Path Planning for Point Robots

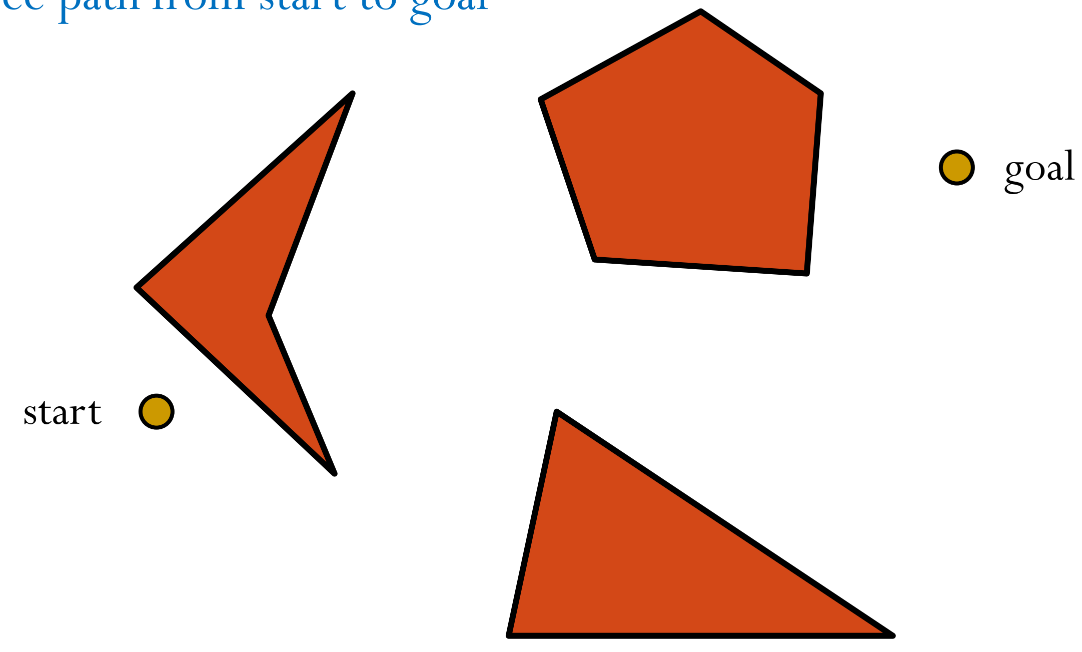
Jane Li
Assistant Professor
Mechanical Engineering & Robotics Engineering
http://users.wpi.edu/~zli11

- Presentation Topic Preferences due today!
  - Make sure you have voted on piazza

Path Planning for Point Robots

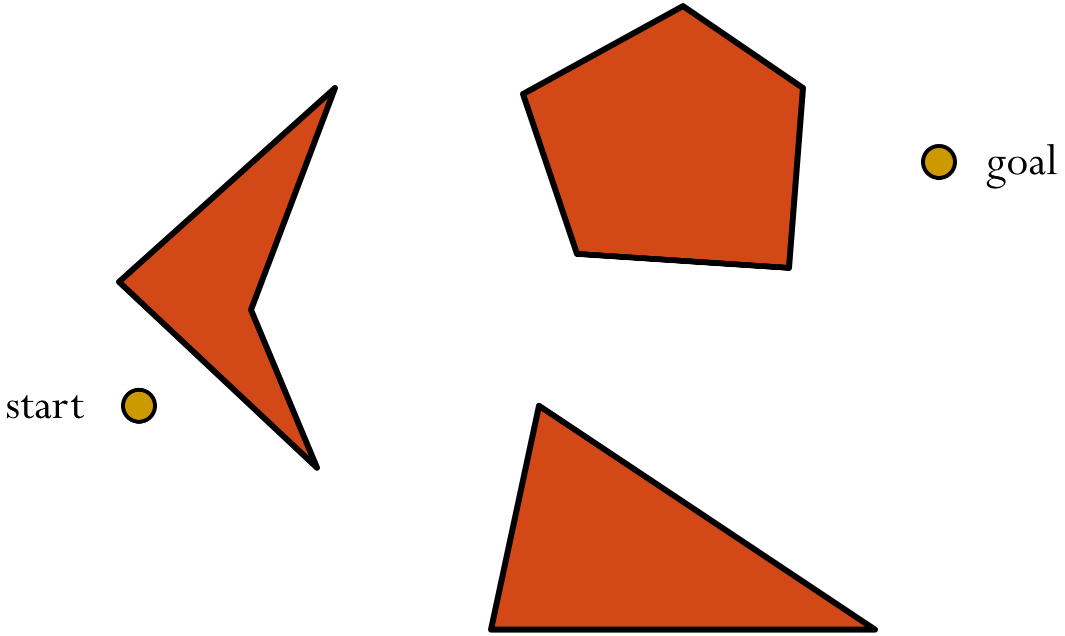
Path Planning for Point Robots

- Problem setup
  - **Point robot**
  - **2D** environment, with **polygonal** obstacles
- Objective
  - Find a collision-free path from start to goal



Method

- Roadmaps
  - Visibility graph
  - Voronoi graph
- Cell decomposition
- Potential field



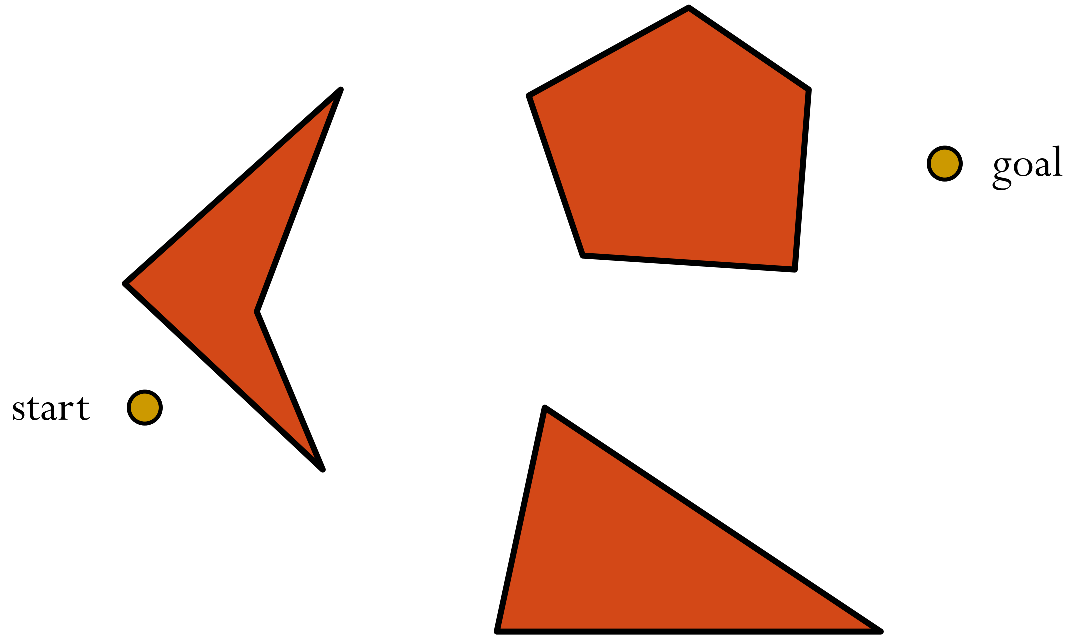
Framework

Framework

- Continuous representation
- Discretization
- Graph searching

Modified based on Slides
by Prof. David Hsu, University of Singapore

Continuous representation

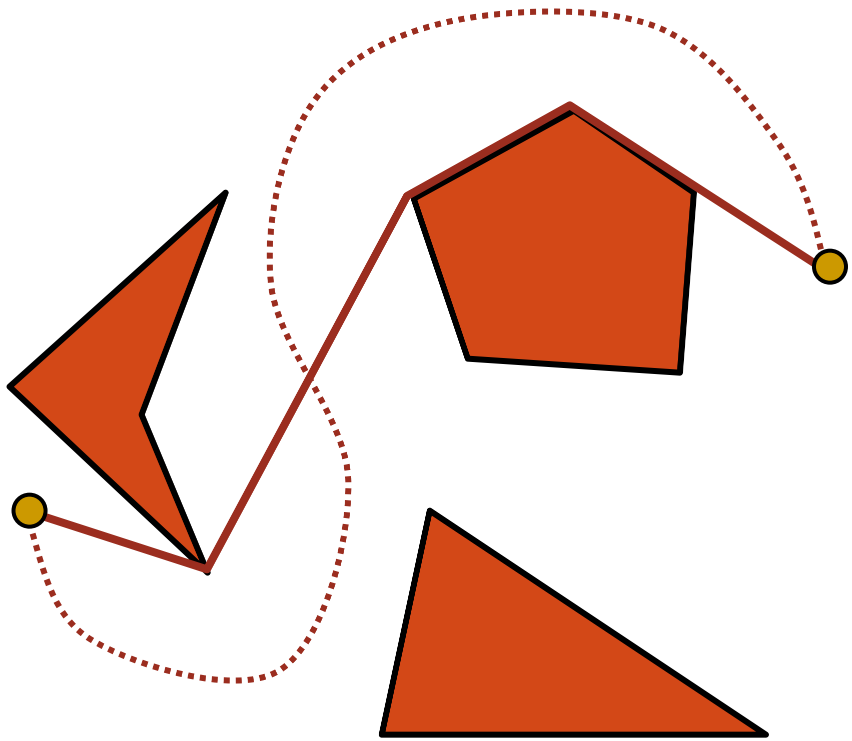


Framework

- Continuous representation
- **Discretization**
  - Sampling (random, with bias)
  - **Processing critical geometric features**
- Graph searching

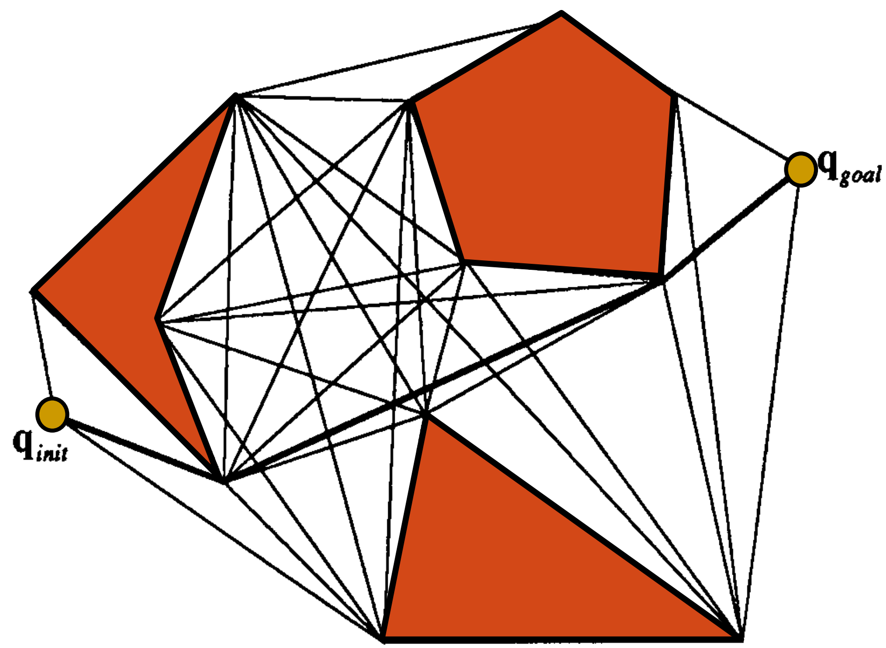
Discretization – Visibility Graph

- If a collision-free path exists
  - There must be a piecewise linear path that bends only at the obstacles vertices



Visibility Graph

- Nodes
  - q_{init}, q_{goal}, obstacle vertices
- Edges
  - Obstacle edges
  - No intersection with obstacles



Naïve Algorithm for Computing Visibility Graph

Input: q_{init}, q_{goal}, polygonal obstacles
Output: visibility graph G

```
1: for every pair of nodes u,v
2:   if segment(u,v) is an obstacle edge then
3:     insert edge(u,v) into G;
4:   else
5:     for every obstacle edge e
6:       if segment(u,v) intersects e
7:         go to (1);
8:     insert edge(u,v) into G.
```

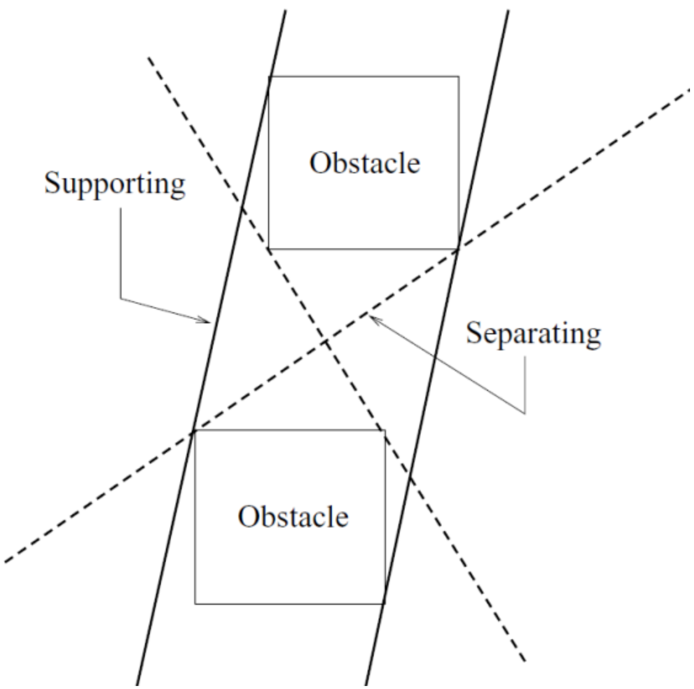
Running time?

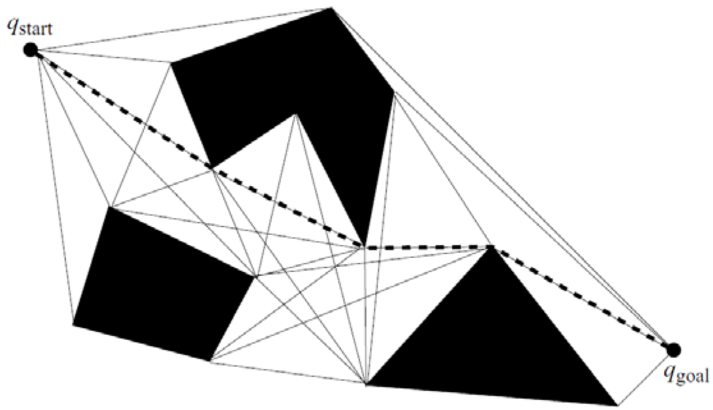
```
1: for every pair of nodes u,v                        O(n^2)
2:  if segment(u,v) is an obstacle edge then          O(n)
3:    insert edge(u,v) into G;
4:  else
5:    for every obstacle edge e                       O(n)
6:      if segment(u,v) intersects e
7:        go to (1);
8:    insert edge(u,v) into G.
```

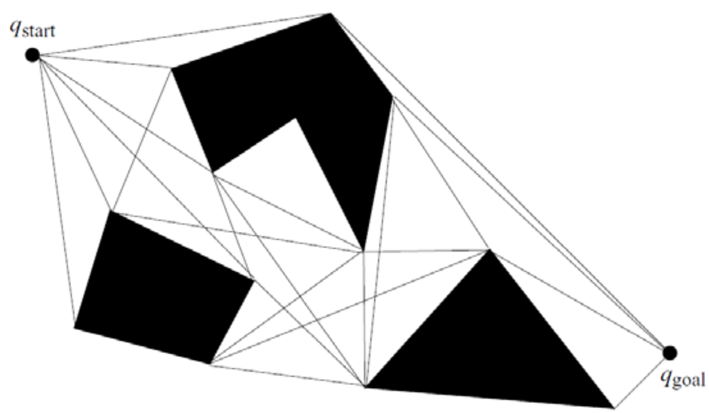
- Running time? **O(n^3)**
- More efficient algorithm?
  - Sweep-line algorithm – **O(n^2 log n)** (see Principles 5.1.2)
  - Optimal –Using line arrangement – **O(n^2)**

Reduced Visibility Graph

- Construct visibility graph from
  - Supporting lines
  - Separating lines





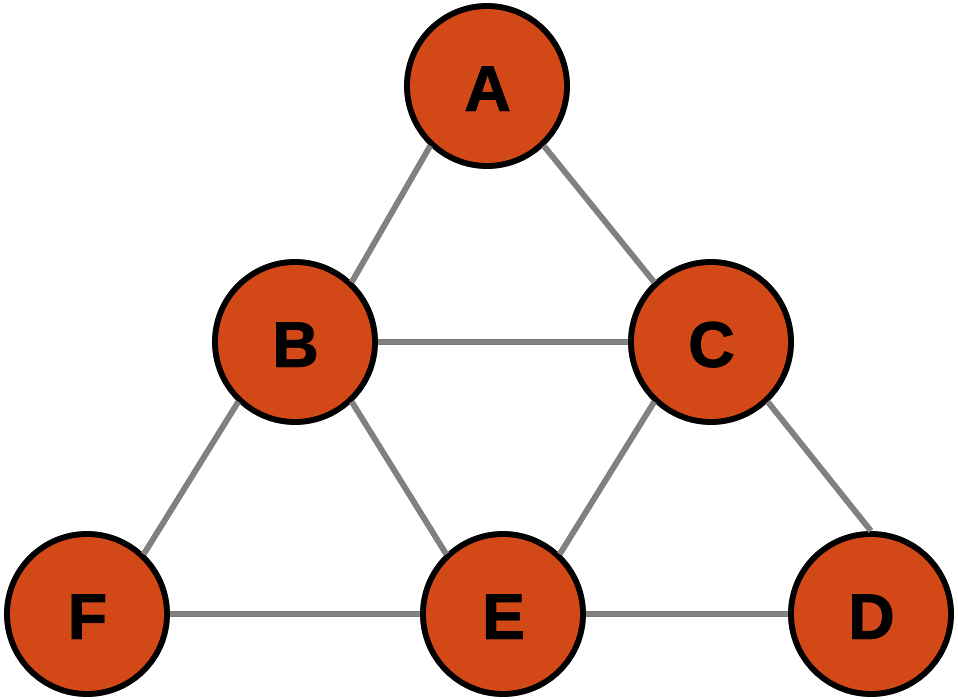


Framework

- Continuous representation
- Discretization
  - Sampling (random, with bias)
  - Processing critical geometric features
- **Graph searching**
  - Breadth first, depth first, A*, Dijkstra, etc

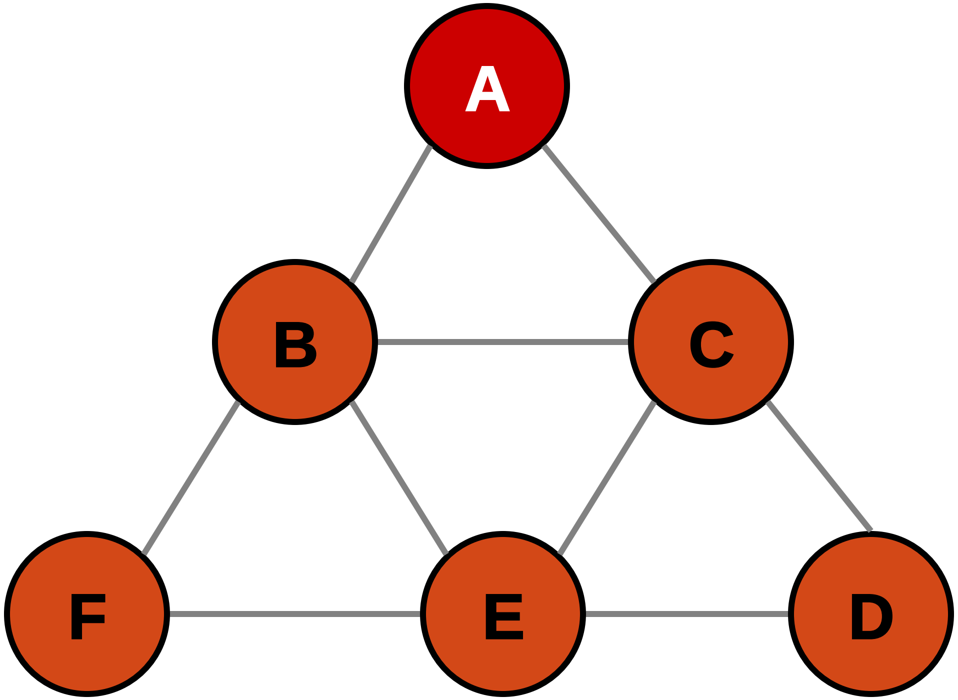
Modified based on Slides
by Prof. David Hsu, University of Singapore

Breadth-first search



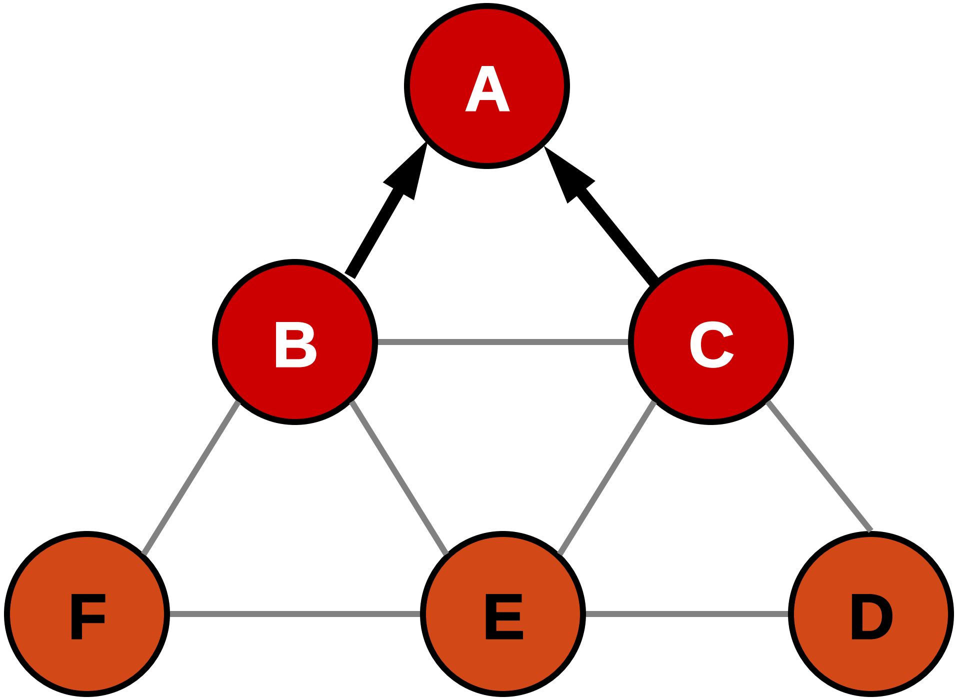
Modified based on Slides
by Prof. David Hsu, University of Singapore

Breadth-first search



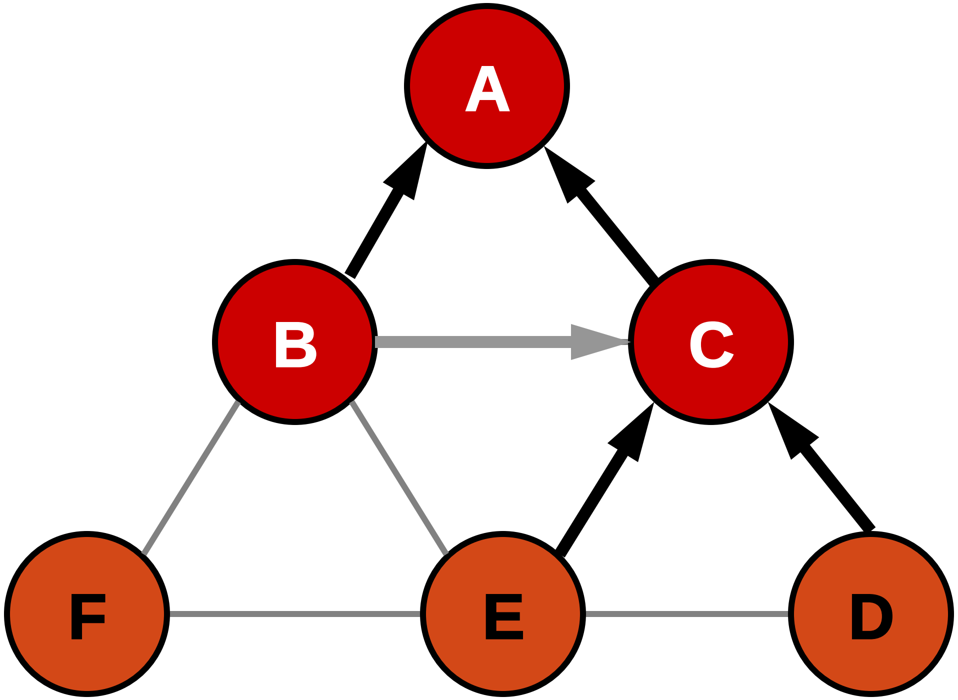
Modified based on Slides
by Prof. David Hsu, University of Singapore

Breadth-first search



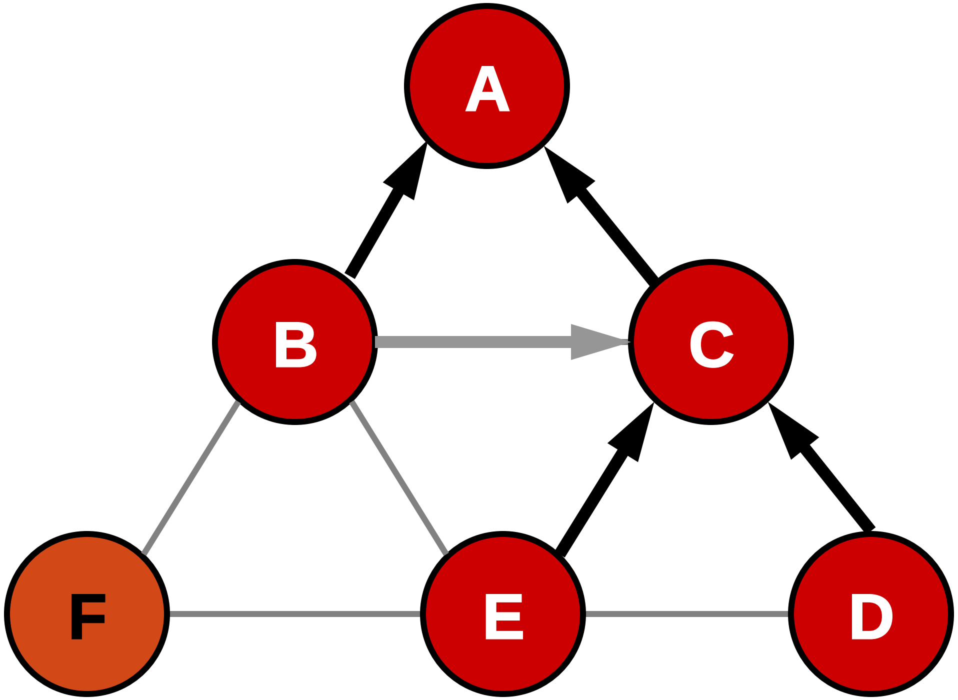
Modified based on Slides
by Prof. David Hsu, University of Singapore

Breadth-first search



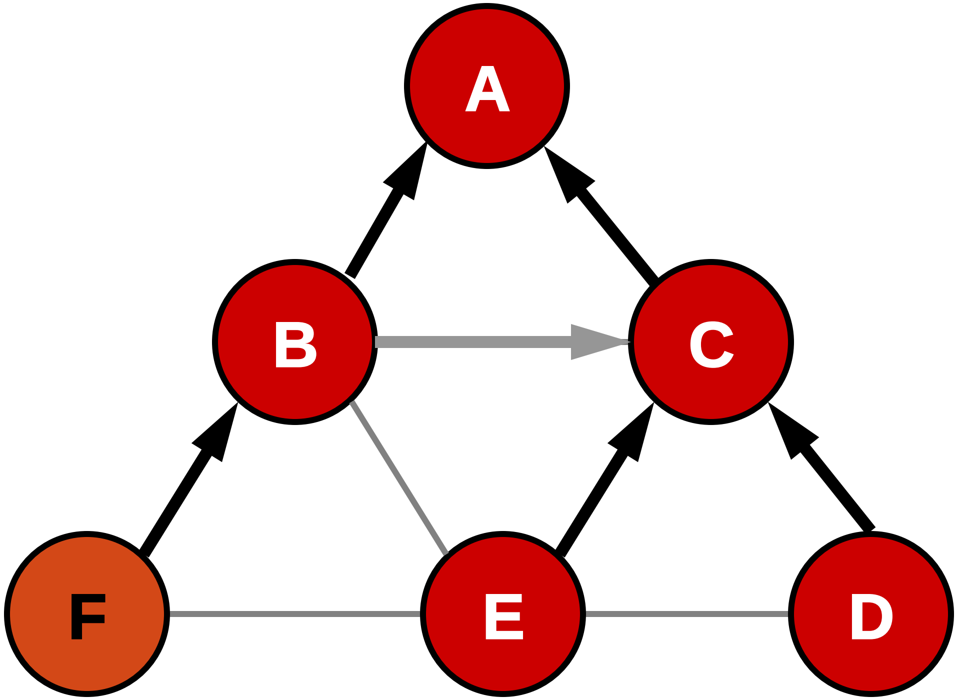
Modified based on Slides
by Prof. David Hsu, University of Singapore

Breadth-first search



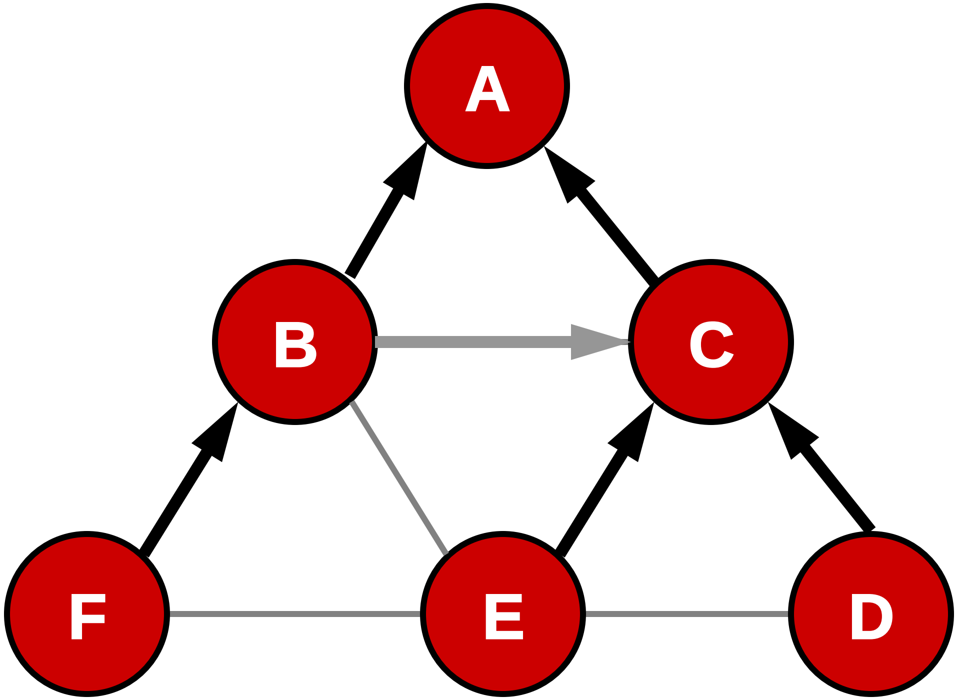
Modified based on Slides
by Prof. David Hsu, University of Singapore

Breadth-first search



Modified based on Slides
by Prof. David Hsu, University of Singapore

Breadth-first search

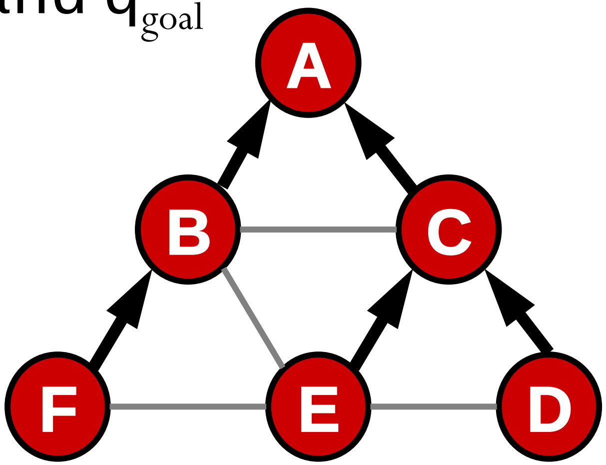


Modified based on Slides
by Prof. David Hsu, University of Singapore

Breadth-first search

Input: q_{init}, q_{goal}, visibility graph G
Output: a path between q_{init} and q_{goal}

```
Q = new queue;
Q.enqueue(q_init);
mark q_init as visited;
while Q is not empty
   curr = Q.dequeue();
   if curr == q_goal then
      return curr;
   for each w adjacent to curr
      if w is not visited
         w.parent = curr;
          Q.enqueue(w)
          mark w as visited
```



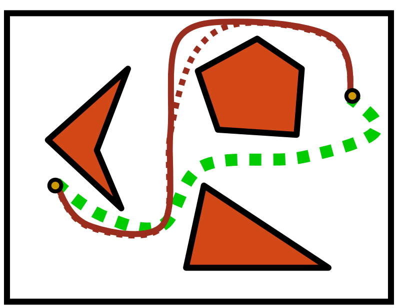
Other graph search algorithms

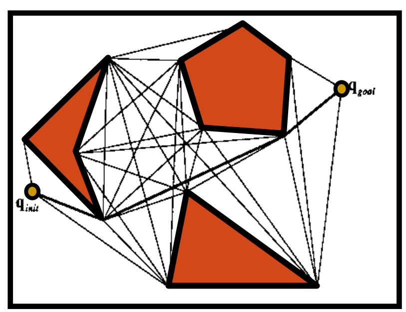
- Depth-first
  - Explore newly-discovered nodes first
  - Guaranteed to generate shortest path in the graph? **No**
- Dijkstra's Search
  - Find shortest paths to the goal node in the graph from the start
- A*
  - Heuristically-guided search
  - Guaranteed to find shortest path

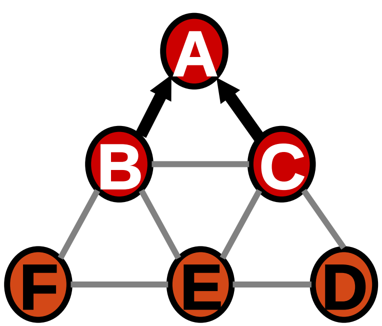
Modified based on Slides
by Prof. David Hsu, University of Singapore

Recap

- Continuous representation
- Discretization
  - Visibility graph
- Graph searching
  - Breadth first search







Recap

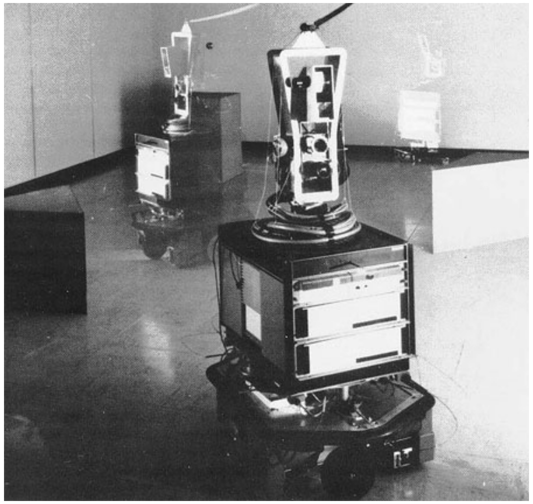
- Running time
  - Compute the visibility graph – Naïve method – O(n^3)
    - An optimal $O(n^2)$ time algorithm exists.
- Space?
  - Store graph as adjacency list or adjacency matrix **O(n^2)**

Classic path planning approaches

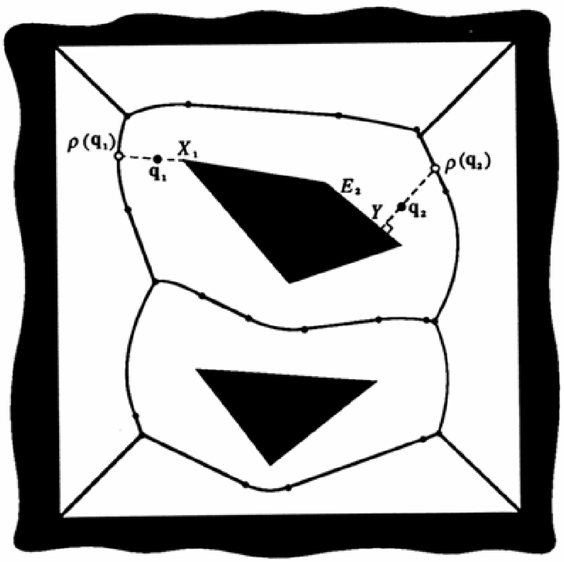
- **Roadmap**
  - Represent the **connectivity** of the free space by a network of **1-D curves**
- **Cell decomposition**
  - **Decompose** the free space **into** simple **cells** and represent the connectivity of the free space by the **adjacency graph** of these cells
- **Potential field**
  - Define a **potential function** over the free space that has a global minimum at the goal and follow the steepest descent of the potential function

Roadmap

Roadmaps



- Visibility Graph
  - Shakey robot, SRI [Nilsson, 1969]
- **Voronoi graph**
  - Introduced by **computational geometry** researchers.
  - Generate paths that **maximizes clearance**.

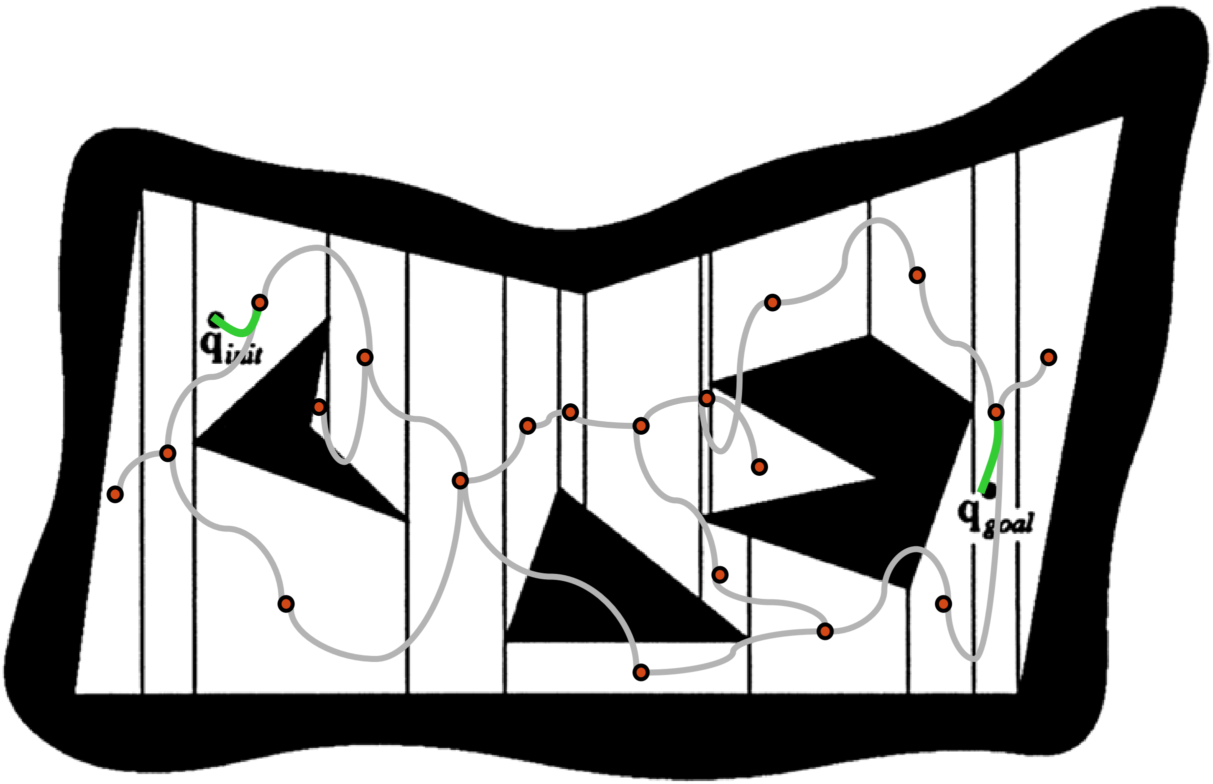


Cell decomposition

Modified based on Slides
by Prof. David Hsu, University of Singapore

Cell decomposition

- Exact methods
  - 2D - Trapezoids, triangles, etc.
  - Adjacency map

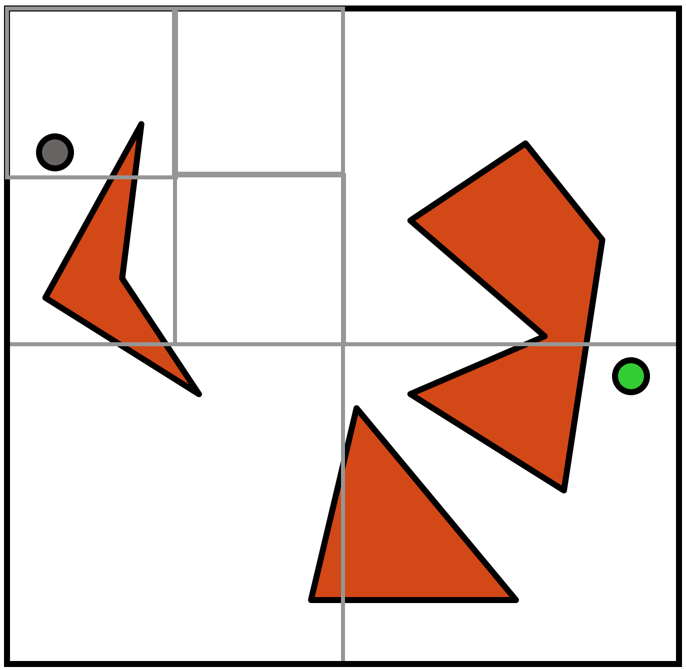


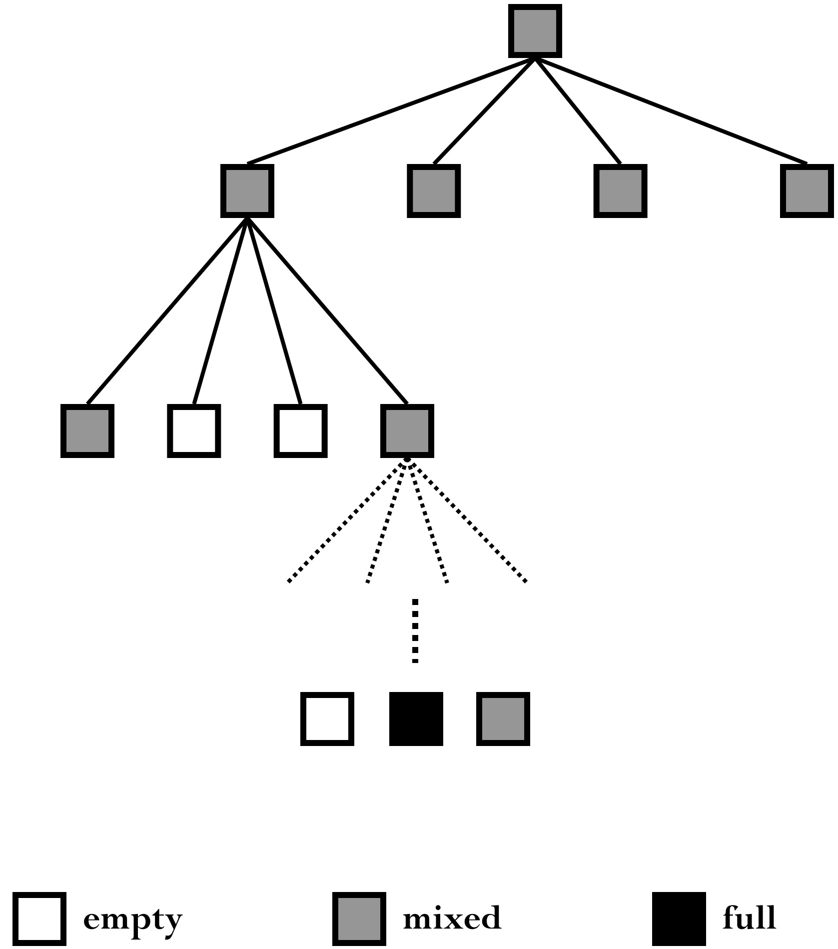
Cell decomposition

- Approximate methods
  - Decompose space into cells usually have **simple, regular** shapes, e.g., rectangles, squares.
  - Facilitate **hierarchical** space decomposition

Modified based on Slides
by Prof. David Hsu, University of Singapore

Quadtree decomposition

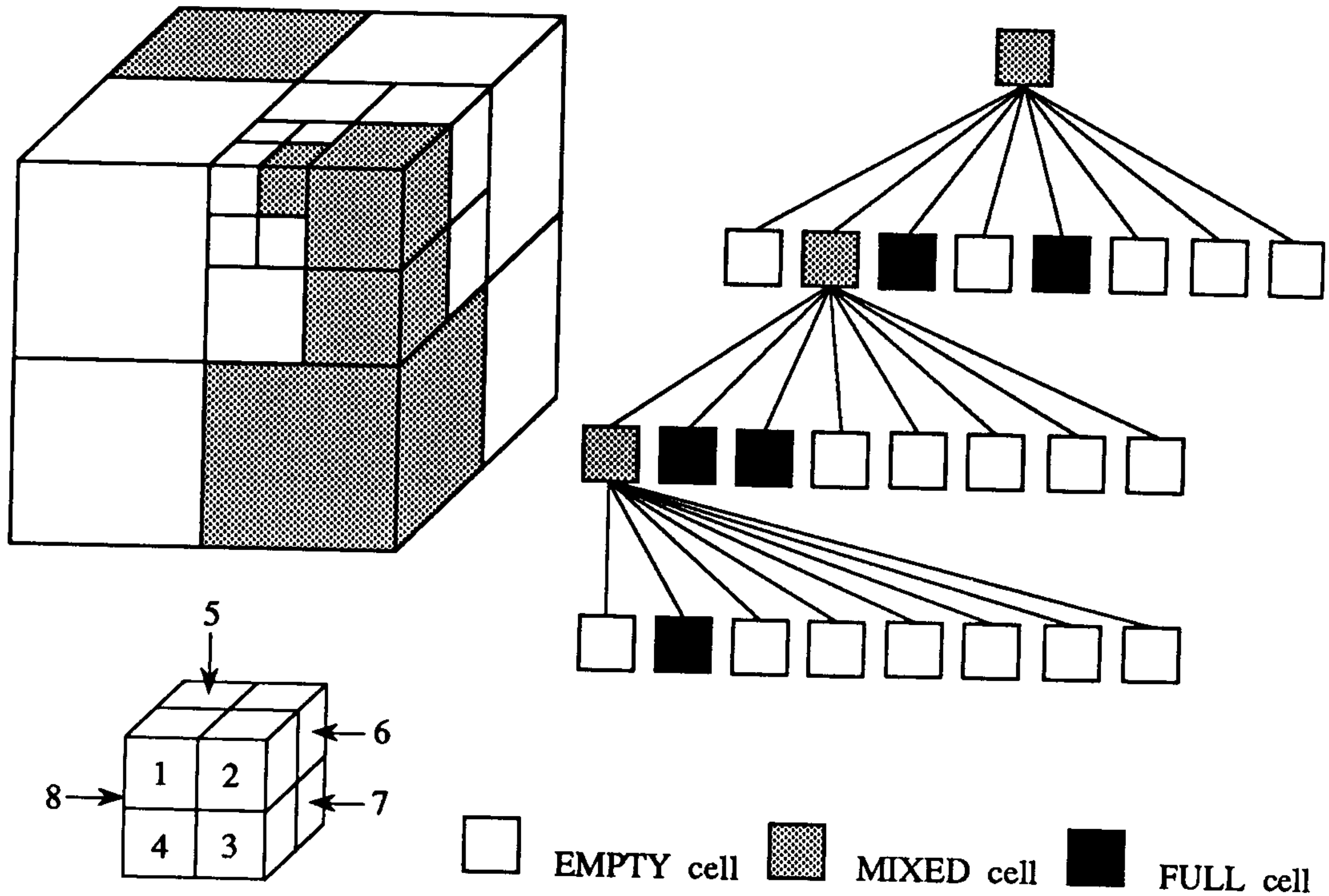




Hierarchical Decomposition

- Strategy
  - **Decompose** the free space into cells.
  - **Search** for a sequence of **mixed or empty cells** that connect the initial and goal positions.
  - **Further decompose** the **mixed**.
  - Repeat (2) and (3) until a sequence of empty cells is found.

Octree decomposition

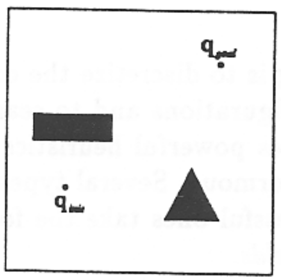


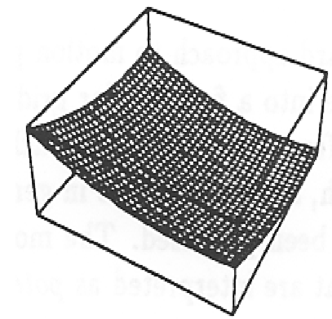
Modified based on Slides
by Prof. David Hsu, University of Singapore

Potential Field

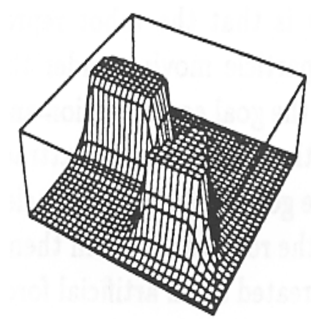
Potential field

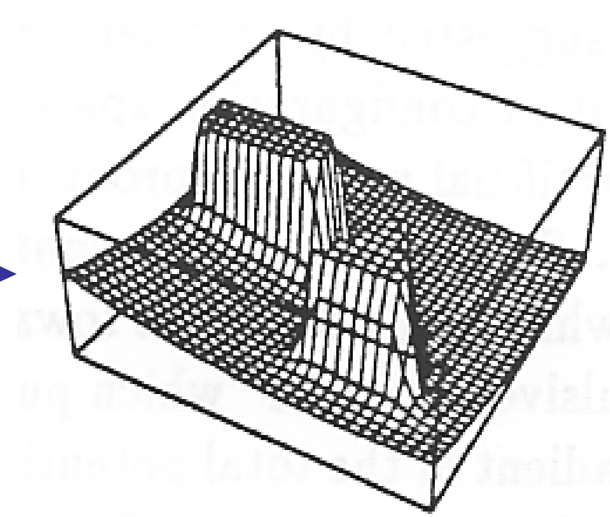
$$\phi_{\text{att}} = \frac{1}{2} k_{\text{att}} (x - x_{\text{goal}})^2$$





+





$\boldsymbol{k_{att}}$, $\boldsymbol{k_{rep}}$: positive scaling factors

$\boldsymbol{x}$: position of the robot

$\boldsymbol{\rho}$: distance to the obstacle

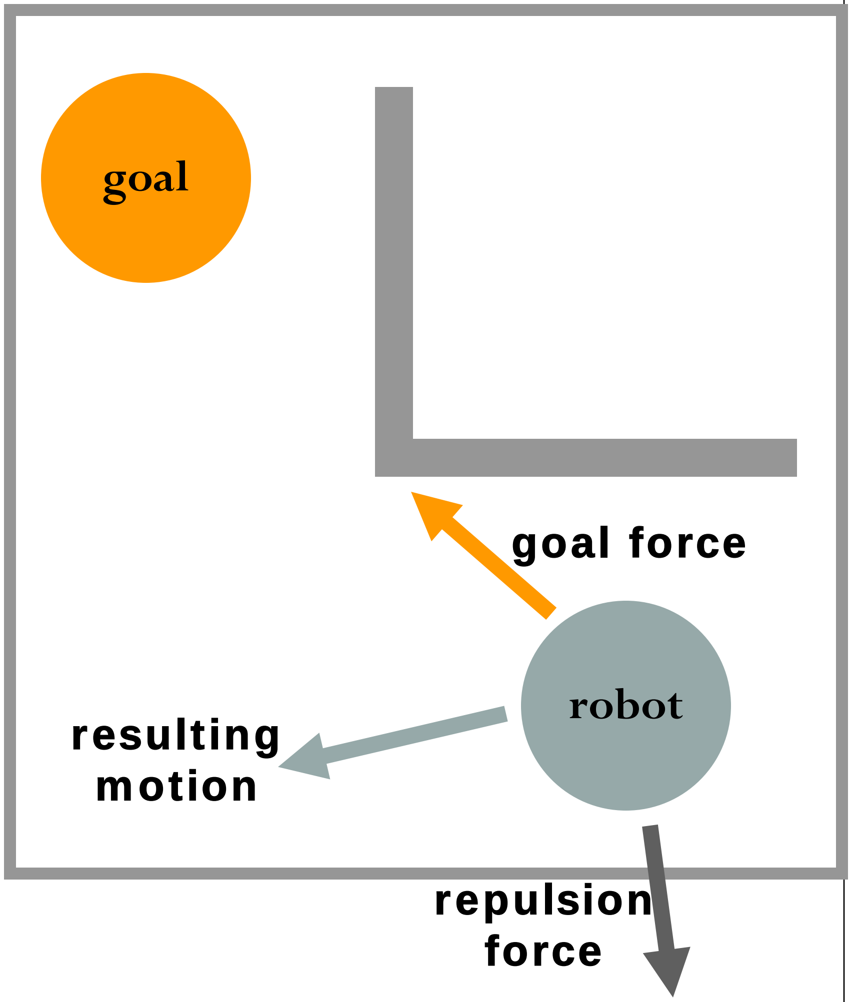
$\boldsymbol{\rho_0}$: distance of influence

$$\phi_{\text{rep}} = \begin{cases} \frac{1}{2} k_{\text{rep}} \left(\frac{1}{\rho} - \frac{1}{\rho_0} \right)^2 & \text{if } \rho \le \rho_0, \\ 0 & \text{if } \rho > \rho_0 \end{cases}$$

Attractive & repulsive fields

$$F_{\text{att}} = -\nabla \phi_{\text{att}} = -k_{\text{att}}(x - x_{\text{goal}})$$

$$F_{\text{rep}} = -\nabla \phi_{\text{rep}} = \begin{cases} k_{\text{rep}} \left(\frac{1}{\rho} - \frac{1}{\rho_0} \right) \frac{1}{\rho^2} \frac{\partial \rho}{\partial x} & \text{if } \rho \le \rho_0, \\ 0 & \text{if } \rho > \rho_0 \end{cases}$$



$k_{\text{att}}, k_{\text{rep}}$: positive scaling factors

x : position of the robot

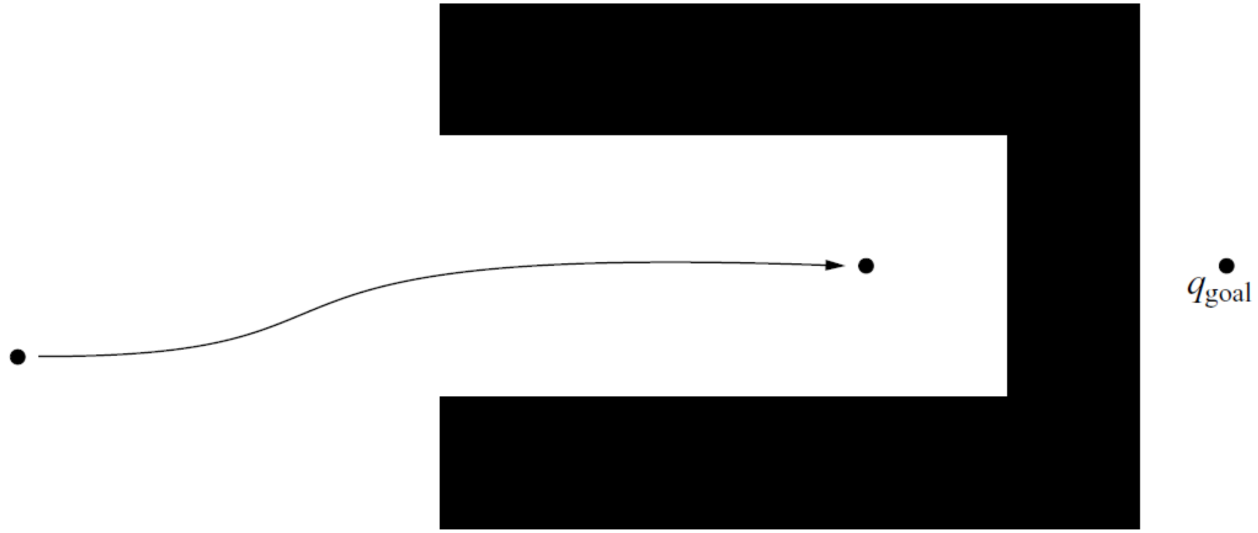
ρ : distance to the obstacle

ρ_0 : distance of influence

[Khatib, 1986]

Local minima

- How to get out of local minima?
  - Back up
  - Random Walk
  - Wall following



Note that …

- A potential field is a **scalar** function over the free space.
- To navigate, the robot applies a force **proportional to gradient** of the potential field, in the **opposite direction**.
- Ideally potential field function?
  - has global minimum at the goal
  - has no local minima
  - grows to infinity near obstacles
  - is smooth

Completeness

- A **complete motion planner** always returns a solution when one exists and indicates that no such solution exists otherwise.
  - Is the visibility graph algorithm complete?
  - Is the exact cell decomposition algorithm complete?
  - Is the approximate cell decomposition algorithm complete?
  - Is the potential field algorithm complete?

Homework

- Read Principles CH 3 – Configuration space