MONASH University

URBAN COMPUTING LAB

Ultrafast Euclidean Shortest Path Computation using Hub Labeling
AAAI 2023

Jinchun Du, Bojie Shen, Muhammad Aamir Cheema

Monash University

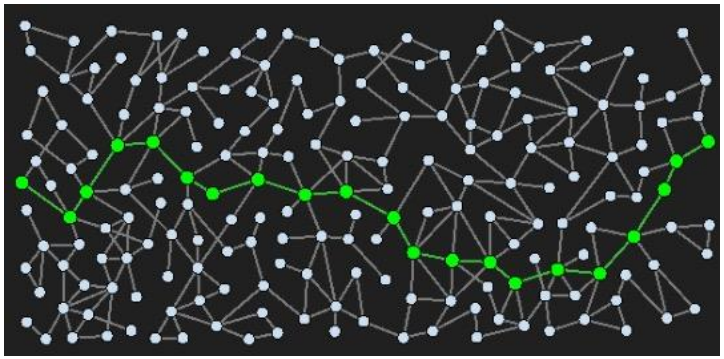
Outline

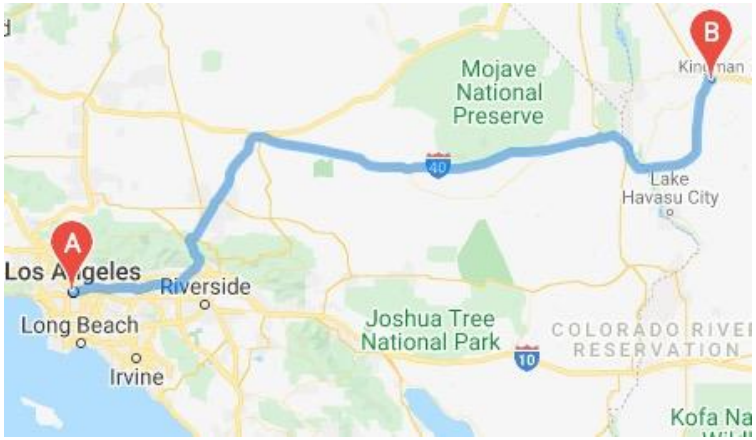
MONASH University

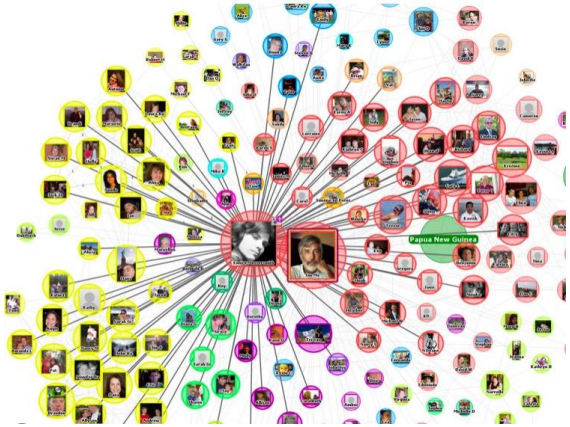
- Problem definition
- Previous works
  - Polyanya
  - End Point Search (EPS)
- Euclidean Hub Labeling
  - Preprocessing
  - Query processing
  - Optimisations
- Results

Pathfinding

MONASH University



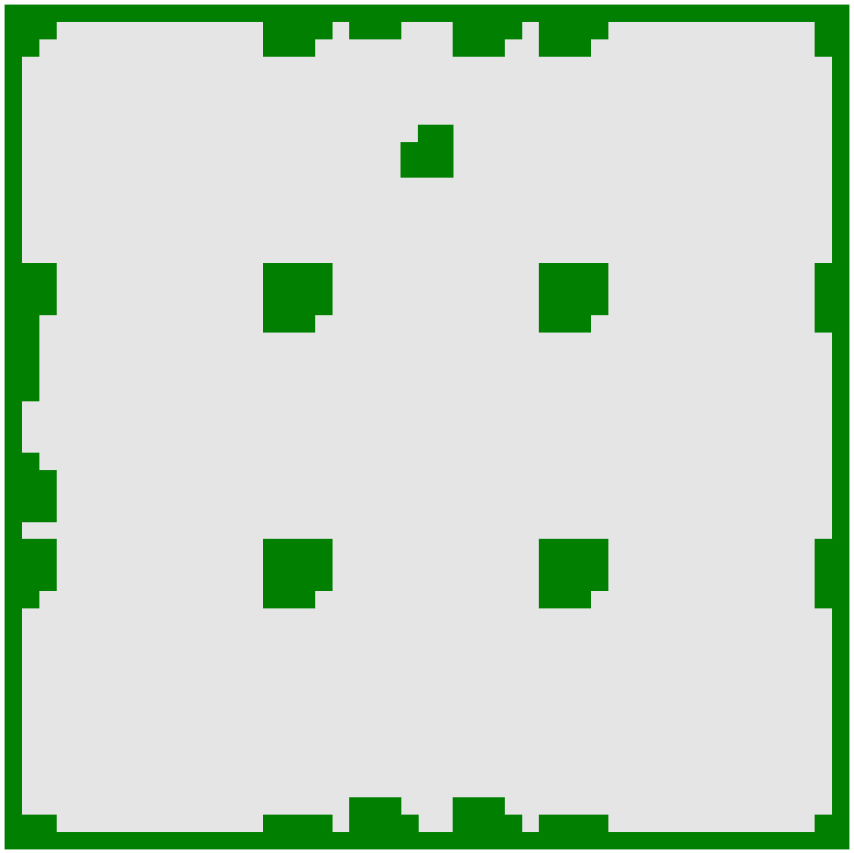




Problem Definition

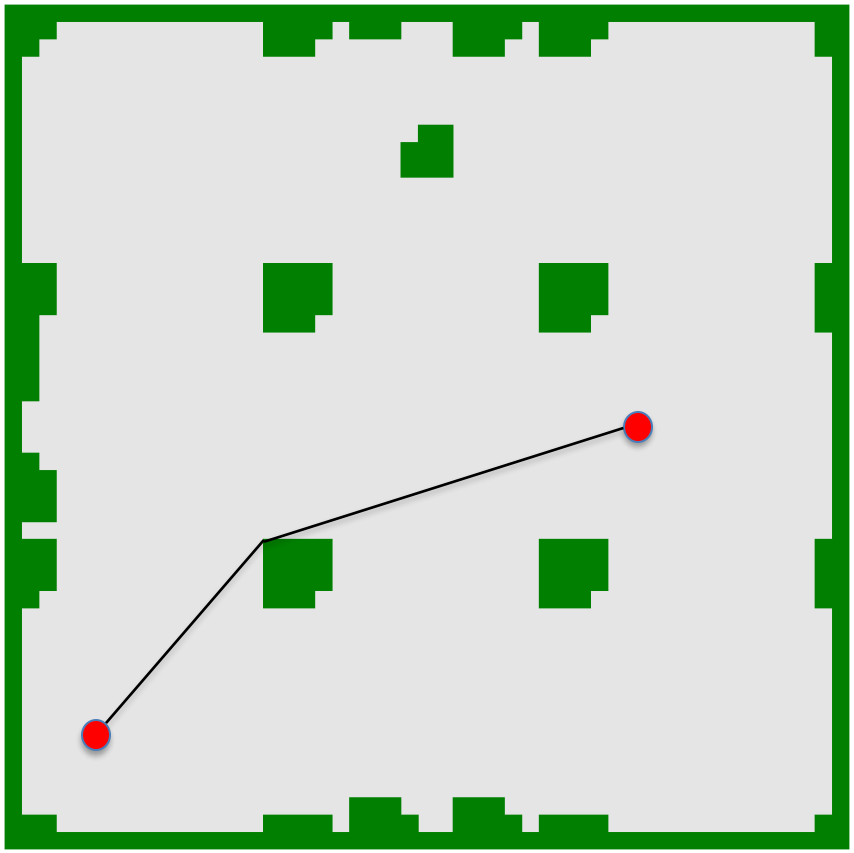
MONASH University

- Euclidean plane:
  - polygonal obstacles
    - a set of vertices
    - set of closed edges



Problem Definition

- Euclidean plane:
  - polygonal obstacles
    - a set of vertices
    - set of closed edges
- Euclidean Shortest Path Problem:
  - For a given pair of start s and target t, ESPP finds a shortest obstacles-avoiding path for an agent to travel from s to t.



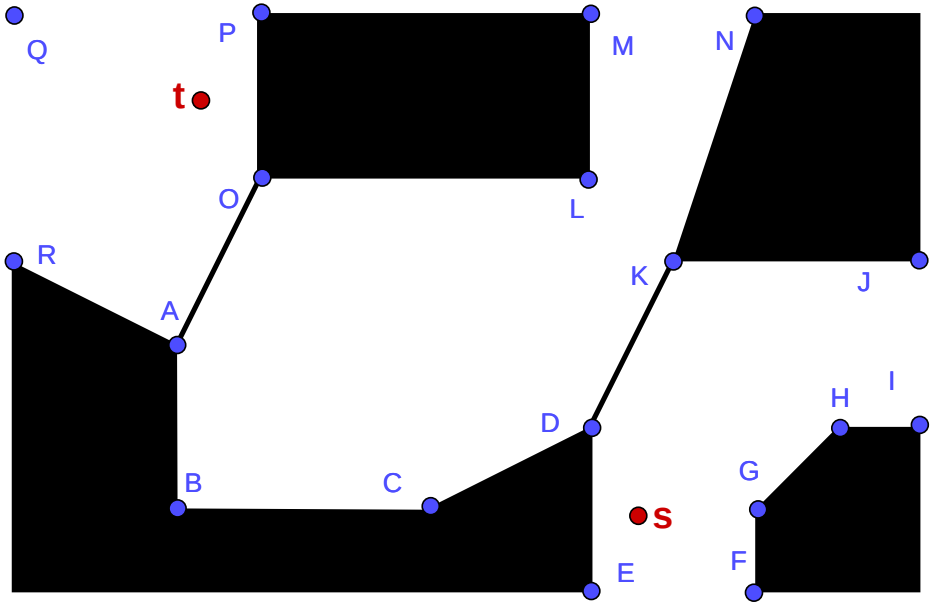
Outline

MONASH University

- Problem definition
- Previous works
  - Polyanya
  - End Point Search (EPS)
- Euclidean Hub Labeling
  - Preprocessing
  - Query processing
  - Optimisations
- Results

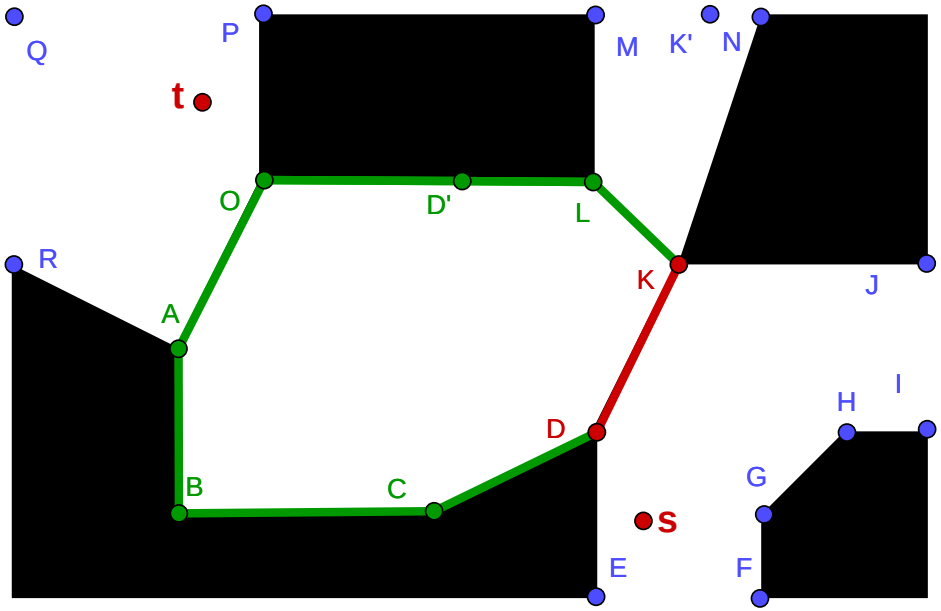
Previous work
Polyanya [1]

- Runs on navigation mesh with a set of convex polygons



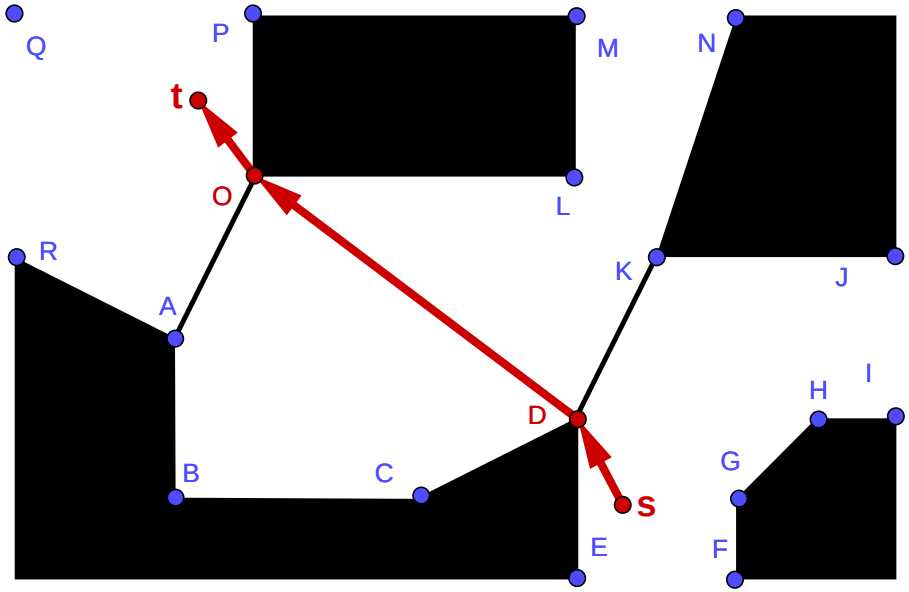
Previous work
Polyanya [1]

- Runs on navigation mesh with a set of convex polygons
- Run an A*-like search starting from s



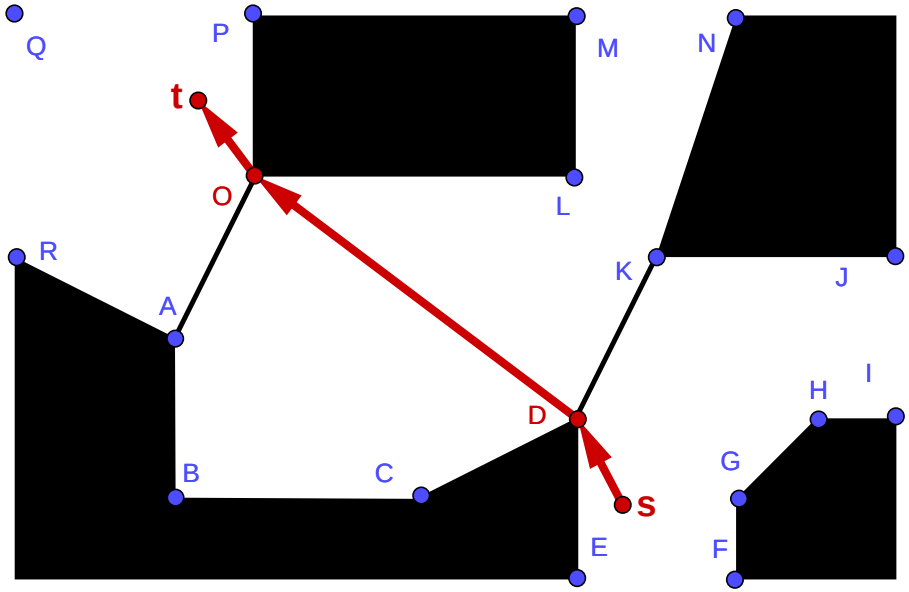
Previous work
Polyanya [1]

- Runs on navigation mesh with a set of convex polygons
- Run an A*-like search starting from s
- Advantages and disadvantages?



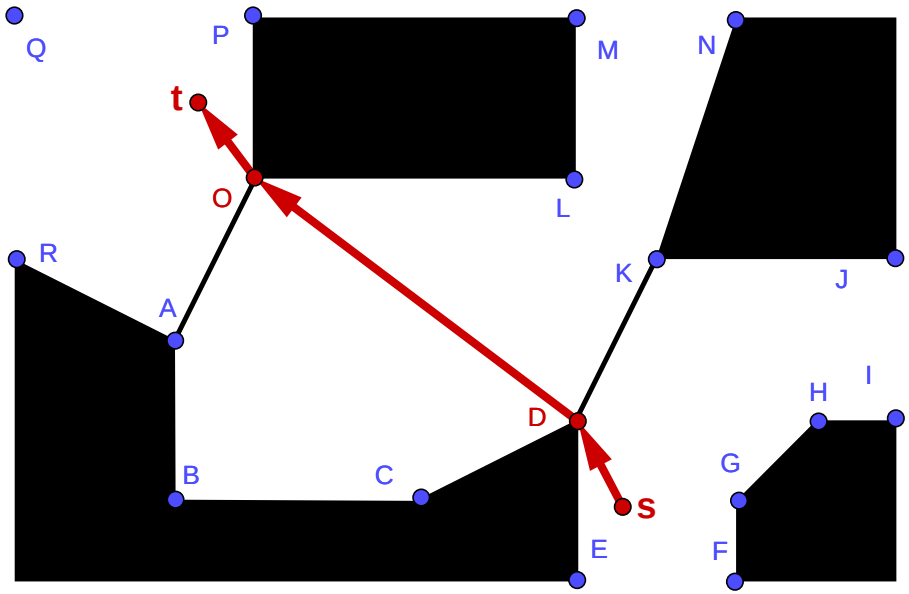
Previous work
Polyanya [1]

- Runs on navigation mesh with a set of convex polygons
- Run an A*-like search starting from s
- Advantage: no preprocessing needed



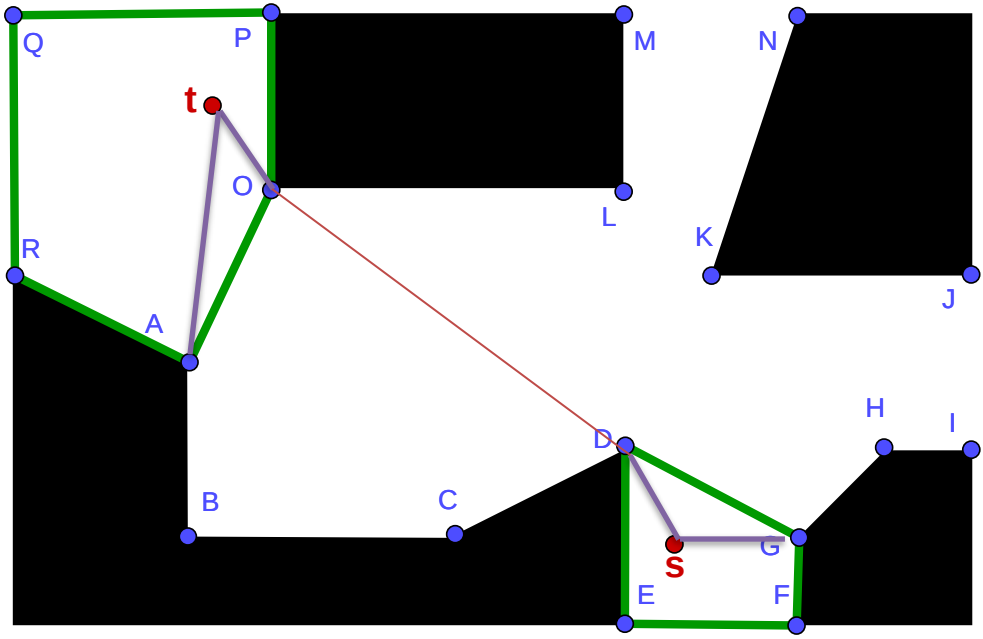
Previous work
Polyanya [1]

- Runs on navigation mesh with a set of convex polygons
- Run an A*-like search starting from s
- Advantage: no preprocessing needed
- Disadvantage: query runtime increases as s and t are further apart



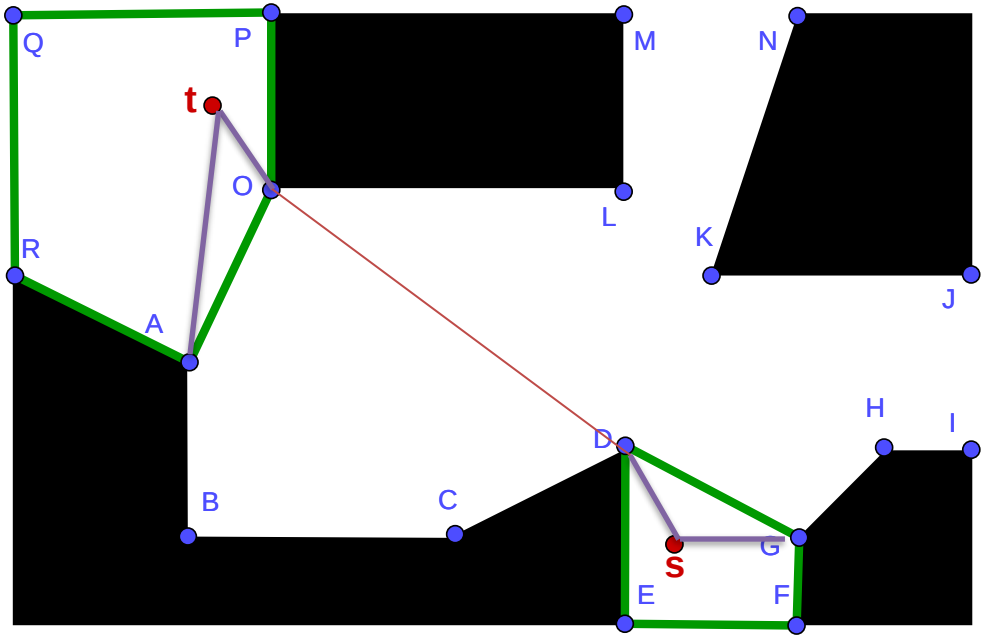
Previous work
End Point Search (EPS) [2]

- Distance oracle – CPD [3]



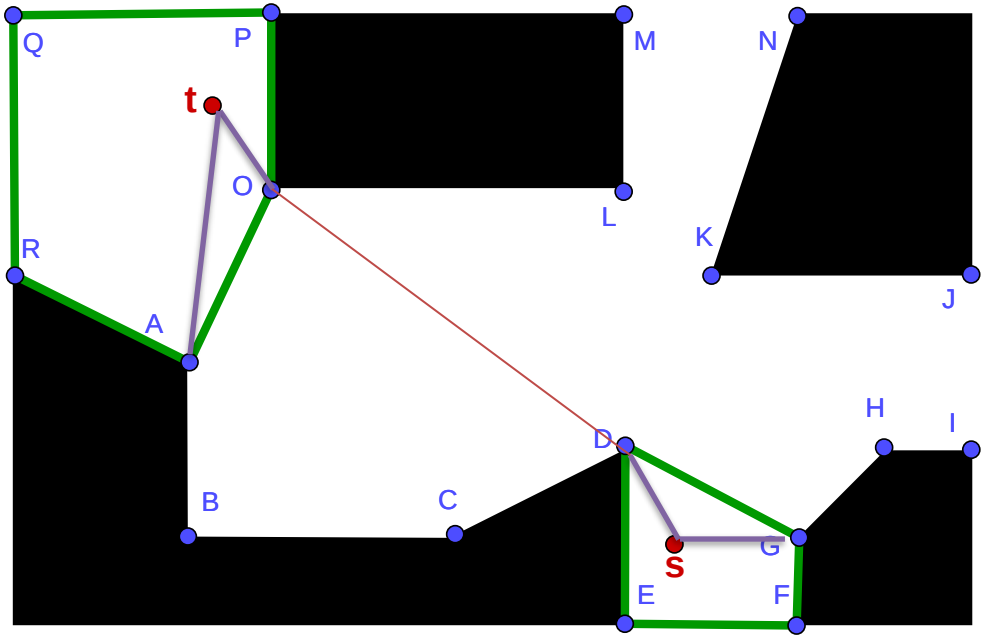
Previous work
End Point Search (EPS) [2]

- Distance oracle – CPD [3]
- EPS - query
  - Incremental exploration from s and t
  - Connects s and t to visible vertices V_s and V_t
  - For each u in V_s and v in V_t
    - CPD finds a path between u and v
    - Maintains the best distance



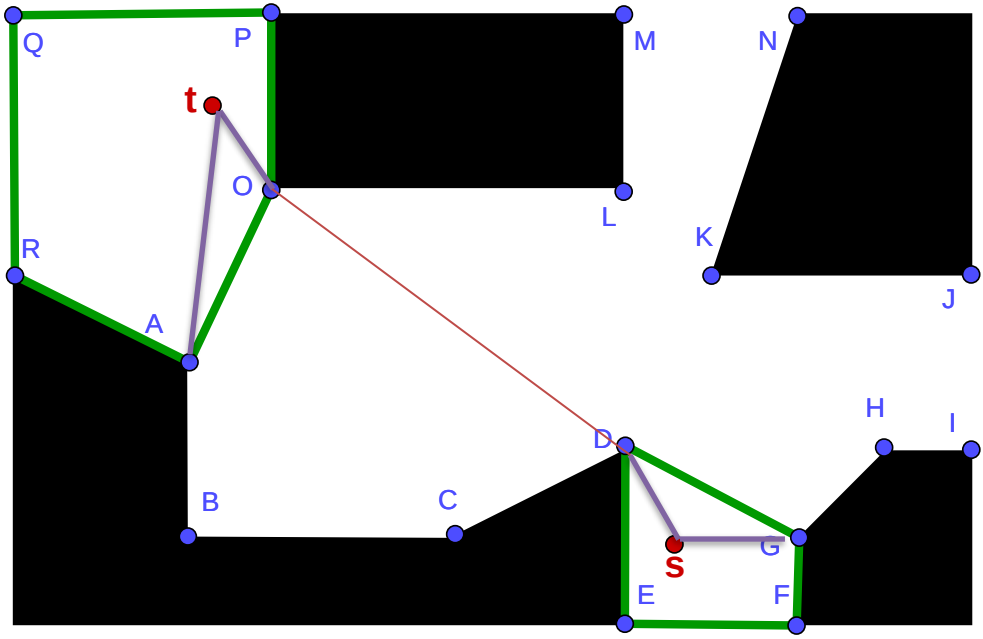
Previous work
End Point Search (EPS) [2]

- Distance oracle – CPD [3]
- EPS
- Advantages and disadvantages?



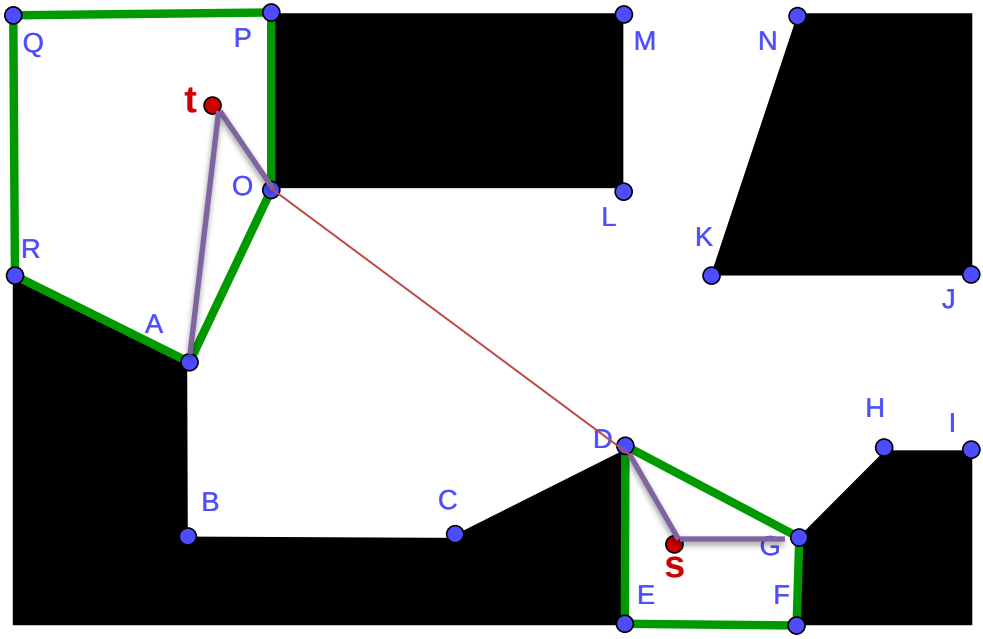
Previous work
End Point Search (EPS) [2]

- Distance oracle – CPD [3]
- EPS
- Advantage: 10x faster than Polyanya



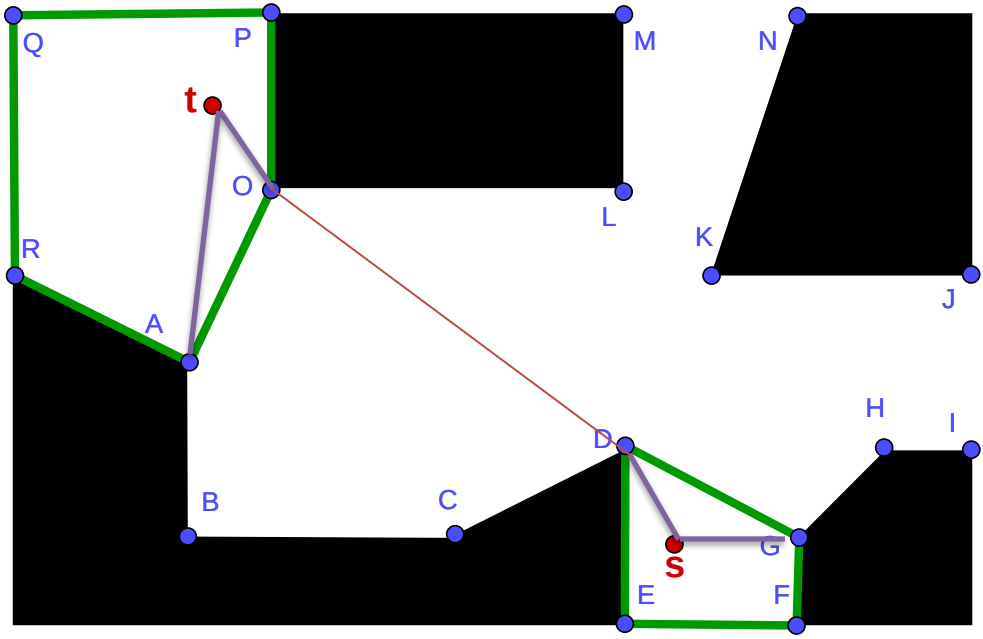
Previous work
End Point Search (EPS) [2]

- Distance oracle – CPD [3]
- EPS
- Advantage: 10x faster than Polyanya
- Disadvantage:
  - requires finding vertices visible from s and t



Previous work
End Point Search (EPS) [2]

- Distance oracle – CPD [3]
- EPS
- Advantage: 10x faster than Polyanya
- Disadvantage:
  - requires finding vertices visible from s and t
  - Pairwise comparisons



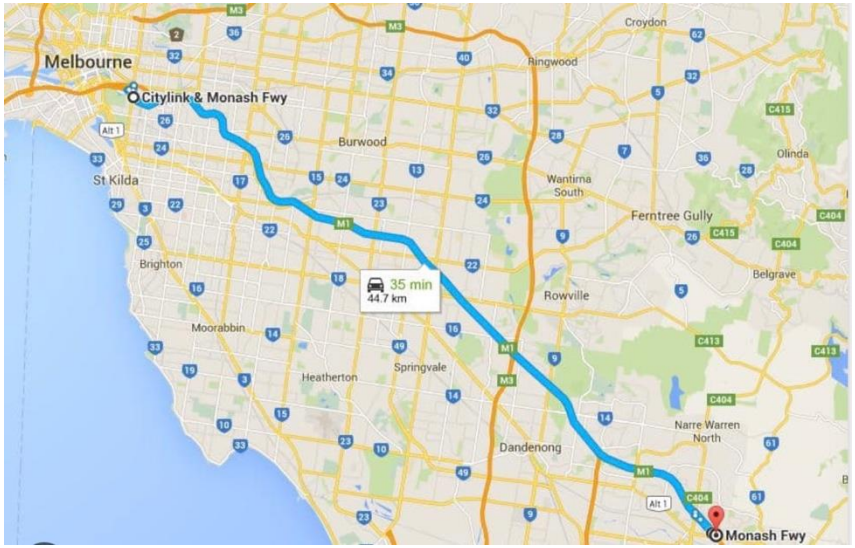
Outline

MONASH University

- Problem definition
- Previous works
  - Polyanya
  - End Point Search (EPS)
- Euclidean Hub Labeling
  - Preprocessing
  - Query processing
  - Optimisations
- Results

Hub Labeling [4]

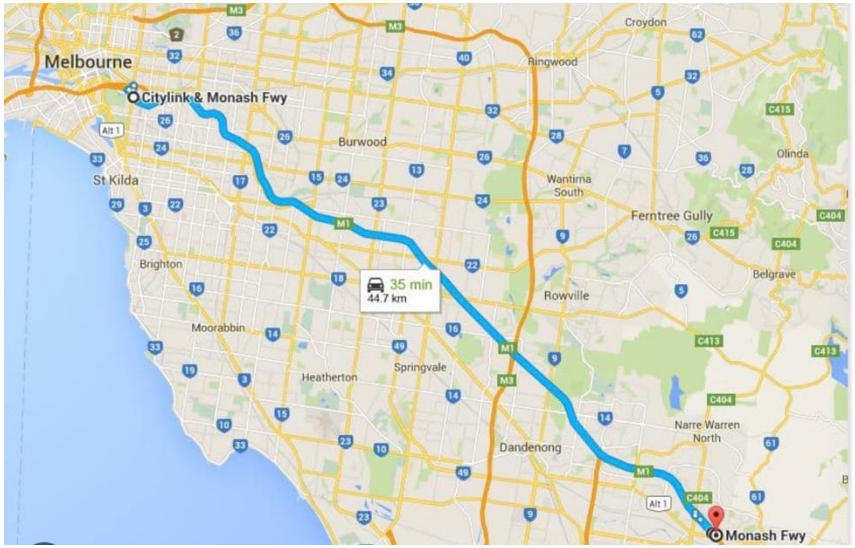
- Popular technique used in road networks



Melbourne road network with highway M1 highlighted

Hub Labeling [4]

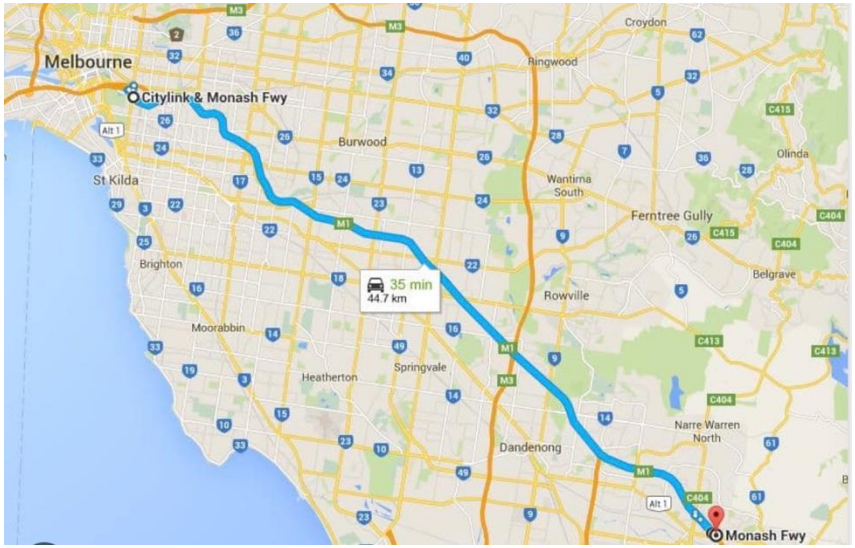
- Popular technique used in road networks
- Some nodes in a graph are more important than others



Melbourne road network with highway M1 highlighted

Hub Labeling [4]

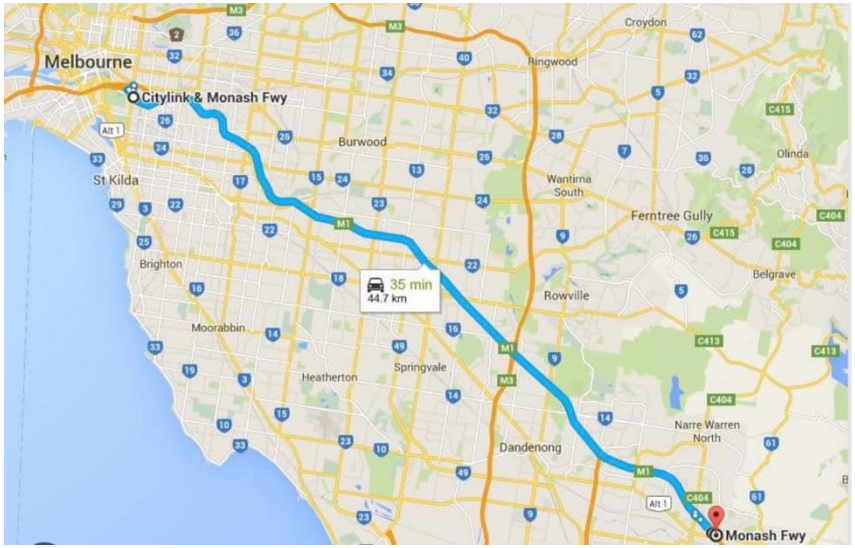
- Popular technique used in road networks
- Some nodes in a graph are more important than others
- Computing distance labels from the important hub node to all other nodes in the graph



Melbourne road network with highway M1 highlighted

Hub Labeling [4]

- Popular technique used in road networks
- Some nodes in a graph are more important than others
- Computing distance labels from the important hub node to all other nodes in the graph
- Efficient distance retrieval by merging the labels



Melbourne road network with highway M1 highlighted

Hub Labeling [4]

- Computes and stores a set of hub labels for each vertex in a graph

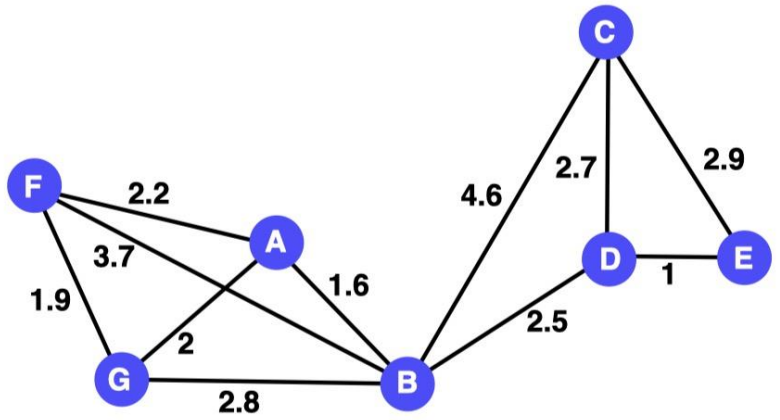


Figure 1: Visibility graph for the example in Figure 2

Vertex	Hub labels
A	(A, 0)
B	(A, 1.6), (B, 0)
C	(A, 6.2), (B, 4.6), (C, 0)
D	(A, 4.1), (B, 2.5), (C, 2.7), (D, 0)
E	(A, 5.1), (B, 3.5), (C, 2.9), (D, 1), (E, 0)
F	(A, 2.2), (B, 3.7), (F, 0)
G	(A, 2), (B, 2.8), (F, 1.9), (G,0)

Table 1: Hub labeling for the graph in Figure 1

Hub Labeling [4]

- Computes and stores a set of hub labels for each vertex in a graph
- Data structure: each hub label is tuple $(h_i, d_{ij}) \in H(v_j)$ with:
  - i) Hub vertex $h_i \in V$
  - ii) Shortest distance d_{ij}
  - iii) Predecessor

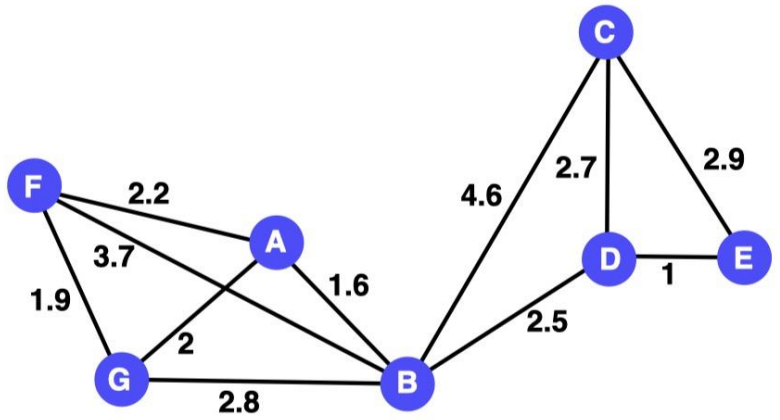


Figure 1: Visibility graph for the example in Figure 2

Vertex	Hub labels
A	(A, 0)
B	(A, 1.6), (B, 0)
C	(A, 6.2), (B, 4.6), (C, 0)
D	(A, 4.1), (B, 2.5), (C, 2.7), (D, 0)
E	(A, 5.1), (B, 3.5), (C, 2.9), (D, 1), (E, 0)
F	(A, 2.2), (B, 3.7), (F, 0)
G	(A, 2), (B, 2.8), (F, 1.9), (G,0)

Table 1: Hub labeling for the graph in Figure 1

Hub Labeling [4]

- Computes and stores a set of hub labels for each vertex in a graph
- Data structure
- Coverage property: at least one common hub node on the shortest path for any two reachable vertices

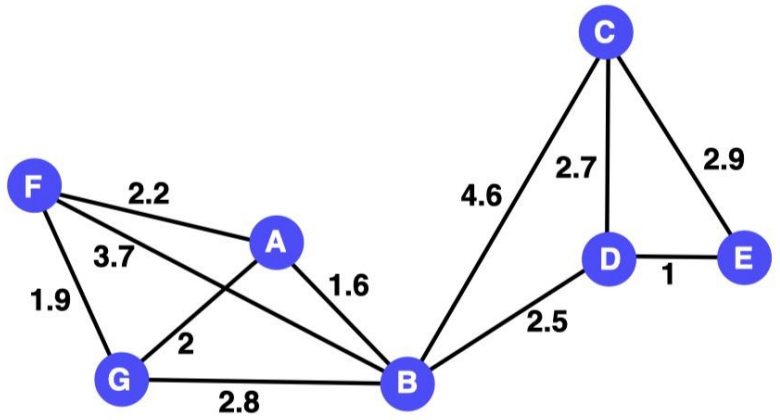


Figure 1: Visibility graph for the example in Figure 2

Vertex	Hub labels
A	(A, 0)
B	(A, 1.6), (B, 0)
C	(A, 6.2), (B, 4.6), (C, 0)
D	(A, 4.1), (B, 2.5), (C, 2.7), (D, 0)
E	(A, 5.1), (B, 3.5), (C, 2.9), (D, 1), (E, 0)
F	(A, 2.2), (B, 3.7), (F, 0)
G	(A, 2), (B, 2.8), (F, 1.9), (G,0)

Table 1: Hub labeling for the graph in Figure 1

Euclidean Hub Labeling
Hub Labeling [4]

- Computes and stores a set of hub labels for each vertex in a graph
- Data structure
- Coverage property
- The shortest path between any $v_s \in V$ and $v_t \in V$ can be computed by merging the hub labels of v_s and v_t.

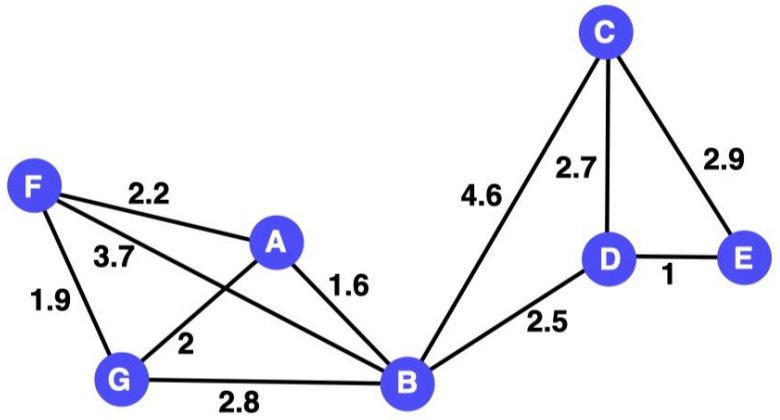


Figure 1: Visibility graph for the example in Figure 2

Vertex	Hub labels
A	(A, 0)
B	(A, 1.6), (B, 0)
C	(A, 6.2), (B, 4.6), (C, 0)
D	(A, 4.1), (B, 2.5), (C, 2.7), (D, 0)
E	(A, 5.1), (B, 3.5), (C, 2.9), (D, 1), (E, 0)
F	(A, 2.2), (B, 3.7), (F, 0)
G	(A, 2), (B, 2.8), (F, 1.9), (G,0)

Table 1: Hub labeling for the graph in Figure 1

Hub Labeling [4]

- Computes and stores a set of hub labels for each vertex in a graph
- Data structure
- Coverage property
- The shortest path between any $v_s \in V$ and $v_t \in V$ can be computed by merging the hub labels of v_s and v_t.

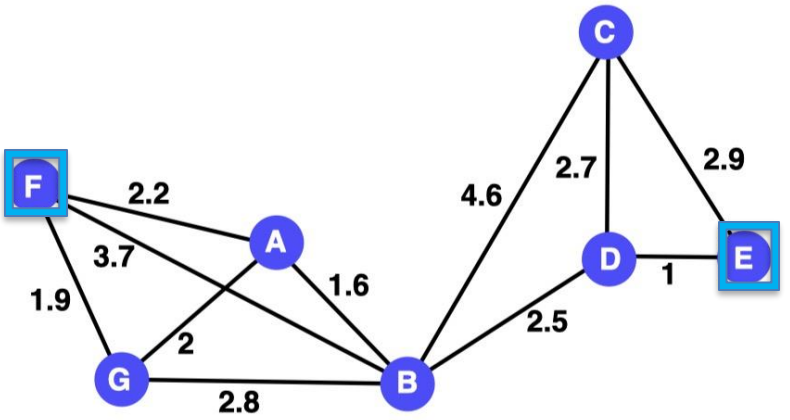


Figure 1: Visibility graph for the example in Figure 2

Vertex	Hub labels
A	(A, 0)
B	(A, 1.6), (B, 0)
C	(A, 6.2), (B, 4.6), (C, 0)
D	(A, 4.1), (B, 2.5), (C, 2.7), (D, 0)
E	(A, 5.1), (B, 3.5), (C, 2.9), (D, 1), (E, 0)
F	(A, 2.2), (B, 3.7), (F, 0)
G	(A, 2), (B, 2.8), (F, 1.9), (G,0)

Table 1: Hub labeling for the graph in Figure 1

Euclidean Hub Labeling
Hub Labeling [4]

- Computes and stores a set of hub labels for each vertex in a graph
- Data structure
- Coverage property
- The shortest path between any $v_s \in V$ and $v_t \in V$ can be computed by merging the hub labels of v_s and v_t.

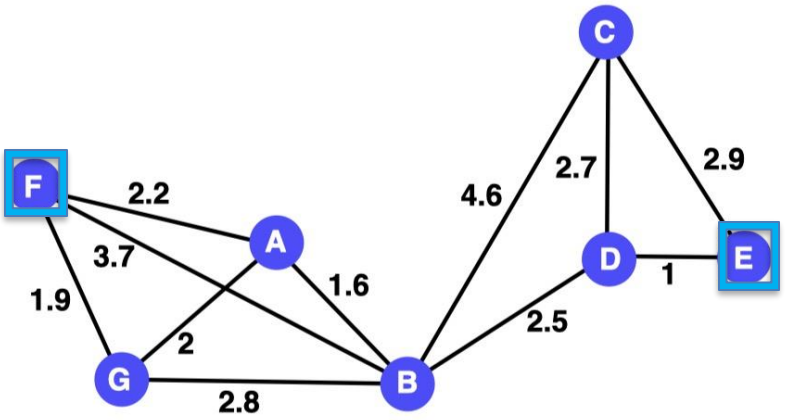


Figure 1: Visibility graph for the example in Figure 2

Vertex	Hub labels
A	(A, 0)
B	(A, 1.6), (B, 0)
C	(A, 6.2), (B, 4.6), (C, 0)
D	(A, 4.1), (B, 2.5), (C, 2.7), (D, 0)
E	(A, 5.1), (B, 3.5), (C, 2.9), (D, 1), (E, 0)
F	(A, 2.2), (B, 3.7), (F, 0)
G	(A, 2), (B, 2.8), (F, 1.9), (G,0)

Table 1: Hub labeling for the graph in Figure 1

Distance = E(A, 5.1) + F(A, 2.2)
= 7.3

Euclidean Hub Labeling
Hub Labeling [4]

MONASH University

- Computes and stores a set of hub labels for each vertex in a graph
- Data structure
- Coverage property
- The shortest path between any $v_s \in V$ and $v_t \in V$ can be computed by merging the hub labels of v_s and v_t.

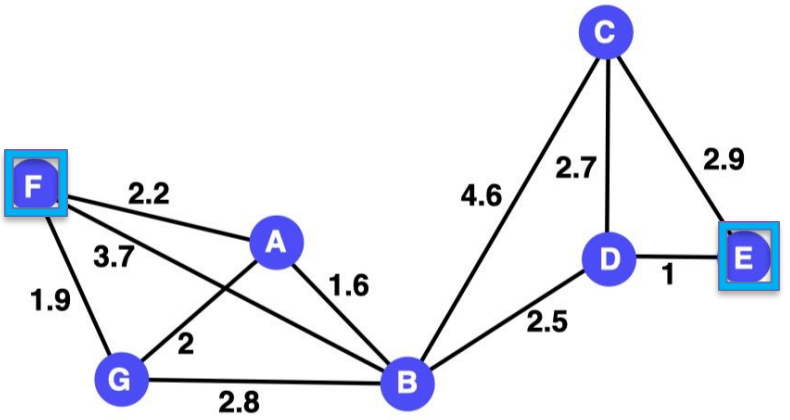


Figure 1: Visibility graph for the example in Figure 2

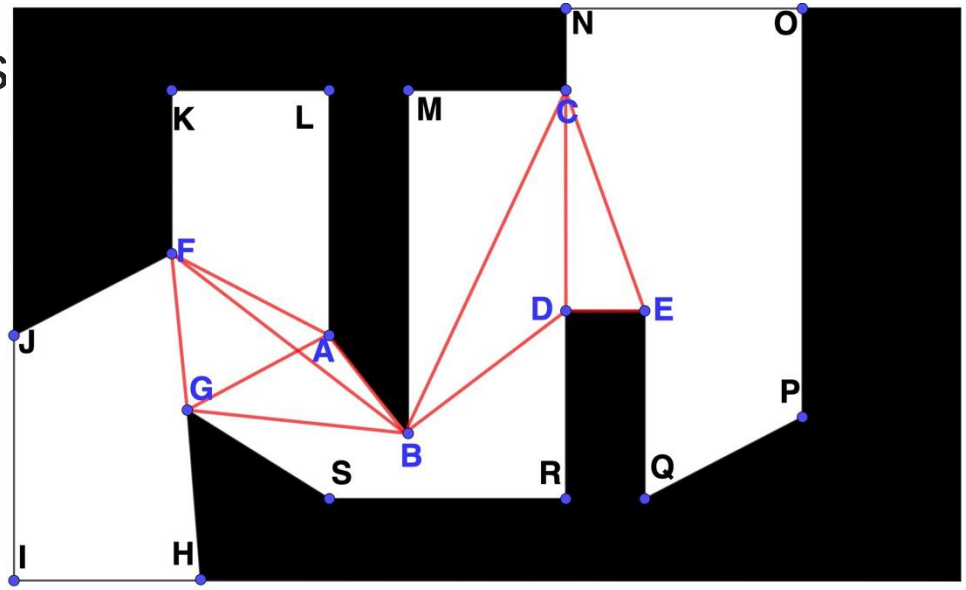
Vertex	Hub labels
A	(A, 0)
B	(A, 1.6), (B, 0)
C	(A, 6.2), (B, 4.6), (C, 0)
D	(A, 4.1), (B, 2.5), (C, 2.7), (D, 0)
E	(A, 5.1), (B, 3.5), (C, 2.9), (D, 1), (E, 0)
F	(A, 2.2), (B, 3.7), (F, 0)
G	(A, 2), (B, 2.8), (F, 1.9), (G,0)

Table 1: Hub labeling for the graph in Figure 1

Distance = E(B, 3.5) + F(B, 3.7)
= 7.2

Euclidean Hub Labeling
Preprocessing

1. Construct a visibility graph on convex vertices in the map



Euclidean Hub Labeling
Preprocessing

1. Construct a visibility graph on convex vertices in the map
2. Compute hub labels on top of the visibility graph

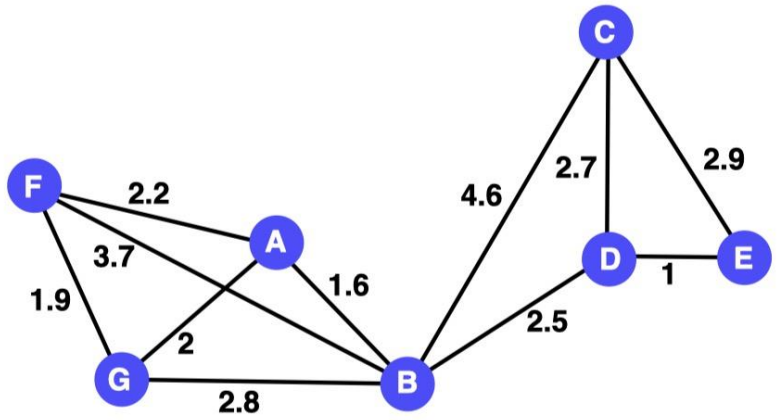


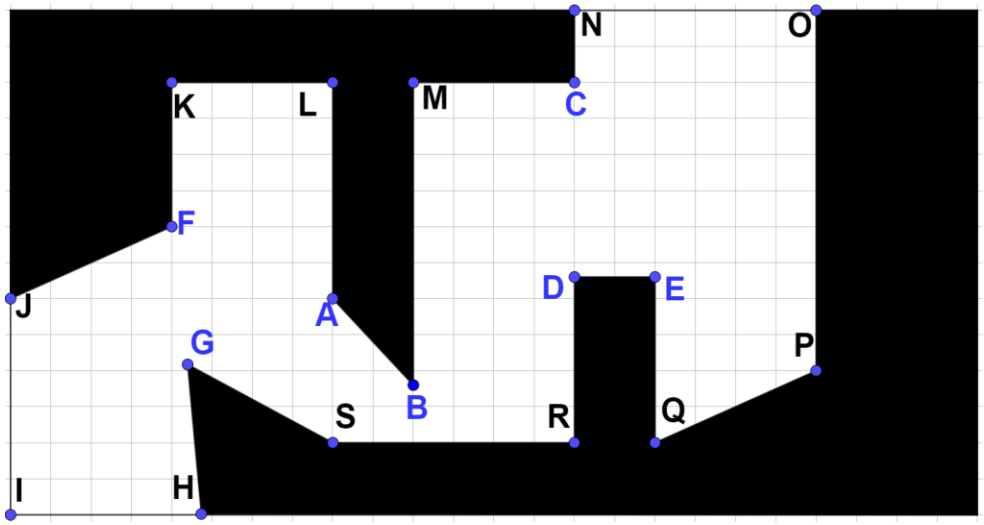
Figure 1: Visibility graph for the example in Figure 2

Vertex	Hub labels
A	(A, 0)
B	(A, 1.6), (B, 0)
C	(A, 6.2), (B, 4.6), (C, 0)
D	(A, 4.1), (B, 2.5), (C, 2.7), (D, 0)
E	(A, 5.1), (B, 3.5), (C, 2.9), (D, 1), (E, 0)
F	(A, 2.2), (B, 3.7), (F, 0)
G	(A, 2), (B, 2.8), (F, 1.9), (G,0)

Table 1: Hub labeling for the graph in Figure 1

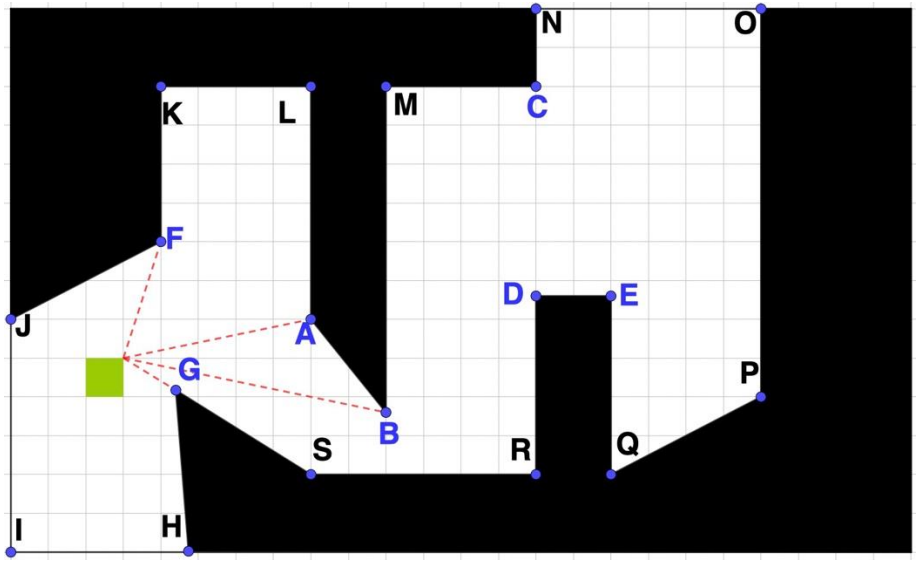
Euclidean Hub Labeling
Preprocessing

1. Construct a visibility graph on convex vertices in the map
2. Compute hub labels on top of the visibility graph
3. Superimpose a uniform grid covering the whole graph



Euclidean Hub Labeling Preprocessing

1. Construct a visibility graph on convex vertices in the map
2. Compute hub labels on top of the visibility graph
3. Superimpose a uniform grid covering the whole graph
4. For each cell c, copy labels of vertices visible from it (via labels)



Vertex	Hub labels
A	(A, 0)
B	(A, 1.6), (B, 0)
C	(A, 6.2), (B, 4.6), (C, 0)
D	(A, 4.1), (B, 2.5), (C, 2.7), (D, 0)
E	(A, 5.1), (B, 3.5), (C, 2.9), (D, 1), (E, 0)
F	(A, 2.2), (B, 3.7), (F, 0)
G	(A, 2), (B, 2.8), (F, 1.9), (G,0)

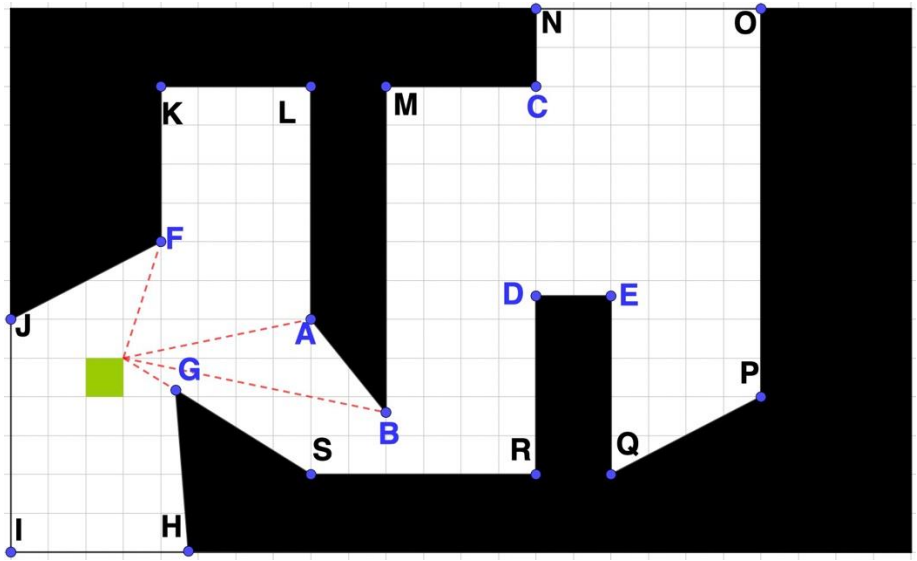
Hub nodes	Via labels
A	(A, 0), (B, 1.6), (G, 2), (F, 2.2)
B	(B, 0), (G, 2.8), (F, 3.7)
F	(F, 0), (G, 1.9)
G	(G, 0)

Table 2: Hub labels and via labels for the figure above

Euclidean Hub Labeling Preprocessing

MONASH University

1. Construct a visibility graph on convex vertices in the map
2. Compute hub labels on top of the visibility graph
3. Superimpose a uniform grid covering the whole graph
4. For each cell c, copy labels of vertices visible from it (via labels)



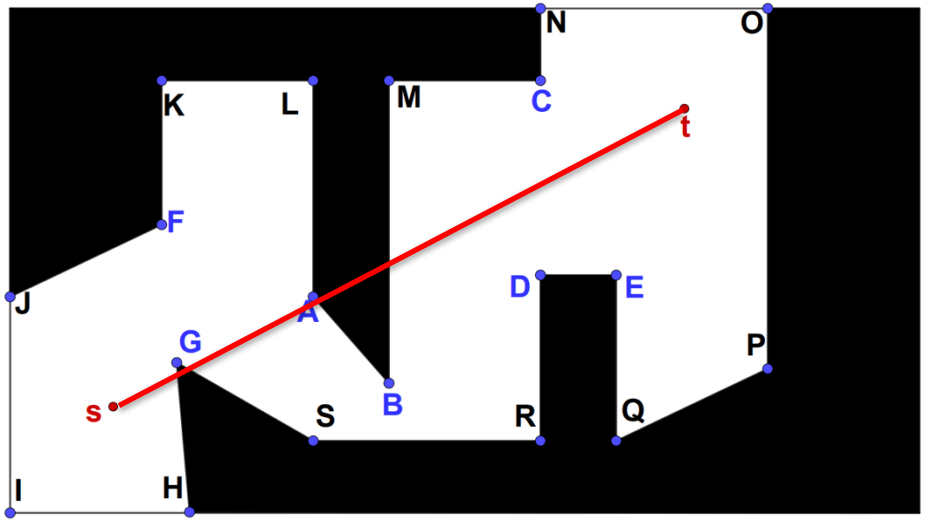
Vertex	Hub labels
A	(A, 0)
B	(A, 1.6), (B, 0)
C	(A, 6.2), (B, 4.6), (C, 0)
D	(A, 4.1), (B, 2.5), (C, 2.7), (D, 0)
E	(A, 5.1), (B, 3.5), (C, 2.9), (D, 1), (E, 0)
F	(A, 2.2), (B, 3.7), (F, 0)
G	(A, 2), (B, 2.8), (F, 1.9), (G,0)

Hub nodes	Via labels
A	(A, 0), (B, 1.6), (G, 2), (F, 2.2)
B	(B, 0), (G, 2.8), (F, 3.7)
F	(F, 0), (G, 1.9)
G	(G, 0)

Table 2: Hub labels and via labels for the figure above

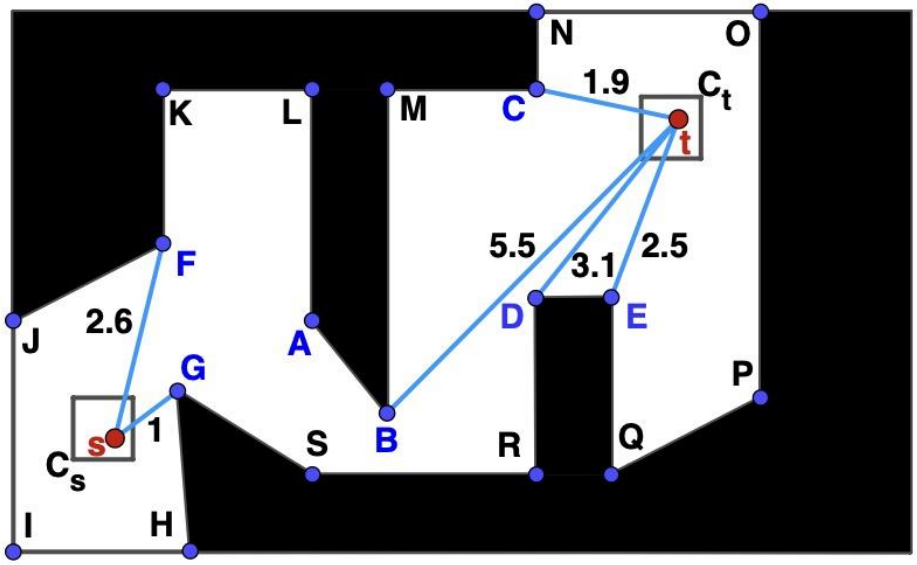
Euclidean Hub Labeling
Query processing

1. Check if s and t are visible to each other



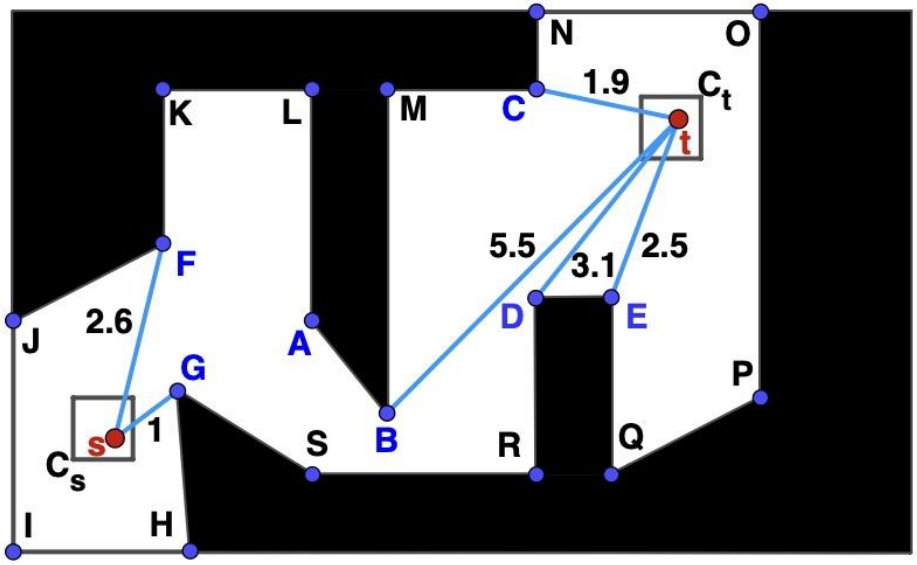
Euclidean Hub Labeling
Query processing

1. Check if s and t are visible to each other
2. Locate the grid cells of s and t, retrieving all the labels within it



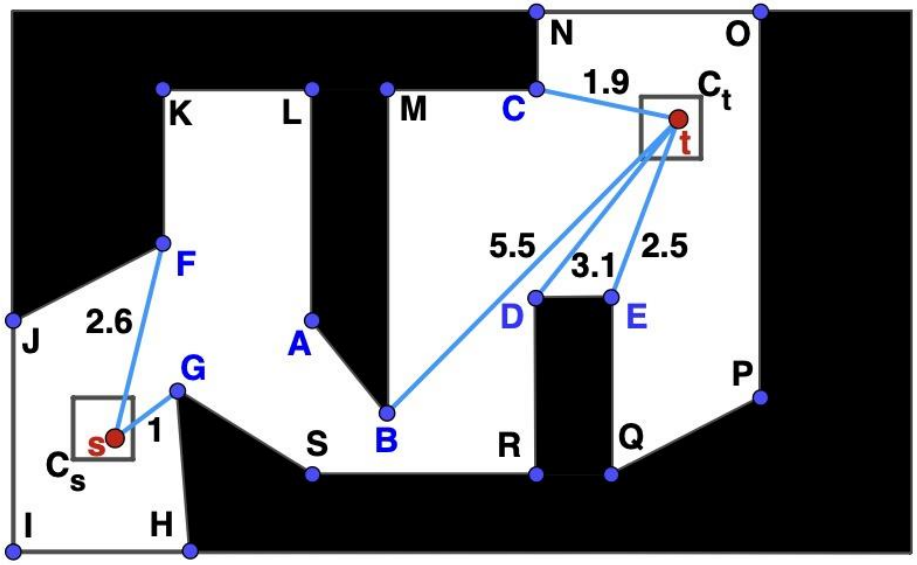
Euclidean Hub Labeling
Query processing

1. Check if s and t are visible to each other
2. Locate the grid cells of s and t, retrieving all the labels within it
3. Scan the via labels when there is a common hub node found and calculate the distance

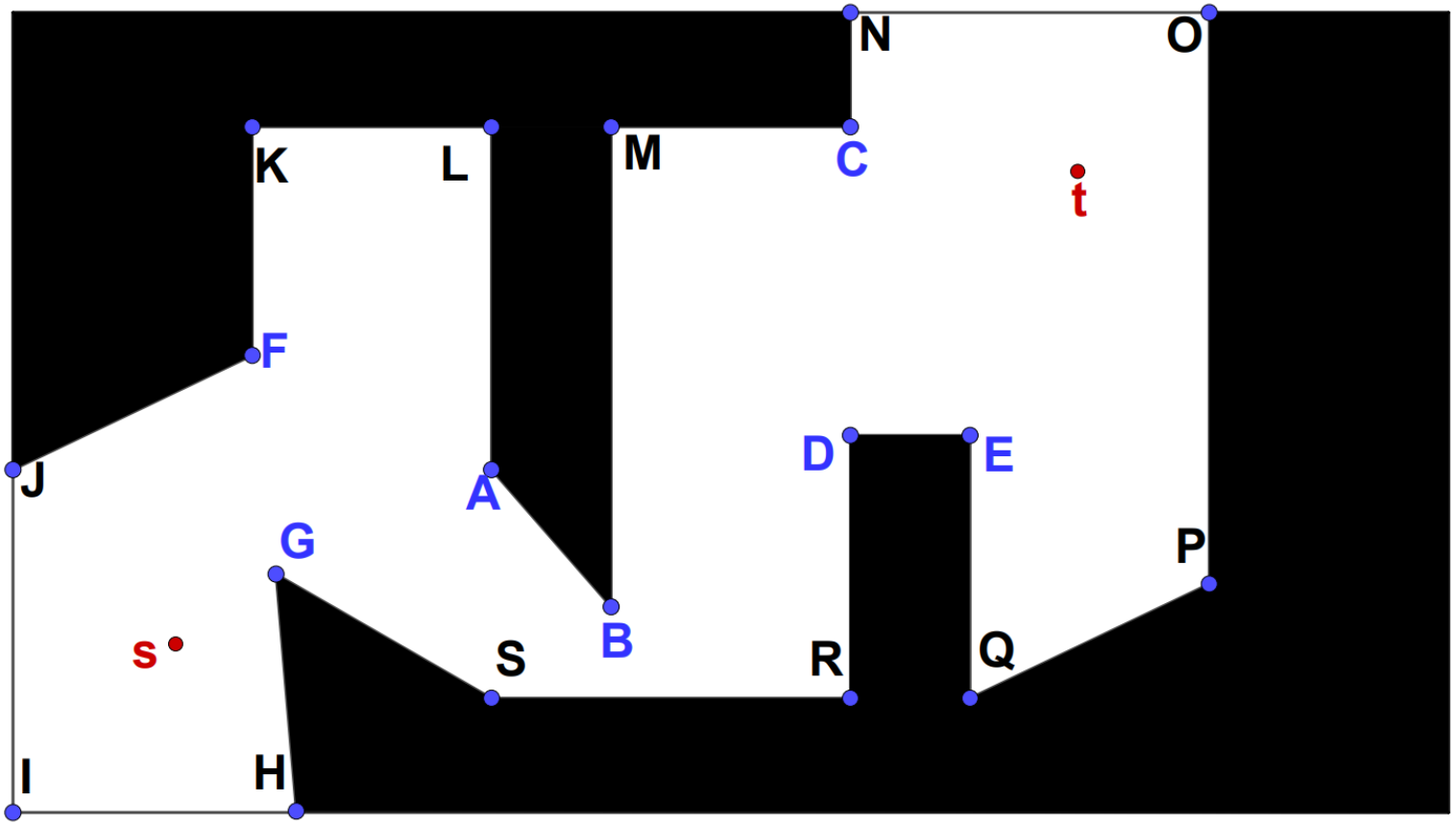


Euclidean Hub Labeling
Query processing

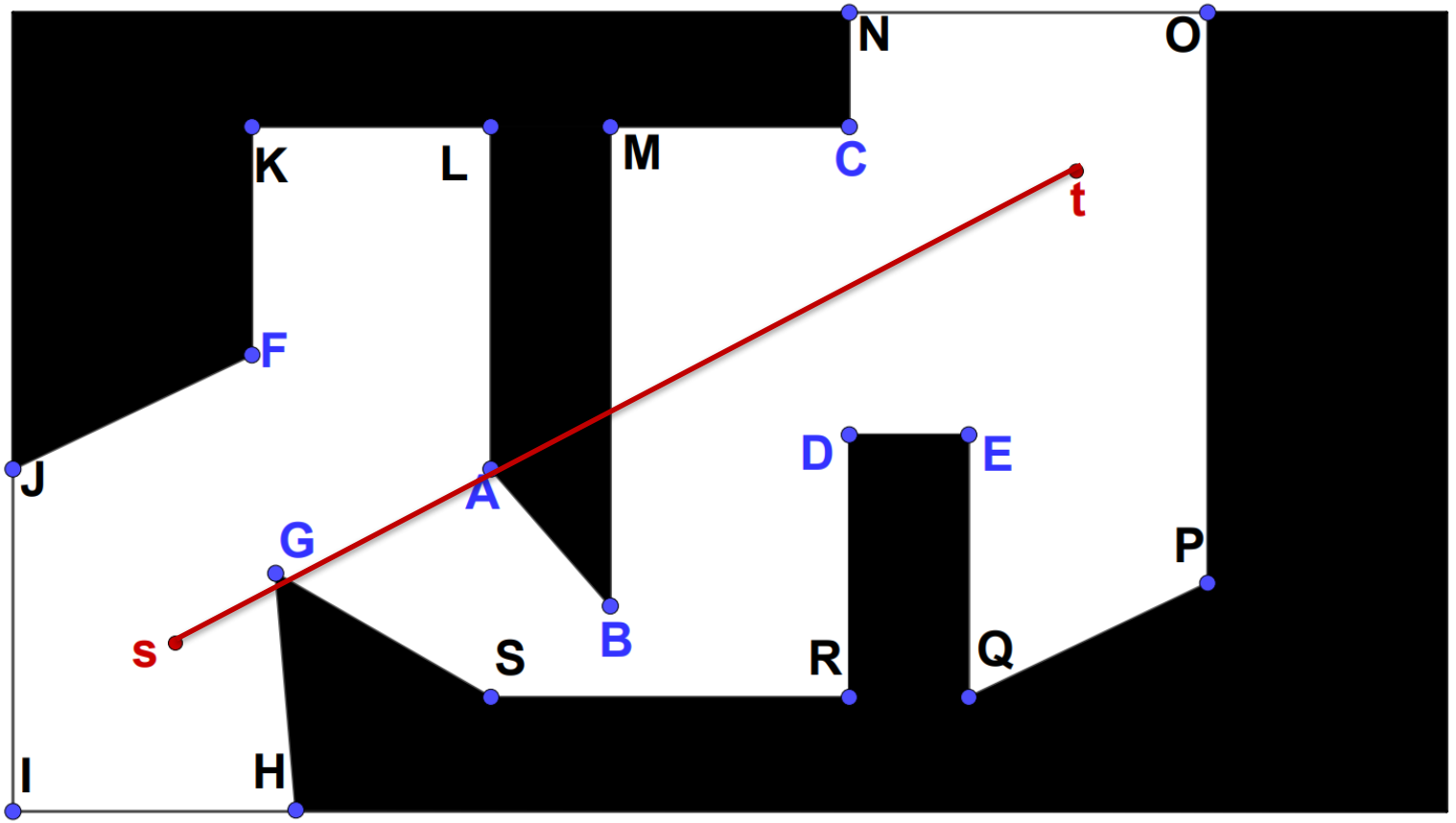
1. Check if s and t are visible to each other
2. Locate the grid cells of s and t, retrieving all the labels within it
3. Scan the via labels when there is a common hub node found and calculate the distance
4. The minimum distance is the shortest path



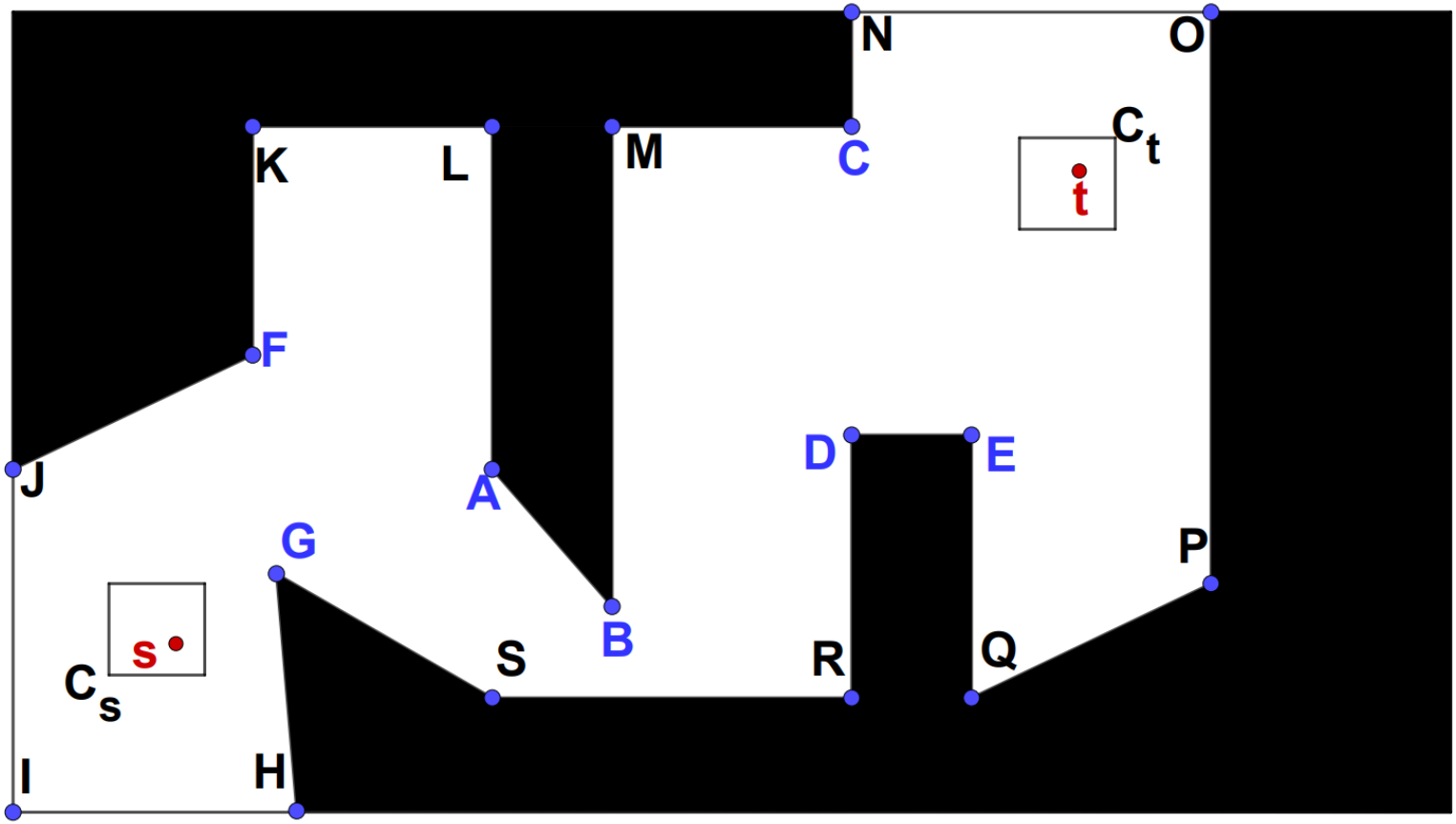
Euclidean Hub Labeling
Running example



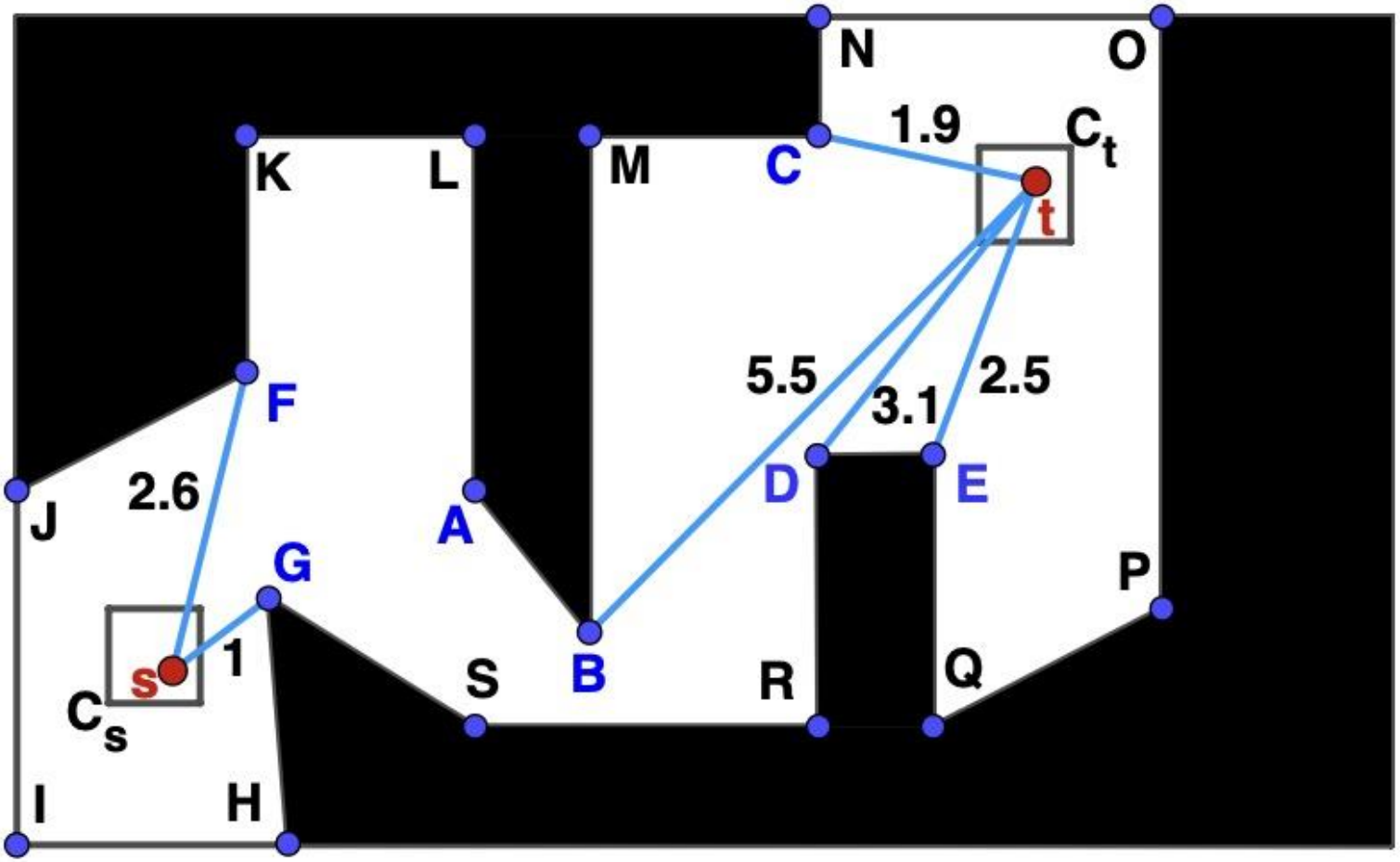
Euclidean Hub Labeling
Running example



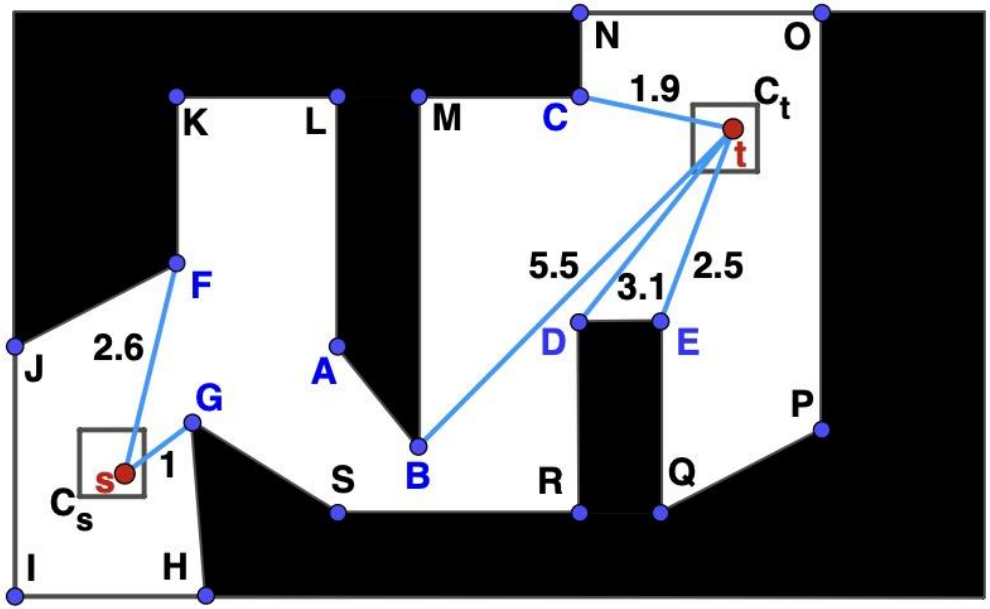
Euclidean Hub Labeling
Running example



Euclidean Hub Labeling
Running example



Euclidean Hub Labeling
Running example



Current minimum distance = infinity

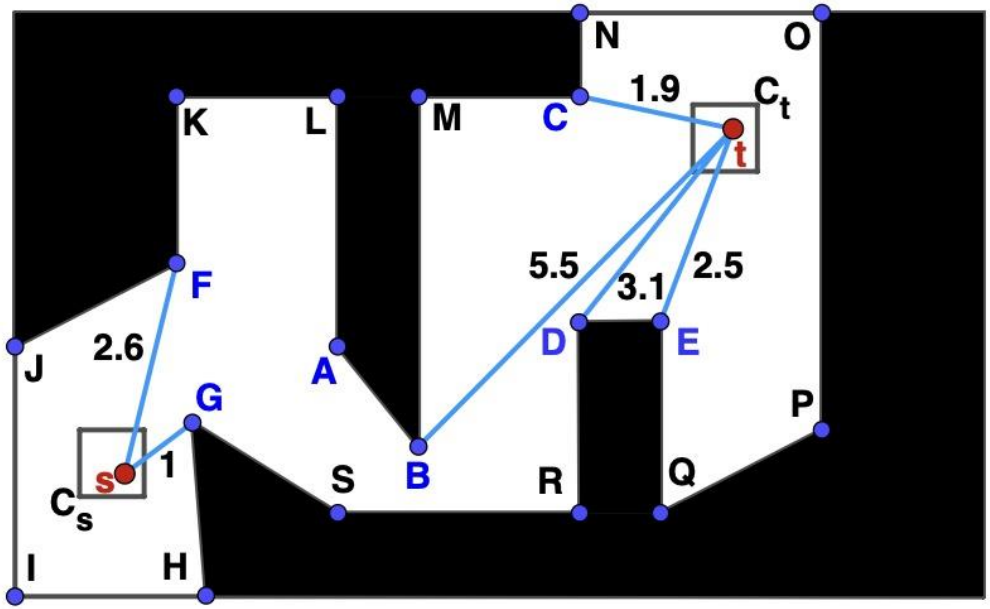
$H(c_s)$	Via Labels $VL_{h_i}(c_s)$
A:	(A, 0), (F, 2.2), (G, 2)
B:	(F, 3.7), (G, 2.8)
F:	(F, 0), (G, 1.9)
G:	(G, 0)

$H(c_t)$	Via Labels $VL_{h_i}(c_t)$
A:	(B, 1.6), (C, 6.2), (D, 4.1), (E, 5.1)
B:	(B, 0), (C, 4.6), (D, 2.5), (E, 3.5)
C:	(C, 0), (D, 2.7), (E, 2.9)
D:	(D, 0), (E, 1)
E:	(E, 0)

Dist(s,A) = 1 + 2
= 3

Dist(t,A) = 5.5 + 1.6
= 7.1

Euclidean Hub Labeling
Running example



Current minimum distance = 10.1

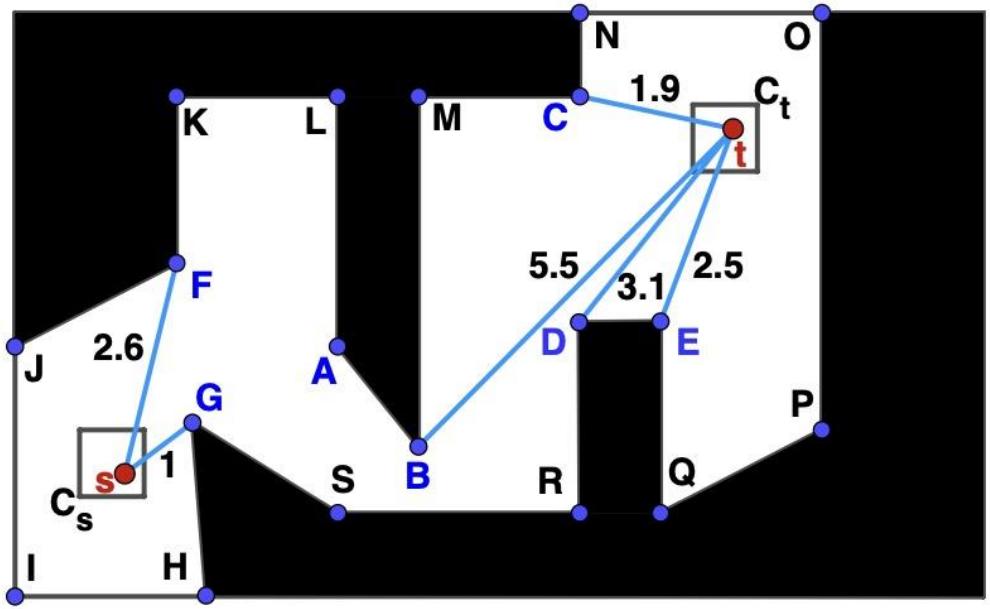
$H(c_s)$	Via Labels $VL_{h_i}(c_s)$
A:	(A, 0), (F, 2.2), (G, 2)
B:	(F, 3.7), (G, 2.8)
F:	(F, 0), (G, 1.9)
G:	(G, 0)

$H(c_t)$	Via Labels $VL_{h_i}(c_t)$
A:	(B, 1.6), (C, 6.2), (D, 4.1), (E, 5.1)
B:	(B, 0), (C, 4.6), (D, 2.5), (E, 3.5)
C:	(C, 0), (D, 2.7), (E, 2.9)
D:	(D, 0), (E, 1)
E:	(E, 0)

Dist(s,A) = 1 + 2
= 3

Dist(t,A) = 5.5 + 1.6
= 7.1

Euclidean Hub Labeling
Running example



Current minimum distance = 10.1

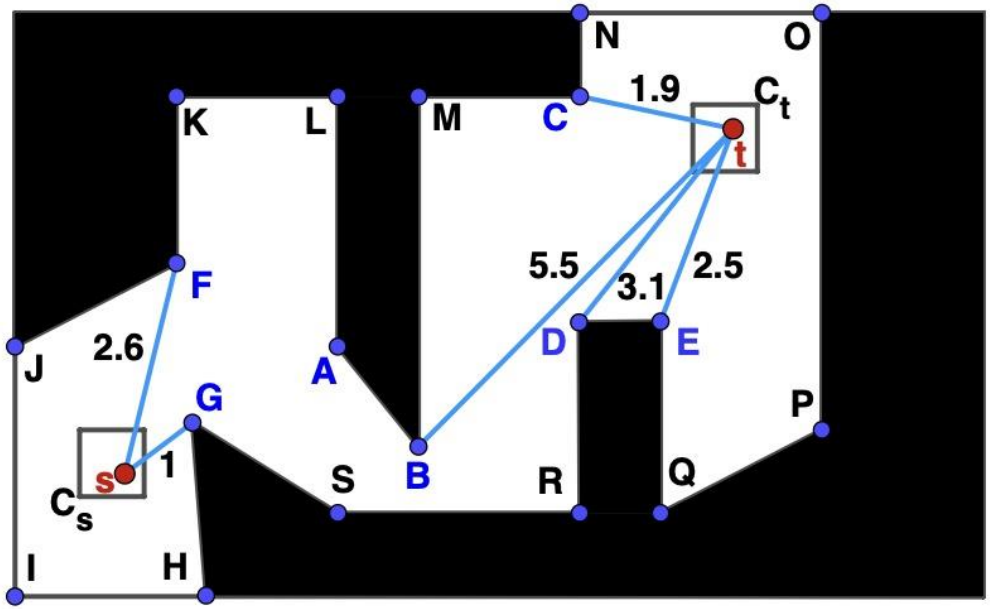
$H(c_s)$	Via Labels $VL_{h_i}(c_s)$
A:	(A, 0), (F, 2.2), (G, 2)
B:	(F, 3.7), (G, 2.8)
F:	(F, 0), (G, 1.9)
G:	(G, 0)

$H(c_t)$	Via Labels $VL_{h_i}(c_t)$
A:	(B, 1.6), (C, 6.2), (D, 4.1), (E, 5.1)
B:	(B, 0), (C, 4.6), (D, 2.5), (E, 3.5)
C:	(C, 0), (D, 2.7), (E, 2.9)
D:	(D, 0), (E, 1)
E:	(E, 0)

Dist(s,B) = 1 + 2.8
= 3.8

Dist(t,B) = 5.5

Euclidean Hub Labeling
Running example



Current minimum distance = 9.3

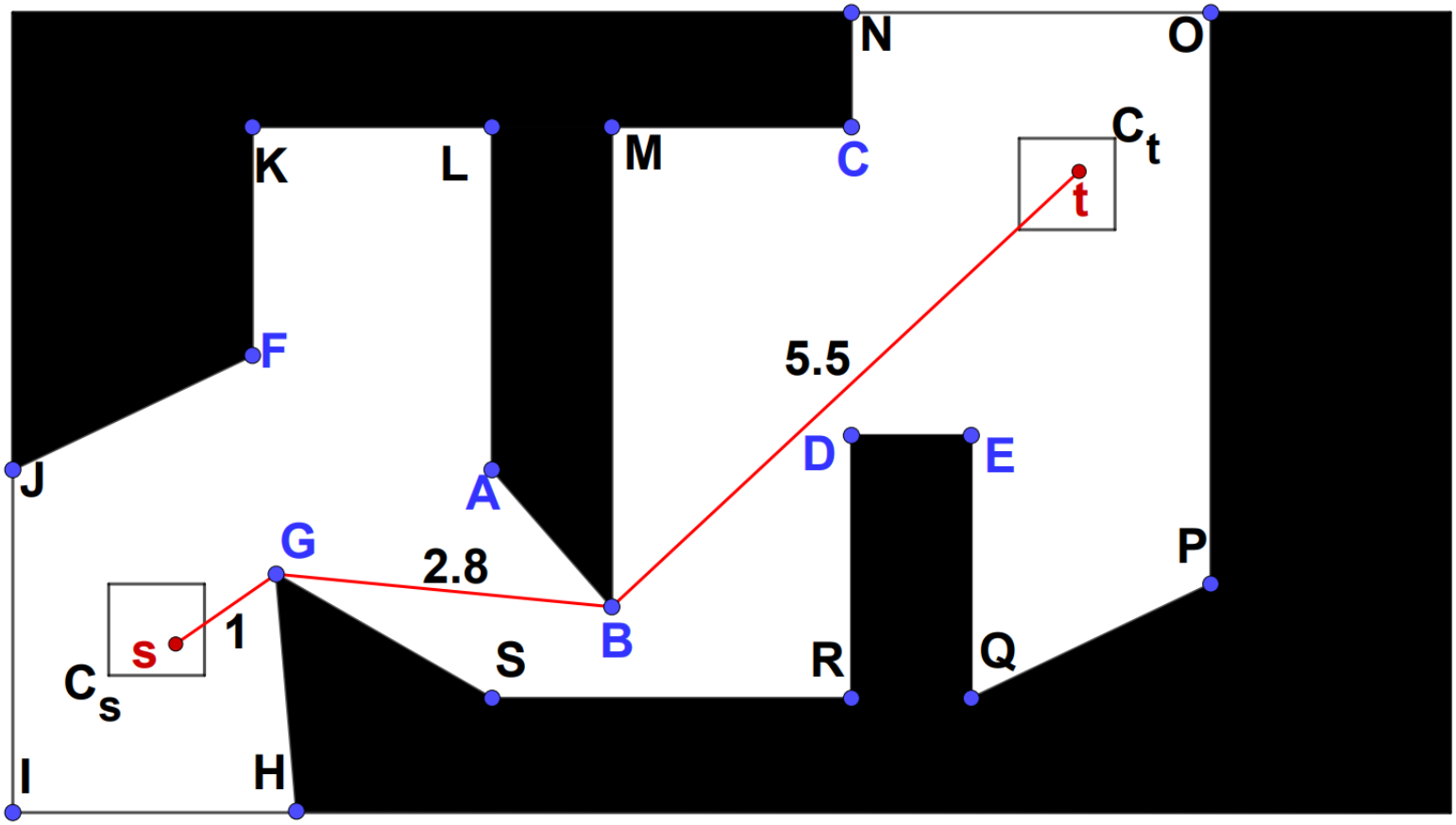
$H(c_s)$	Via Labels $VL_{h_i}(c_s)$
A:	(A, 0), (F, 2.2), (G, 2)
B:	(F, 3.7), (G, 2.8)
F:	(F, 0), (G, 1.9)
G:	(G, 0)

$H(c_t)$	Via Labels $VL_{h_i}(c_t)$
A:	(B, 1.6), (C, 6.2), (D, 4.1), (E, 5.1)
B:	(B, 0), (C, 4.6), (D, 2.5), (E, 3.5)
C:	(C, 0), (D, 2.7), (E, 2.9)
D:	(D, 0), (E, 1)
E:	(E, 0)

Dist(s,B) = 1 + 2.8
= 3.8

Dist(t,B) = 5.5

Euclidean Hub Labeling
Running example



Shortest distance = 9.3
Shortest path = s, G, B, t

Euclidean Hub Labeling Optimisations

1. Non-taut region pruning
2. Non-taut labels pruning
3. Distance bound pruning
4. Dead-end labels pruning

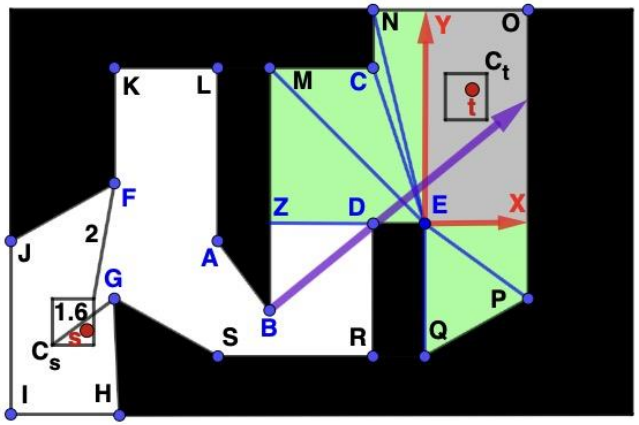


Figure 3: Illustrating pruning rules

$H(c_s)$	Via Labels $VL_{h_i}(c_s)$
A:	~~(A, 0)~~, ~~(F, 2.2)~~, ~~(G, 2)~~
B:	~~(F, 3.7)~~, (G, 2.8)
F:	(F, 0), ~~(G, 1.9)~~
G:	(G, 0)

$H(c_t)$	Via Labels $VL_{h_i}(c_t)$
A:	(B, 1.6), ~~(C, 6.2)~~, ~~(D, 4.1)~~, ~~(E, 5.1)~~
B:	(B, 0), ~~(C, 4.6)~~, ~~(D, 2.5)~~, ~~(E, 3.5)~~
C:	~~(C, 0)~~, ~~(D, 2.7)~~, ~~(E, 2.9)~~
D:	(D, 0), ~~(E, 1)~~
E:	~~(E, 0)~~

Table 3: Labels pruned by different pruning rules (PR) are shown in different colors. PR1: blue, PR2: orange, PR3: red, PR4: purple. The hub nodes with all labels pruned are struckthrough gray.

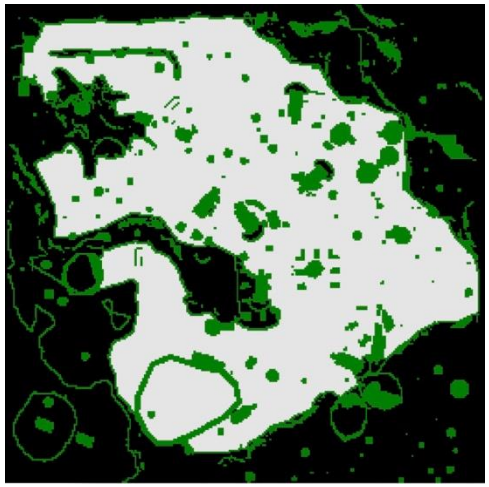
Outline

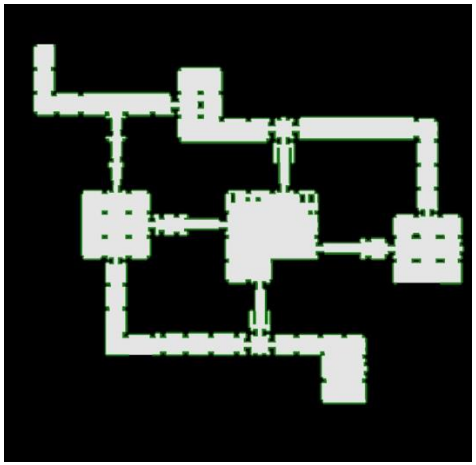
MONASH University

- Problem definition
- Previous works
  - Polyanya
  - End Point Search (EPS)
- Euclidean Hub Labeling
  - Preprocessing
  - Query processing
  - Optimisations
- Results

Results
Experimental setup

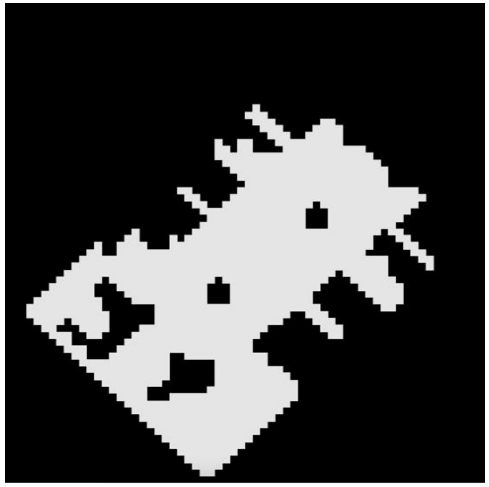
- Benchmarks: widely used game map benchmarks [5]

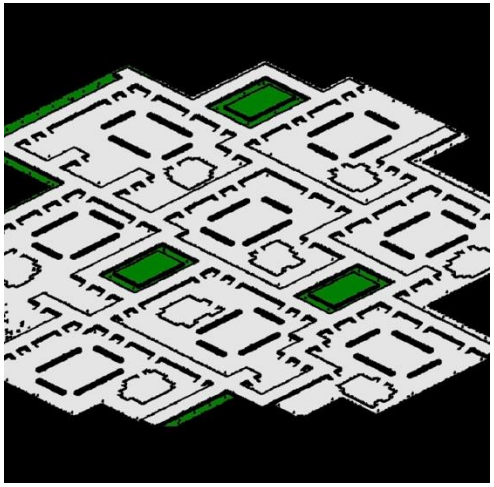




	#Maps	# Queries	#Vertices	#Convex Vertices
DAO	156	159,464	1727.6	926.5
DA	67	68,150	1182.9	610.8
BG	75	93,160	1294.4	667.7
SC	75	198,224	11487.5	5792.7

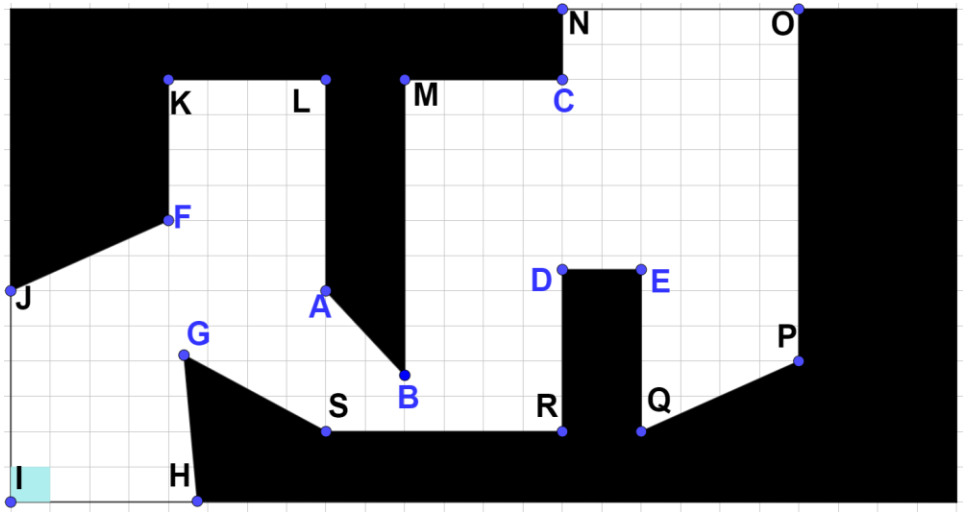
Table 4: Total number of maps and queries, and average number of vertices and convex vertices in each benchmark.



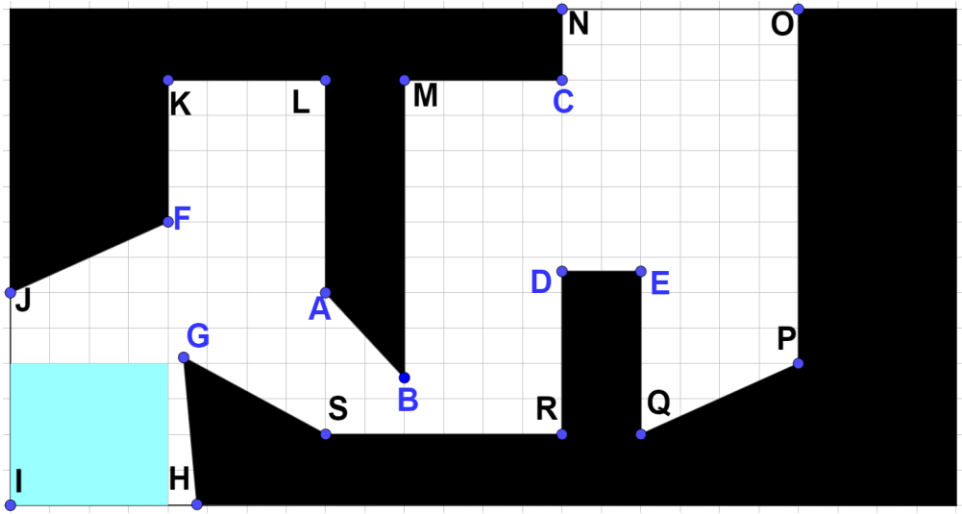


Results
Experimental setup

- Benchmarks [5]
- Grid sizes: a parameter of the superimposed grid sizes
  - EHL 1x
  - EHL 4x
  - EHL 16x
  - EHL 64x



EHL 1x



EHL 4x

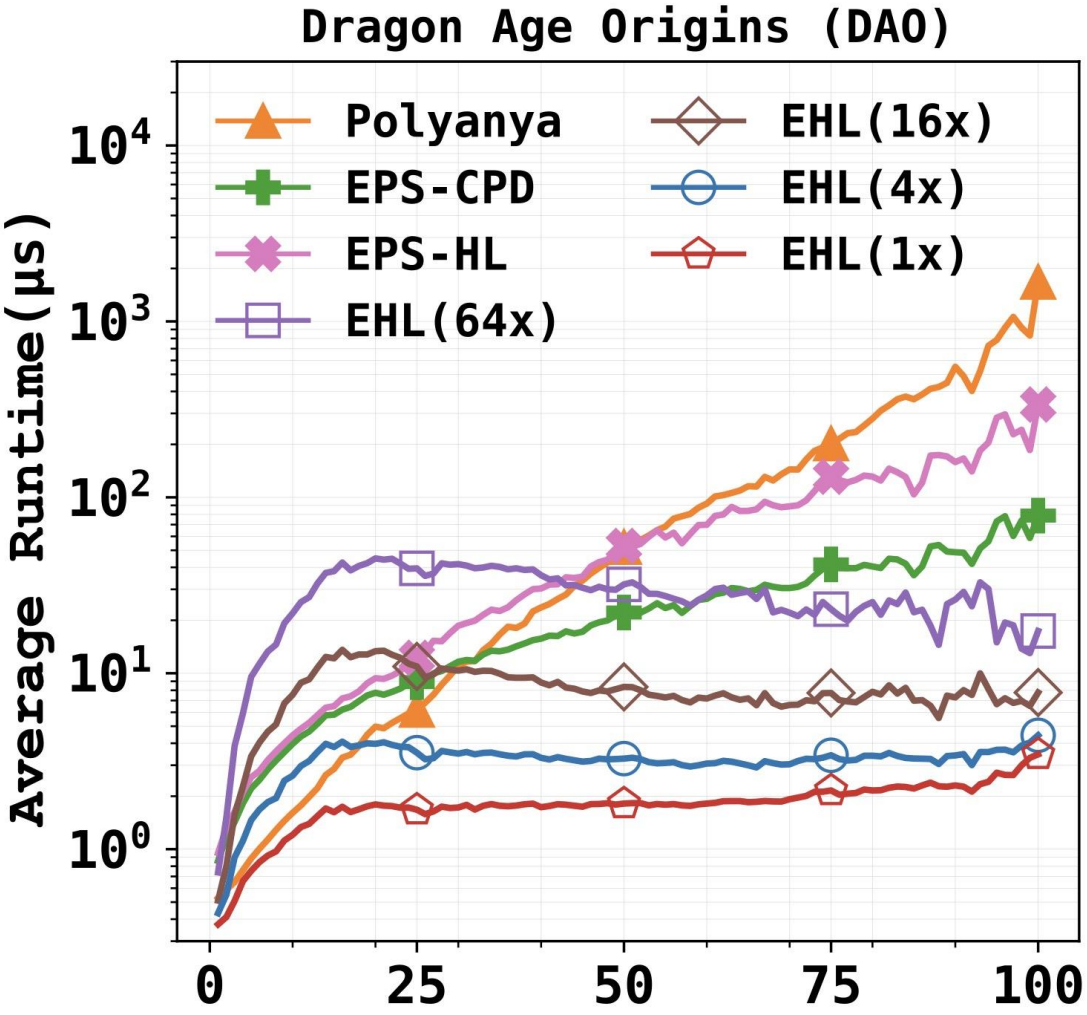
Results
Preprocessing costs

MONASH University

Approach	Build Time (Secs)	Memory (MB)
EHL (1x)	4.41	137
EHL (4x)	0.35	27.5
EHL (16x)	0.05	7.1
EHL (64x)	0.02	2.4
EPS-CPD	0.11	0.21
EPS-HL	0.05	0.78

Average cost (build time and memory) for DAO benchmark for EHL and EPS

Results
Query performance



The x-axis shows the percentile ranks of queries in number of node expansions needed by A* search to solve them.

MONASH University

Thank you for listening ☺

Contact us!

EHL Paper

References

[1] Michael Cui, Daniel Harabor, Alban Grastien, *"Compromise-free Pathfinding on a Navigation Mesh."* In *IJCAI*, 2017.

[2] Bojie Shen, Muhammad Aamir Cheema, Daniel D. Harabor, and Peter J. Stuckey. "Euclidean pathfinding with compressed path databases." In *IJCAI* 2021.

[3] Adi Botea. *"Ultra-fast optimal pathfinding without runtime search.* In AIIDE 2011.

[4] Ye Li, Leong Hou, Man Lung Yiu, and Ngai Meng Kou. "An experimental study on hub labeling based shortest path algorithms." *In PVLDB 2017*.

[5] Nathan Sturtevant. *"Benchmarks for Grid-Based Pathfinding". In IEEE Transactions on Computational Intelligence and AI in Games, 4(2): 144–148 2012.*