Connecting Obstacles in Vertex-Disjoint Paths

Marwan Al-Jubeh
Mashhood Ishaque
Csaba D. Toth

Gill Barequet
Diane L. Souvaine
Andrew Winslow
Outline

1. Problem Definition.
2. Lower Bound Constructions.
3. Upper Bound (Algorithm).
Problem Definition

Given:

- $k$ disjoint polygonal obstacles
- Triangular container

Add Straight Line, and
Non-Crossing Edges:

such that each obstacle has
3 vertex-disjoint paths to
container vertices.
Problem Definition

Given:

- k disjoint polygonal obstacles
- Triangular container

Add Straight Line, and Non-Crossing Edges:

- such that each obstacle has 3 vertex-disjoint paths to container vertices.
Problem Definition

Given:

- $k$ disjoint polygonal obstacles
- Triangular container

Add Straight Line, and Non-Crossing Edges:

such that each obstacle has 3 vertex-disjoint paths to container vertices.
Problem Definition

Questions:
• Is it always possible?
• For k obstacles, how many edges are necessary?
• How many edges are enough?
Is Augmentation Always Possible?

For non-convex obstacles:
Is Augmentation Always Possible?

For non-convex obstacles:

No.

The innermost obstacle sees only the vertices of a single obstacle.

There cannot be three vertex-disjoint paths unless we can add edges in an obstacle’s interior.
Is Augmentation Always Possible?

For convex obstacles:
Is Augmentation Always Possible?

For convex obstacles:

Yes.

Triangulation of the free space is 3-connected. [TV09]

Is Augmentation Always Possible?

For convex obstacles:

Yes.

Augment the triangulation with nodes p and q. The augmented graph is 3-connected.

→ 3 vertex-disjoint paths between nodes p and q.

→ 3 vertex disjoint paths between the obstacle and the container.
Problem Definition

Given:
- \(k\) disjoint convex polygonal obstacles
- Triangular container

Add Straight Line, and Non-Crossing Edges:
- such that each obstacle has 3 vertex-disjoint paths to container vertices.
Outline

1. Problem Definition.
2. Lower Bound Constructions.
3. Upper Bound (Algorithm).
How Many Edges are Necessary?

For \( k \) convex obstacles:

\[ 3k - 1 \text{ edges}. \]
How Many Edges are Necessary?

For \( k \) convex obstacles:

\[ 3k - 1 \text{ edges.} \]

A big obstacle (\( k \)-gon).
One obstacle hidden behind each side (except the base).

Hidden obstacles need 3 edges each.
\( k \)-gon needs 2 edges.

For a single obstacle, trivial lower bound of 3.
How Many Edges are Necessary?

For $k$ convex obstacles, each with at most $s$ sides:

$$3k - \frac{(k-1)}{(s-1)}$$ edges.
How Many Edges are Necessary?

For $k$ convex obstacles, each with at most $s$ sides:

$$3k - \frac{(k-1)}{(s-1)}$$ edges.

Hide obstacles recursively. A complete $(s-1)$-ary tree. No. of leaves = $k - \frac{(k-1)}{(s-1)}$.

Each leaf obstacle need 3 edges. All other obstacles need 2.

$$3 \left( k - \frac{(k-1)}{(s-1)} \right) + 2 \left( \frac{(k-1)}{(s-1)} \right)$$
How Many Edges are Necessary?

For $k$ convex obstacles, each with at most $s$ sides:

$$3k - \frac{(k-1)}{(s-1)}$$ edges.

For $s = 3$:

$$\frac{5}{2} k$$ edges are necessary.
Outline
1. Problem Definition.
2. Lower Bound Constructions.
3. Upper Bound (Algorithm).
How Many Edges are Enough?

For $k$ convex obstacles:

$3k$ edges.
Augmentation Algorithm (skeleton)

Given: A polygon P with 3-connected triangulation, and three unique colored vertices on its boundary.

- Pick an arbitrary obstacle.
- Find 3 vertex-disjoint paths.
- Shorten paths if necessary.
- Generate sub-problems.
- Handle 2-Cuts if necessary.
- Recurse.
Augmentation Algorithm (skeleton)

Given: A polygon P with 3-connected triangulation, and three unique colored vertices on its boundary.

- Pick an arbitrary obstacle.
- Find 3 vertex-disjoint paths.
- Shorten paths if necessary.
- Generate sub-problems.
- Handle 2-Cuts if necessary.
- Recurse.
Augmentation Algorithm (skeleton)

Given: A polygon $P$ with 3-connected triangulation, and three unique colored vertices on its boundary.

- Pick an arbitrary obstacle.
- Find 3 vertex-disjoint paths.
- Shorten paths if necessary.
- Generate sub-problems.
- Handle 2-Cuts if necessary.
- Recurse.
Augmentation Algorithm (skeleton)

Given: A polygon P with 3-connected triangulation, and three unique colored vertices on its boundary.

- Pick an arbitrary obstacle.
- Find 3 vertex-disjoint paths.
- **Shorten paths if necessary.**
- Generate sub-problems.
- Handle 2-Cuts if necessary.
- Recurse.
Augmentation Algorithm (skeleton)

Given: A polygon $P$ with 3-connected triangulation, and three unique colored vertices on its boundary.

• Pick an arbitrary obstacle.
• Find 3 vertex-disjoint paths.
• Shorten paths if necessary.
• Generate sub-problems.
• Handle 2-Cuts if necessary.
• Recurse.
Augmentation Algorithm (skeleton)

Given: A polygon P with 3-connected triangulation, and three designated vertices on its boundary.

- Pick an arbitrary obstacle.
- Find 3 vertex-disjoint paths.
- Shorten paths if necessary.
- Generate sub-problems.
- Handle 2-Cuts if necessary.
- Recurse.
**Shorten Path Algorithm**

**Given:** 3 vertex-disjoint paths $\pi_R$, $\pi_G$, and $\pi_B$.

For a path $\pi_i$ if the two non-adjacent vertices $v_1$ and $v_2$ see each other or are incident on the same obstacle:

- Create a simple polygon $Q$ using the sub-path between $v_1$ and $v_2$, and the segment $v_1v_2$. (Q is empty of other paths.)
- Find geodesic path between $v_1$ and $v_2$ inside $Q$. 

![Diagram showing paths and obstacles]
Shorten Path Algorithm

Given: 3 vertex-disjoint paths $\pi_R$, $\pi_G$, and $\pi_B$.

For a path $\pi_i$ if the two non-adjacent vertices $v_1$ and $v_2$ see each other or are incident on the same obstacle:

- Create a simple polygon $Q$ using the sub-path between $v_1$ and $v_2$, and the segment $v_1v_2$. (Q is empty of other paths.)
- Find geodesic path between $v_1$ and $v_2$ inside $Q$. 
Shorten Path Algorithm

Given: 3 vertex-disjoint paths $\pi_R$, $\pi_G$, and $\pi_B$.

For a path $\pi_i$ if the two non-adjacent vertices $v_1$ and $v_2$ see each other or are incident on the same obstacle:

- Create a simple polygon $Q$ using the sub-path between $v_1$ and $v_2$, and the segment $v_1v_2$. (Q is empty of other paths.)
- Find geodesic path between $v_1$ and $v_2$ inside $Q$. 
Shorten Path Algorithm

Given: 3 vertex-disjoint paths \( \pi_R, \pi_G, \) and \( \pi_B \).

For a path \( \pi_i \) if the two non-adjacent vertices \( v_1 \) and \( v_2 \) see each other or are incident on the same obstacle:

- Create a simple polygon \( Q \) using the sub-path between \( v_1 \) and \( v_2 \), and the segment \( v_1v_2 \). (\( Q \) is empty of other paths.)
- Find geodesic path between \( v_1 \) and \( v_2 \) inside \( Q \).
Shorten Path Algorithm

Given: 3 vertex-disjoint paths $\pi_R$, $\pi_G$, and $\pi_B$.

For a path $\pi_i$ if the two non-adjacent vertices $v_1$ and $v_2$ see each other or are incident on the same obstacle:

- Create a simple polygon $Q$ using the sub-path between $v_1$ and $v_2$, and the segment $v_1v_2$. (Q is empty of other paths.)
- Find geodesic path between $v_1$ and $v_2$ inside $Q$. 
Shorten Path Algorithm

Given: 3 vertex-disjoint paths $\pi_R$, $\pi_G$, and $\pi_B$.

For a path $\pi_i$ if the two non-adjacent vertices $v_1$ and $v_2$ see each other or are incident on the same obstacle:

- Create a simple polygon $Q$ using the sub-path between $v_1$ and $v_2$, and the segment $v_1v_2$.
  (Q is empty of other paths.)
- Find geodesic path between $v_1$ and $v_2$ inside $Q$. 
Shorten Path Algorithm

Given: 3 vertex-disjoint paths \( \pi_R \), \( \pi_G \), and \( \pi_B \).

For a path \( \pi_i \) if the two non-adjacent vertices \( v_1 \) and \( v_2 \) see each other or are incident on the same obstacle:
- Create a simple polygon \( Q \) using the sub-path between \( v_1 \) and \( v_2 \), and the segment \( v_1v_2 \). (\( Q \) is empty of other paths.)
- Find geodesic path between \( v_1 \) and \( v_2 \) inside \( Q \).
Handle 2-Cuts Algorithm

**Given:** A subpolygon $P'$ such that every triangulation of $P'$ contain a 2-Cut.

- Find an extremal 2-cut $C_1$.
- $C_1$ divides $P'$ into $P_1'$ and $P_2'$.
- Designate vertices on $P_1'$.
- Recurse on $P_1'$.
- Handle 2-Cuts in $P_2'$ if necessary.
- Otherwise recurse on $P_2'$. 

Handle 2-Cuts Algorithm

**Given:** A subpolygon P’ such that every triangulation of P’ contain a 2-Cut.

- Find an extremal 2-cut $C_1$.
- $C_1$ divides P’ into $P_1’$ and $P_2’$.
- Designate vertices on $P_1’$.
- Recurse on $P_1’$.
- Handle 2-Cuts in $P_2’$ if necessary
- Otherwise recurse on $P_2’$. 
Augmentation Algorithm (skeleton)

Given: A polygon P with 3-connected triangulation, and three unique colored vertices on its boundary.

- Pick an arbitrary obstacle.
- Find 3 vertex-disjoint paths.
- Shorten paths if necessary.
- Generate sub-problems.
- Handle 2-Cuts if necessary.
- Recurse.
How Many Edges are Enough?

For $k$ convex obstacles:

3$k$ edges.

- Edges along obstacles are free.
- Each path $\pi_i$ enters and leaves an obstacle at most once.
- Each obstacle is charged for the leaving edge.
- Each obstacle is charged at most three times, once for each color.
- Each edge is charged to some obstacle.
Connecting polygonal obstacles in 3 vertex-disjoint paths

- Not always possible for non-convex obstacles.
- Triangulation of the free space around convex obstacles contains desired augmentation.
- For k convex obstacles, 3k-1 edges are necessary.
- For k convex obstacle, each with at most s sides: 3k – (k-1)/(s-1) edges are necessary.
- There is an augmentation using only 3k edges.
Open Problems

For $k$ triangular obstacles, we need:

$$\frac{5}{2} k$$ edges.

Find an augmentation that matches the lower bound.
Open Problems

Augmenting a 3-augmentable 2-regular graph to a 3-connected graph
Open Problems

Augmenting a 3-augmentable 2-regular graph to a 3-connected graph

A graph is 3-augmentable iff:
1) Not all vertices in convex position
2) No chord. [TV09]
Open Problems

Augmenting a 3-augmentable 2-regular graph to a 3-connected graph

A 3-augmentable graph on n vertices.

Upper bound for general graphs: 3n – 5
• n-1 edges, to get connected graph
• n-2 edges, to get connected to 2-connected graph
• n-2 edges, to get 2-connected to 3-connected graph

Lower bound for general graph: 2n - 2

Special case of 2-regular graphs?
Thanks for Listening.