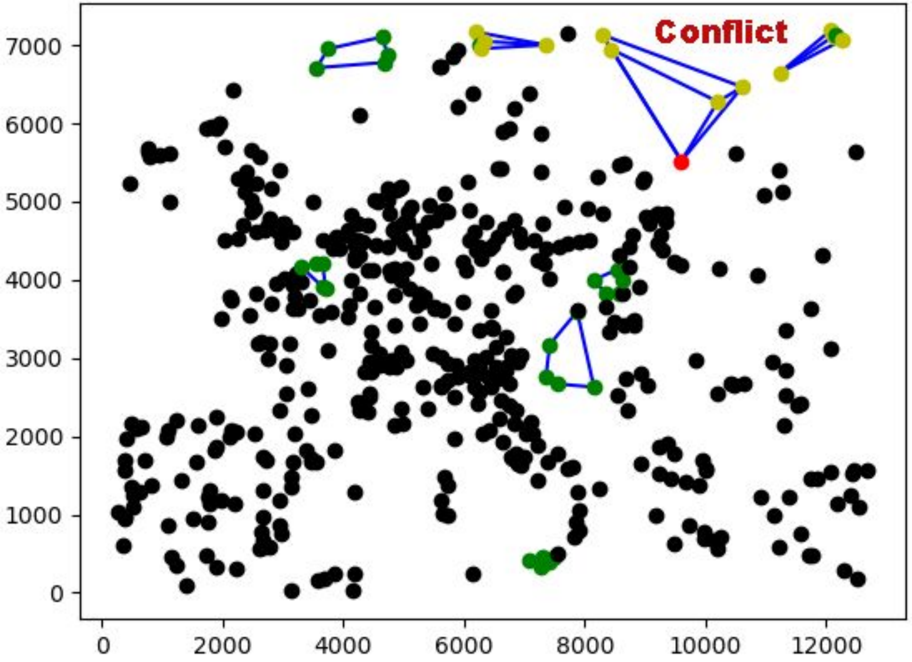
5. Merged Convex Waves



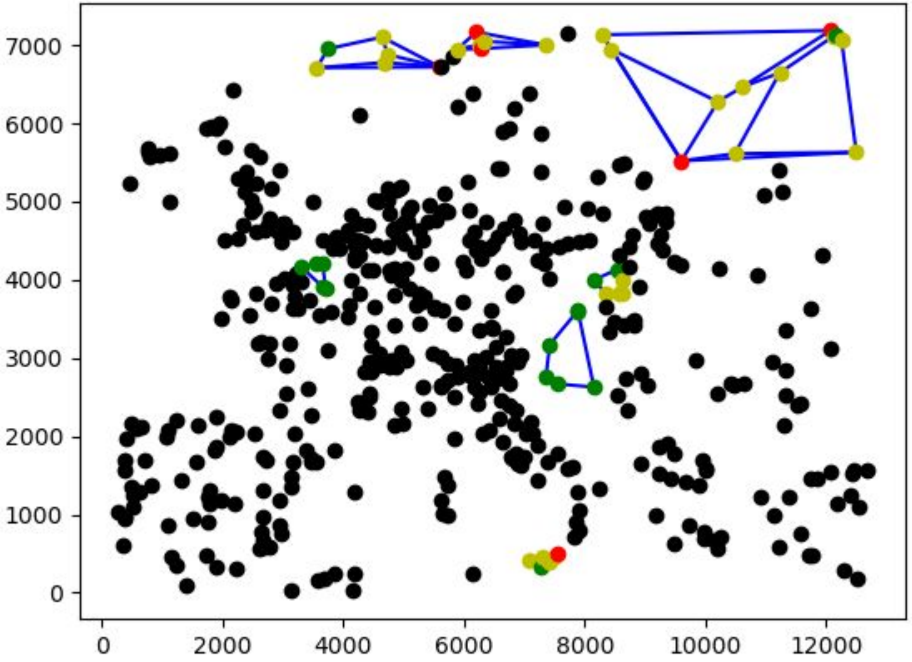
5. Merged Convex Waves



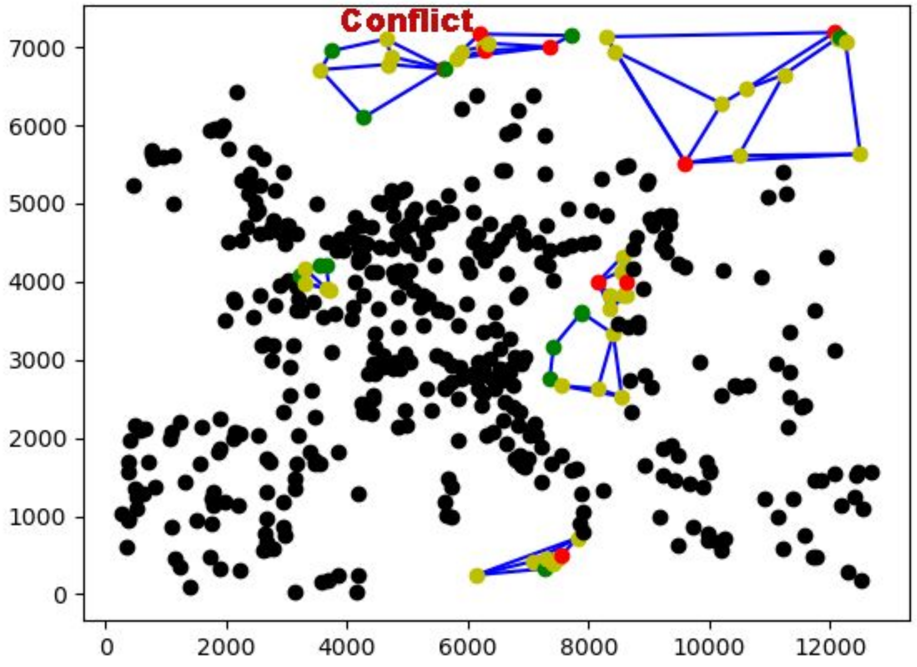
5. Merged Convex Waves



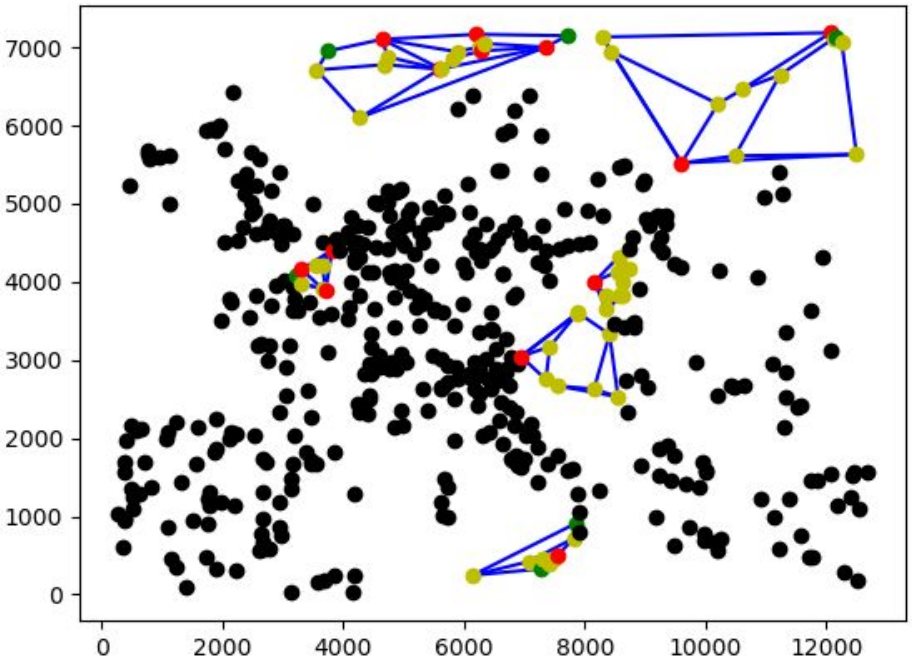
5. Merged Convex Waves



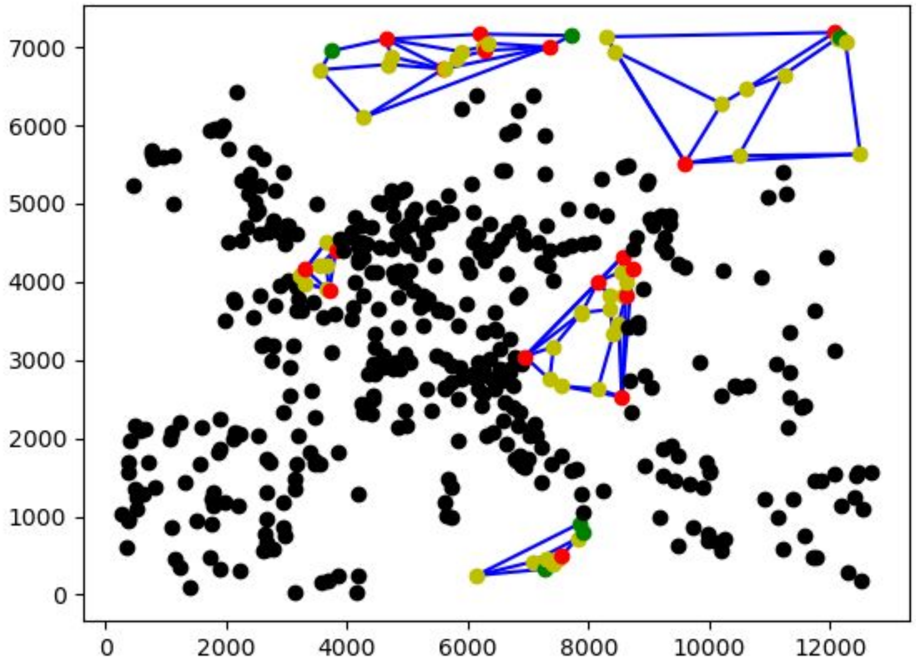
5. Merged Convex Waves



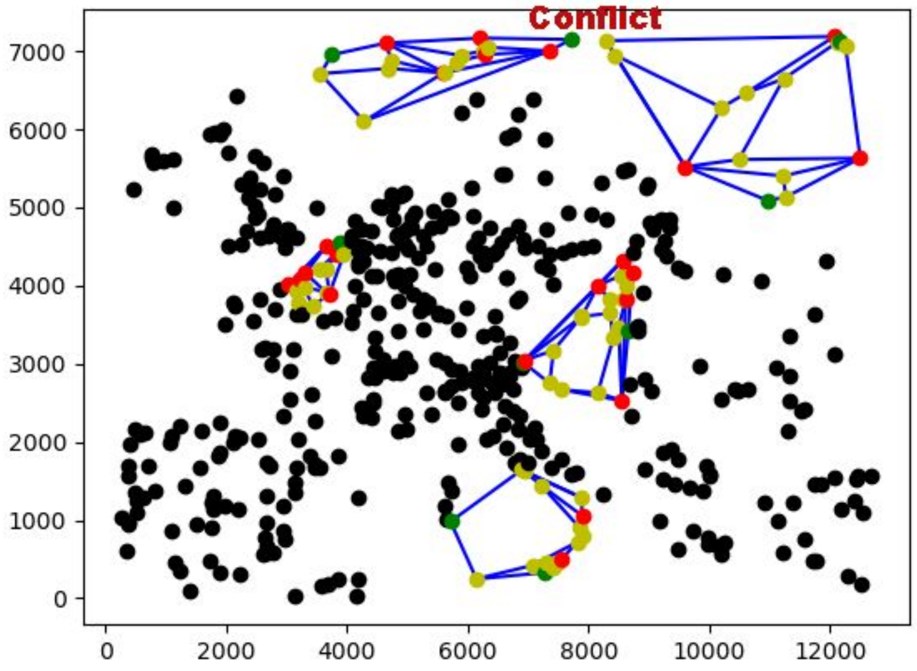
5. Merged Convex Waves



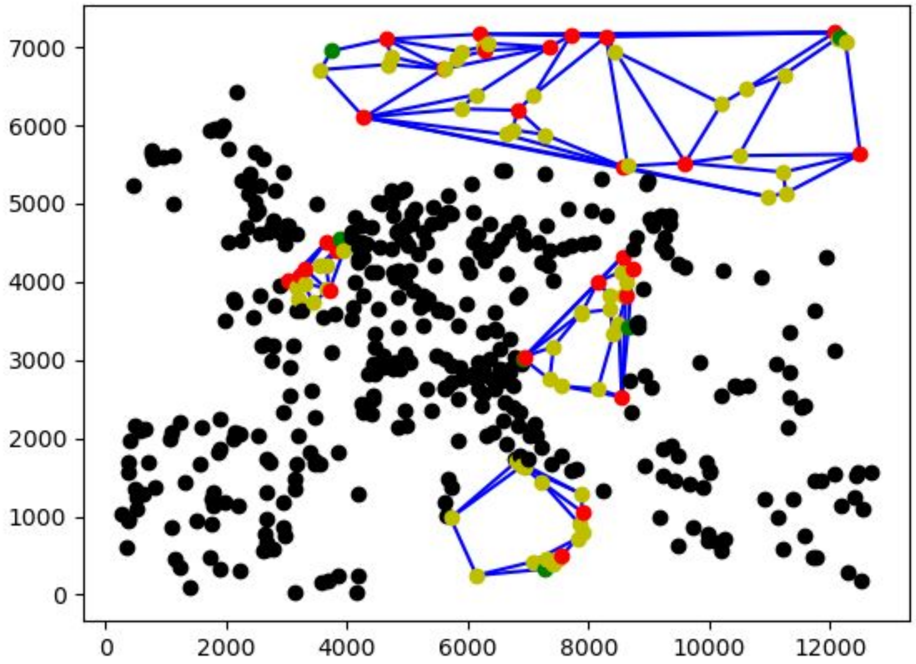
5. Merged Convex Waves



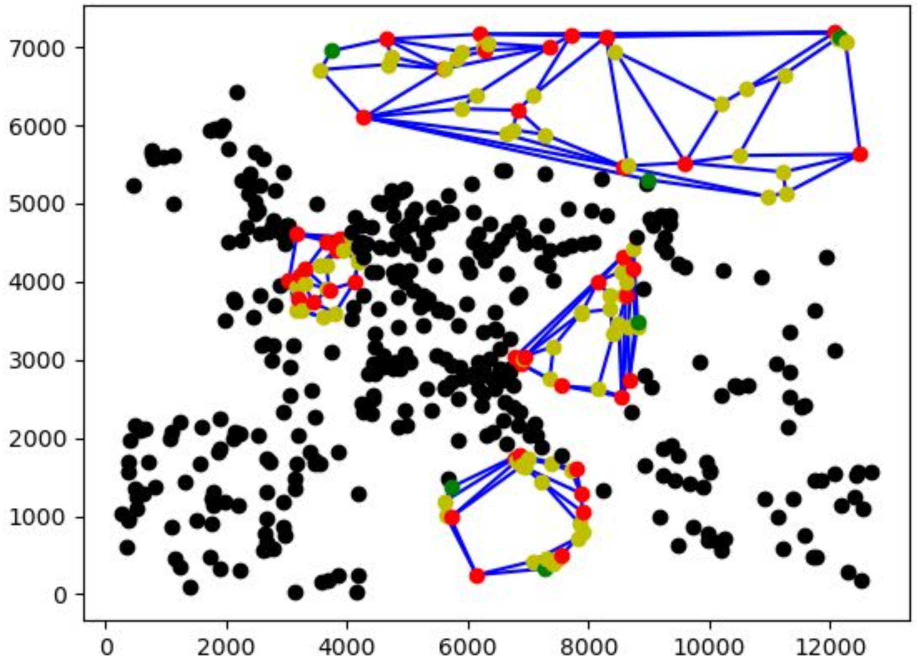
5. Merged Convex Waves



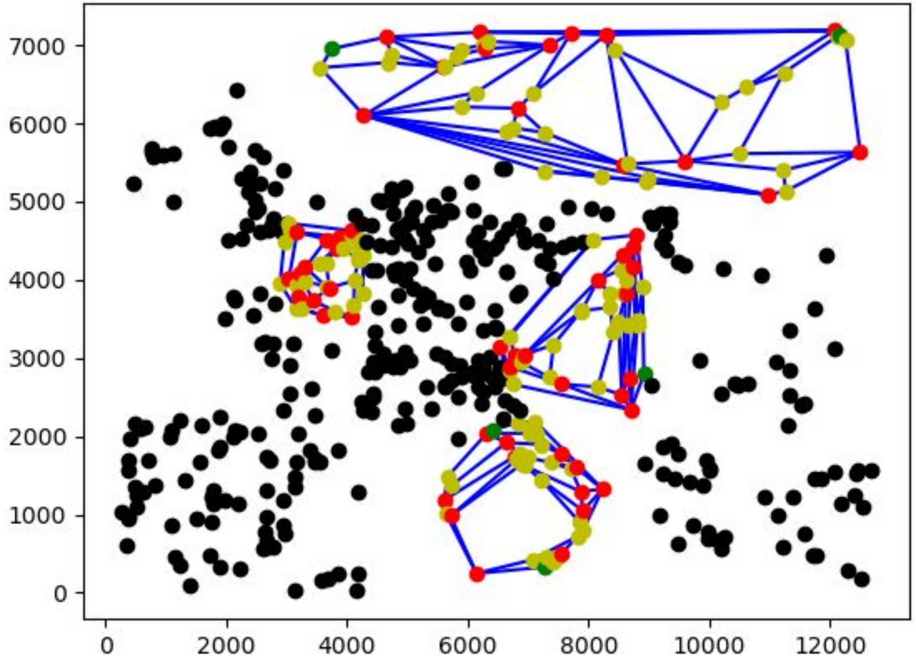
5. Merged Convex Waves



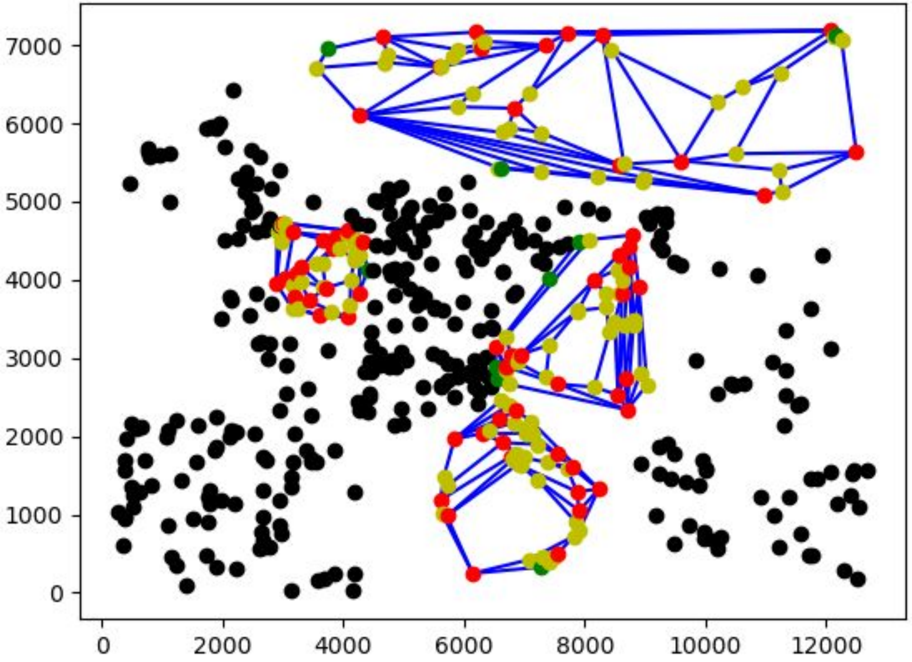
5. Merged Convex Waves



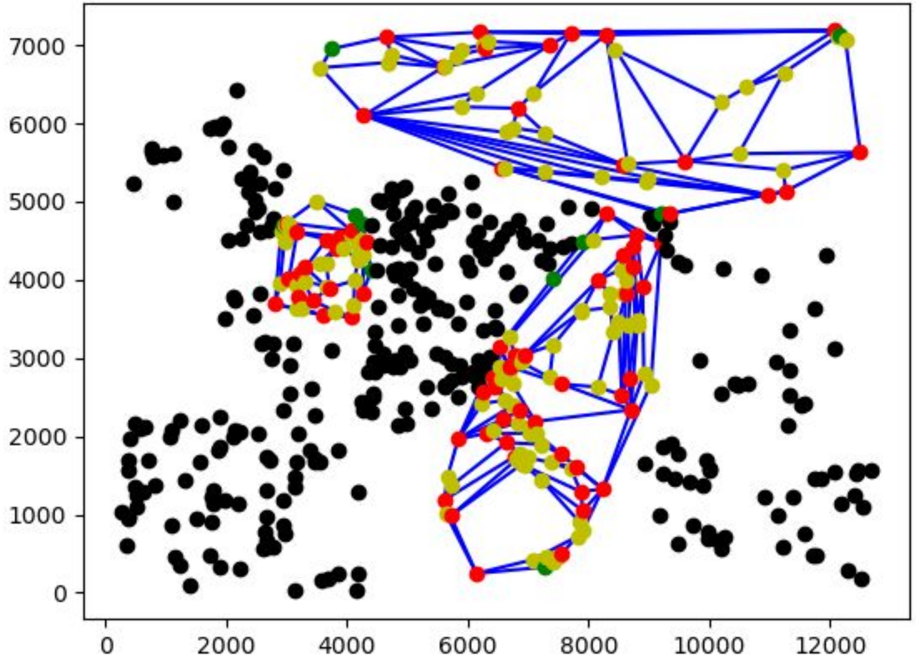
5. Merged Convex Waves



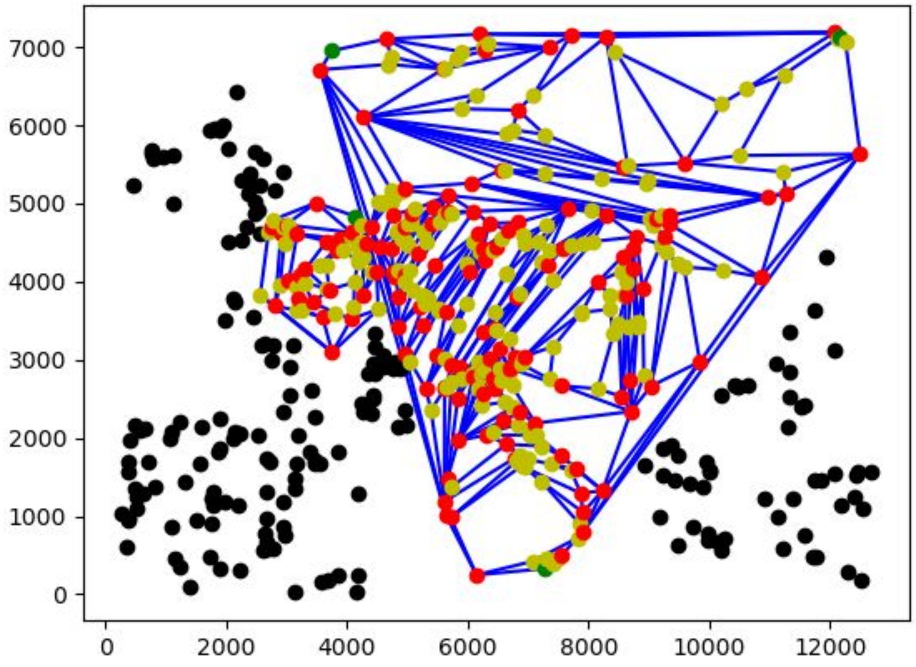
5. Merged Convex Waves



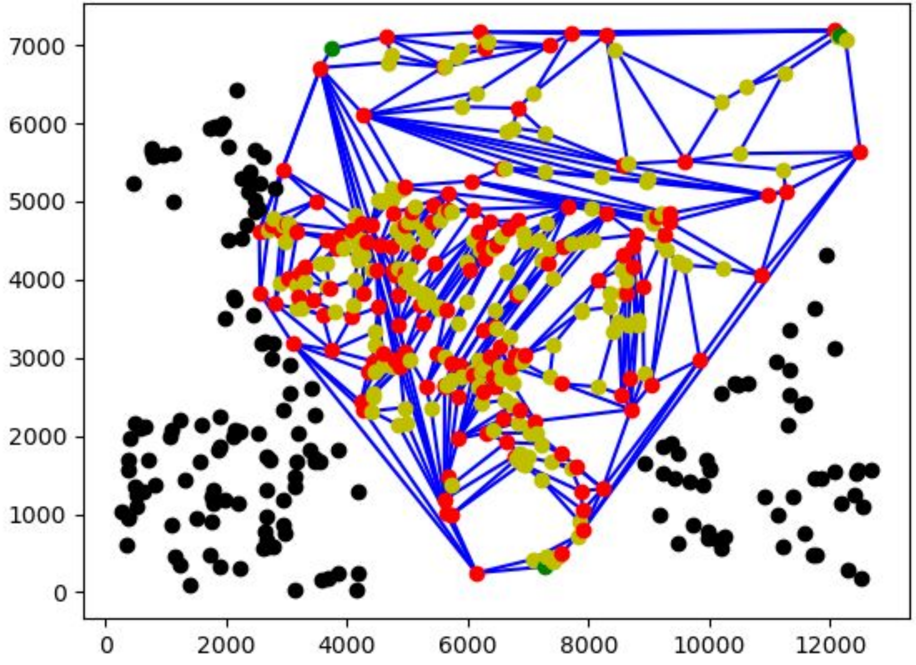
5. Merged Convex Waves



5. Merged Convex Waves

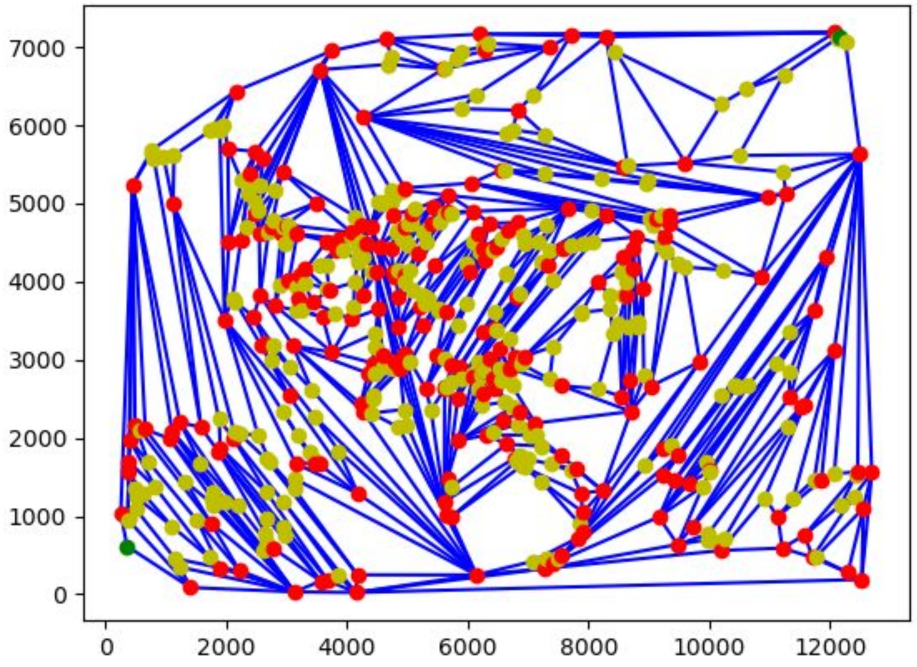


5. Merged Convex Waves



Freie Universität Berlin

5. Merged Convex Waves



5. Merged Convex Waves

- Good results in starting areas, poor results everywhere else
- Merged instances lead to stretched polygons and long edges
- Convex hulls broken during merge need to be triangulated
- Produces more edges than other algorithms on almost all instances

→ *Approach discarded*

6. Pass-Based Algorithm

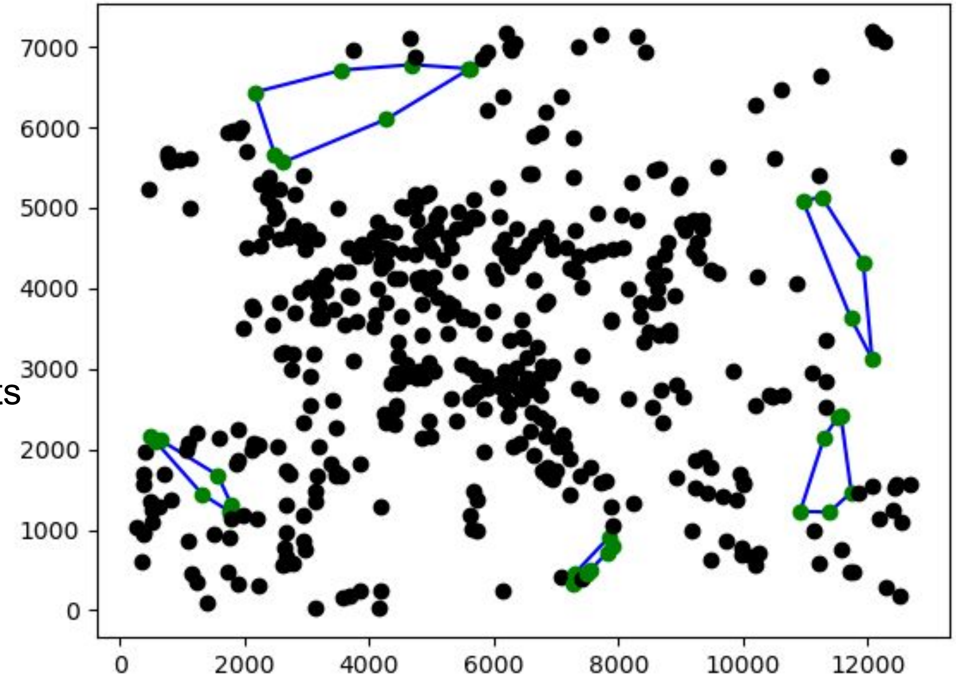
6. Pass-Based Algorithm

- Perform a set of independent passes
- Prioritize areas around startpoints
- No complex merging step required
- Waves constrained to a single convex polygon

6. Pass-Based Algorithm

First Pass: Secure startpoints

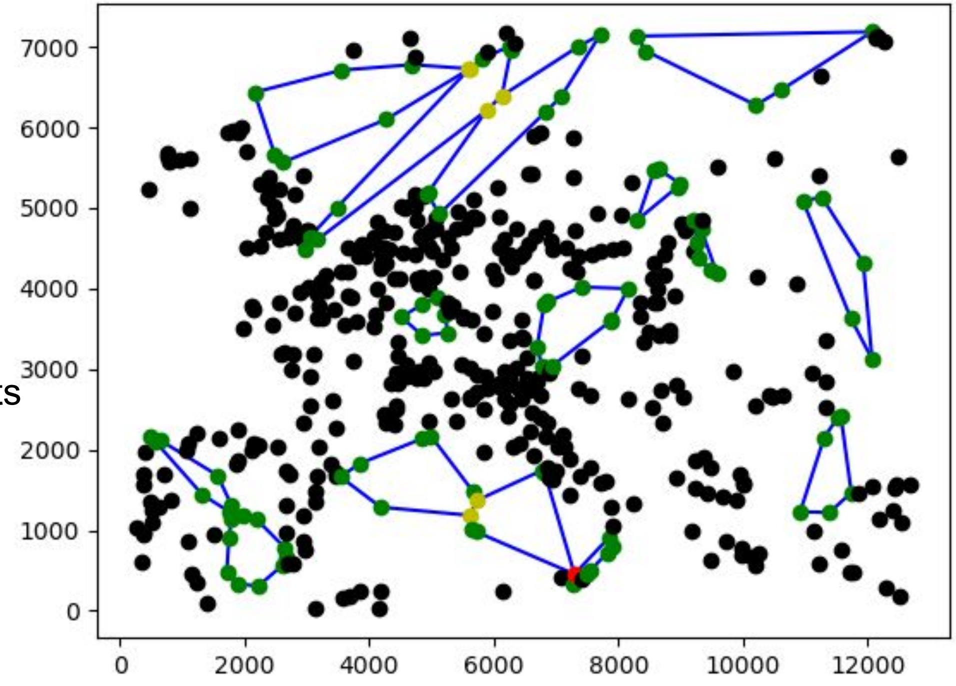
- Start a convex wave at each startpoint
- Better startpoints have higher priority
- Only add a point if...
  - ...it can only see a single edge
  - ...it is not occluded by other points or edges



6. Pass-Based Algorithm

First Pass: Secure startpoints

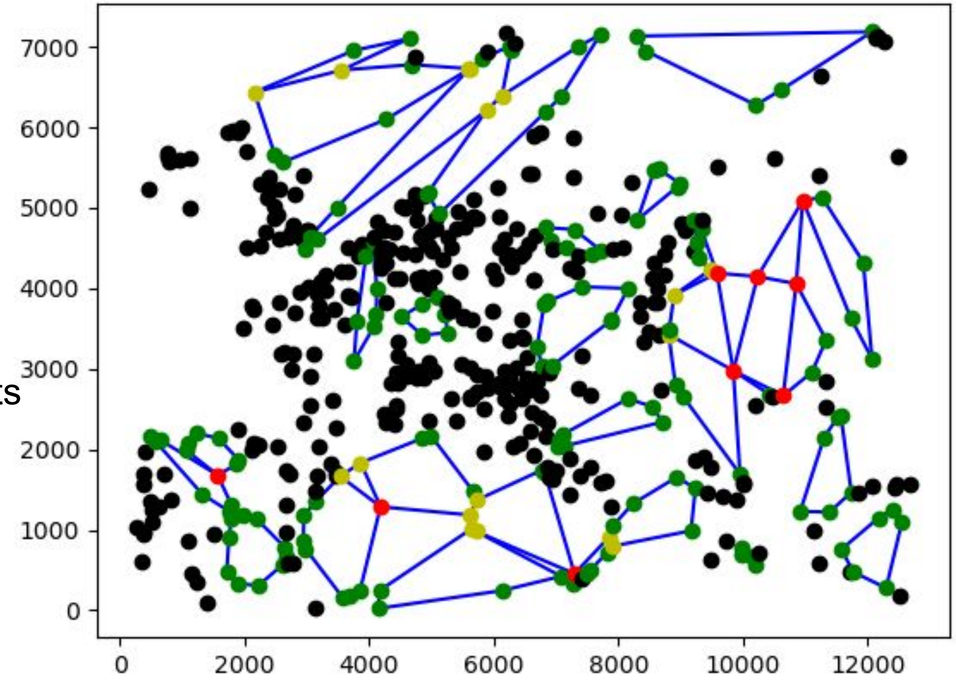
- Start a convex wave at each startpoint
- Better startpoints have higher priority
- Only add a point if...
  - ...it can only see a single edge
  - ...it is not occluded by other points or edges



6. Pass-Based Algorithm

First Pass: Secure startpoints

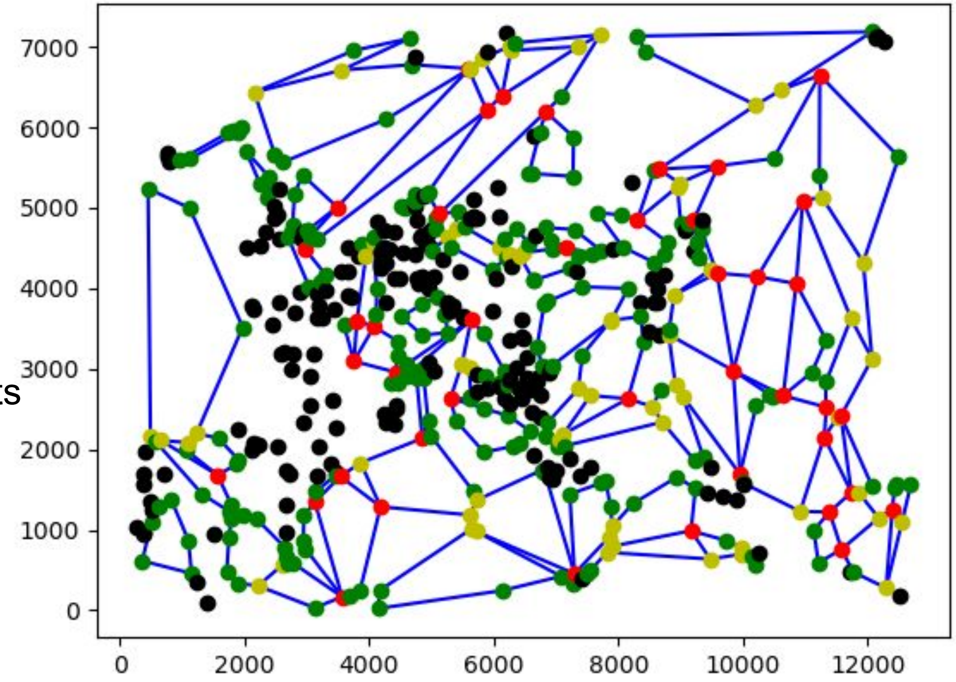
- Start a convex wave at each startpoint
- Better startpoints have higher priority
- Only add a point if...
  - ...it can only see a single edge
  - ...it is not occluded by other points or edges



6. Pass-Based Algorithm

First Pass: Secure startpoints

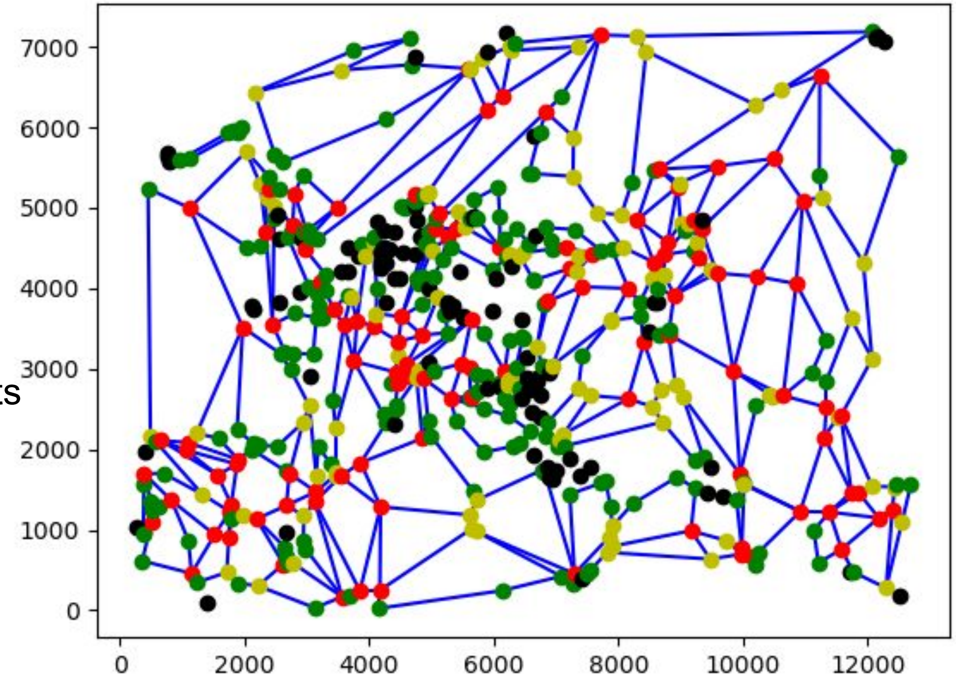
- Start a convex wave at each startpoint
- Better startpoints have higher priority
- Only add a point if...
  - ...it can only see a single edge
  - ...it is not occluded by other points or edges



6. Pass-Based Algorithm

First Pass: Secure startpoints

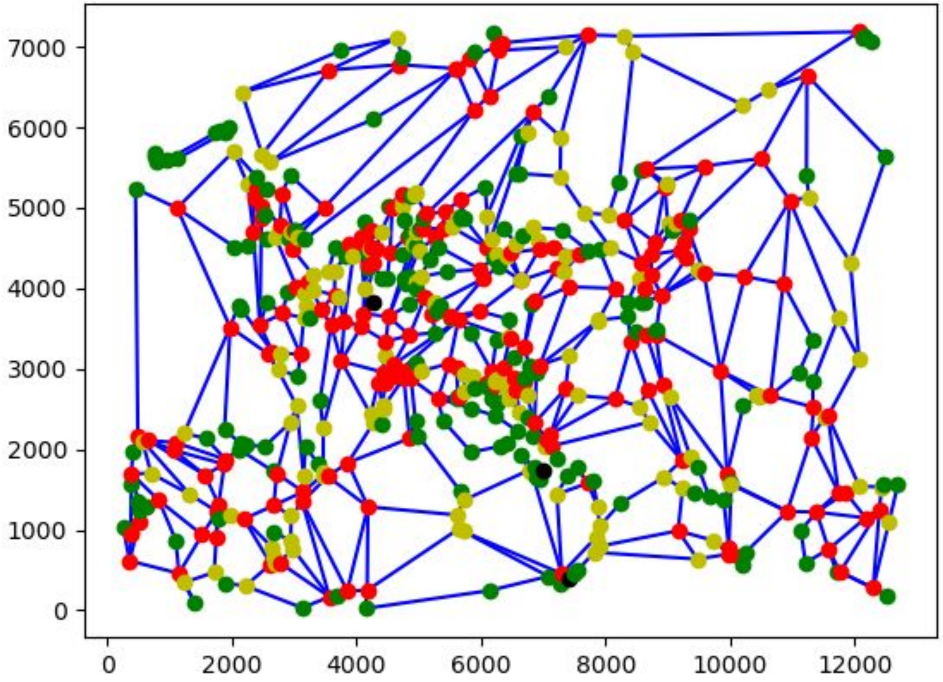
- Start a convex wave at each startpoint
- Better startpoints have higher priority
- Only add a point if...
  - ...it can only see a single edge
  - ...it is not occluded by other points or edges



6. Pass-Based Algorithm

Second Pass: Gather stray points

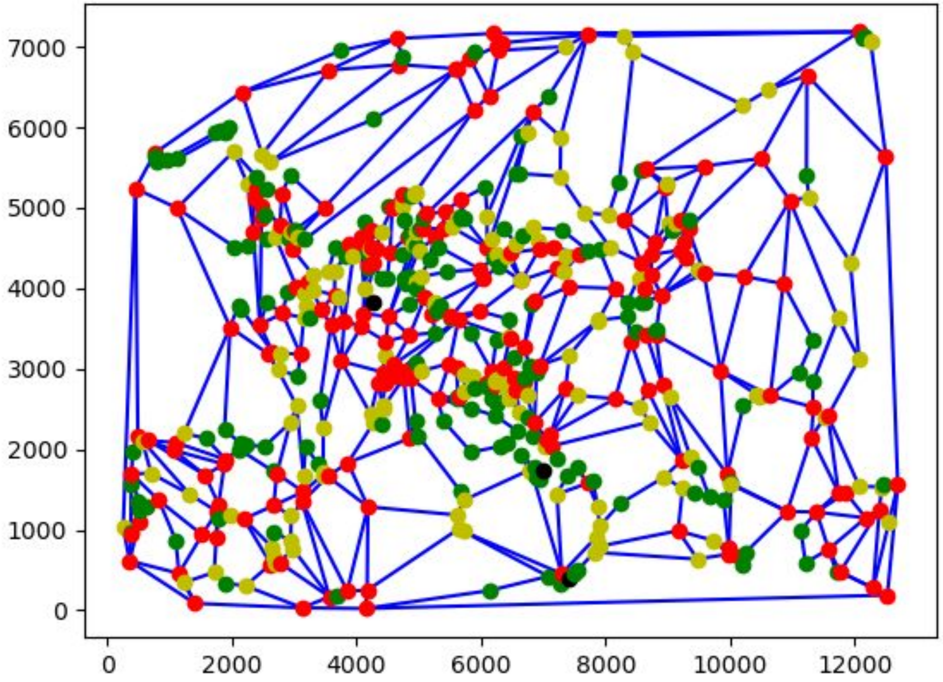
- Start a convex wave at each remaining stray vertex



6. Pass-Based Algorithm

Intermediate Pass: Convex Hull

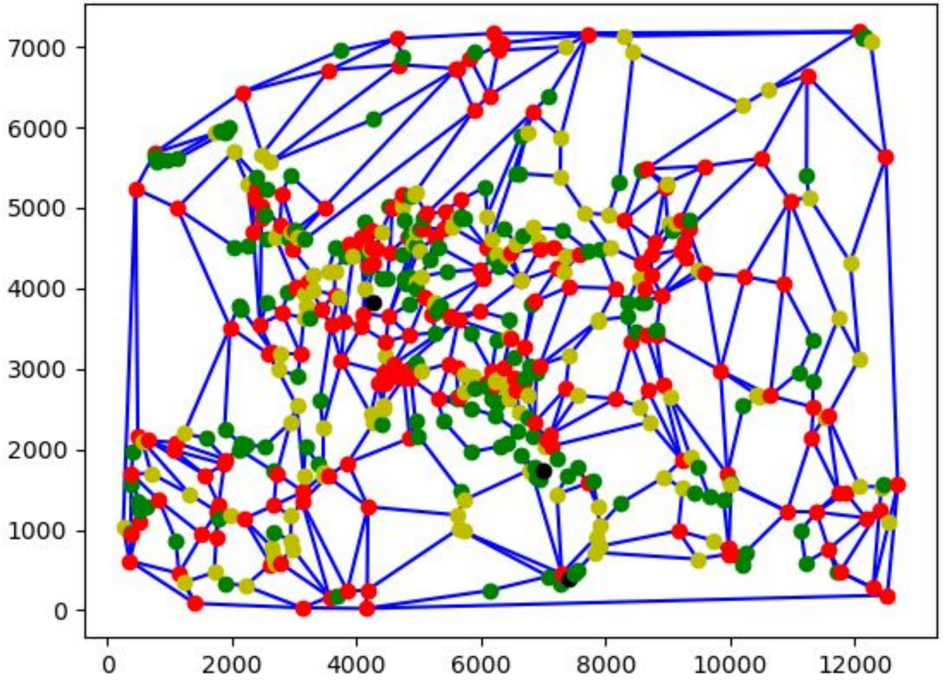
- Incorporate convex hull



6. Pass-Based Algorithm

Intermediate Pass: Integrate islands

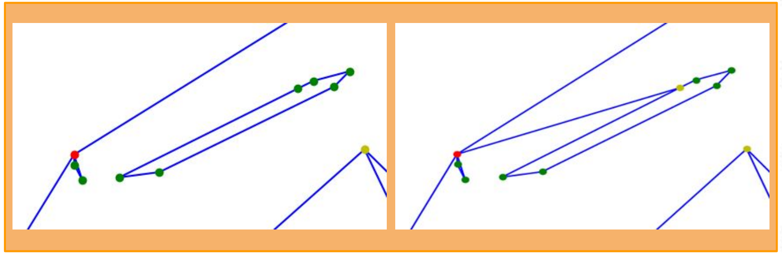
- Find islands via DFS on leftmost vertex
- Connect each to their surrounding face

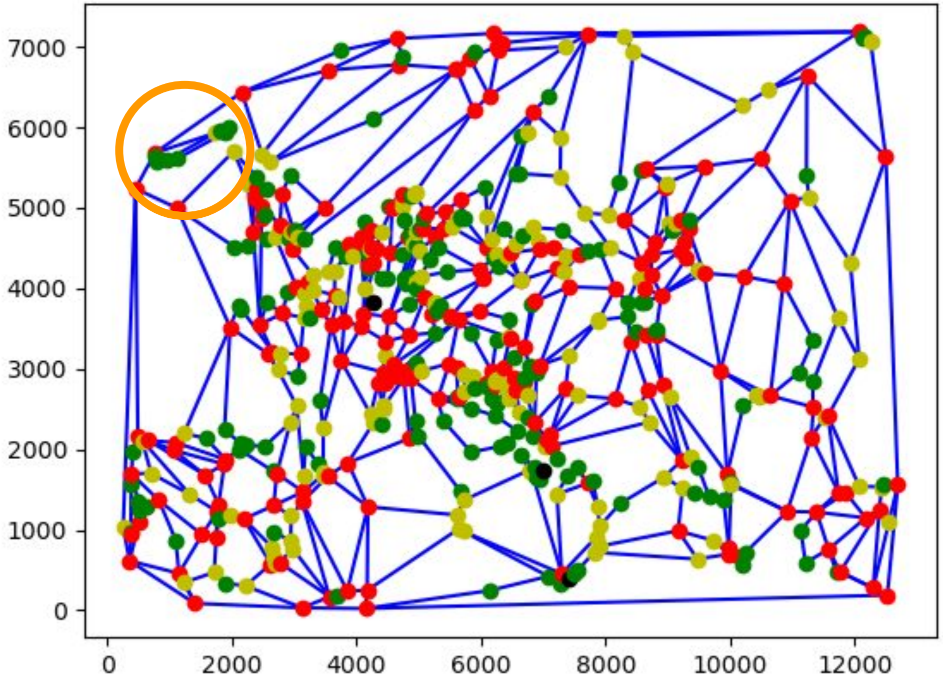


6. Pass-Based Algorithm

Intermediate Pass: Integrate islands

- Find islands via DFS on leftmost vertex
- Connect each to their surrounding face

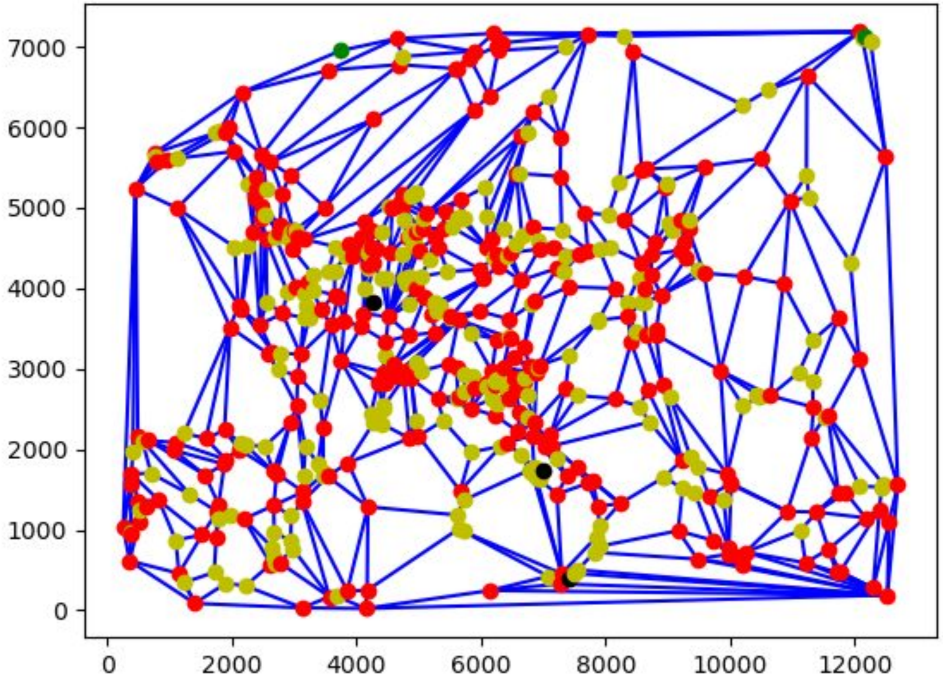




6. Pass-Based Algorithm

Third Pass: Resolve inflexes

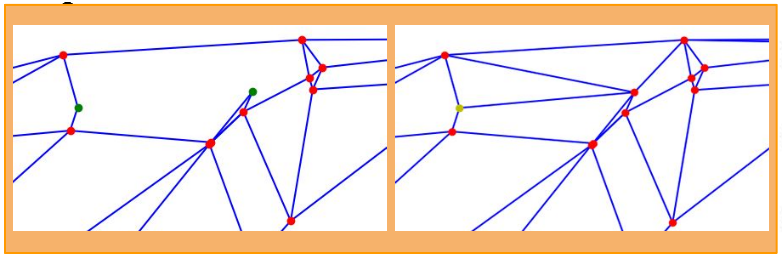
- Find all inflex vertices
- Resolve these by connecting them to 1-2 opposing vertices

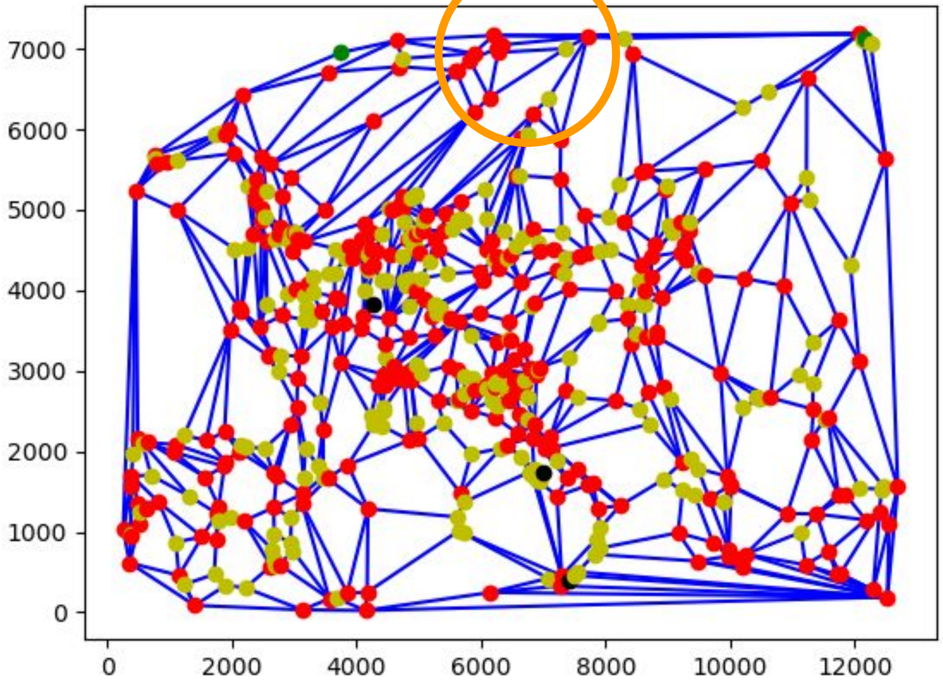


6. Pass-Based Algorithm

Third Pass: Resolve inflexes

- Find all inflex vertices
- Resolve these by connecting them to 1-2 opposing vertices

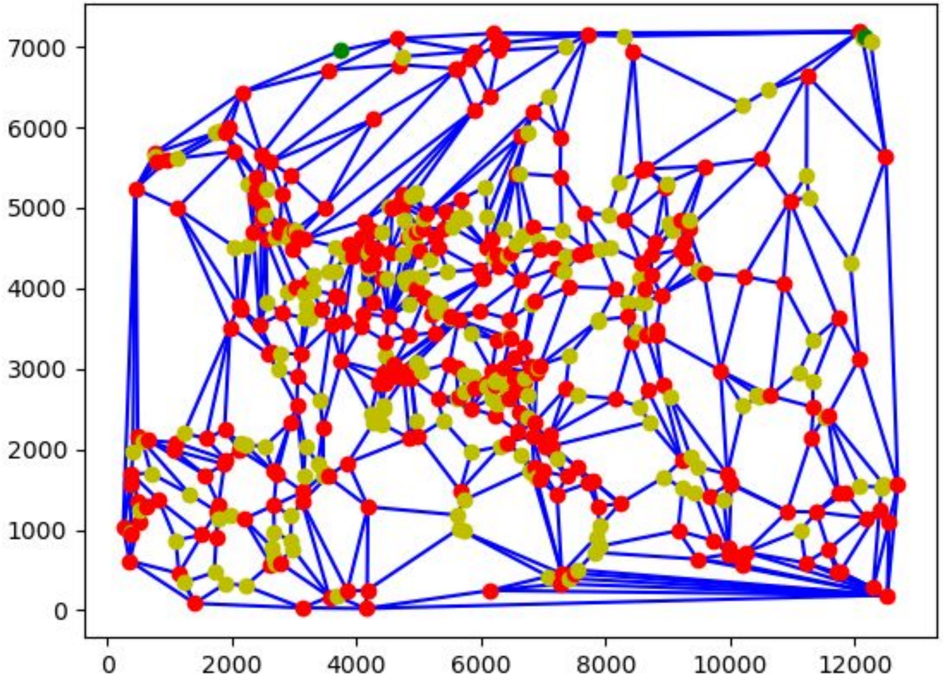




6. Pass-Based Algorithm

Intermediate Pass: Integrate stray points

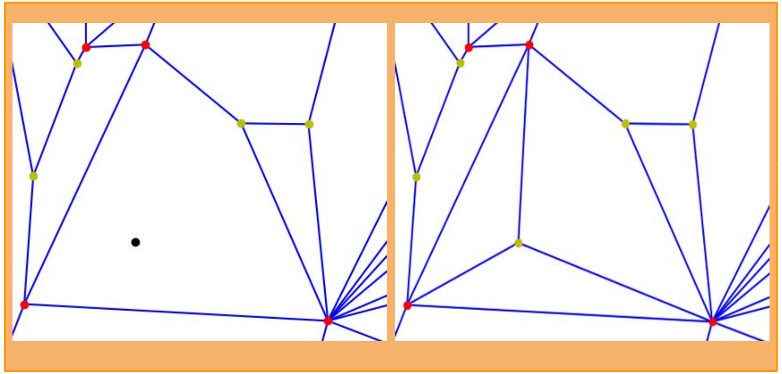
- Find any remaining stray points as well as their respective convex face
- Incorporate them by iterating around the surrounding face

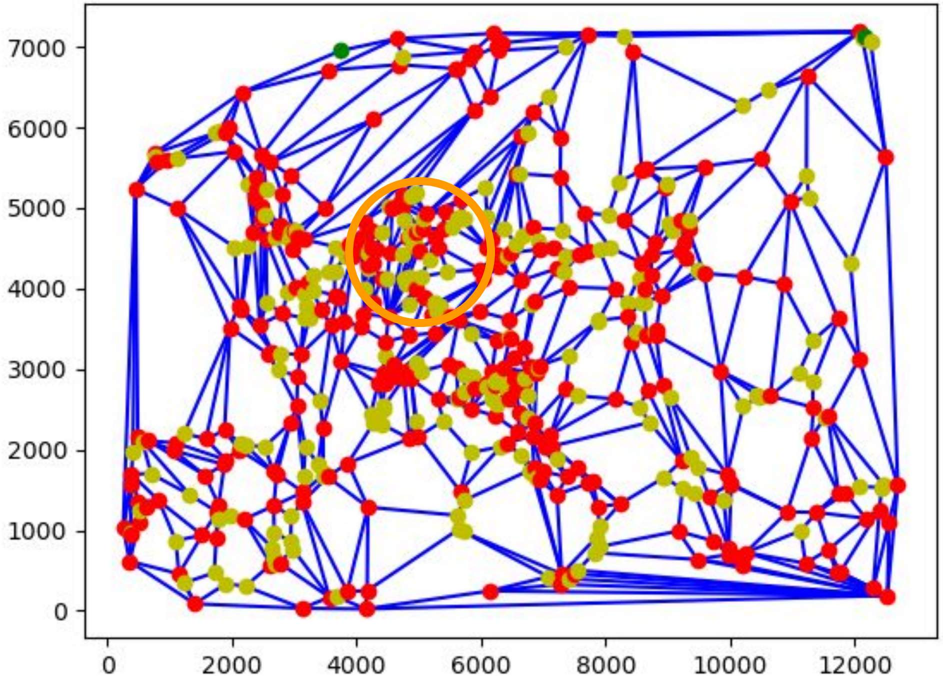


6. Pass-Based Algorithm

Intermediate Pass: Integrate stray points

- Find any remaining stray points as well as their respective convex face
- Incorporate them by iterating around the surrounding face

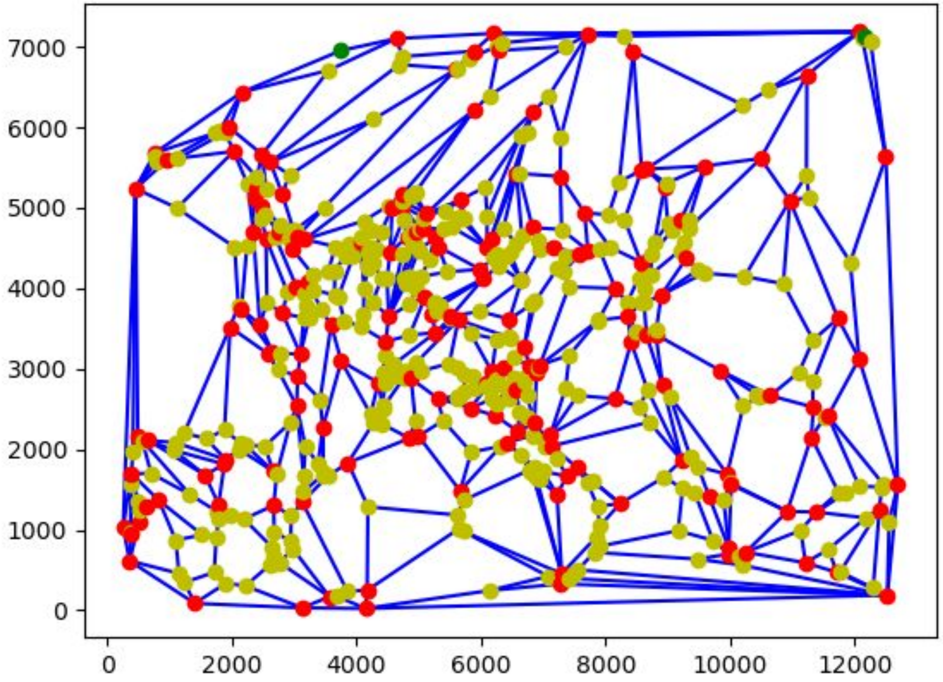




6. Pass-Based Algorithm

Fourth Pass: Clean

- Iterate over all edges and verify that they are required
- Remove the ones that are not

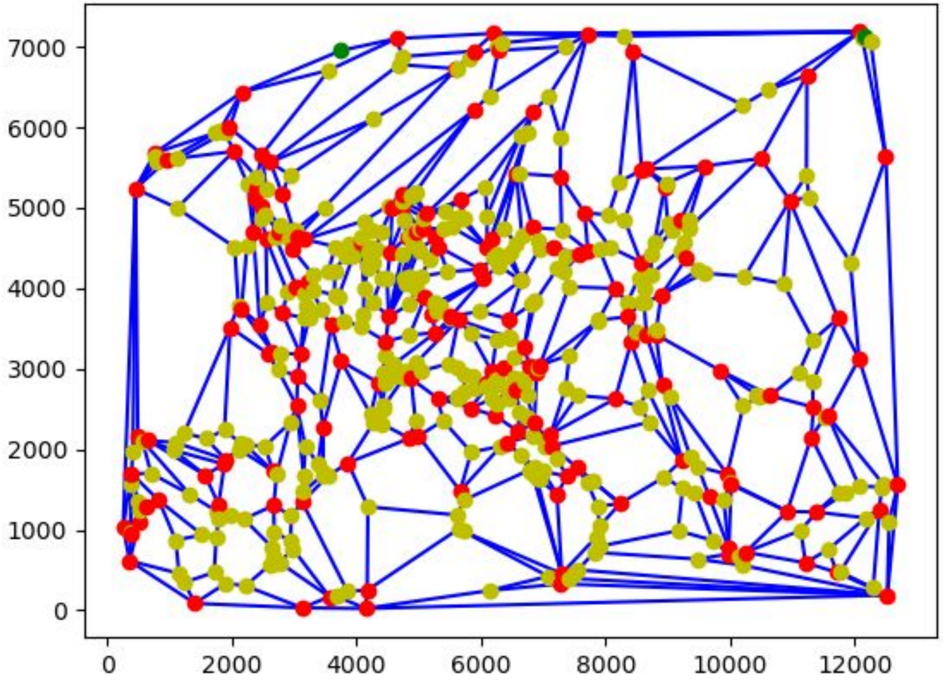


6. Pass-Based Algorithm

Fourth Pass: Clean

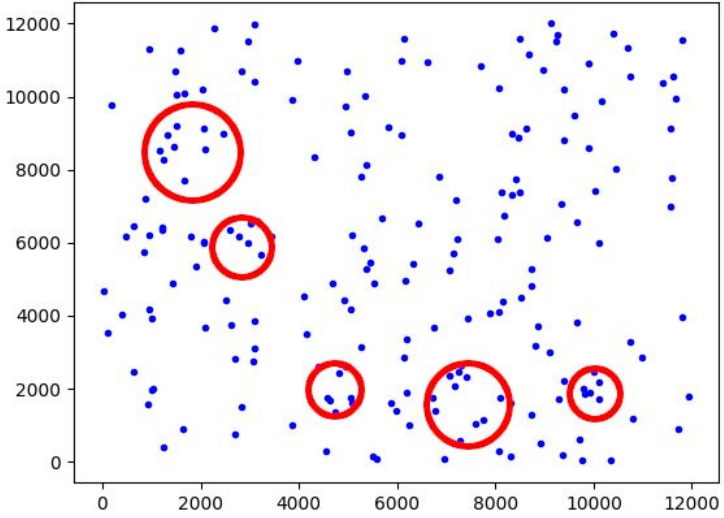
- Iterate over all edges and verify that they are required
- Remove the ones that are not

➡ *How do we chose adequate startpoints?*

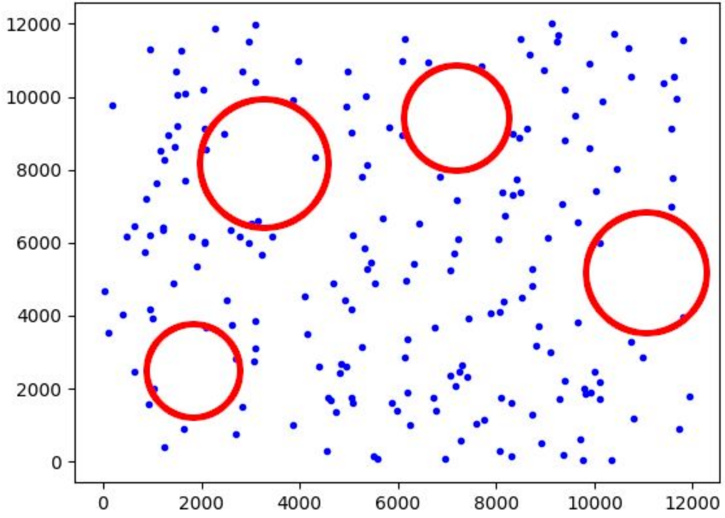


7. Startpoints

7. Startpoints - What are Good Start Points?

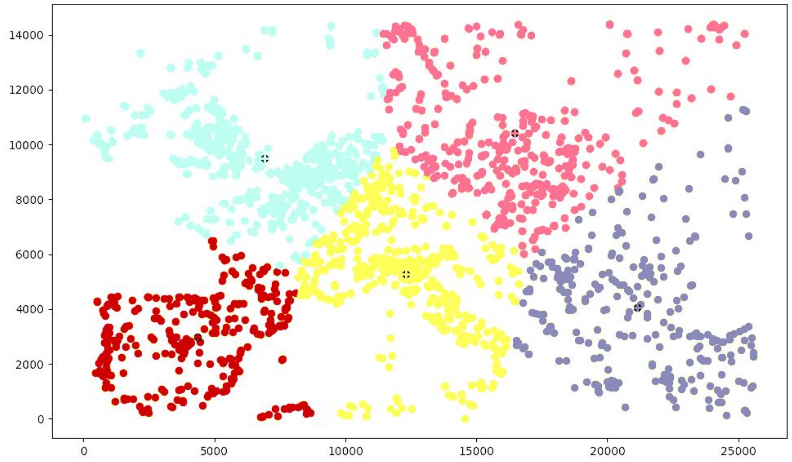


starting within clusters

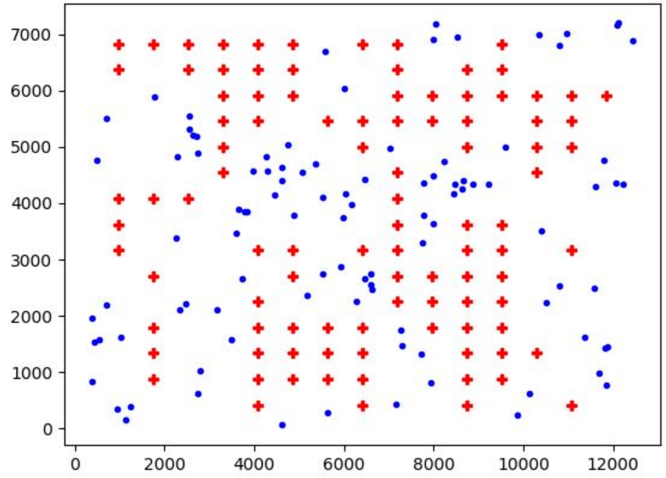


starting in empty spaces

7. Startpoints - Dropped Concepts

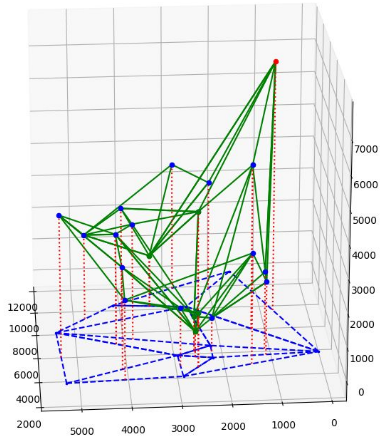


clustering with kmeans



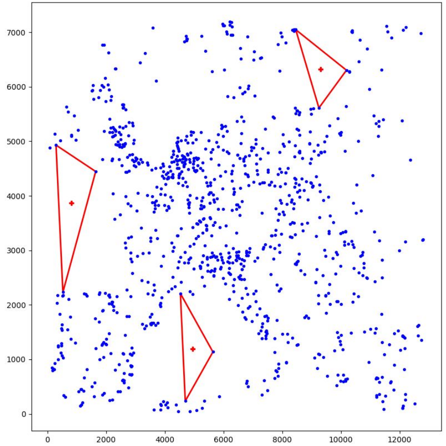
empty spaces with fixed-distance grid

7. Startpoints - Promising Concepts



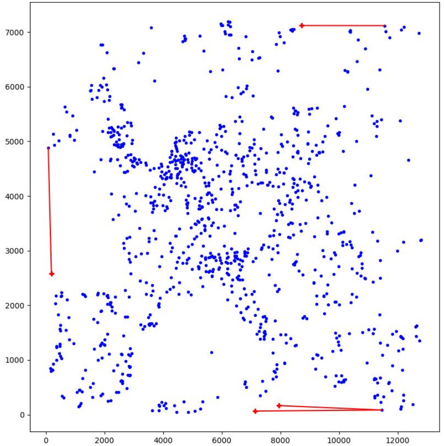
triangulation edge length

$$O(n^2)$$



max area triangle

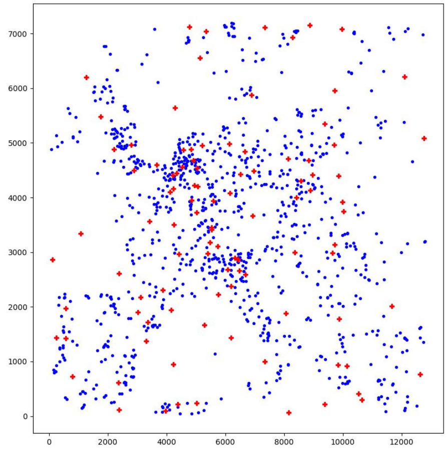
$$O(n \log n)$$



max spanning triangle

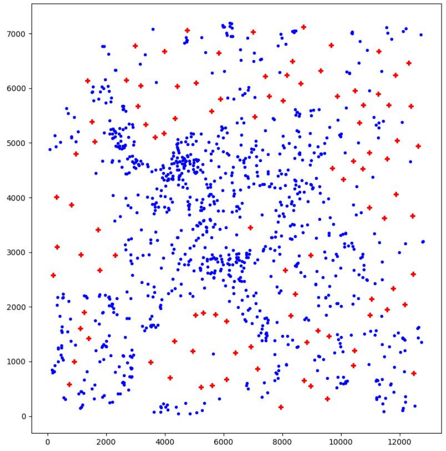
$$O(n \log n)$$

7. Startpoints - Distribution



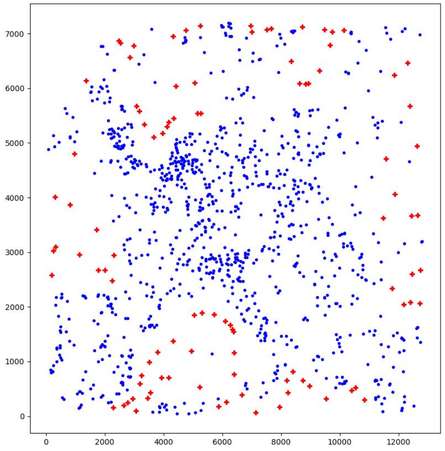
triangulation edge length

$O(n^2)$



max area triangle

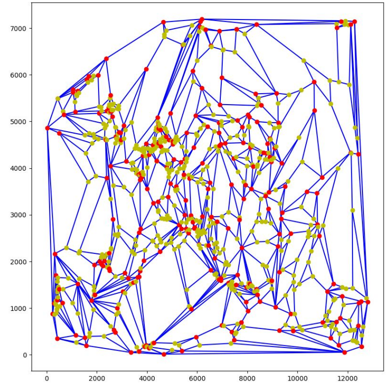
$O(n \log n)$



max spanning triangle

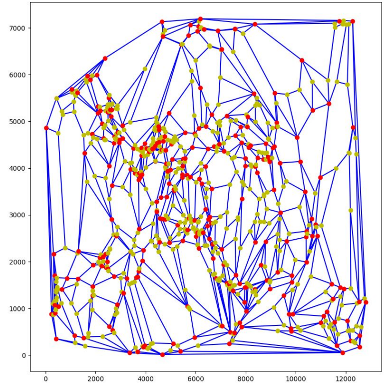
$O(n \log n)$

7. Startpoints - Was It Worth It?



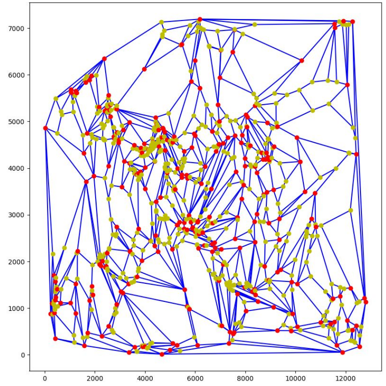
triangulation edge length
1254 edges in solution
2082 start points

$$O(n^2)$$



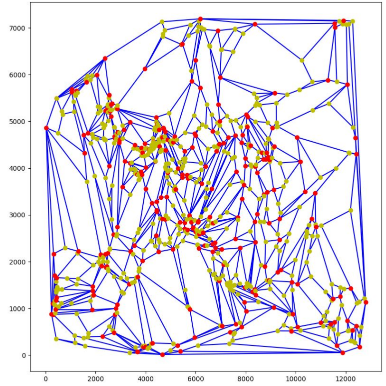
max area triangle
1250 edges in solution
1383 start points

$$O(n \log n)$$



max spanning triangle
1253 edges in solution
1383 start points

$$O(n \log n)$$

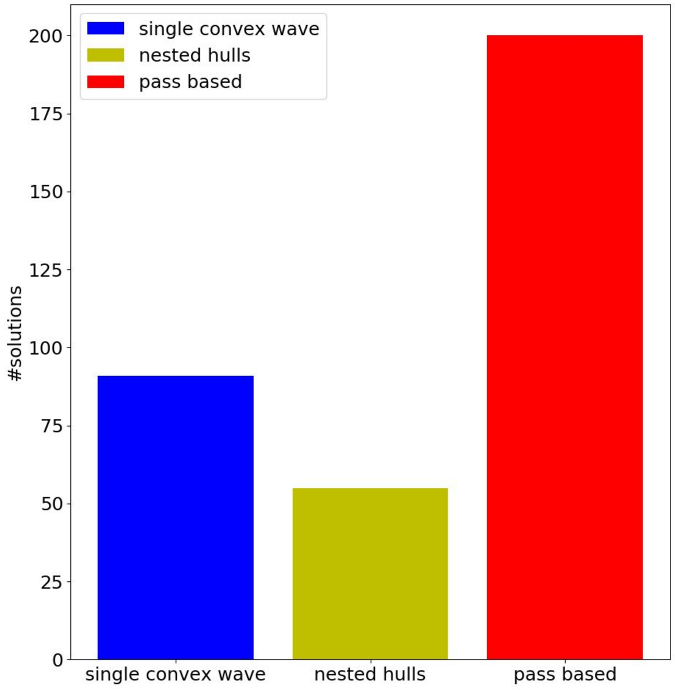


random
1242 edges in solution
4 start points

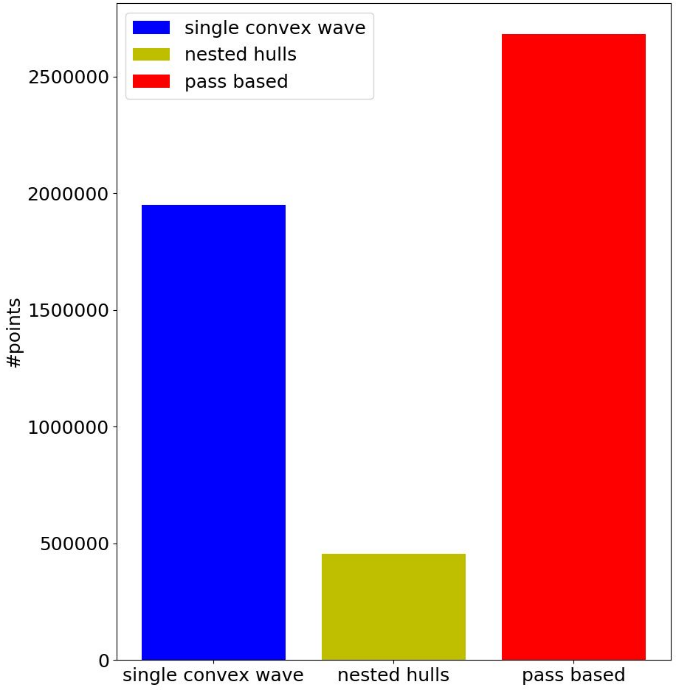
$$O(1)$$

8. Solutions

8. Solutions - Comparing the Algorithms



Solved Instances by Algorithm



Quantity of Points solved by Algorithm

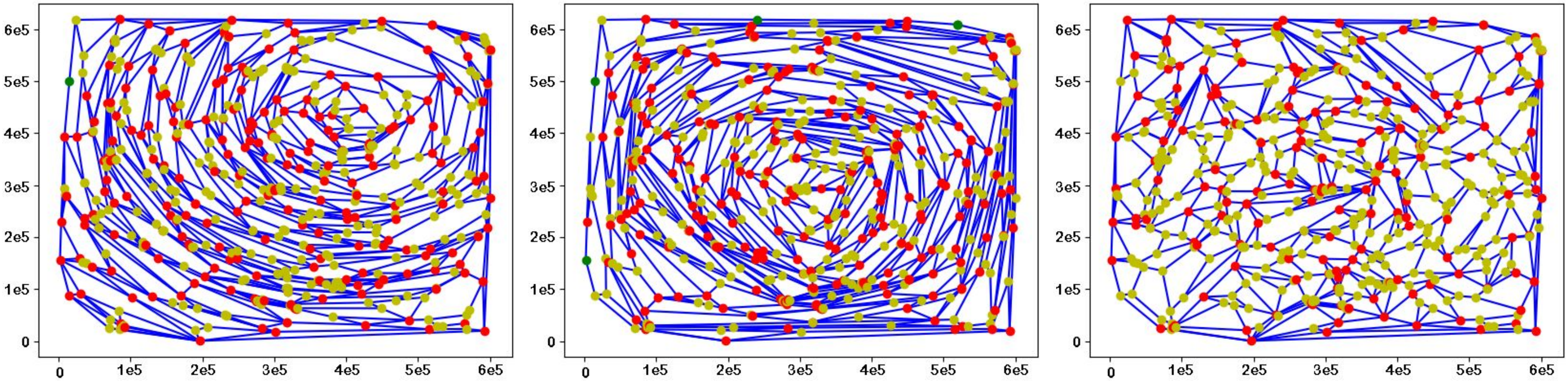
8. Solutions - Score

$$score = \frac{T - A}{T}$$

- $T \triangleq$ #edges in triangulation
- $A \triangleq$ #edges in solution
- $0 < score < 1$
- % of deleted edges from triangulation
- bigger score is better

8. Solutions - Plotting the Algorithms

#points: 500, Triangulation #edges: 1480



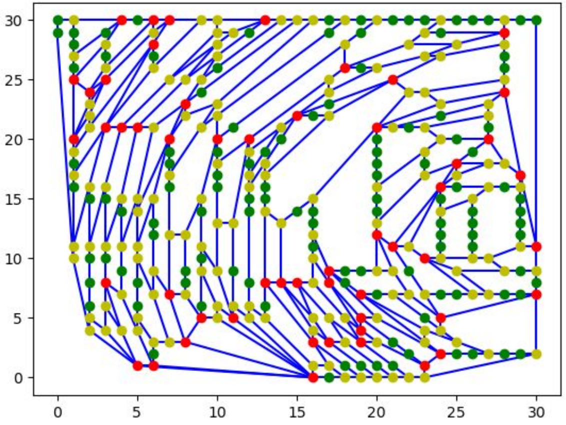
single convex waves (927 edges)
score: 0.37

nested hulls (918 edges)
score: 0.38

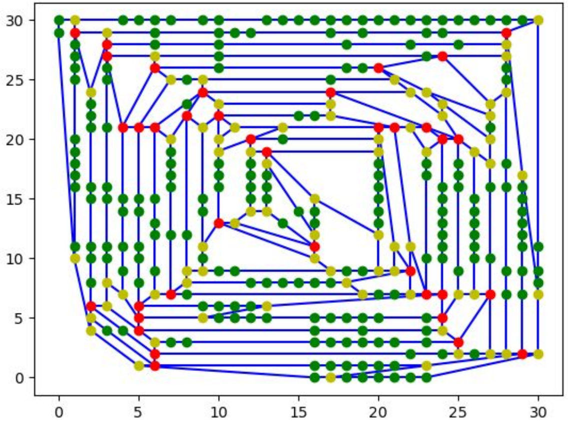
pass based (879 edges)
score: 0.41

8. Solutions - Many Collinear Points

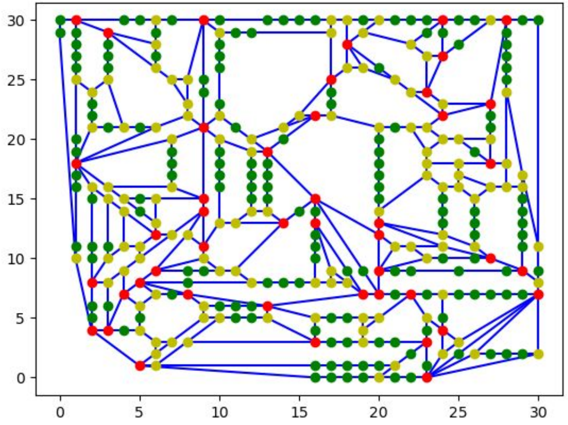
#points: 326, Triangulation #edges: 932



single convex waves (463 edges)
score: 0.5

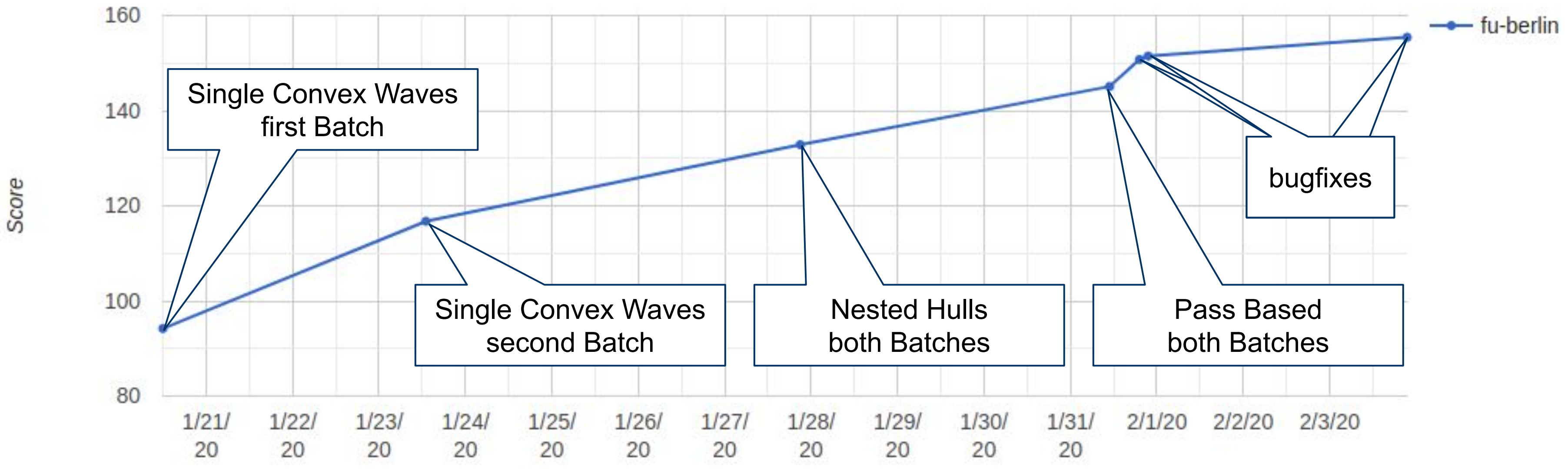


nested hulls (403 edges)
score: 0.57



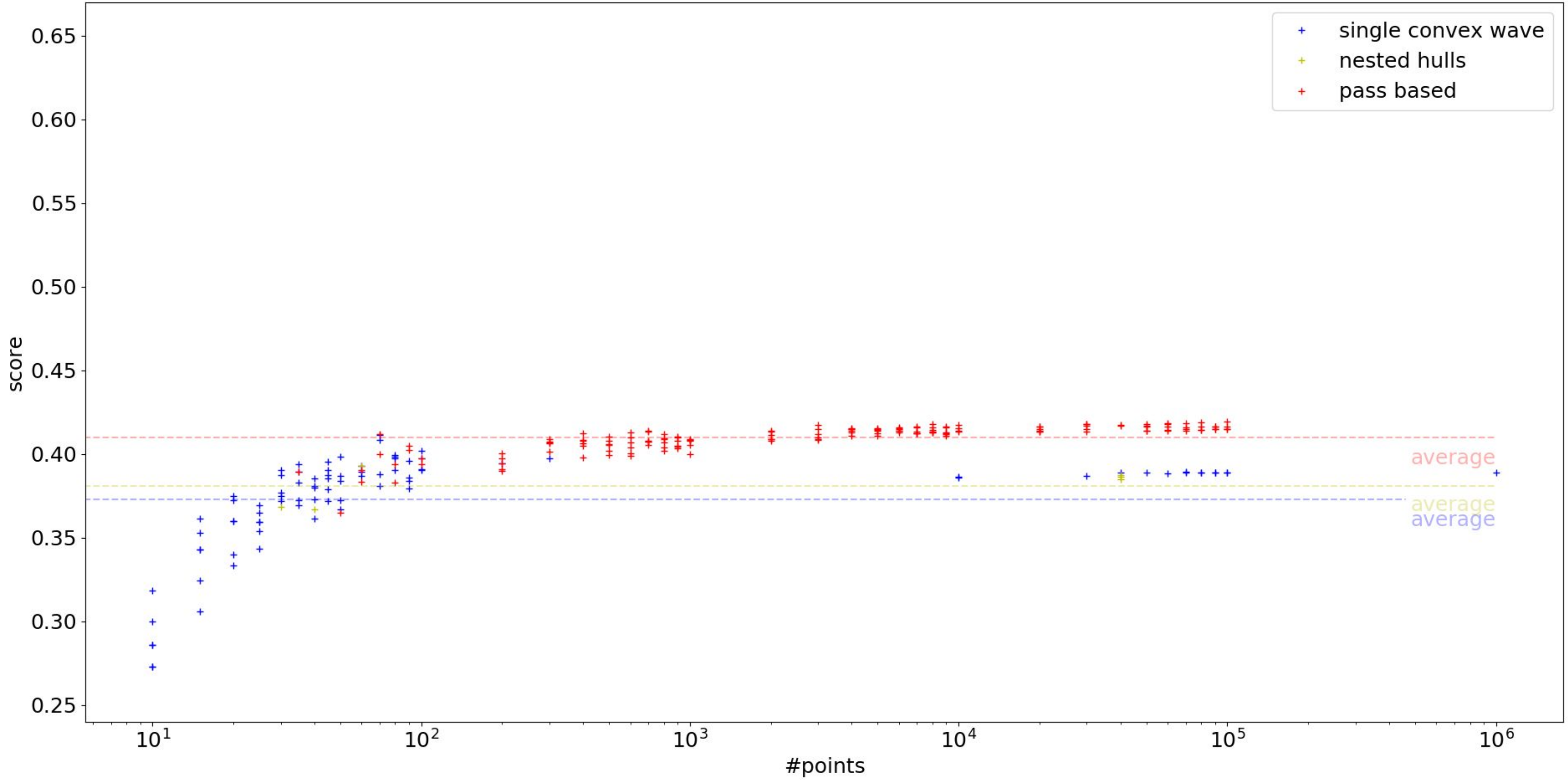
pass based (438 edges)
score: 0.53

8. Solution

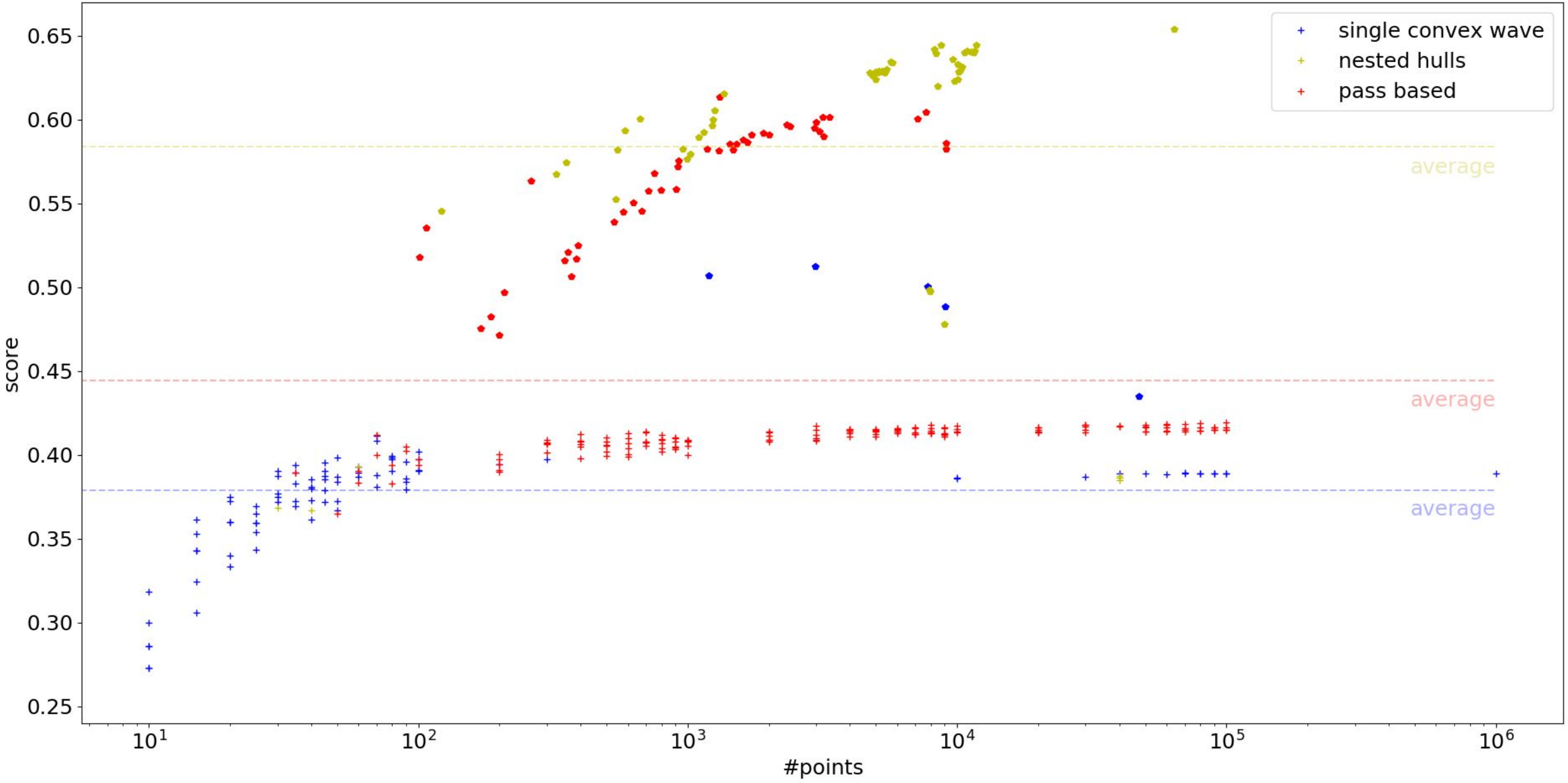


Final score: 155.432 - Instances: 346 - deleted Edges: 44,9%

8. Solutions - Score Distribution



8. Solutions - Score (Many Collinear Points)



Thank You For Your Attention