# Competitive Routing on a Bounded-Degree Plane Spanner

Prosenjit Bose, Rolf Fagerberg, André van Renssen and Sander Verdonschot

Carleton University, University of Southern Denmark

August 4, 2012

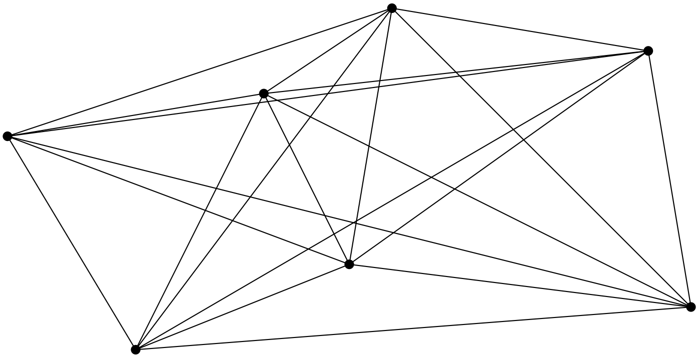
# Geometric Spanners

Given:

- Set of points in the plane

Goal:

- Approximate the complete Euclidean graph



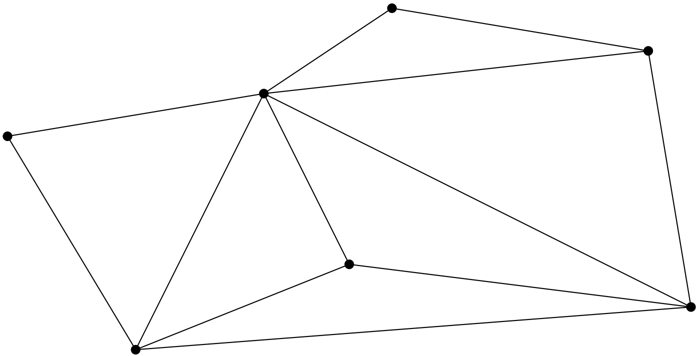
# Geometric Spanners

Given:

- Set of points in the plane

Goal:

- Approximate the complete Euclidean graph



# Geometric Spanners

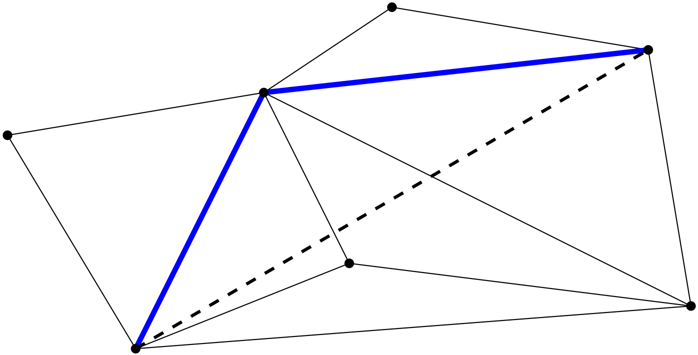
Given:

- Set of points in the plane

Goal:

- Approximate the complete Euclidean graph

shortest path $\leq k \cdot$ Euclidean distance



# Competitive Routing

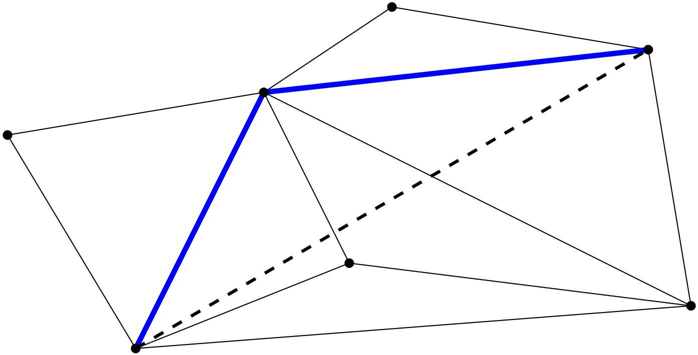
Given:

- Geometric spanner
- Using only local information

Goal:

- Find a short path between any two vertices

path length $\leq r \cdot$ Euclidean distance



# Previous Work

Half-$\theta_6$-graph
(Bonichon et al. 2010)

# Previous Work

Bounded-degree variants
(Bonichon et al. 2010)

Half-$\theta_6$-graph
(Bonichon et al. 2010)

Competitive routing
(Bose et al. 2012)

# Previous Work

Bounded-degree variants
(Bonichon et al. 2010)

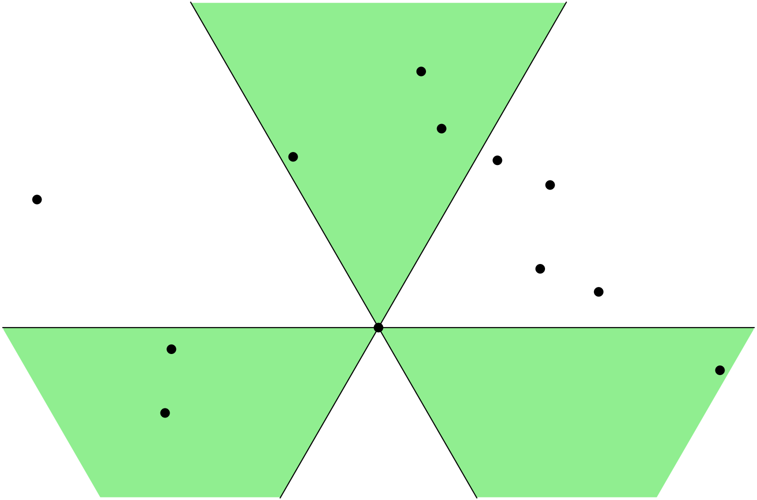
Half-$\theta_6$-graph
(Bonichon et al. 2010)

Competitive routing on
bounded-degree variants
(This result)

Competitive routing
(Bose et al. 2012)

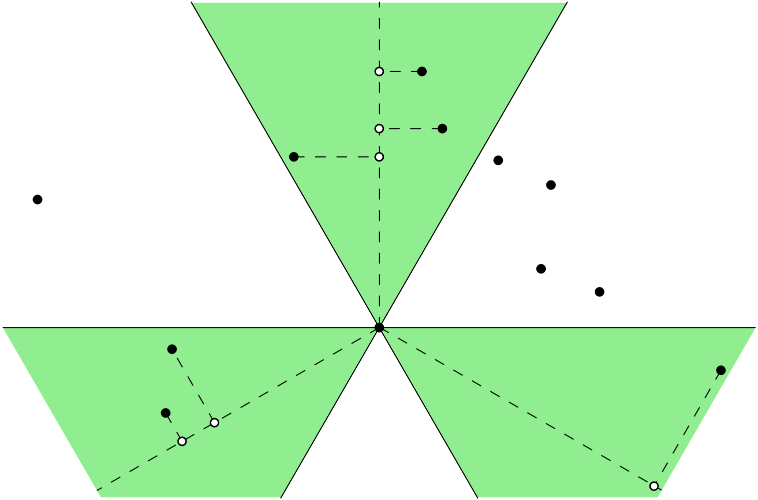
# Half-$\theta_6$-graph

- 6 Cones around each vertex: 3 positive, 3 negative



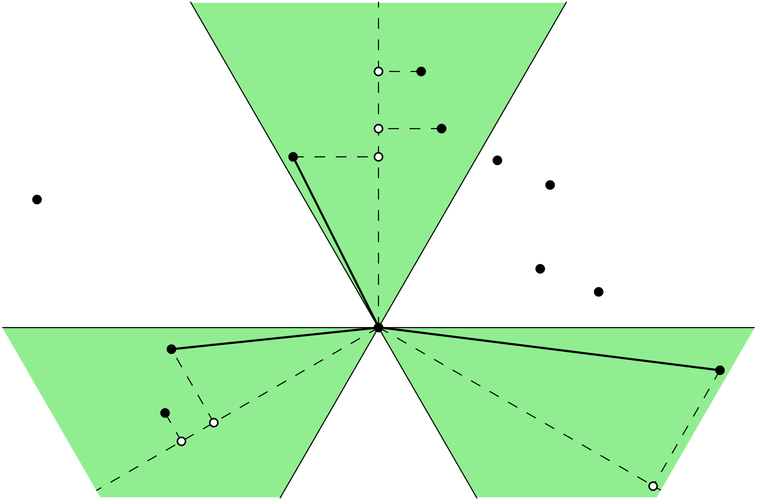
# Half-$\theta_6$-graph

- Connect to 'closest' vertex in each positive cone



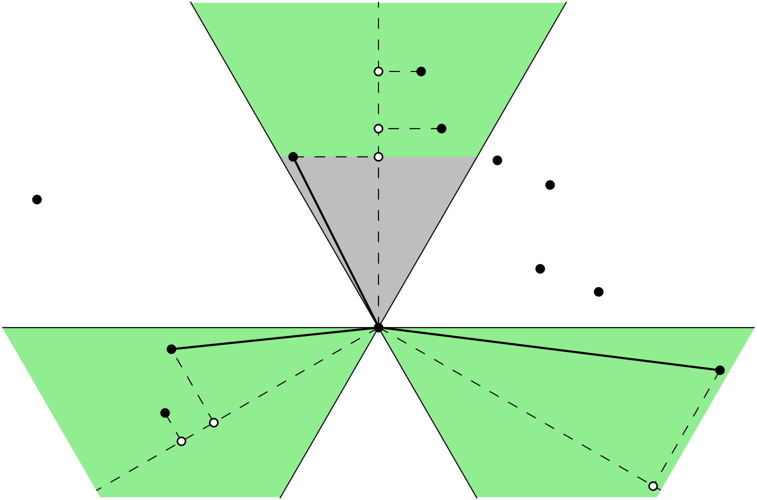
# Half-$\theta_6$-graph

- Connect to 'closest' vertex in each positive cone



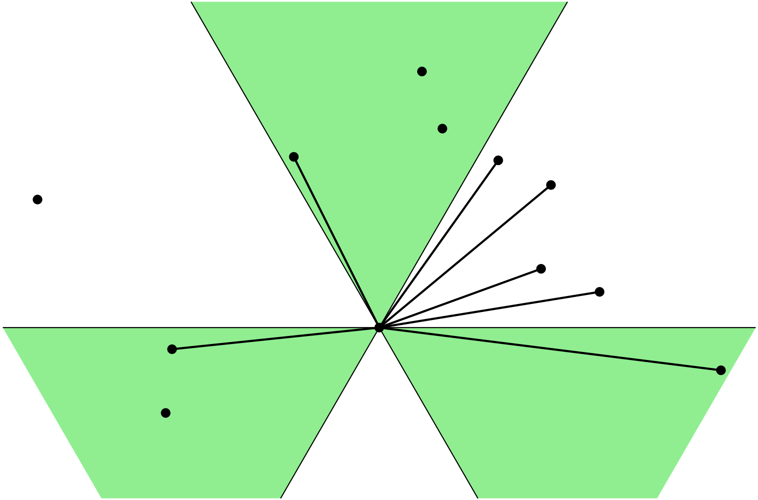
# Half-$\theta_6$-graph

- Connect to 'closest' vertex in each positive cone



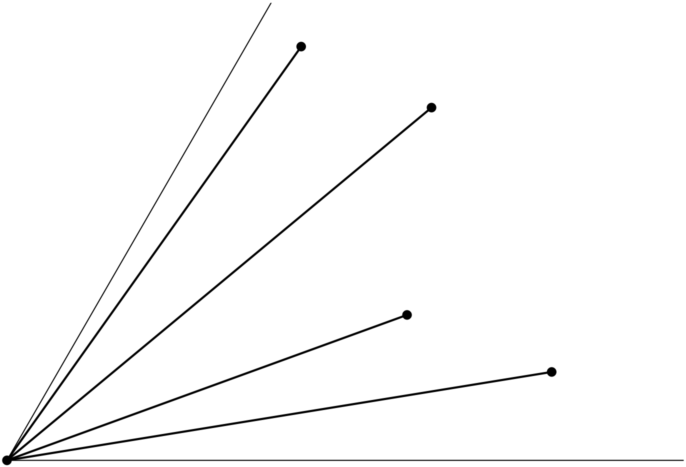
# Bounded Degree

- Negative cones can have unbounded in-degree.



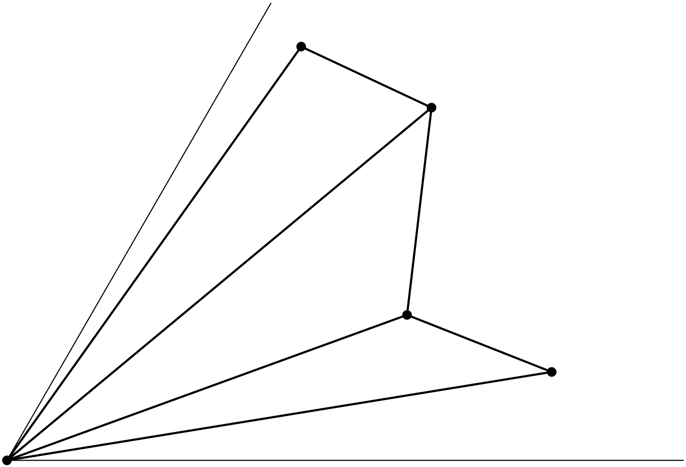
# Bounded Degree

- Negative cones can have unbounded in-degree.



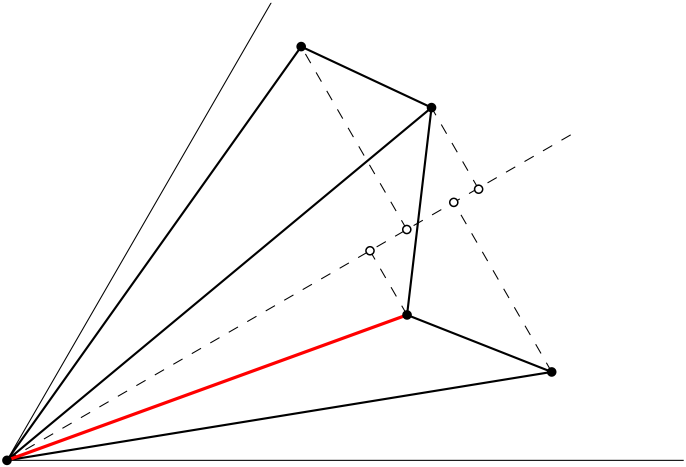
# Bounded Degree

- Consecutive vertices are connected by a *canonical path*.



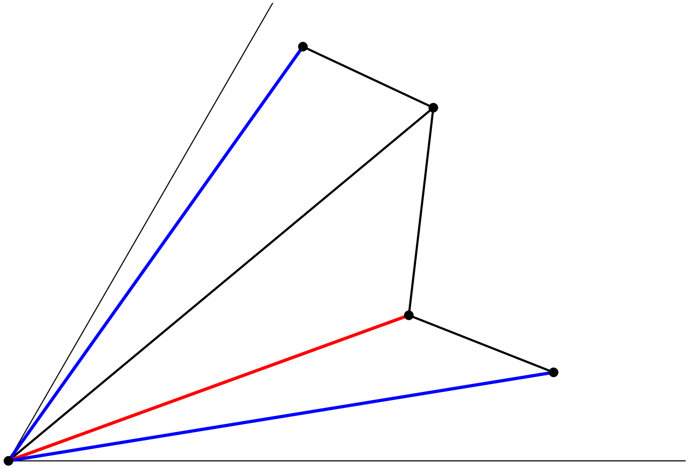
# Bounded Degree

- Keep the edge to the closest vertex...



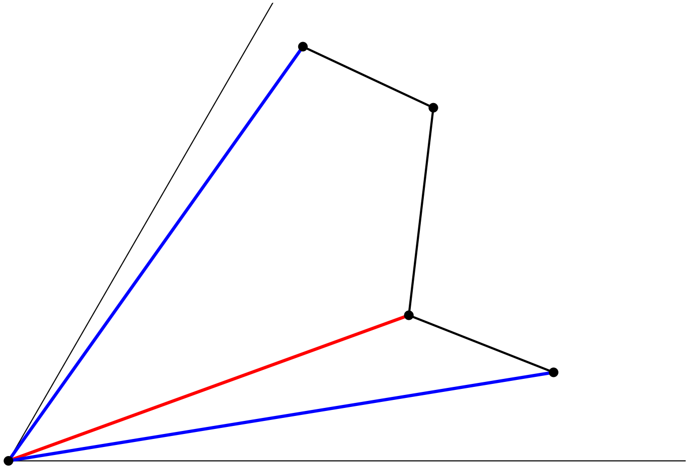
# Bounded Degree

- Keep the edge to the closest vertex and the extreme edges.



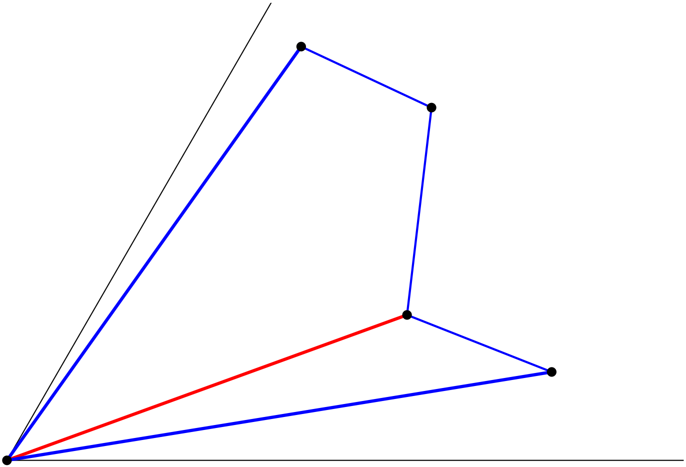
# Bounded Degree

- Keep the edge to the closest vertex and the extreme edges.



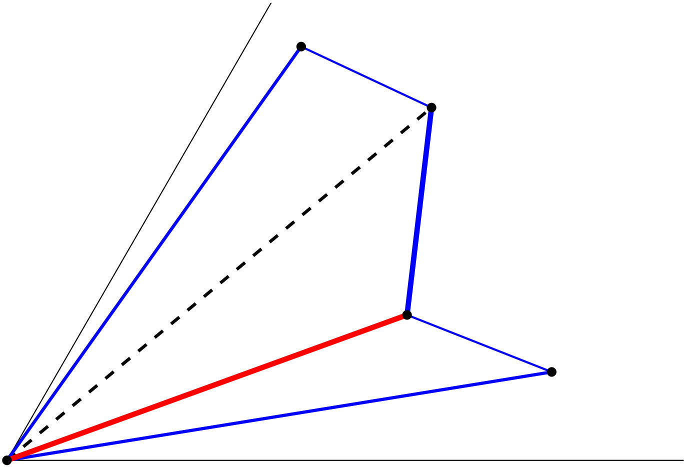
# Bounded Degree

- Edges on the canonical path are always extreme.



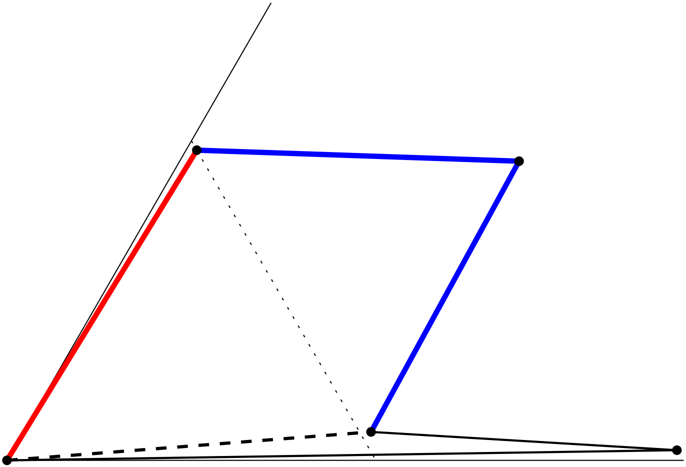
# Bounded Degree

- There is an *approximation path* for every removed edge.



# Bounded Degree

- Result: A 3-spanner of the half-$\theta_6$-graph.

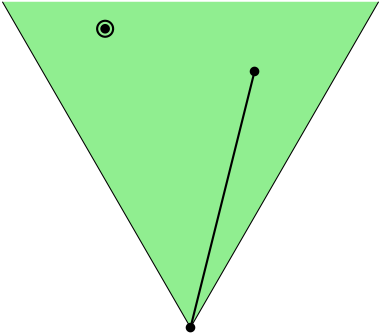


# Routing Algorithm

If $t$ lies in a positive cone:

- Follow the edge in that cone

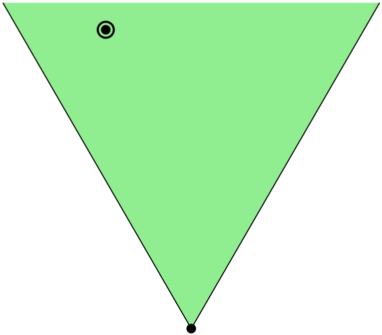
In the half-$\theta_6$-graph.



If $t$ lies in a positive cone:

- Follow the edge in that cone

In the bounded-degree subgraph.

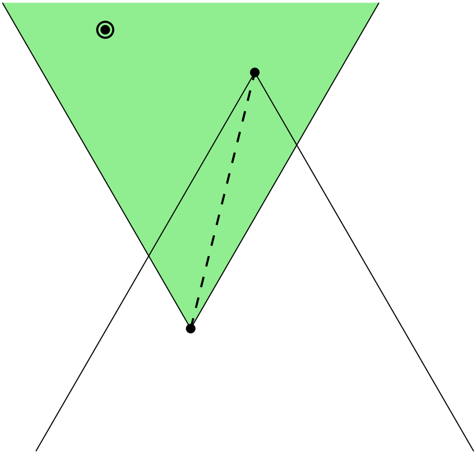


# Routing Algorithm

If $t$ lies in a positive cone:

- Follow the edge in that cone

In the bounded-degree subgraph.

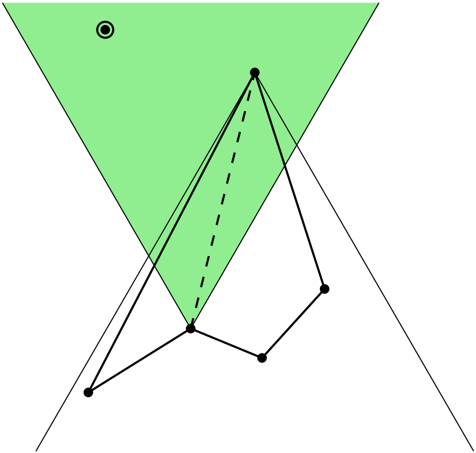


# Routing Algorithm

If $t$ lies in a positive cone:

- Follow the edge in that cone

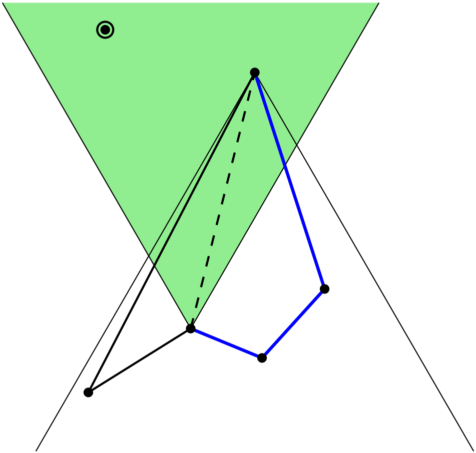
In the bounded-degree subgraph.



If $t$ lies in a positive cone:

- Follow the edge in that cone

In the bounded-degree subgraph.

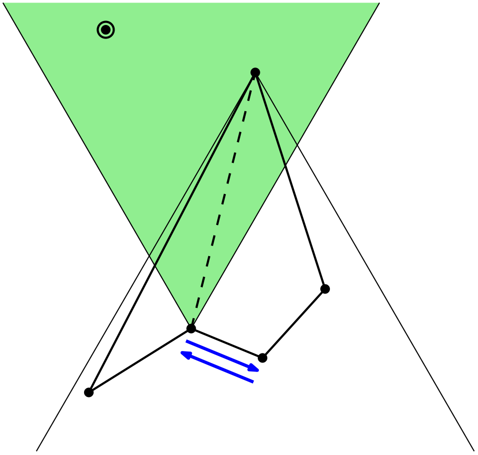


# Routing Algorithm

If $t$ lies in a positive cone:

- Follow the edge in that cone

In the bounded-degree subgraph.

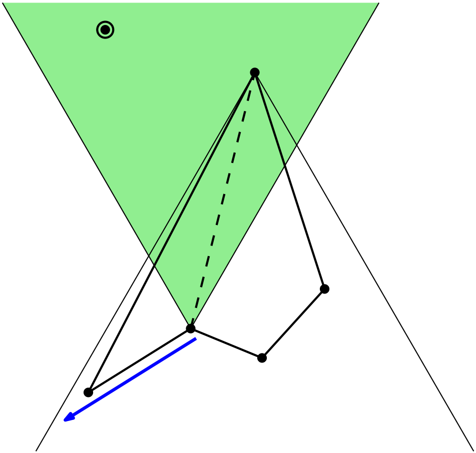


# Routing Algorithm

If $t$ lies in a positive cone:

- Follow the edge in that cone

In the bounded-degree subgraph.

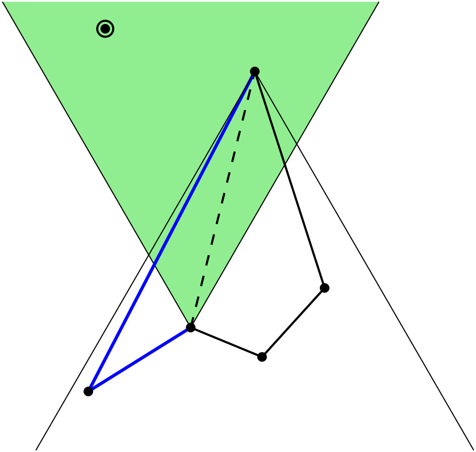


# Routing Algorithm

If $t$ lies in a positive cone:

- Follow the edge in that cone

In the bounded-degree subgraph.

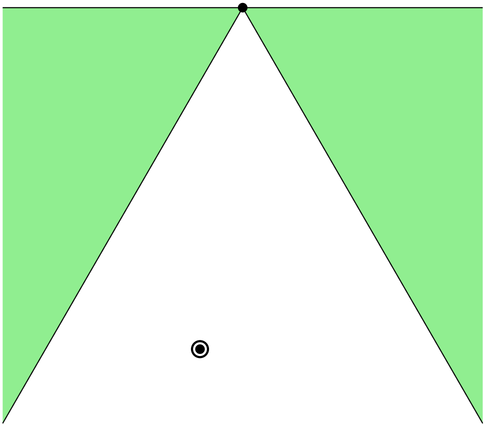


# Routing Algorithm

If $t$ lies in a negative cone and we did not mark a side yet:

- Follow an edge in that cone
- Follow an edge to the shorter side
- Follow an edge to the longer side and mark the shorter side

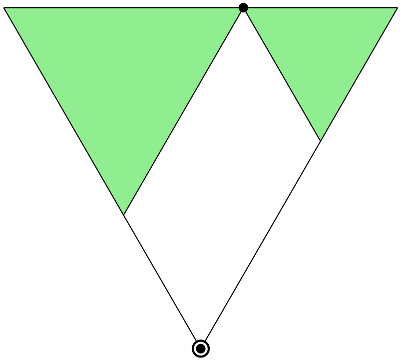
In the half-$\theta_6$-graph.



If $t$ lies in a negative cone and we did not mark a side yet:

- Follow an edge in that cone
- Follow an edge to the shorter side
- Follow an edge to the longer side and mark the shorter side

In the half-$\theta_6$-graph.

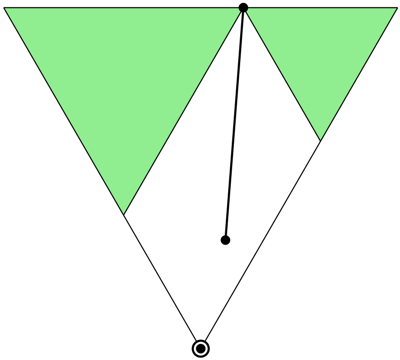


# Routing Algorithm

If $t$ lies in a negative cone and we did not mark a side yet:

- Follow an edge in that cone
- Follow an edge to the shorter side
- Follow an edge to the longer side and mark the shorter side

In the half-$\theta_6$-graph.

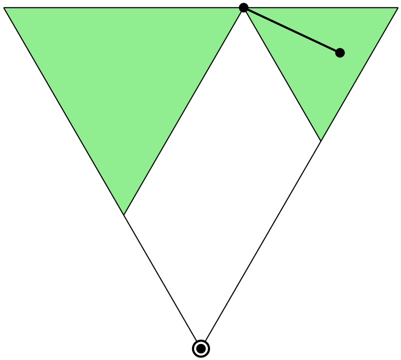


# Routing Algorithm

If $t$ lies in a negative cone and we did not mark a side yet:

- Follow an edge in that cone
- Follow an edge to the shorter side
- Follow an edge to the longer side and mark the shorter side

In the half-$\theta_6$-graph.

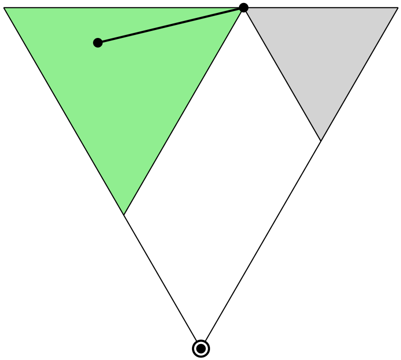


# Routing Algorithm

If $t$ lies in a negative cone and we did not mark a side yet:

- Follow an edge in that cone
- Follow an edge to the shorter side
- Follow an edge to the longer side and mark the shorter side

In the half-$\theta_6$-graph.

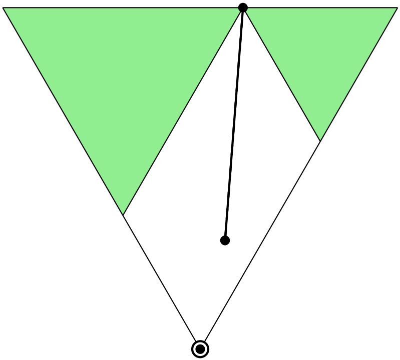


# Routing Algorithm

If $t$ lies in a negative cone and we did not mark a side yet:

- Follow an edge in that cone
- Follow an edge to the shorter side
- Follow an edge to the longer side and mark the shorter side

In the bounded-degree subgraph.

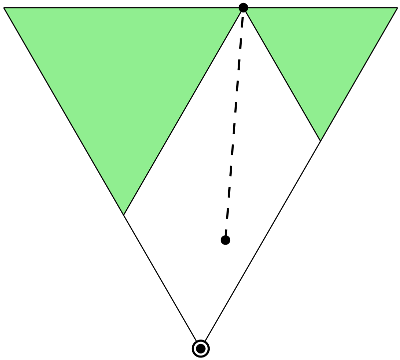


# Routing Algorithm

If $t$ lies in a negative cone and we did not mark a side yet:

- Follow an edge in that cone
- Follow an edge to the shorter side
- Follow an edge to the longer side and mark the shorter side

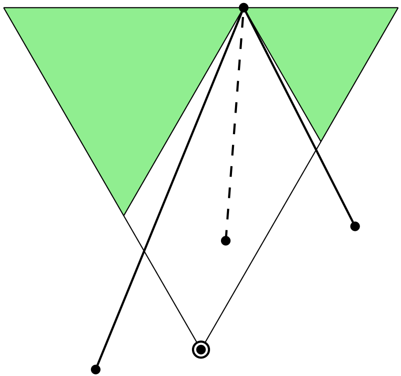
In the bounded-degree subgraph.



If $t$ lies in a negative cone and we did not mark a side yet:

- Follow an edge in that cone
- Follow an edge to the shorter side
- Follow an edge to the longer side and mark the shorter side

In the bounded-degree subgraph.

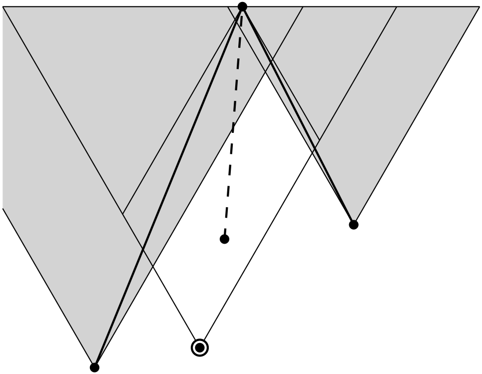


# Routing Algorithm

If $t$ lies in a negative cone and we did not mark a side yet:

- Follow an edge in that cone
- Follow an edge to the shorter side
- Follow an edge to the longer side and mark the shorter side

In the bounded-degree subgraph.

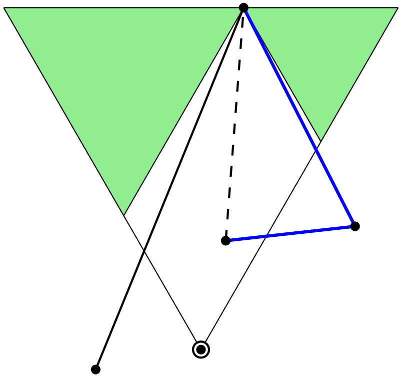


# Routing Algorithm

If $t$ lies in a negative cone and we did not mark a side yet:

- Follow an edge in that cone
- Follow an edge to the shorter side
- Follow an edge to the longer side and mark the shorter side

In the bounded-degree subgraph.

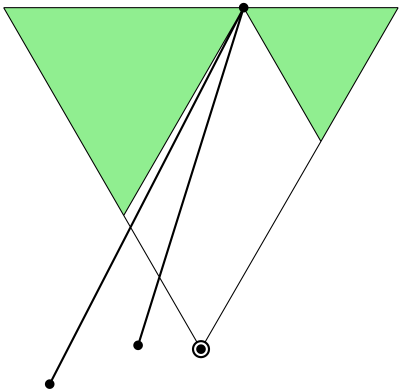


# Routing Algorithm

If $t$ lies in a negative cone and we did not mark a side yet:

- Follow an edge in that cone
- Follow an edge to the shorter side
- Follow an edge to the longer side and mark the shorter side

In the bounded-degree subgraph.

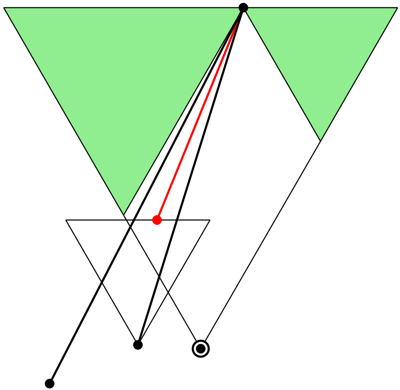


# Routing Algorithm

If $t$ lies in a negative cone and we did not mark a side yet:

- Follow an edge in that cone
- Follow an edge to the shorter side
- Follow an edge to the longer side and mark the shorter side

In the bounded-degree subgraph.

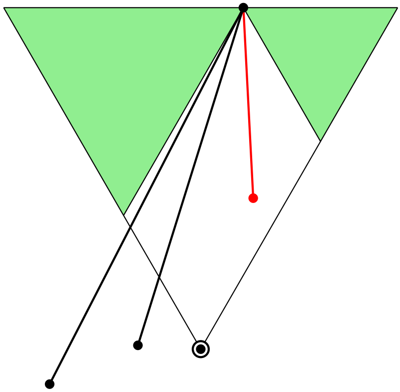


# Routing Algorithm

If $t$ lies in a negative cone and we did not mark a side yet:

- Follow an edge in that cone
- Follow an edge to the shorter side
- Follow an edge to the longer side and mark the shorter side

In the bounded-degree subgraph.

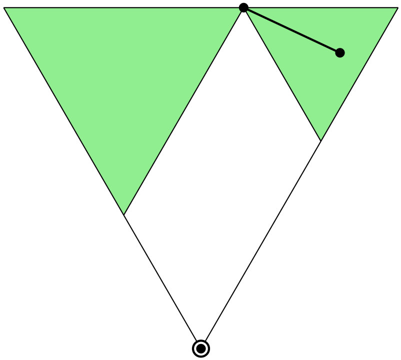


# Routing Algorithm

If $t$ lies in a negative cone and we did not mark a side yet:

- Follow an edge in that cone
- Follow an edge to the shorter side
- Follow an edge to the longer side and mark the shorter side

In the bounded-degree subgraph.

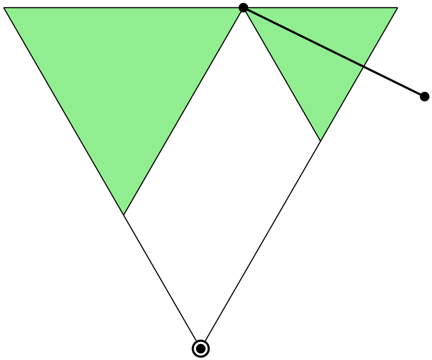


# Routing Algorithm

If $t$ lies in a negative cone and we did not mark a side yet:

- Follow an edge in that cone
- Follow an edge to the shorter side
- Follow an edge to the longer side and mark the shorter side

In the bounded-degree subgraph.

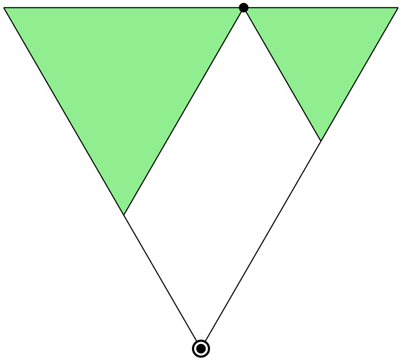


# Routing Algorithm

If $t$ lies in a negative cone and we did not mark a side yet:

- Follow an edge in that cone
- Follow an edge to the shorter side
- Follow an edge to the longer side and mark the shorter side

In the bounded-degree subgraph.

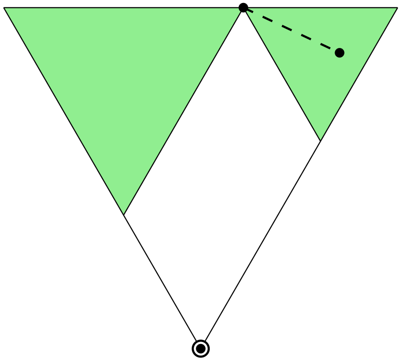


# Routing Algorithm

If $t$ lies in a negative cone and we did not mark a side yet:

- Follow an edge in that cone
- Follow an edge to the shorter side
- Follow an edge to the longer side and mark the shorter side

In the bounded-degree subgraph.

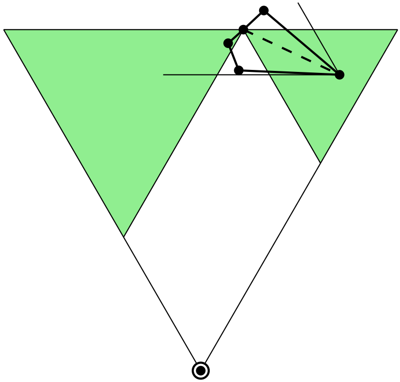


# Routing Algorithm

If $t$ lies in a negative cone and we did not mark a side yet:

- Follow an edge in that cone
- Follow an edge to the shorter side
- Follow an edge to the longer side and mark the shorter side

In the bounded-degree subgraph.

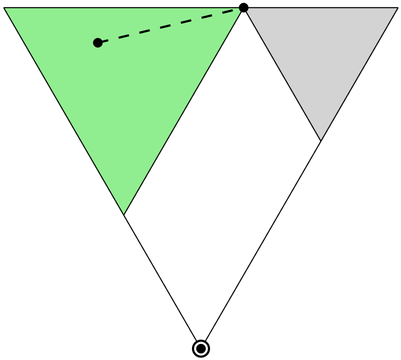


# Routing Algorithm

If $t$ lies in a negative cone and we did not mark a side yet:

- Follow an edge in that cone
- Follow an edge to the shorter side
- Follow an edge to the longer side and mark the shorter side

In the bounded-degree subgraph.

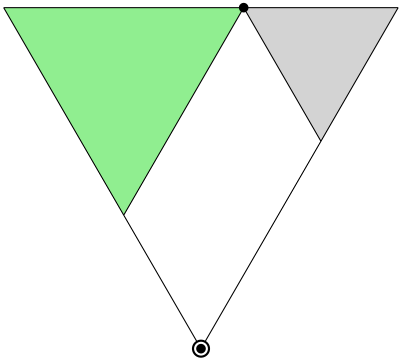


# Routing Algorithm

If $t$ lies in a negative cone and we marked a side:

- Follow the edge closest to the marked side

In the half-$\theta_6$-graph.

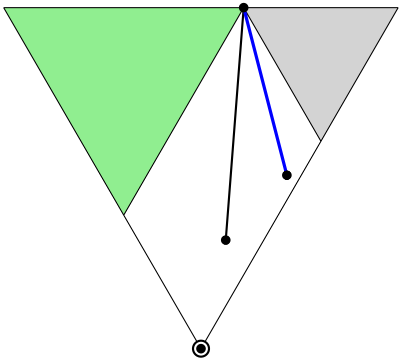


# Routing Algorithm

If $t$ lies in a negative cone and we marked a side:

- Follow the edge closest to the marked side

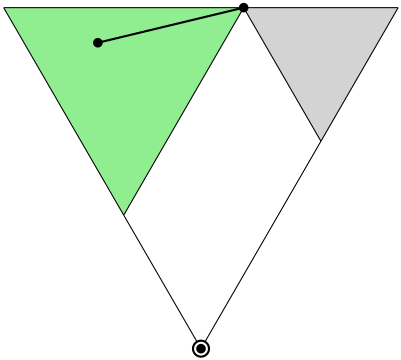
In the half-$\theta_6$-graph.



If $t$ lies in a negative cone and we marked a side:

- Follow the edge closest to the marked side

In the half-$\theta_6$-graph.

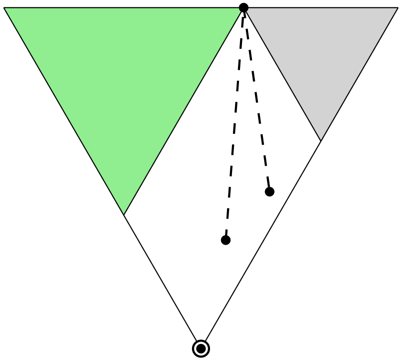


# Routing Algorithm

If $t$ lies in a negative cone and we marked a side:

- Follow the edge closest to the marked side

In the bounded-degree subgraph.

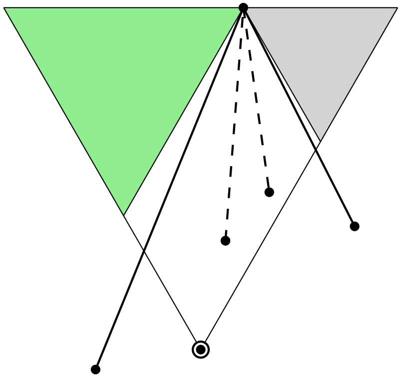


# Routing Algorithm

If $t$ lies in a negative cone and we marked a side:

- Follow the edge closest to the marked side

In the bounded-degree subgraph.

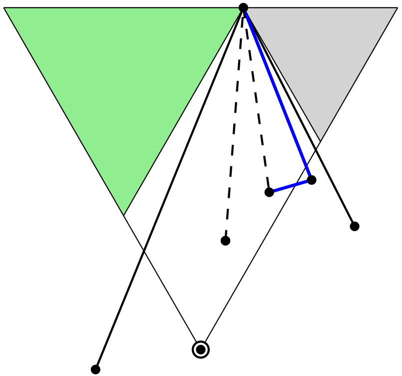


# Routing Algorithm

If $t$ lies in a negative cone and we marked a side:

- Follow the edge closest to the marked side

In the bounded-degree subgraph.

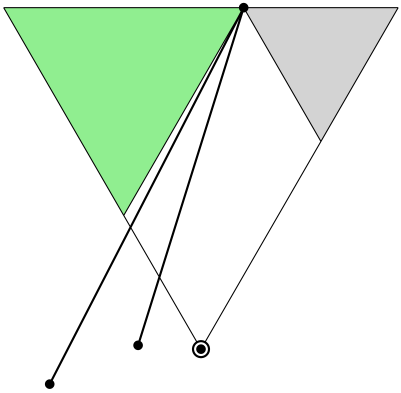


# Routing Algorithm

If $t$ lies in a negative cone and we marked a side:

- Follow the edge closest to the marked side

In the bounded-degree subgraph.

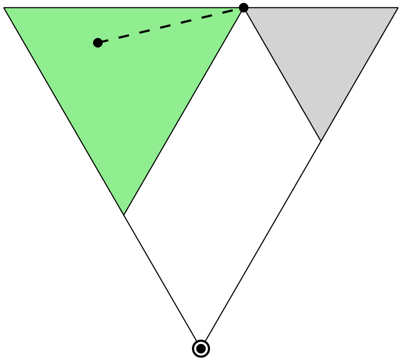


# Routing Algorithm

If $t$ lies in a negative cone and we marked a side:

- Follow the edge closest to the marked side

In the bounded-degree subgraph.



# Conclusion

- Bounded-degree spanners allow competitive routing.

# Conclusion

- Bounded-degree spanners allow competitive routing.
- Routing ratio can be improved by storing information at vertices.