PATH PLANNING

Undergraduate course (Spring 2020)

HUMAN MACHINE INTERACTION LABORATORY

MOTION PLANNING

What is Motion Planning

A robot arm is to build an assembly from a set of parts.

Tasks for the robot:

- Grasping: position gripper on object
  design a path to this position
- Transferring: determine geometry path for arm
  avoid obstacles + clearance
- Positioning

Information Required

- Knowledge of spatial arrangement of workspace. E.g., location of obstacles
- **Full knowledge ->** full motion planning
- **Partial knowledge ->** combine planning and execution

motion planning = collection of problems

Basic Problem

A simplified version of the problem assumes

- Robot is the only moving object in the workspace
- No dynamics, no temporal issues
- Only non-contact motions

Motion Planning = pure “geometrical” problem

World consists of

- Obstacles
  - Already occupied spaces of the world
  - In other words, robots can’t go there
- Free Space
  - Unoccupied space within the world
  - Robots “might” be able to go here
  - To determine where a robot can go, we need to discuss what a *Configuration Space* is

Notion of Configuration Space

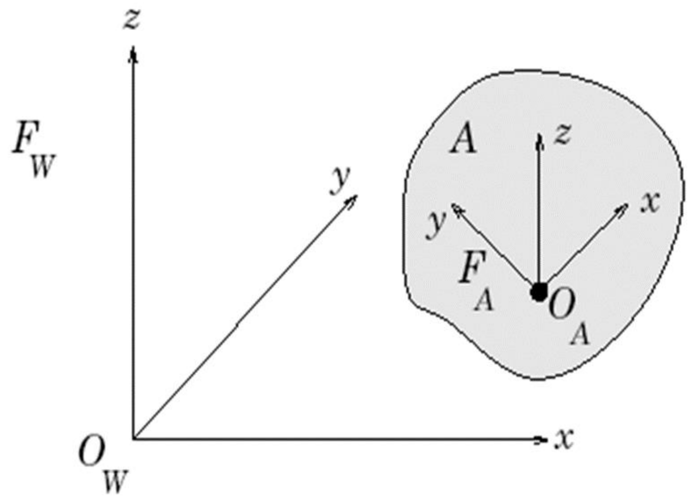
- *Main Idea:* Represent the robot as a point, called a configuration, in a parameter space, the *configuration space* (or C-space).

- Importance: Reduce the problem of planning the motion of a robot in Euclidean Space to planning the motion of a point in C-space.

C-space of Rigid Objects

Workspace $\mathcal{W}$: physical workspace

- represented as N-dimensional Euclidean Space $\mathbb{R}^N$, where $N = 2, 3$
- $\mathcal{F}_\mathcal{W}$: *fixed* Cartesian coordinate system (frame) of $\mathcal{W}$
- $\mathcal{O}_\mathcal{W}$: *fixed* origin of $\mathcal{F}_\mathcal{W}$



Robot $\mathcal{A}$: moving rigid object/robot

- represented as compact subset of $\mathbb{R}^N$ (at reference position and orientation)
- $\mathcal{F}_\mathcal{A}$: frame of $\mathcal{A}$ (aka 'local' frame of $\mathcal{A}$)
  - *fixed* wrt $\mathcal{A}$ (i.e., each point in $\mathcal{A}$ has fixed coordinates in $\mathcal{F}_\mathcal{A}$)
  - *moving* wrt $\mathcal{F}_\mathcal{W}$
- $\mathcal{O}_\mathcal{A}$: origin of $\mathcal{F}_\mathcal{A}$ (aka the **reference point** of $\mathcal{A}$)

C-space of Rigid Objects

Definitions:

- A **configuration q** of $\mathcal{A}$ is a specification of the position and orientation of $\mathcal{F}_{\mathcal{A}}$ wrt $\mathcal{F}_{\mathcal{W}}$
- The **configuration space** of $\mathcal{A}$ is the space $\mathcal{C}$ of all the possible configurations of $\mathcal{A}$

Notation:

- $\mathcal{A}(\mathbf{q})$: subset of $\mathcal{W}$ occupied by $\mathcal{A}$ at configuration $\mathbf{q}$
- $a(\mathbf{q})$: position of point $a \in \mathcal{A}$ in $\mathcal{W}$ when $\mathcal{A}$ at configuration $\mathbf{q}$

C-space of Rigid Objects

Robot Configurations Can be:

- 1. Free configurations: robot and obstacles do not overlap
- 2. Contact configurations: robot and obstacles touch
- 3. Blocked configurations: robot and obstacles overlap
- Configuration Space partitioned into free (C_free), contact, and blocked sets.

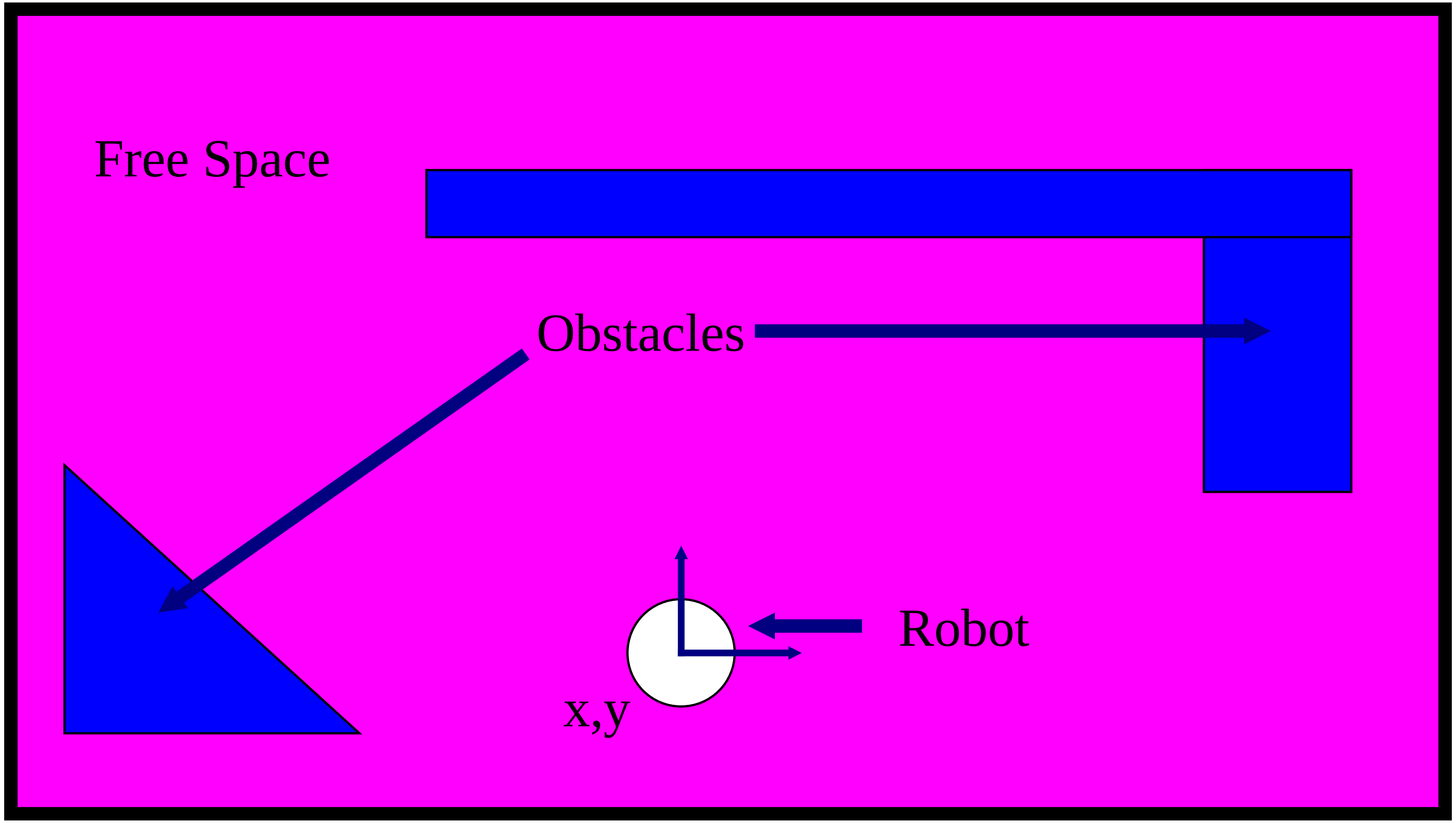
Definition: The obstacle $\mathcal{B}_i$ in $\mathcal{W}$ maps in $\mathcal{C}$ to the region

$$\mathcal{CB}_i = \{\mathbf{q} \in \mathcal{C} | \mathcal{A}(\mathbf{q}) \cap \mathcal{B}_i \neq \emptyset\}$$

$\mathcal{CB}_i$ is called a *C-obstacle*. The union of all C-obstacles is called the *C-obstacle region*.

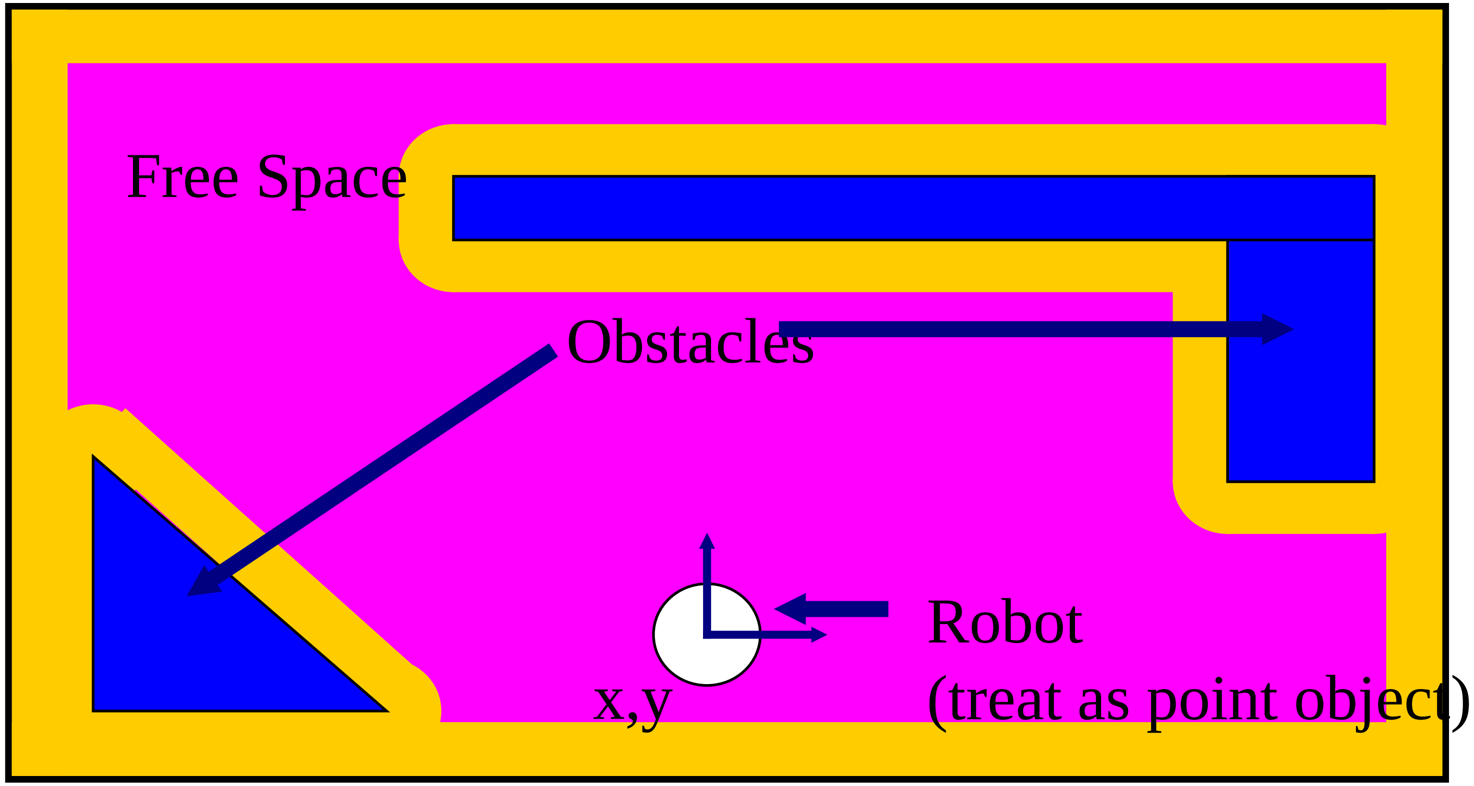
C_space = C_free + C_obstacle

Example of a World (and Robot)



The Configuration Space

- How to create it
  - First abstract the robot as a point object.
  - Then, enlarge the obstacles to account for the robot's footprint and degrees of freedom
  - In example, the robot was circular, so it enlarge obstacles by the robot's radius.



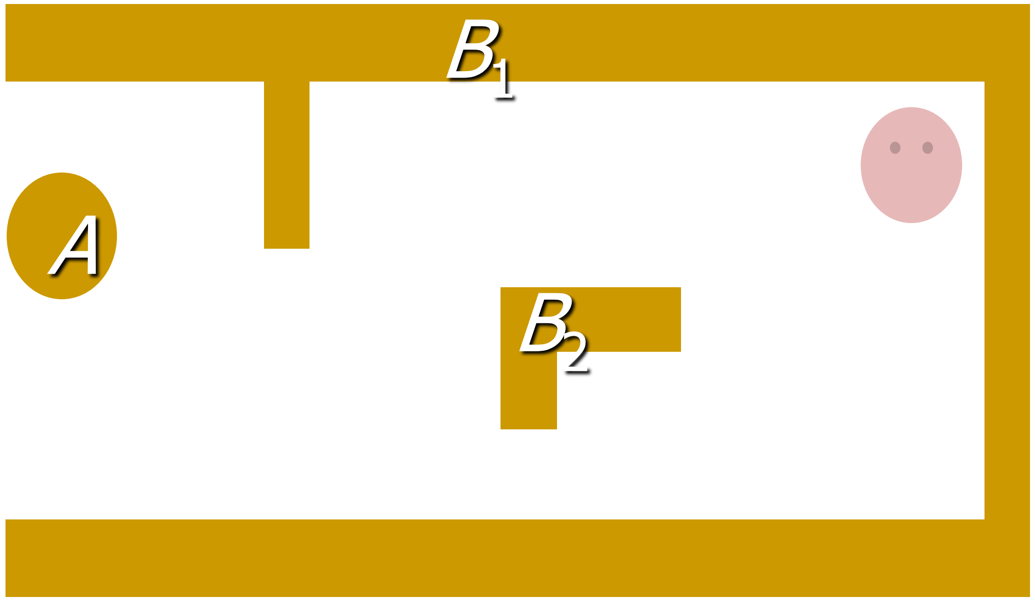
Components

- A: single rigid object - the robot - moving in Euclidean space W (the workspace).

  $W = R^N$, $N=2,3$
- B_i , I=1,…,q. Rigid objects in W. The obstacles

<u>Assume</u>

- Geometry of A and B_i is perfectly known
- Location of B_i is known
- No kinematic constraints on A: a “free flying” object



Path in C-Space

- Mathematically definition

A *path in C-space* is a continuous map:

$$\tau : s \in [0,1] \mapsto \tau(s) \in \mathcal{C}$$

where $\tau(0) = \mathbf{q}_{init}$ is the initial configuration and $\tau(1) = \mathbf{q}_{goal}$ is the goal configuration of the path.

"Continuous map" means that:

$$\forall s_1, s_2 \in [0,1] : \lim_{s_2 \to s_1} d(\tau(s_1), \tau(s_2)) = 0$$

where $d : \mathcal{C} \times \mathcal{C} \to \mathbf{R}^+ \cup \{0\}$ is the chosen metric over $\mathcal{C}$.

Motion Planning

- **General Goal**: compute motion commands to achieve a goal arrangement of physical objects from an initial arrangement
- **Basic problem**: Collision-free path planning for one rigid or articulated object (the “robot”) among static obstacles.

Inputs

- geometric descriptions of the obstacles and the robot
- kinematic and dynamic properties of the robot
- initial and goal positions (configurations) of the robot

Output

- Continuous sequence of collision-free configurations connecting the initial and goal configurations

Motion Planning

- **Path planning**

  design of only geometric (**kinematic**) *specifications* of the **positions** and **orientations** of robots
- **Trajectory = Path** + assignment of **time** to points along the path
- **Trajectory planning**

  path planning + design of **linear** and **angular velocities**
- **Motion Planning (MP),** a general term, either:
  - **Path** planning, or
  - **Trajectory** planning
- **Path planning < Trajectory planning**

Classification of MP algorithms

<u>Completeness</u>

- **Exact**
  - usually computationally expensive
- **Heuristic**
  - aimed at generating a solution in a short time
  - may fail to find solution or find poor one at difficult problems
  - important in engineering applications
- **Probabilistically complete** (probabilistic completeness →1)

Classification of MP algorithms

<u>Scope</u>

- **Global**
  - take into account all environment information
  - plan a motion from start to goal configuration

- **Local**
  - avoid obstacles in the vicinity of the robot
  - use information about nearby obstacles only
  - used when start and goal are close together
  - used as component in global planner, or
  - used as safety feature to avoid unexpected obstacles not present in environment model, but sensed during motion

Point-to-point Path Planning

- Point-to-point path planning:
  - Looks for the best route to move an entity from point A to point B
  - Avoiding known obstacles in its path
  - Not leaving the map boundaries, and not violating the entity's mobility constraints.
- This type of path planning is used:
  - Finding routes for autonomous robots
  - Planning the manipulator's movement of a stationary robot
  - For moving entities to different locations in a map to accomplish certain goals in a gaming application.

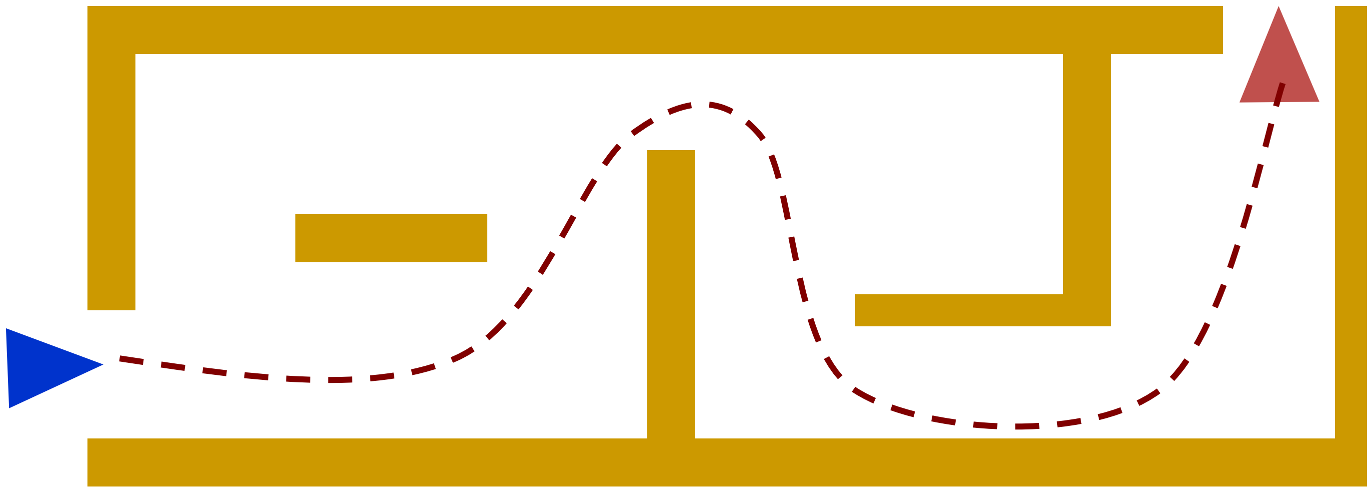
HMI
HUMAN MACHINE INTERACTION LABORATORY

Region Filling Path Planning

- Tasks such as vacuuming a room, plowing a field, or mowing a lawn require region filling path planning operations that are defined as follows:
  - The mobile robot must move through an entire area, i.e., the overall travel must cover a whole region.
  - Continuous and sequential operation without any repetition of paths is required of the robot.
  - The robot must avoid all obstacles in a region.
  - An "optimal" path is desired under the available conditions.

Components

- The Problem:
  - Given an initial position and orientation PO_{init}
  - Given a goal position and orientation PO_{goal}
  - Generate: continuous path t from PO_{init} to PO_{goal}
- t is a continuous sequence of Pos'



THE WAVEFRONT PLANNER

The Wavefront Planner

- A common algorithm used to determine the shortest paths between two points
  - In essence, a breadth first search of a graph

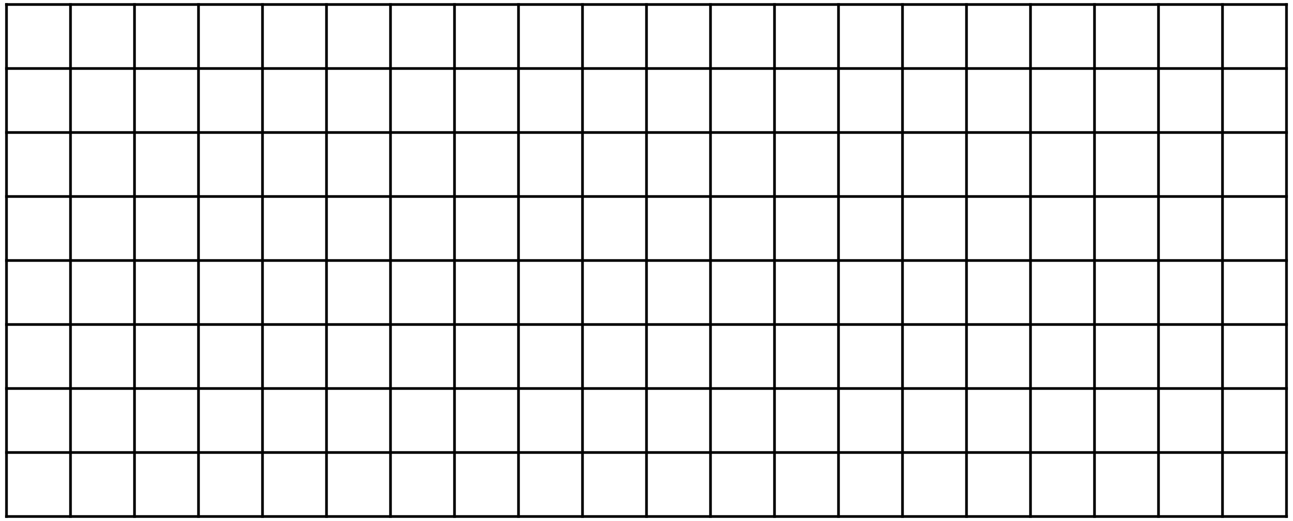
- For simplification, we'll present the world as a two-dimensional grid

- Setup:
  - Label free space with 0
  - Label C-Obstacle as 1
  - Label the destination as 2

7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
3	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

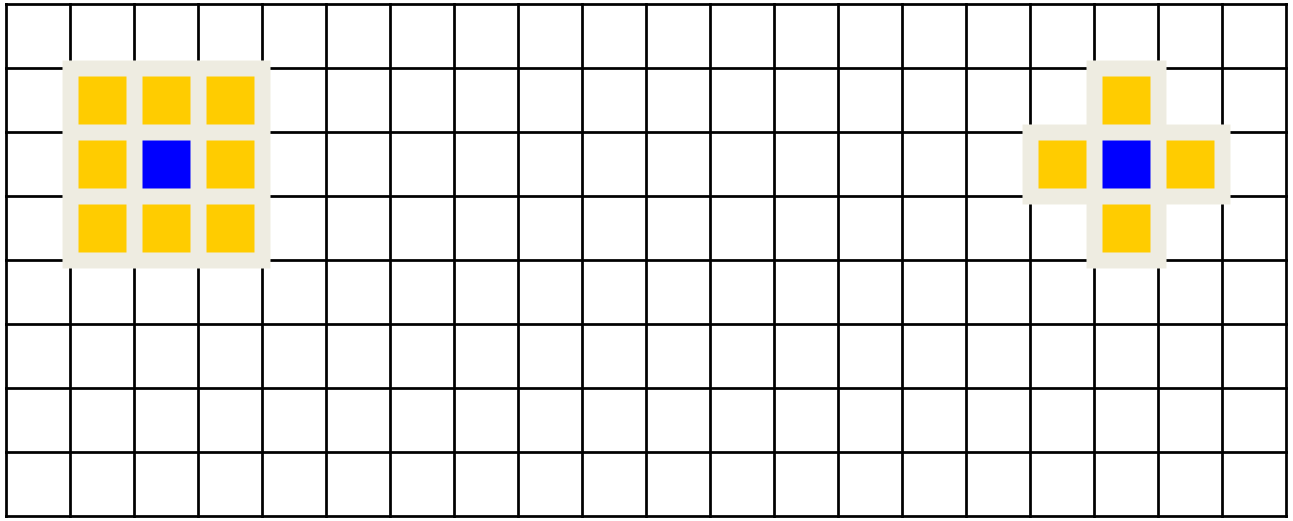
Representations: A Grid

- Distance is reduced to discrete steps
  - For simplicity, we’ll assume distance is uniform
- Direction is now limited from one adjacent cell to another



Representations: Connectivity

- 8-Point Connectivity
- 4-Point Connectivity



The Wavefront in Action (Part 1)

- Starting with the goal, set all adjacent cells with “0” to the current cell + 1
  - 4-Point Connectivity or 8-Point Connectivity?
  - Your Choice. We’ll use 8-Point Connectivity in our example

7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
3	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	2
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

HMI
HUMAN MACHINE INTERACTION LABORATORY

The Wavefront in Action (Part 2)

- Now repeat with the modified cells
  - This will be repeated until no 0’s are adjacent to cells with values >= 2
    - 0’s will only remain when regions are unreachable

7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
3	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4
1	0	0	0	0	0	0	0	0	0	0	0	0	0	4	3	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	3	2
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The Wavefront in Action (Part 3)

- Repeat again...

7	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
3	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	5	5	5	5
2	0	0	0	0	0	0	0	0	0	0	0	0	5	4	4	4
1	0	0	0	0	0	0	0	0	0	0	0	0	5	4	3	3
0	0	0	0	0	0	0	0	0	0	0	0	0	5	4	3	**2**
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The Wavefront in Action (Part 4)

- And again...

7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	1	1	1	1	1	1	1	6	6	6	6
3	0	0	0	0	1	1	1	1	1	1	1	1	5	5	5	5
2	0	0	0	0	0	0	0	0	0	0	0	6	5	4	4	4
1	0	0	0	0	0	0	0	0	0	0	0	6	5	4	3	3
0	0	0	0	0	0	0	0	0	0	0	0	6	5	4	3	2
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The Wavefront in Action (Part 5)

- And again until…

7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	7	7	7	7	7
4	0	0	0	0	1	1	1	1	1	1	1	1	6	6	6	6
3	0	0	0	0	1	1	1	1	1	1	1	1	5	5	5	5
2	0	0	0	0	0	0	0	0	0	0	7	6	5	4	4	4
1	0	0	0	0	0	0	0	0	0	0	7	6	5	4	3	3
0	0	0	0	0	0	0	0	0	0	0	7	6	5	4	3	2
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The Wavefront in Action (Done)

- You’re done
  - Remember, 0’s should only remain if unreachable regions exist

7	**18**	17	16	15	14	13	12	11	10	9	9	9	9	9	9	9
6	17	17	16	15	14	13	12	11	10	9	8	8	8	8	8	8
5	17	16	16	15	14	13	12	11	10	9	8	7	7	7	7	7
4	17	16	15	15	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	6	6	6	6
3	17	16	15	14	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	5	5	5	5
2	17	16	15	14	13	12	11	10	9	8	7	6	5	4	4	4
1	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	3
0	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	**2**
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The Wavefront, Now What?

- To find the shortest path, according to your metric, simply always move toward a cell with a lower number
  - The numbers generated by the Wavefront planner are roughly proportional to their distance from the goal

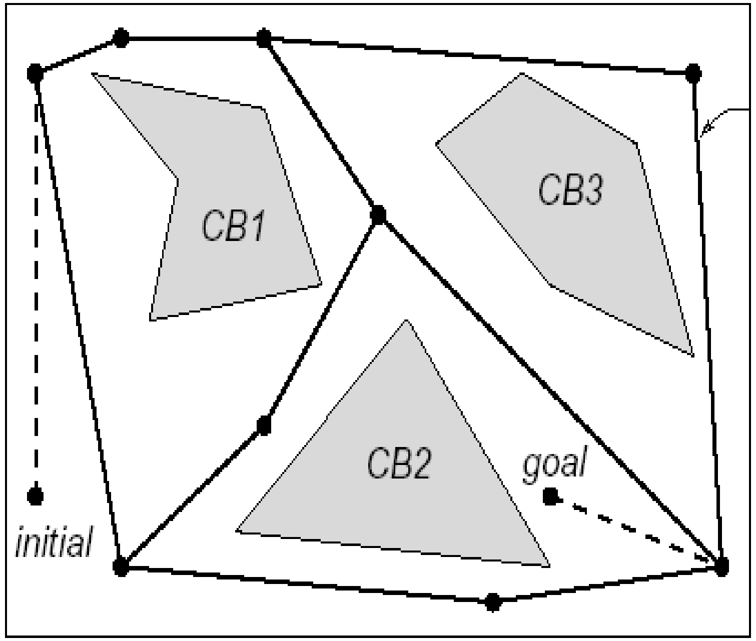
Two possible shortest paths shown

7	**18**	17	16	15	14	13	12	11	10	9	9	9	9	9	9	9
6	17	17	16	15	14	13	12	11	10	9	8	8	8	8	8	8
5	17	16	16	15	14	13	12	11	10	9	8	7	7	7	7	7
4	17	16	15	15	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	6	6	6	6
3	17	16	15	14	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	5	5	5	5
2	17	16	15	14	13	12	11	10	9	8	7	6	5	4	4	4
1	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	3
0	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	**2**
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

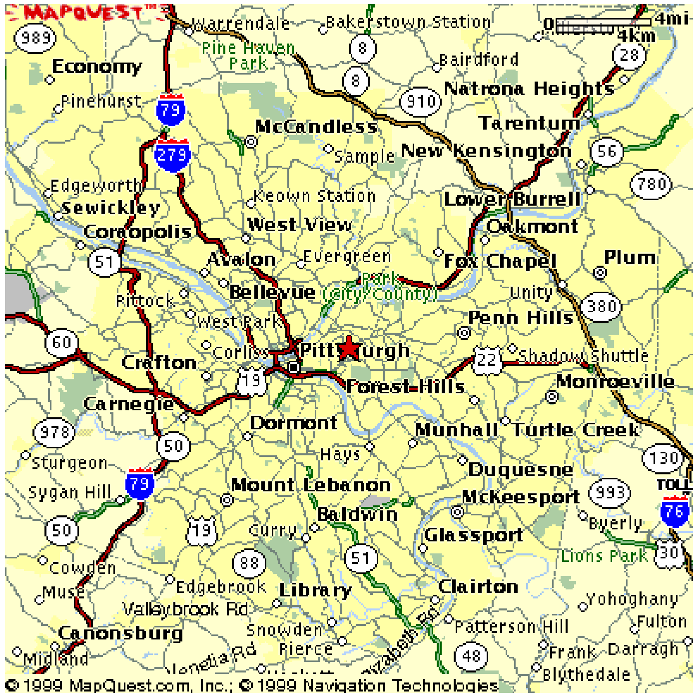
MAP-BASED APPROACHES: VISIBILITY GRAPH

Map-Based Approaches: Roadmap Theory

- Idea: capture the connectivity of C-free with a roadmap (graph or network) of one-dimensional curves



Free configurations: robot and obstacles do not overlap



Roadmap Methods

<u>PATH PLANNING WITH A ROADMAP</u>

input: configurations $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$, and $\mathcal{B}$

output: a path in $\mathcal{C}_{free}$ connecting $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$

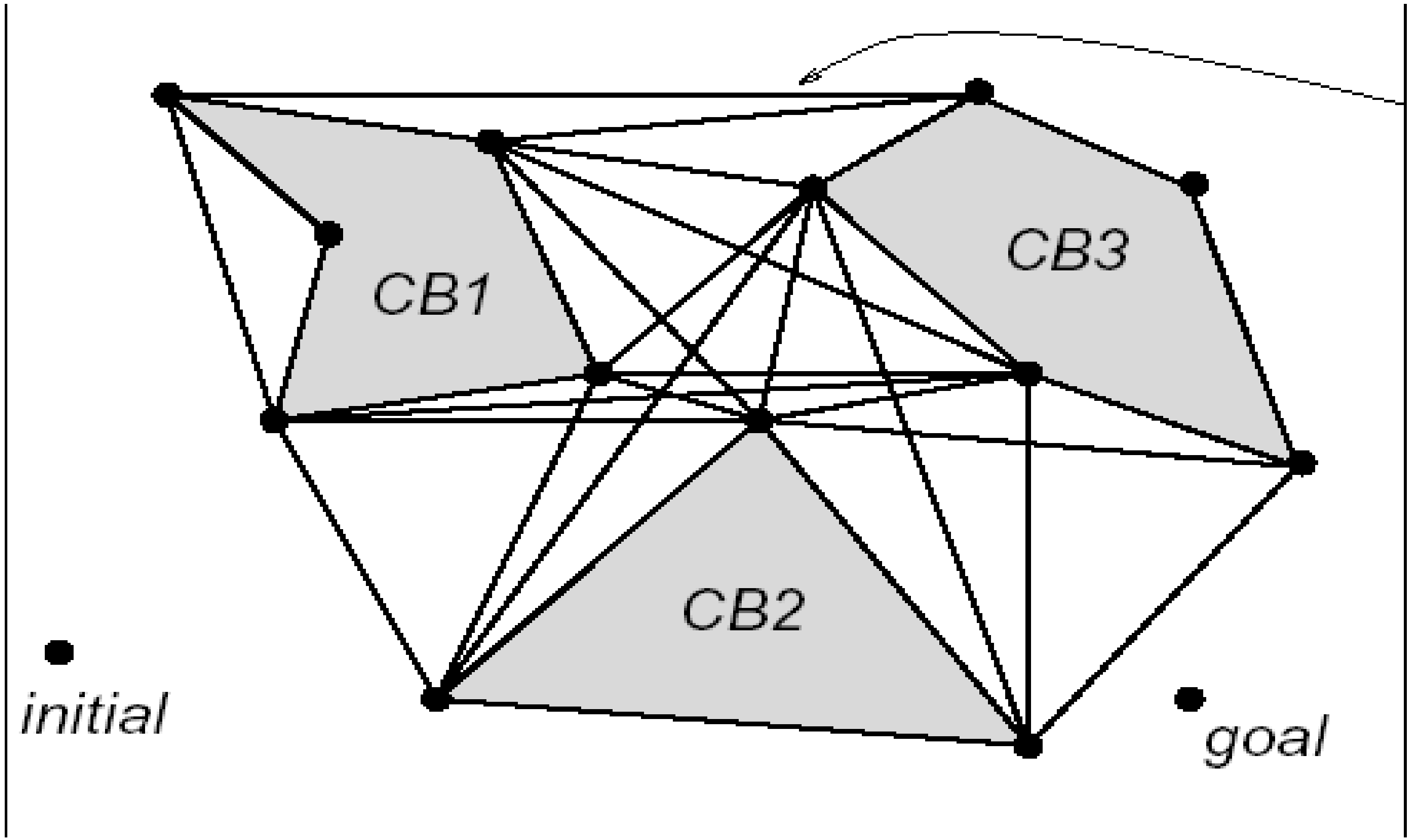
1. build a roadmap in $\mathcal{C}_{free}$ (preprocessing)
   - roadmap nodes are free configurations (or semi-free)
   - two nodes connected by edge if can (easily) move between them
2. connect $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ to roadmap nodes v_{init} and v_{goal} (in same connected component)
3. find a path in the roadmap between v_{init} and v_{goal}
   – directly gives a path in $\mathcal{C}_{free}$

Roadmap Methods

- Properties of a roadmap:
  - Accessibility: there exists a collision-free path from the start to the road map
  - Departability: there exists a collision-free path from the roadmap to the goal.
  - Connectivity: there exists a collision-free path from the start to the goal (on the roadmap).

- Examples of Roadmaps
  - Visibility Graph
  - Generalized Voronoi Graph (GVG)

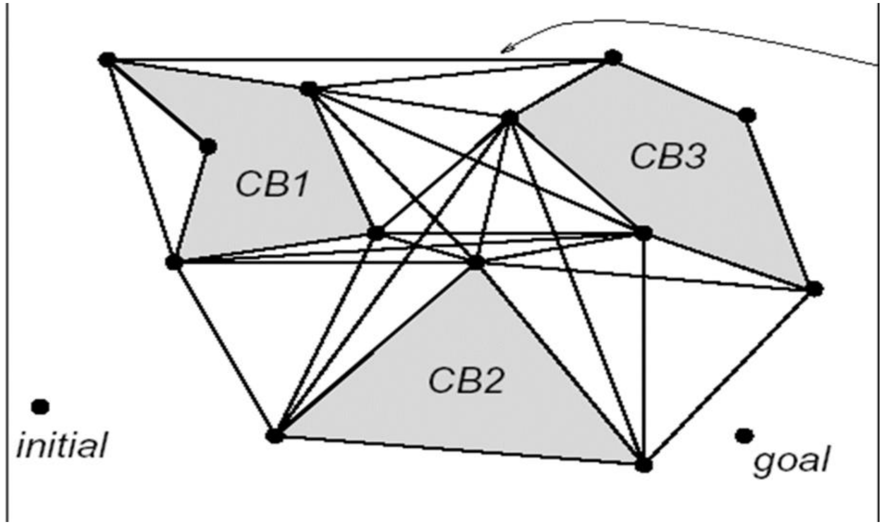
Roadmap: Visibility Graph



Visibility Graph of C-Space

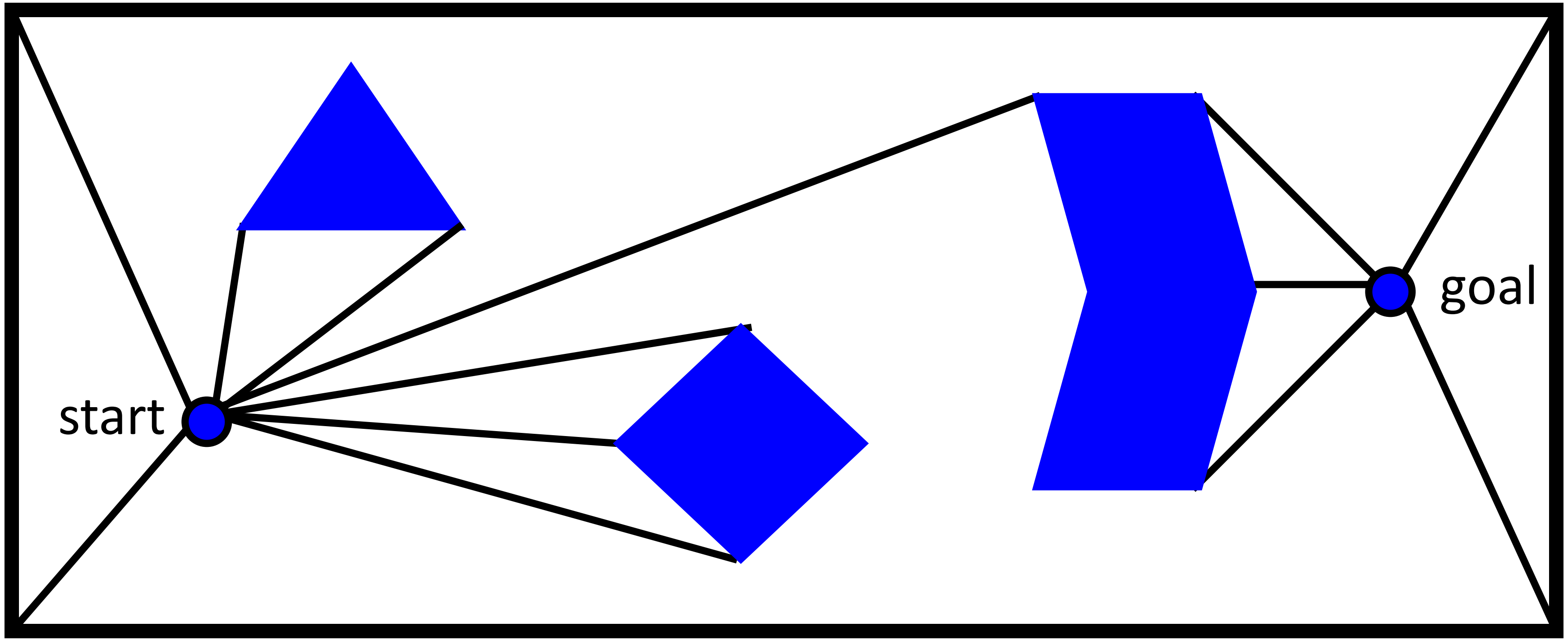
A **visibility graph** of C-space for a given $\mathcal{CB}$ is an undirected graph G where

- nodes in G correspond to vertices of $\mathcal{CB}$
- nodes connected by edge in G if
  - they are connected by an edge in $\mathcal{CB}$, or
  - the straight line segment connecting them lies entirely in $\mathcal{C}_{free}$
- (could add $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ as roadmap nodes)



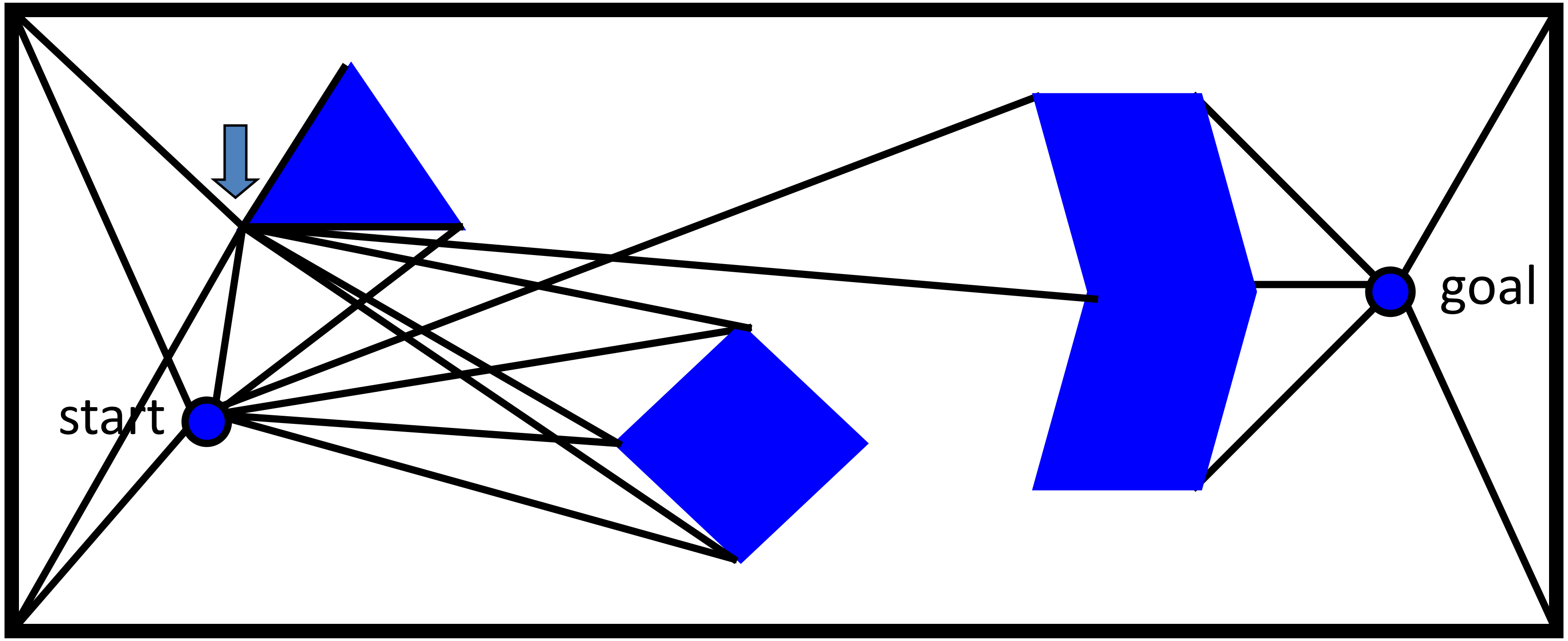
The Visibility Graph in Action (Part 1)

- First, draw lines of sight from the start and goal to all “visible” vertices and corners of the world.



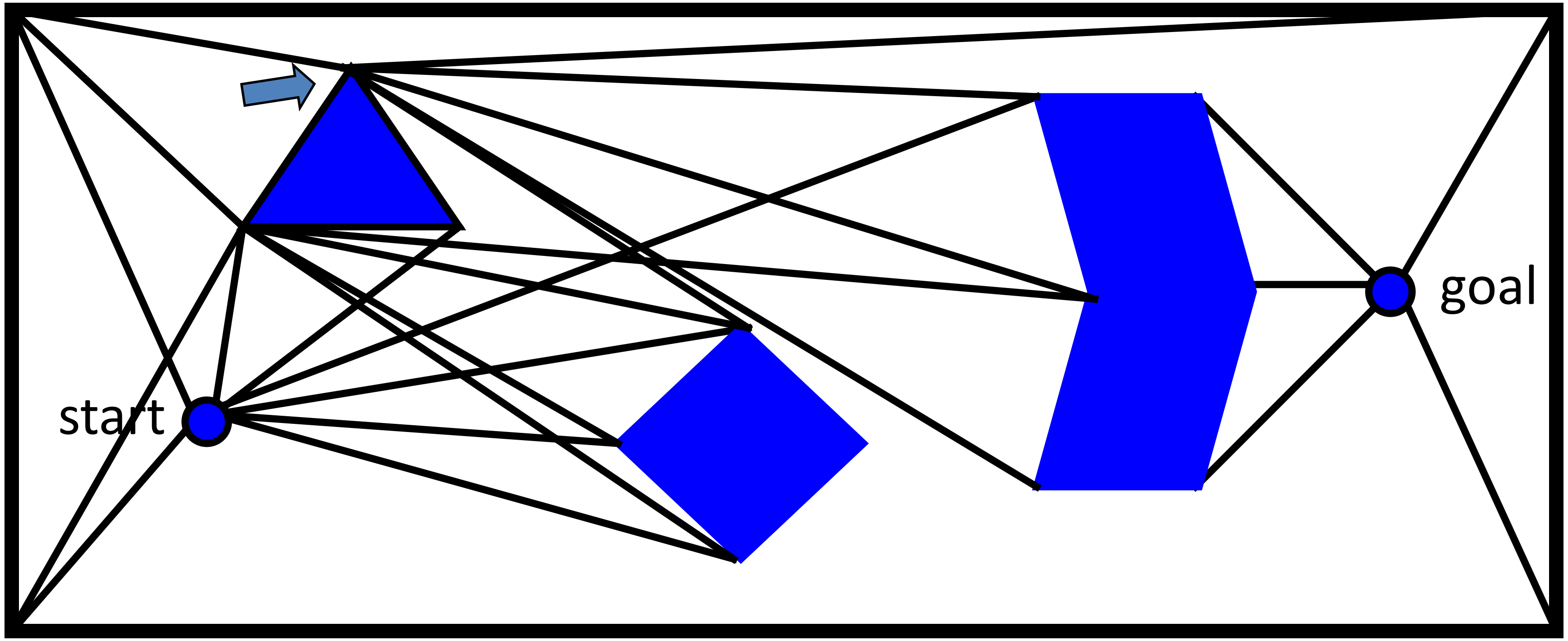
The Visibility Graph in Action (Part 2)

- Second, draw lines of sight from every vertex of every obstacle like before. Remember lines along edges are also lines of sight.



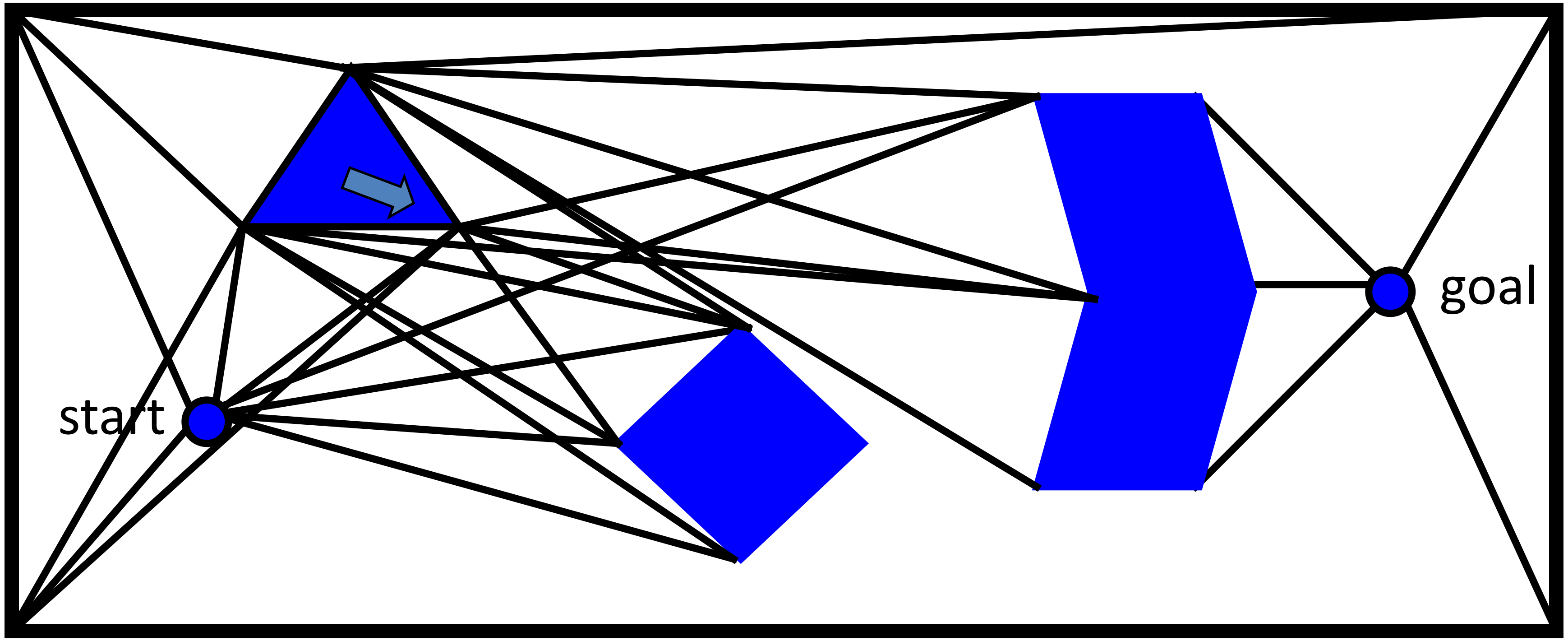
The Visibility Graph in Action (Part 3)

- Second, draw lines of sight from every vertex of every obstacle like before. Remember lines along edges are also lines of sight.



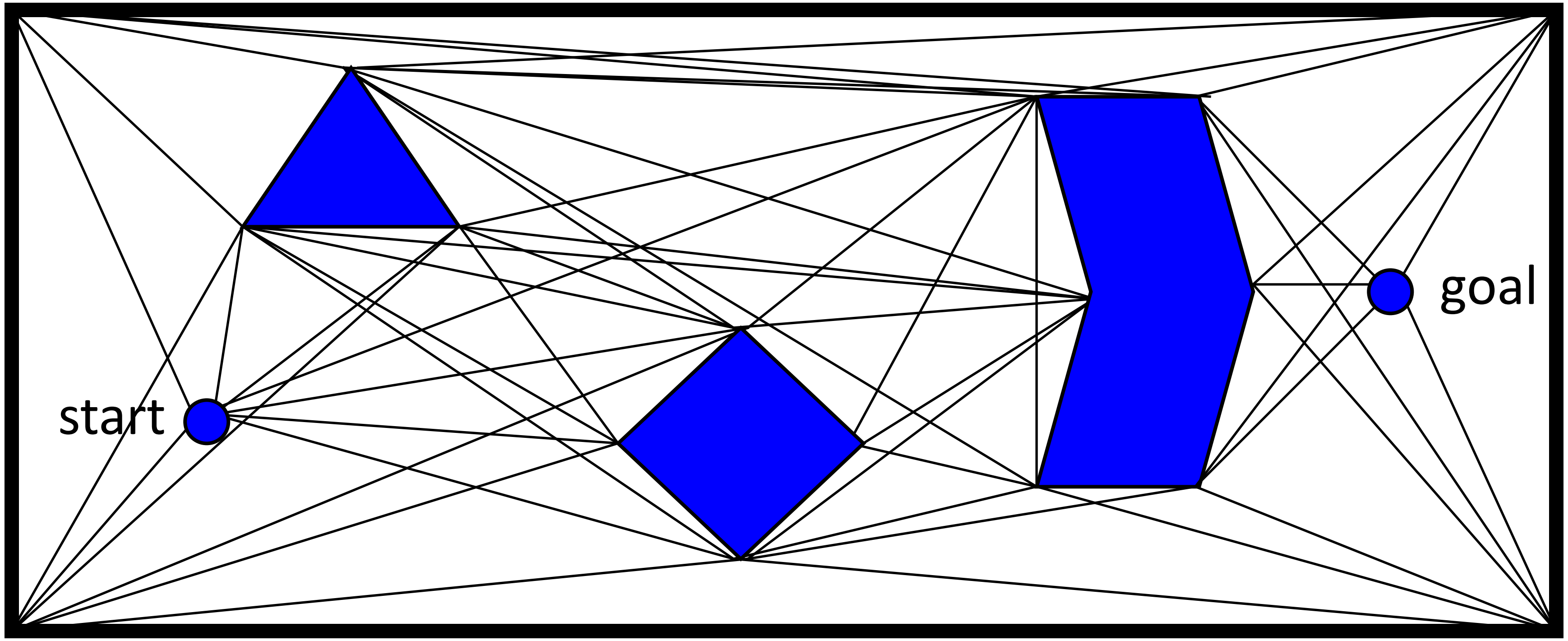
The Visibility Graph in Action (Part 4)

- Second, draw lines of sight from every vertex of every obstacle like before. Remember lines along edges are also lines of sight.



The Visibility Graph (Done)

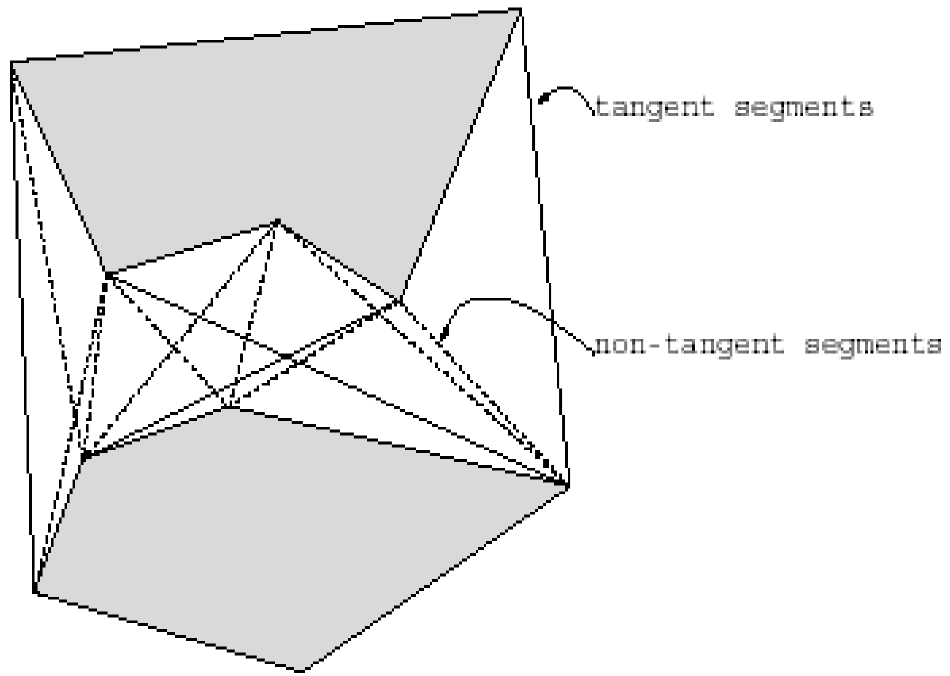
- Repeat until you're done.



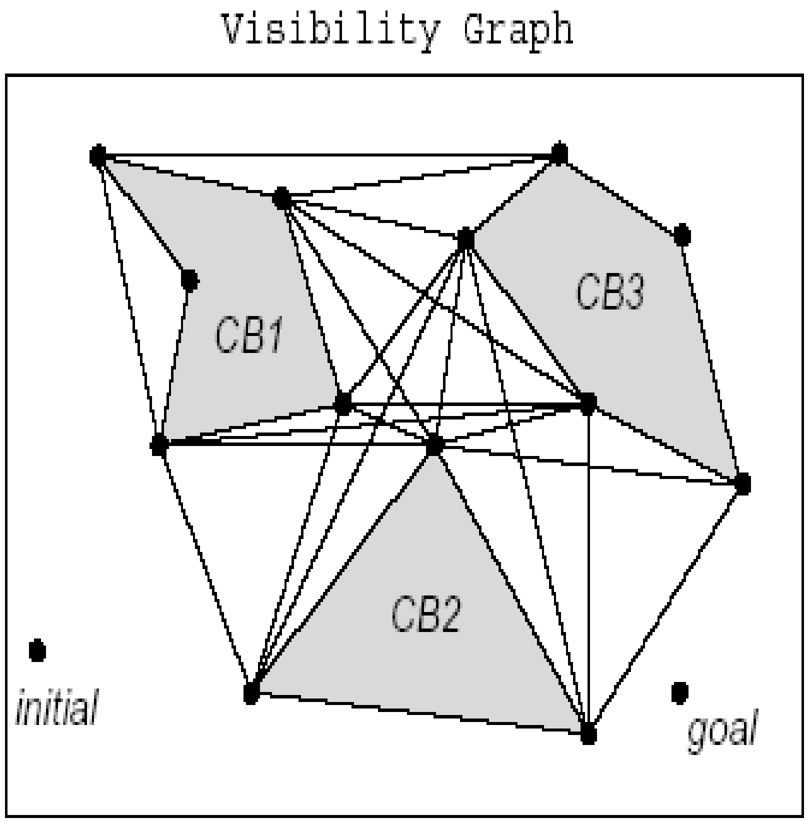
Reduced Visibility Graphs

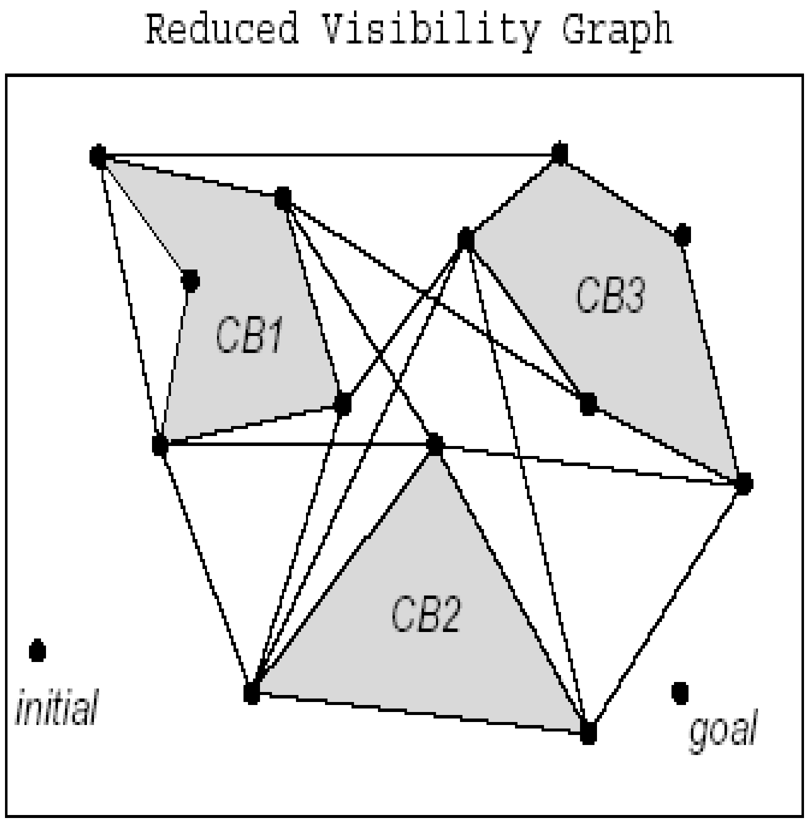
- *Idea:* we don't really need all the edges in the visibility graph (even if we want shortest paths)

- *Definition:* Let L be the line passing through an edge (x,y) in the visibility graph G. The segment (x,y) is a tangent segment iff L is tangent to CB at both x and y.



Reduced Visibility Graphs

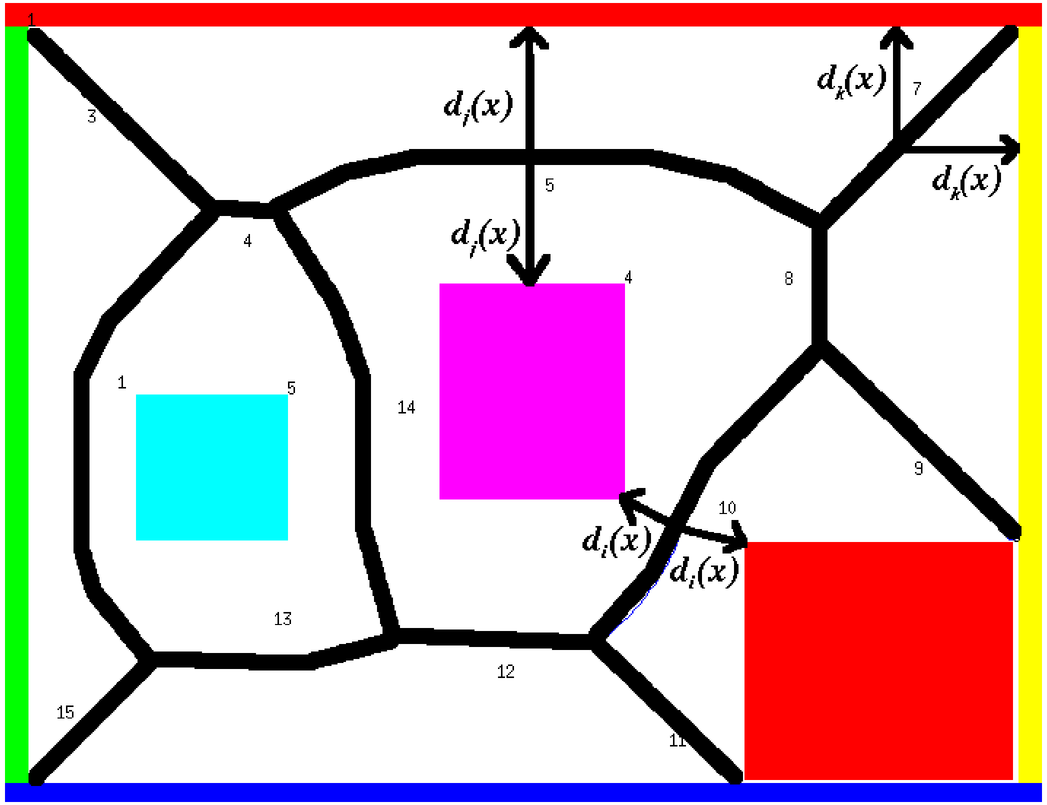




MAP-BASED APPROACHES: THE RETRACTION APPROACH

Retraction Example: Generalized Voronoi Diagrams

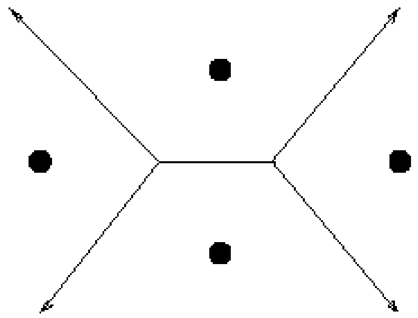
- A GVG is formed by paths equidistant from the two closest objects



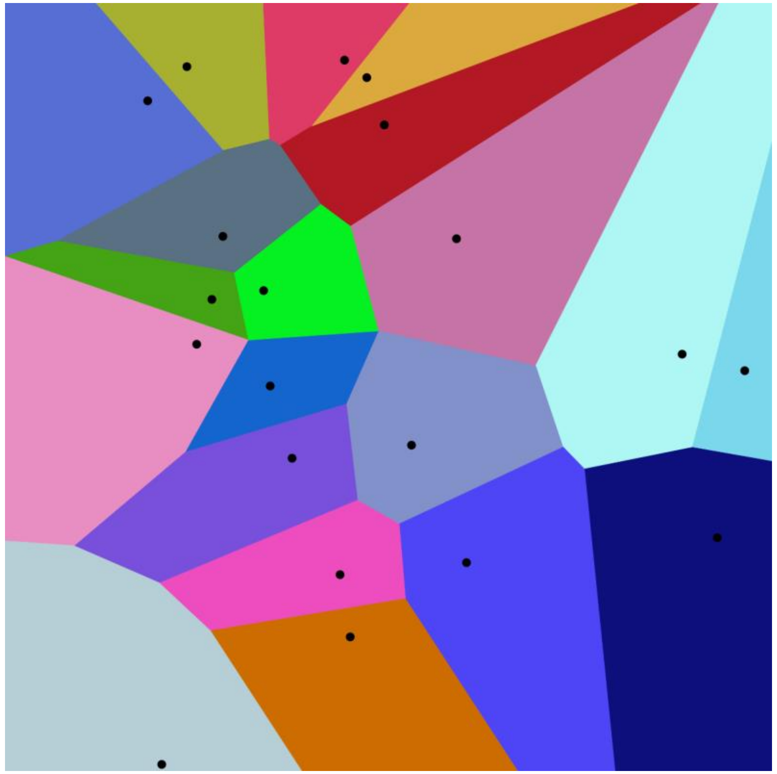
- This generates a very safe roadmap which avoids obstacles as much as possible

Retraction Example: Generalized Voronoi Diagrams

Example: **Voronoi Diagram for point sets** (original)



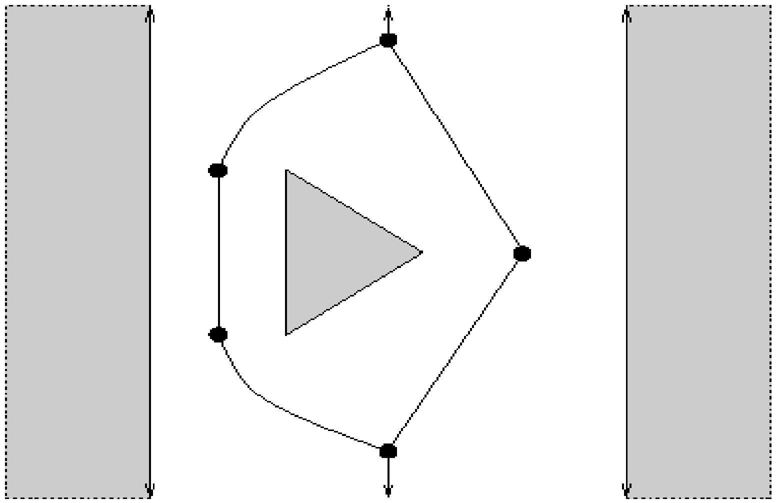
- Voronoi diagram of point set X consists of straight line segments
- constructed by
  - computing lines bisecting each pair of points and their intersections
  - computing intersections of these lines
  - keeping segments with more than one nearest neighbor



Generalized Voronoi Diagrams

When $\mathcal{C} = \mathbb{R}^2$ and polygonal $\mathcal{CB}$, $Vor(\mathcal{C}_{free})$ consists of a finite collection of straight line segments and *parabolic curve segments* (called **arcs**)

- **straight arcs** are defined by two vertices or two edges of $\mathcal{CB}$, i.e., the set of points equally close to two points (or two line segments) is a line
- **parabolic arcs** are defined by one vertex and one edge of $\mathcal{CB}$, i.e., the set of points equally close to a point and a line is a parabola

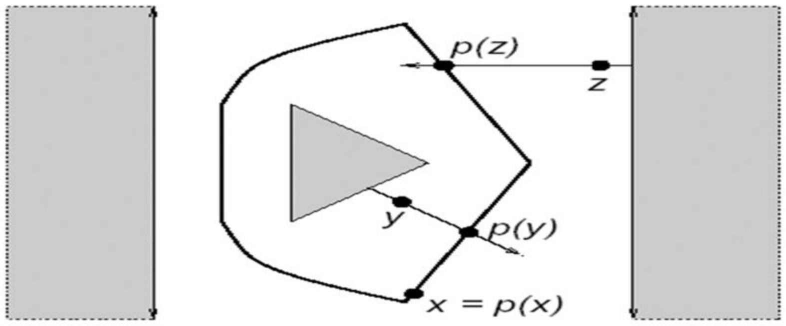


Generalized Voronoi Diagrams

To use $Vor(\mathcal{C}_{free})$ as our roadmap R, we need to define the retraction

$$\rho : \mathcal{C}_{free} \rightarrow Vor(\mathcal{C}_{free})$$

<u>Case 1: $\mathbf{q} \in Vor(\mathcal{C}_{free})$</u>: $\rho(\mathbf{q}) = \mathbf{q}$



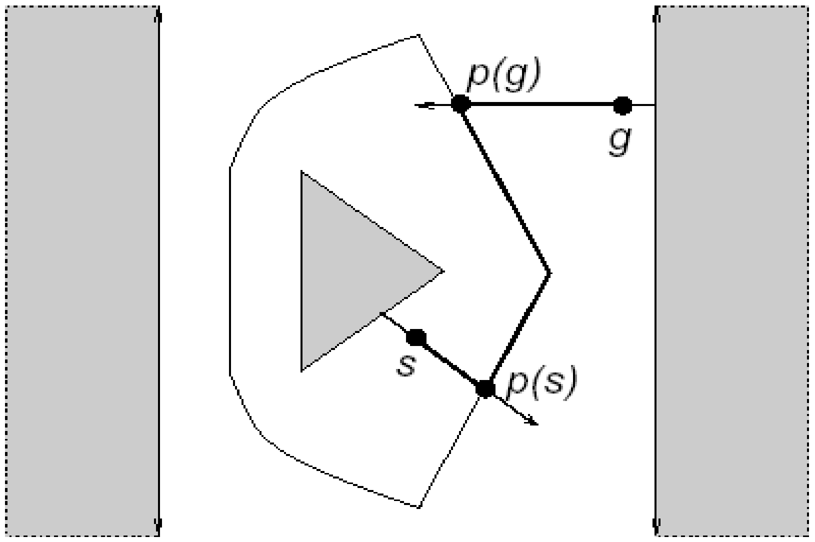
<u>Case 2: $\mathbf{q} \notin Vor(\mathcal{C}_{free})$</u>

- let $\mathbf{p}$ be the closest point of the boundary of $\mathcal{C}_{free}$ to $\mathbf{q}$
- let L be the ray from $\mathbf{p}$ passing through $\mathbf{q}$ (L follows the steepest ascent of the $clearance()$ function from $\mathbf{p}$)
- define $\rho(\mathbf{q})$ to be the intersection of L with $Vor(\mathcal{C}_{free})$

Generalized Voronoi Diagrams

To find a path:

1. compute $Vor(\mathcal{C}_{free})$
2. find paths from $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ to $\rho(\mathbf{q}_{init})$ and $\rho(\mathbf{q}_{goal})$, respectively
3. search $Vor(\mathcal{C}_{free})$ for a set of arcs connecting $\rho(\mathbf{q}_{init})$ and $\rho(\mathbf{q}_{goal})$



MAP-BASED APPROACHES:
CELL DECOMPOSITION METHODS

Cell Decomposition Methods

PREPROCESSING:

- represent $\mathcal{C}_{free}$ as a collection of *cells* (connected regions of $\mathcal{C}_{free}$)
  $\Longrightarrow$ planning between configurations in the same cell should be 'easy'
- build *connectivity graph* representing adjacency relations between cells
  $\Longrightarrow$ cells adjacent if can move directly between them

QUERY PROCESSING:

1. locate cells k_{init} and k_{goal} containing start and goal configurations
2. search the connectivity graph for a 'channel' or sequence of adjacent cells connecting k_{init} and k_{goal}
3. find a path that is contained in the channel of cells

Cell Decomposition Methods

Two major variants of methods:

- *exact cell decomposition*:
  - set of cells exactly covers $\mathcal{C}_{free}$
  - complicated cells with irregular boundaries (contact constraints)
  - harder to compute
- *approximate cell decomposition*:
  - set of cells approximately covers $\mathcal{C}_{free}$
  - simpler cells with more regular boundaries
  - easier to compute

Exact Cell Decomposition Method

Idea: decompose $\mathcal{C}_{free}$ into a collection $\mathcal{K}$ of non-overlapping *cells* such that the union of all the cells *exactly* equals the free C-space, i.e., $\mathcal{C}_{free} = \cup_{k \in \mathcal{K}} k$

<u>Cell Characteristics:</u>

- geometry of cells should be simple so that it is easy to compute a path between any two configurations in a cell
- it should be pretty easy to test the adjacency of two cells, i.e., whether they share a boundary

Exact Cell Decomposition Method

Definitions:

A **convex polygonal decomposition** $\mathcal{K}$ of $\mathcal{C}_{free}$ is a finite collection of convex polygons, called **cells**, such that the interiors of any two cells do not intersect and the union of all cells is $\mathcal{C}_{free}$.

Two cells $k, k' \in \mathcal{K}$ are **adjacent** iff $k \cap k'$ is a line segment of non-zero length (i.e., not a single point)

The **connectivity graph** associated with a convex polygonal decomposition $\mathcal{K}$ of $\mathcal{C}_{free}$ is an undirected graph G where

- nodes in G correspond to cells in $\mathcal{K}$
- nodes connected by edge in G iff corresponding cells adjacent in $\mathcal{K}$

Path Planning with a Convex Polygonal Decomposition

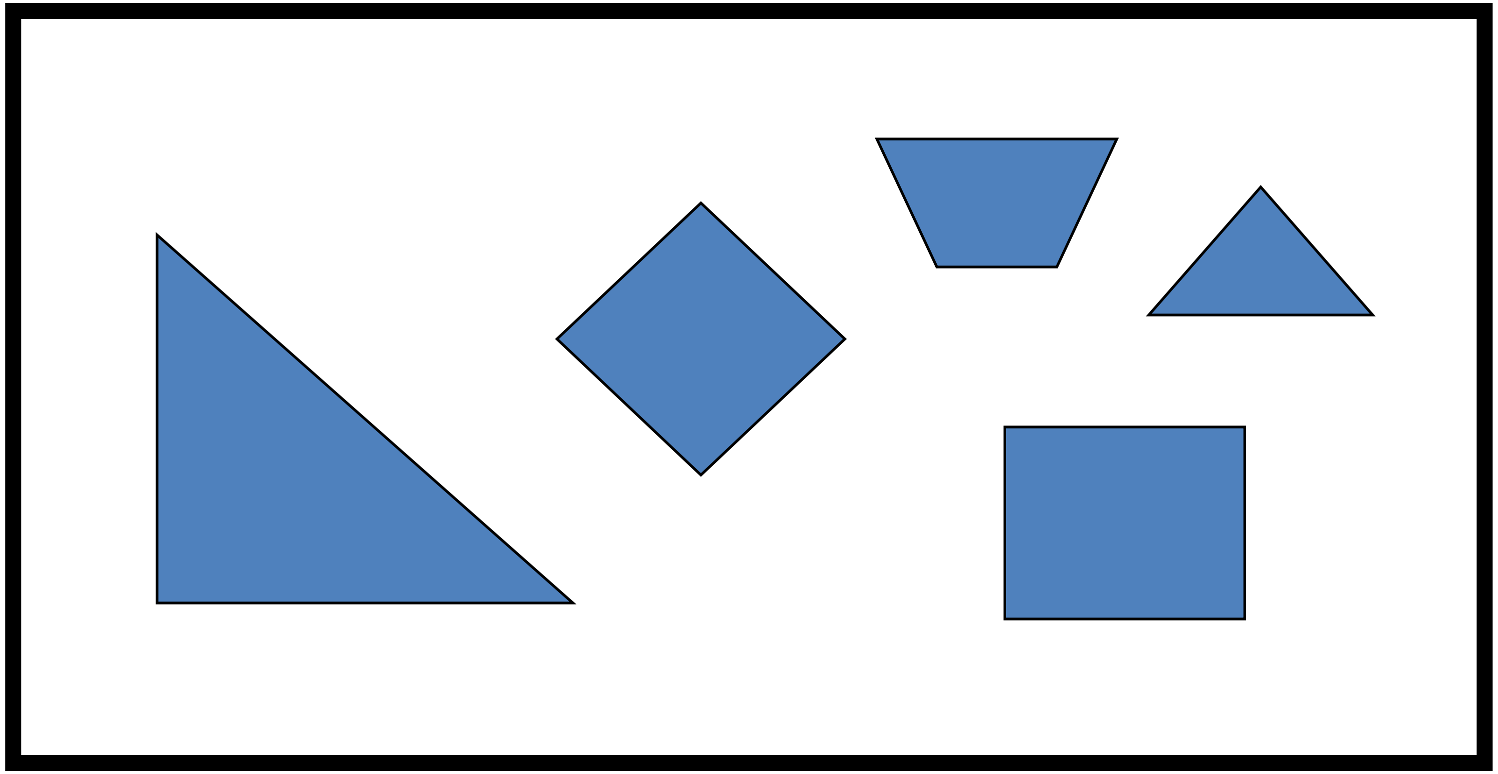
input: configurations $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$, and $\mathcal{CB}$ which is a polygonal region

output: a path in $\mathcal{C}_{free}$ connecting $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$

1. Build $\mathcal{K}$, the convex polygonal decomposition of $\mathcal{CB}$
2. Construct the connectivity graph G of $\mathcal{K}$
3. locate the cells k_{init} and k_{goal} in $\mathcal{K}$ containing $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$
4. find a path in G between the nodes corresponding to k_{init} and k_{goal}
   – corresponds to a sequence of cells forming a **channel** in $\mathcal{C}_{free}$
5. find a free path from $\mathbf{q}_{init}$ to $\mathbf{q}_{goal}$ in the channel

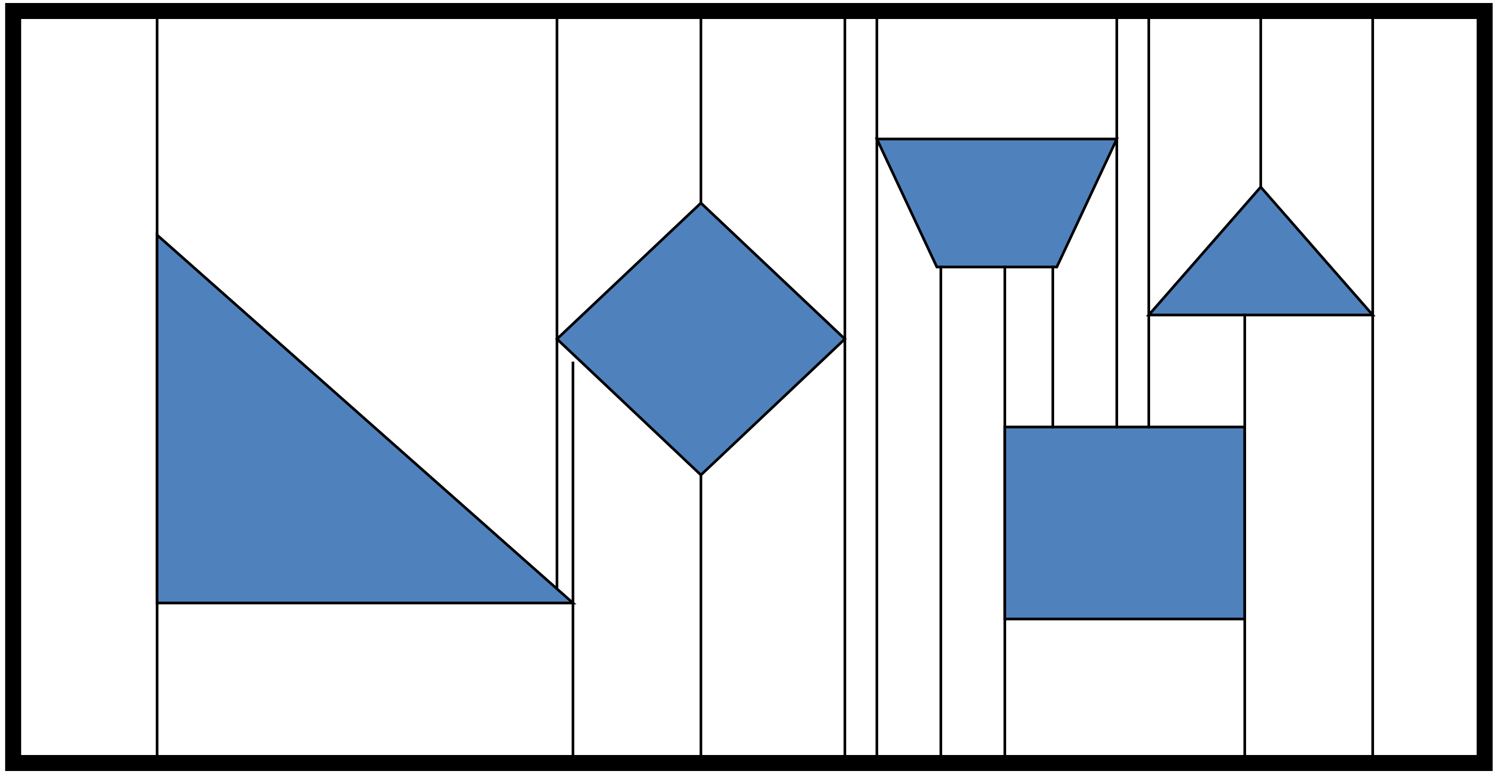
Exact Cell Decompositions: Trapezoidal Decomposition

- A way to divide the world into smaller regions
- Assume a polygonal world



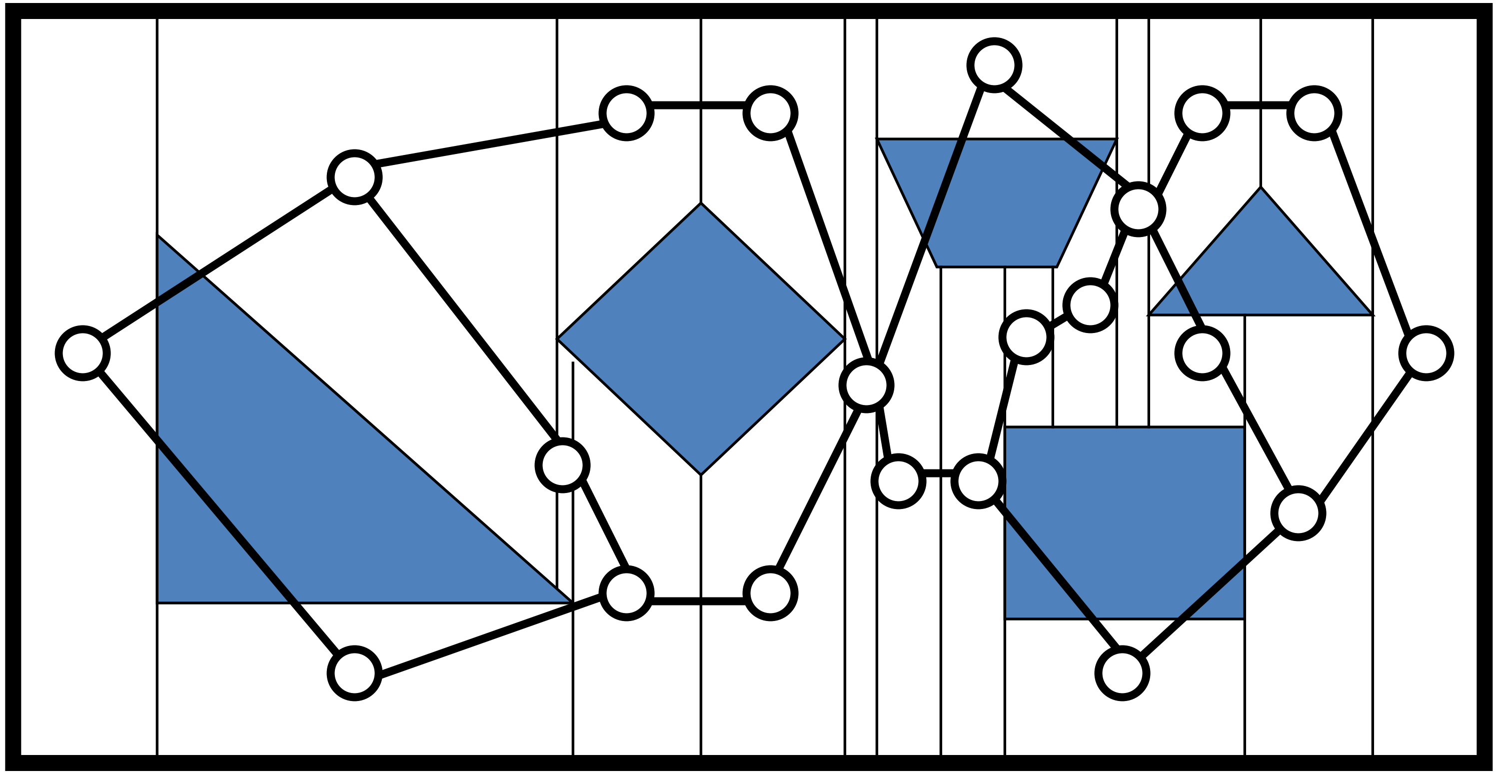
Exact Cell Decompositions: Trapezoidal Decomposition

Basic Idea: at every vertex of $\mathcal{CB}$, extend a vertical line up and down in $\mathcal{C}_{free}$ until it touches a C-obstacle or the boundary of $\mathcal{C}_{free}$



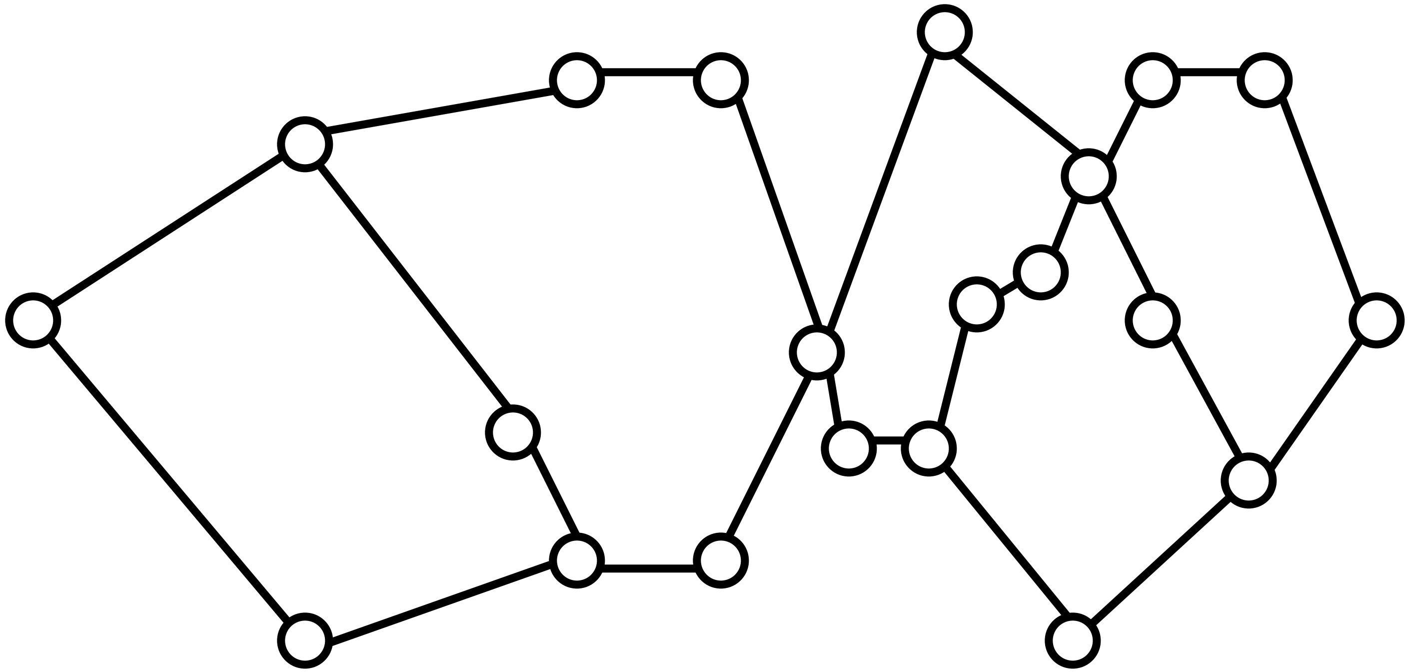
Applications: Coverage

- By reducing the world to cells, we've essentially abstracted the world to a graph.



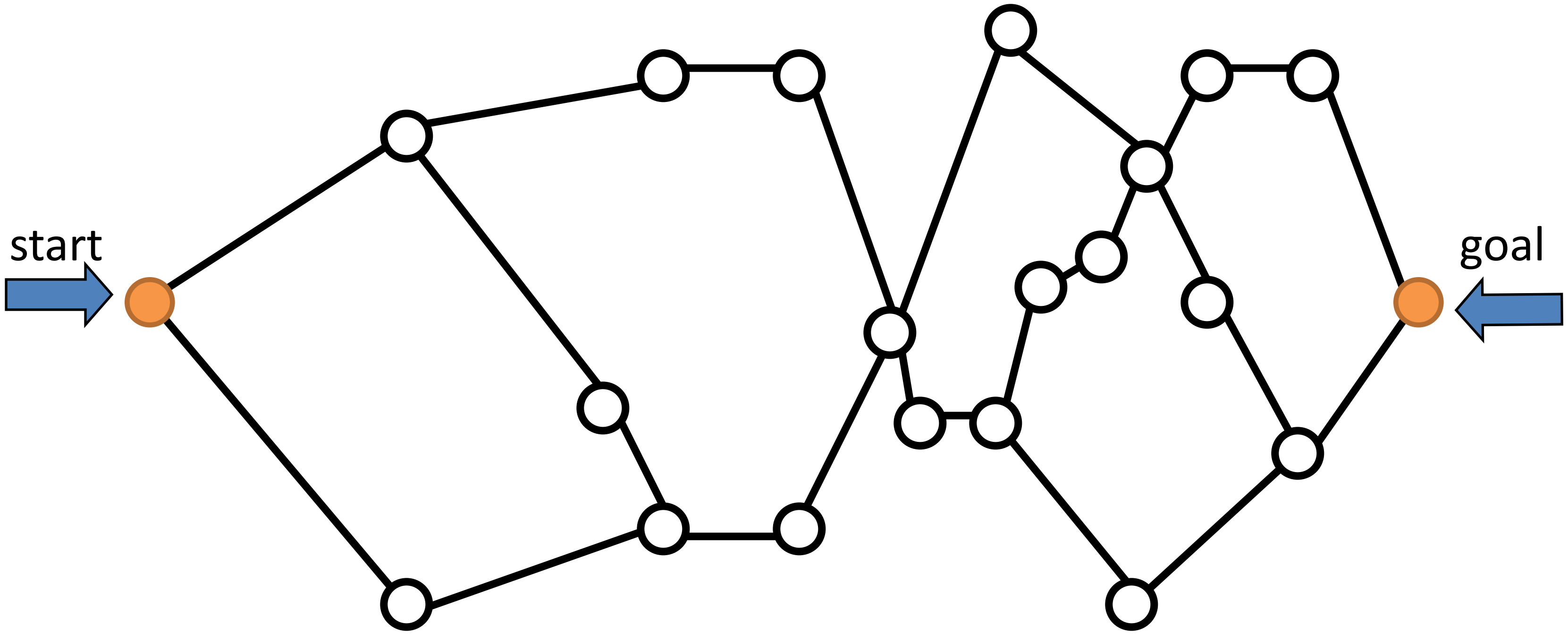
Find a path

- By reducing the world to cells, we’ve essentially abstracted the world to a graph.



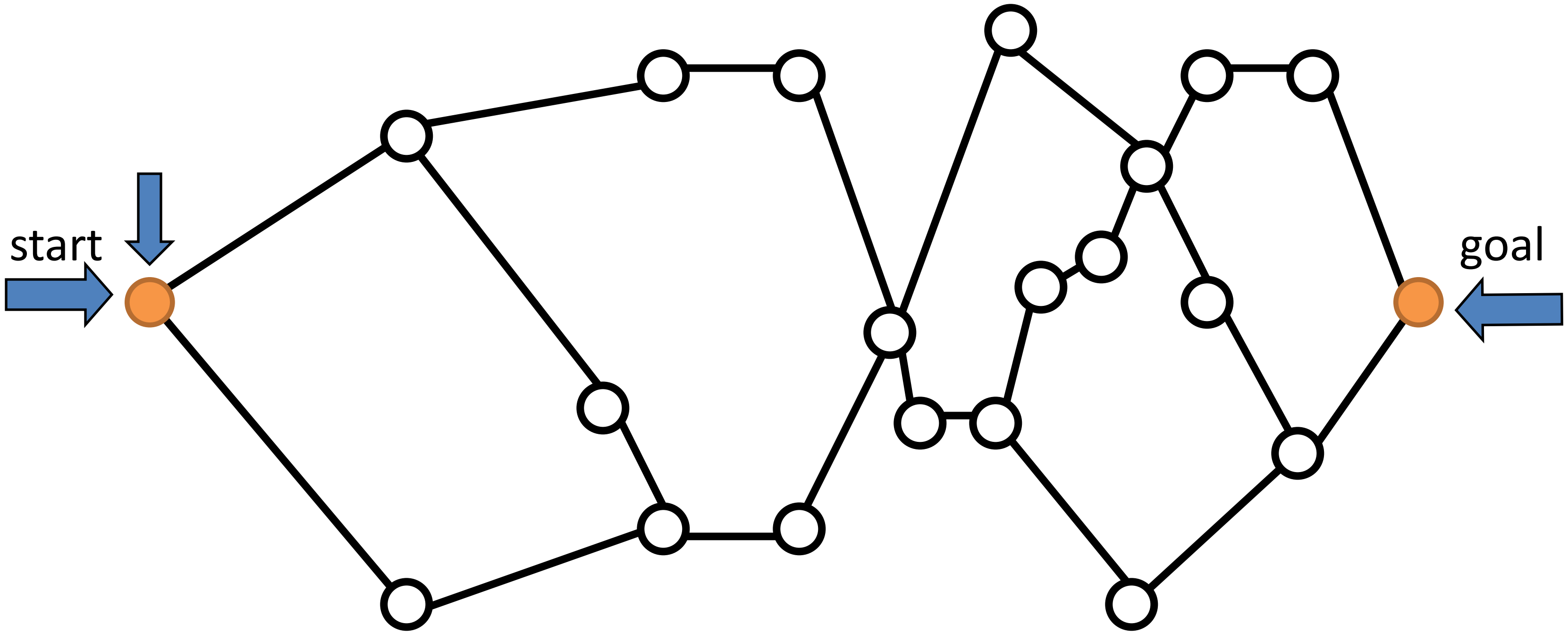
Find a path

- With an adjacency graph, a path from start to goal can be found by simple traversal



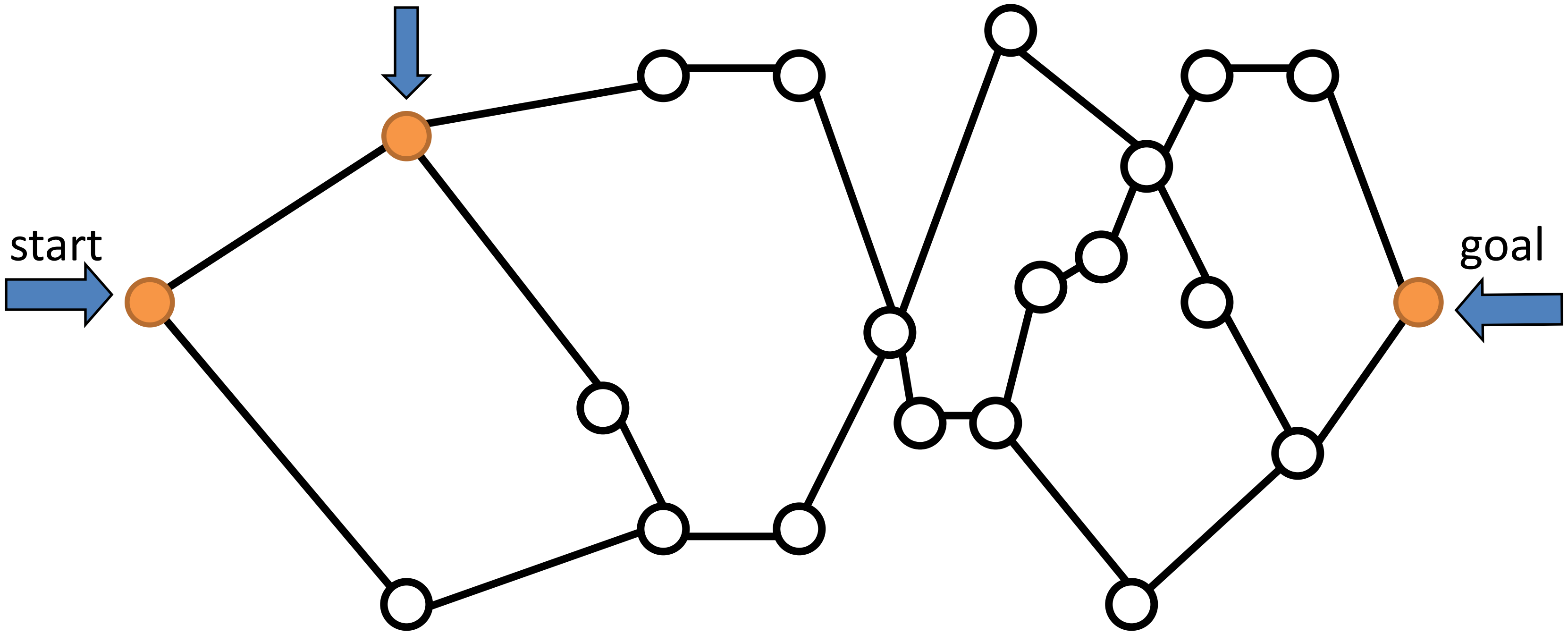
Find a path

- With an adjacency graph, a path from start to goal can be found by simple traversal



Find a path

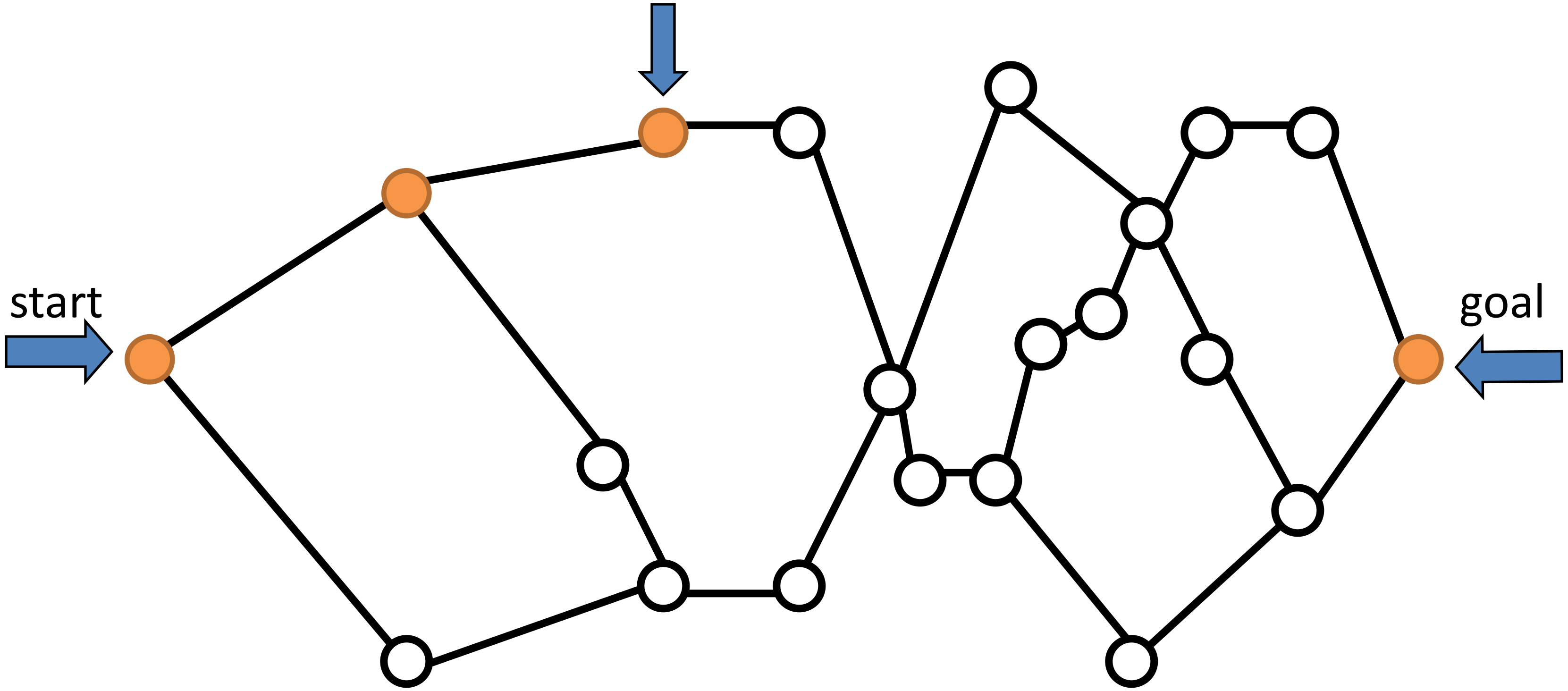
- With an adjacency graph, a path from start to goal can be found by simple traversal



HMI
HUMAN MACHINE INTERACTION LABORATORY

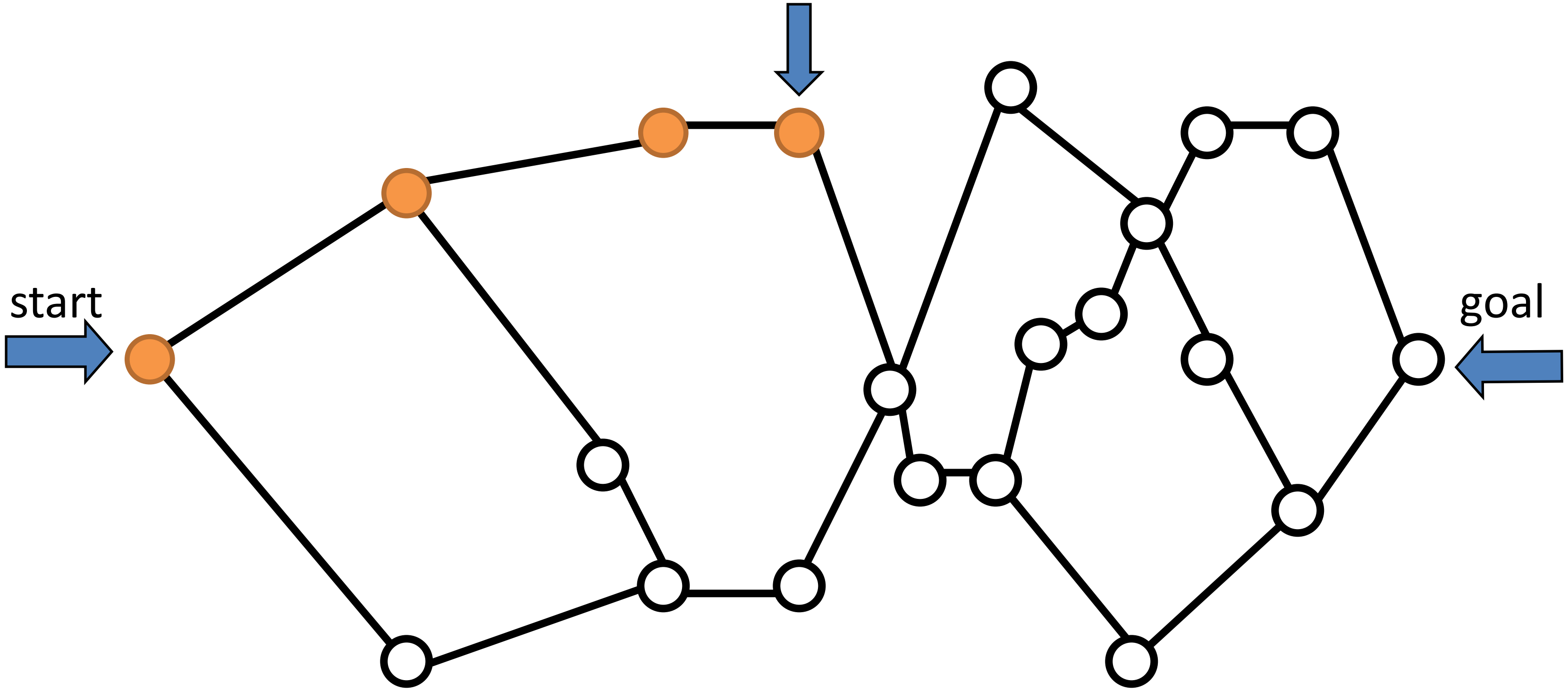
Find a path

- With an adjacency graph, a path from start to goal can be found by simple traversal



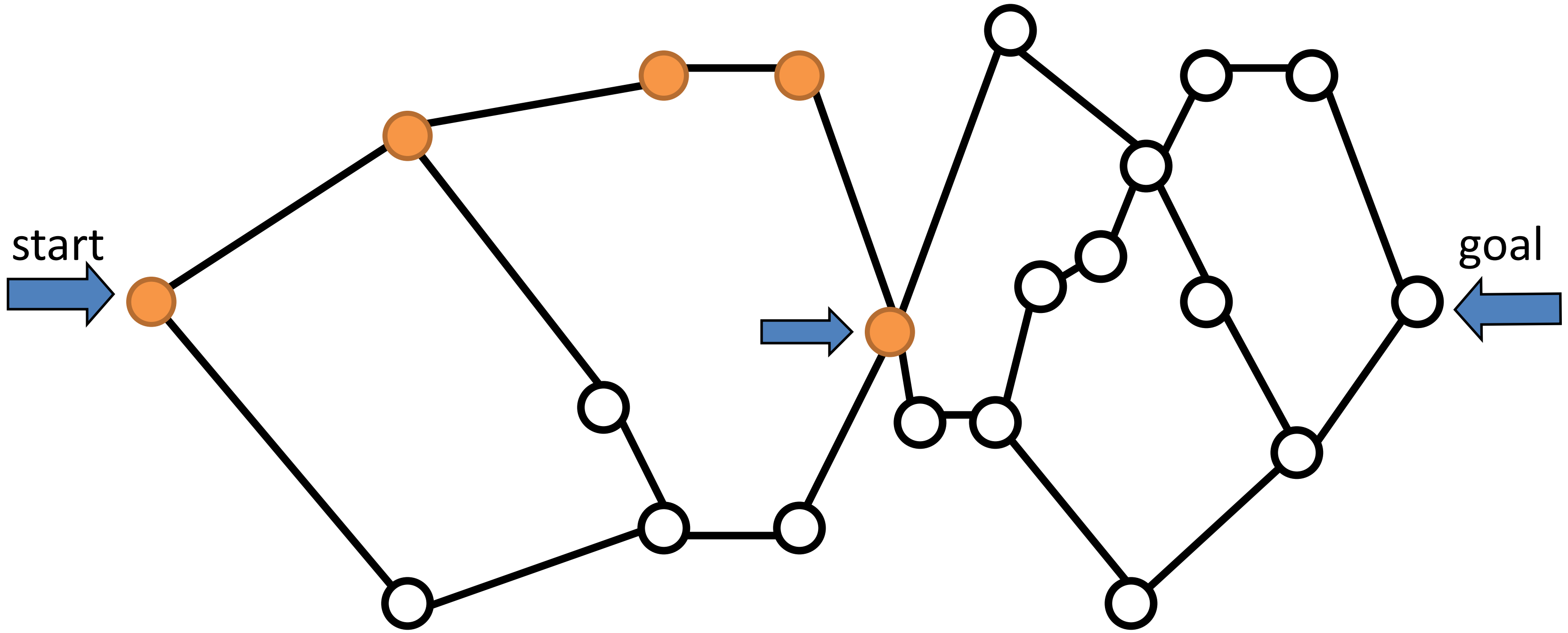
Find a path

- With an adjacency graph, a path from start to goal can be found by simple traversal



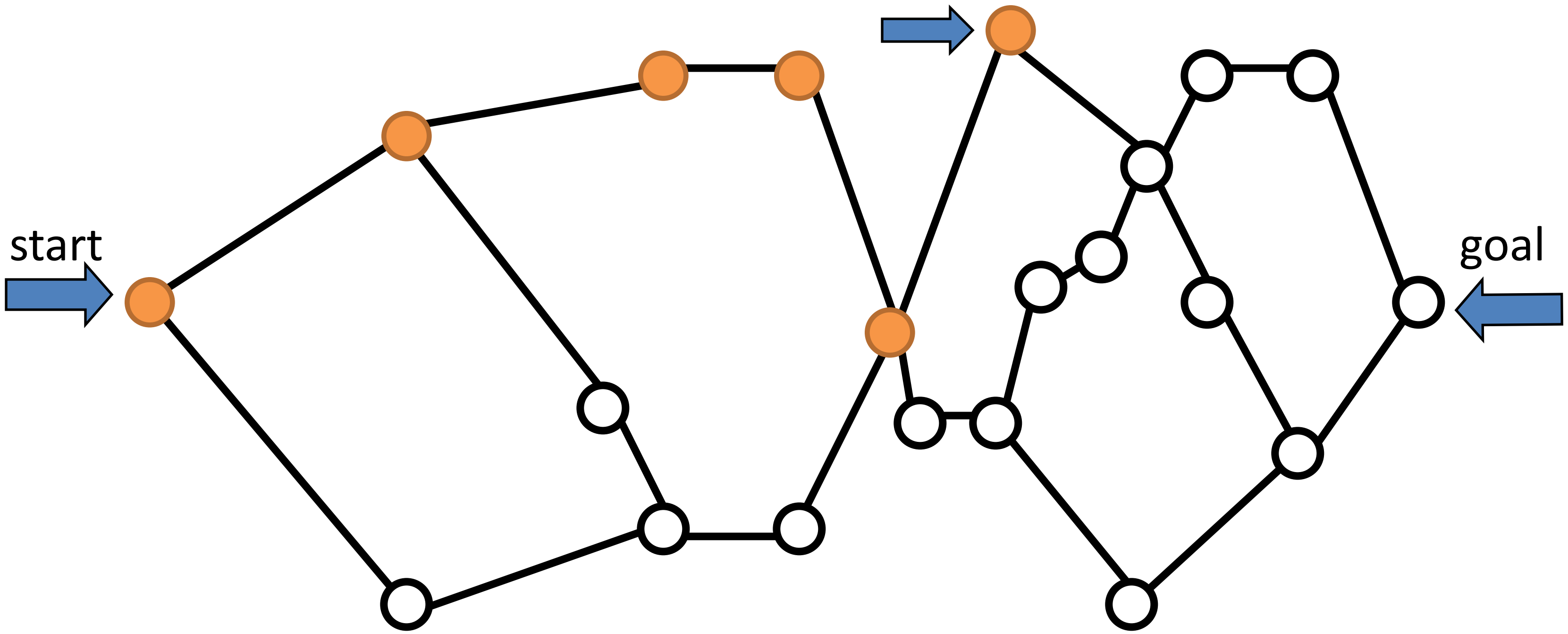
Find a path

- With an adjacency graph, a path from start to goal can be found by simple traversal



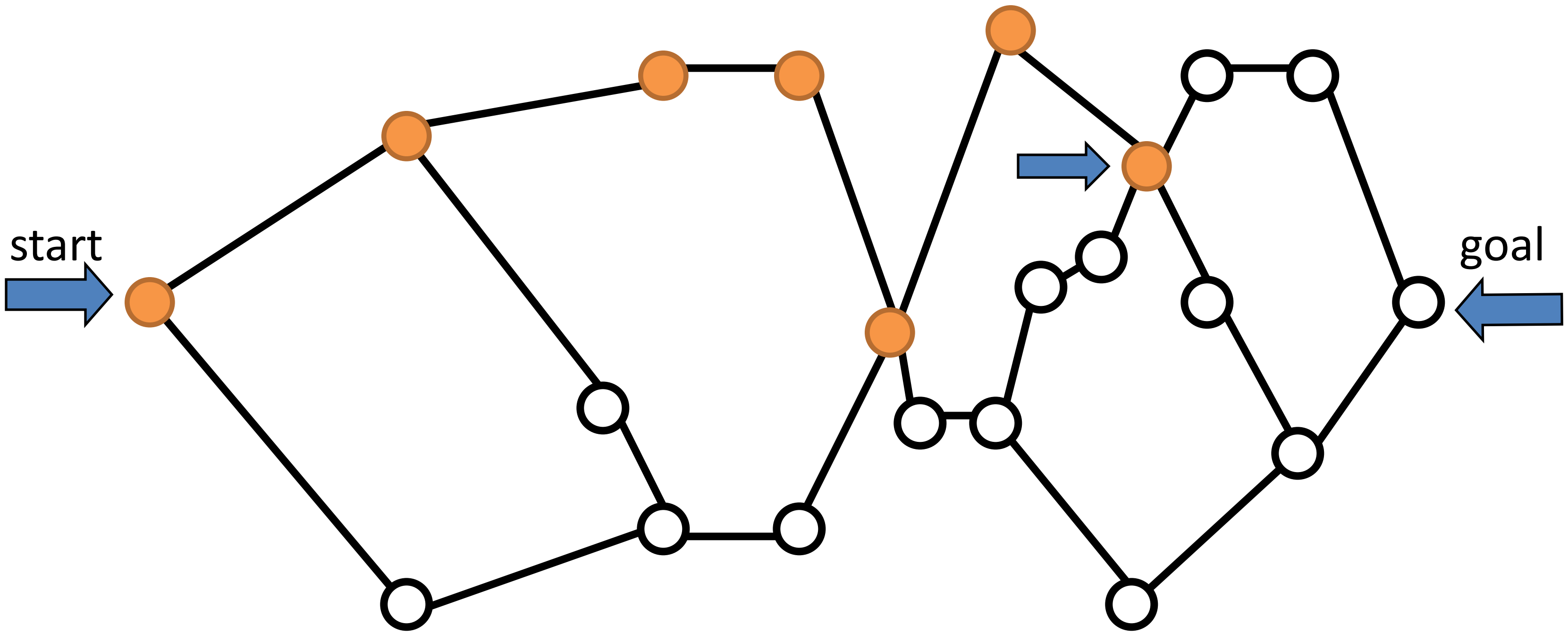
Find a path

- With an adjacency graph, a path from start to goal can be found by simple traversal



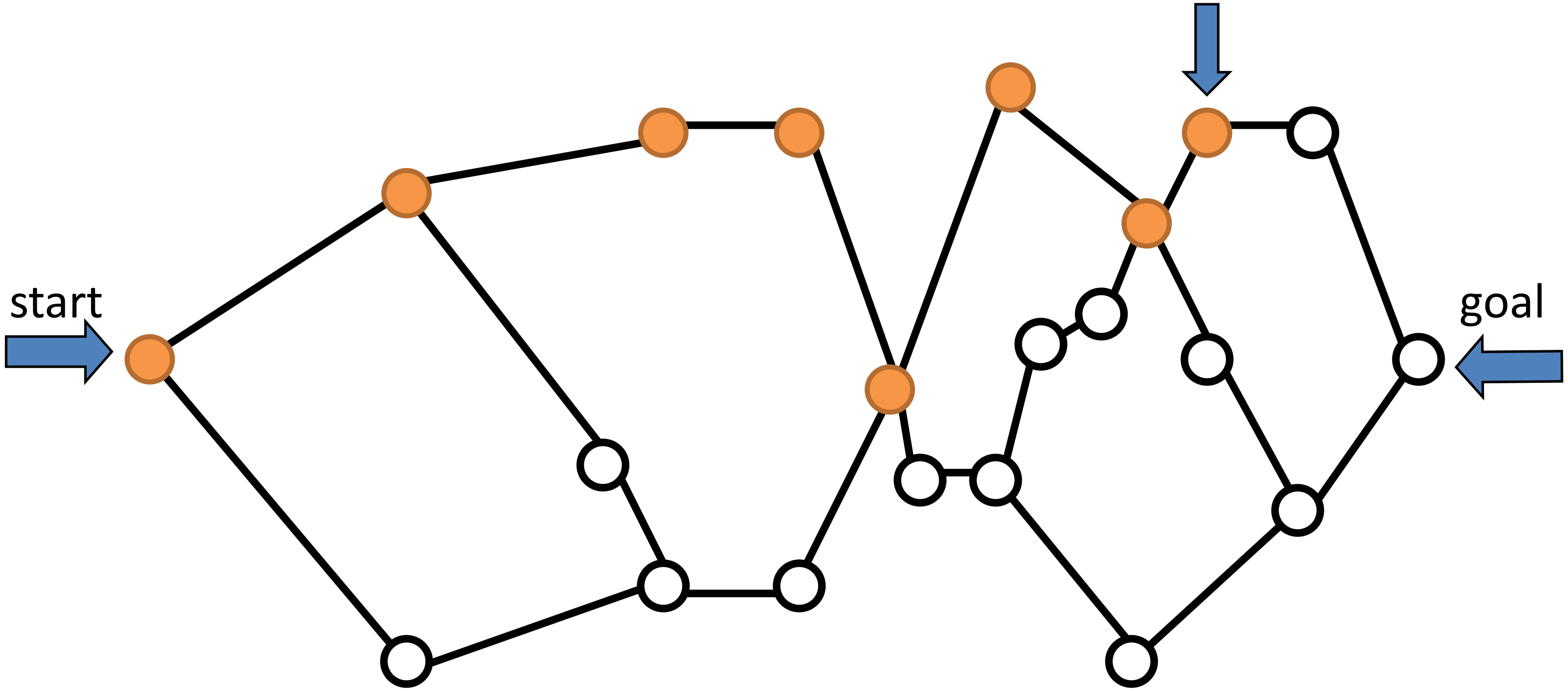
Find a path

- With an adjacency graph, a path from start to goal can be found by simple traversal



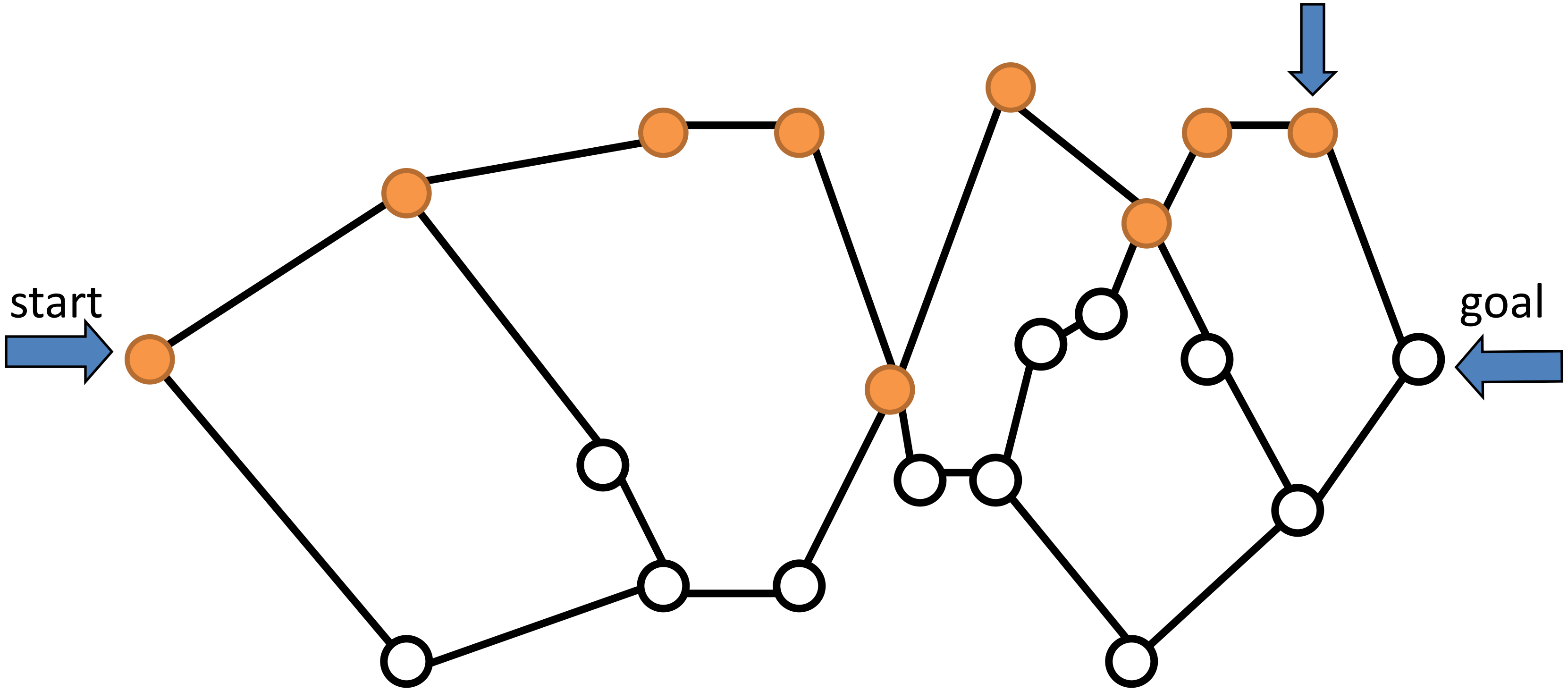
Find a path

- With an adjacency graph, a path from start to goal can be found by simple traversal



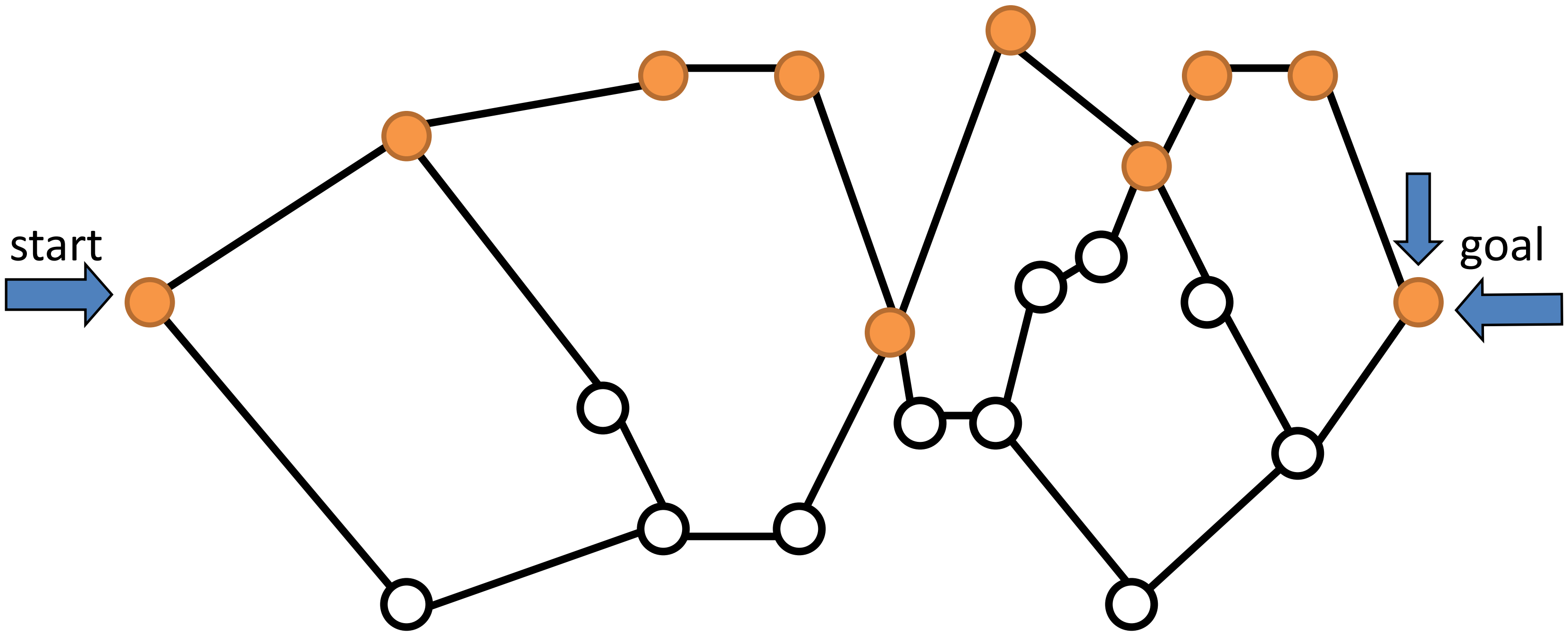
Find a path

- With an adjacency graph, a path from start to goal can be found by simple traversal



Find a path

- With an adjacency graph, a path from start to goal can be found by simple traversal



Extensions to the Basic Problem

- movable obstacles
- moving obstacles
- multiple robots
- incomplete knowledge/uncertainty in geometry, sensing, etc.