MONASH University

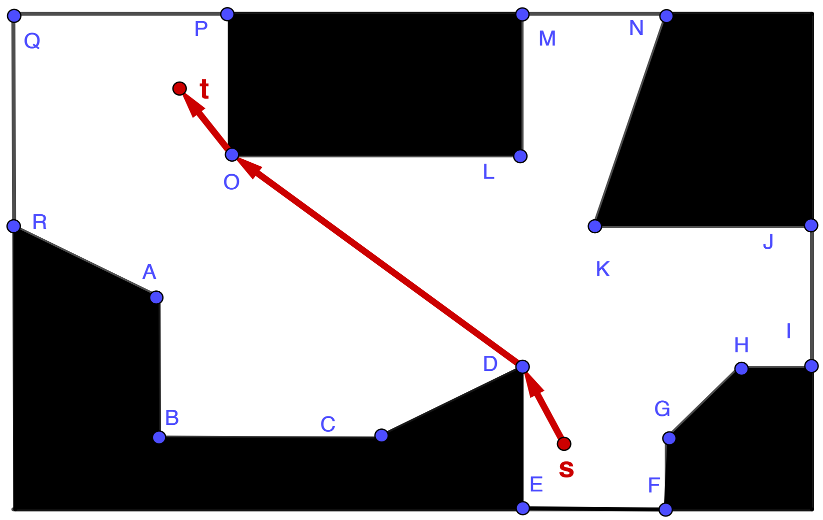
Euclidean Pathfinding with Compressed Path Databases IJCAI 2020

Bojie Shen, Muhammad Aamir Cheema, Daniel D. Harabor, Peter J. Stuckey

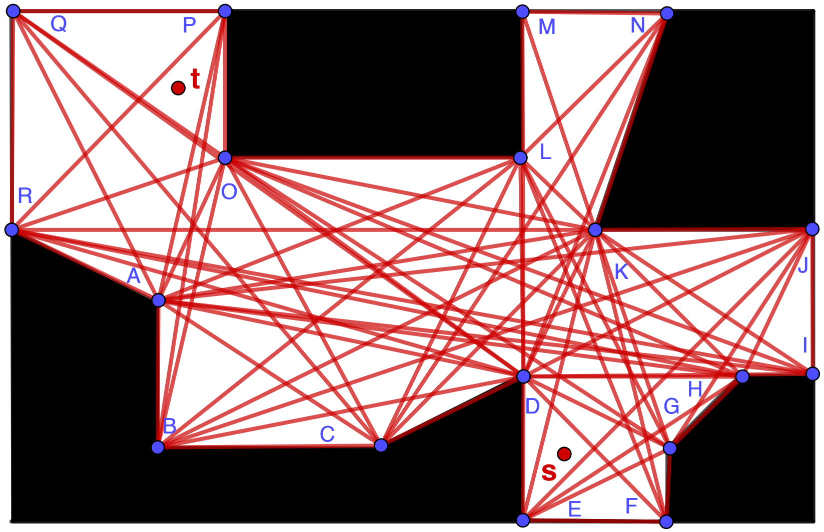
Monash University

Euclidean Shortest Path Problem

- Euclidean plane:
  - Polygonal obstacles
    - A set of vertices
    - A set of closed edges
- Pathfinding:
  - Given a start s and target t
  - Optimal non-obstructed path.

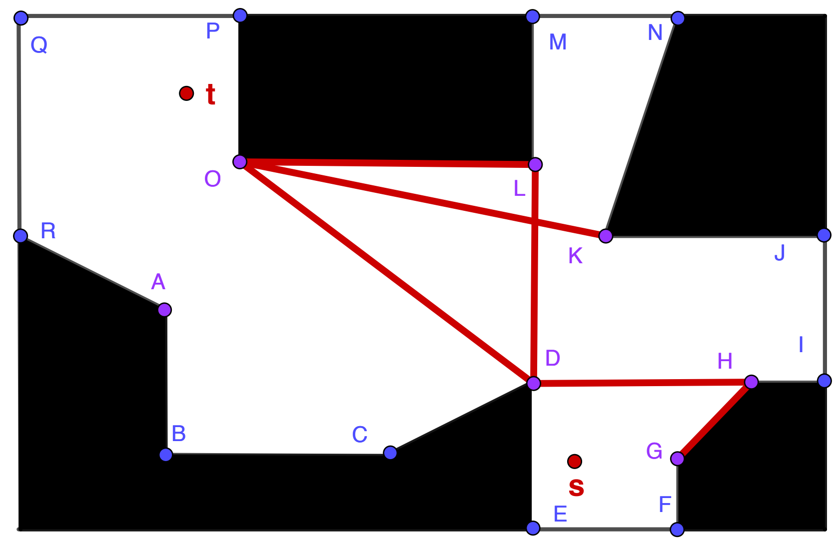


- Visibility graph [1] :
  - Connecting co-visible vertices

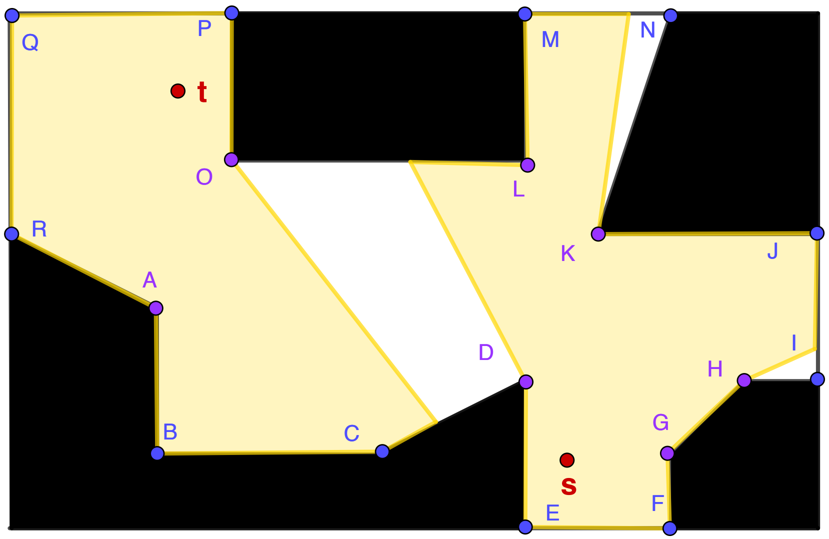


Sparse visibility graph [2]

- Visibility graph [1]
- Sparse visibility graph [2] :
  - Remove unnecessary vertices and edges



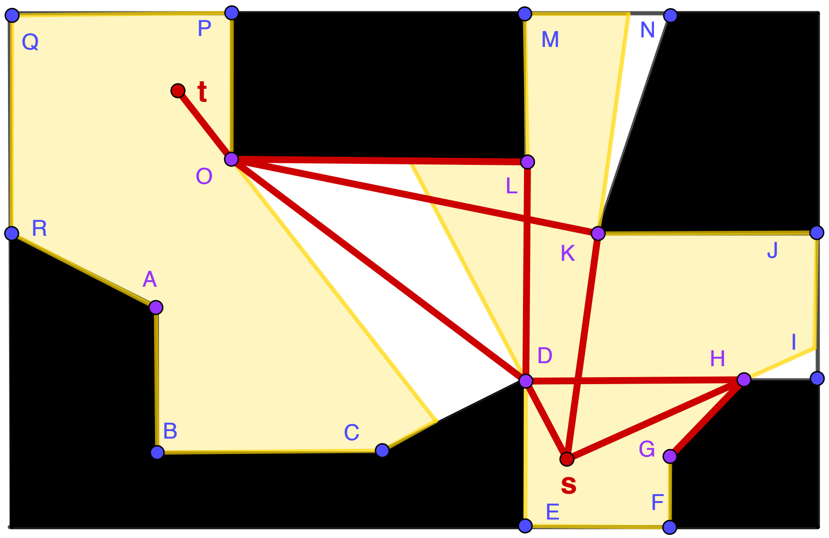
- Visibility graph [1]
- Sparse visibility graph [2]
- Pathfinding:
  - Full insertion
  - Heuristic search



Shaded yellow areas correspond to the area visible from s and t

MONASH University

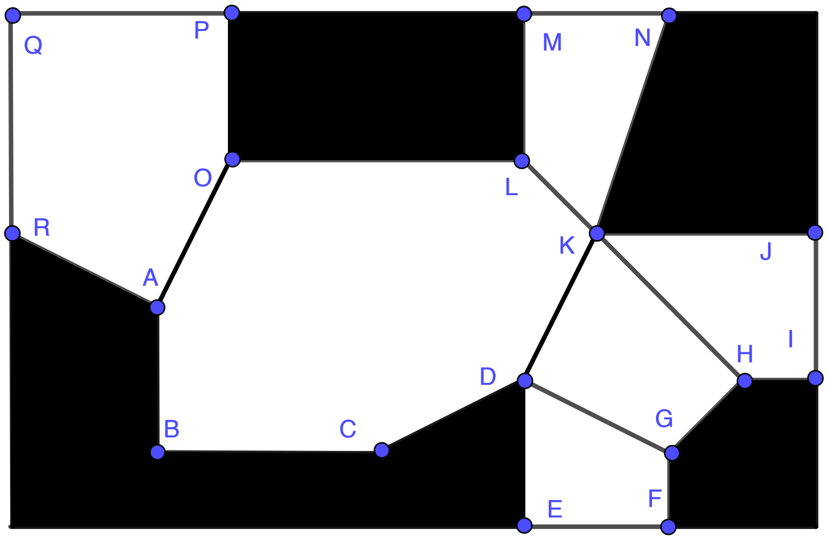
- Visibility graph [1]
- Sparse visibility graph [2]
- Pathfinding:
  - Full insertion
  - Heuristic search



Shaded yellow areas correspond to the area visible from s and t

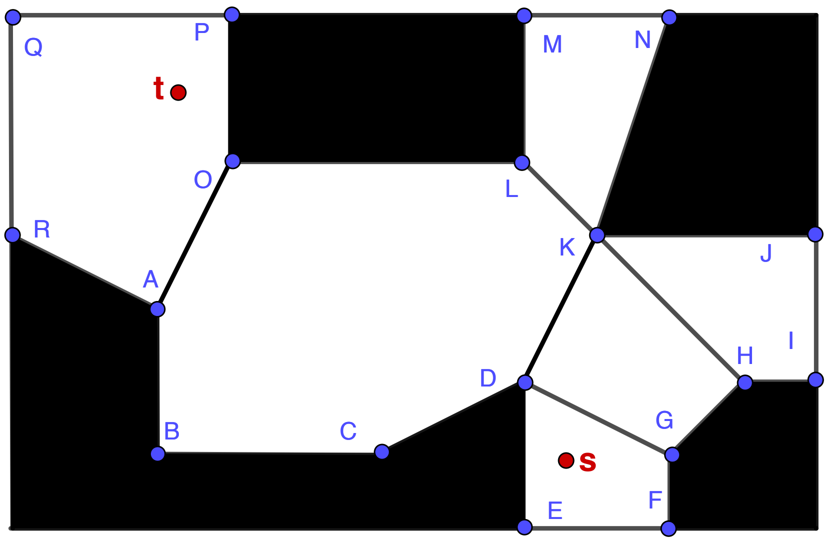
MONASH University

- Navigation Mesh

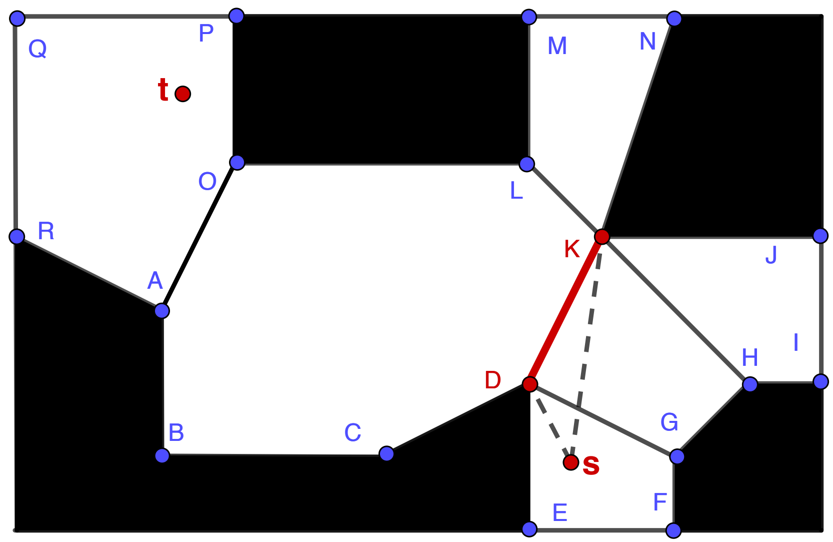


Mesh-based Planner
Polyanya [3]

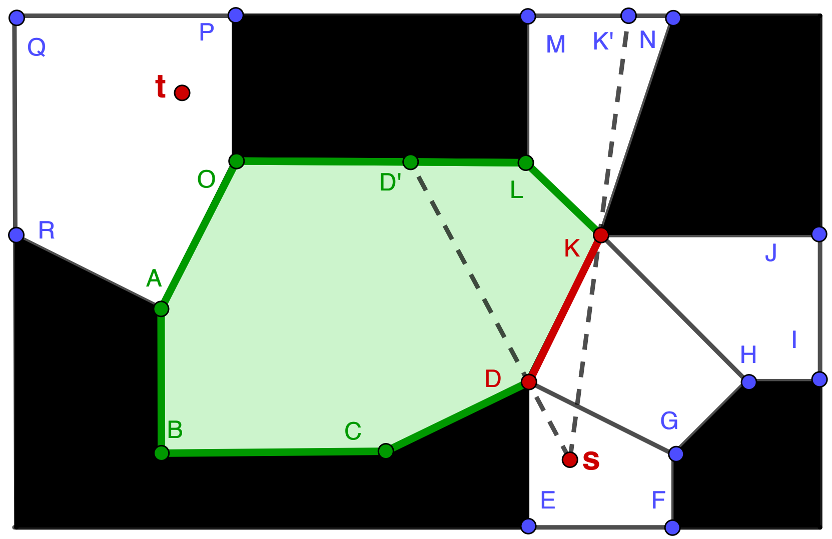
- Navigation Mesh
- Polyanya [3]
  - Search Nodes
  - Successors
  - Evaluation Function



- Navigation Mesh
- Polyanya [3]
  - Search Nodes:
    - Interval & Root
      - i.e. ([D,K] , s)

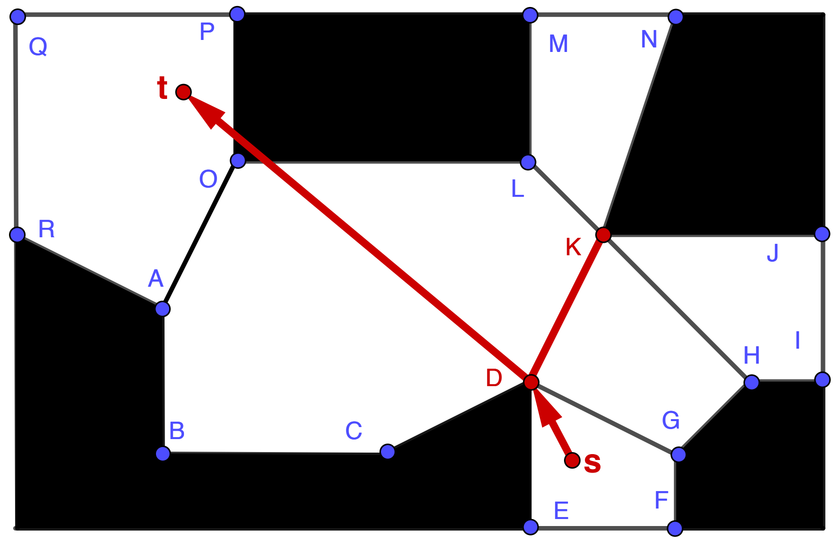


- Navigation Mesh
- Polyanya [3]
  - Search Nodes:
    - Interval & Root
  - Successors:
    - Observable Successors
      - i.e. ([L,K] , s)
    - Non-observable Successors
      - i.e. ([A,O] , D)



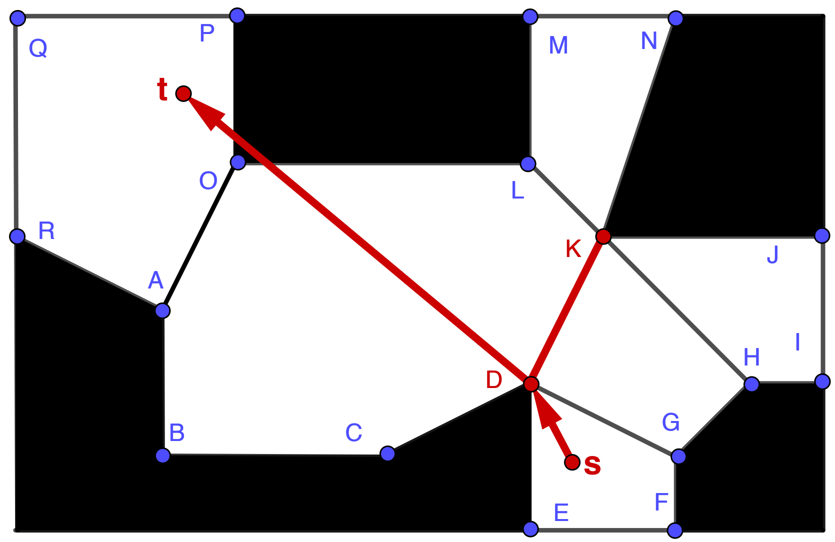
Node expansion in Polyanya. When the current node ([D,K],s) is expanded, it generates the observable successors ([D',L],s), and ([L,K],s); and non-observable successors ([D',O],D), ([O,A],D), ([A,B],D) , ([B,C],D), and ([C,D],D).

- Navigation Mesh
- Polyanya [3]
  - Search Nodes:
    - Interval & Root
  - Successors:
    - Observable Successors
    - Non-observable Successors
  - Evaluation Function:
    - $f(n) = g(n) + h(n)$
      - i.e. $h(n) = d(s,D) + d(D,t)$



Mesh-based Planner
Polyanya [3]

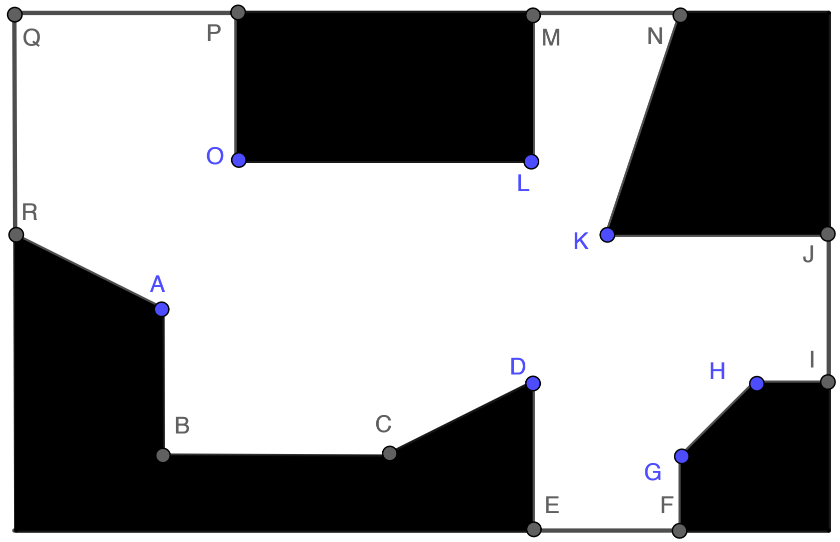
- Navigation Mesh
- Polyanya [3]
  - Search Nodes:
    - Interval & Root
  - Successors:
    - Observable Successors
    - Non-observable Successors
  - Evaluation Function:
    - f(n) = g(n) + h(n)
  - Pathfinding:
    - Depends on the number of polygons expanded and the number of edges on each polygon



Compressed Path Databases (CPD) [4]

- Euclidean-based CPD:
  - First Move Table:
    - Convex vertices

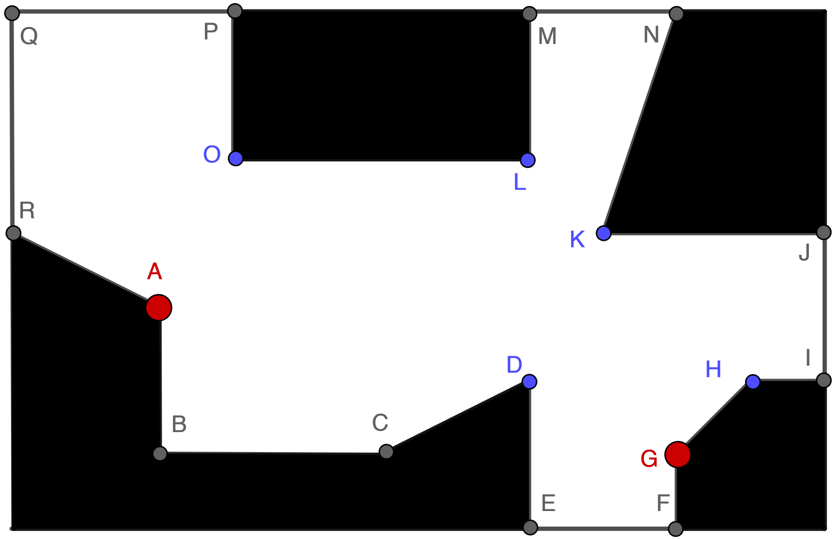
Ordering	A	D	G	H	K	L	O
A	A	D	D	H	K	L	O
D	A	D	G	H	K	L	O
G	D	D	G	H	K	L	O



Our Approach
Compressed Path Databases (CPD) [4]

- Euclidean-based CPD:
  - First Move Table:
    - Convex vertices

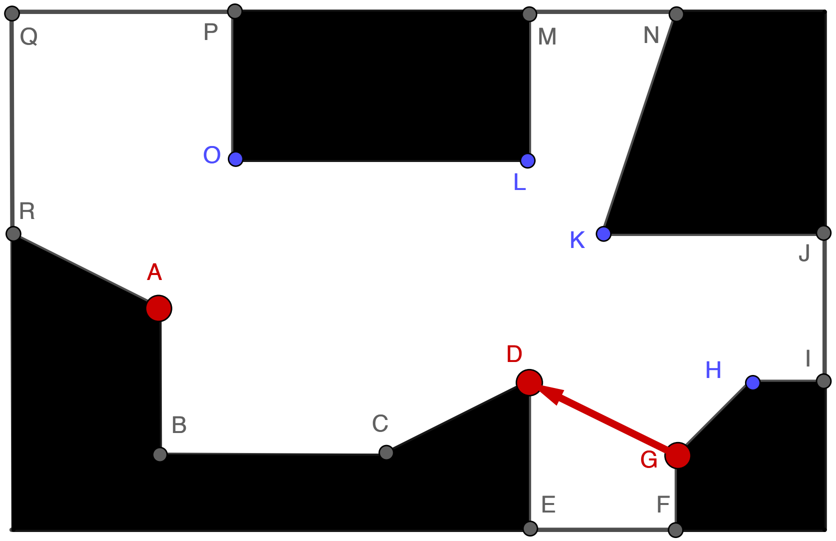
Ordering	A	D	G	H	K	L	O
A	A	D	D	H	K	L	O
D	A	D	G	H	K	L	O
G	D	D	G	H	K	L	O



Our Approach
Compressed Path Databases (CPD) [4]

- Euclidean-based CPD:
  - First Move Table:
    - Convex vertices

Ordering	A	D	G	H	K	L	O
A	A	D	D	H	K	L	O
D	A	D	G	H	K	L	O
G	D	D	G	H	K	L	O

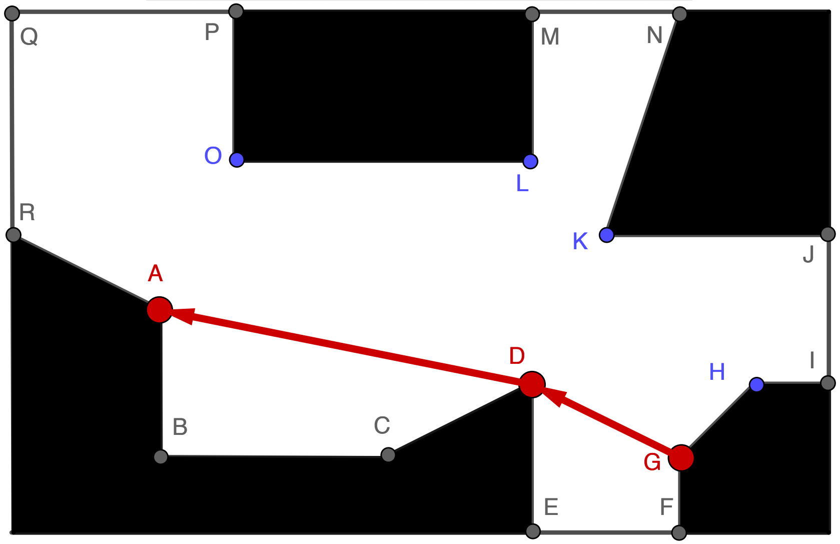


Our Approach
Compressed Path Databases (CPD) [4]

MONASH University

- Euclidean-based CPD:
  - First Move Table:
    - Convex vertices

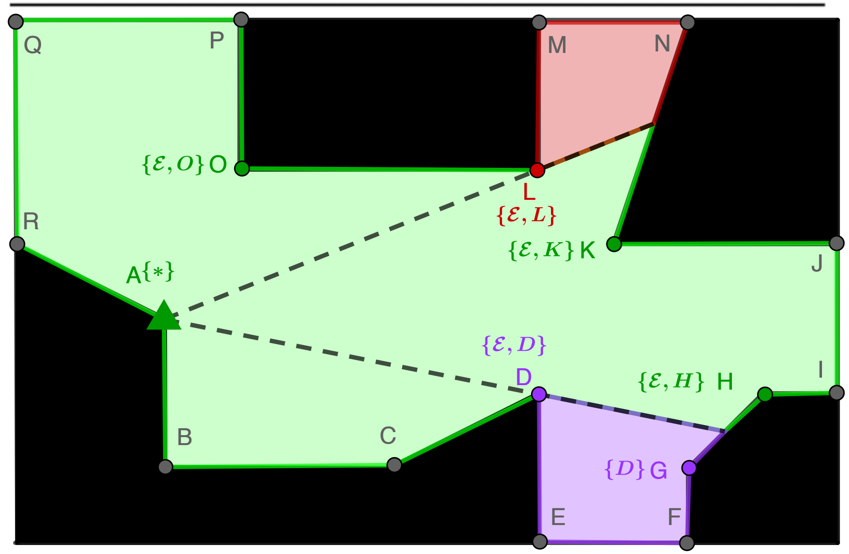
Ordering	A	D	G	H	K	L	O
A	A	D	D	H	K	L	O
D	A	D	G	H	K	L	O
G	D	D	G	H	K	L	O



Compressed Path Databases (CPD) [4]

- Euclidean-based CPD:
  - First Move Table:
    - Convex vertices
  - Symbols:
    - [A – R]: indicates the optimal first move

Ordering	A	D	G	H	K	L	O
A	*	{$\mathcal{E}$,D}	$\boxed{\text{D}}$	{$\mathcal{E}$,H}	{$\mathcal{E}$,K}	{$\mathcal{E}$,L}	{$\mathcal{E}$,O}
D	{$\mathcal{E}$,A}	*	{$\mathcal{E}$,G}	{$\mathcal{E}$,H}	{$\mathcal{E}$,K}	{$\mathcal{E}$,L}	{$\mathcal{E}$,O}
G	D	{$\mathcal{E}$,D}	*	{$\mathcal{E}$,H}	{$\mathcal{E}$,K}	{$\mathcal{E}$,L}	{$\mathcal{E}$,O}

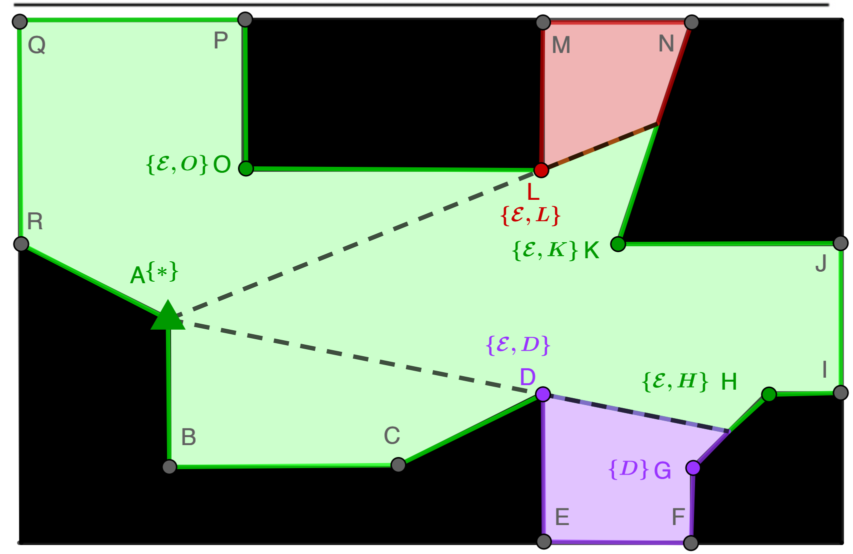


Green area corresponds to the area visible from the source node A. The first move on the optimal path from A to any node in the purple (resp. red) area is D (resp. L).

Our Approach
Compressed Path Databases (CPD) [4]

- Euclidean-based CPD:
  - First Move Table:
    - Convex vertices
  - Symbols:
    - [A – R]: indicates the optimal first move
    - Ɛ: indicates two vertices are directly visible

Ordering	A	D	G	H	K	L	O
A	*	$\boxed{\mathcal{E}}$,D}	D	{$\mathcal{E}$,H}	{$\mathcal{E}$,K}	{$\mathcal{E}$,L}	{$\mathcal{E}$,O}
D	{$\mathcal{E}$,A}	*	{$\mathcal{E}$,G}	{$\mathcal{E}$,H}	{$\mathcal{E}$,K}	{$\mathcal{E}$,L}	{$\mathcal{E}$,O}
G	D	{$\mathcal{E}$,D}	*	{$\mathcal{E}$,H}	{$\mathcal{E}$,K}	{$\mathcal{E}$,L}	{$\mathcal{E}$,O}



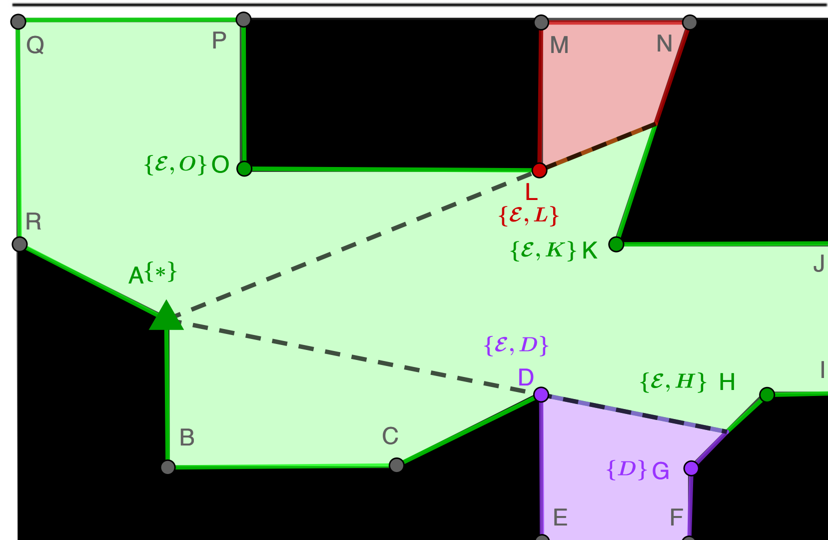
Green area corresponds to the area visible from the source node A. The first move on the optimal path from A to any node in the purple (resp. red) area is D (resp. L).

Our Approach
Compressed Path Databases (CPD) [4]

MONASH University

- Euclidean-based CPD:
  - First Move Table:
    - Convex vertices
  - Symbols:
    - [A – R]: indicates the optimal first move
    - $\mathcal{E}$: indicates two vertices are directly visible
    - {$\mathcal{E}$,D}: redundant symbol [5]

Ordering	A	D	G	H	K	L	O
A	*	$\boxed{\{\mathcal{E},D\}}$	D	$\{\mathcal{E},H\}$	$\{\mathcal{E},K\}$	$\{\mathcal{E},L\}$	$\{\mathcal{E},O\}$
D	$\{\mathcal{E},A\}$	*	$\{\mathcal{E},G\}$	$\{\mathcal{E},H\}$	$\{\mathcal{E},K\}$	$\{\mathcal{E},L\}$	$\{\mathcal{E},O\}$
G	D	$\{\mathcal{E},D\}$	*	$\{\mathcal{E},H\}$	$\{\mathcal{E},K\}$	$\{\mathcal{E},L\}$	$\{\mathcal{E},O\}$

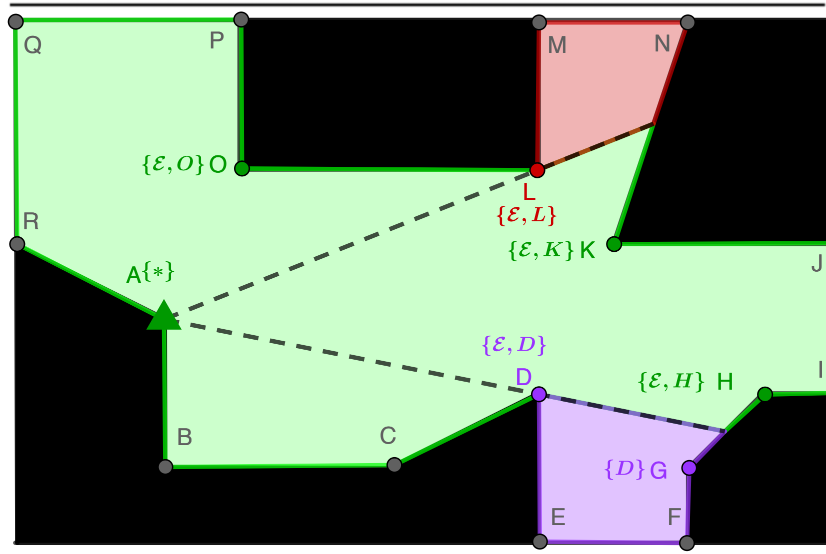


Green area corresponds to the area visible from the source node A. The first move on the optimal path from A to any node in the purple (resp. red) area is D (resp. L).

Our Approach
Compressed Path Databases (CPD) [4]

- Euclidean-based CPD:
  - First Move Table:
    - Convex vertices
  - Symbols:
    - [A – R]: indicates the optimal first move
    - Ɛ: indicate two vertices are directly visible
    - {Ɛ,D}: redundant symbol [5]
    - *: wildcard symbol [5]

Ordering	A	D	G	H	K	L	O
A	*	$\{\mathcal{E}$,D$\}$	D	$\{\mathcal{E}$,H$\}$	$\{\mathcal{E}$,K$\}$	$\{\mathcal{E}$,L$\}$	$\{\mathcal{E}$,O$\}$
D	$\{\mathcal{E}$,A$\}$	*	$\{\mathcal{E}$,G$\}$	$\{\mathcal{E}$,H$\}$	$\{\mathcal{E}$,K$\}$	$\{\mathcal{E}$,L$\}$	$\{\mathcal{E}$,O$\}$
G	D	$\{\mathcal{E}$,D$\}$	*	$\{\mathcal{E}$,H$\}$	$\{\mathcal{E}$,K$\}$	$\{\mathcal{E}$,L$\}$	$\{\mathcal{E}$,O$\}$

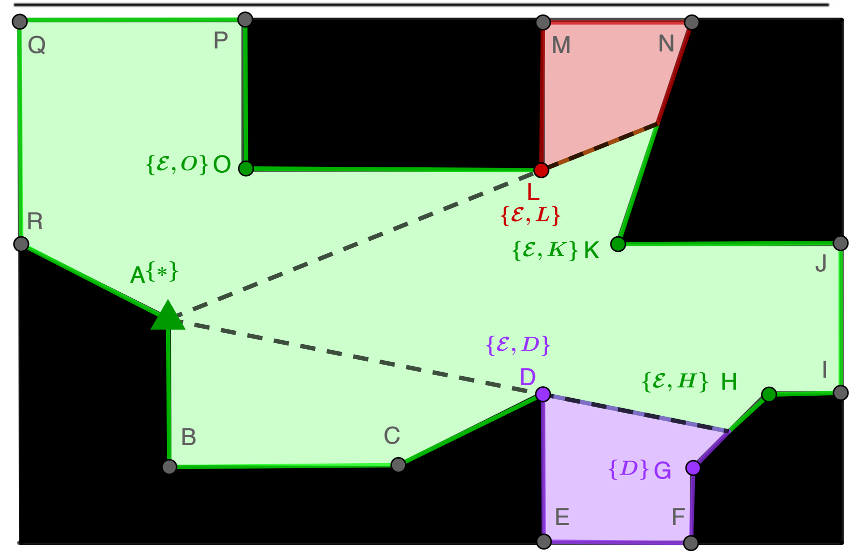


Green area corresponds to the area visible from the source node A. The first move on the optimal path from A to any node in the purple (resp. red) area is D (resp. L).

Our Approach
Compressed Path Databases (CPD) [4]

- Euclidean-based CPD:
  - First Move Table:
    - Convex vertices
  - Symbols:
    - [A – R]: indicates the optimal first move
    - Ɛ: indicate two vertices are directly visible
    - {Ɛ,D}: redundant symbol [5]
    - *: wildcard symbol [5]
  - Compression:
    - Depth first search ordering [4]

Ordering	A	D	G	H	K	L	O
A	*	$\{\mathcal{E}$,D$\}$	D	$\{\mathcal{E}$,H$\}$	$\{\mathcal{E}$,K$\}$	$\{\mathcal{E}$,L$\}$	$\{\mathcal{E}$,O$\}$
D	$\{\mathcal{E}$,A$\}$	*	$\{\mathcal{E}$,G$\}$	$\{\mathcal{E}$,H$\}$	$\{\mathcal{E}$,K$\}$	$\{\mathcal{E}$,L$\}$	$\{\mathcal{E}$,O$\}$
G	D	$\{\mathcal{E}$,D$\}$	*	$\{\mathcal{E}$,H$\}$	$\{\mathcal{E}$,K$\}$	$\{\mathcal{E}$,L$\}$	$\{\mathcal{E}$,O$\}$

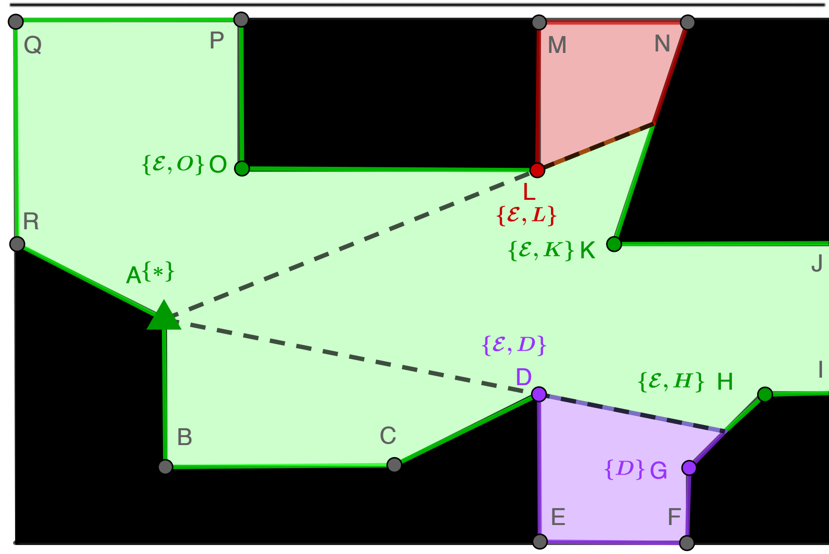


Green area corresponds to the area visible from the source node A. The first move on the optimal path from A to any node in the purple (resp. red) area is D (resp. L).

Our Approach
Compressed Path Databases (CPD) [4]

- Euclidean-based CPD:
  - First Move Table:
    - Convex vertices
  - Symbols:
    - [A – R]: indicates the optimal first move
    - Ɛ: indicate two vertices are directly visible
    - {Ɛ,D}: redundant symbol [5]
    - *: wildcard symbol [5]
  - Compression:
    - Depth first search ordering [4]
    - Run length encoding [4]
      - (i.e. Row A: 1D; 4Ɛ)

Ordering	A	D	G	H	K	L	O
A	*	{Ɛ,D}	D	{Ɛ,H}	{Ɛ,K}	{Ɛ,L}	{Ɛ,O}
D	{Ɛ,A}	*	{Ɛ,G}	{Ɛ,H}	{Ɛ,K}	{Ɛ,L}	{Ɛ,O}
G	D	{Ɛ,D}	*	{Ɛ,H}	{Ɛ,K}	{Ɛ,L}	{Ɛ,O}

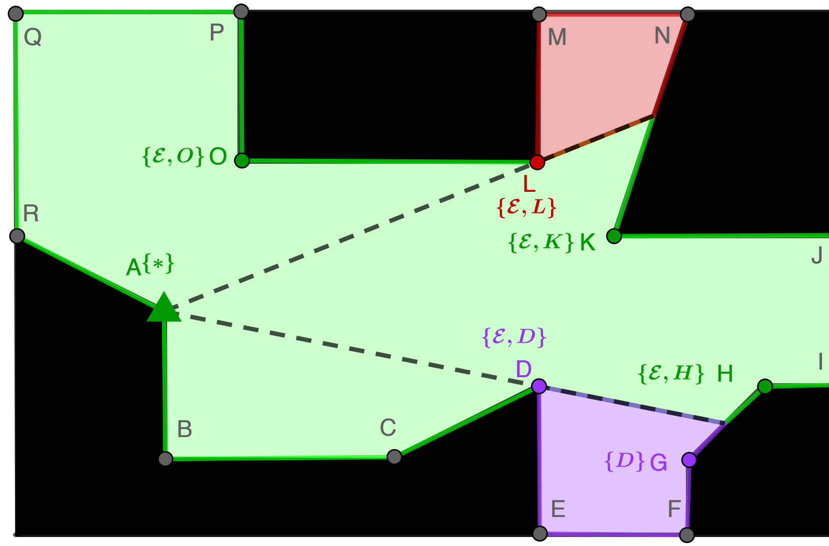


Green area corresponds to the area visible from the source node A. The first move on the optimal path from A to any node in the purple (resp. red) area is D (resp. L).

Our Approach
Compressed Path Databases (CPD) [4]

- Euclidean-based CPD:
  - First Move Table:
    - Convex vertices
  - Symbols:
    - [A – R]: indicates the optimal first move
    - Ɛ: indicate two vertices are directly visible
    - {Ɛ,D}: redundant symbol [5]
    - *: wildcard Symbol [5]
  - Compression:
    - Depth first search ordering [4]
    - Run length encoding [4]
  - First Move Extraction:
    - A simple binary search

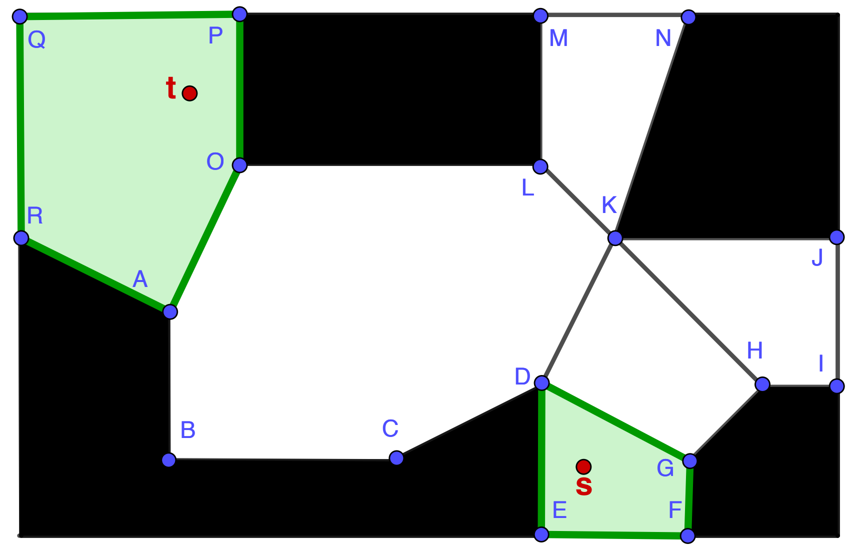
Ordering	A	D	G	H	K	L	O
A	*	{$\mathcal{E}$,D}	D	{$\mathcal{E}$,H}	{$\mathcal{E}$,K}	{$\mathcal{E}$,L}	{$\mathcal{E}$,O}
D	{$\mathcal{E}$,A}	*	{$\mathcal{E}$,G}	{$\mathcal{E}$,H}	{$\mathcal{E}$,K}	{$\mathcal{E}$,L}	{$\mathcal{E}$,O}
G	D	{$\mathcal{E}$,D}	*	{$\mathcal{E}$,H}	{$\mathcal{E}$,K}	{$\mathcal{E}$,L}	{$\mathcal{E}$,O}



Green area corresponds to the area visible from the source node A. The first move on the optimal path from A to any node in the purple (resp. red) area is D (resp. L).

Our Approach
End Point Search (EPS)

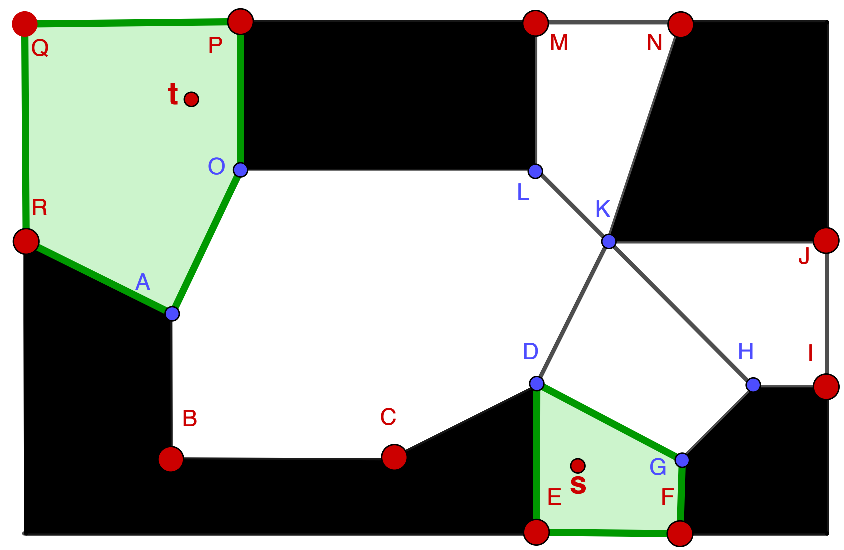
- Euclidean-based CPD
- End Point Search (EPS)
  - Incremental Insertion
    - $Search_s$ & $Search_t$
    - V_s & V_t
    - Shortest Path (sp)



Shaded green area shows the space incrementally explored by $search_s$ and $search_t$

Our Approach
End Point Search (EPS)

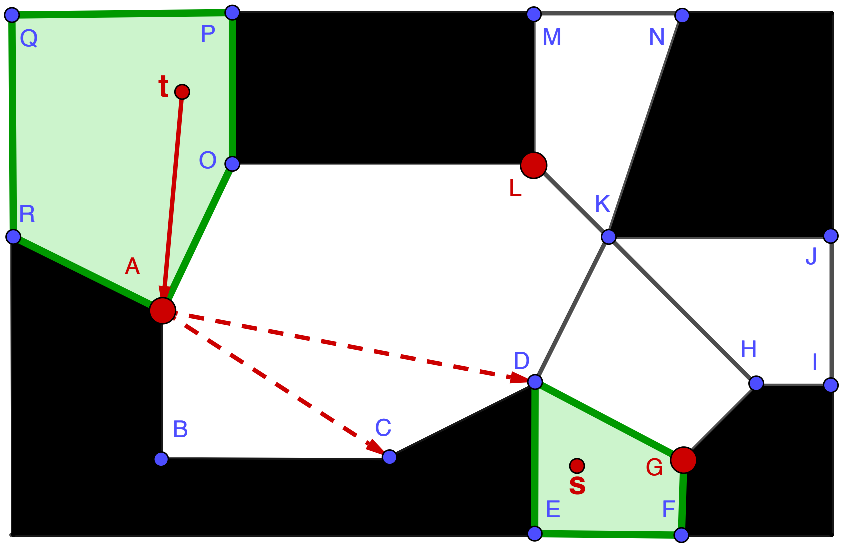
- Euclidean-based CPD
- End Point Search (EPS)
  - Incremental Insertion
    - $Search_s$ & $Search_t$
    - V_s & V_t
    - Shortest Path (sp)
  - Vertex Pruning
    - Dead-end Vertices



Avoid accessing the non-convex visible vertices

Our Approach
End Point Search (EPS)

- Euclidean-based CPD
- End Point Search (EPS)
  - Incremental Insertion
    - $Search_s$ & $Search_t$
    - V_s & V_t
    - Shortest Path (sp)
  - Vertex Pruning
    - Dead-end Vertices
    - Non-turn Vertices

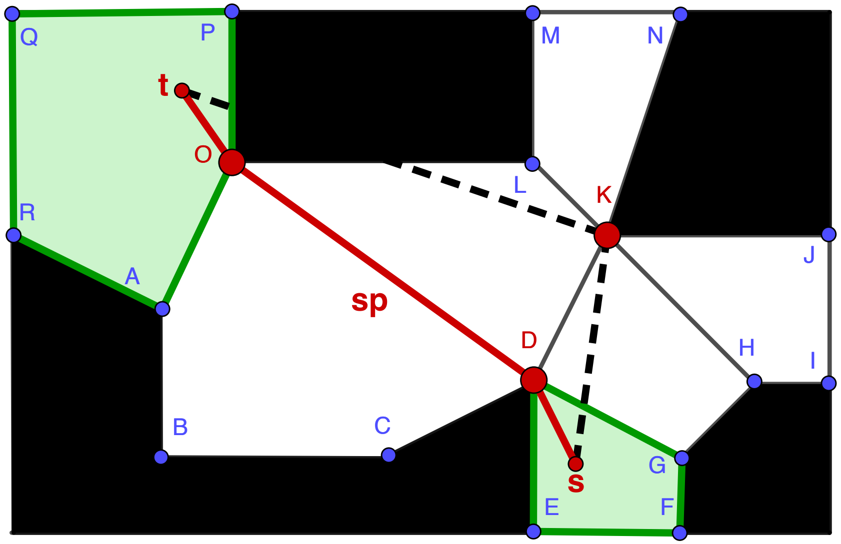


Avoid accessing the visible vertices can not possibly turn

End Point Search (EPS)

- Euclidean-based CPD
- End Point Search (EPS)
  - Incremental Insertion
    - $Search_s$ & $Search_t$
    - V_s & V_t
    - Shortest Path (sp)
  - Vertex Pruning
    - Dead-end Vertices
    - Non-turn Vertices
    - Distance Pruning

      d(s, K) + h (K, t) > |sp|

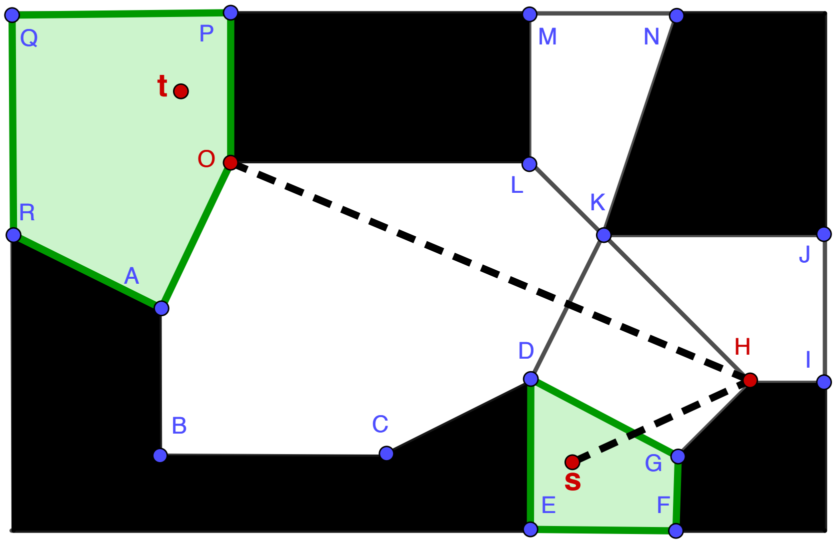


Avoid accessing the visible vertices that can be pruned based on the current solution

MONASH University

- Euclidean-based CPD
- End Point Search (EPS)
  - Incremental Insertion
    - $Search_s$ & $Search_t$
    - V_s & V_t
    - Shortest Path (sp)
  - Vertex Pruning
    - Dead-end Vertices
    - Non-turn Vertices
    - Distance-based Pruning
  - Path Pruning
    - Non-taut Paths

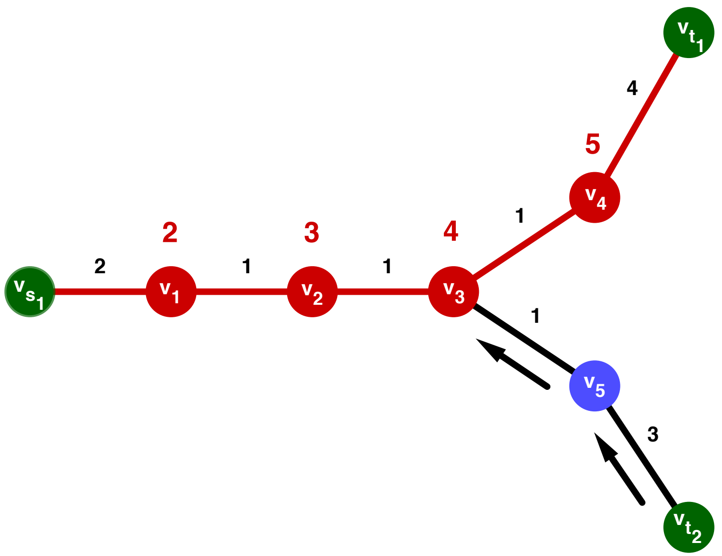
    i.e. (H,O): <s, H, O>



Stop path extraction when first move gives non-taut path

Our Approach
End Point Search (EPS)

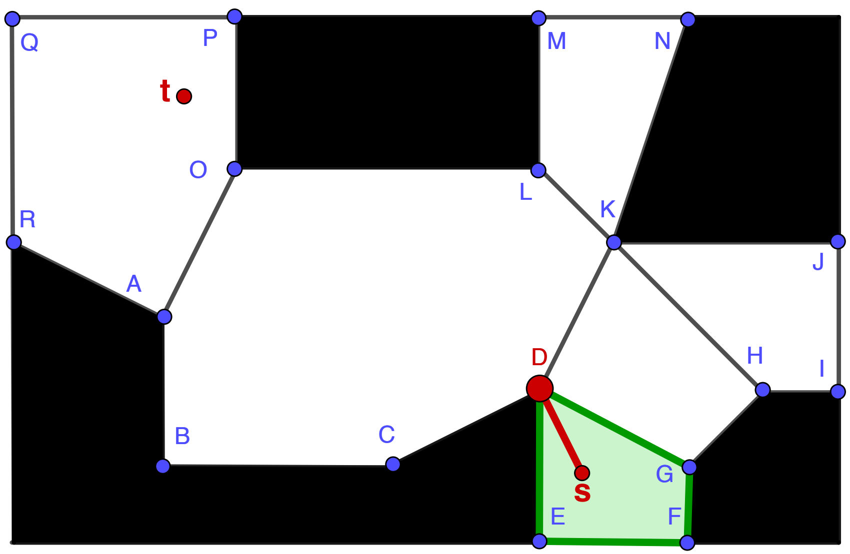
- Euclidean-based CPD
- End Point Search (EPS)
  - Incremental Insertion
    - $Search_s$ & $Search_t$
    - V_s & V_t
    - Shortest Path (sp)
  - Vertex Pruning
    - Dead-end Vertices
    - Non-turn Vertices
    - Distance-based Pruning
  - Path Pruning
    - Non-taut Paths
    - Cost Caching



EPS cache distance on each node when extract paths from V_{t1} to V_{s1}. The path extraction from V_{t2} to V_{s1} terminate when reach a cached node (i.e. V_3)

Our Approach
End Point Search (EPS)

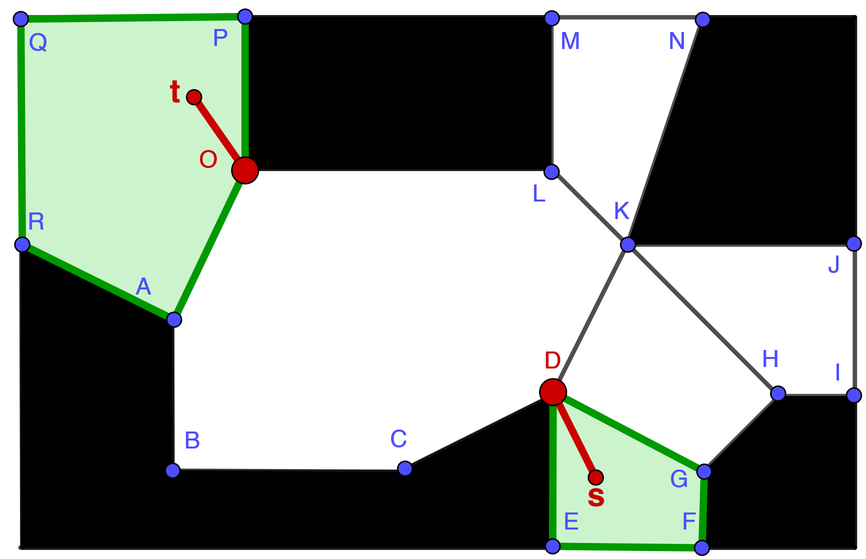
- Euclidean-based CPD
- End Point Search (EPS)
  - Incremental Insertion
    - $Search_s$ & $Search_t$
    - V_s & V_t
    - Shortest Path (sp)
  - Vertex Pruning
    - Dead-end Vertices
    - Non-turn Vertices
    - Distance-based Pruning
  - Path Pruning
    - Non-taut Paths
    - Cost Caching
  - Running Example



The shaded green area shows the space incrementally explored by $search_s$ and $search_t$

MONASH University

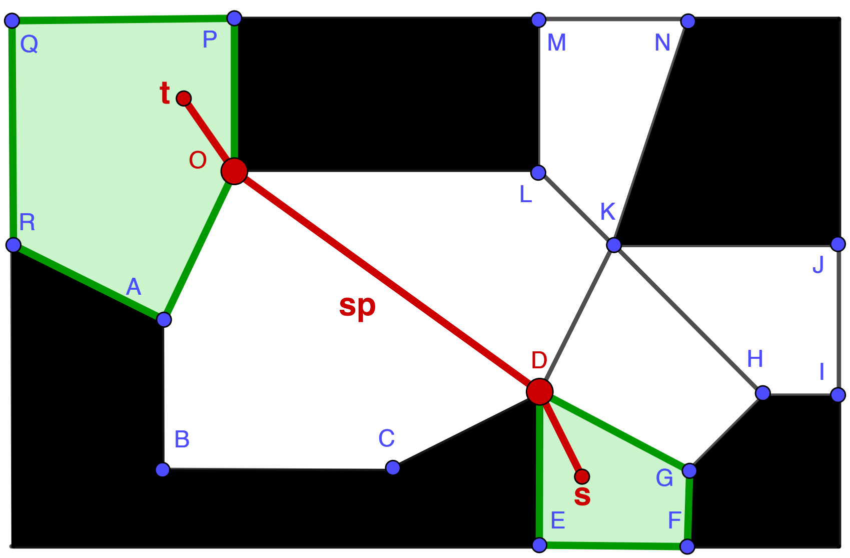
- Euclidean-based CPD
- End Point Search (EPS)
  - Incremental Insertion
    - Search_s & Search_t
    - V_s & V_t
    - Shortest Path (sp)
  - Vertex Pruning
    - Dead-end Vertices
    - Non-turn Vertices
    - Distance-based Pruning
  - Path Pruning
    - Non-taut Paths
    - Cost Caching
  - Running Example



The shaded green areas show the space incrementally explored by search_s and search_t

Our Approach
End Point Search (EPS)

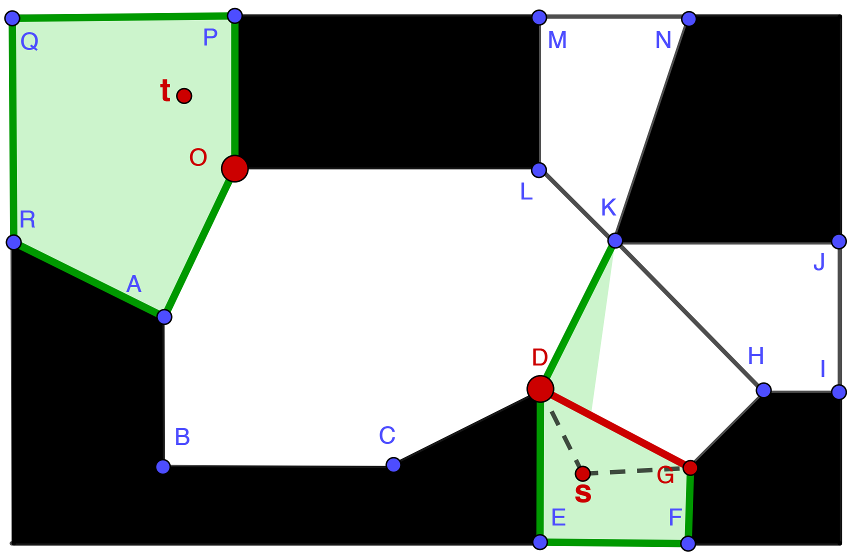
- Euclidean-based CPD
- End Point Search (EPS)
  - Incremental Insertion
    - $Search_s$ & $Search_t$
    - V_s & V_t
    - Shortest Path (sp)
  - Vertex Pruning
    - Dead-end Vertices
    - Non-turn Vertices
    - Distance-based Pruning
  - Path Pruning
    - Non-taut Paths
    - Cost Caching
  - Running Example



The shaded green areas show the space incrementally explored by $search_s$ and $search_t$

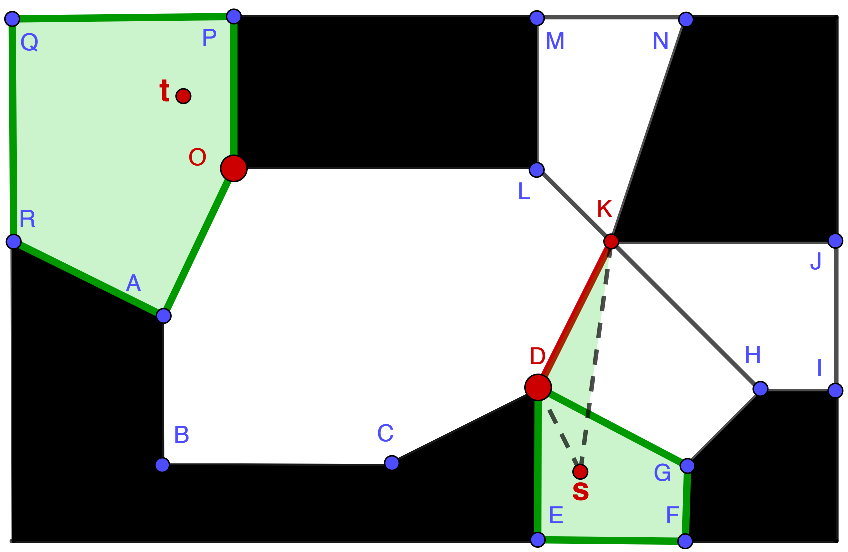
MONASH University

- Euclidean-based CPD
- End Point Search (EPS)
  - Incremental Insertion
    - $Search_s$ & $Search_t$
    - V_s & V_t
    - Shortest Path (sp)
  - Vertex Pruning
    - Dead-end Vertices
    - Non-turn Vertices
    - Distance-based Pruning
  - Path Pruning
    - Non-taut Paths
    - Cost Caching
  - Running Example



The shaded green areas show the space incrementally explored by $search_s$ and $search_t$

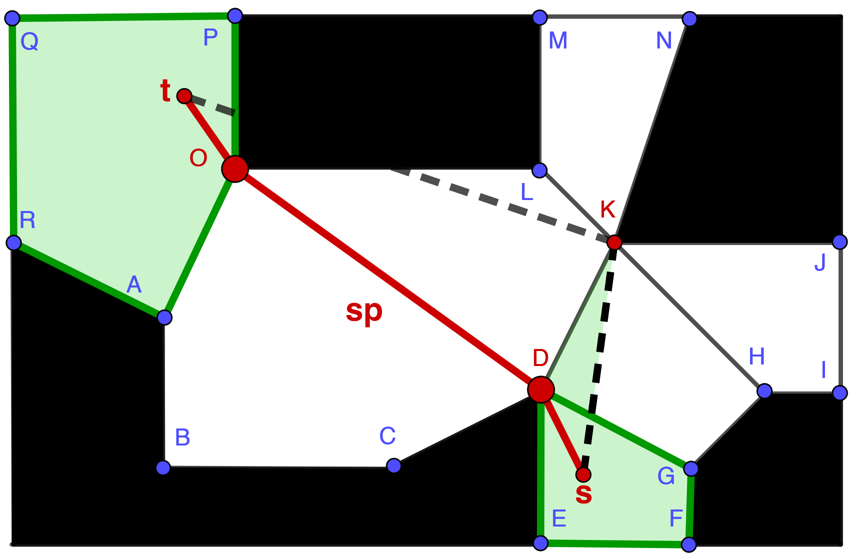
- Euclidean-based CPD
- End Point Search (EPS)
  - Incremental Insertion
    - $Search_s$ & $Search_t$
    - V_s & V_t
    - Shortest Path (sp)
  - Vertex Pruning
    - Dead-end Vertices
    - Non-turn Vertices
    - Distance-based Pruning
  - Path Pruning
    - Non-taut Paths
    - Cost Caching
  - Running Example



The shaded green areas show the space incrementally explored by $search_s$ and $search_t$

Our Approach
End Point Search (EPS)

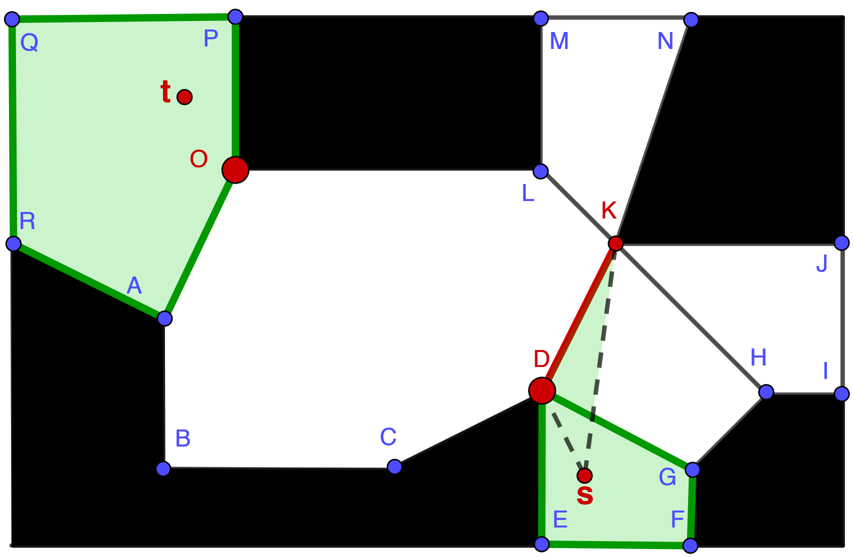
- Euclidean-based CPD
- End Point Search (EPS)
  - Incremental Insertion
    - $Search_s$ & $Search_t$
    - V_s & V_t
    - Shortest Path (sp)
  - Vertex Pruning
    - Dead-end Vertices
    - Non-turn Vertices
    - Distance-based Pruning
  - Path Pruning
    - Non-taut Paths
    - Cost Caching
  - Running Example



The shaded green areas show the space incrementally explored by $search_s$ and $search_t$

Our Approach
End Point Search (EPS)

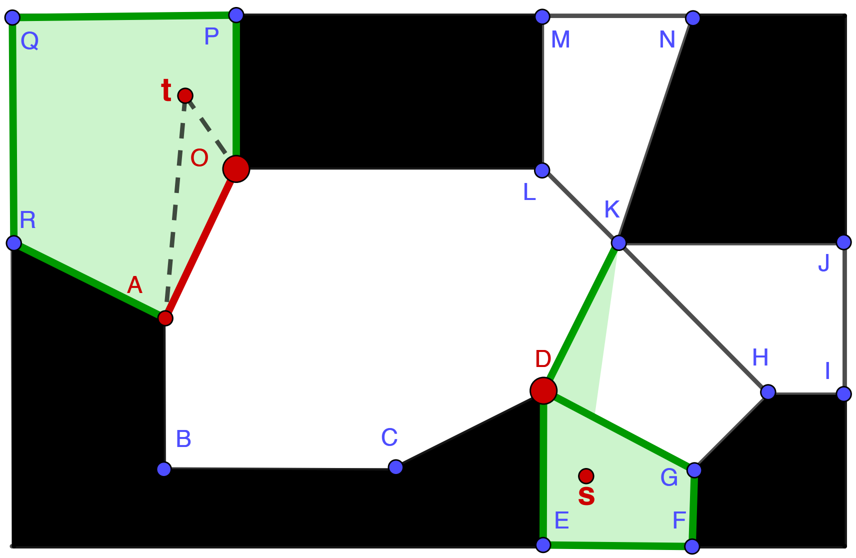
- Euclidean-based CPD
- End Point Search (EPS)
  - Incremental Insertion
    - $Search_s$ & $Search_t$
    - V_s & V_t
    - Shortest Path (sp)
  - Vertex Pruning
    - Dead-end Vertices
    - Non-turn Vertices
    - Distance-based Pruning
  - Path Pruning
    - Non-taut Paths
    - Cost Caching
  - Running Example



The shaded green areas show the space incrementally explored by $search_s$ and $search_t$

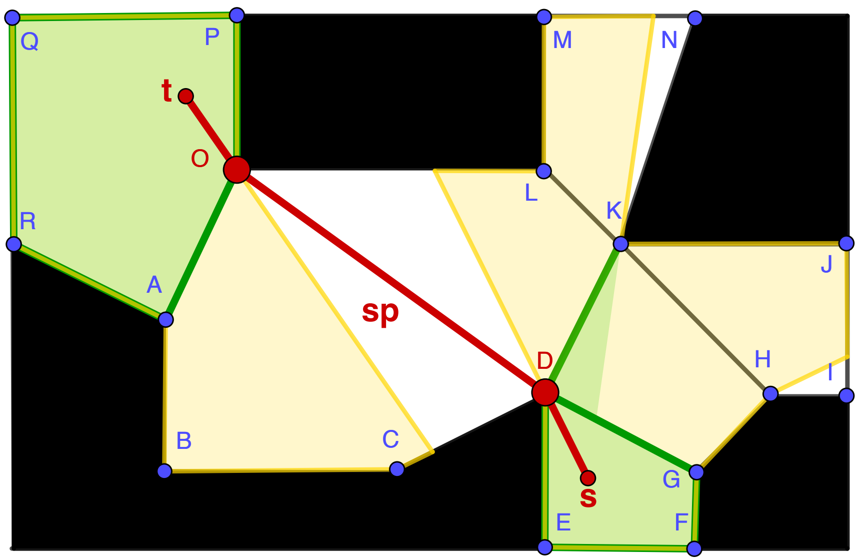
Our Approach
End Point Search (EPS)

- Euclidean-based CPD
- End Point Search (EPS)
  - Incremental Insertion
    - $Search_s$ & $Search_t$
    - V_s & V_t
    - Shortest Path (sp)
  - Vertex Pruning
    - Dead-end Vertices
    - Non-turn Vertices
    - Distance-based Pruning
  - Path Pruning
    - Non-taut Paths
    - Cost Caching
  - Running Example



Shaded green areas show the space incrementally explored by $search_s$ and $search_t$

- Euclidean-based CPD
- End Point Search (EPS)
  - Incremental Insertion
    - $Search_s$ & $Search_t$
    - V_s & V_t
    - Shortest Path (sp)
  - Vertex Pruning
    - Dead-end Vertices
    - Non-turn Vertices
    - Distance-based Pruning
  - Path Pruning
    - Non-taut Paths
    - Cost Caching
  - Running Example

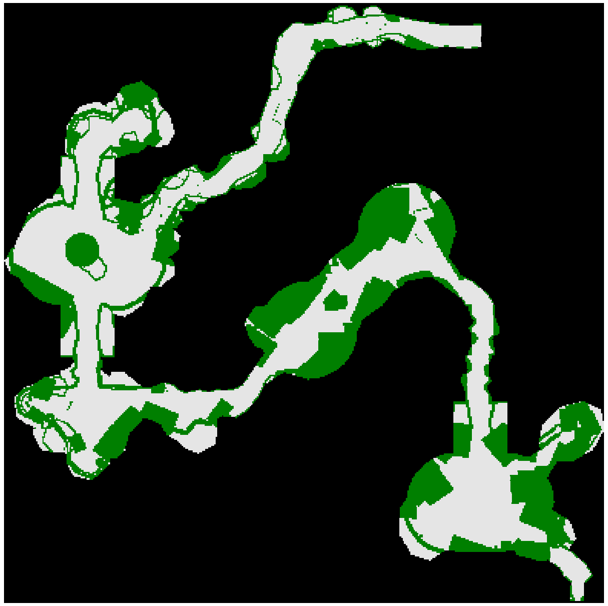


Shaded yellow areas correspond to the area visible from s and t. Shaded green areas show the space incrementally explored by Polyanya when $search_s$ and $search_t$ are both exhausted

Our Approach

Experimental Results

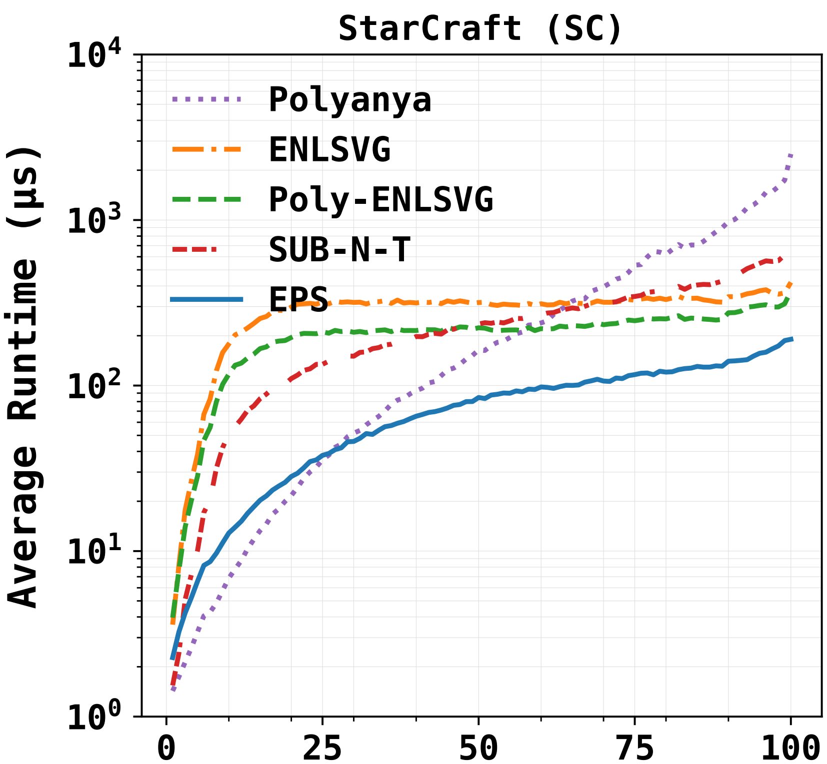
- Euclidean-based CPD
- End Point Search (EPS)
- Experimental Results
  - Benchmarks:
    - Game Maps from Sturtevant's Repository



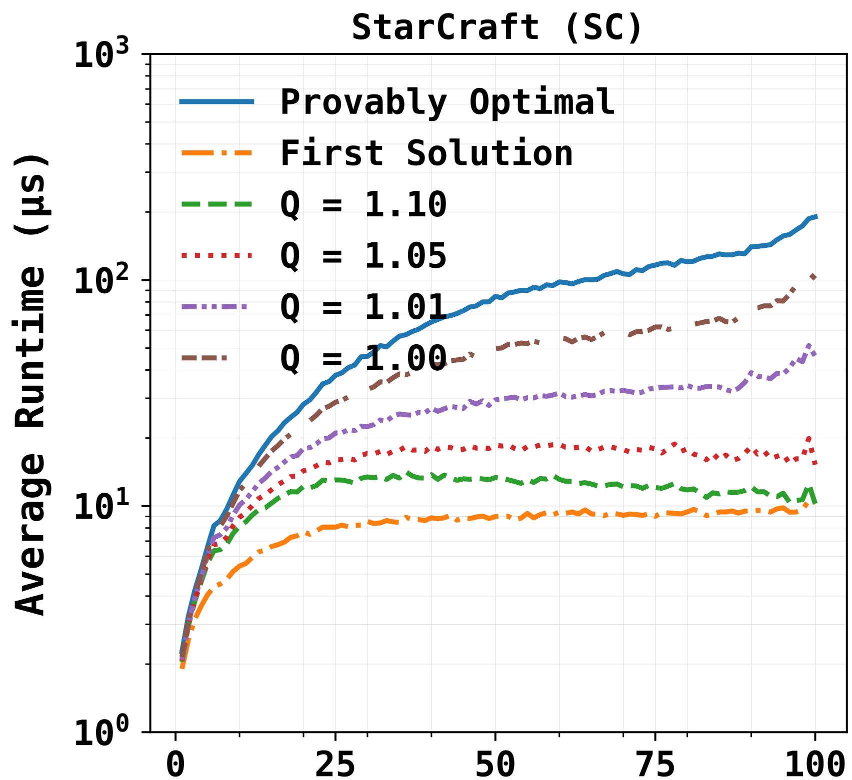
Euclidean-based CPD Statistics								
	#Map	#Queries	Build Time		Raw Memory		CPD Memory	
			Avg	Max	Avg	Max	Avg	Max
SC	75	198k	0.7	8	190	2202	2	14

Our Approach
Experimental Results

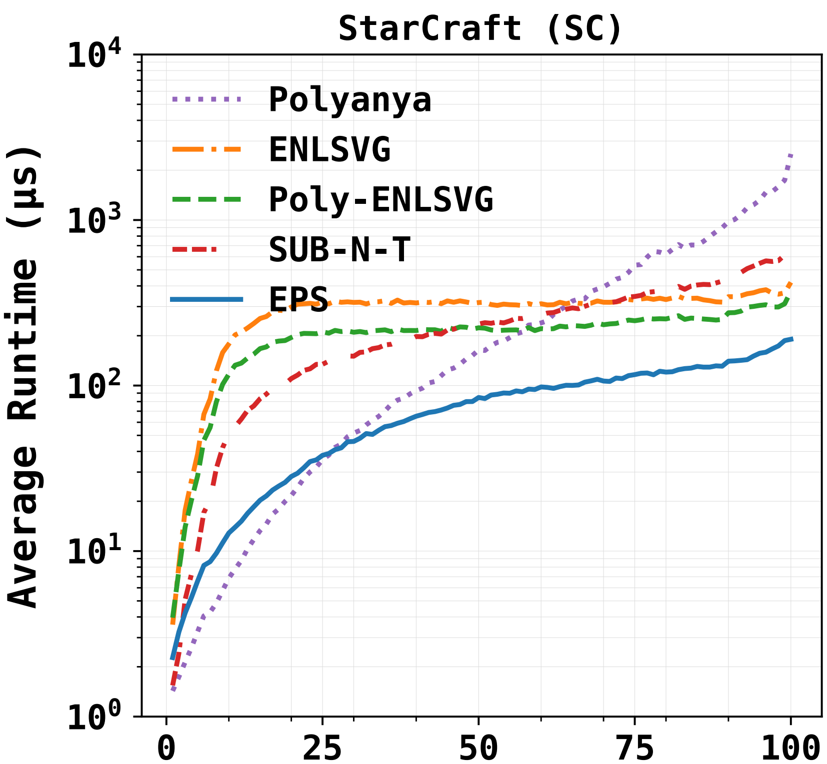
- Optimal Search



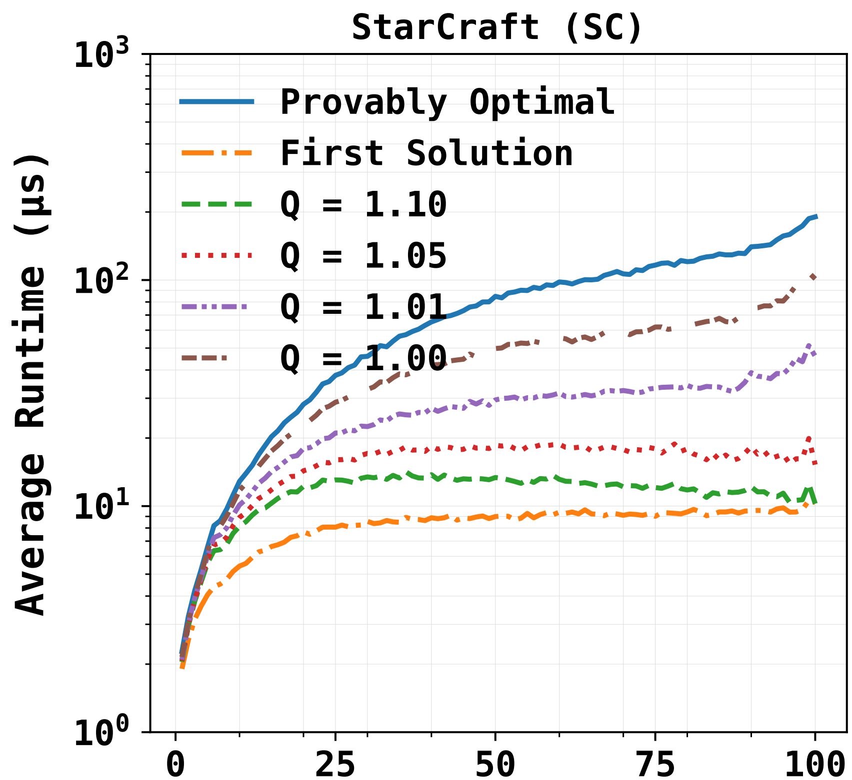
- Any-time Search



- Optimal Search



- Any-time Search



Reference

[1] T. Lozano-Perez, M. A. Wesley, An algorithm for planning collision-free paths among polyhedral obstacles, Commun. ACM 22 (10) (1979) 560-570.

[2] S. Oh, H. W. Leong, Edge n-level sparse visibility graphs: Fast optimal any-angle pathfinding using hierarchical taut paths, in: Proceedings of the Tenth International Symposium on Combinatorial Search, SOCS 2017, 16-17 June 2017, Pittsburgh, Pennsylvania, USA, AAAI Press, 2017, pp. 64-72.

[3] M. Cui, D. D. Harabor, A. Grastien, Compromise-free pathfinding on a navigation mesh, in: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017, ijcai.org, 2017, pp. 496–502.

[4] B. Strasser, D. Harabor, A. Botea, Fast first-move queries through run-length encoding, in: Proceedings of the Seventh Annual Symposium on Combinatorial Search, SOCS 2014, Prague, Czech Republic, 15-17 August 2014, AAAI Press, 2014.

[5] M. Chiari, S. Zhao, A. Botea, A. E. Gerevini, D. Harabor, A. Saetti, M. Salvetti, P. J. Stuckey, Cutting the size of compressed path databases with wildcards and redundant symbols, in: Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling, ICAPS 2018, Berkeley, CA, USA, July 11-15, 2019, AAAI Press,5852019, pp. 106–113.

Thank you for listening