Visibility Solutions for Sensor Networks and UAVs

Karl J. Obermeyer

Department of Mechanical Engineering
Center for Control, Dynamical Systems and Computation
University of California at Santa Barbara

2010:05:21
PhD Defense

Acknowledgments

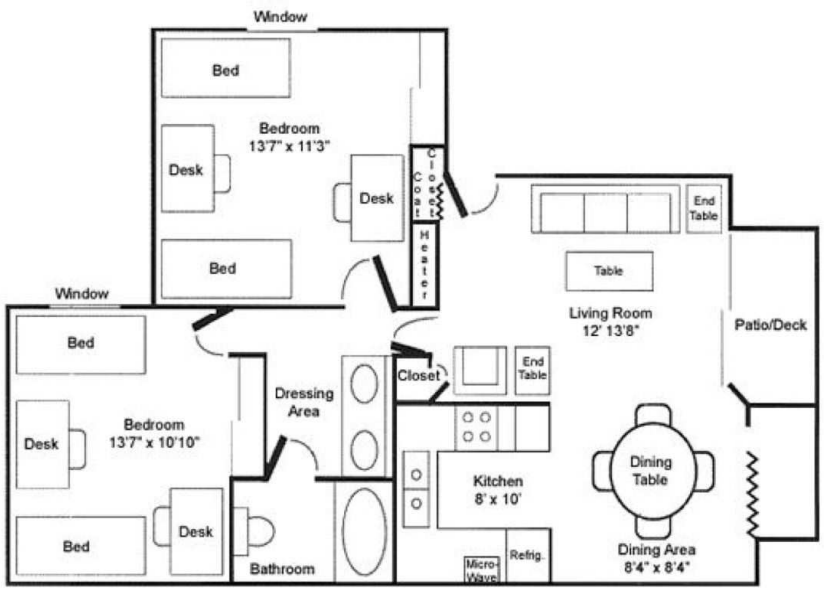
Advisor:	Francesco Bullo
Collaborators:	Anurag Ganguli, Swaroop Darbha, Paul Oberlin
Committee:	Bassam Bamieh, Jeff Moehlis, João Hespanha, Subhash Suri
Support:	DoD SMART Fellowship

Visibility Problem Prototype

How should an autonomous system move such that it achieves or maintains line of sight of some object(s) of interest in a nonconvex environment?

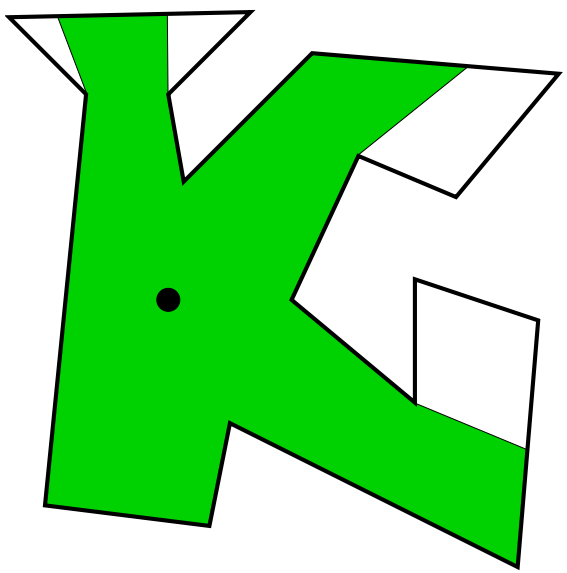
Visibility Problem Prototype

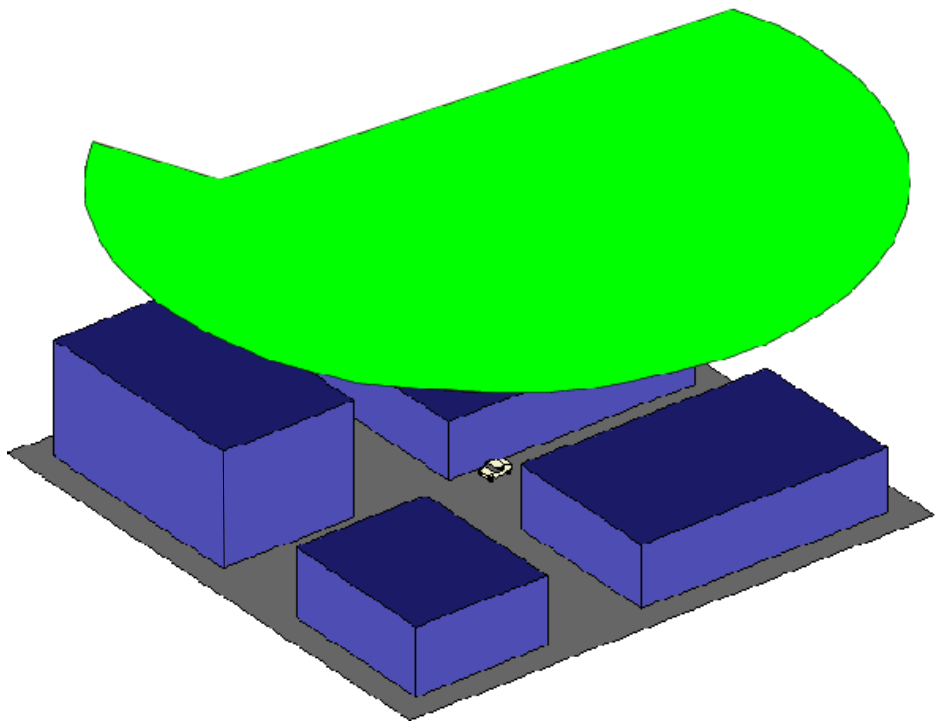
How should an autonomous system move such that it achieves or maintains line of sight of some object(s) of interest in a nonconvex environment?



Visibility Problem Prototype

How should an autonomous system move such that it achieves or maintains line of sight of some object(s) of interest in a nonconvex environment?





Visibility-Based Pursuit-Evasion

Visual Surveillance

Visual Reconnaissance

Exploration

Outline

Visibility-Based Pursuit-Evasion
Part I: Searchlight and Camera Scheduling

Visual Surveillance
Part II: Multi-Agent Deployment

Visual Reconnaissance
Part III: UAV Path Planning

Selected Publications

Journal Articles and Conference Proceedings

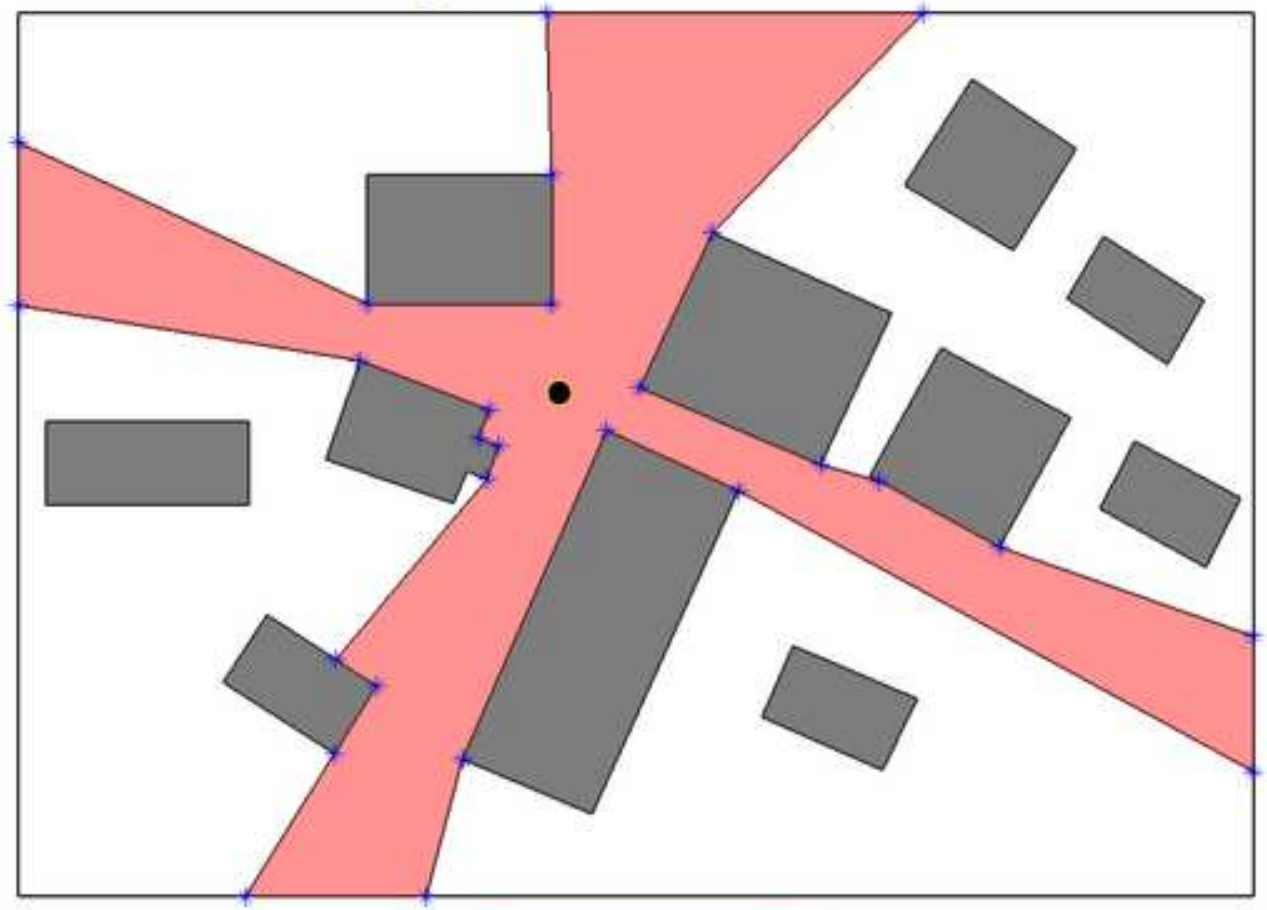
A Complete Algorithm for Searchlight Scheduling – Obermeyer, Ganguli, and Bullo, in *International J. of Comp. Geometry and Applications*, Note: Submitted.

Asynchronous Distributed Searchlight Scheduling – Obermeyer, Ganguli, and Bullo, in *IEEE Conf. on Decision and Control*, 2007.

Multi-Agent Deployment for Visibility Coverage in Polygonal Environments with Holes – Obermeyer, Ganguli, and Bullo, Note: Journal Article in Preparation.

Sampling-Based Roadmap Methods for a Visual Reconnaissance UAV – Obermeyer, Oberlin, and Darbha, *AIAA J. of Guidance, Control, and Dynamics*, Note: Submitted.

VisiLibity.org for Floating-Point Visibility Computations



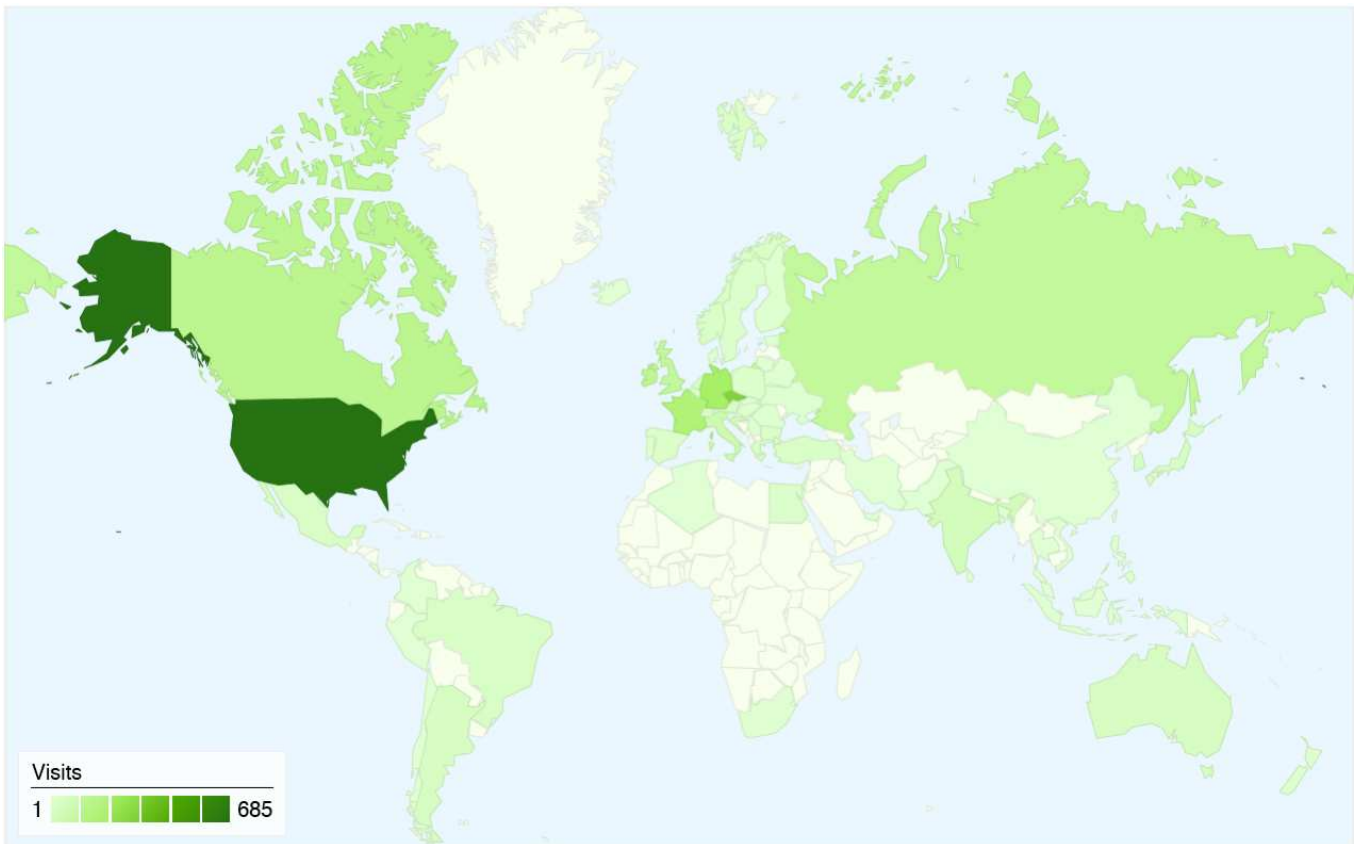
What is VisiLibity?
Download VisiLibity
Using VisiLibity

LGPL Free Software

Math behind VisiLibity
Acknowledgements
License, Links

VisiLibity Stats (May 2008 – May 2010)

- 1500 unique visitors in 66 countries
- 300 downloads



Part I

Searchlight and Camera Scheduling

The Searchlight Scheduling Problem

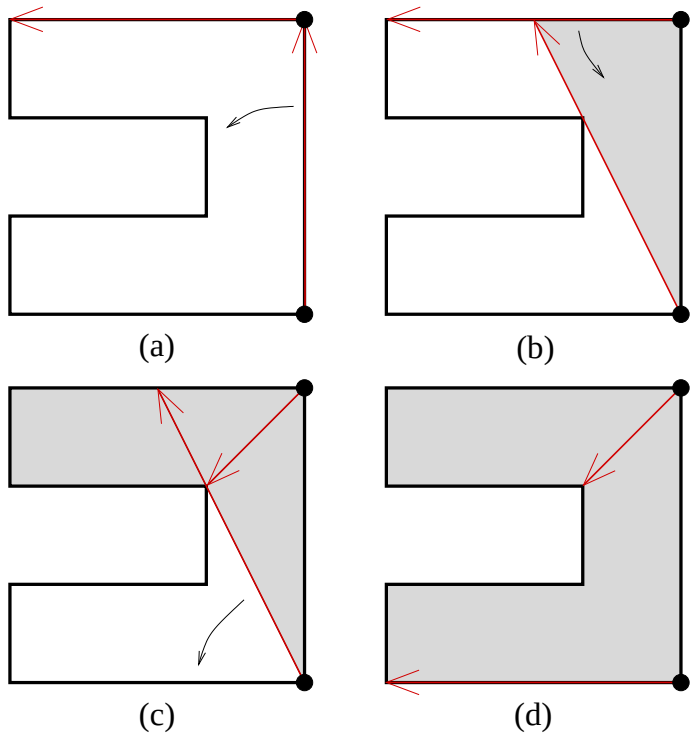
[Sugihara, Suzuki, and Yamashita, 1990]

Goal

Find a **schedule** *to rotate a set of searchlights such that any evader in an environment will necessarily be detected in finite time*

Assumptions

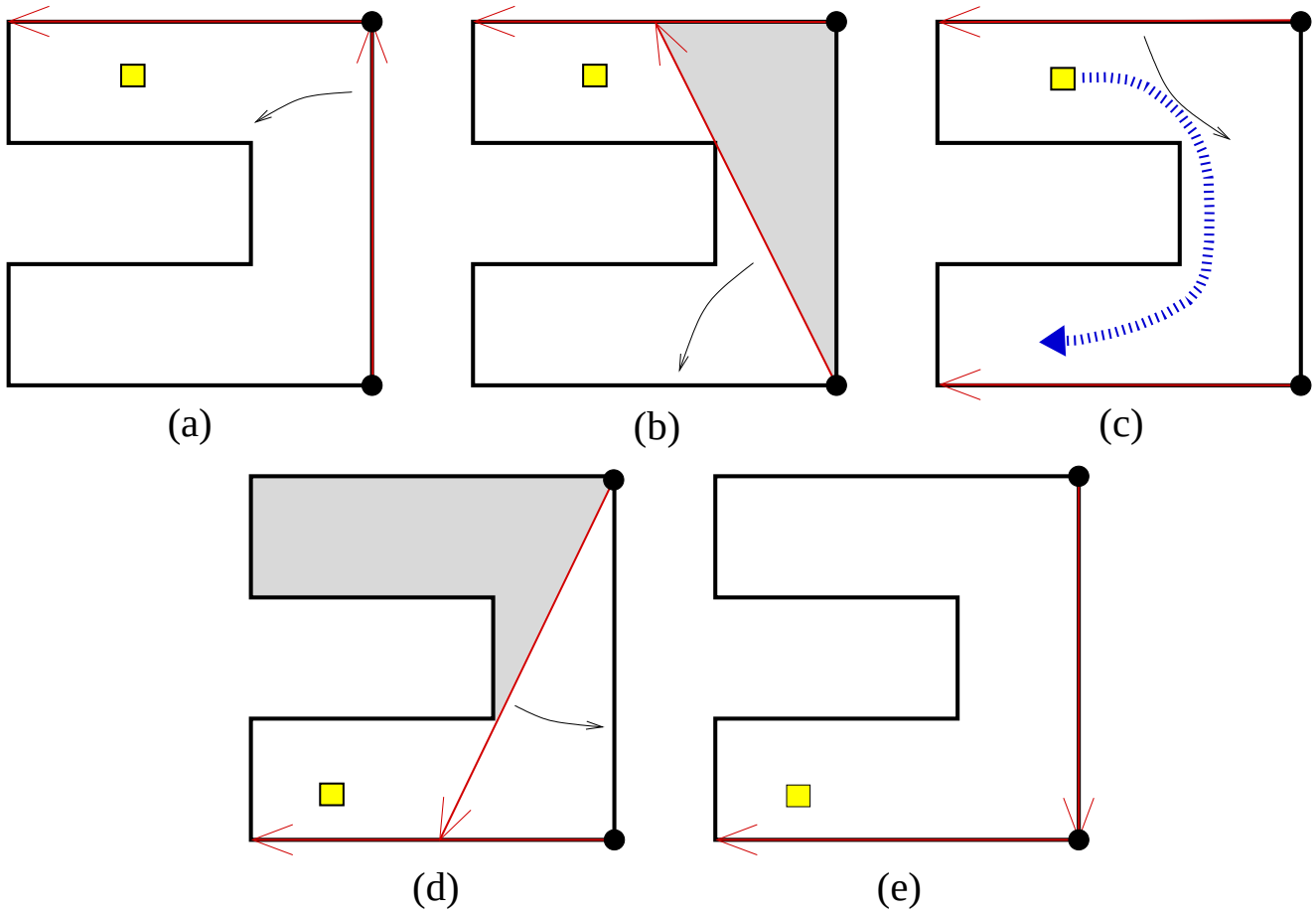
- Polygonal environment
- No colocated searchlights
- Evaders have unbounded speed



The Searchlight Scheduling Problem

[Sugihara, Suzuki, and Yamashita, 1990]

Example of a schedule which fails



The Searchlight Scheduling Problem

[Sugihara, Suzuki, and Yamashita, 1990]

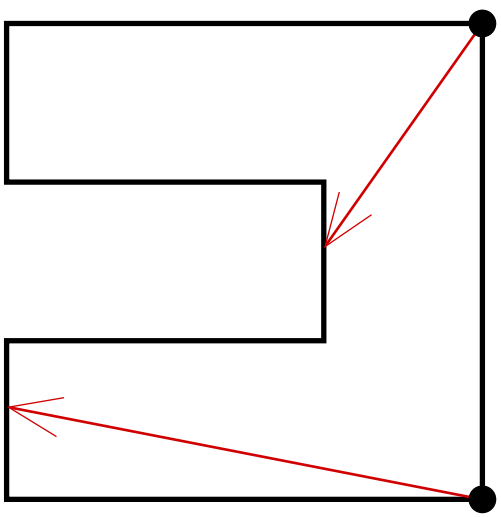
Schedule for a single searchlight

$\theta^{[i]} : [0, T] \longrightarrow \mathbb{T}^1$

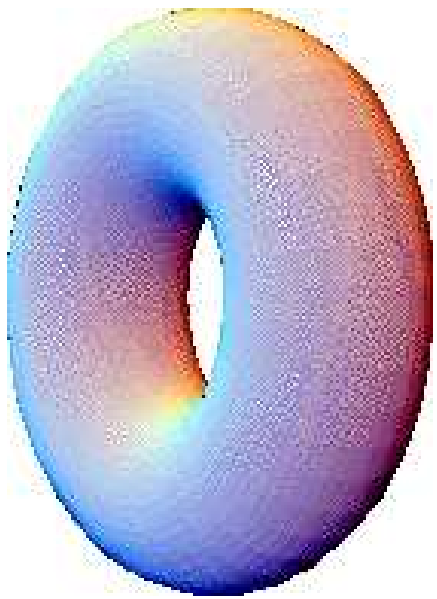
Joint Schedule for N searchlights

$\Theta^{[i]} : [0, T] \longrightarrow \mathbb{T}^N$

problem instance

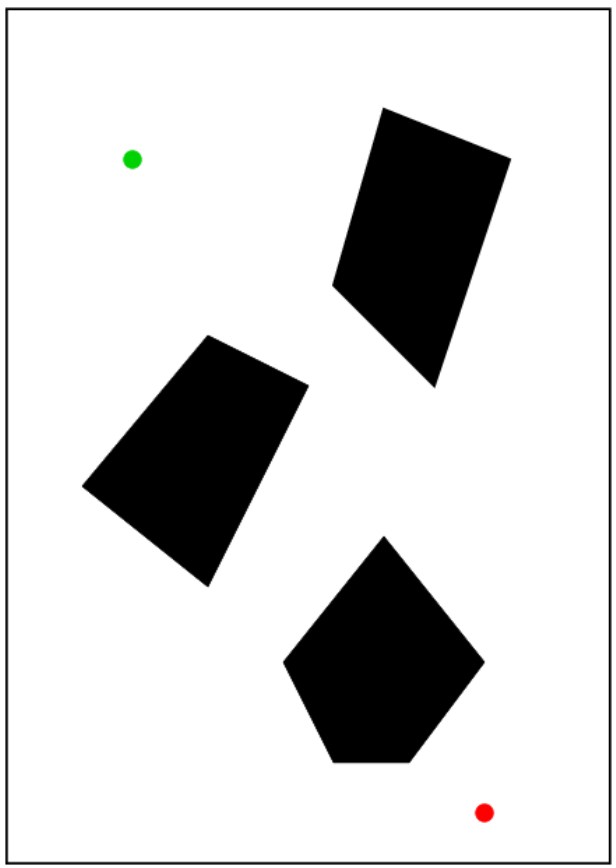


configuration space $\mathbb{T}^N$

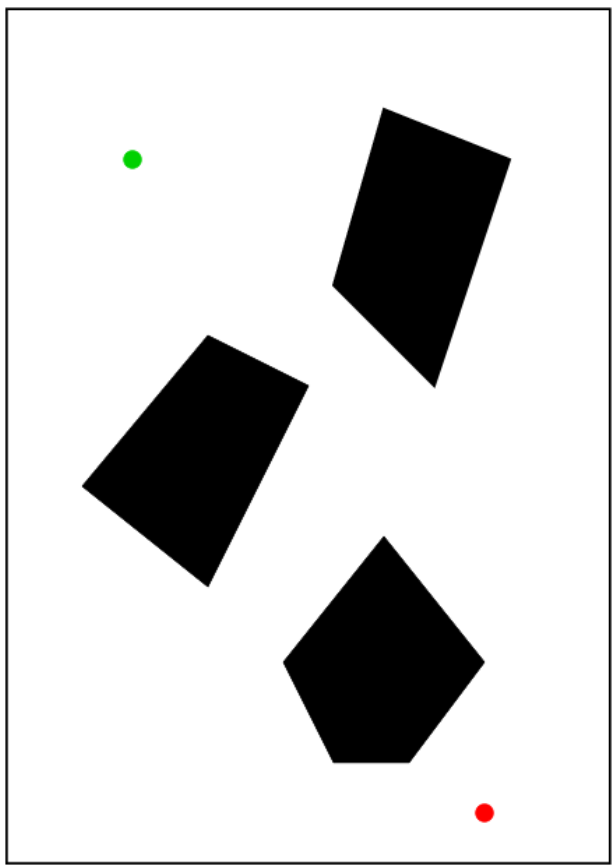


Technical Approach

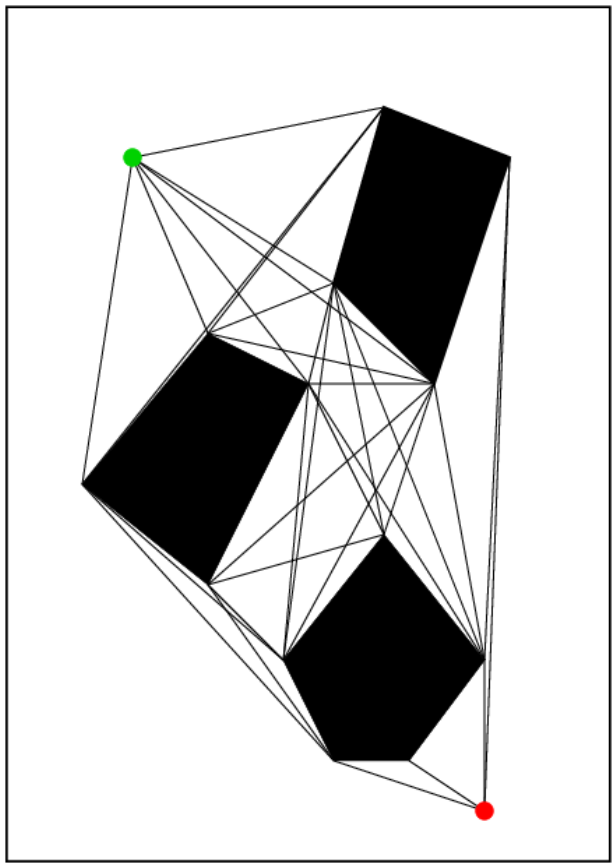
Inspiration from Euclidean Shortest Path Problem



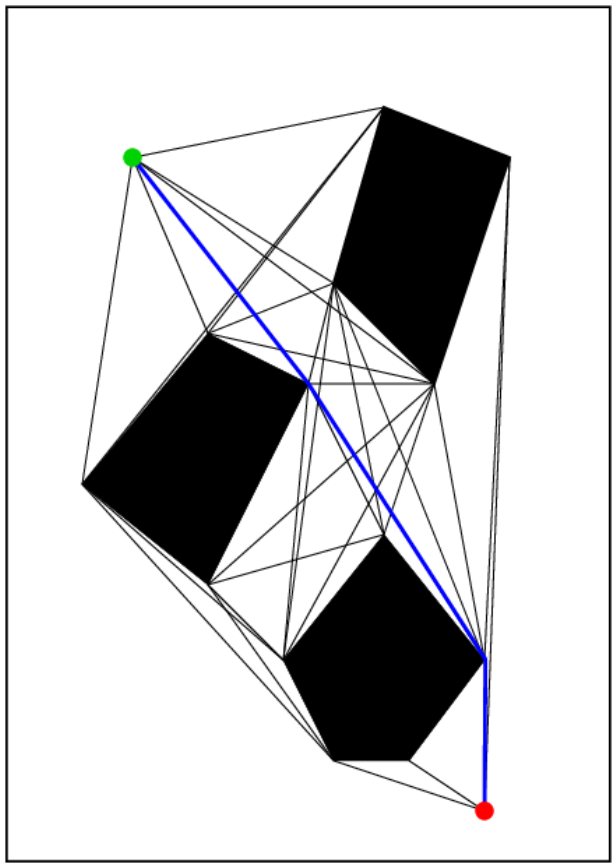
Reduction Theorem [Nilsson, 1969]
Shortest paths hug corners



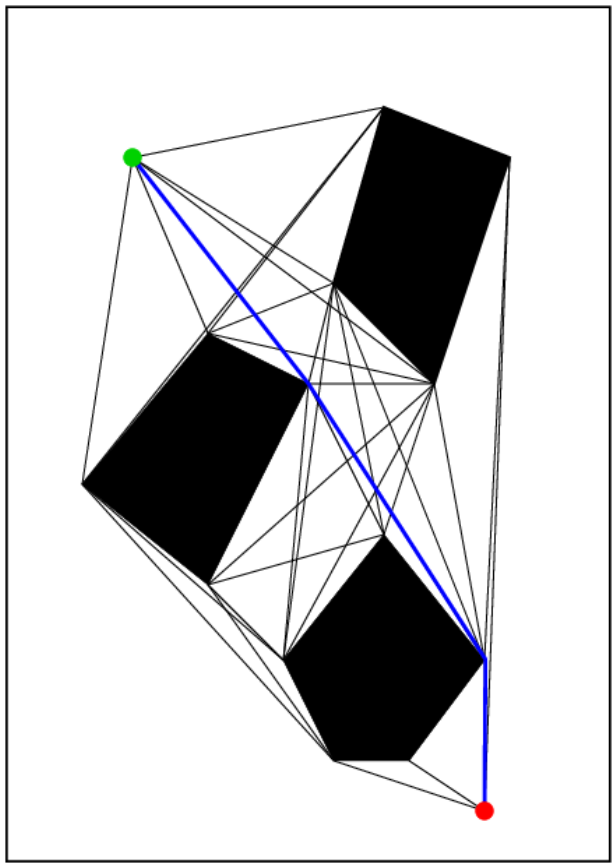
Reduction Theorem [Nilsson, 1969]
Shortest paths hug corners



Reduction Theorem [Nilsson, 1969]
Shortest paths hug corners



Reduction Theorem [Nilsson, 1969]
Shortest paths hug corners

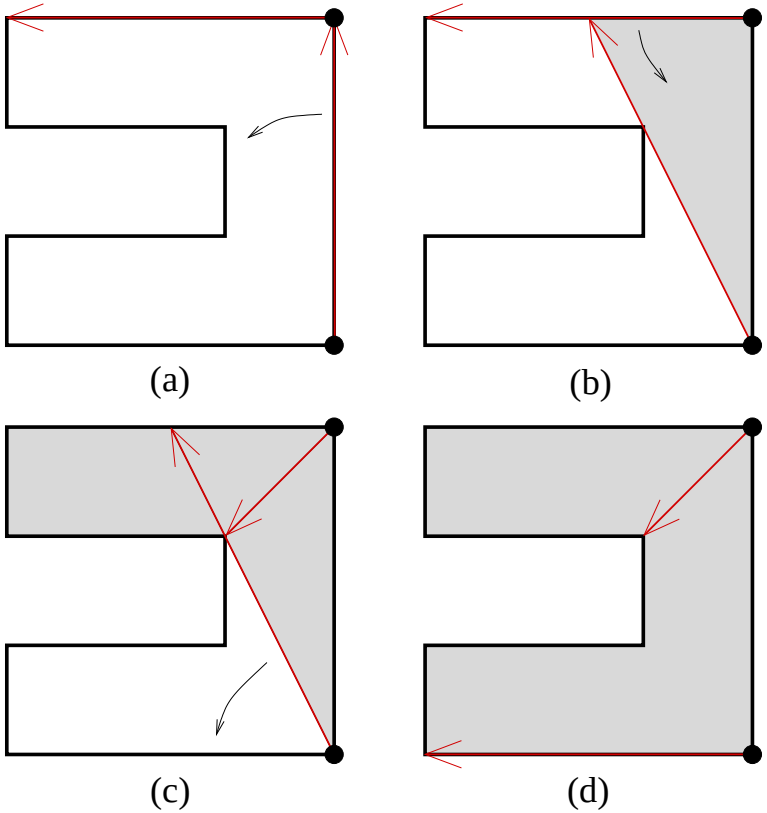


Roadmap Method*

(1) discretize continuous space, then
(2) search resulting graph with shortest path algorithm

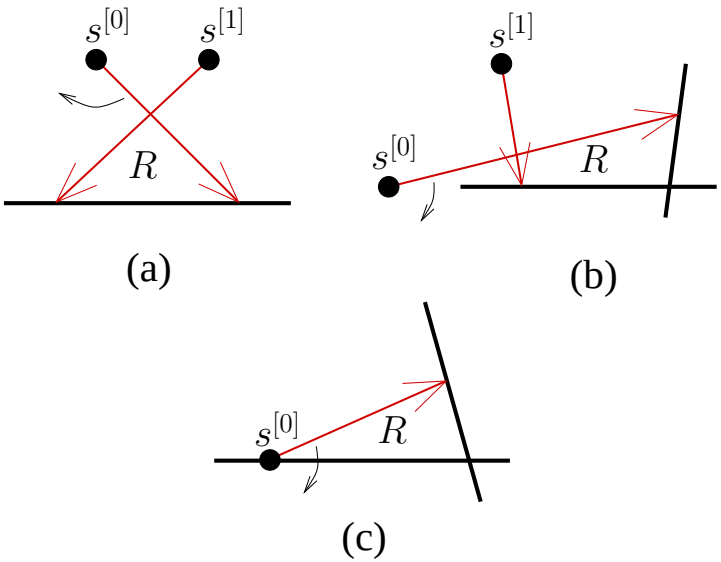
*First applied to visibility-based pursuit-evasion in [Guibas et al, 1999]

Definition: Maximal Nonseparable Region

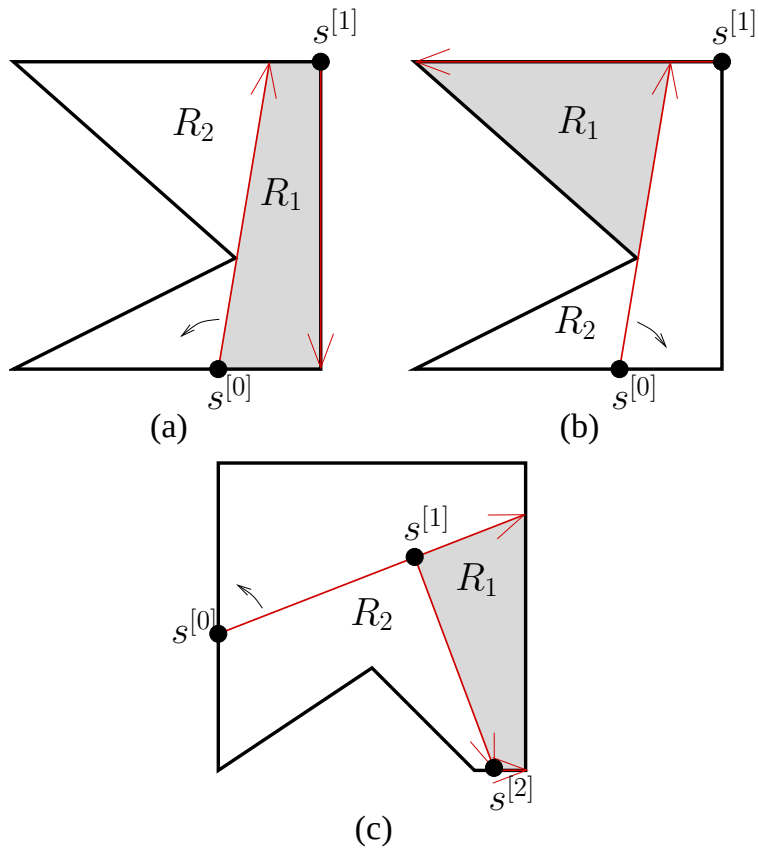


Maximal nonseparable regions undergo discrete **topological changes**:

disappear/appear



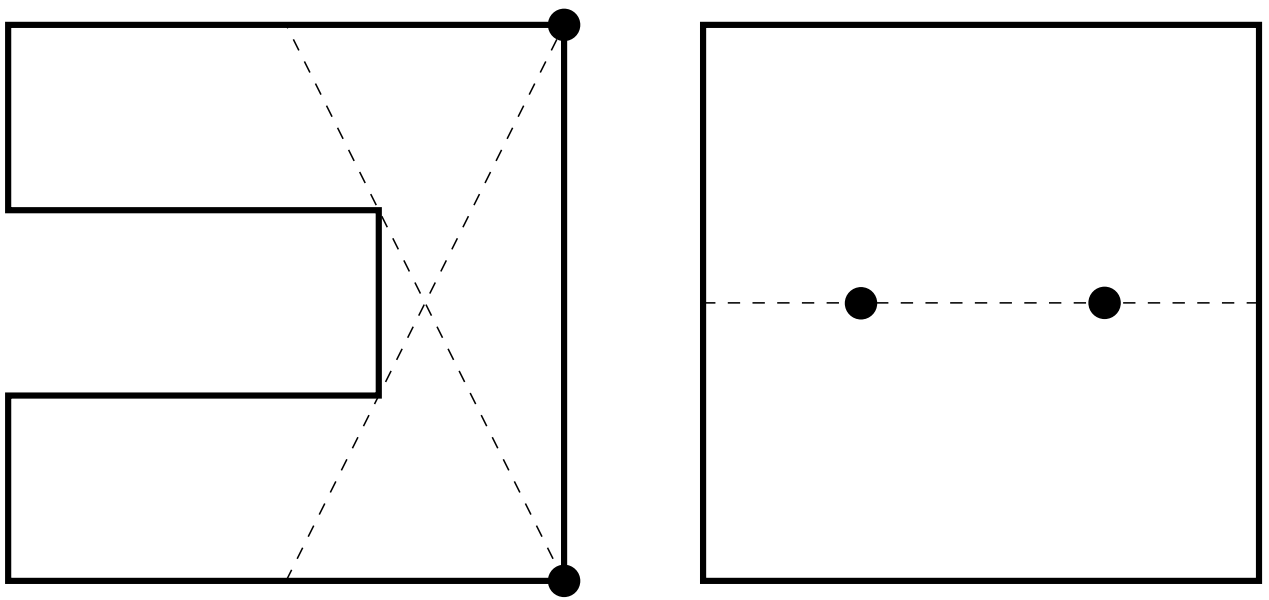
merge/split



Definition: Critical Angles

A searchlight has a **critical angle** wherever

1. it is aimed along a wall,
2. its visibility is occluded by a reflex vertex, or
3. it has line of sight to another searchlight

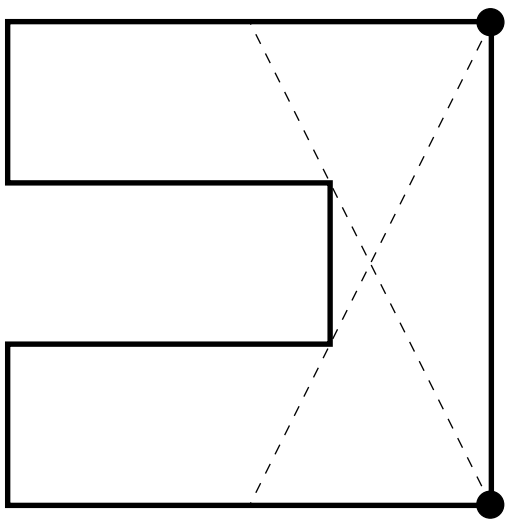


Reduction Theorem

Without loss of generality, searchlights may

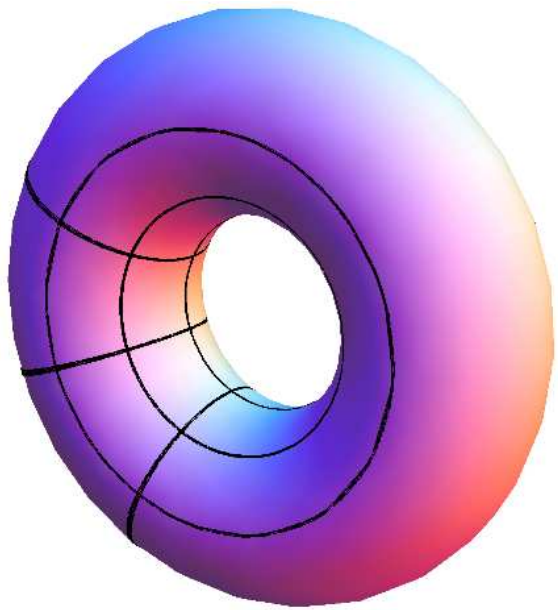
1. rotate only one at a time, and
2. only stop at critical angles

critical angles



$\Rightarrow$

roadmap in $\mathbb{T}^N$

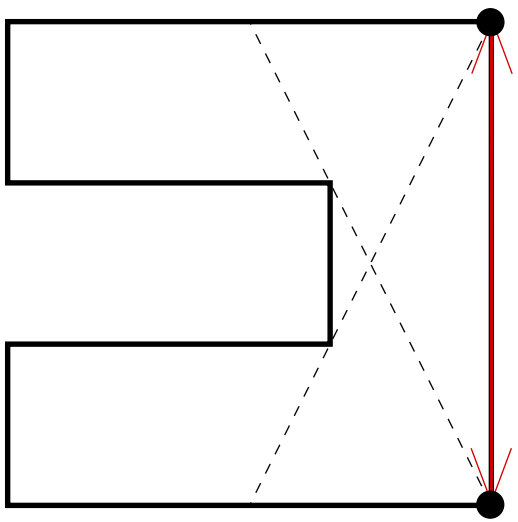


Reduction Theorem

Without loss of generality, searchlights may

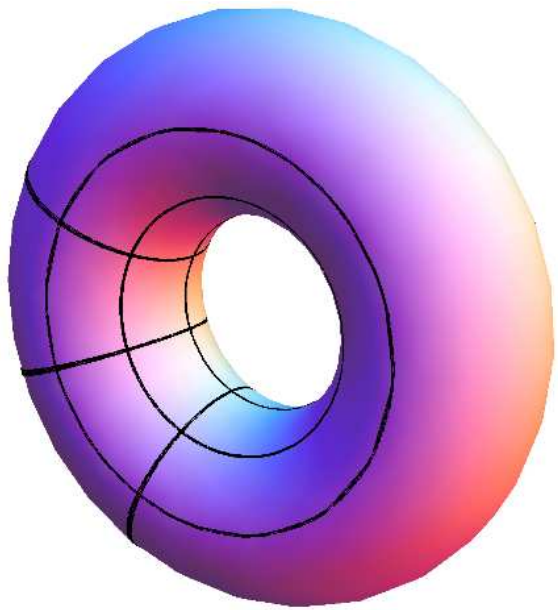
1. rotate only one at a time, and
2. only stop at critical angles

critical angles



$\Rightarrow$

roadmap in $\mathbb{T}^N$

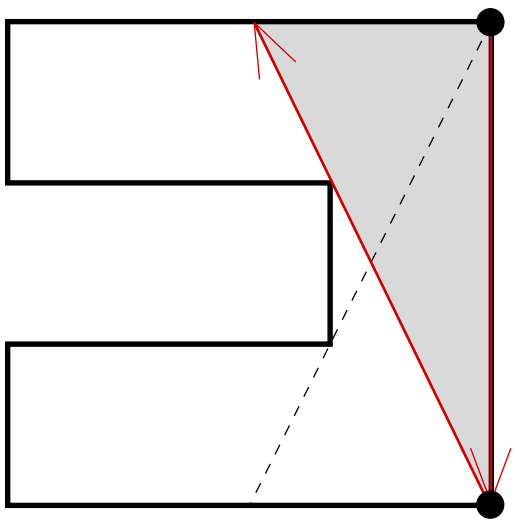


Reduction Theorem

Without loss of generality, searchlights may

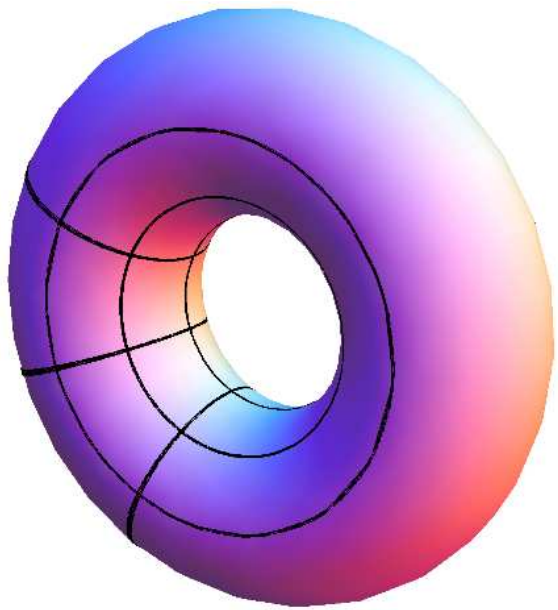
1. rotate only one at a time, and
2. only stop at critical angles

critical angles



$\Rightarrow$

roadmap in $\mathbb{T}^N$

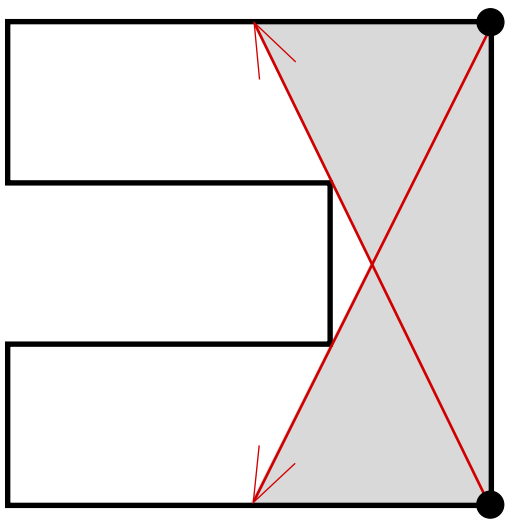


Reduction Theorem

Without loss of generality, searchlights may

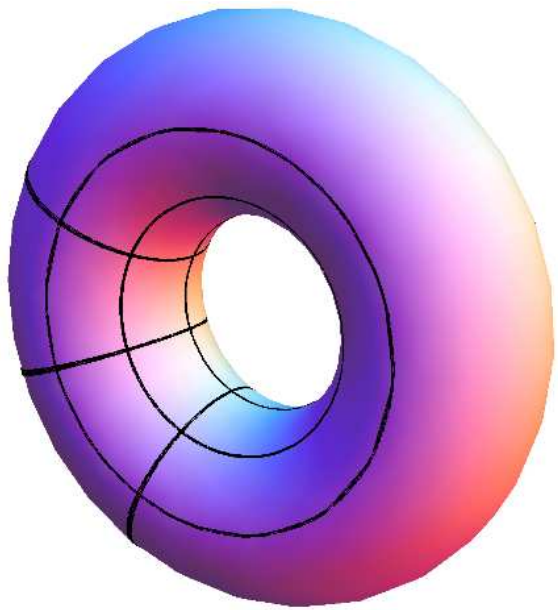
1. rotate only one at a time, and
2. only stop at critical angles

critical angles



$\Rightarrow$

roadmap in $\mathbb{T}^N$

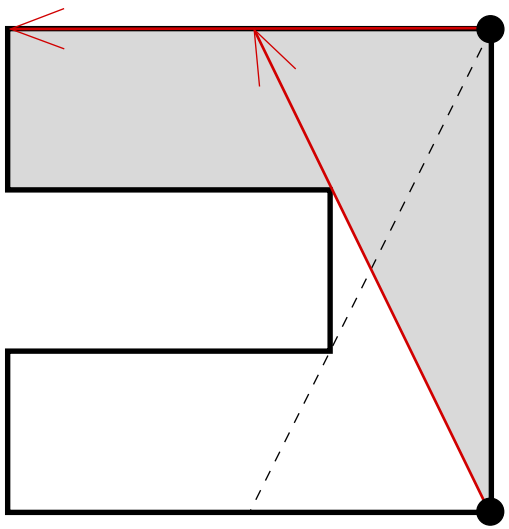


Reduction Theorem

Without loss of generality, searchlights may

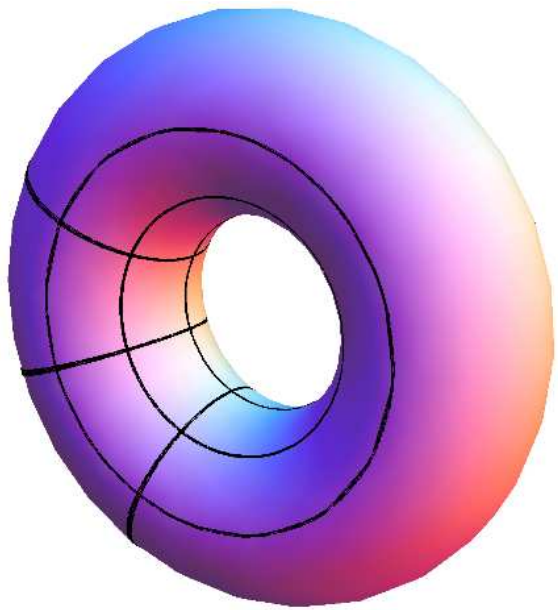
1. rotate only one at a time, and
2. only stop at critical angles

critical angles



$\Rightarrow$

roadmap in $\mathbb{T}^N$

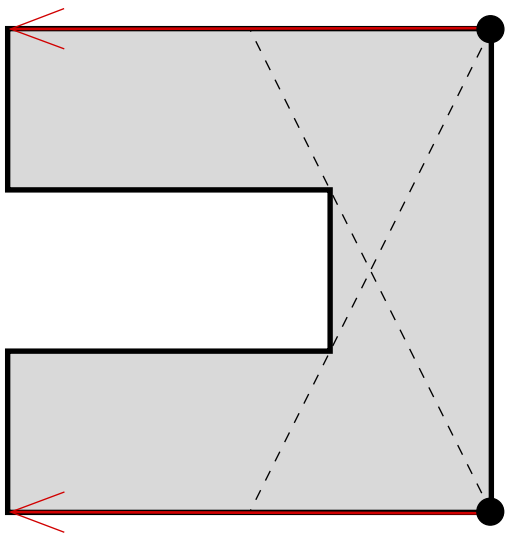


Reduction Theorem

Without loss of generality, searchlights may

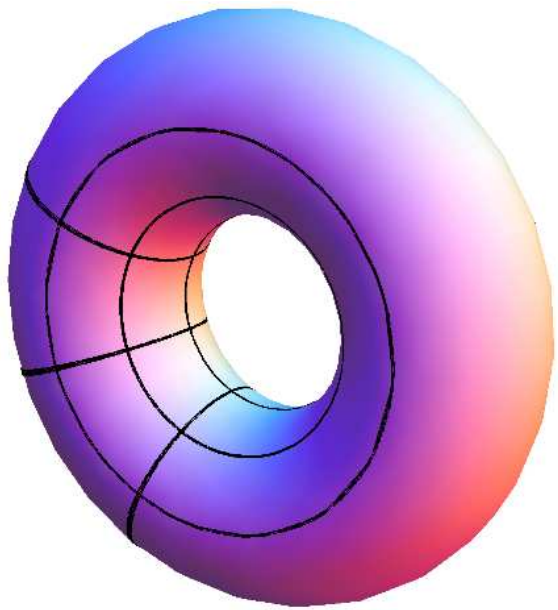
1. rotate only one at a time, and
2. only stop at critical angles

critical angles



$\Rightarrow$

roadmap in $\mathbb{T}^N$



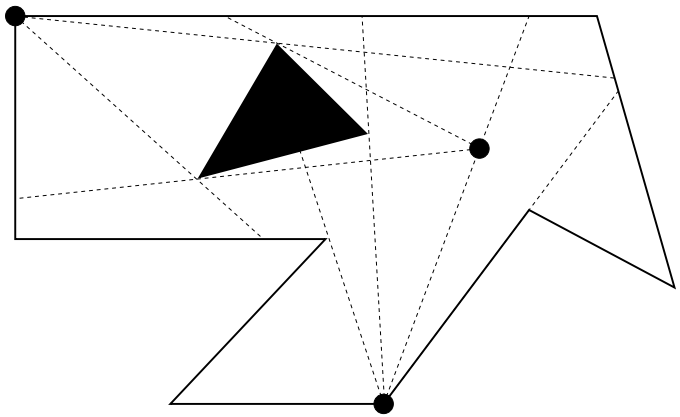
Combinatorial Roadmap Method

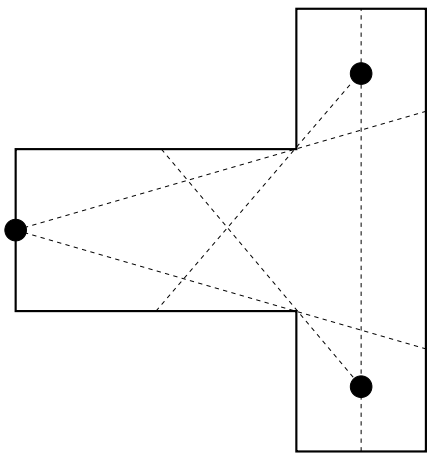
1. Find critical angles
2. Construct roadmap
3. Perform modified breadth-first search of roadmap

Computed Examples

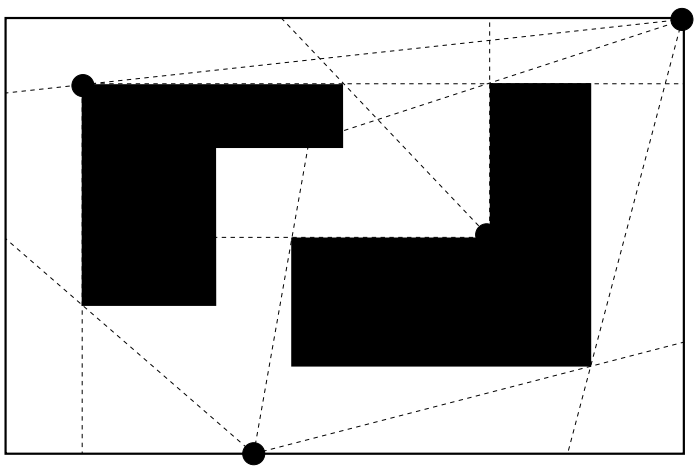
C++, 2.33 GHz i686

solutions in < 1 **second**





solution in < 4 **seconds**



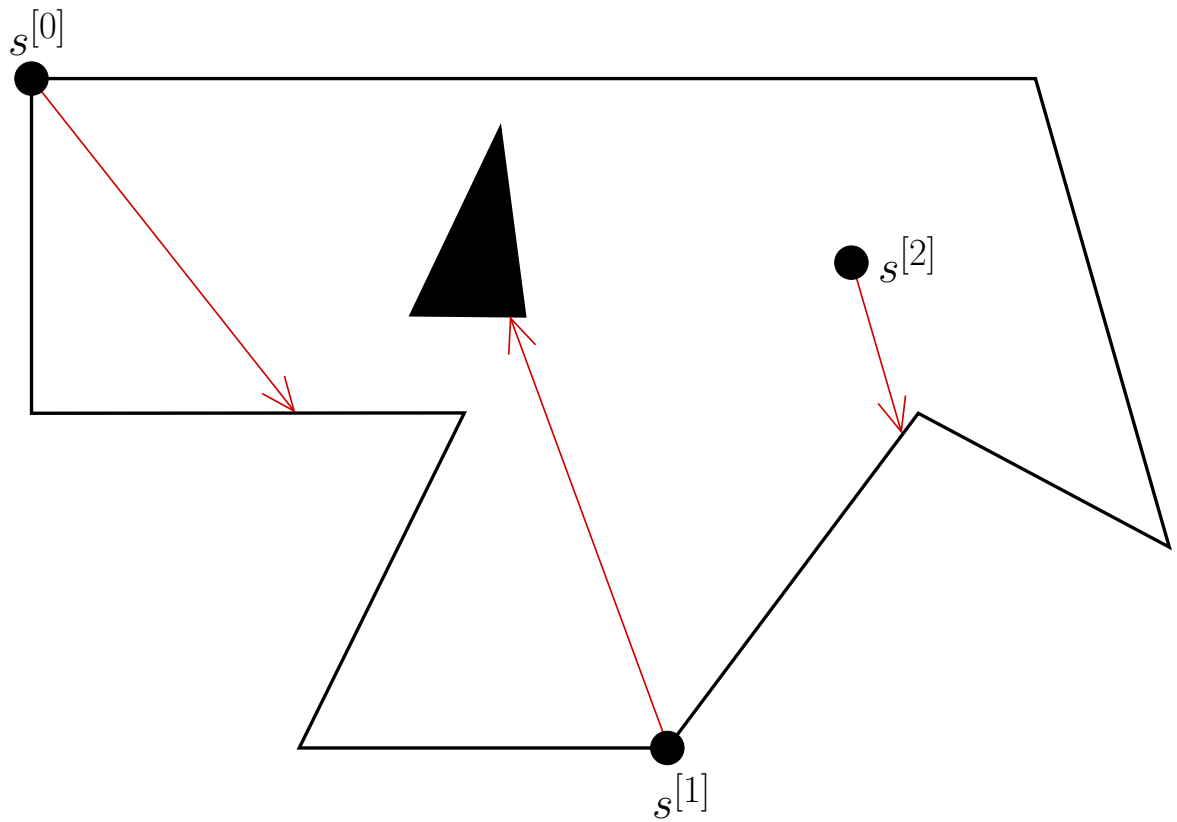
Computed Examples

C++, 2.33 GHz i686

Node Bound	**Nodes Visited**	**Computation Time**	# Actions
$\sim 10^{10}$	**1321**	$<$ **1 sec.**	12
$\sim 10^{6}$	**463**	$<$ **1 sec.**	8
$\sim 10^{9}$	**6176**	$<$ **4 sec.**	16

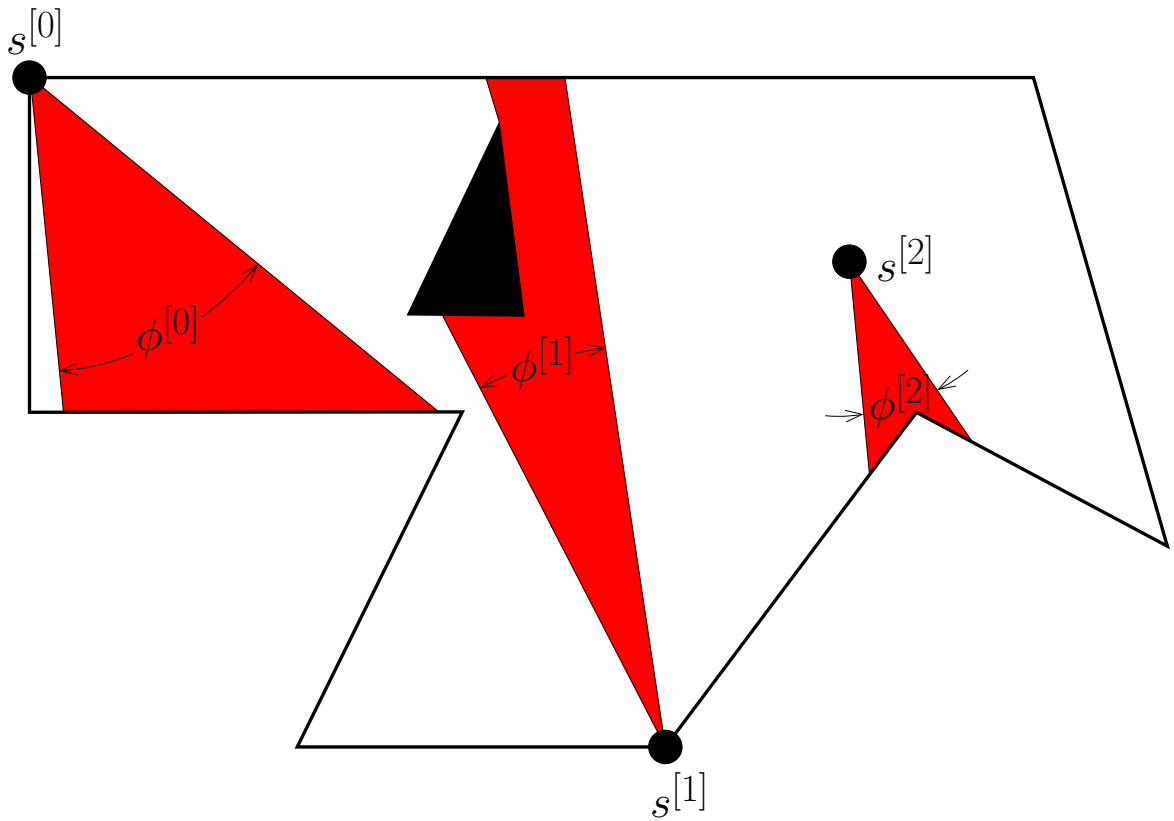
Goal

Given N **ϕ-searchlights** with FOVs $\phi^{[0]}, \phi^{[1]}, \phi^{[2]}, \ldots, \phi^{[N-1]}$, find a search schedule



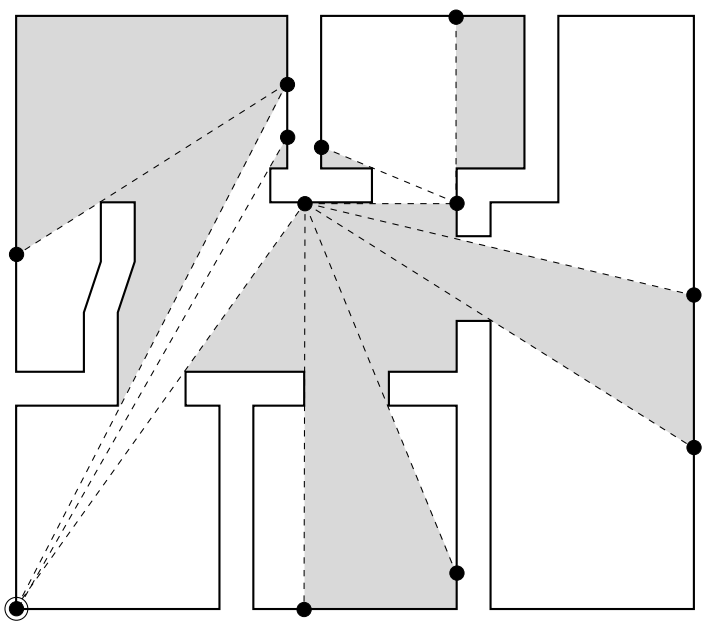
Goal

Given N **ϕ-searchlights** with FOVs $\phi^{[0]}, \phi^{[1]}, \phi^{[2]}, \ldots, \phi^{[N-1]}$, find a search schedule



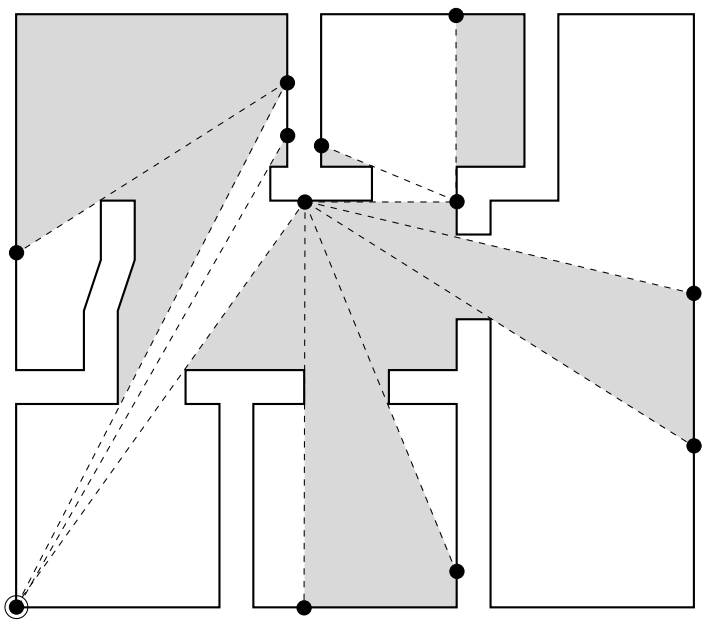
Serendipity: Searchlight Placement Scheme

Reflex Vertex Straddling (**RVS**) Positions



Serendipity: Searchlight Placement Scheme

Reflex Vertex Straddling (**RVS**) Positions



Theorem

RVS requires

- $\leq r + 1 \leq n - 2$ searchlights, or $\leq \frac{n}{2} - 2$ in ortho. environment
- time to clear $\mathcal{O}(r)$

Note: n:= total number of vertices
r:= number of reflex vertices

Part II

Multi-Agent Deployment

Distributed Deployment Problem

Goal

Design a distributed algorithm for agents to deploy into an environment such that

1. *line-of-sight connectivity is maintained, and*
2. *entire environment is visible from final positions*

Assumptions

- unmapped static polygonal environment (n vertices, h holes)
- agents identical except for UIDs
- omnidirectional vision*
- only line-of-site communication and sensing
- agents may establish only local common reference frame
- first order dynamics: $\dot{p}^{[i]} = u \leq u_{\max}$

Algorithm Design Requirements

Upper bounds on

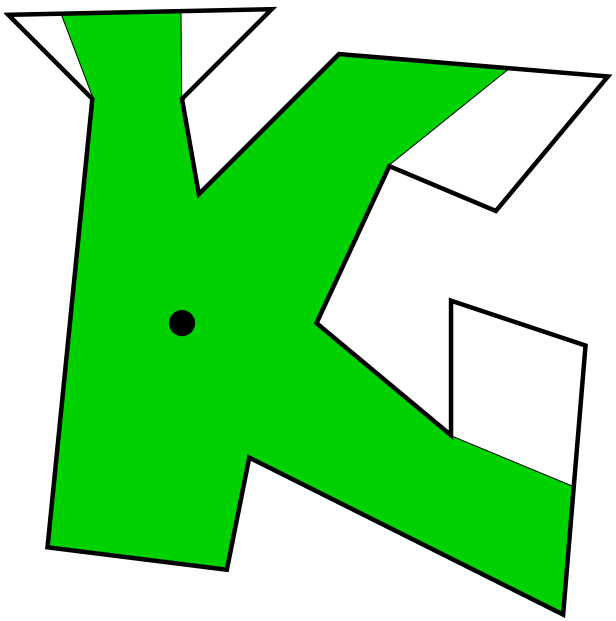
- **number of agents** required
- **time** until full coverage
- **memory** and **communication** complexity

1. Design centralized incremental partition algorithm
2. Distributedly emulate

Definitions: Visibility Polygons

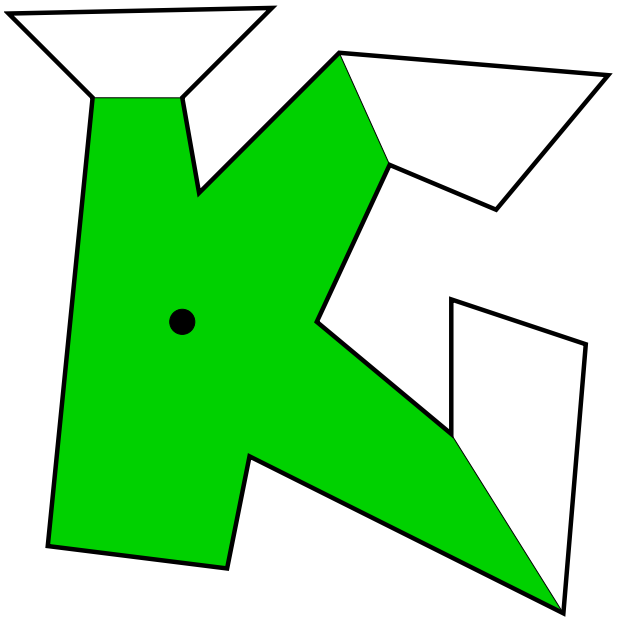
Visibility Polygon

sensing and communication



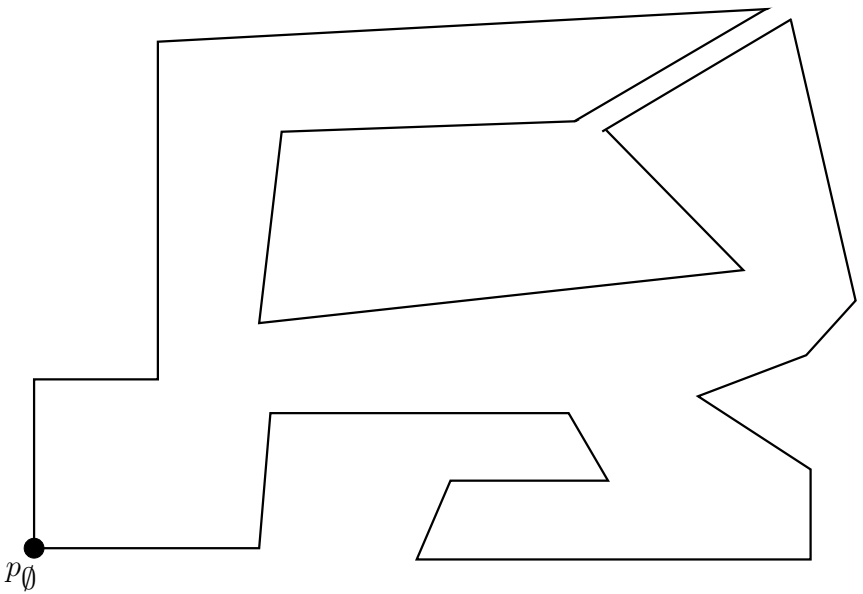
Vertex-Limited Visibility Polygon

land grabbing



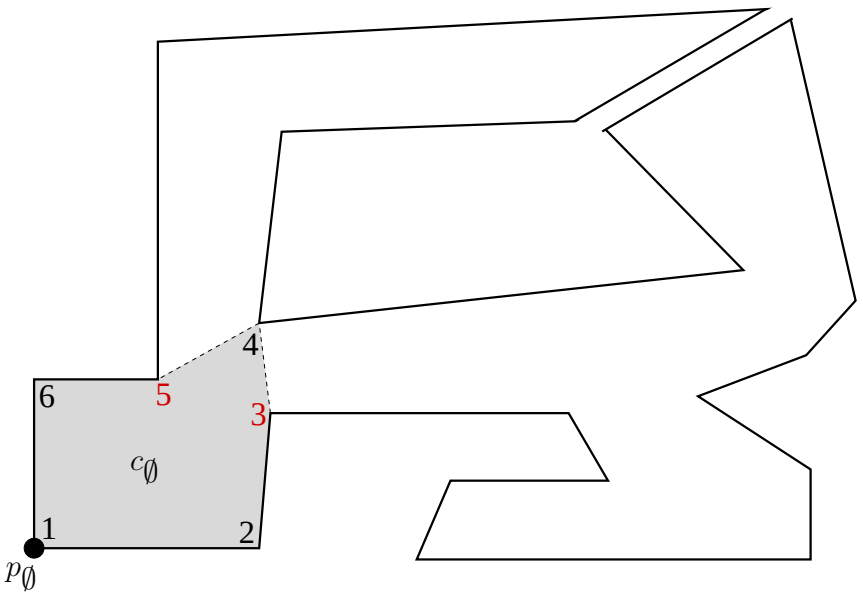
Centralized Incremental Partition Algorithm

Without Holes [Ganguli et al, 2007]



Centralized Incremental Partition Algorithm

Without Holes [Ganguli et al, 2007]

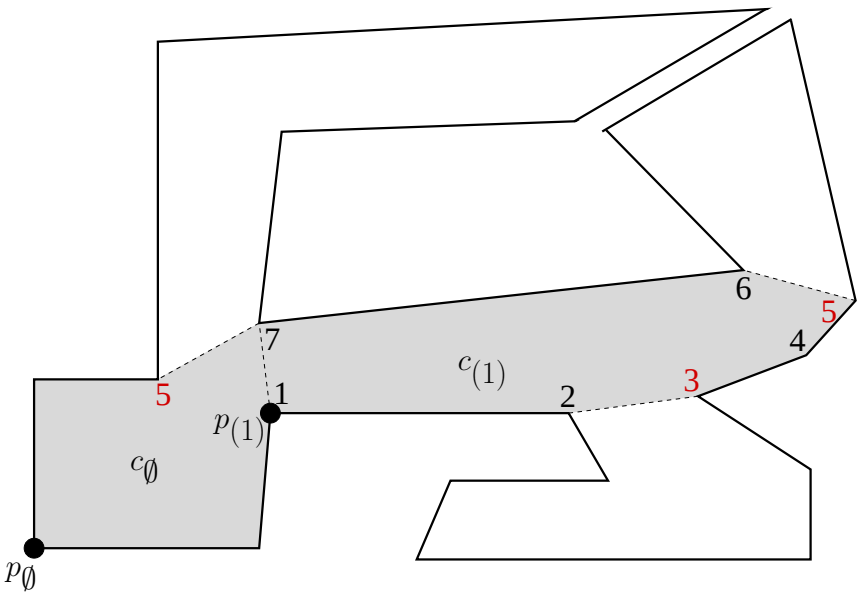


Partition Tree

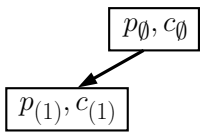
$p_\emptyset, c_\emptyset$

Centralized Incremental Partition Algorithm

Without Holes [Ganguli et al, 2007]

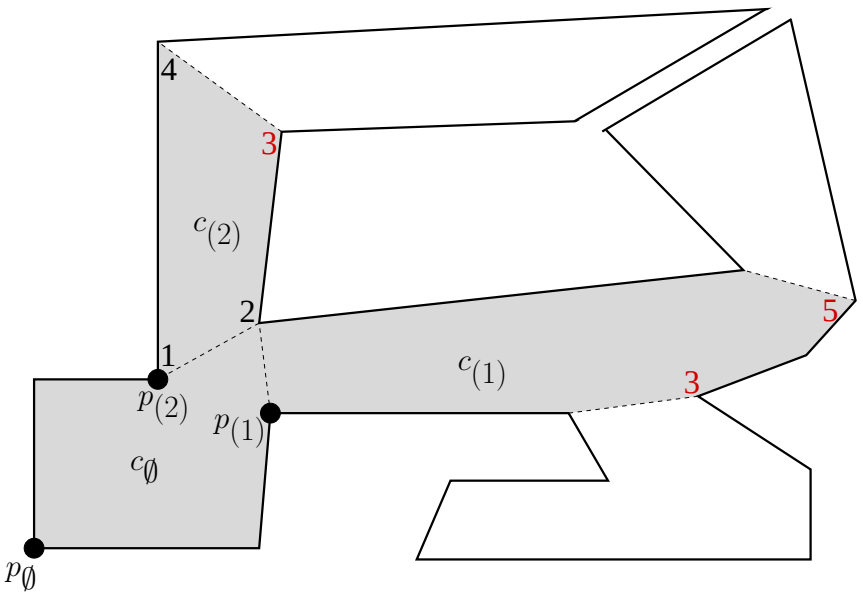


Partition Tree

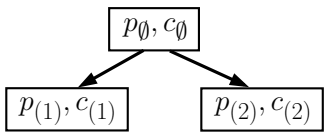


Centralized Incremental Partition Algorithm

Without Holes [Ganguli et al, 2007]

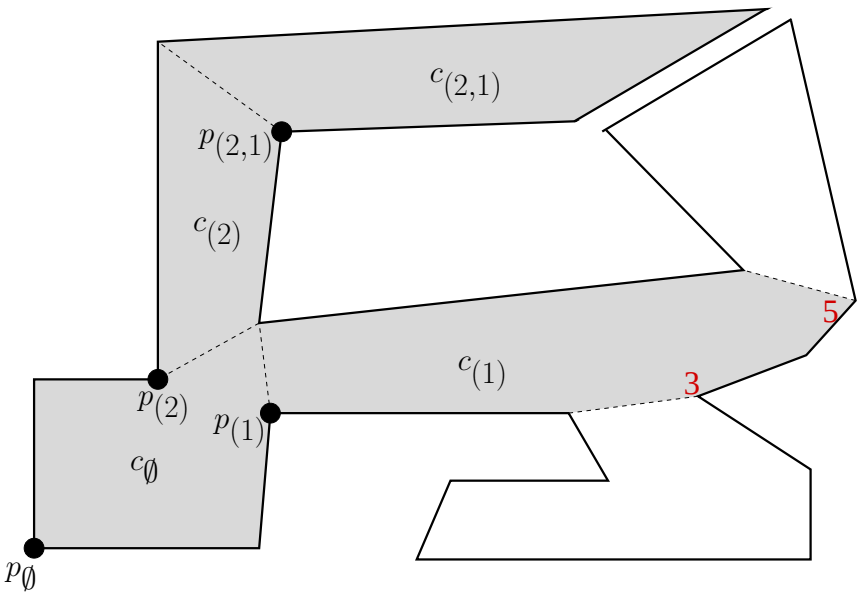


Partition Tree

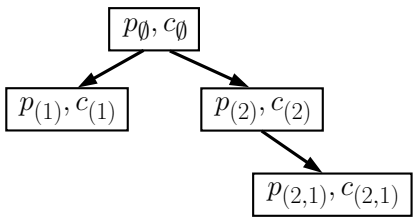


Centralized Incremental Partition Algorithm

Without Holes [Ganguli et al, 2007]

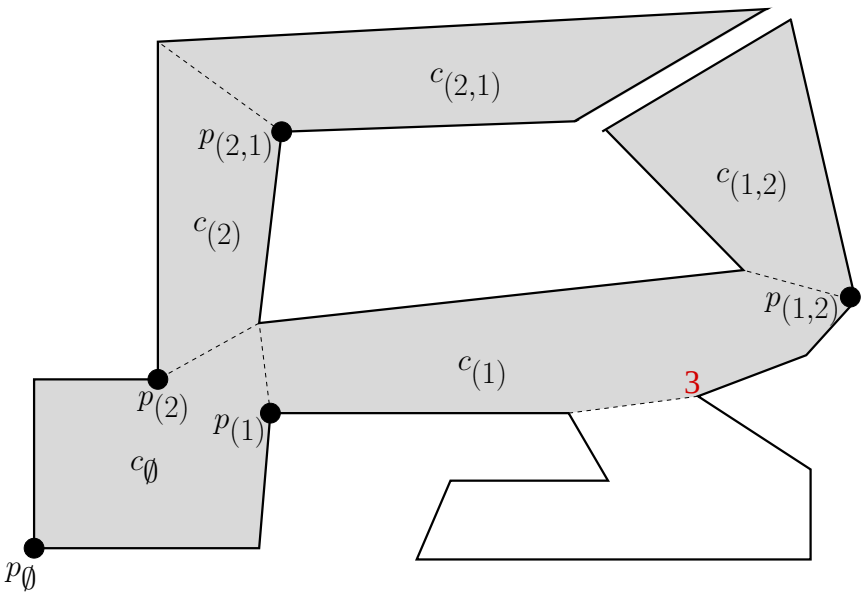


Partition Tree

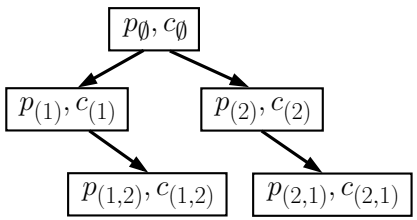


Centralized Incremental Partition Algorithm

Without Holes [Ganguli et al, 2007]

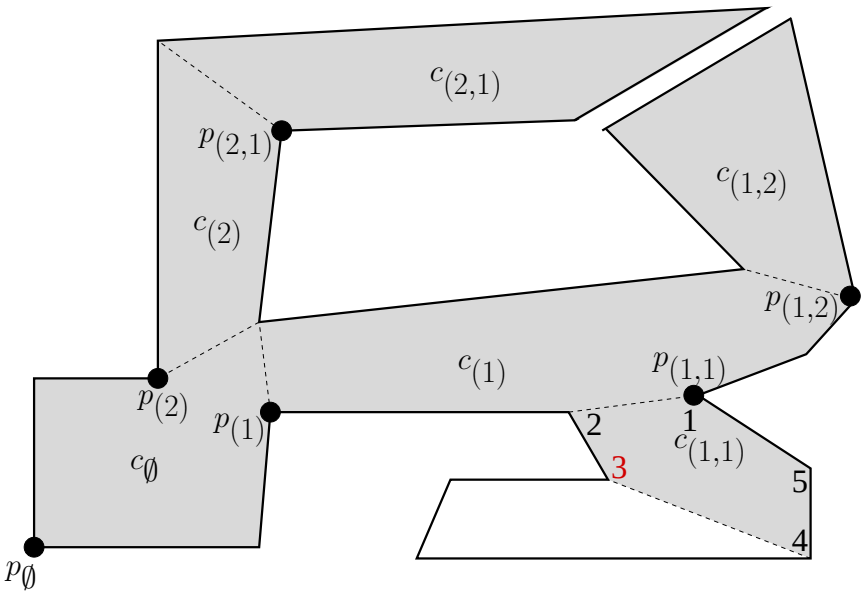


Partition Tree

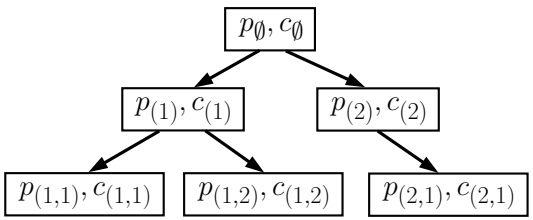


Centralized Incremental Partition Algorithm

Without Holes [Ganguli et al, 2007]

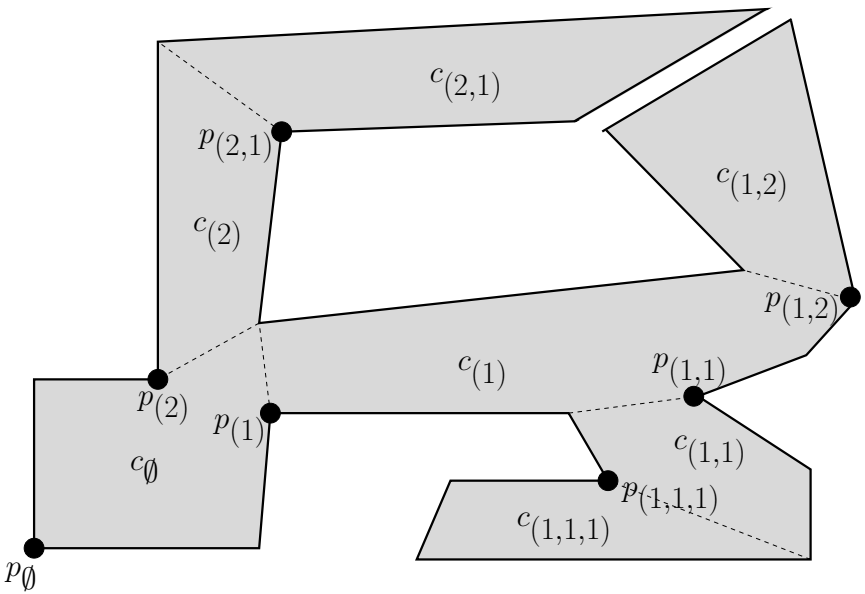


Partition Tree

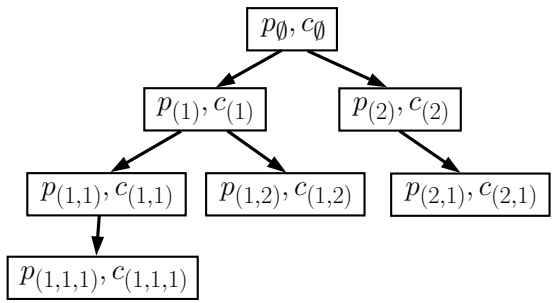


Centralized Incremental Partition Algorithm

Without Holes [Ganguli et al, 2007]



Partition Tree



Centralized Incremental Partition Algorithm

Without Holes [Ganguli et al, 2007]

while(there is an unexplored gap edge g_ξ)

pick vantage point p_ξ on g_ξ;

construct cell c_ξ from vertex-limited visibility polygon;

insert vertex (p_ξ, c_ξ) into partition tree;

Centralized Incremental Partition Algorithm

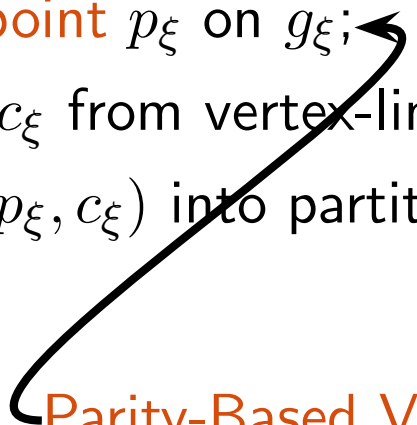
Without Holes [Ganguli et al, 2007]

while(there is an unexplored gap edge g_ξ)

pick vantage point p_ξ on g_ξ;

construct cell c_ξ from vertex-limited visibility polygon;

insert vertex (p_ξ, c_ξ) into partition tree;

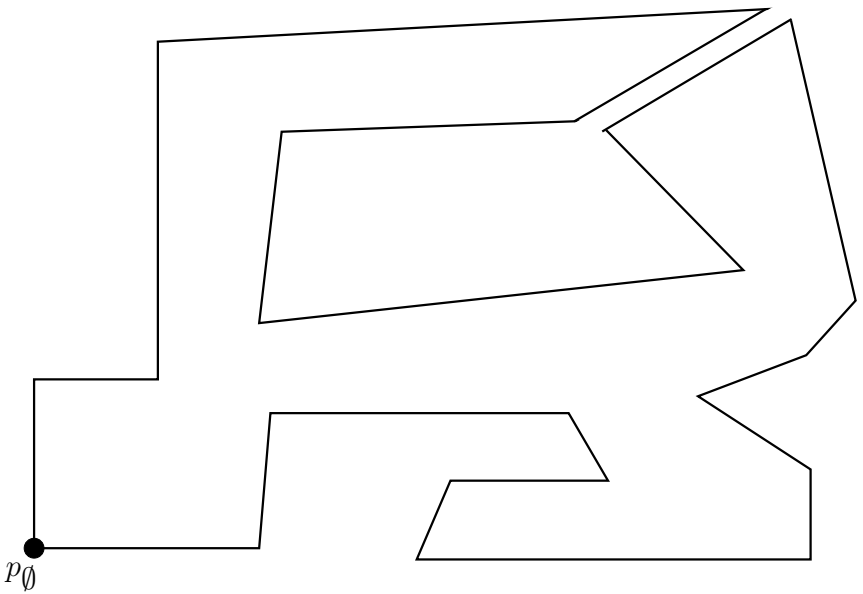


Parity-Based Vantage Point Selection Scheme

typical case: enumerate vertices of parent cell, choose odd vertex on g_ξ

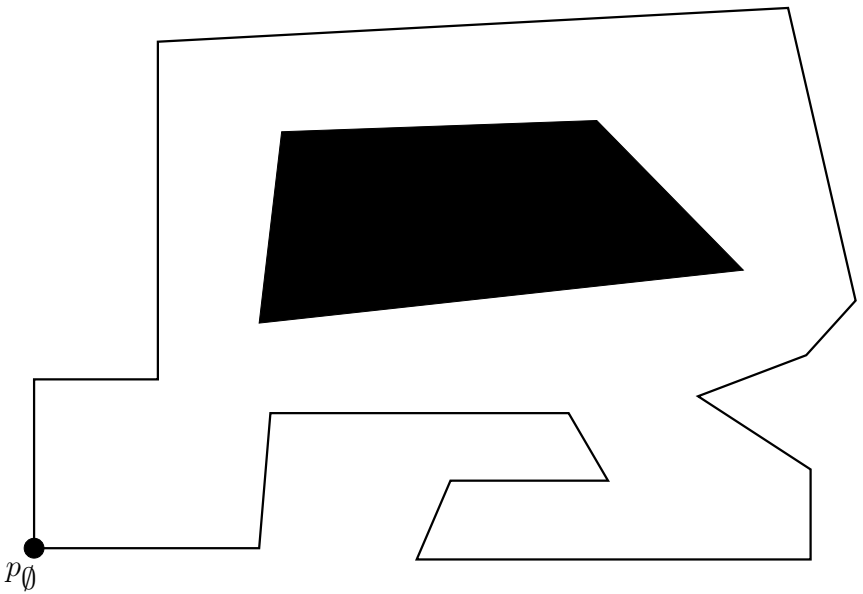
Centralized Incremental Partition Algorithm

What happens with holes?



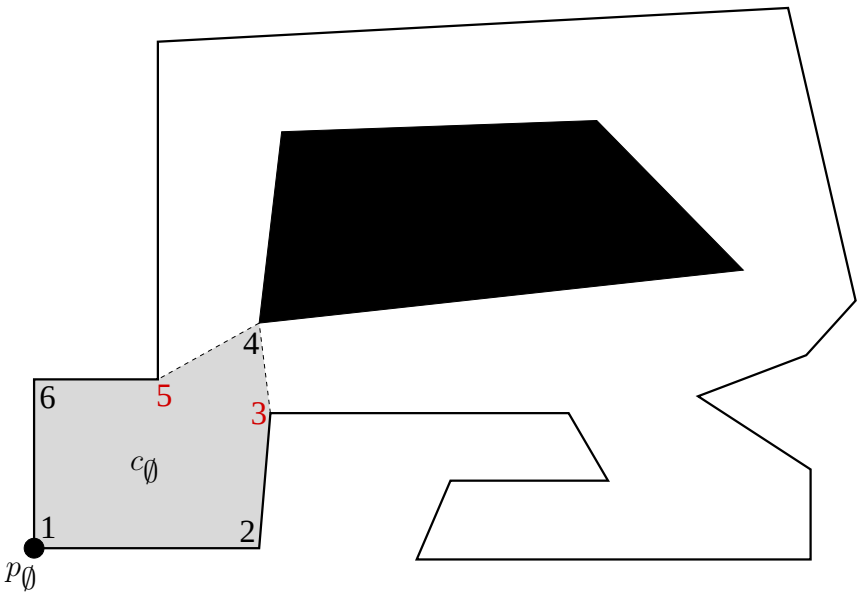
Centralized Incremental Partition Algorithm

What happens with holes?



Centralized Incremental Partition Algorithm

What happens with holes?

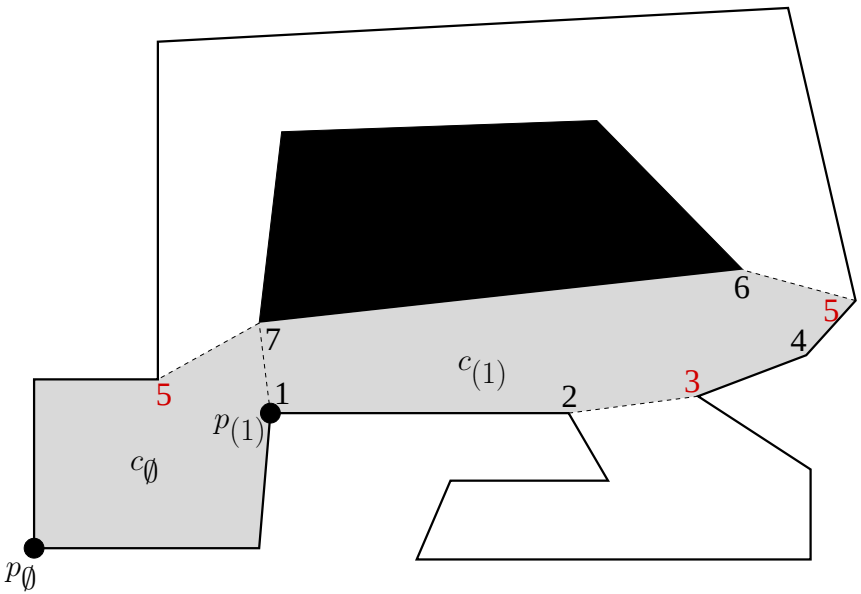


Partition Tree

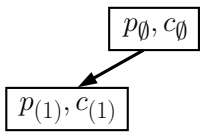
$p_\emptyset, c_\emptyset$

Centralized Incremental Partition Algorithm

What happens with holes?

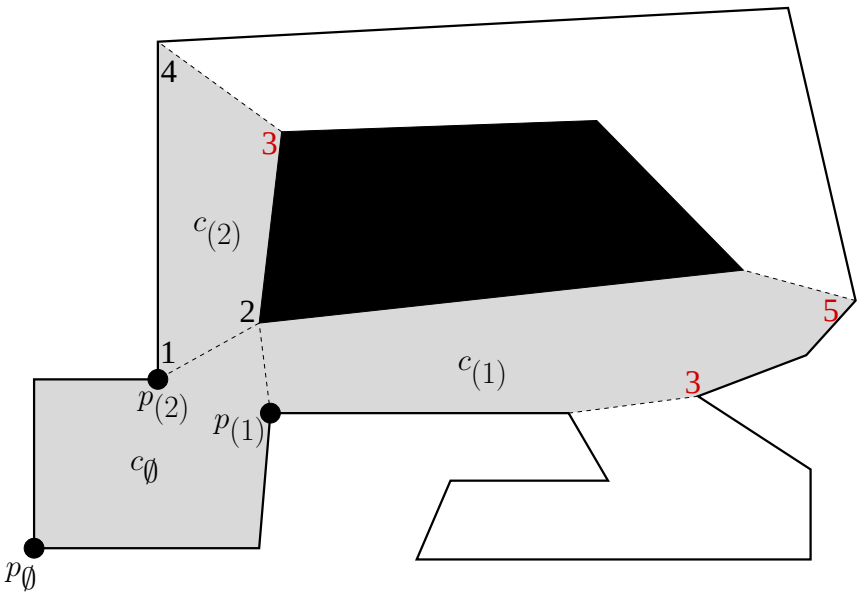


Partition Tree

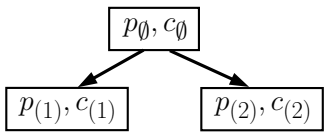


Centralized Incremental Partition Algorithm

What happens with holes?

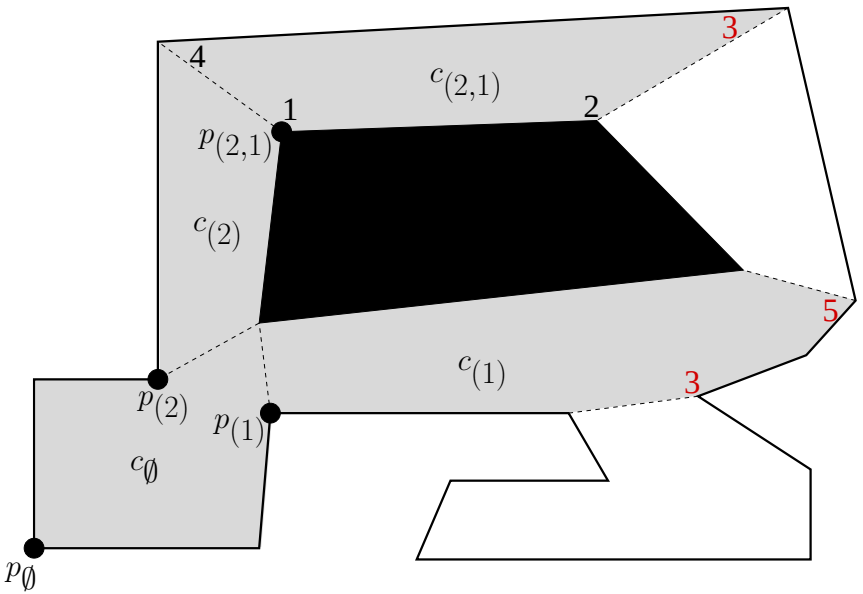


Partition Tree

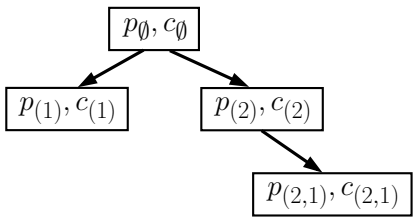


Centralized Incremental Partition Algorithm

What happens with holes?

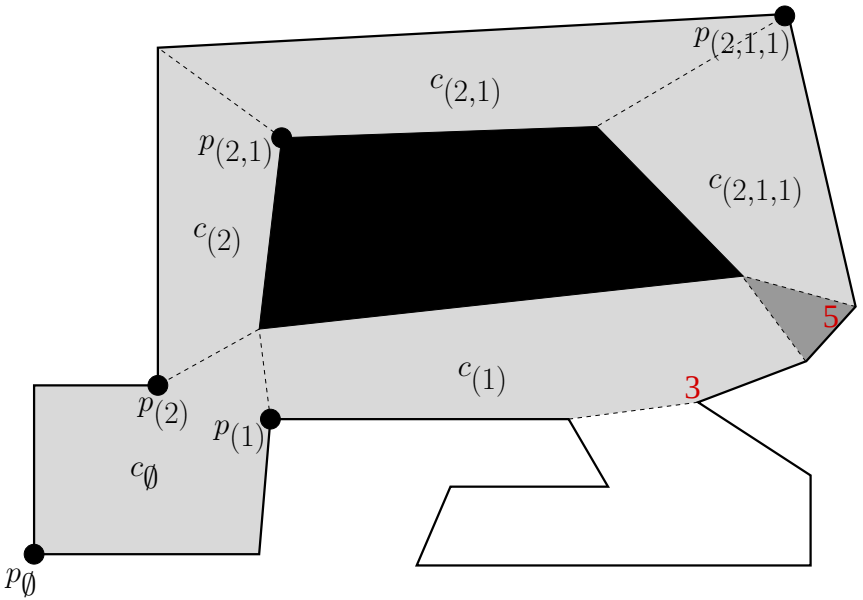


Partition Tree

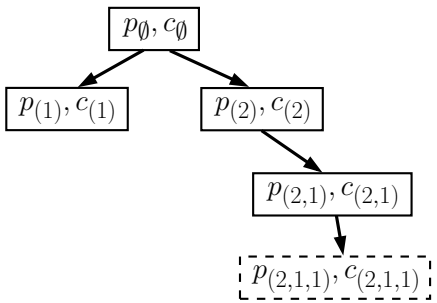


Centralized Incremental Partition Algorithm

What happens with holes?

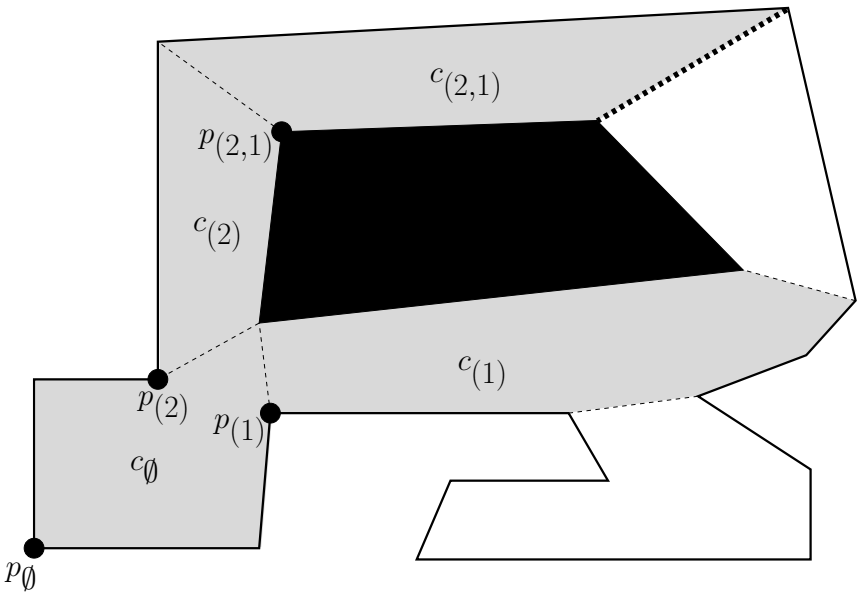


Partition Tree

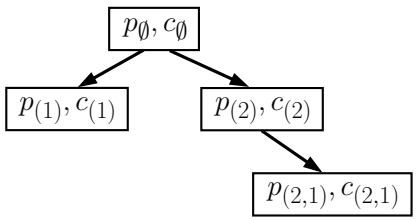


Centralized Incremental Partition Algorithm

What happens with holes?

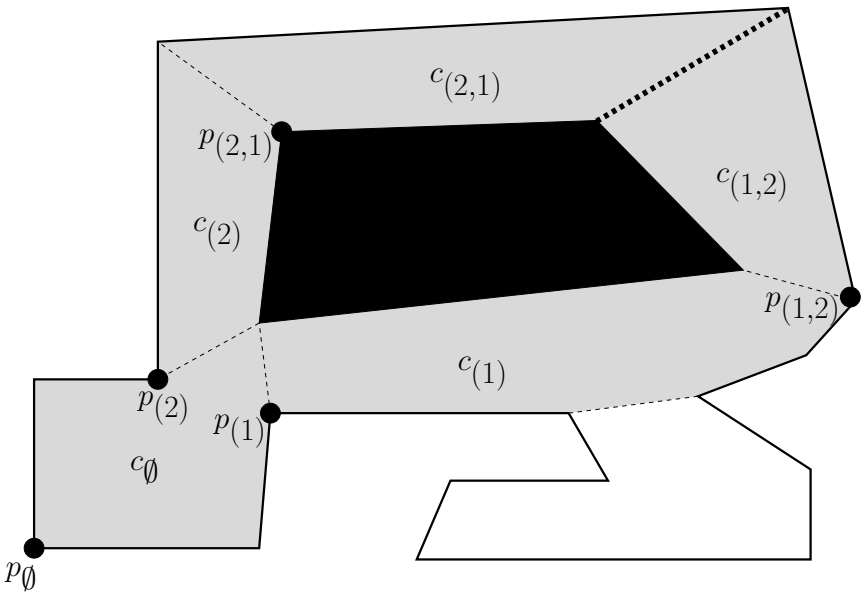


Partition Tree

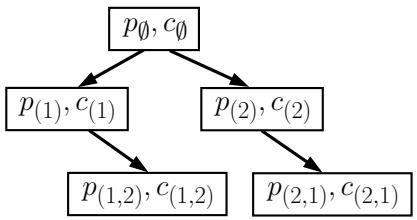


Centralized Incremental Partition Algorithm

What happens with holes?

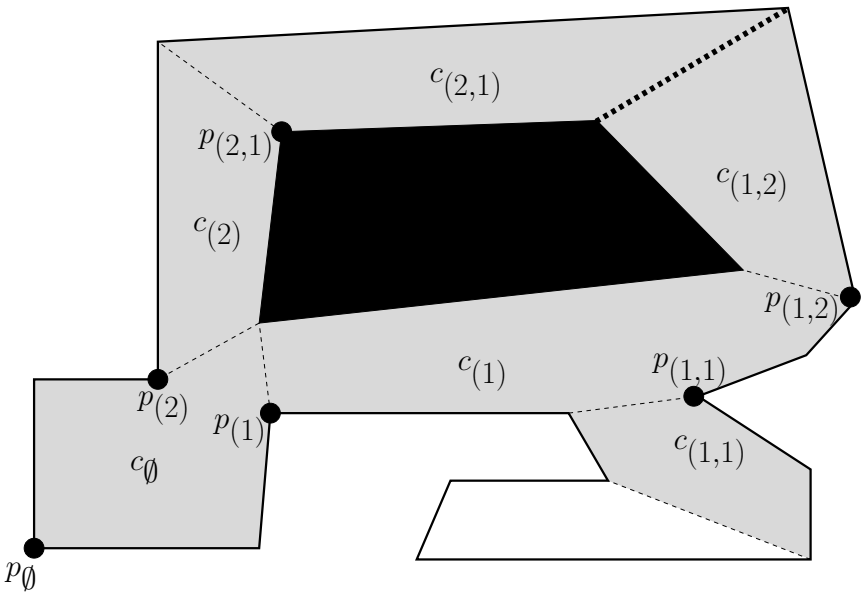


Partition Tree

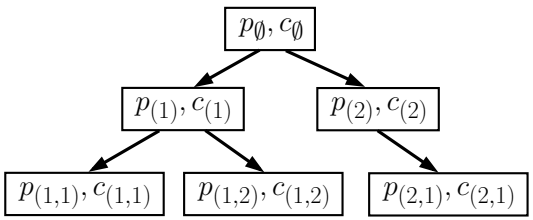


Centralized Incremental Partition Algorithm

What happens with holes?

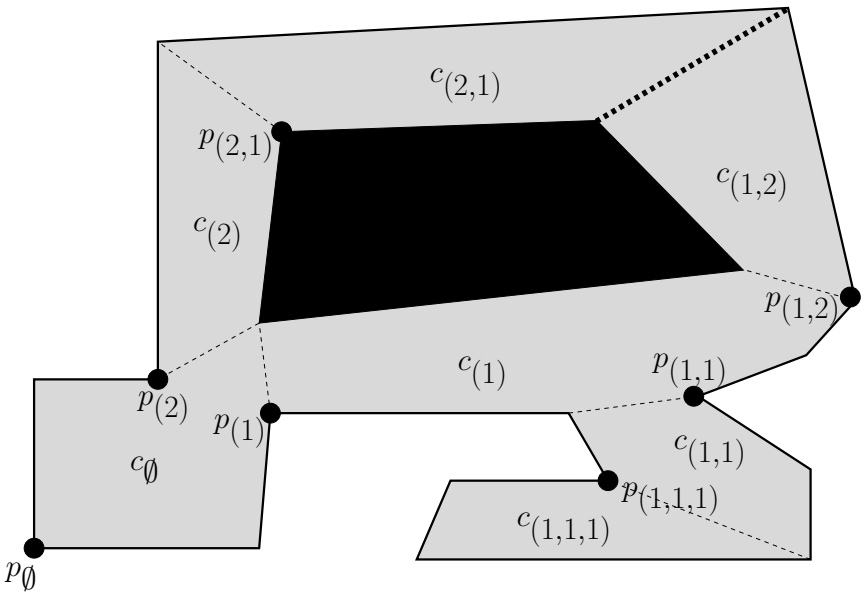


Partition Tree

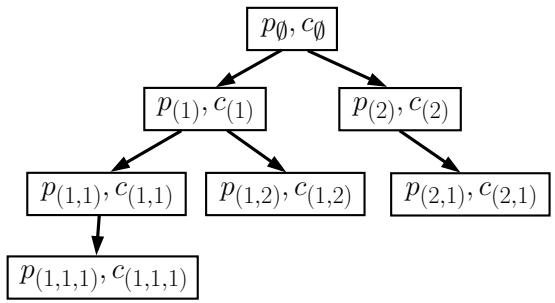


Centralized Incremental Partition Algorithm

What happens with holes?



Partition Tree



while(there is an unexplored gap edge g_ξ)

pick vantage point p_ξ on g_ξ;

construct cell c_ξ from vertex-limited visibility polygon;

insert vertex (p_ξ, c_ξ) into partition tree;

while(there is an unexplored gap edge g_ξ)

pick vantage point p_ξ on g_ξ;

construct cell c_ξ from vertex-limited visibility polygon;

if(c_ξ is in **branch conflict**)

discard vertex (p_ξ, c_ξ) and mark phantom wall at g_ξ;

else

insert vertex (p_ξ, c_ξ) into partition tree;

Convergence Theorem

The incremental partition algorithm

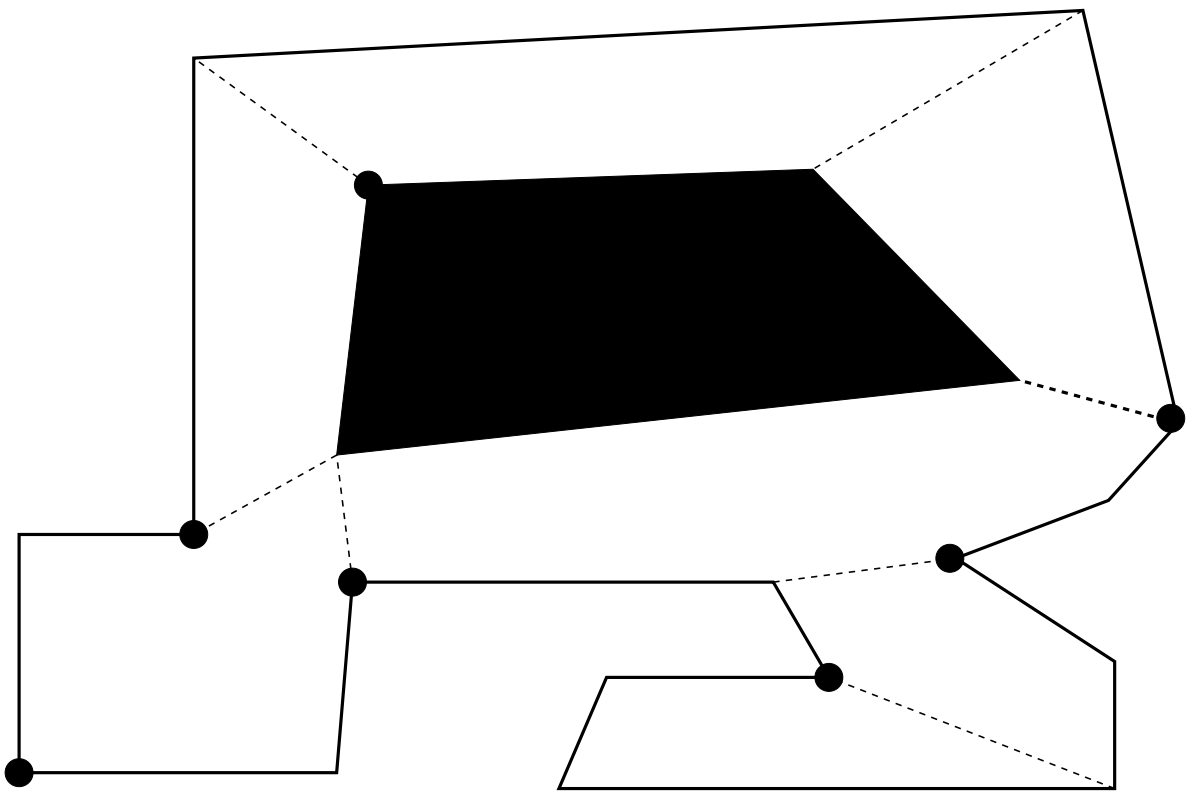
1. converges to full coverage in finite time,
2. the vantage points' visibility graph is connected, and
3. number of vantage points $N \leq \left\lfloor \frac{n+2h-1}{2} \right\rfloor$

Proof:

There are $n + 2h - 2$ triangles in any triangulation.

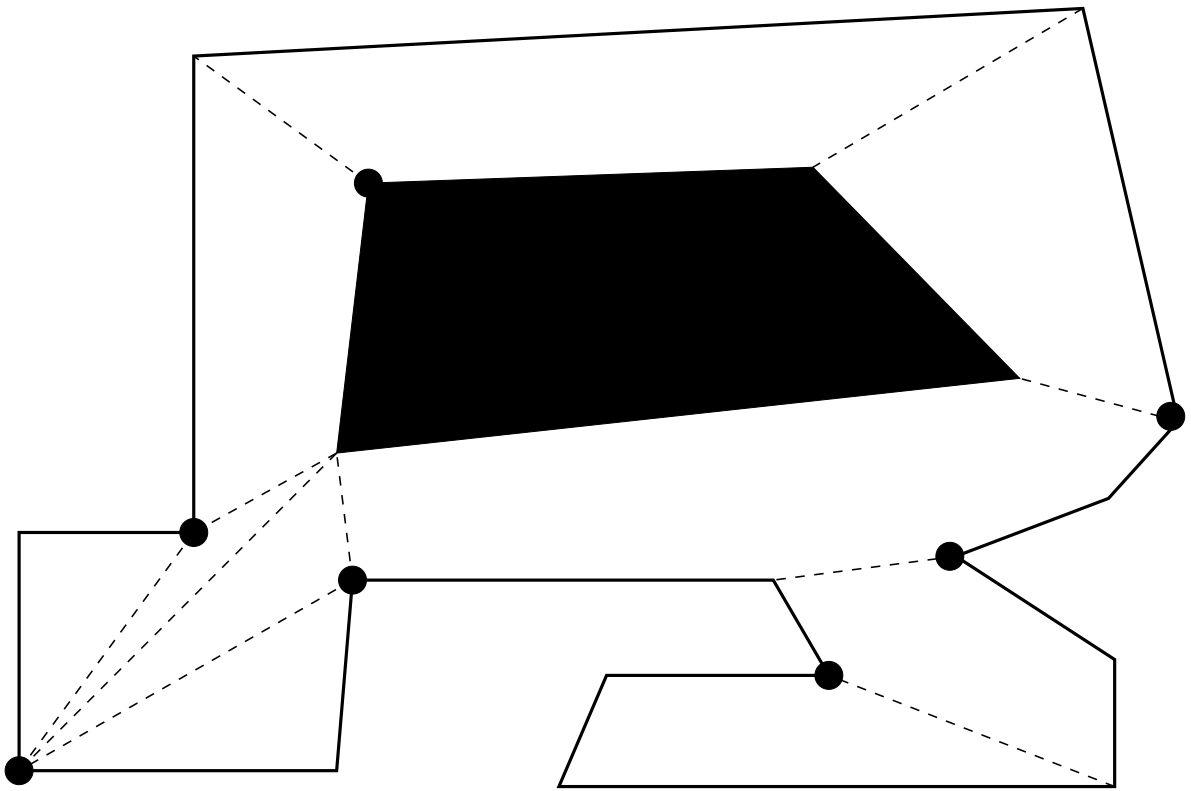
Centralized Incremental Partition Algorithm

Properties



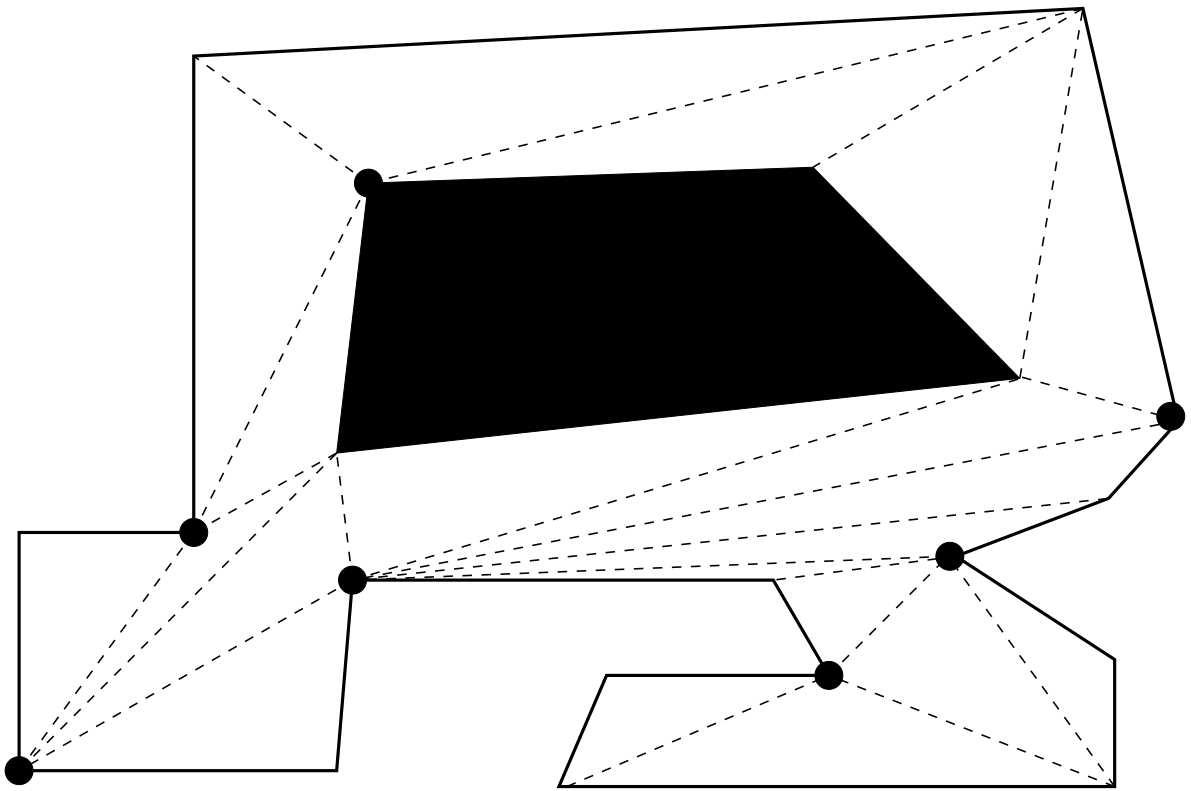
Centralized Incremental Partition Algorithm

Properties



Centralized Incremental Partition Algorithm

Properties



Convergence Theorem

The incremental partition algorithm

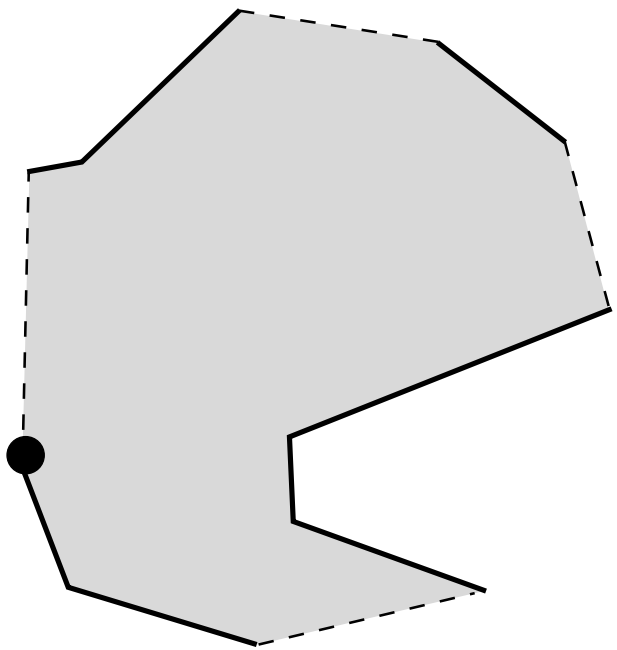
1. converges to full coverage in finite time,
2. the vantage points' visibility graph is connected, and
3. number of vantage points $N \leq \left\lfloor \frac{n+2h-1}{2} \right\rfloor$.

Proof:

There are $n + 2h - 2$ triangles in any triangulation.
Parity-based vantage point selection scheme $\Rightarrow$
Can assign 1 triangle to root vantage point,
2 triangles to other vantage points
$\Rightarrow 1 + 2(N - 1) \leq n + 2h - 2$

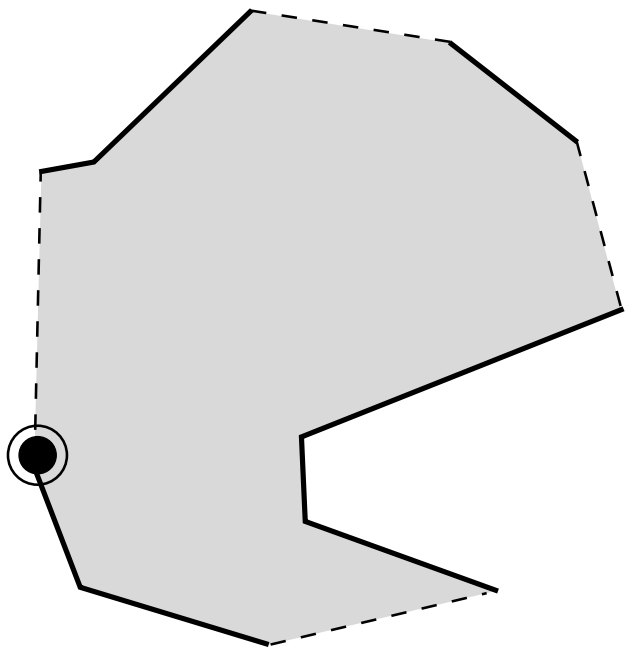
Depth-First Search

if(not all children visited)
    move to next child;
else
    move to parent;



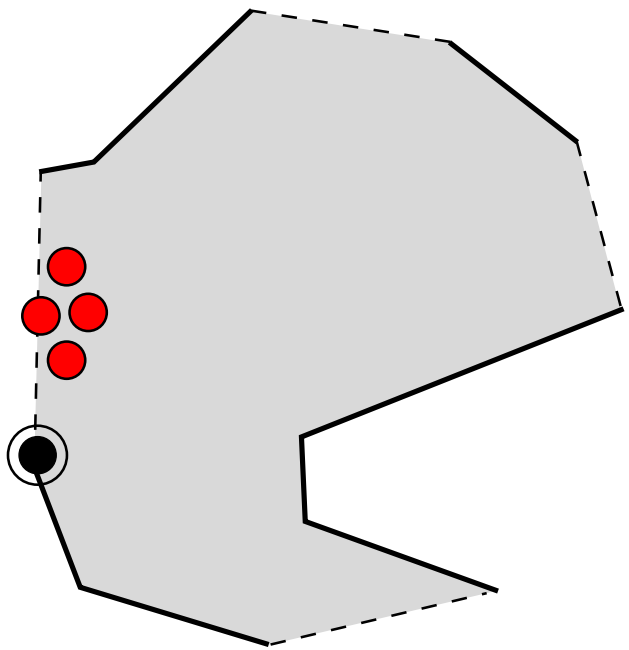
Depth-First Search

if(not all children visited)
    move to next child;
else
    move to parent;



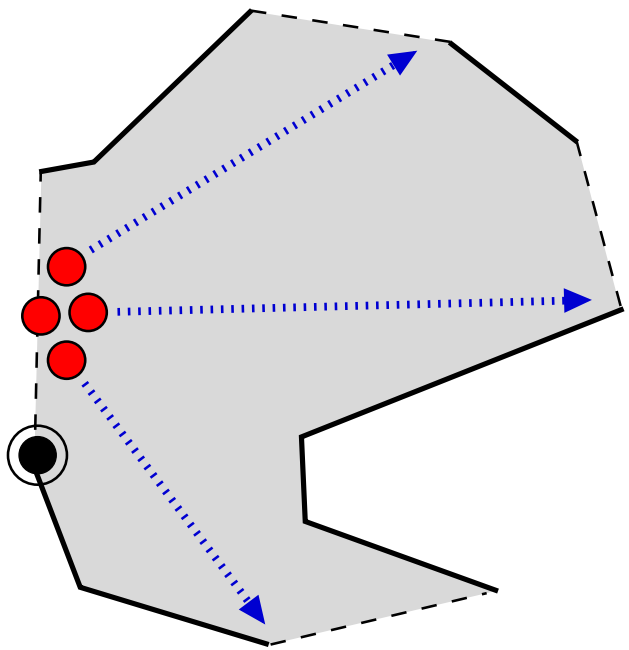
Depth-First Search

if(not all children visited)
    move to next child;
else
    move to parent;



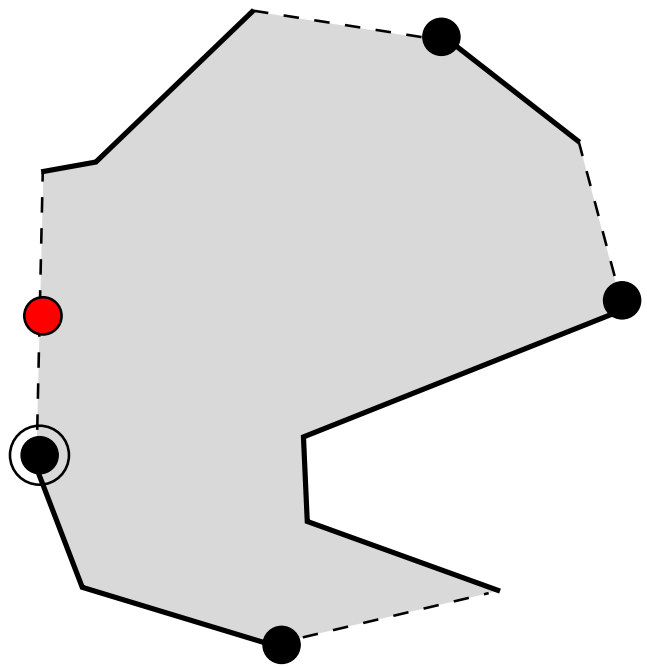
Depth-First Search

if(not all children visited)
    move to next child;
else
    move to parent;



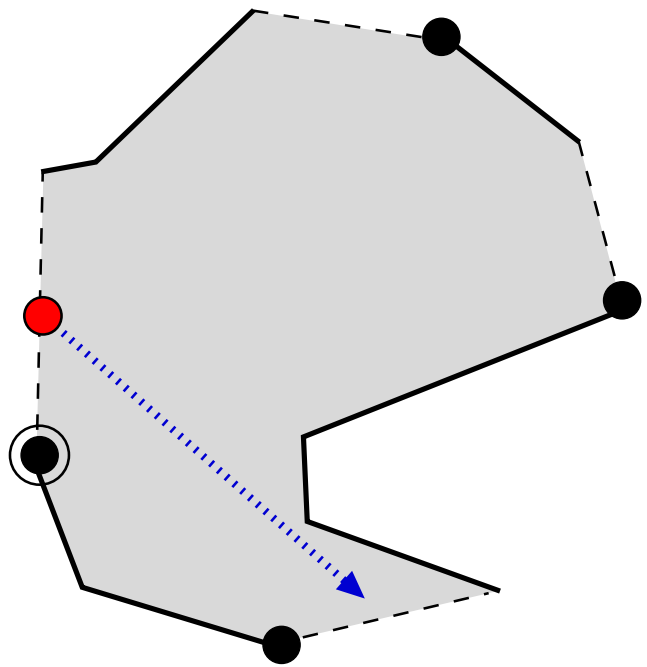
Depth-First Search

if(not all children visited)
    move to next child;
else
    move to parent;



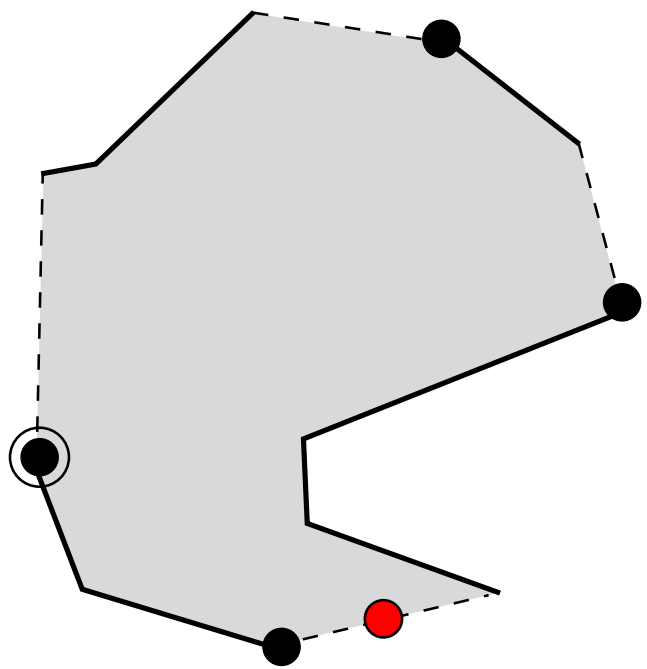
Depth-First Search

if(not all children visited)
    move to next child;
else
    move to parent;



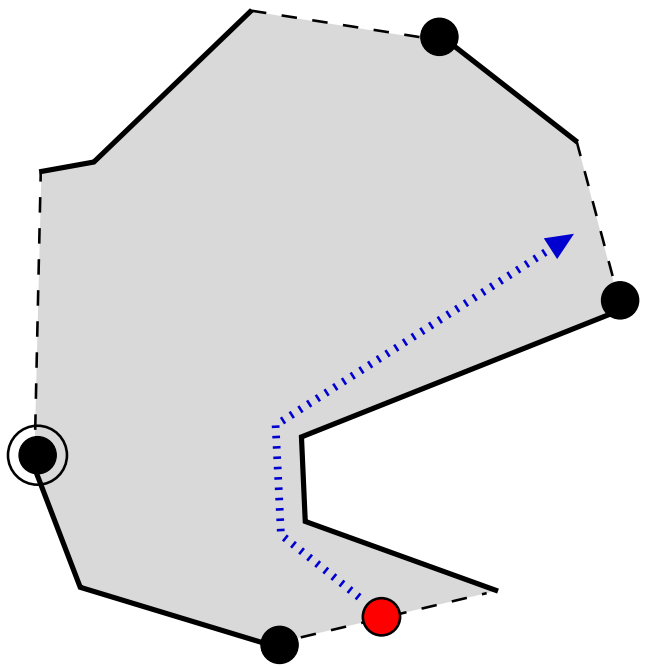
Depth-First Search

if(not all children visited)
    move to next child;
else
    move to parent;



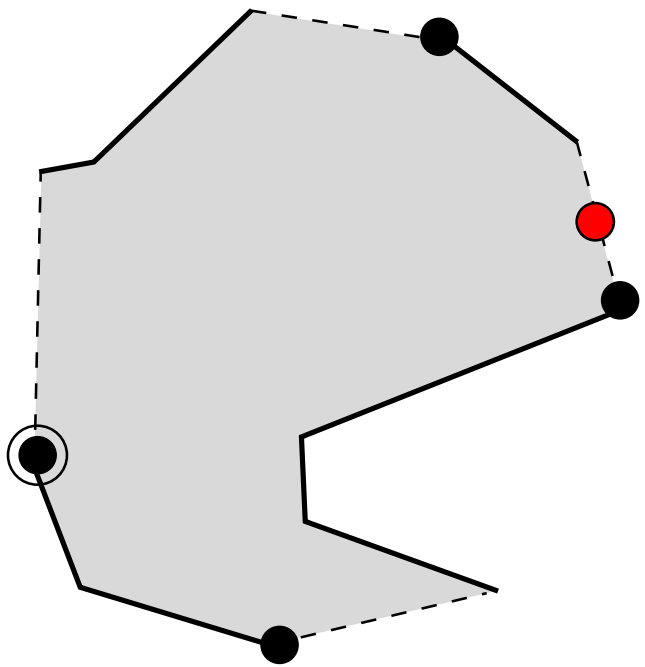
Depth-First Search

if(not all children visited)
    move to next child;
else
    move to parent;



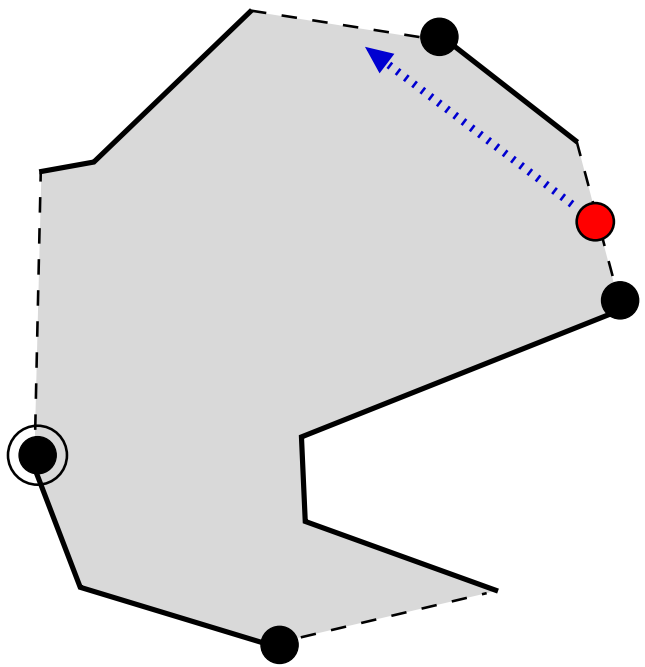
Depth-First Search

if(not all children visited)
    move to next child;
else
    move to parent;



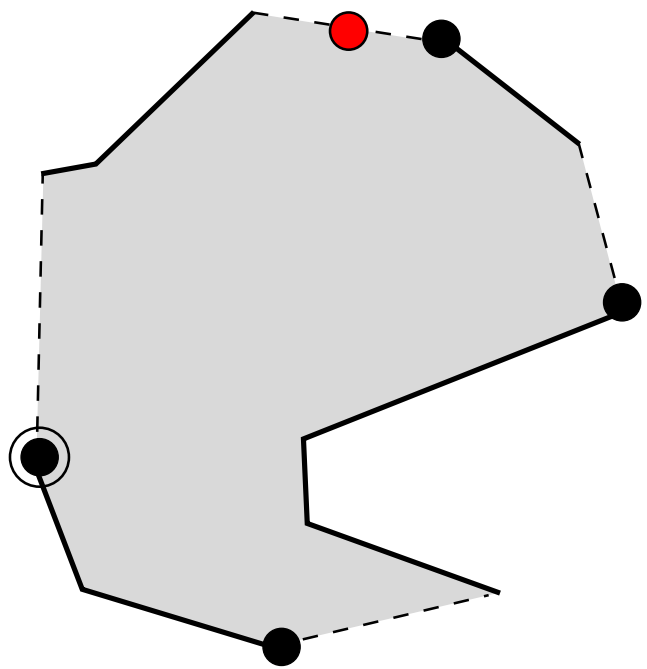
Depth-First Search

if(not all children visited)
    move to next child;
else
    move to parent;



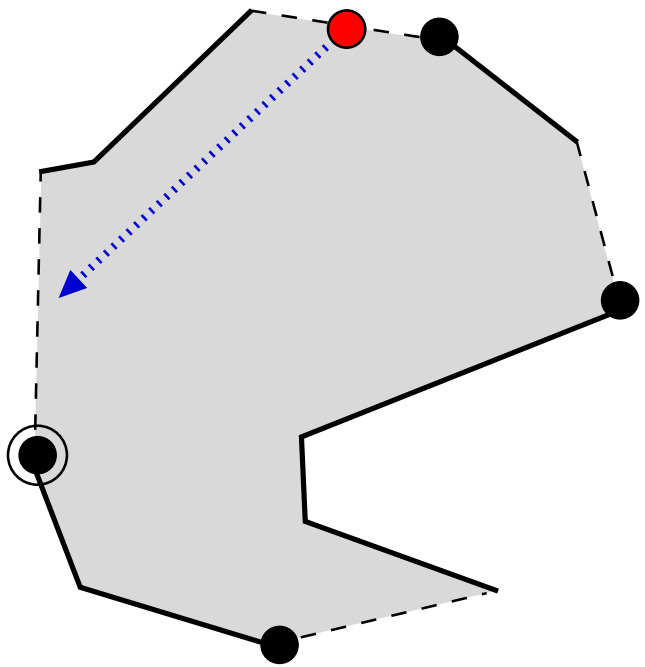
Depth-First Search

if(not all children visited)
    move to next child;
else
    move to parent;



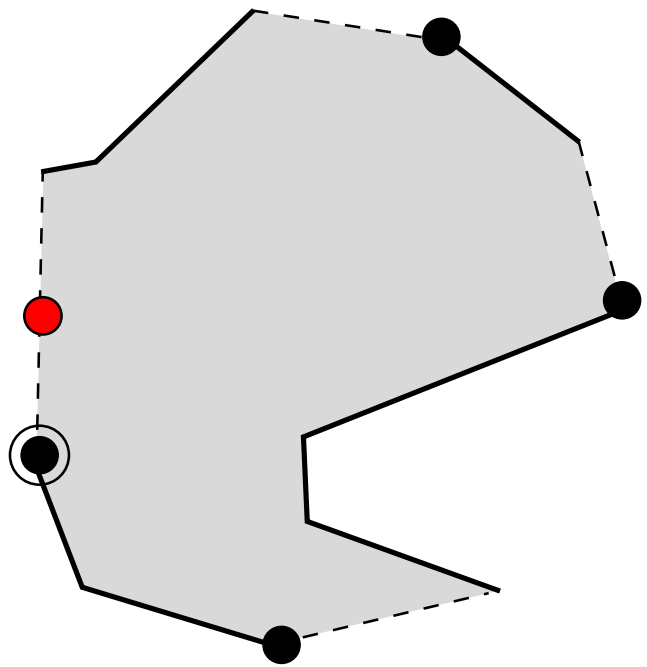
Depth-First Search

if(not all children visited)
    move to next child;
else
    move to parent;



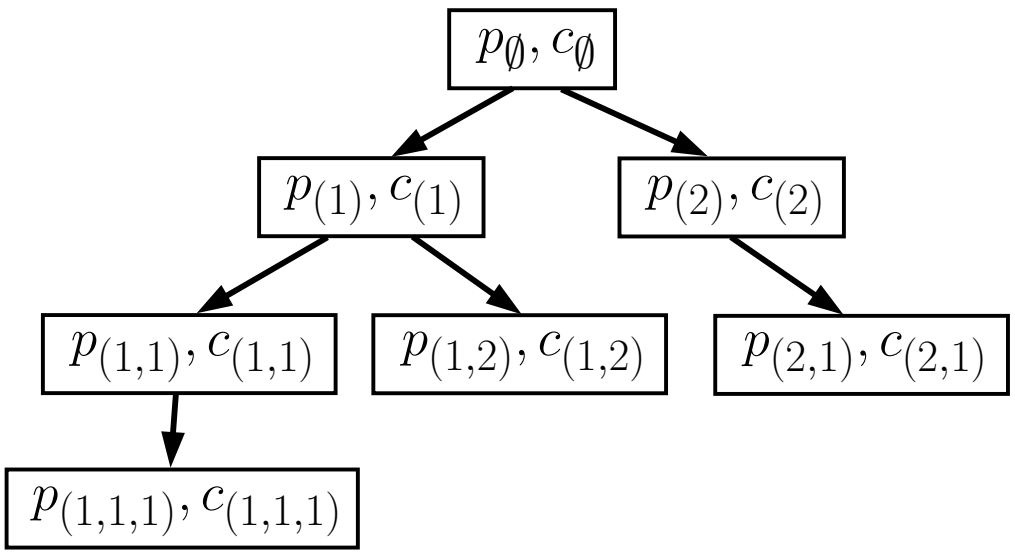
Depth-First Search

if(not all children visited)
    move to next child;
else
    move to parent;



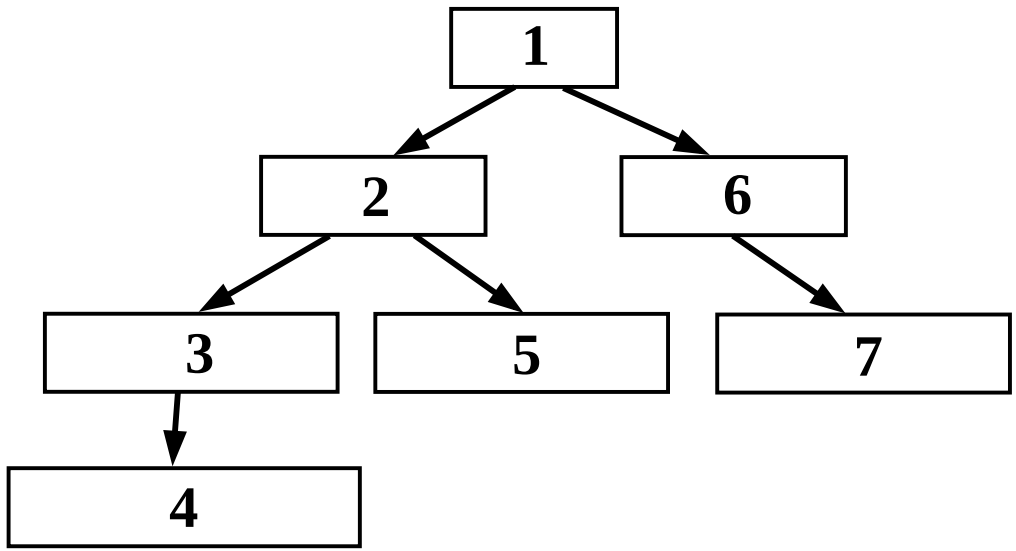
Depth-First Search

if(not all children visited)
    move to next child;
else
    move to parent;



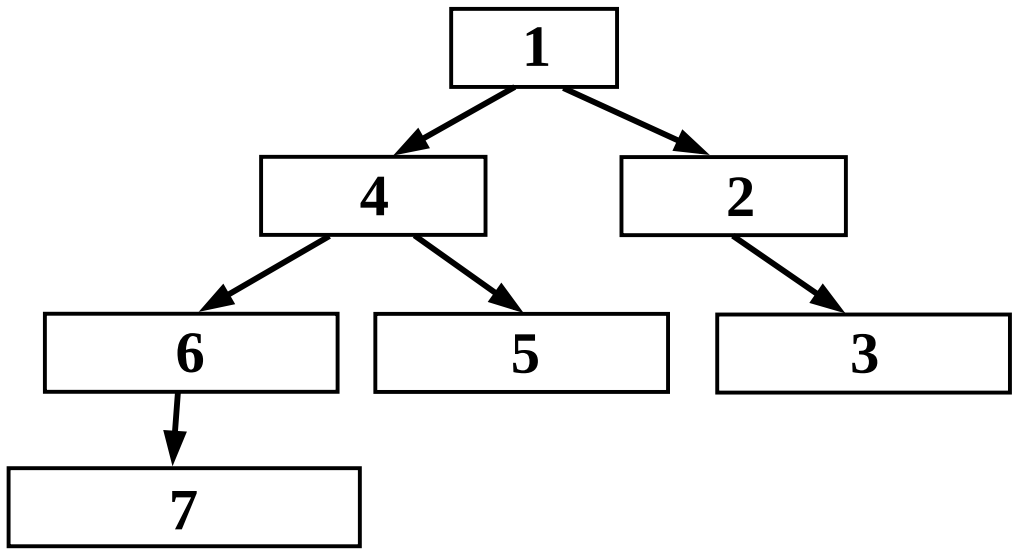
Depth-First Search

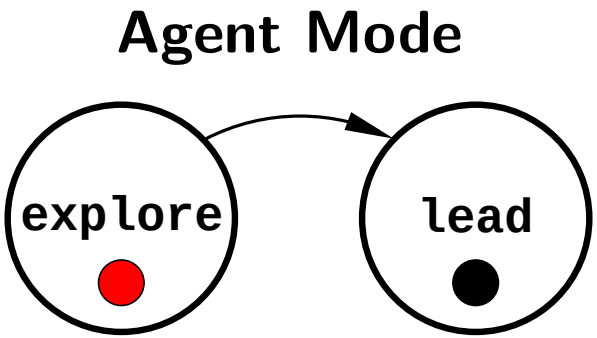
if(not all children visited)
    move to next child;
else
    move to parent;

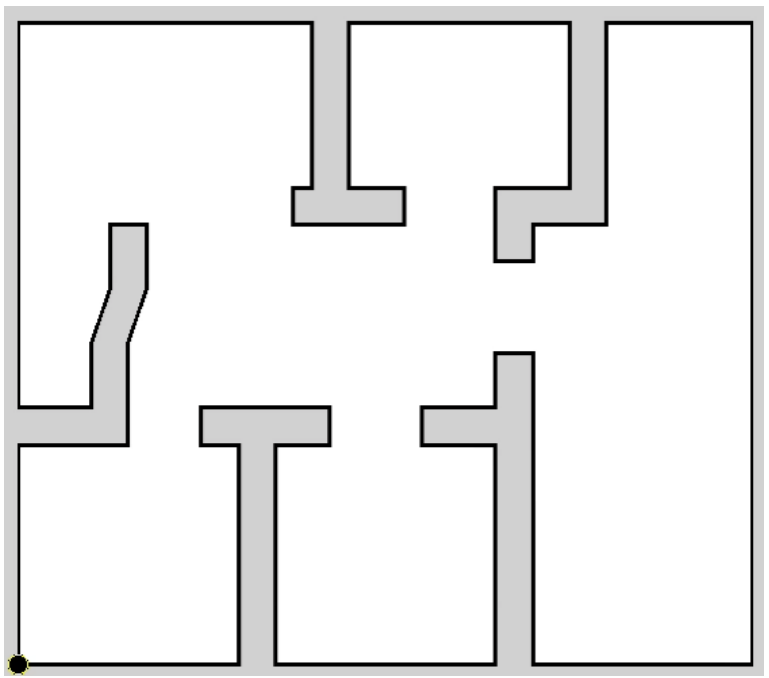


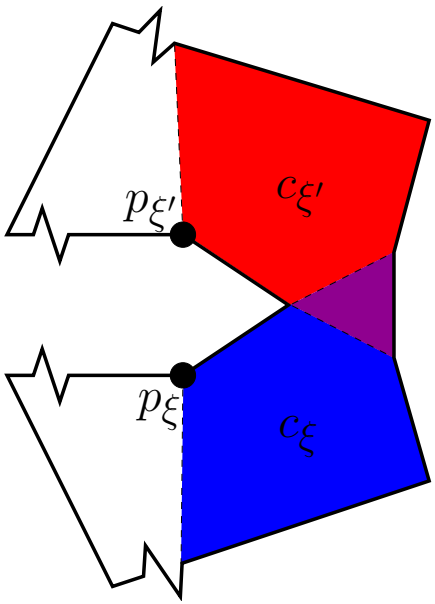
Depth-First Search

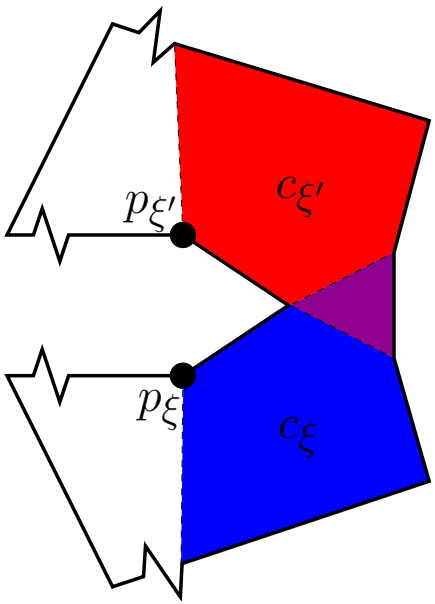
if(not all children visited)
    move to next child;
else
    move to parent;

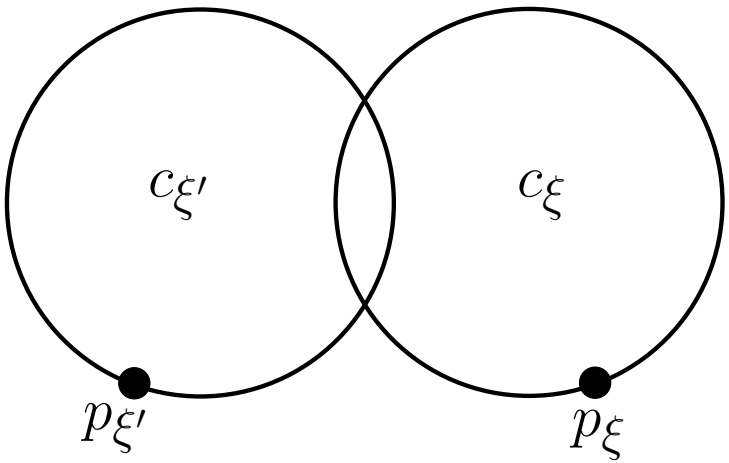



Agent Mode
explore
lead



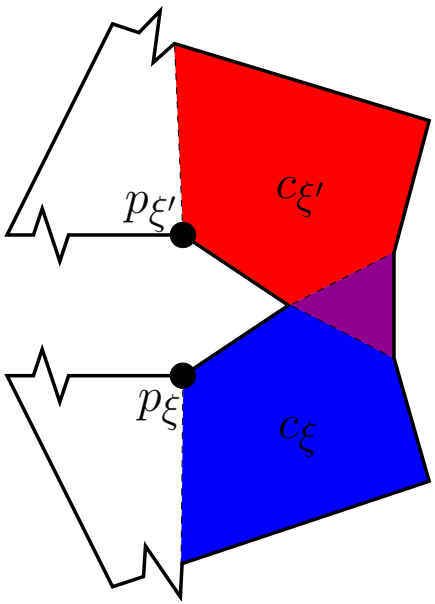

$p_{\xi'}$
$c_{\xi'}$
p_{ξ}
c_{ξ}

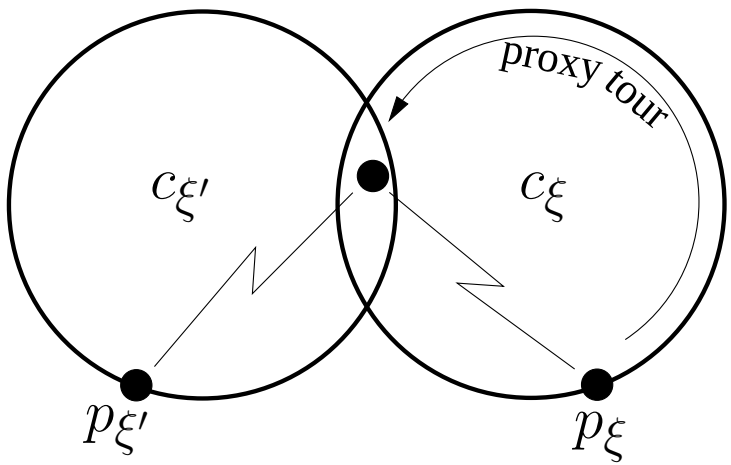

$p_{\xi'}$
$c_{\xi'}$
p_{ξ}
c_{ξ}


$c_{\xi'}$
c_{ξ}
$p_{\xi'}$
p_{ξ}

Distributed Deployment Algorithm

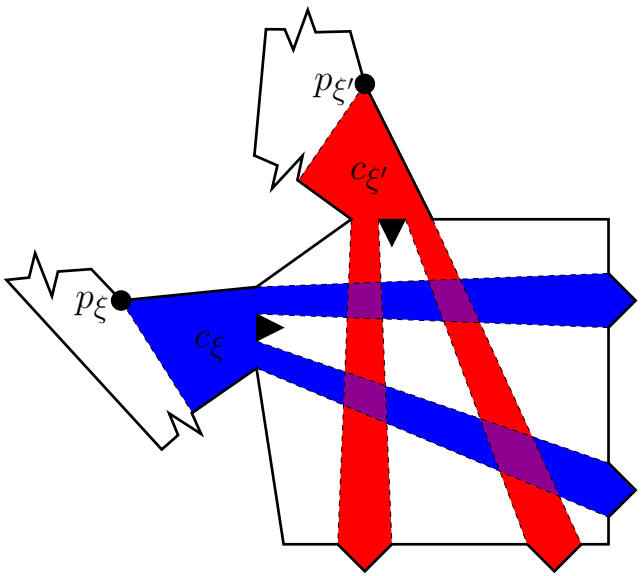
Branch Deconfliction via Proxy Agents

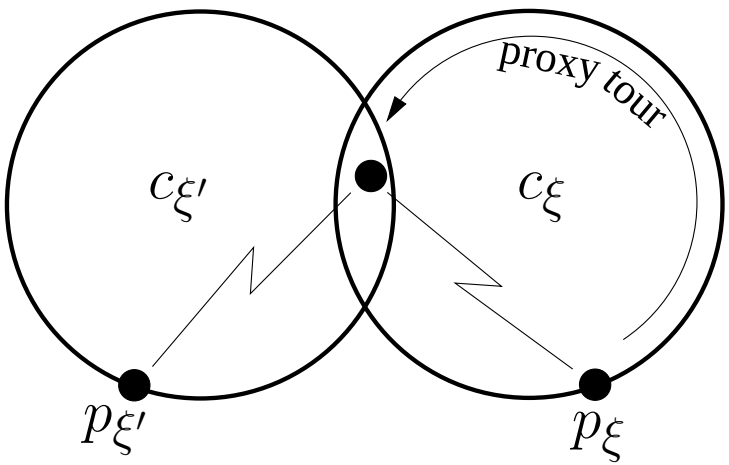


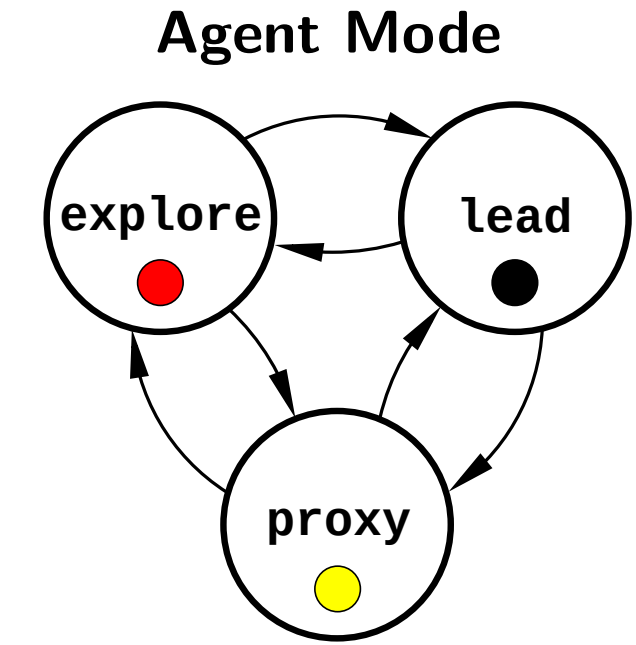


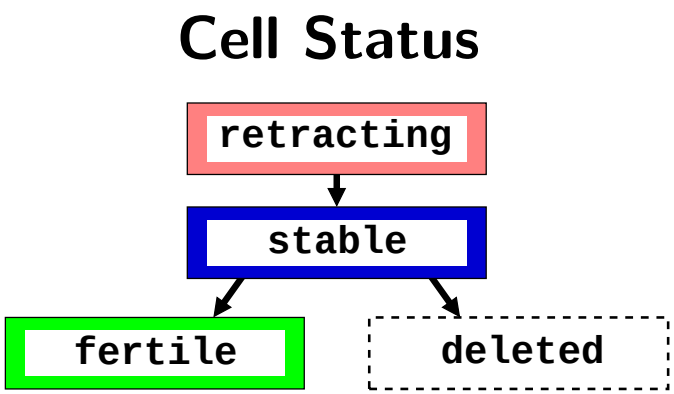
Distributed Deployment Algorithm

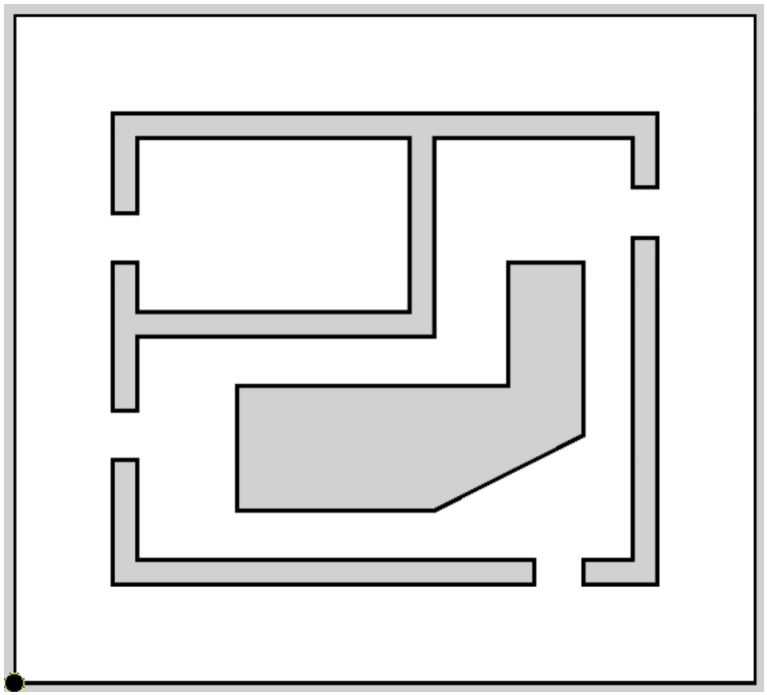
Branch Deconfliction via Proxy Agents






Agent Mode
explore
lead
proxy


Cell Status
retracting
stable
fertile
deleted



Convergence Theorem

If N agents operate according to the Distributed Deployment Algorithm, then

1. the agents' visibility graph remains connected,
2. there exists a finite time t^*, such that the partition tree remains unchanged for all $t > t^*$, and
3. if $N \geq \lfloor \frac{n+2h-1}{2} \rfloor$, then for all $t > t^*$ there is full coverage

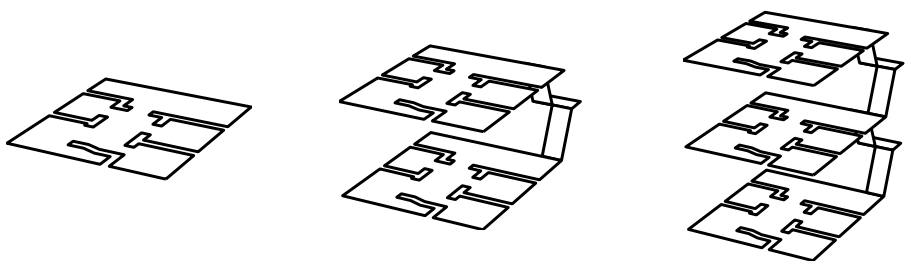
Theorem(Time to Converge)
Assuming uniformly bounded environment diameter, time until full coverage is $\mathcal{O}(n^2 + nh)$. If max cell perimeter uniformly bounded, then $\mathcal{O}(n + h)$

Lemma(Memory and Message Size)
Let k:= max number of vertices of any vertex-limited visibility polygon. Then an agent

1. requires $\mathcal{O}(Nk)$ bits of memory, and
2. sends messages of size $\mathcal{O}(k)$ bits

Extensions

- Robustness to agent failure
- Robustness to opening doors
- Multiple roots
- Combining with distributed assignment
- 2.5D environments



Part III
UAV Path Planning

Visual Reconnaissance Problem

Goal

Fly a camera-equipped UAV over a terrain such that a set of ground targets can be photographed in minimum time

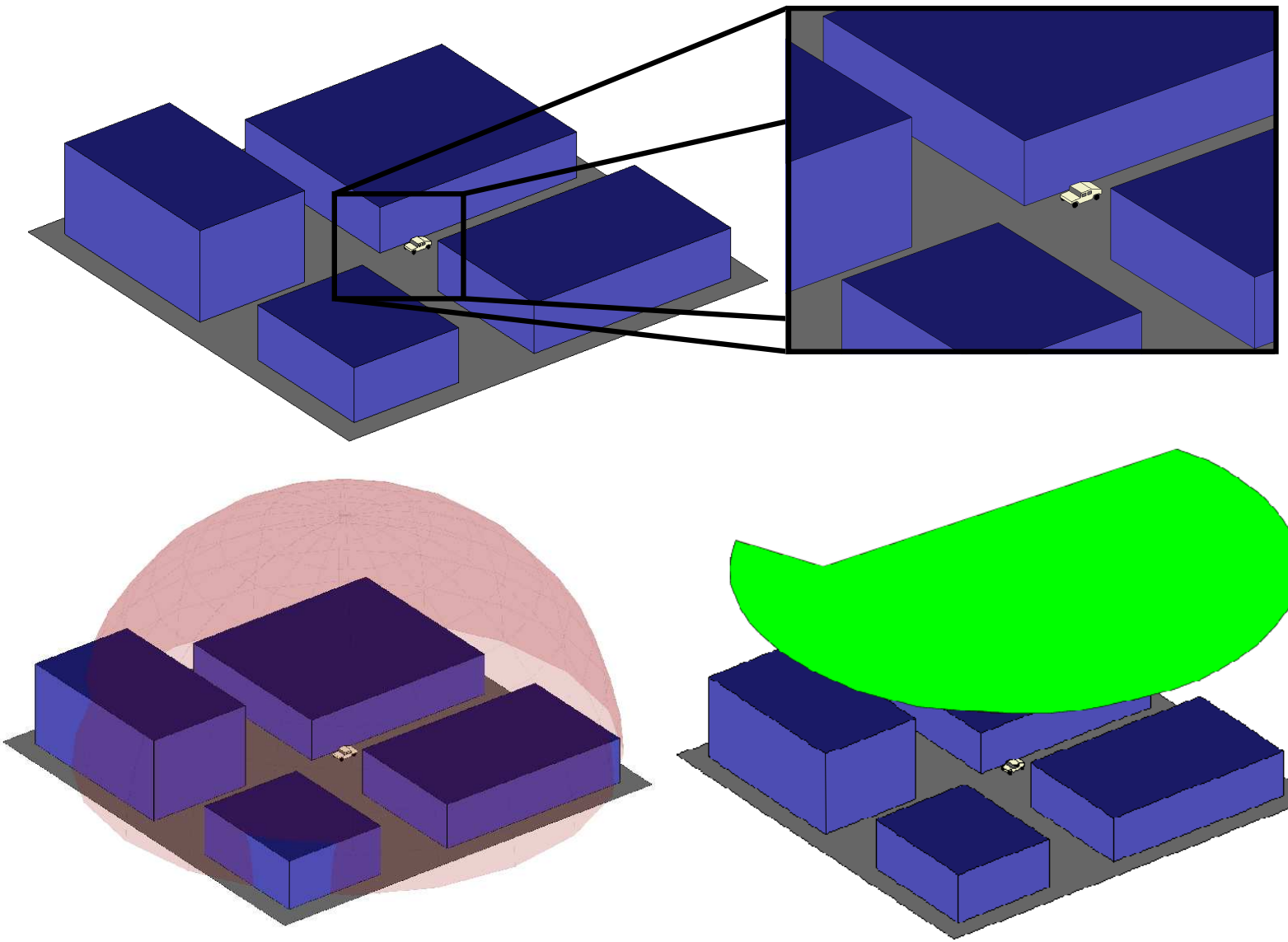
Assumptions

- fixed-wing UAV
- constant altitude, constant airspeed
- static terrain, static targets
- fast* gimbaled camera

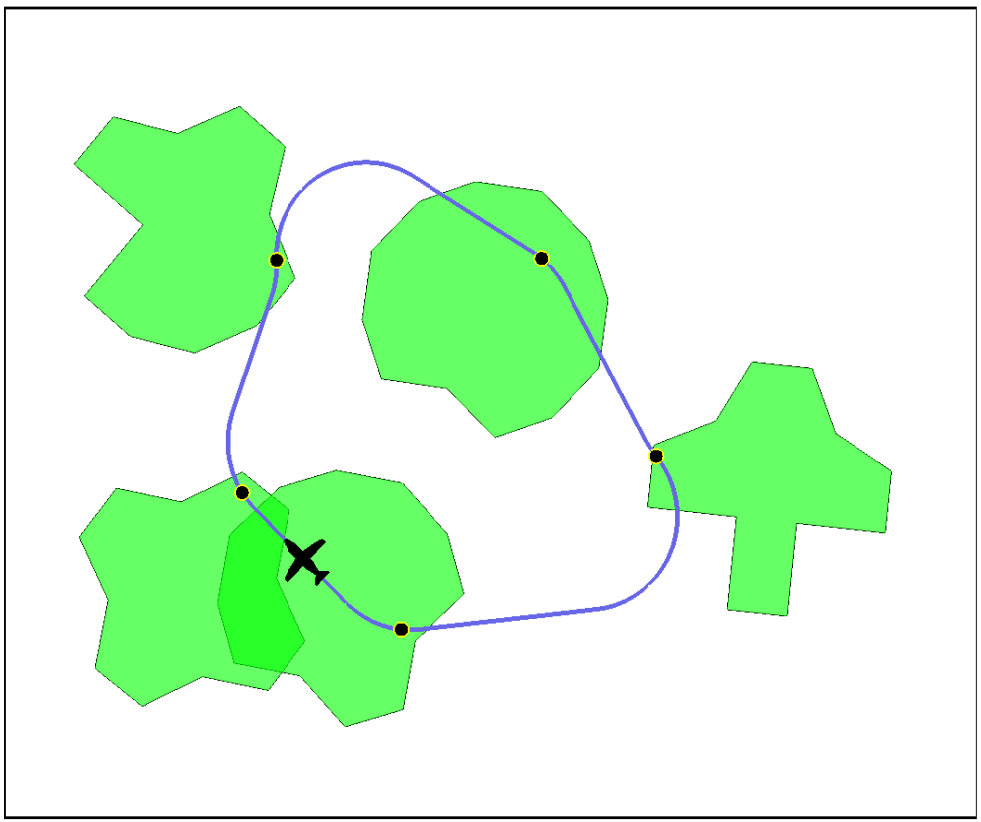
*Gimbal dynamics much faster than the UAV dynamics

Visibility of a Target in Terrain

Intersecting **Range**, **Altitude**, and **Occlusion** Constraints

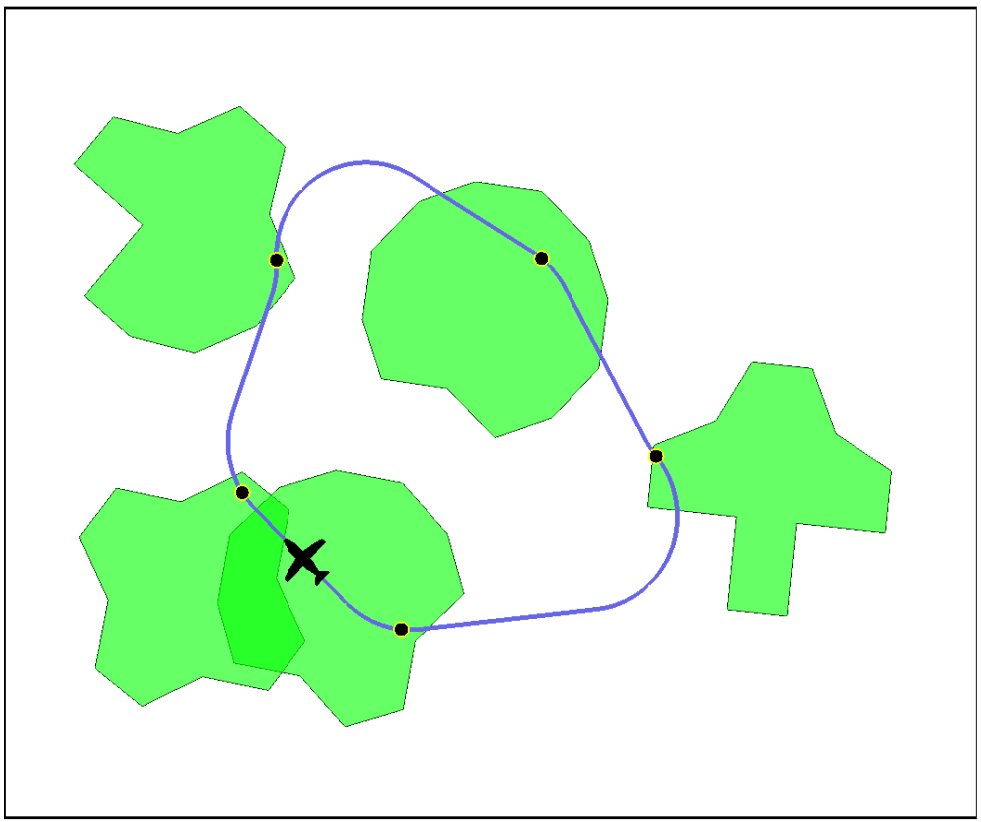


Polygon-Visiting Dubins TSP (PVDTSP)



Polygon-Visiting Dubins TSP (PVDTSP)

$$\left(\begin{bmatrix} x_1 \\ y_1 \\ \psi_1 \end{bmatrix}, \begin{bmatrix} x_2 \\ y_2 \\ \psi_2 \end{bmatrix}, \begin{bmatrix} x_3 \\ y_3 \\ \psi_3 \end{bmatrix}, \begin{bmatrix} x_4 \\ y_4 \\ \psi_4 \end{bmatrix}, \begin{bmatrix} x_5 \\ y_5 \\ \psi_5 \end{bmatrix} \right)$$



Vehicle Dynamics

$$\mathbf{x} = (x, y, \psi) \in X = \mathbb{R}^2 \times \mathbb{S} = \mathrm{SE}(2)$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} V_{\mathrm{a}} \sin(\psi) \\ V_{\mathrm{a}} \cos(\psi) \\ u \end{bmatrix}, \quad u \leq u_{\max} \quad (\text{max turn rate})$$

Optimization Problem

$$\begin{aligned} \text{Minimize :} \quad & C(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \textstyle\sum_{i=1}^{n-1} d(\mathbf{x}_i, \mathbf{x}_{i+1}) + d(\mathbf{x}_n, \mathbf{x}_1) \\ \text{Subject To :} \quad & \text{for each } i \text{ there exists } j \text{ s.t. } \mathbf{x}_j \in \mathcal{V}(\mathcal{T}_i) \end{aligned}$$

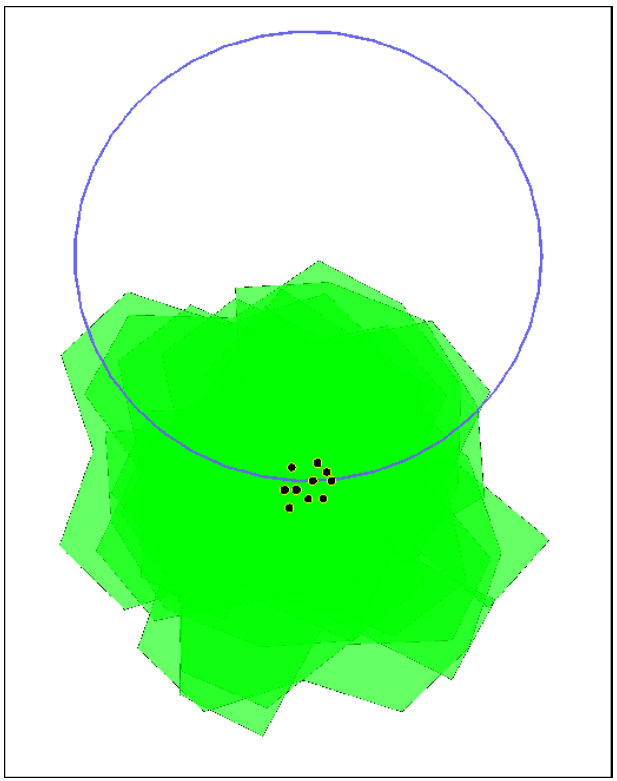
target visibility set $\mathcal{V}(\mathcal{T}_i) \subset X$

state-to-state cost from BVP solver

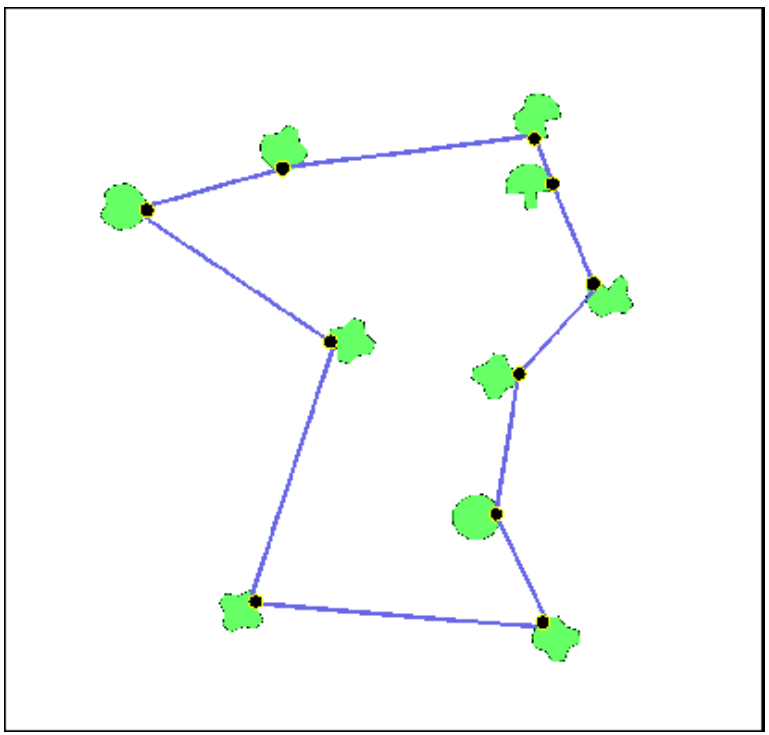
Theorem

Solving DTSP instead of PVDTSP $\Rightarrow$ $\Omega(n)$ penalty in worst case

Dense Limit



Sparse Limit

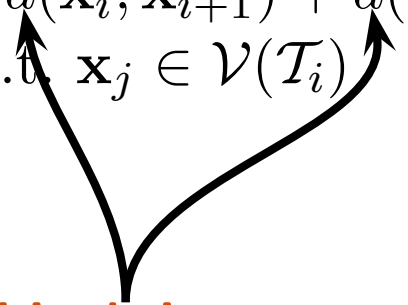


Algorithm Design Requirements

- **Provable convergence**
- **Speed** suitable for online purposes
- **Ability to trade off computation time** for solution quality
- **Extensibility** for
  - Wind
  - Airspace constraints
  - Any vehicle dynamics

Optimization Problem

$$\begin{aligned} \text{Minimize}: \quad & C(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \sum_{i=1}^{n-1} d(\mathbf{x}_i, \mathbf{x}_{i+1}) + d(\mathbf{x}_n, \mathbf{x}_1) \\ \text{Subject To}: \quad & \text{for each } i \text{ there exists } j \text{ s.t. } \mathbf{x}_j \in \mathcal{V}(\mathcal{T}_i) \end{aligned}$$



treating state-to-state distance function as **black box**

$\Rightarrow$ can use any BVP solver which handles

- Wind
- Airspace constraints
- Any vehicle dynamics

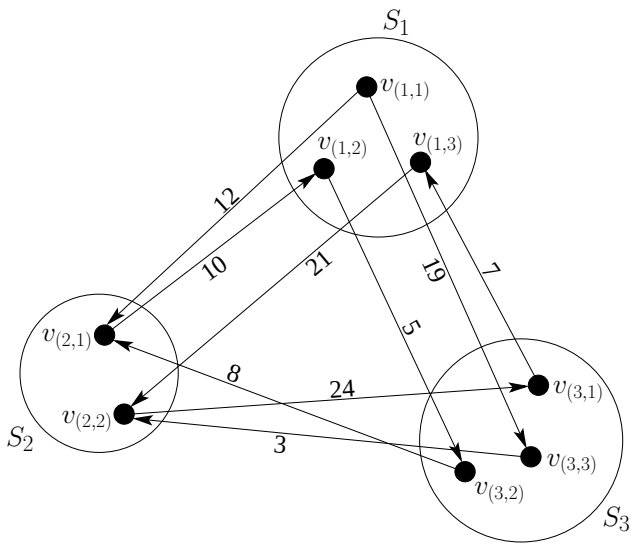
ATSP (Asymmetric TSP)

In a weighted directed graph, find a minimum weight closed tour which visits each vertex exactly once

- NP-Hard, but state-of-the-art heuristics are effectively exact and have $\mathcal{O}(n^{2.2})$ average case runtime [Helsgaun, 2000]

FOTSP (Finite One-in-a-set TSP)

In a weighted directed graph with vertices partitioned into clusters, find a minimum weight closed tour which visits at least one vertex in each cluster



Relevant Variations of the TSP

ATSP (Asymmetric TSP)

In a weighted directed graph, find a minimum weight closed tour which visits each vertex exactly once

- NP-Hard, but state-of-the-art heuristics are effectively exact and have $\mathcal{O}(n^{2.2})$ average case runtime [Helsgaun, 2000]

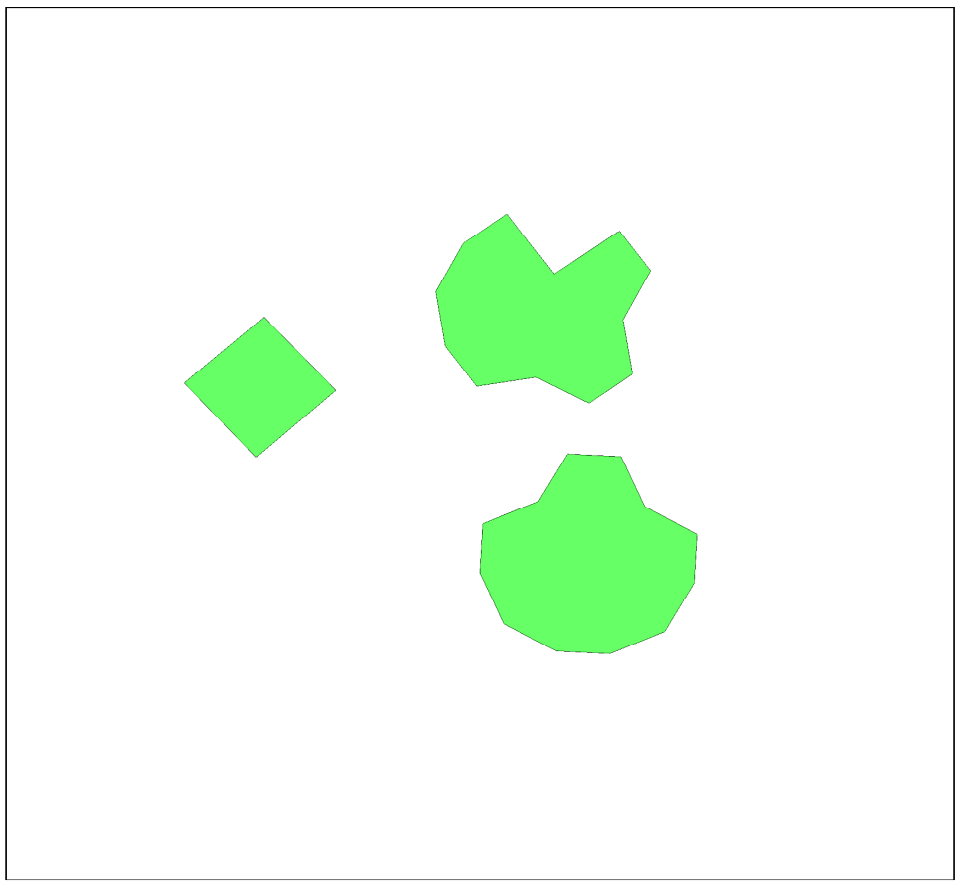
FOTSP (Finite One-in-a-set TSP)

In a weighted directed graph with vertices partitioned into clusters, find a minimum weight closed tour which visits at least one vertex in each cluster

- Can be solved exactly via Noon-Bean transformation [Noon and Bean, 1991]

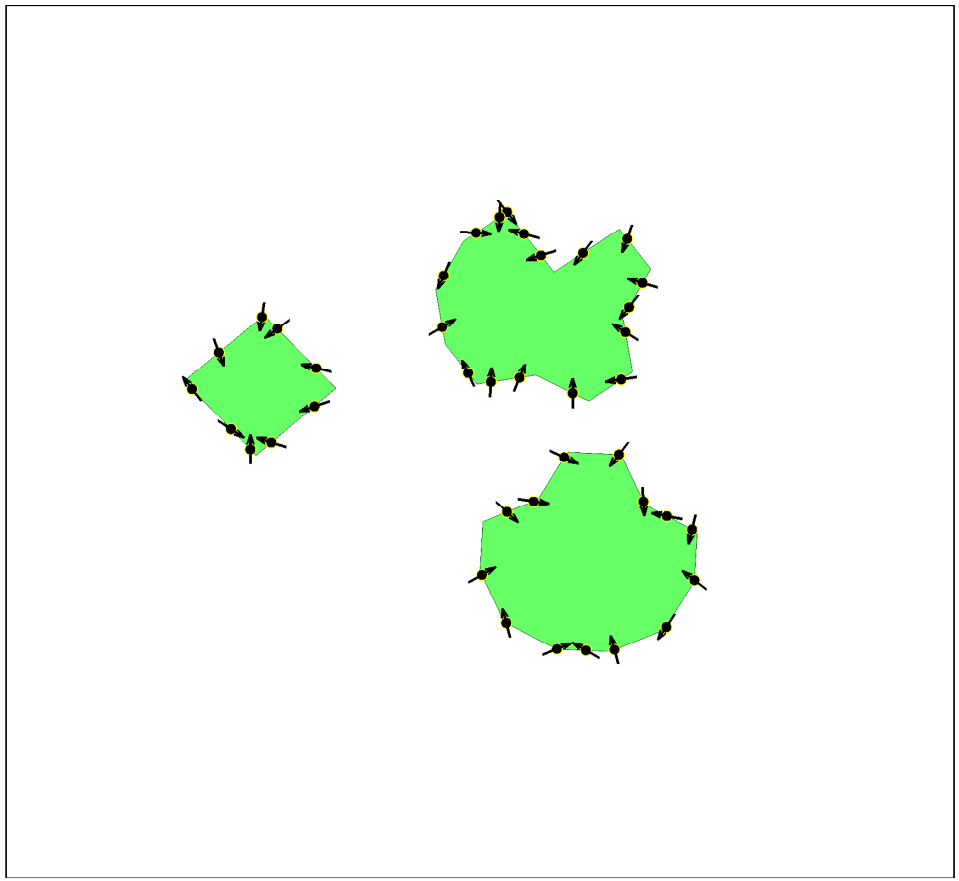
PVDTSP Roadmap Construction

(FOTSP instance)



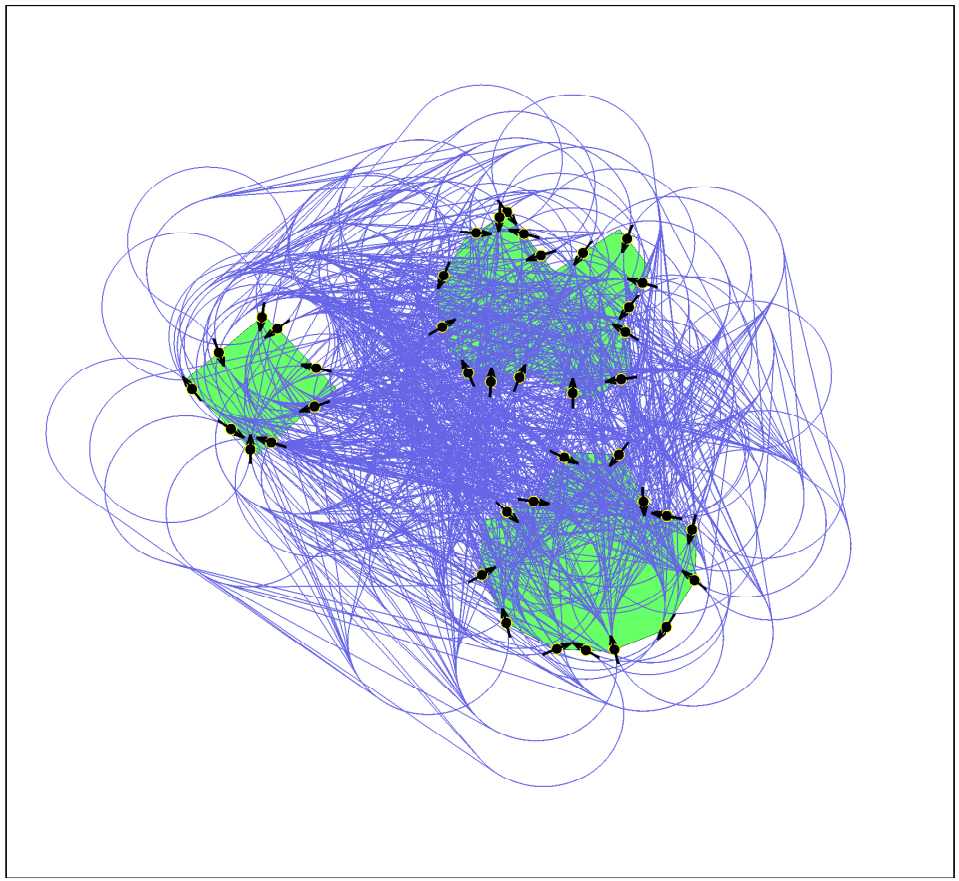
PVDTSP Roadmap Construction

(FOTSP instance)

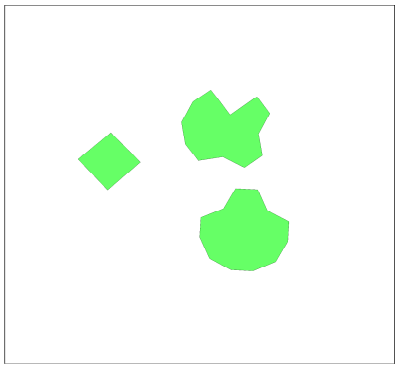


PVDTSP Roadmap Construction

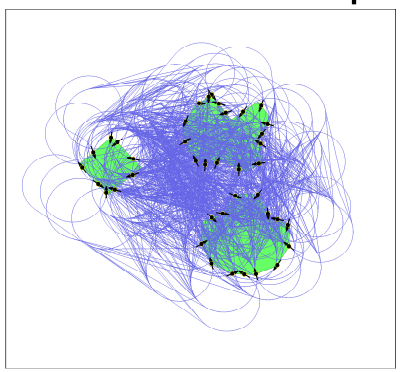
(FOTSP instance)



Input
PVDTSP Instance

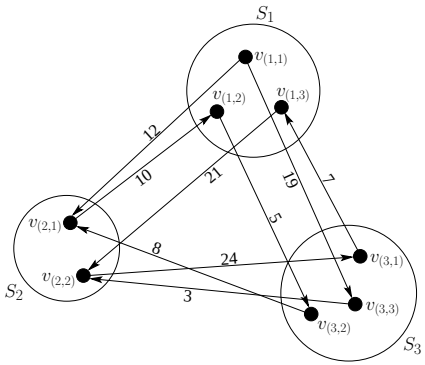


Build Roadmap

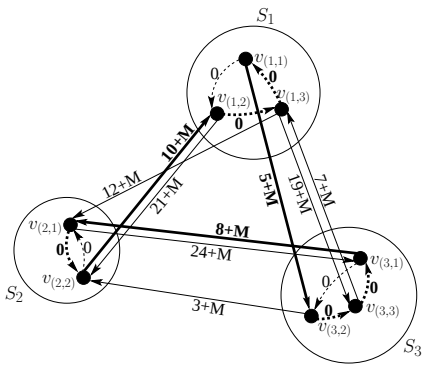


=

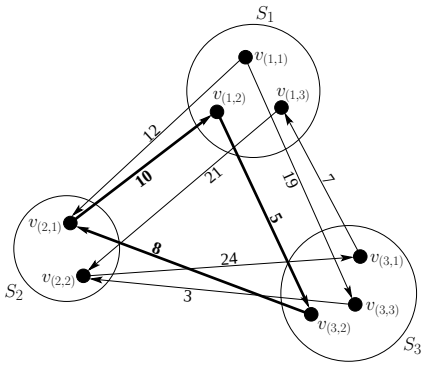
FOTSP Instance



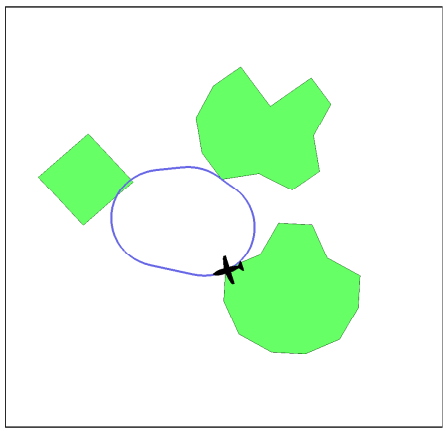
Build + Solve
ATSP Instance
(Noon-Bean Transf.)



Extract
FOTSP Solution



Return
Final Path



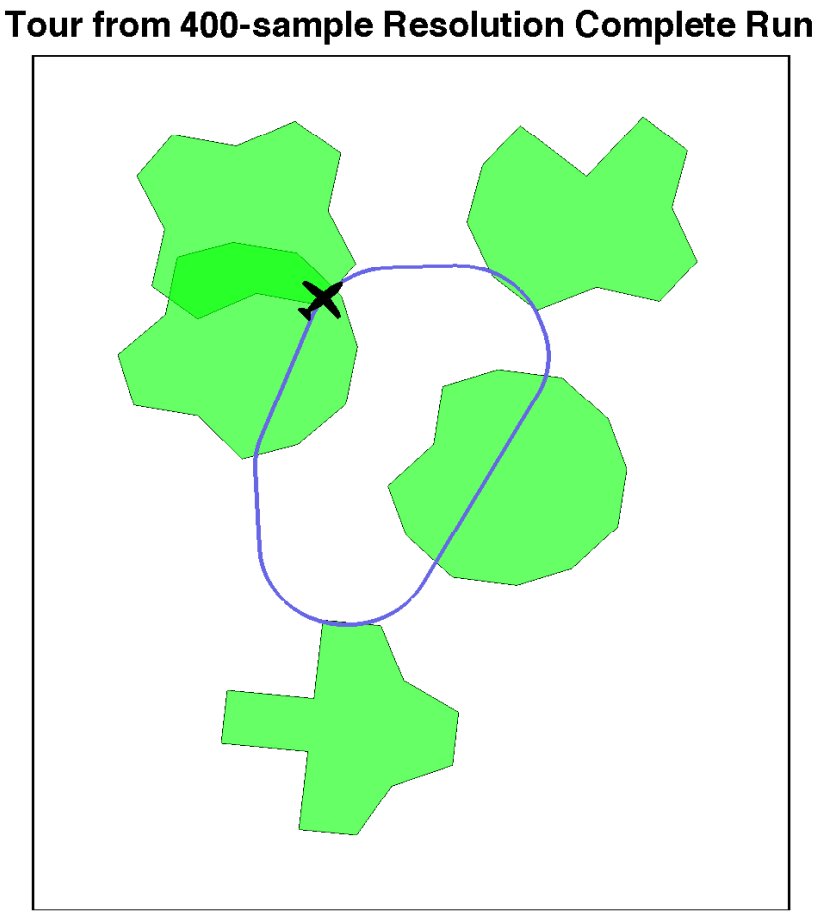
Theorem (Resolution Completeness)
Let $\{\tau_i\}_{i=1}^{\infty}$ be the sequence of tours computed by the sampling-based roadmap method when applied to the sequence of roadmaps $\{\mathcal{R}_i\}_{i=1}^{\infty}$, respectively. Then $\{C(\tau_i)\}_{i=1}^{\infty}$ is nonincreasing and

$$\lim_{i\to\infty} C(\tau_i) \leq \inf_{\tau\in\mathcal{D}^{\circ}_{\text{feas}}} C(\tau).$$

- empirically determined average case time complexity* $\mathcal{O}(n_{\text{samples}}^{2.2})$

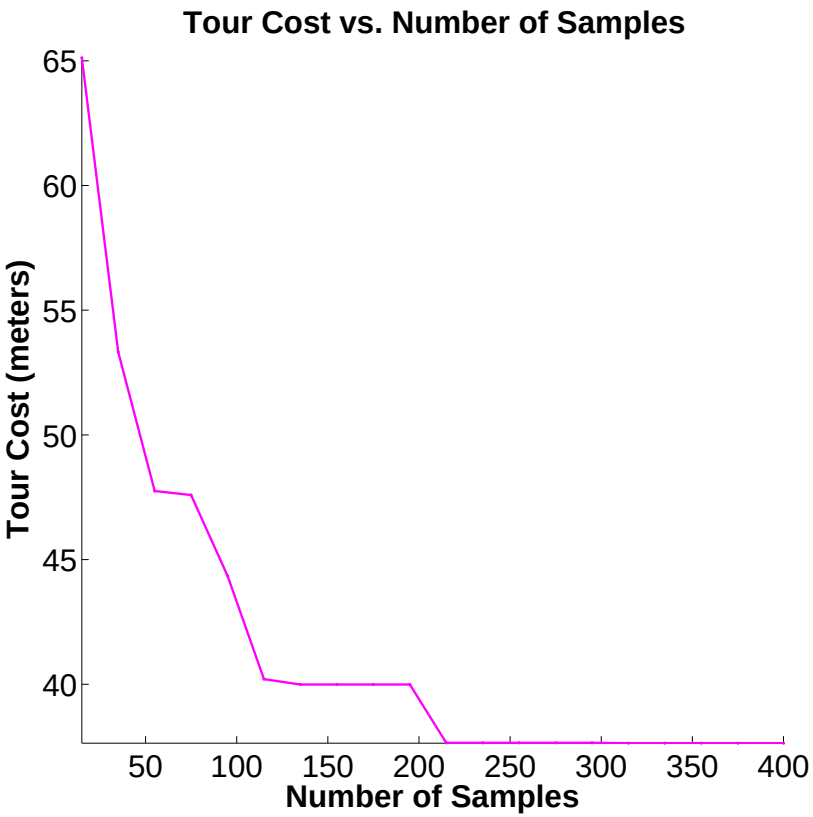
Numerical Study

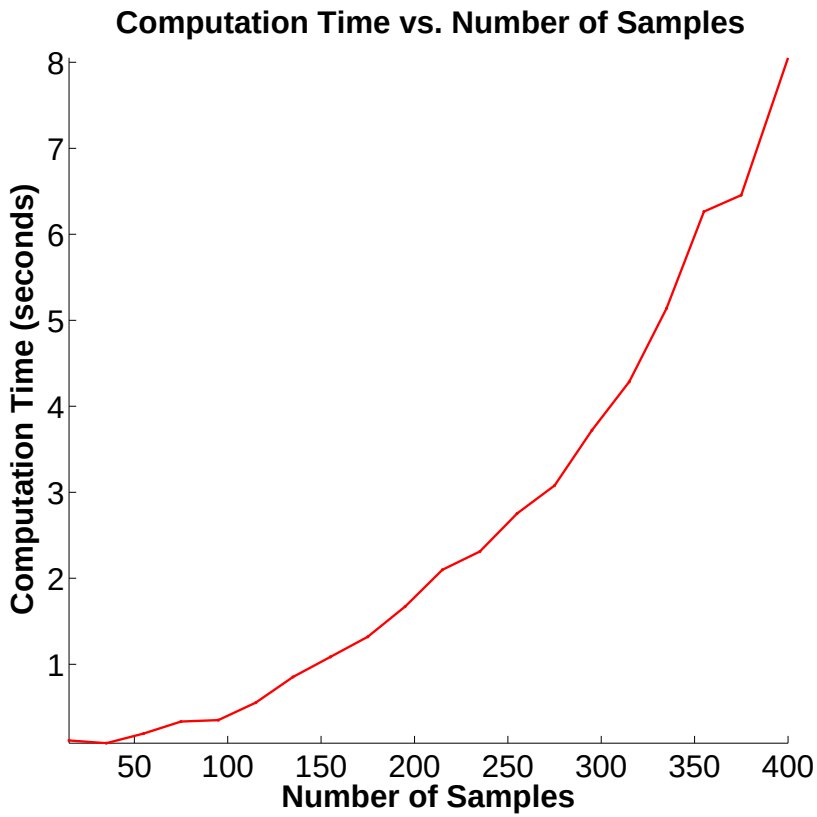
5 Targets, ~ 10 sec. in C++ on 2.33 GHz i686



Numerical Study

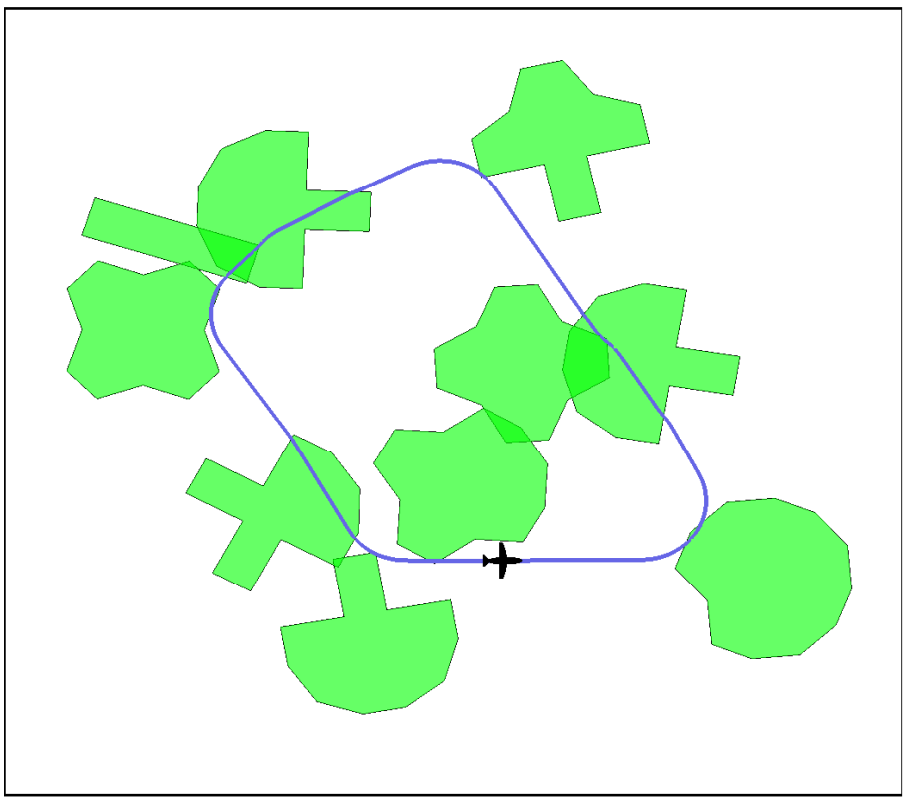
5 Targets, ~ 10 sec. in C++ on 2.33 GHz i686





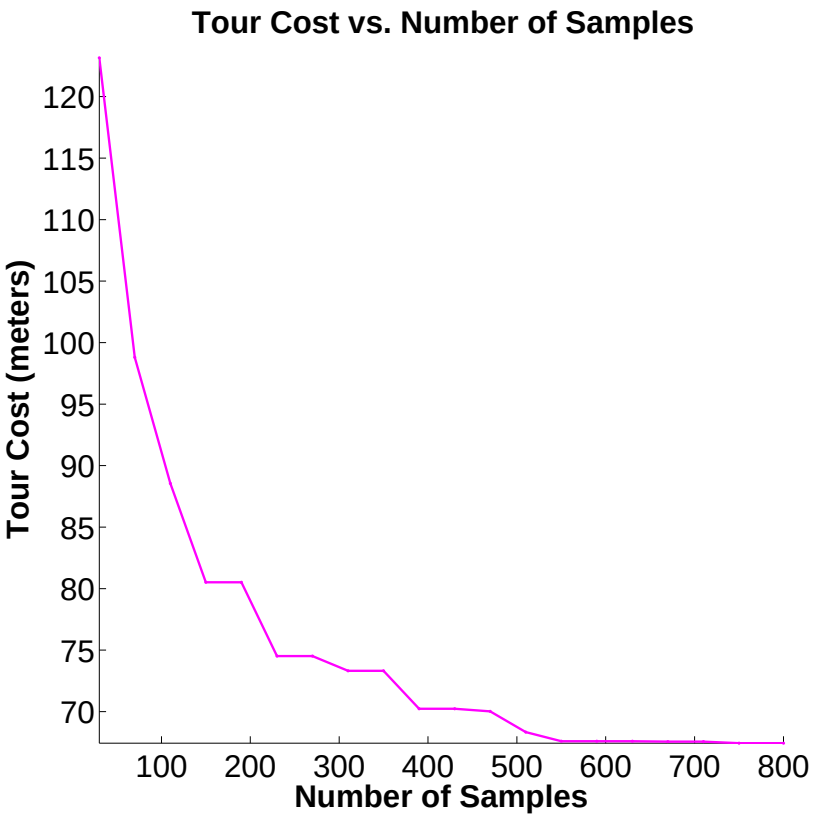
Numerical Study

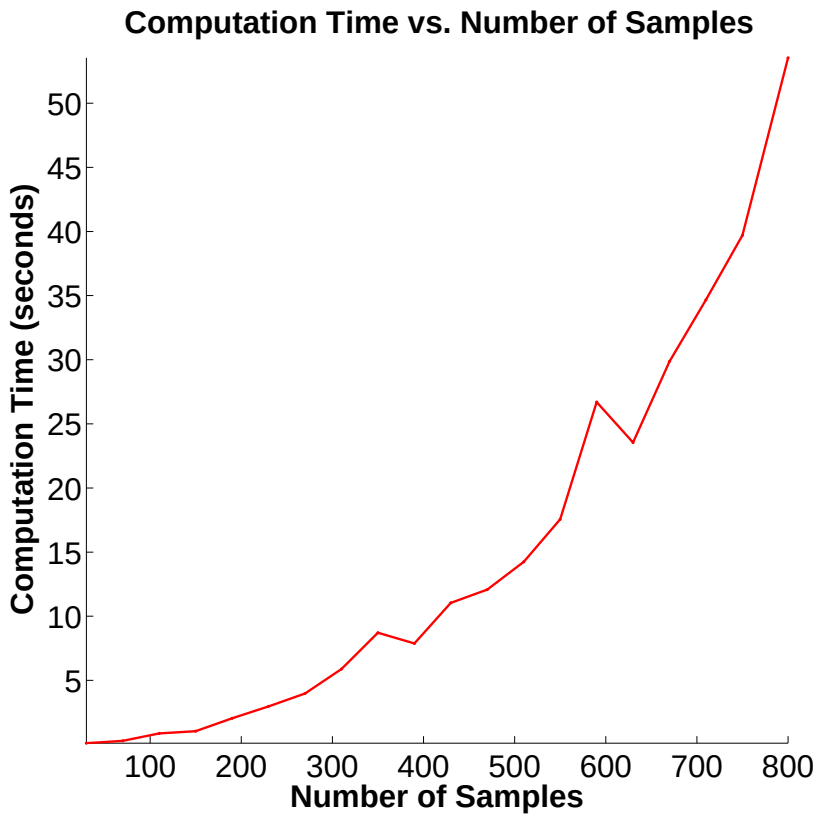
10 Targets, ~ 50 sec. in C++ on 2.33 GHz i686



Numerical Study

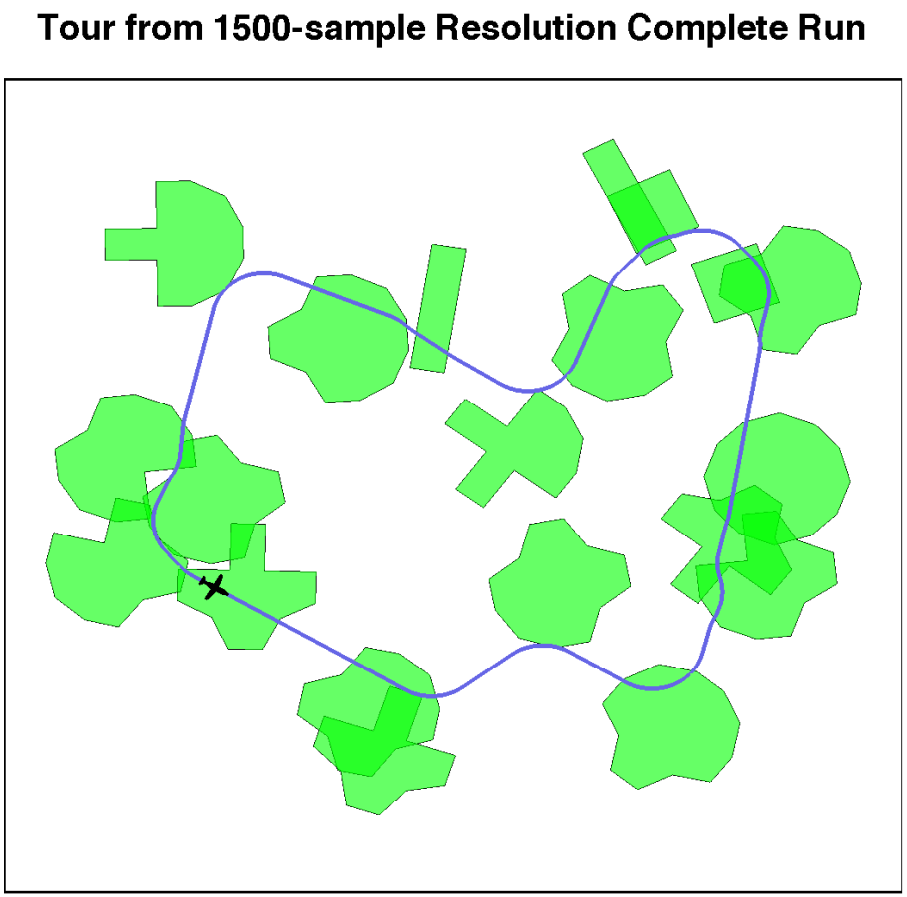
10 Targets, ~ 50 sec. in C++ on 2.33 GHz i686





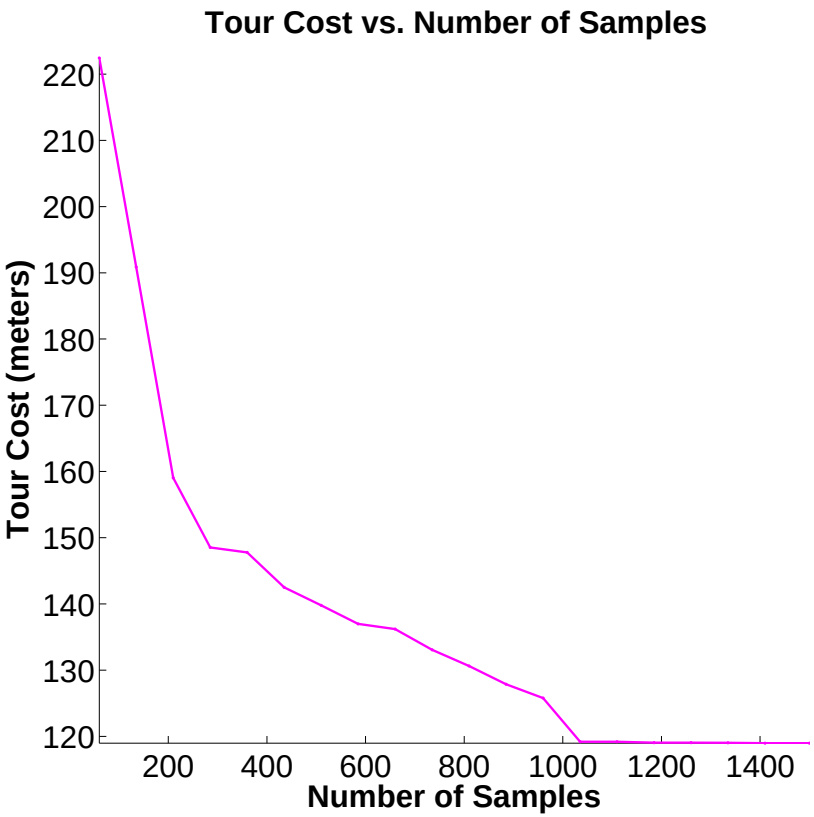
Numerical Study

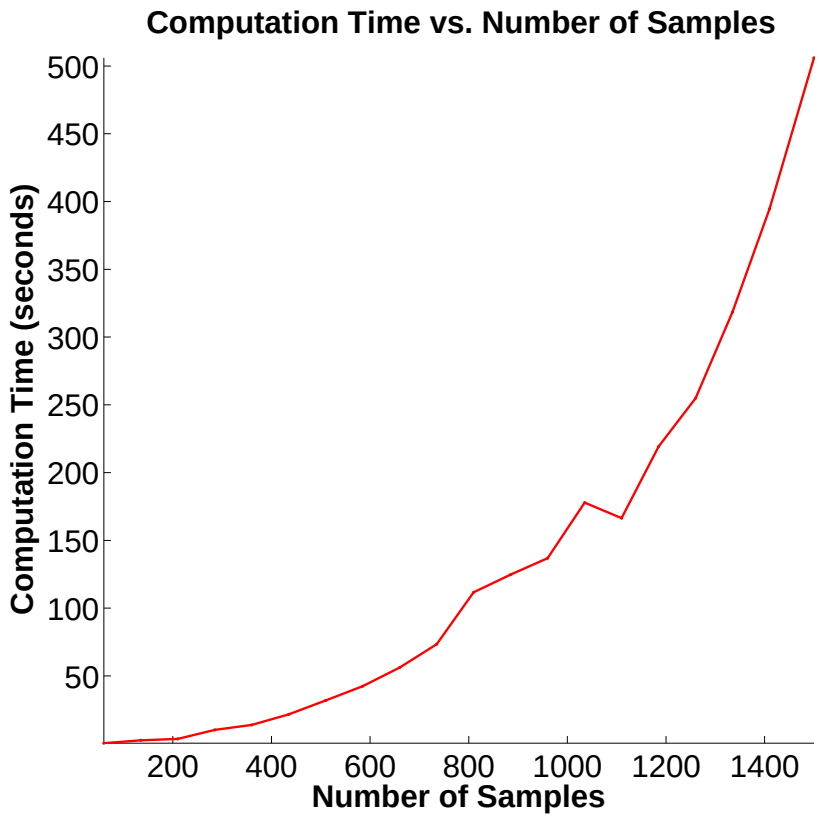
20 Targets, ~ 500 sec. in C++ on 2.33 GHz i686



Numerical Study

20 Targets, ~ 500 sec. in C++ on 2.33 GHz i686





Statistics from Computed Examples

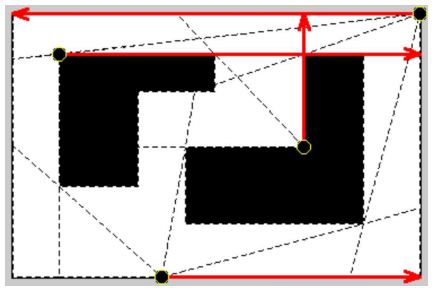
C++, 2.33 GHz i686

Targets	Samples	**Convergence Time**	Tour Length
5	400	**8.05 s**	37.64 m
10	800	**53.54 s**	67.44 m
20	1500	**506.07 s**	118.99 m

Conclusion

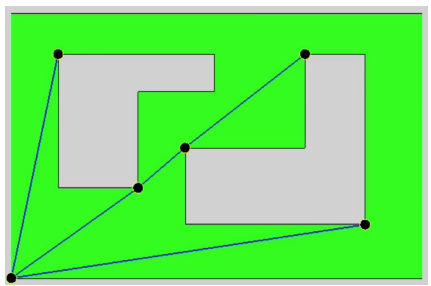
Part I: Searchlight/Camera Scheduling

- Combinatorial roadmap method
- Linear time placement + scheduling



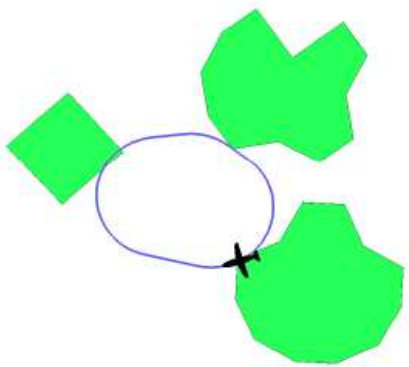
Part II: Multi-Agent Deployment

- Distributed algorithm for environments with holes



Part III: UAV Path Planning

- PVDTSP Formulation
- Sampling-based roadmap method



Visibility-Based Pursuit-Evasion

- Searchlight Scheduling time complexity
- Variations on Searchlight Scheduling
- Minimum time coordinated search with mobile guards

Visibility Coverage

- Practical implementation of deployment
- 3D and dynamic environments
- Optimizing different performance measures

UAV Reconnaissance Path Planning

- Study rate of convergence and parameter sensitivity
- Constant factor approximations
- Multiple vehicles

Visibility-Based Pursuit-Evasion

- Searchlight Scheduling time complexity
- Variations on Searchlight Scheduling
- Minimum time coordinated search with mobile guards

Visibility Coverage

- Practical implementation of deployment
- 3D and dynamic environments
- Optimizing different performance measures

UAV Reconnaissance Path Planning

- Study rate of convergence and parameter sensitivity
- Constant factor approximations
- Multiple vehicles

Visibility-Based Pursuit-Evasion

- Searchlight Scheduling time complexity
- Variations on Searchlight Scheduling
- Minimum time coordinated search with mobile guards

Visibility Coverage

- Practical implementation of deployment
- 3D and dynamic environments
- Optimizing different performance measures

UAV Reconnaissance Path Planning

- Study rate of convergence and parameter sensitivity
- Constant factor approximations
- Multiple vehicles