Motion Planning, Part IV
Graph Search Part II

Howie Choset

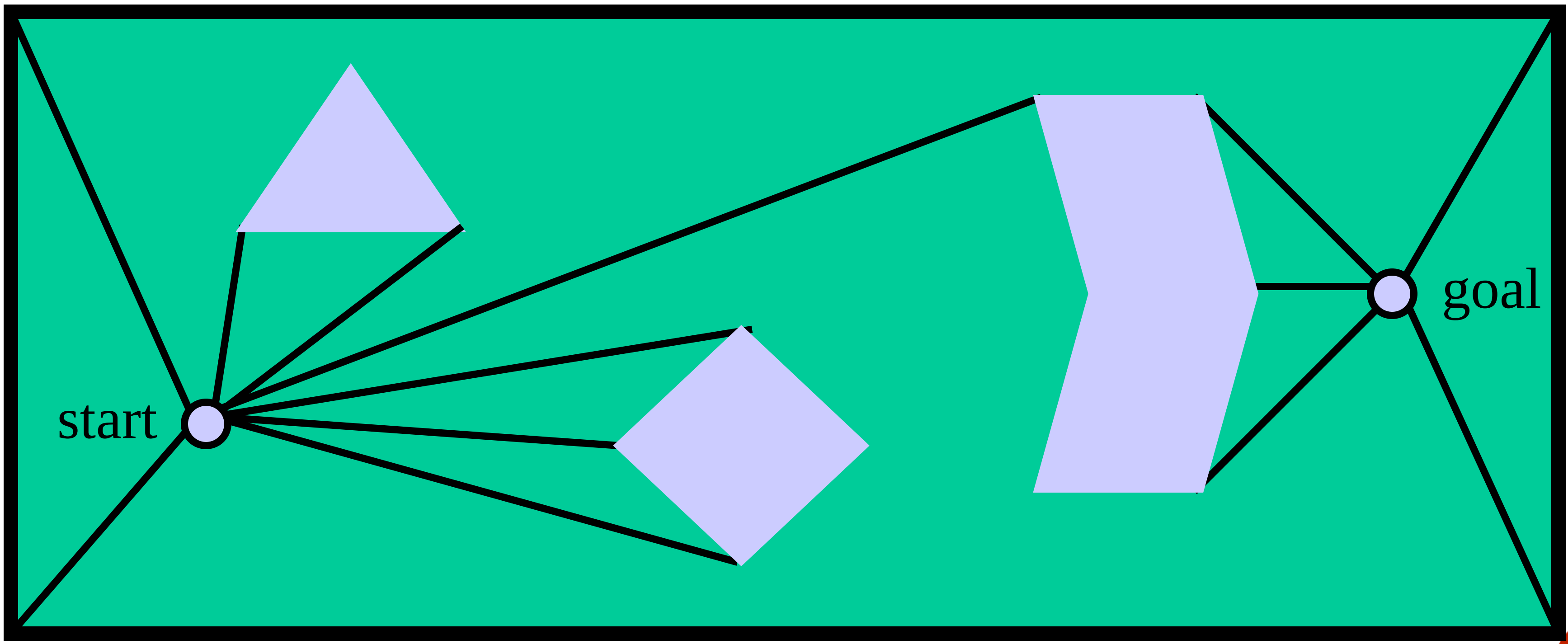
Map-Based Approaches: Roadmap Theory

- 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).

- **a roadmap exists ⇔ a path exists**
- **Examples of Roadmaps**
  - **Generalized Voronoi Graph (GVG)**
  - **Visibility Graph**

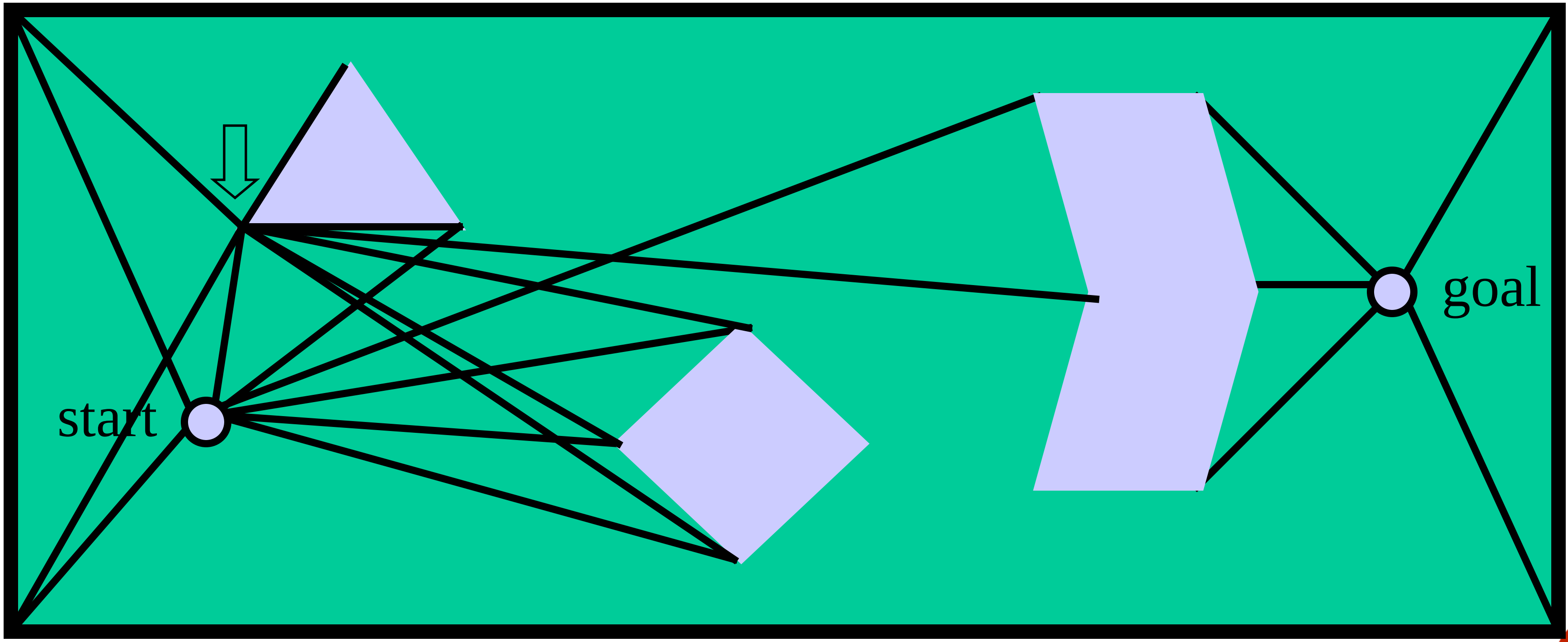
The Visibility Graph in Action (Part 1)

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



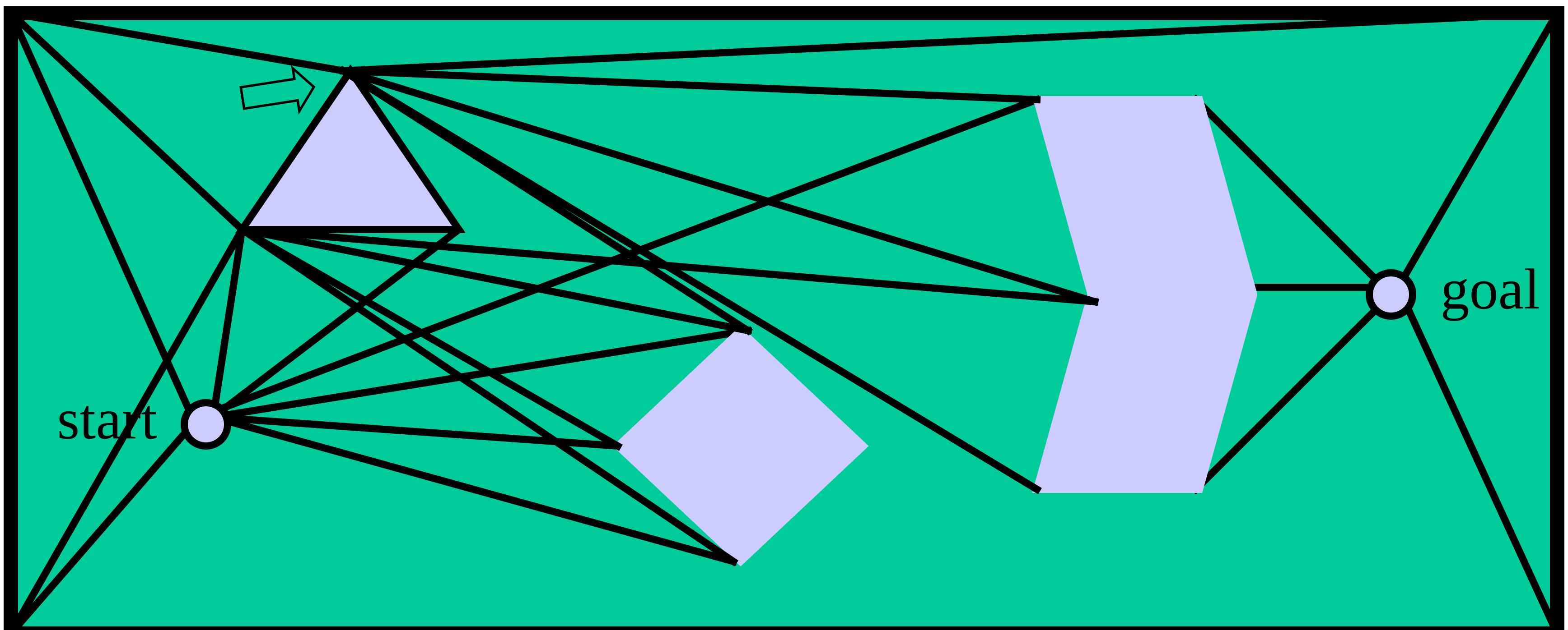
The Visibility Graph in Action (Part 2)

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



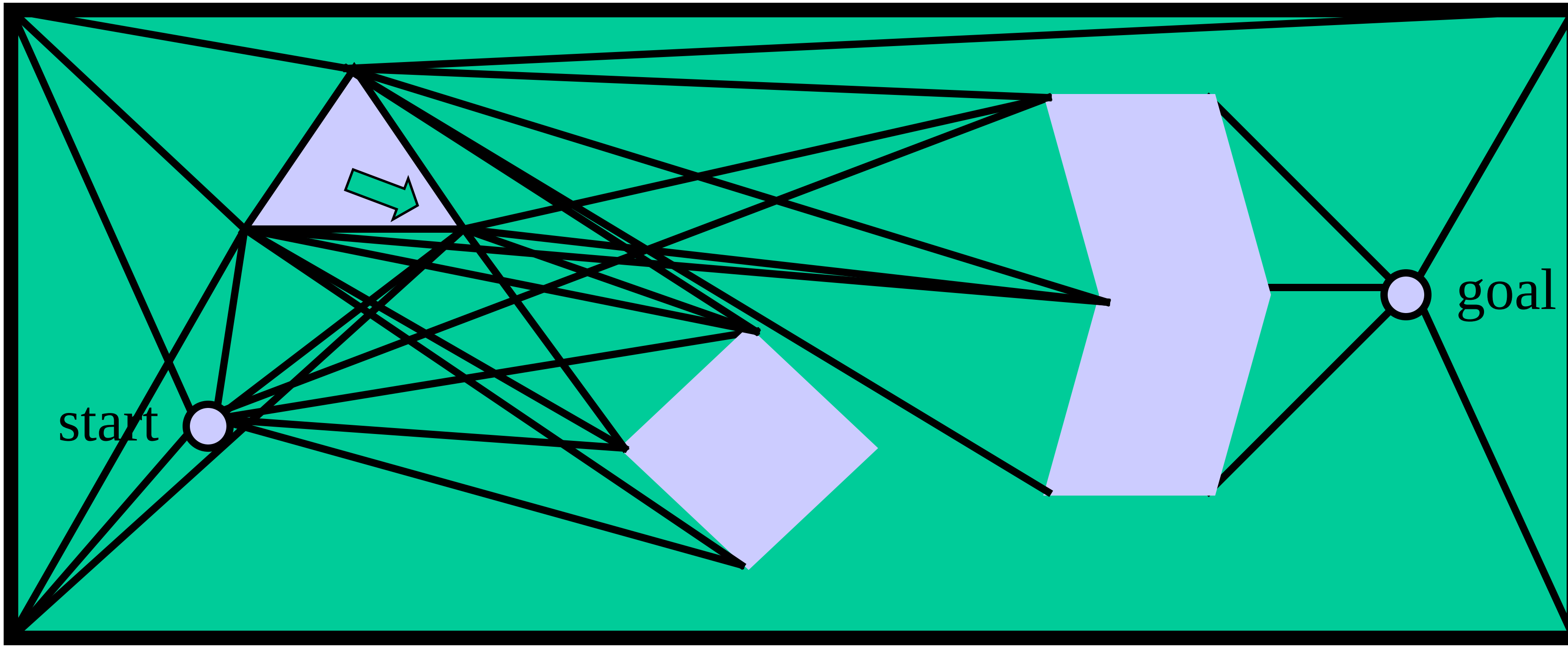
The Visibility Graph in Action (Part 3)

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



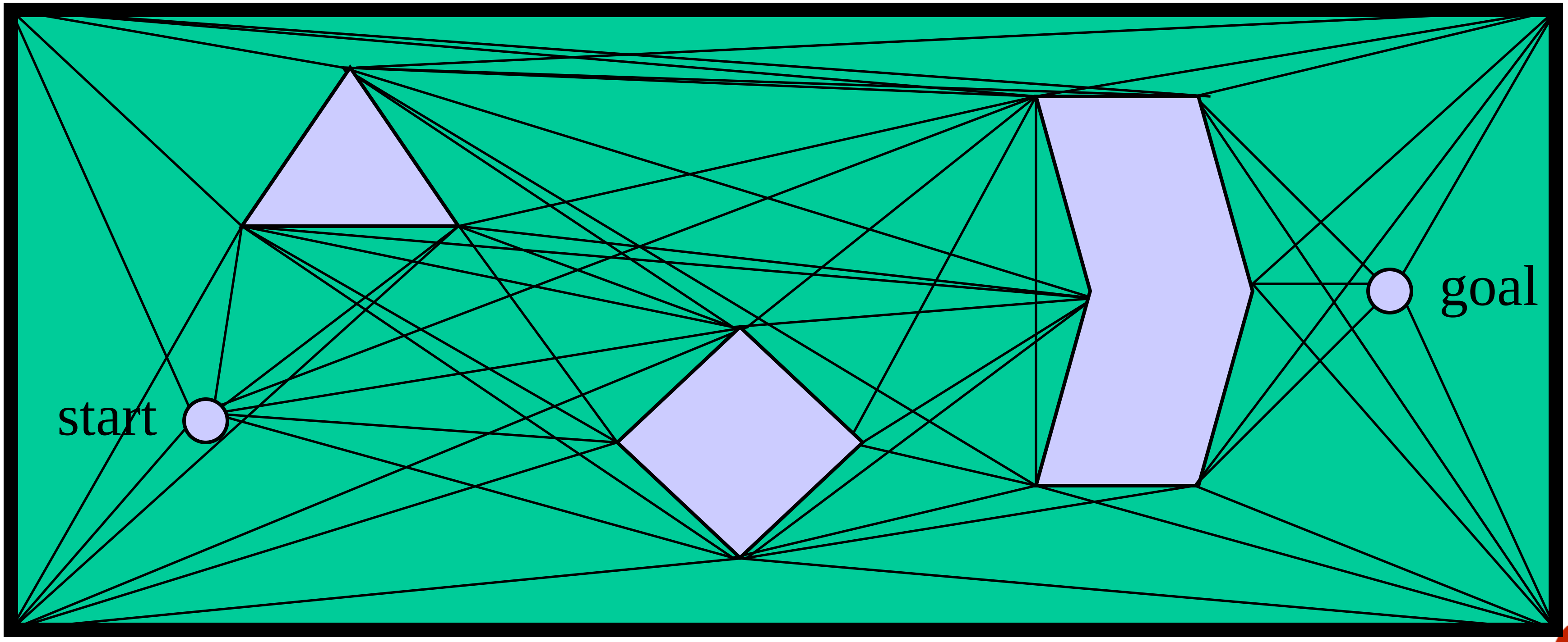
The Visibility Graph in Action (Part 4)

- 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.



Graph Search

Howie Choset

16-311

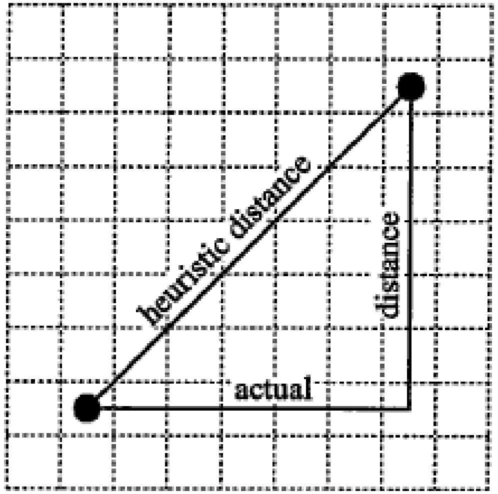
Informed Search: A*

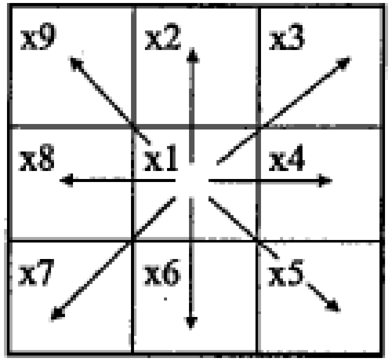
Notation

- ***n*** → node/state
- ***$c(n_1,n_2)$*** → the length of an edge connecting between n_1 and n_2
- ***$b(n_1) = n_2$*** → backpointer of a node n_1 to a node n_2.

Informed Search: A*

- Evaluation function, ***f(n) = g(n) + h(n)***
- Operating cost function, ***g(n)***
  - Actual operating cost having been already traversed
- Heuristic function, ***h(n)***
  - Information used to find the promising node to traverse
  - Admissible → never overestimate the actual path cost





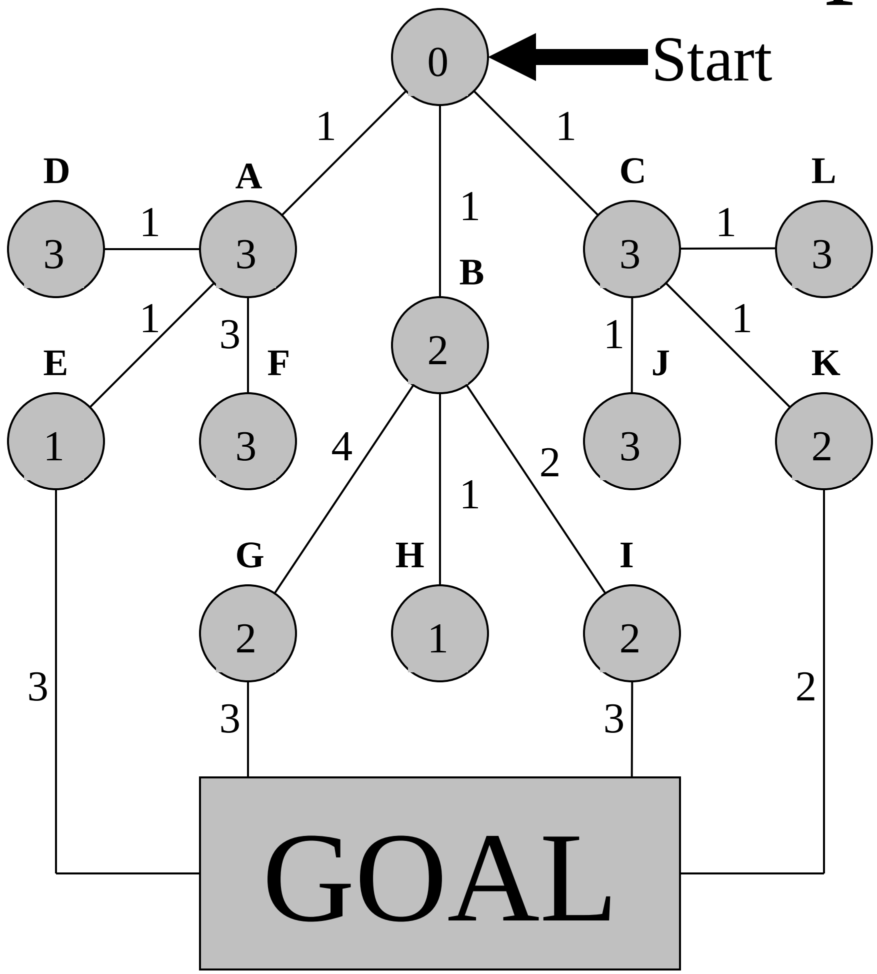
c(x1, x2) = 1
c(x1, x9) = 1.4

c(x1, x8) = 10000, if x8 is in obstacle, x1 is a free cell

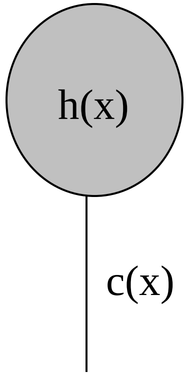
c(x1,x9) = 10000.4, if x9 is in obstacle, x1 is a free cell

Cost on a grid

Example (1/5)



Legend



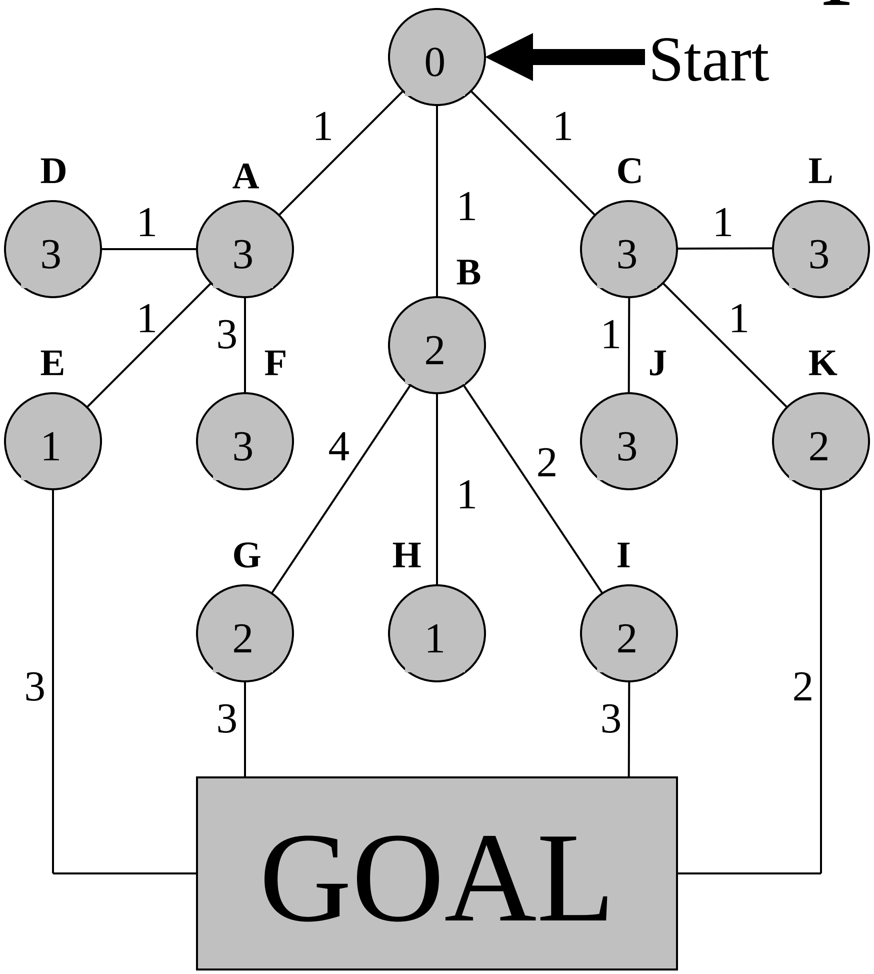
Priority = g(x) + h(x)

Note:

g(x) = sum of all previous arc costs, c(x), from start to x

Example: c(H) = 2

Example (2/5)



First expand the start node

B(3)
A(4)
C(4)

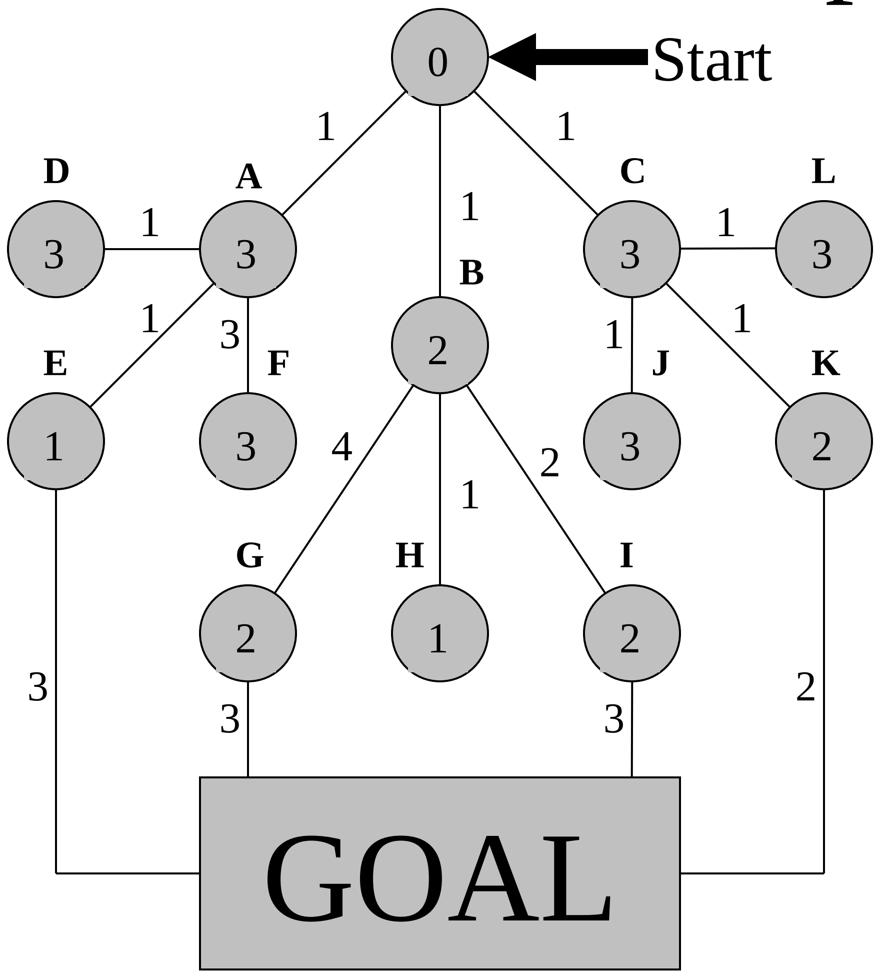
If goal not found, expand the first node in the priority queue (in this case, B)

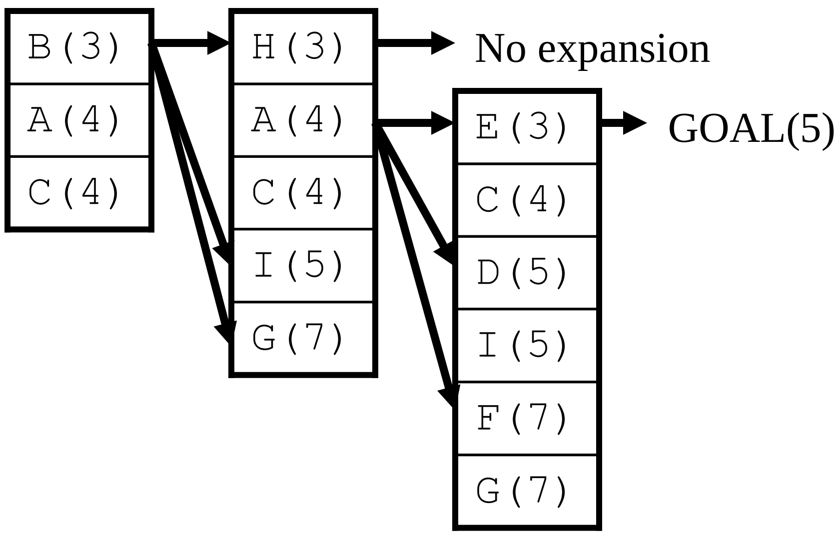
H(3)
A(4)
C(4)
I(5)
G(7)

Insert the newly expanded nodes into the priority que and continue until the goal found, or the priority queu empty (in which case no p exists)

Note: for each expanded node, you also need a pointer to its respective parent. For example, nodes A, B and C point to Start

Example (3/5)

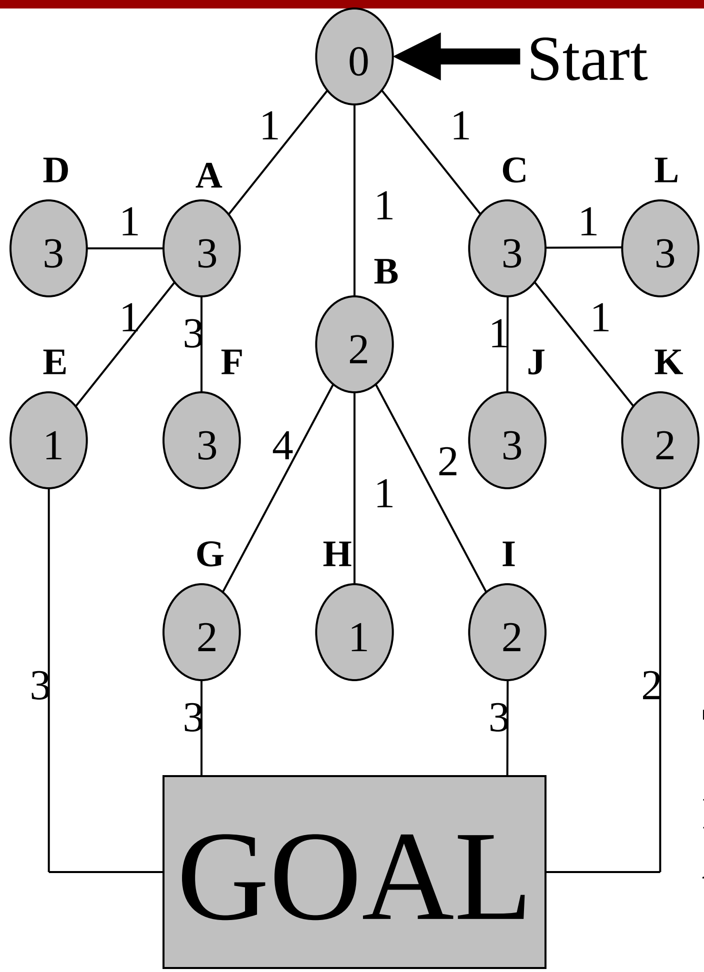


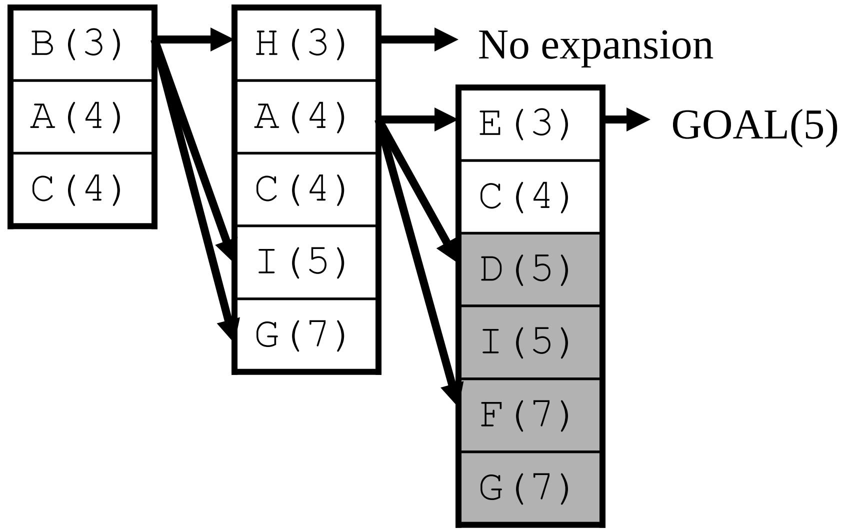


We've found a path to the goal:
Start => A => E => Goal
(*from the pointers*)

Are we done?

Example (4/5)



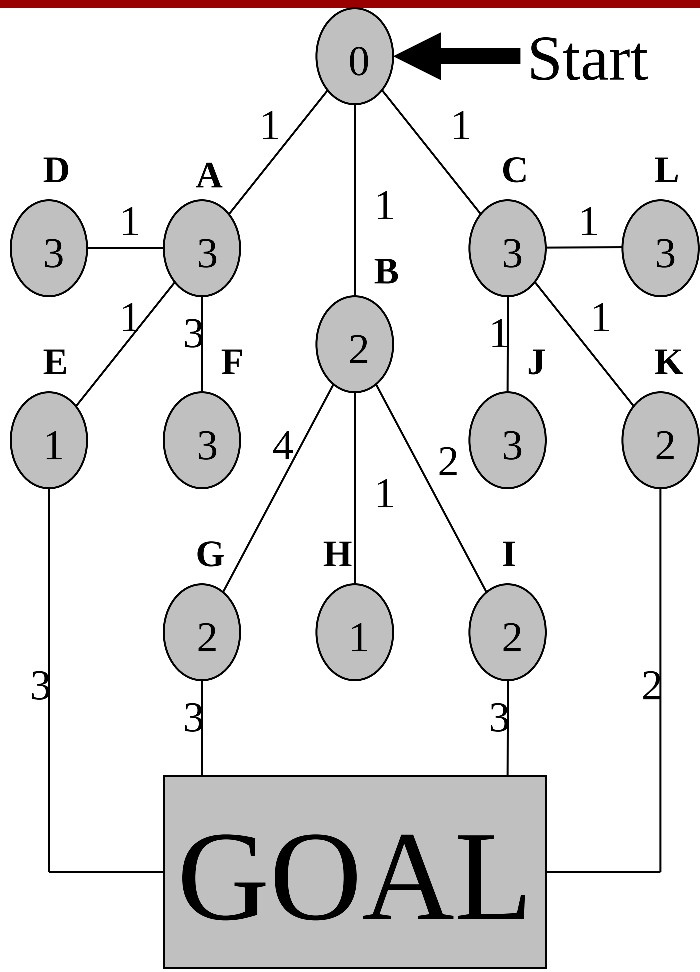


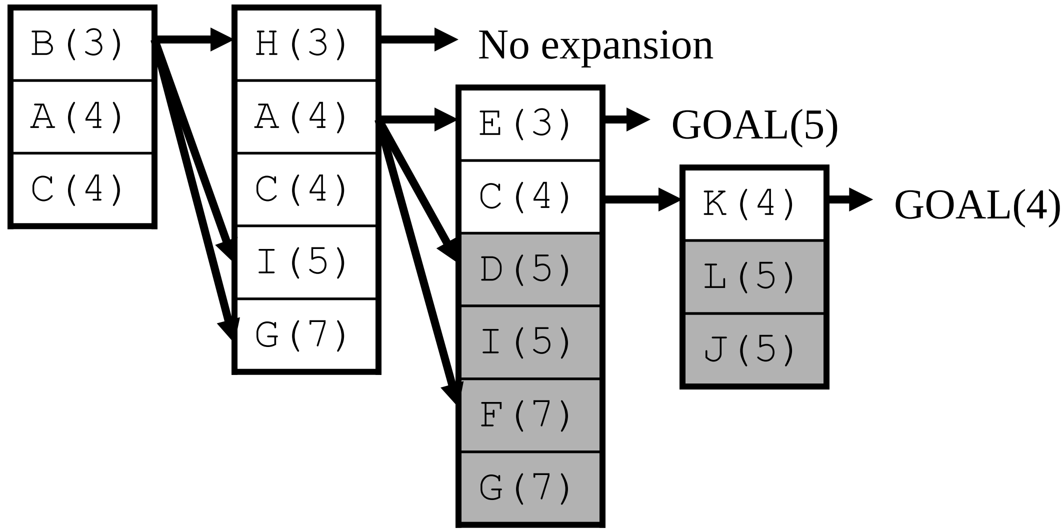
There might be a shorter path, but assuming non-negative arc costs, nodes with a lower pric than the goal cannot yield a better path.

In this example, nodes with a priority greater th equal to 5 can be pruned.

Why don't we expand nodes with an equivalen (why not expand nodes D and I?)

Example (5/5)





We can continue to throw away nodes with priority levels lower than the lowest goal found.

As we can see from this example, there was a shorter path through node K. To find the path, simply follow the back pointers.

Therefore the path would be:
Start => C => K => Goal

If the priority queue still wasn't empty, we would continue expanding while throwing away nodes with priority lower than 4.
(remember, lower numbers = higher priority)

Monotonic

- never overestimates the cost of getting from a node to its neighbor.

- for all paths x,y where y is a successor of x, i.e.,

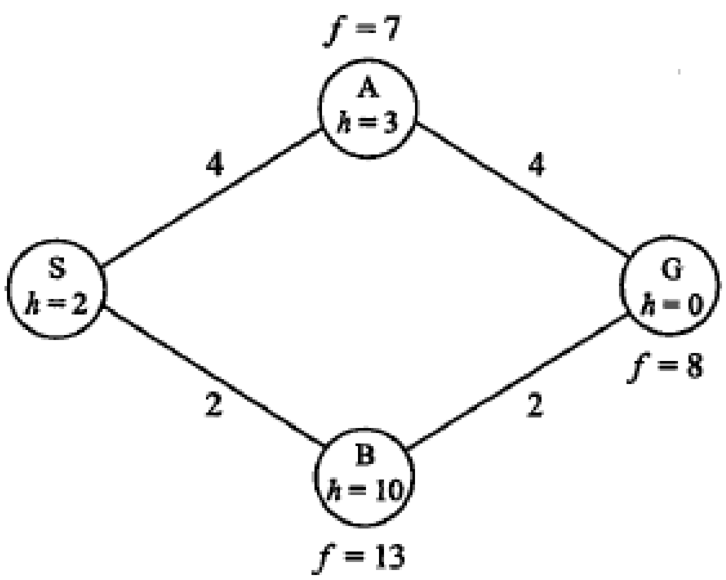
$$h(x) \leq g(y) - g(x) + h(y)$$

- h(A) = 3 g(A) = 1 h(E) = 1 g(E) = 2

$$h(A) = 3 \not\leq g(E) - g(A) + h(E) = 2 - 1 + 1 = 2$$

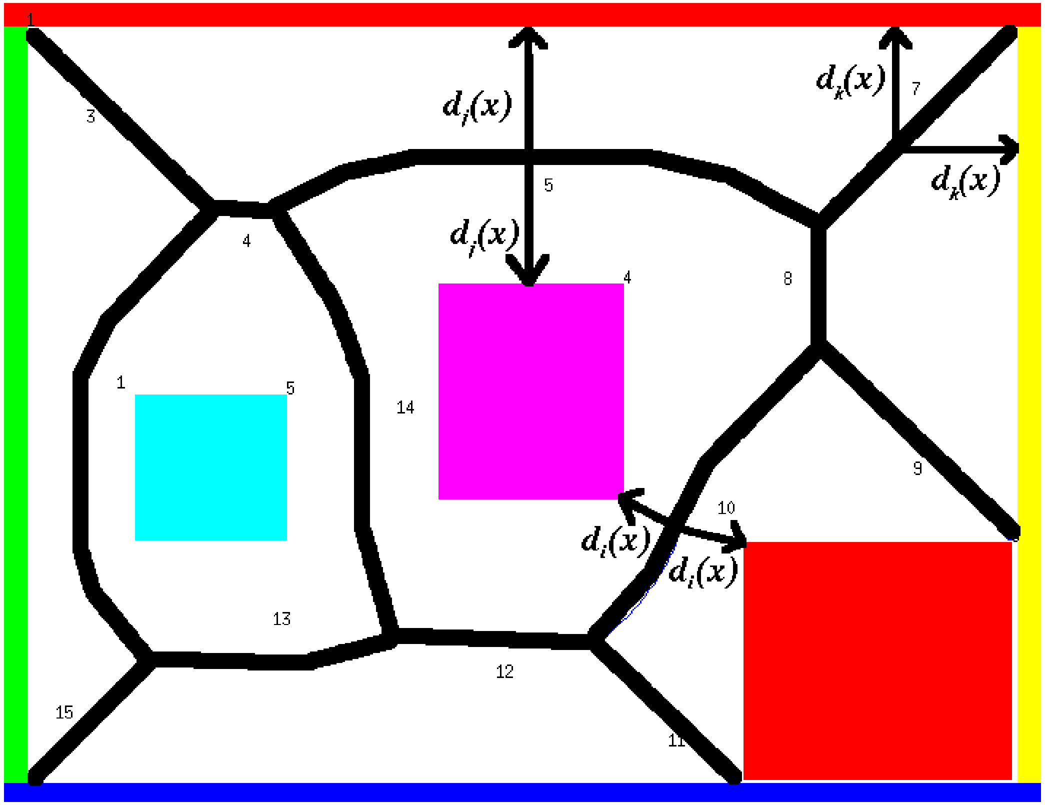
Non-opportunistic

1. Put S on priority Q and expand it
2. Expand A because its priority value is 7
3. The goal is reached with priority value 8
4. This is less than B's priority value which is 13



Roadmap: GVG

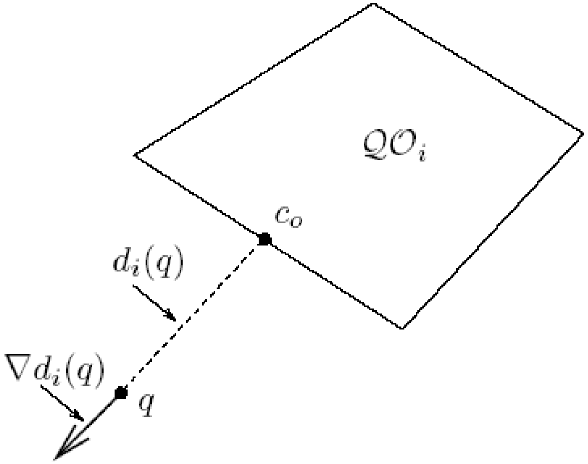
- A GVG is formed by paths equidistant from the two closest objects
- *Remember "spokes", start and goal*



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

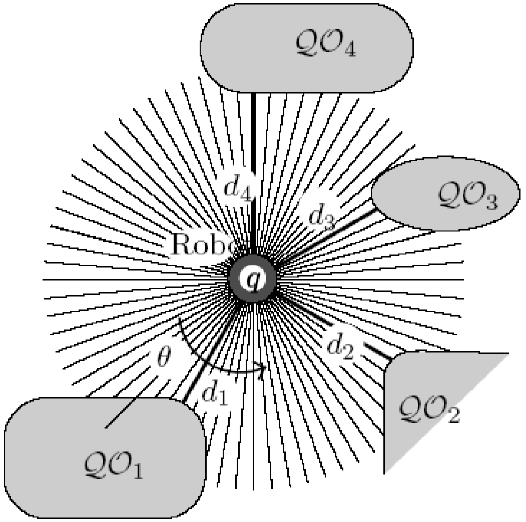
Distance to Obstacle(s)

$$d_i(q) = \min_{c \in \mathcal{QO}_i} d(q, c)$$



$$\nabla d_i(q) = \frac{q - c}{d(q, c)}$$

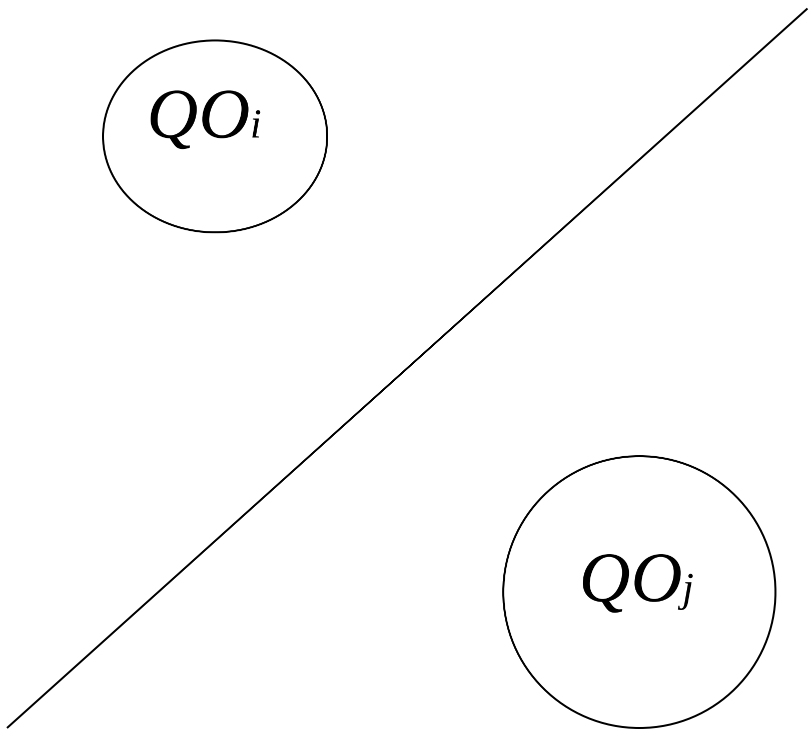
$$D(q) = \min d_i(q)$$



Two-Equidistant

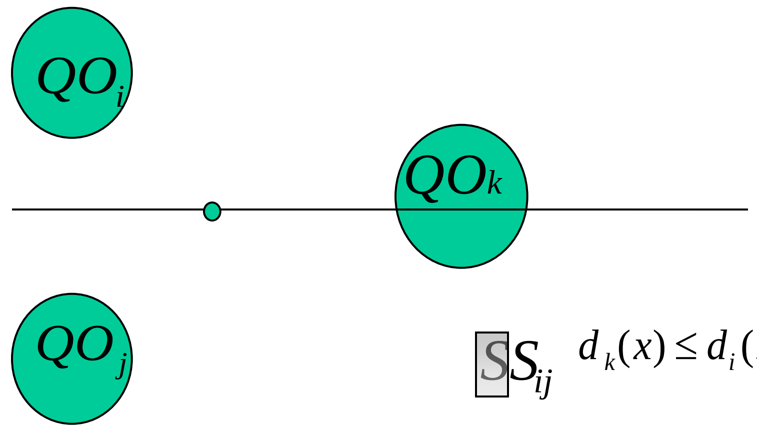
- *Two-equidistant surface*

$$S_{ij} = \{x \in Q_{\text{free}} : d_i(x) - d_j(x) = 0\}$$



More Rigorous Definition

Going through obstacles



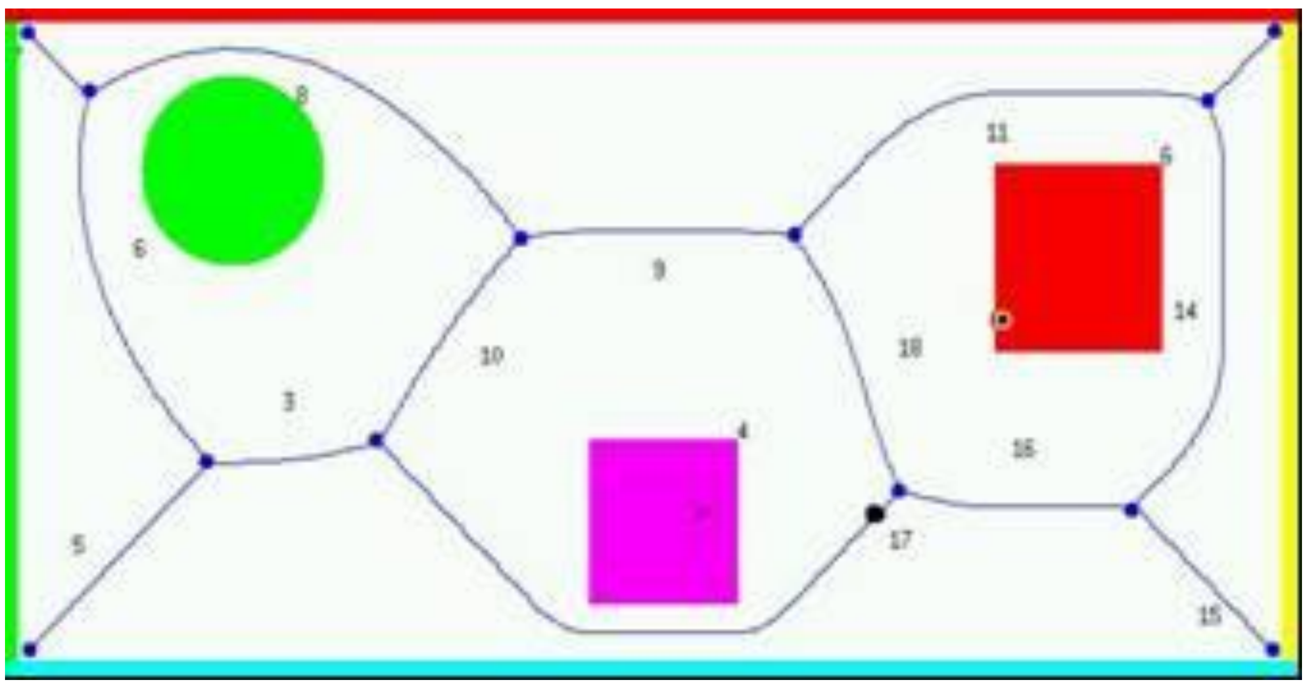
$SS_{ij} \quad d_k(x) \leq d_i(x) = d_j(x)$

Two-equidistant face

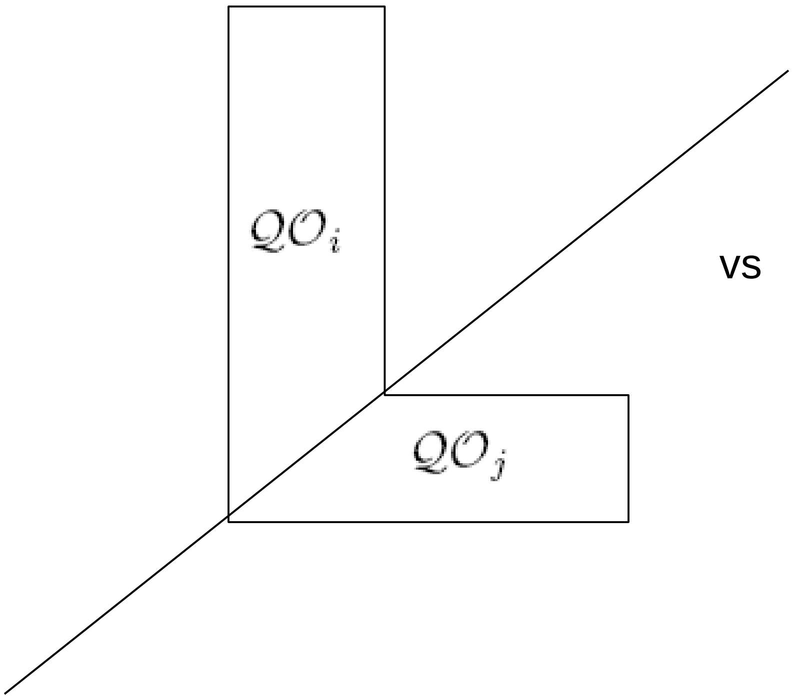
$$F_{ij} = \{x \in SS_{ij} : d_i(x) = d_j(x) \leq d_h(x), \forall h \neq i, j\}$$

General Voronoi Diagram

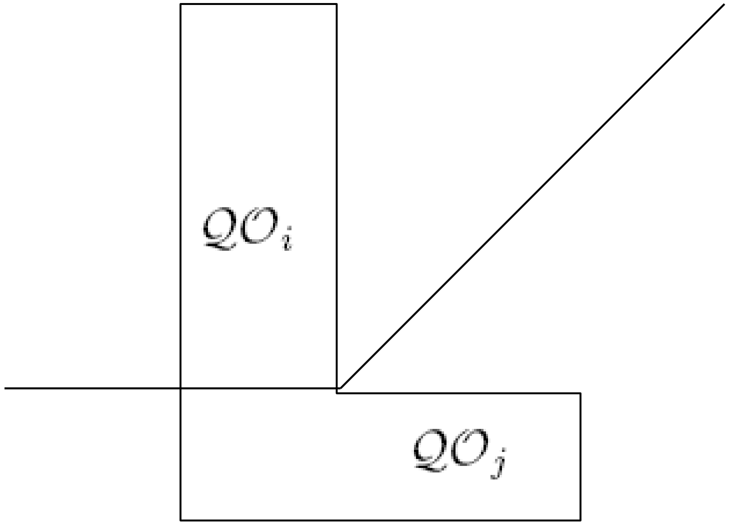
$$\mathrm{GVD} = \bigcup_{i=1}^{n-1} \bigcup_{j=i+1}^{n} F_{ij}$$



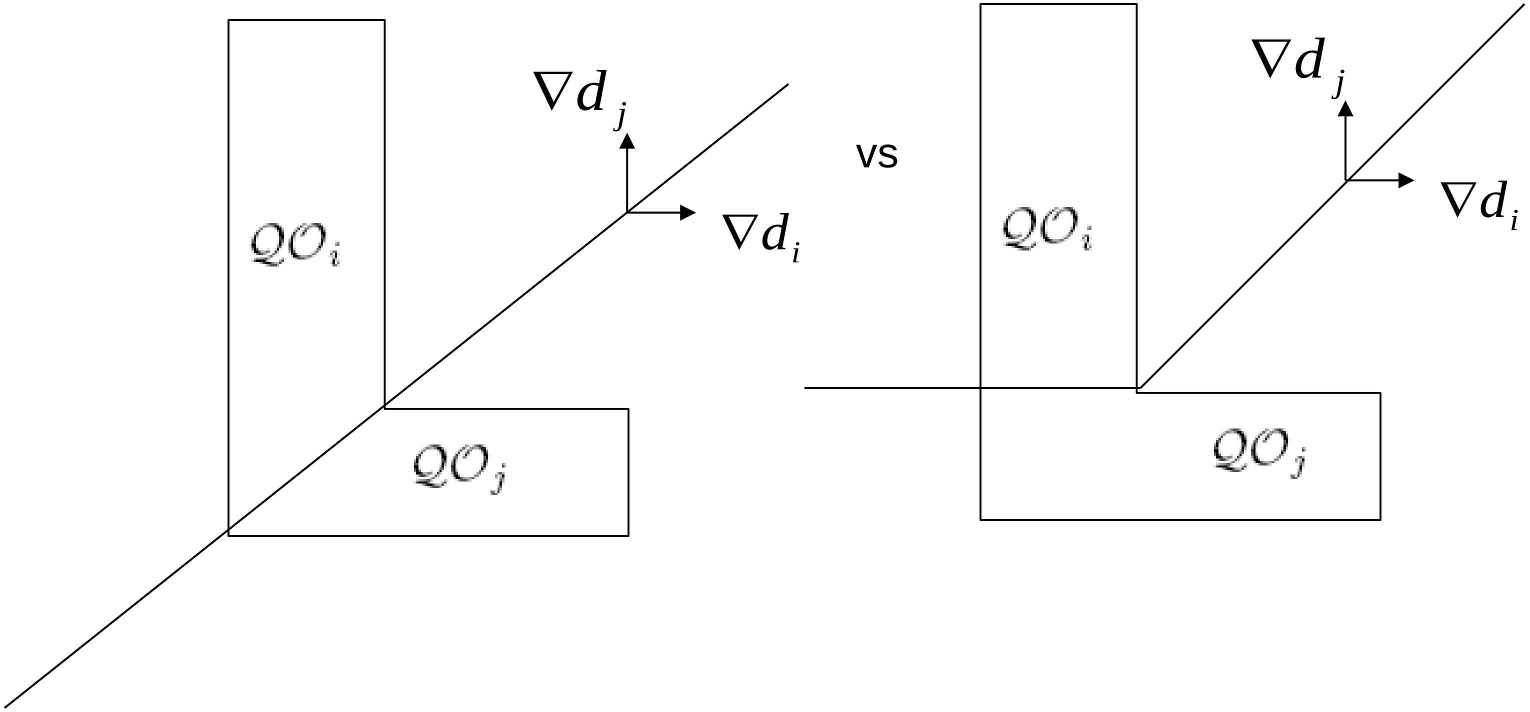
What about concave obstacles?



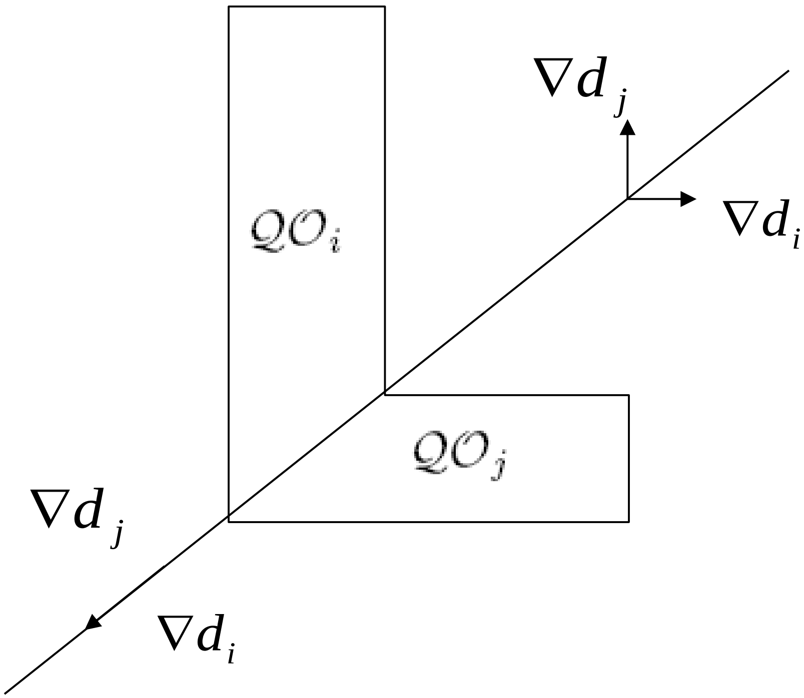
vs



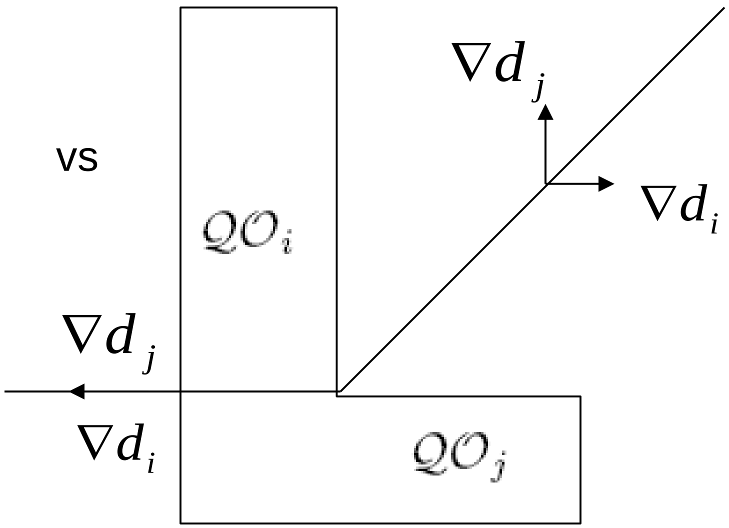
What about concave obstacles?



What about concave obstacles?



vs



Two-Equidistant

- *Two-equidistant surface*

$$S_{ij} = \{x \in Q_{\text{free}} : d_i(x) - d_j(x) = 0\}$$

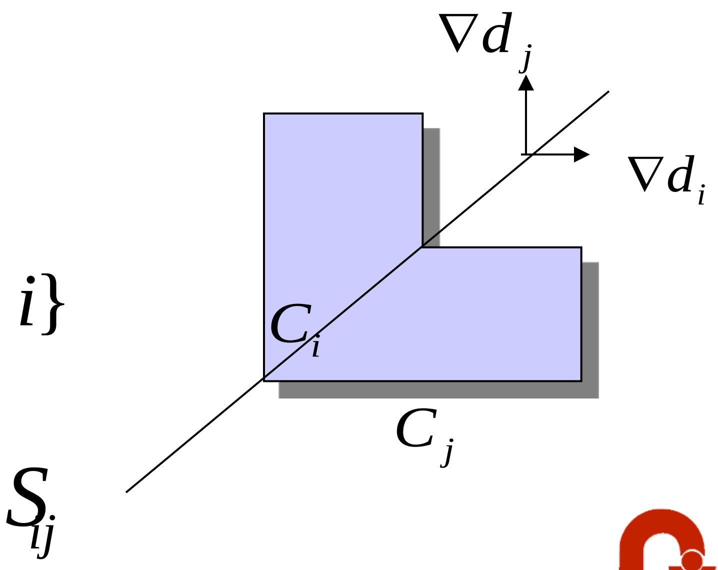
Two-equidistant surjective surface

$$SS_{ij} = \{x \in S_{ij} : \nabla d_i(x) \neq \nabla d_j(x)\}$$

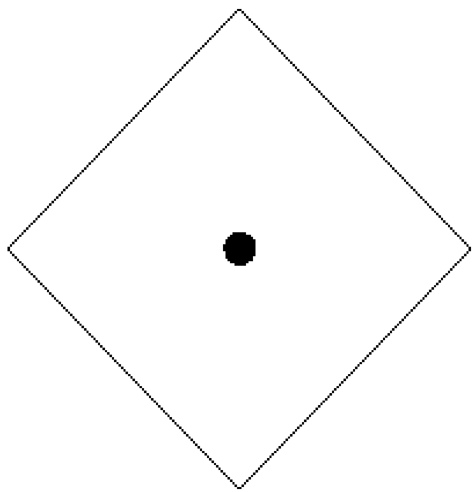
Two-equidistant Face

$$F_{ij} = \{x \in SS_{ij} : d_i(x) \leq d_h(x), \forall h \neq i\}$$

$$\text{GVD} = \bigcup_{i=1}^{n-1} \bigcup_{j=i+1}^{n} F_{ij}$$

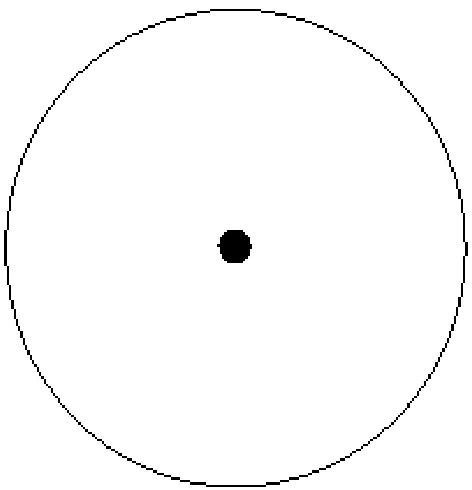


Voronoi Diagram: Metrics



$\{(x,y) : |x| + |y| = \text{const}\}$

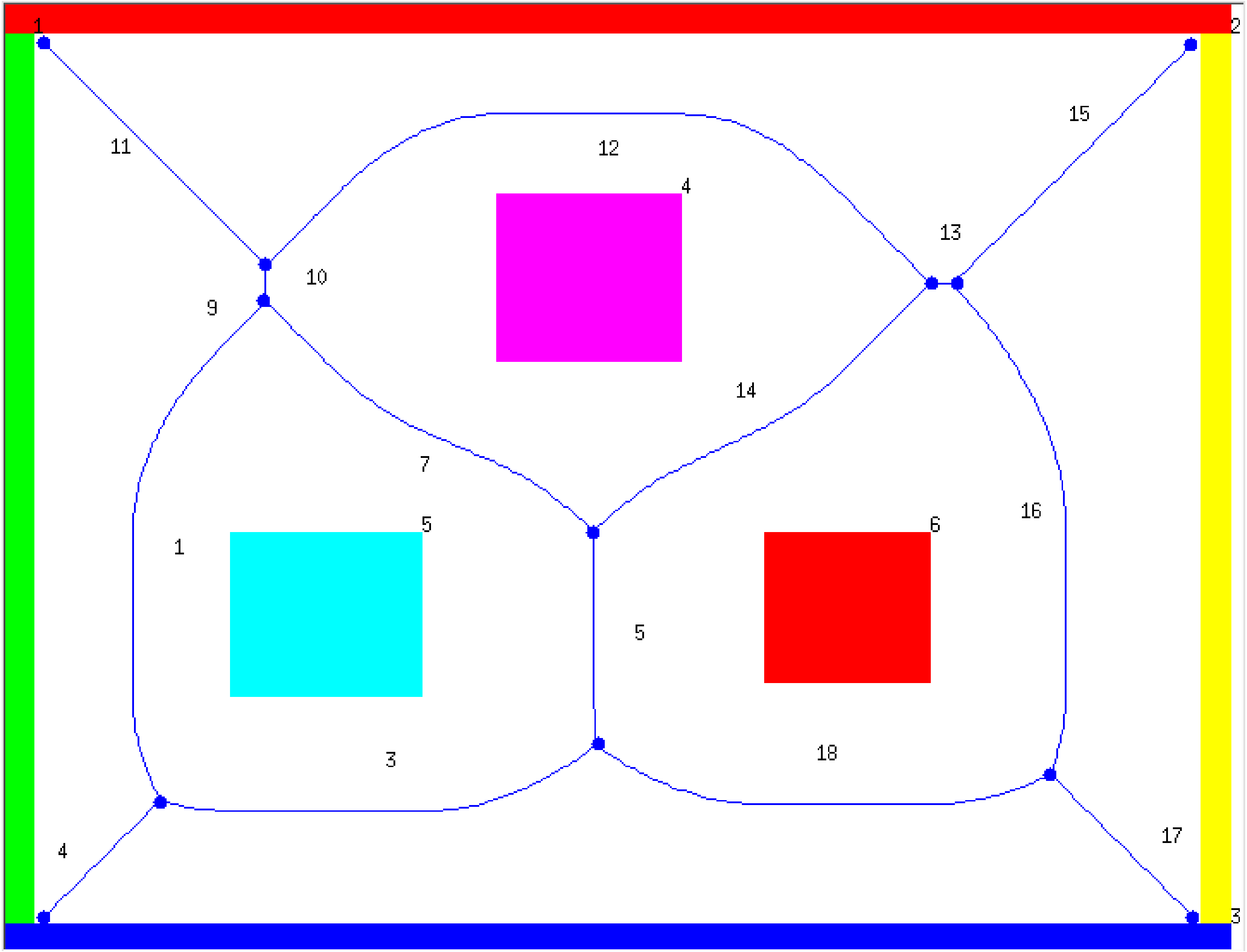
L1



$\{(x,y) : x^2 + y^2 = \text{const}\}$

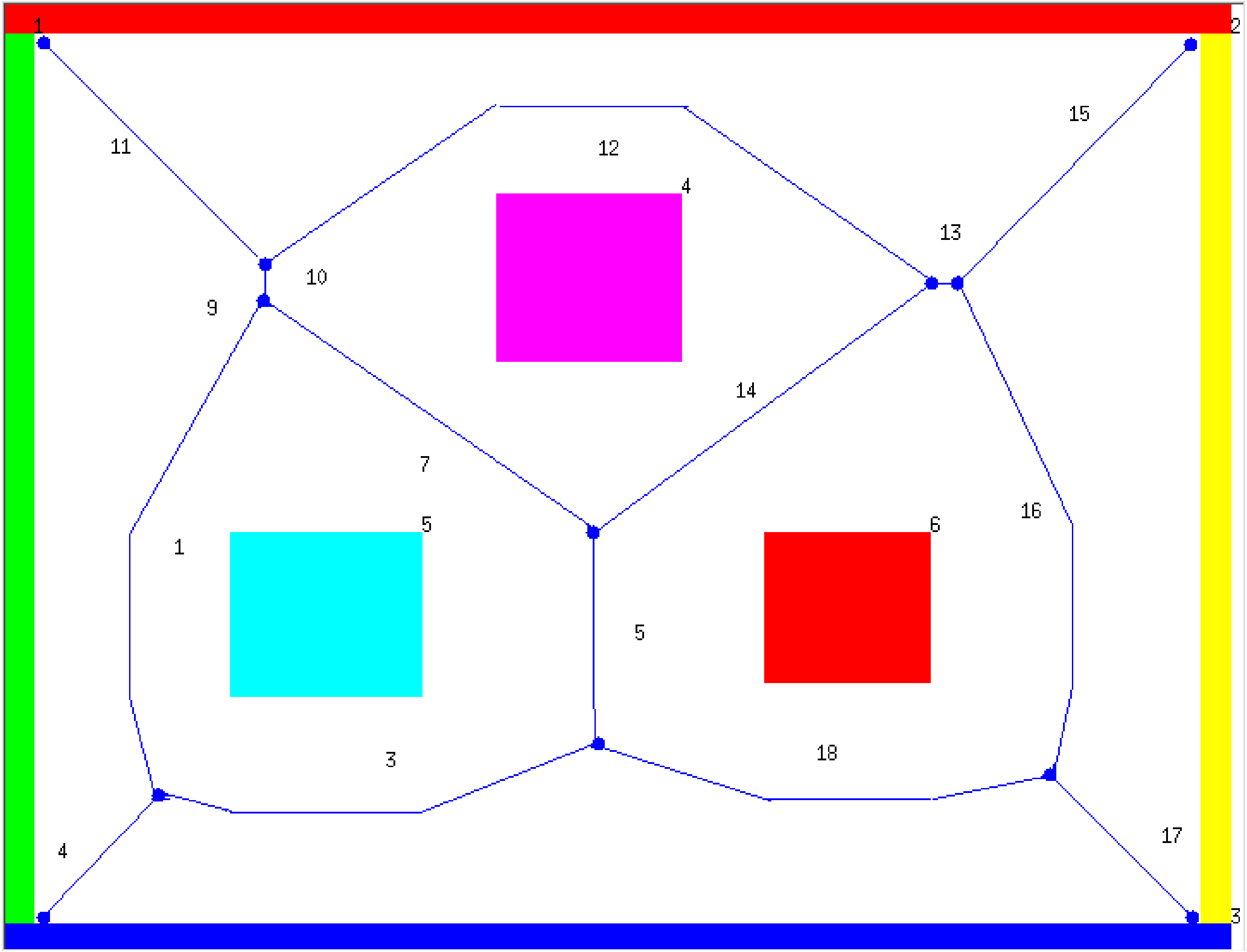
L2

Voronoi Diagram (L2)



Note the curved edges

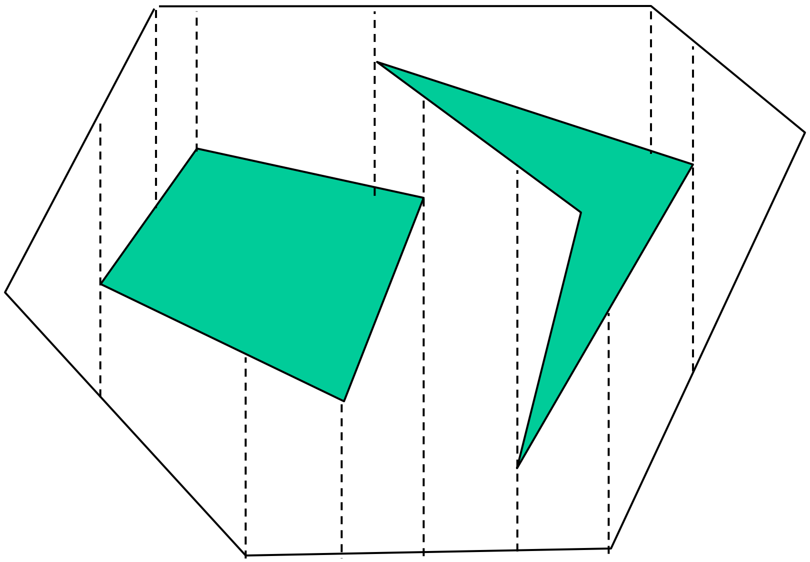
Voronoi Diagram (L1)

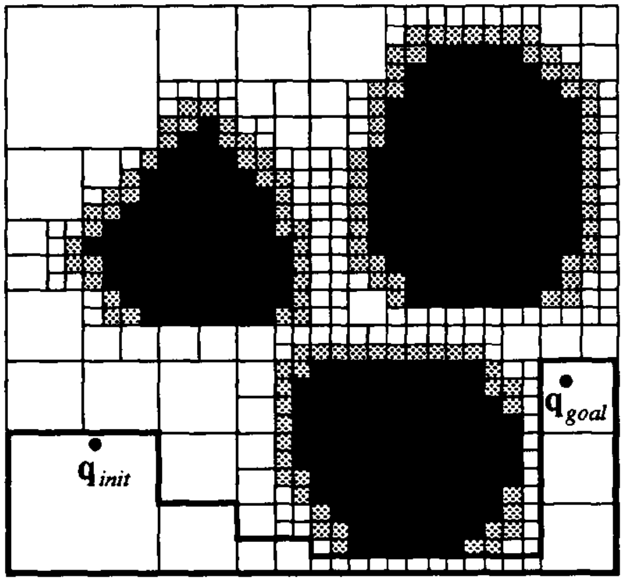


Note the lack of curved edges

Exact Cell vs. Approximate Cell

- Cell: simple region



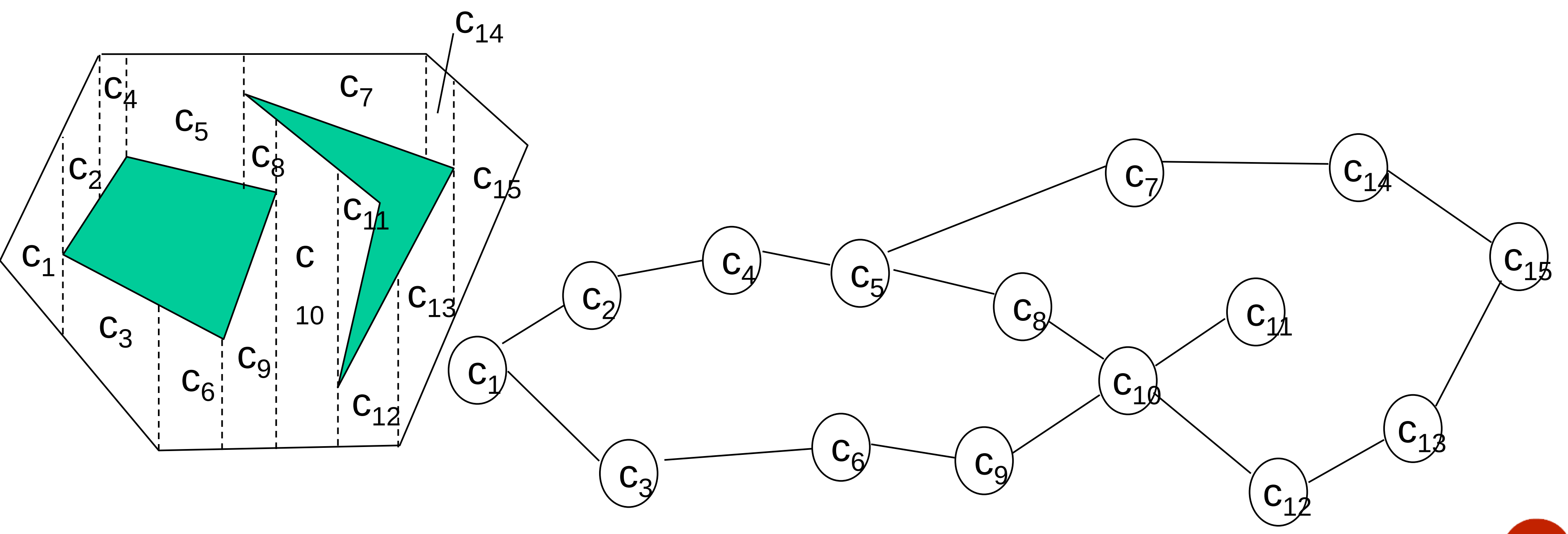


Adjacency Graph

– Node correspond to a cell

– Edge connects nodes of adjacent cells

- Two cells are *adjacent* if they share a common boundary



Set Notation

Some set notation

- Interior of A (int(A)) is the largest open subset of A
- Closure of A (cl(A)) is the smallest closesd set that contains A
- Complement of A ($\bar{A}$) is everything not in A.
- Boundary of A (∂A) is the closure of A take away its interior.

Examples

Examples

- $\mathrm{int}[0,1] = (0,1)$, $\mathrm{int}(0,1) = (0,1)$
- $\mathrm{cl}[0,1] = [0,1]$, $\mathrm{cl}(0,1) = [0,1]$
- $\overline{[0,1]} = (-\infty, 0) \cup (1, \infty)$
- $\partial[0,1] = \partial(0,1) = \{0,1\}$

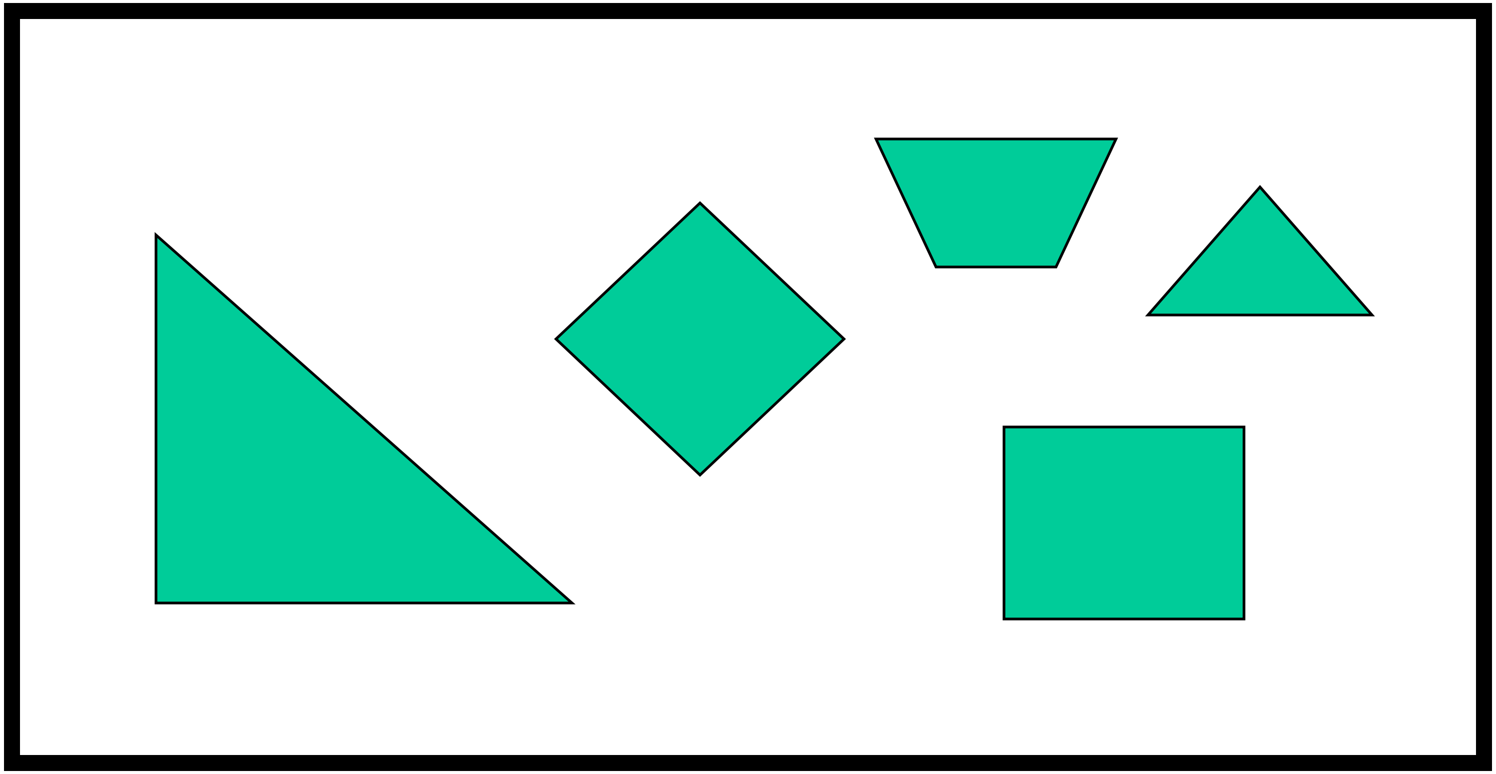
Definition

Exact Cellular Decomposition (as opposed to approximate)

- ν_i is a cell
- $\text{int}(\nu_i) \cap \text{int}(\nu_j) = \emptyset$ if and only if $i \neq j$
- $Fs \cap (\text{cl}(\nu_i) \cap \text{cl}(\nu_j)) \neq \emptyset$ if ν_i and ν_j are adjacent cells
- $Fs = \cup_i (\nu_i)$

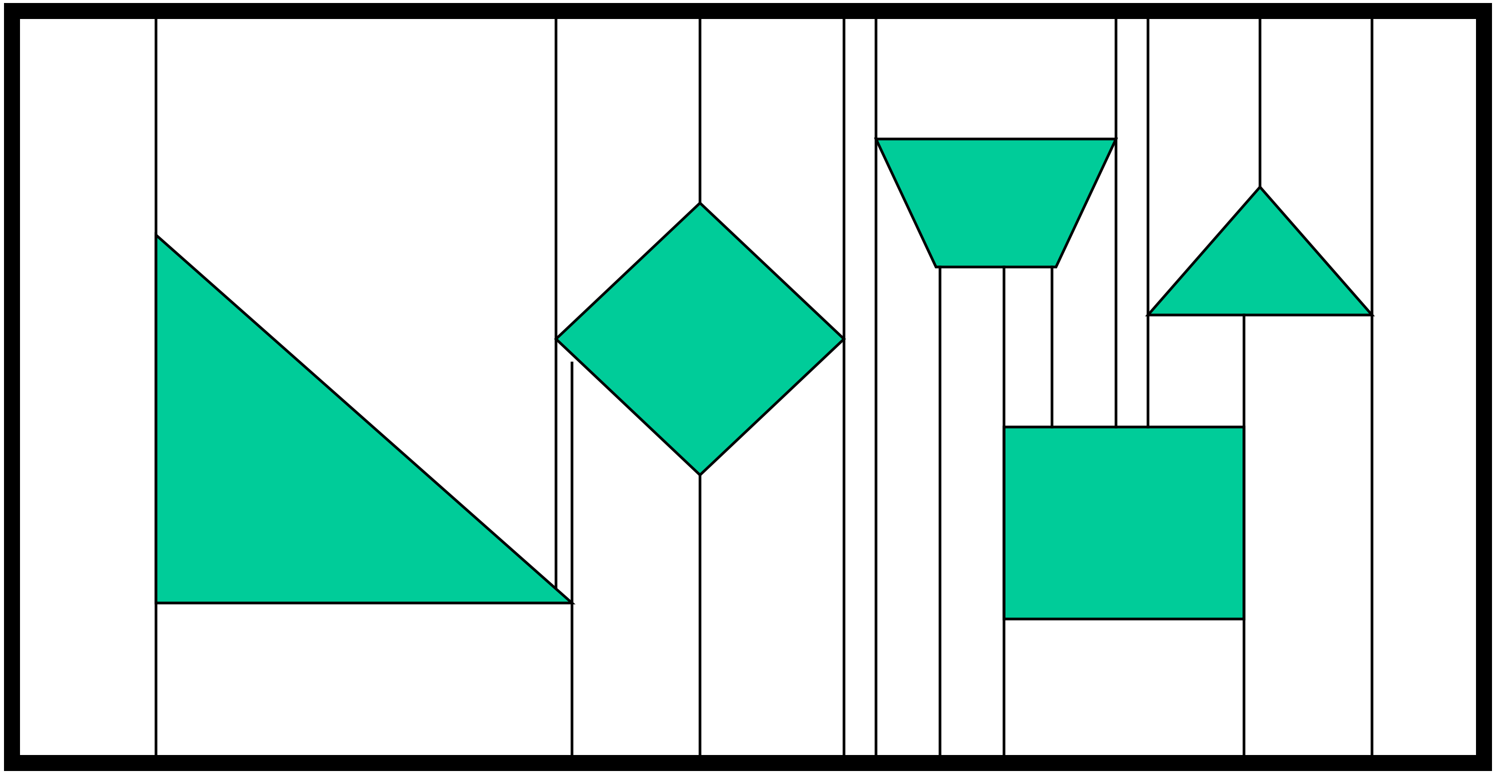
Cell Decompositions: Trapezoidal Decomposition

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



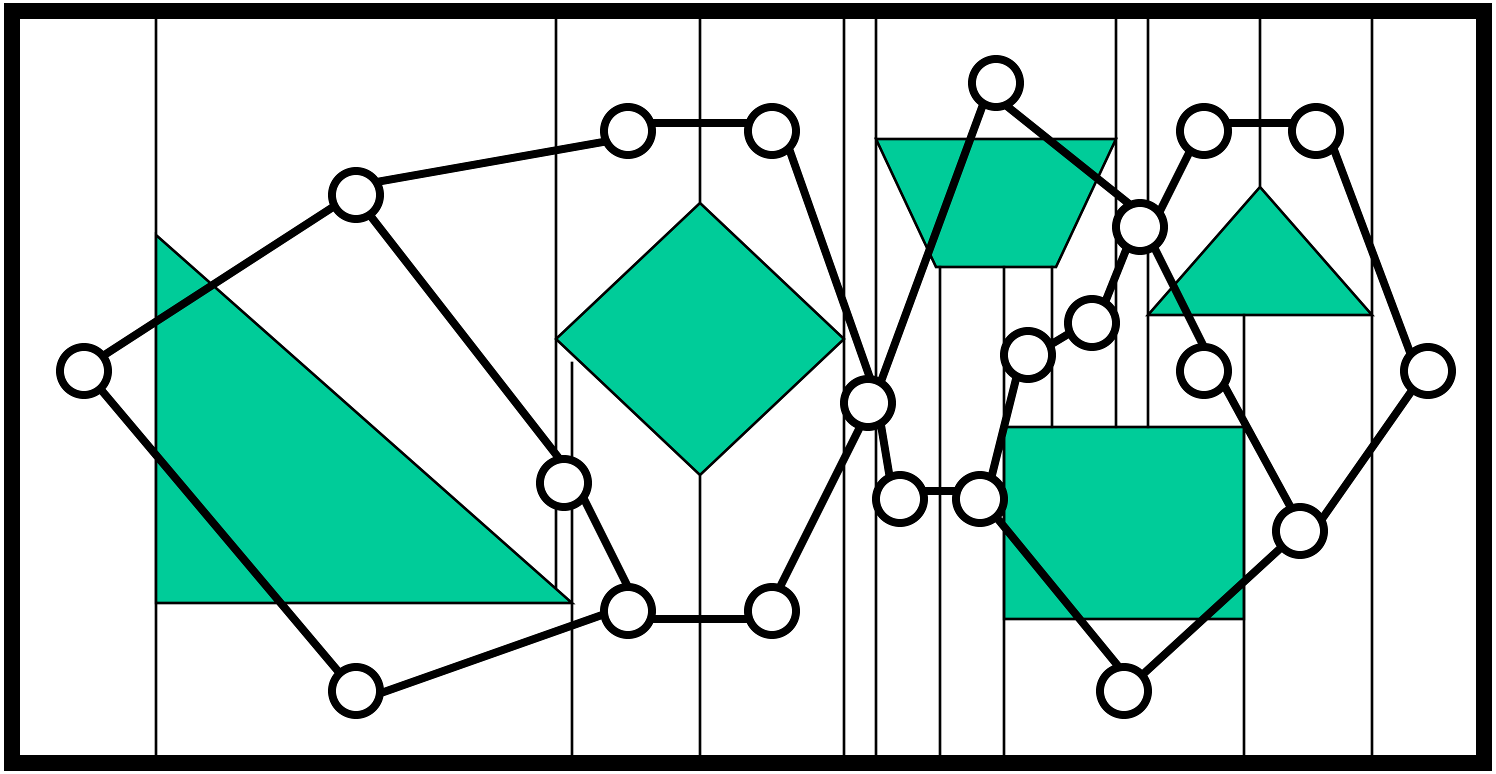
Cell Decompositions: Trapezoidal Decomposition

- Simply draw a vertical line from each vertex until you hit an obstacle. This reduces the world to a union of trapezoid-shaped cells



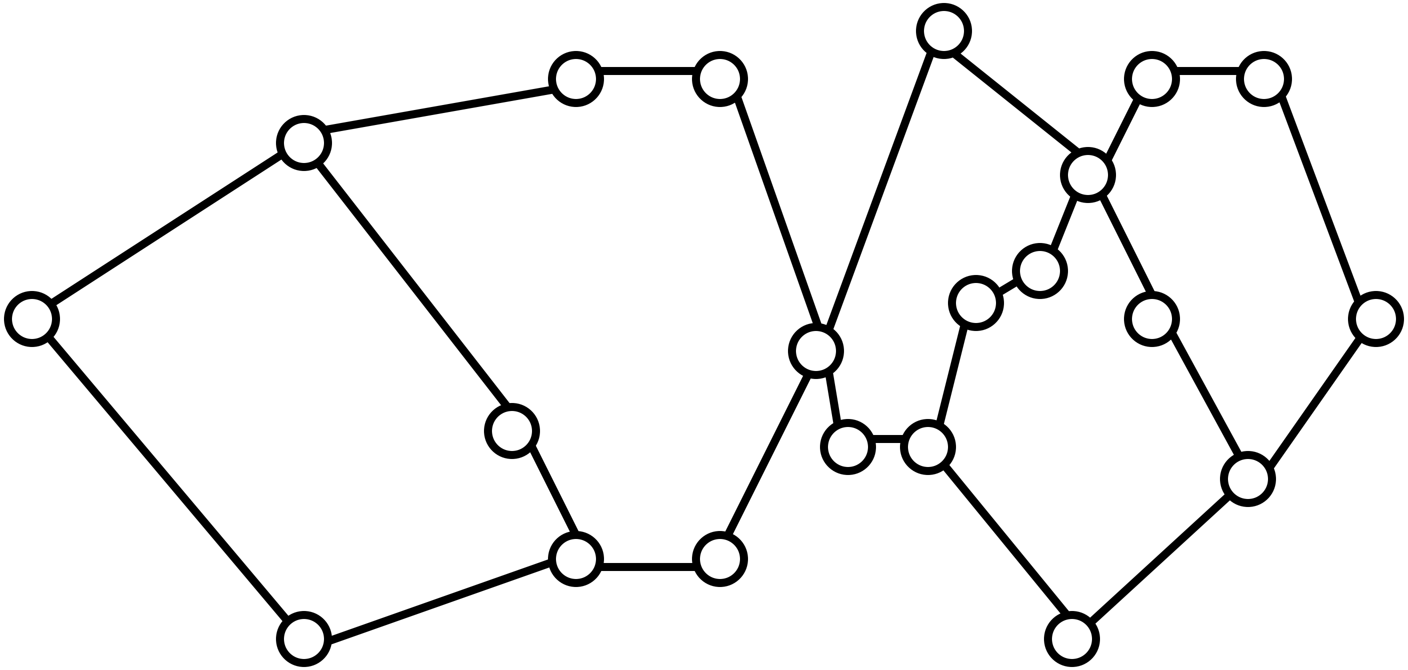
Applications: Coverage

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



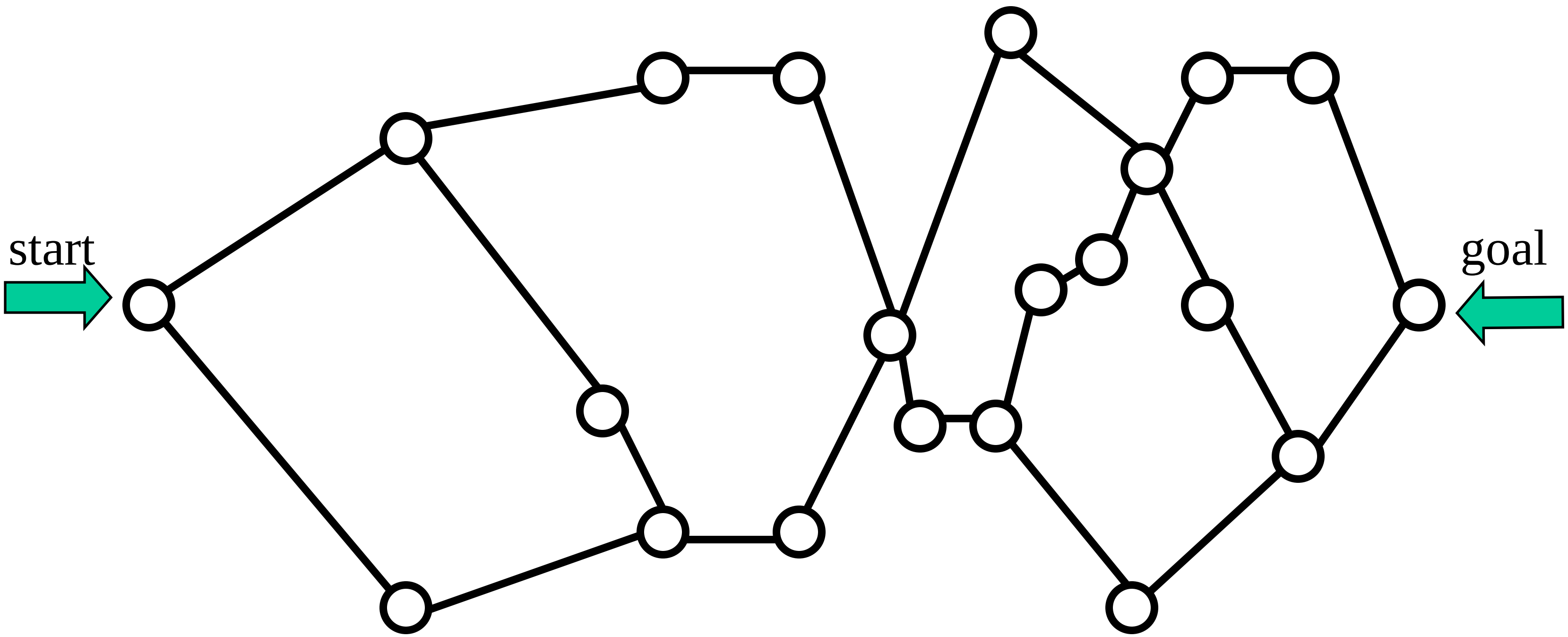
Find a path

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



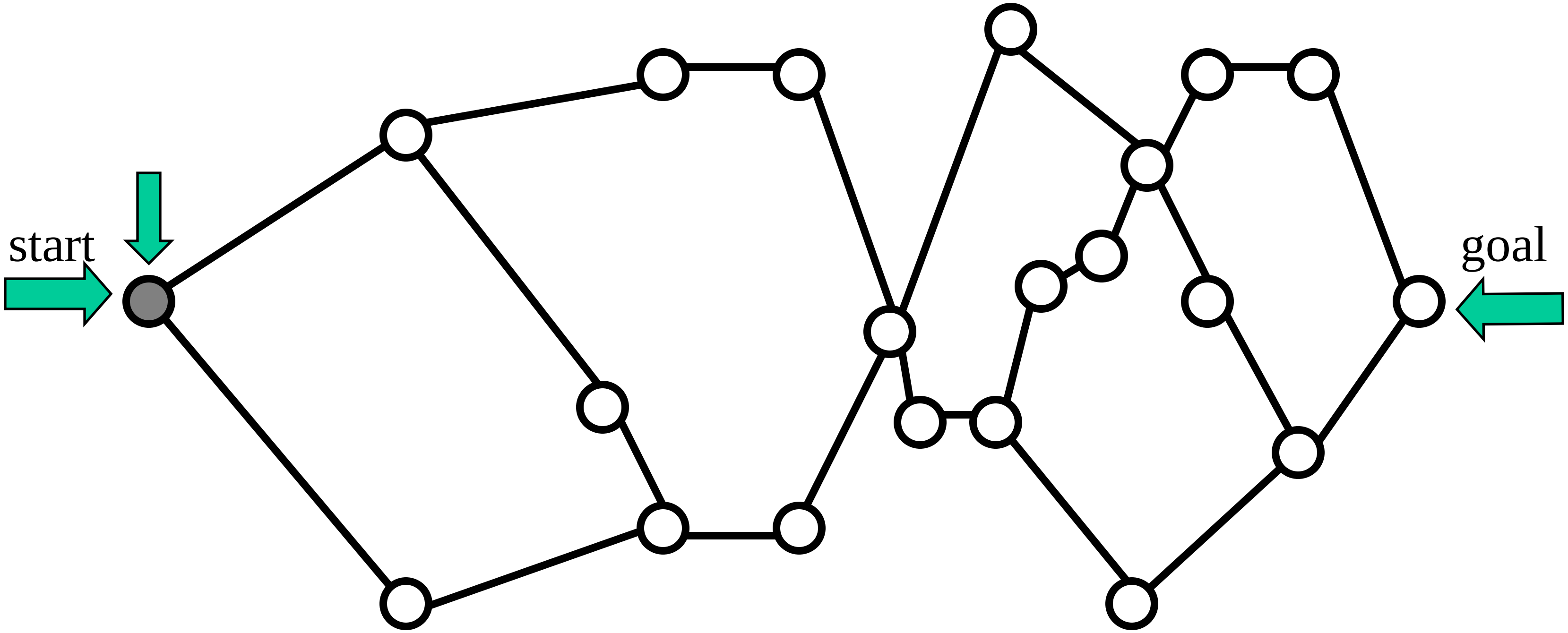
Find a path

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



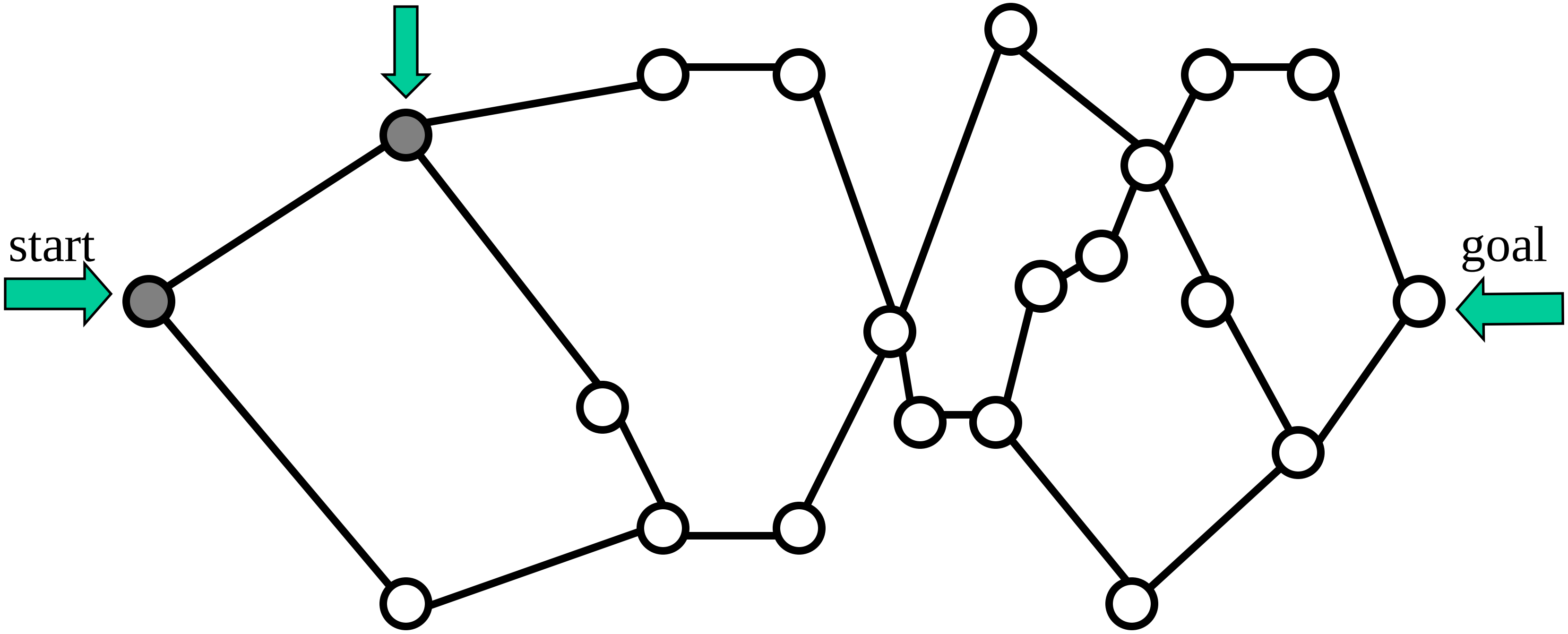
Find a path

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



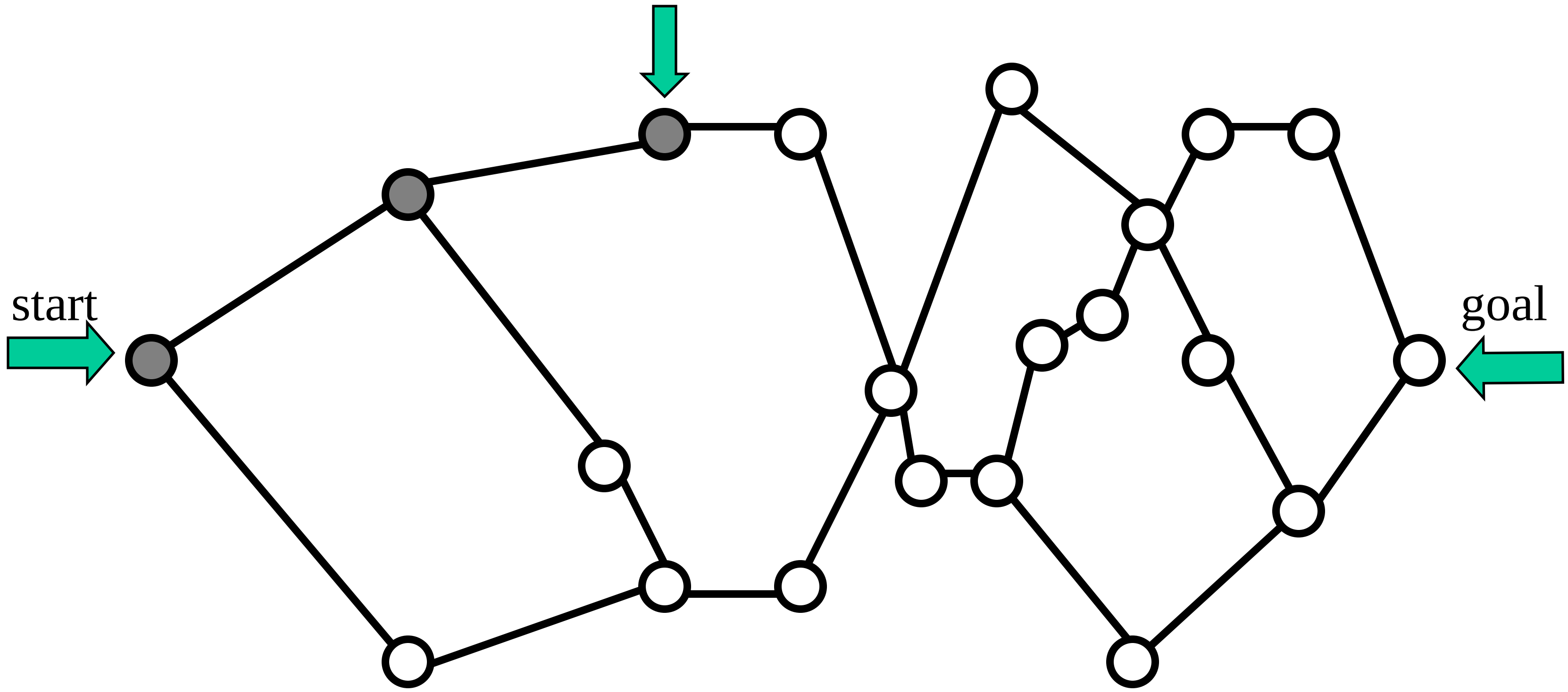
Find a path

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



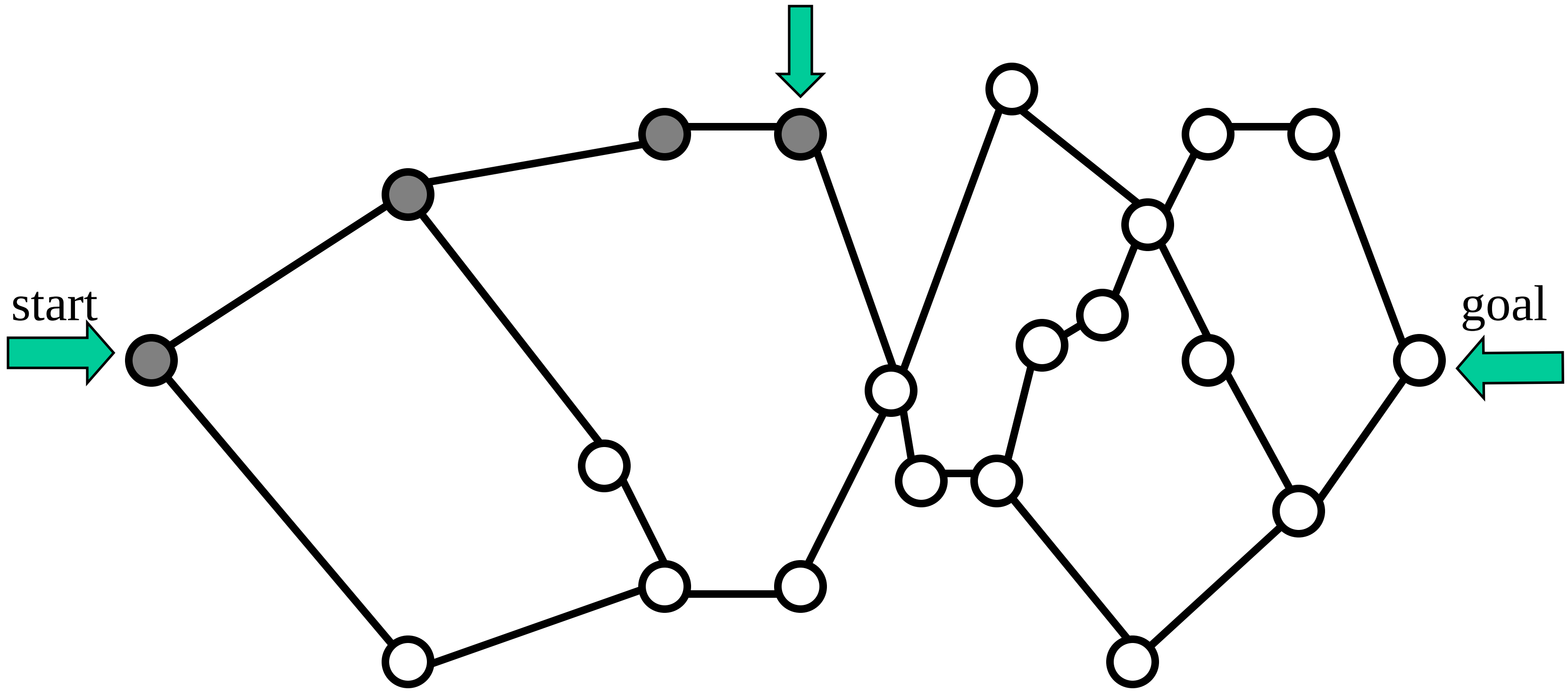
Find a path

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



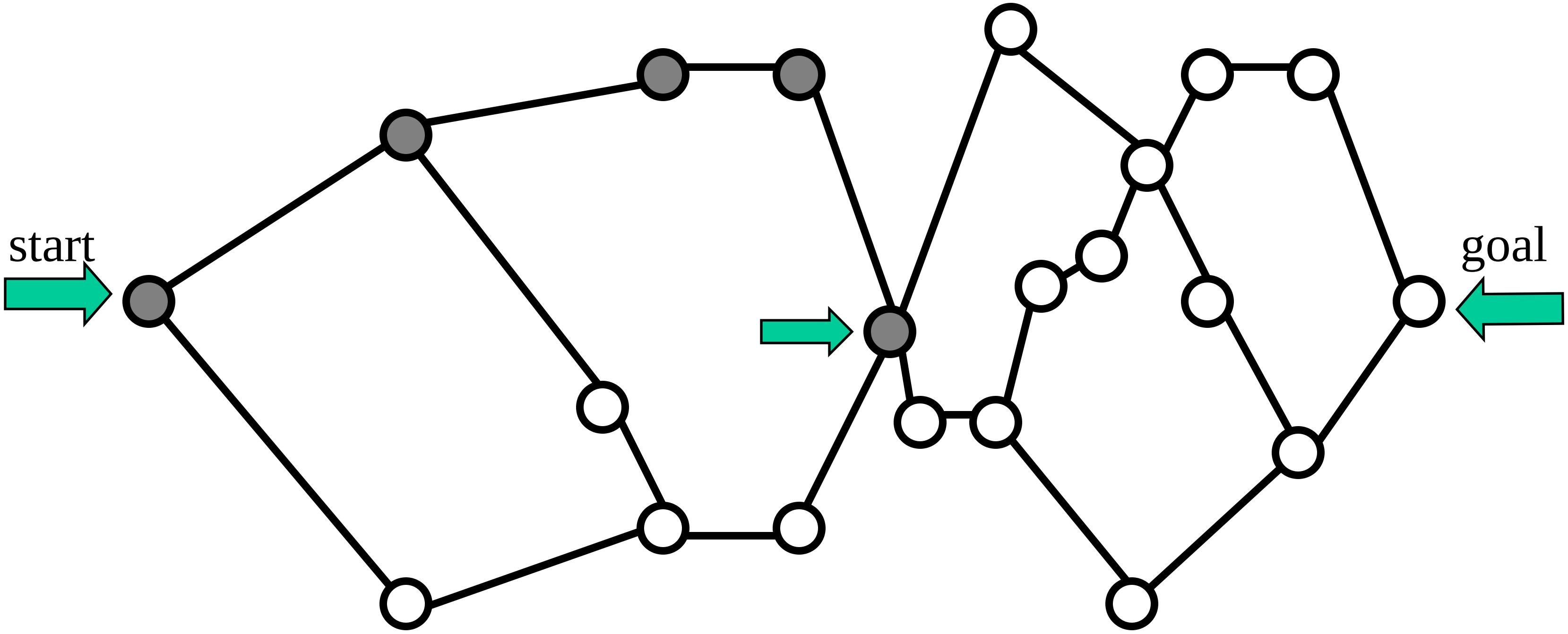
Find a path

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



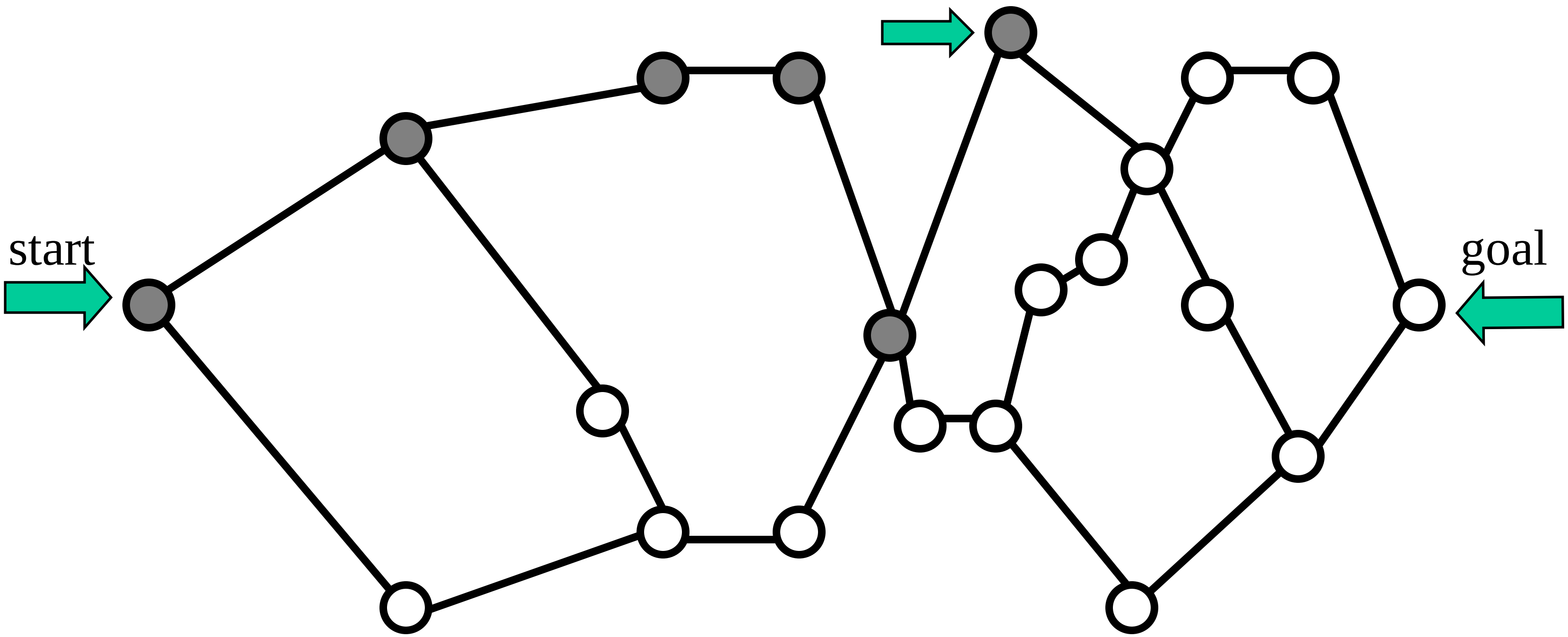
Find a path

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



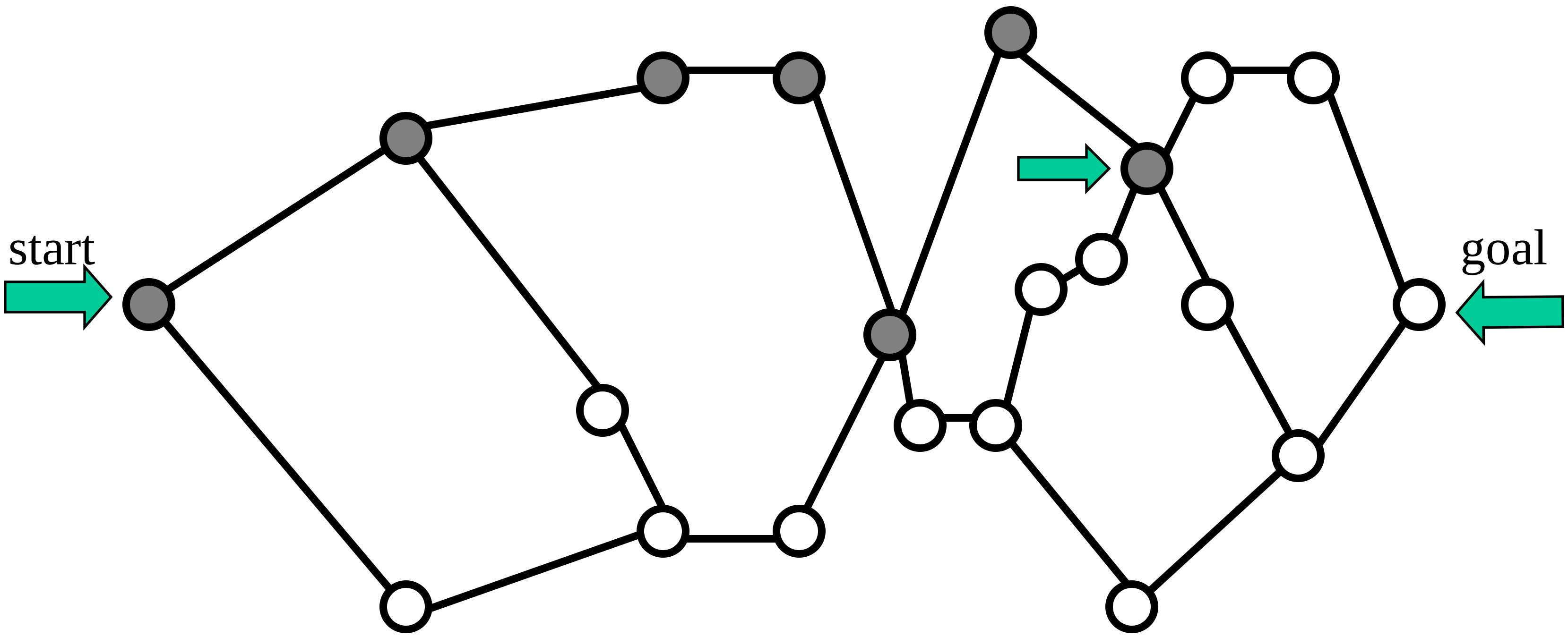
Find a path

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



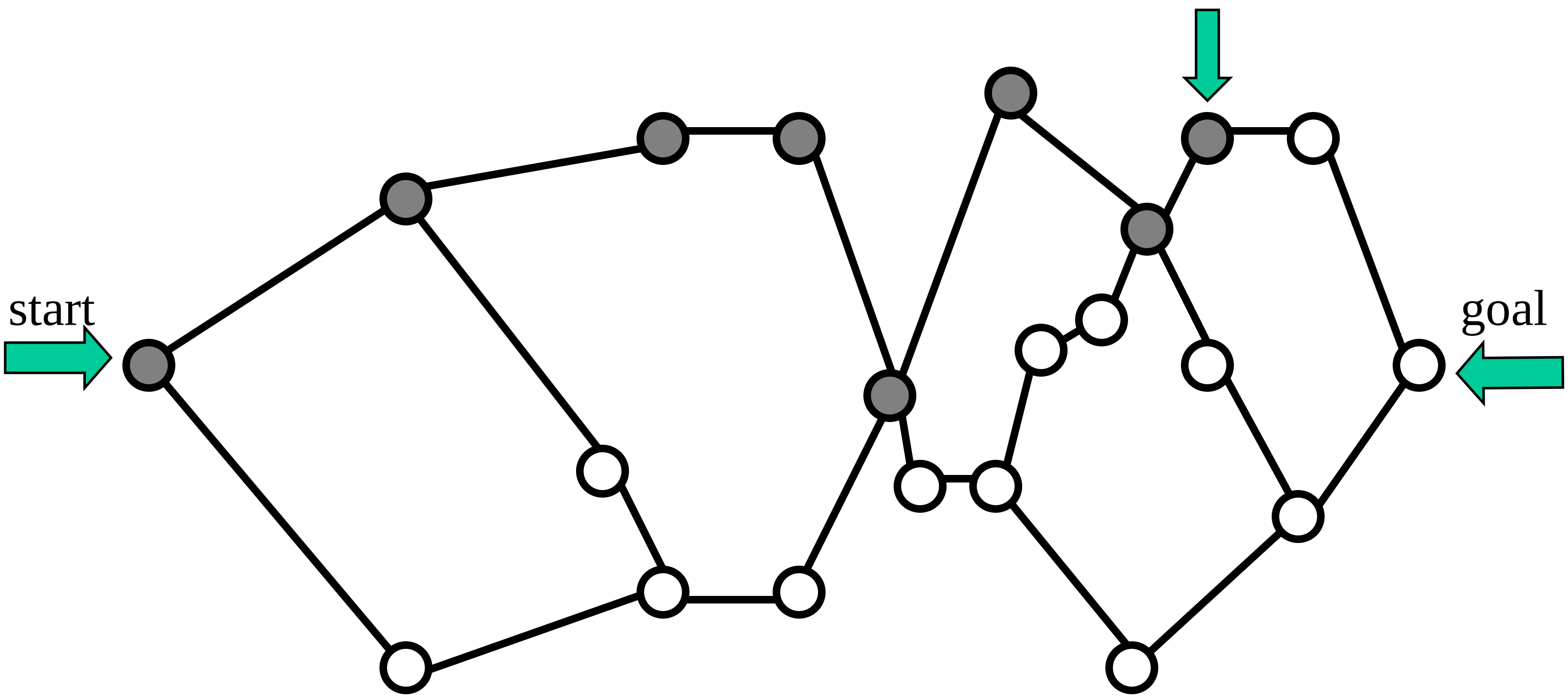
Find a path

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



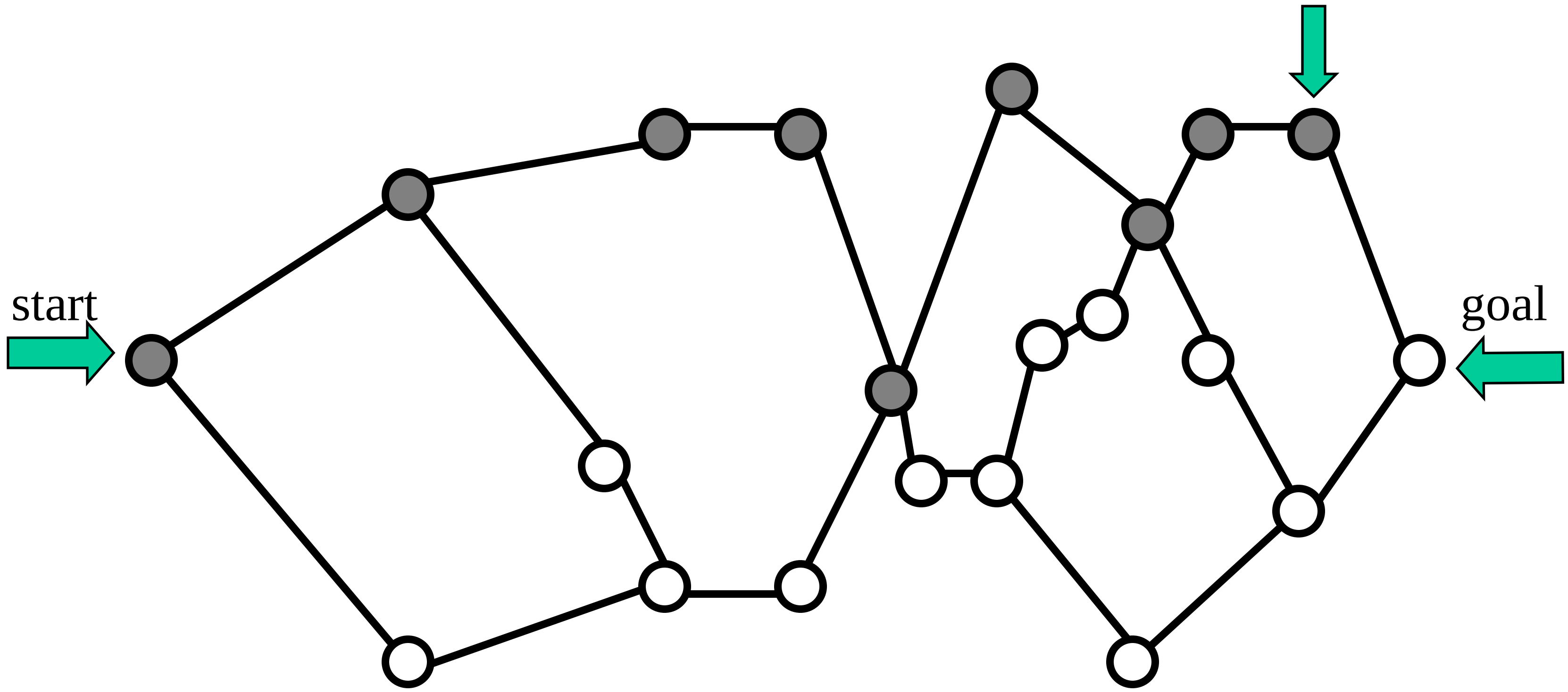
Find a path

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



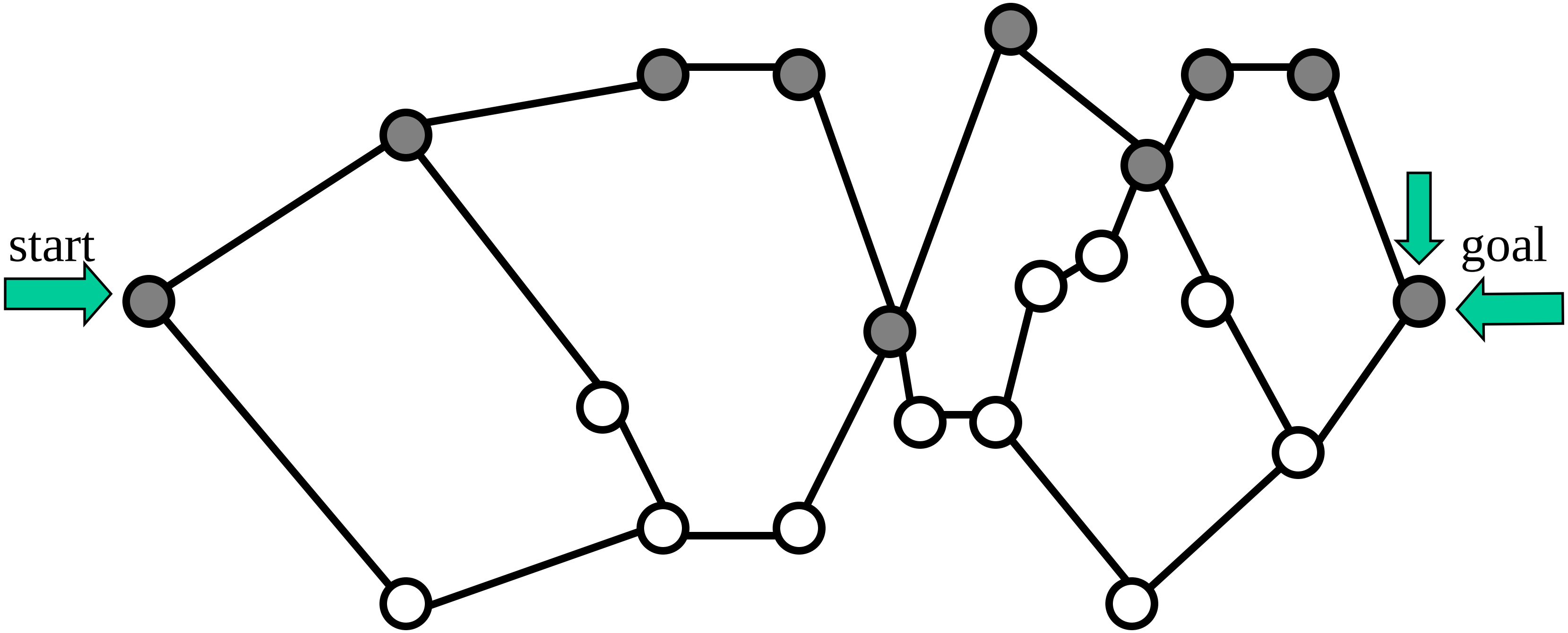
Find a path

- 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

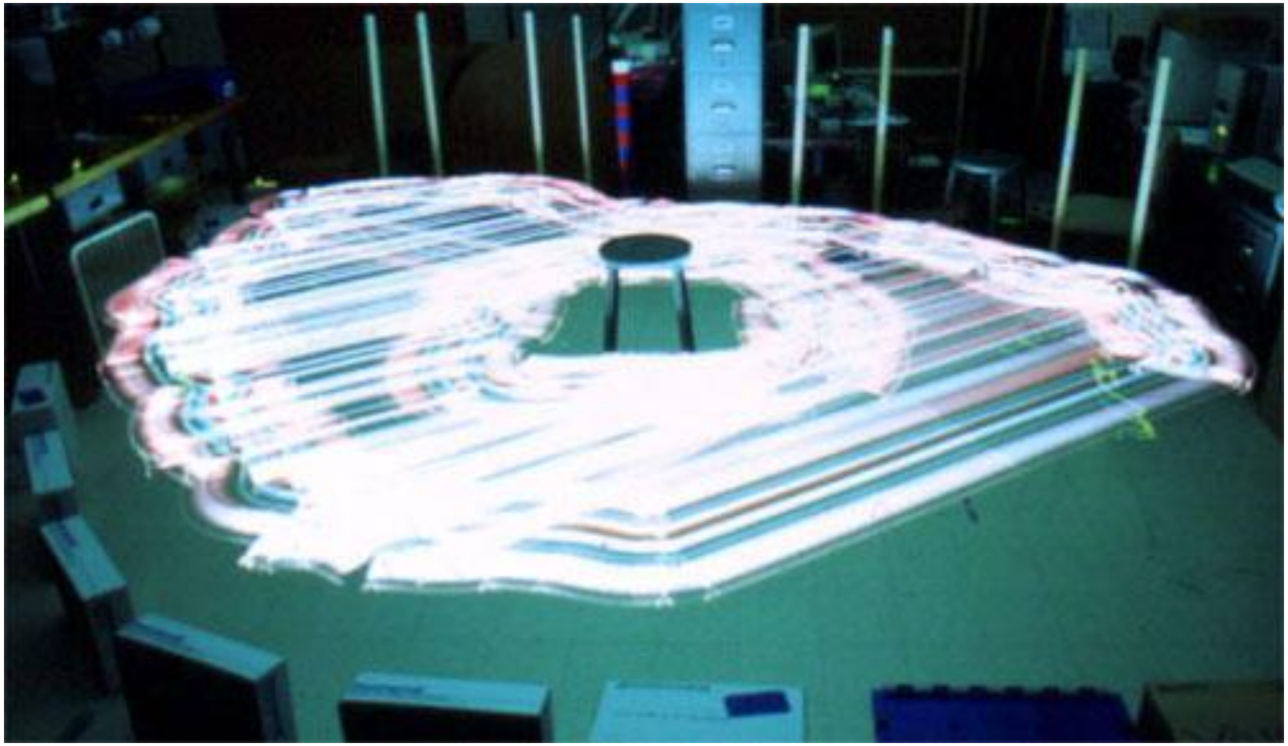


Connect Midpoints of Traps

Applications: Coverage

- First, a distinction between sensor and detector must be made
- *Sensor*: Senses obstacles
- *Detector*: What actually does the coverage
- We’ll be observing the simple case of having an omniscient sensor and having the detector’s footprint equal to the robot’s footprint

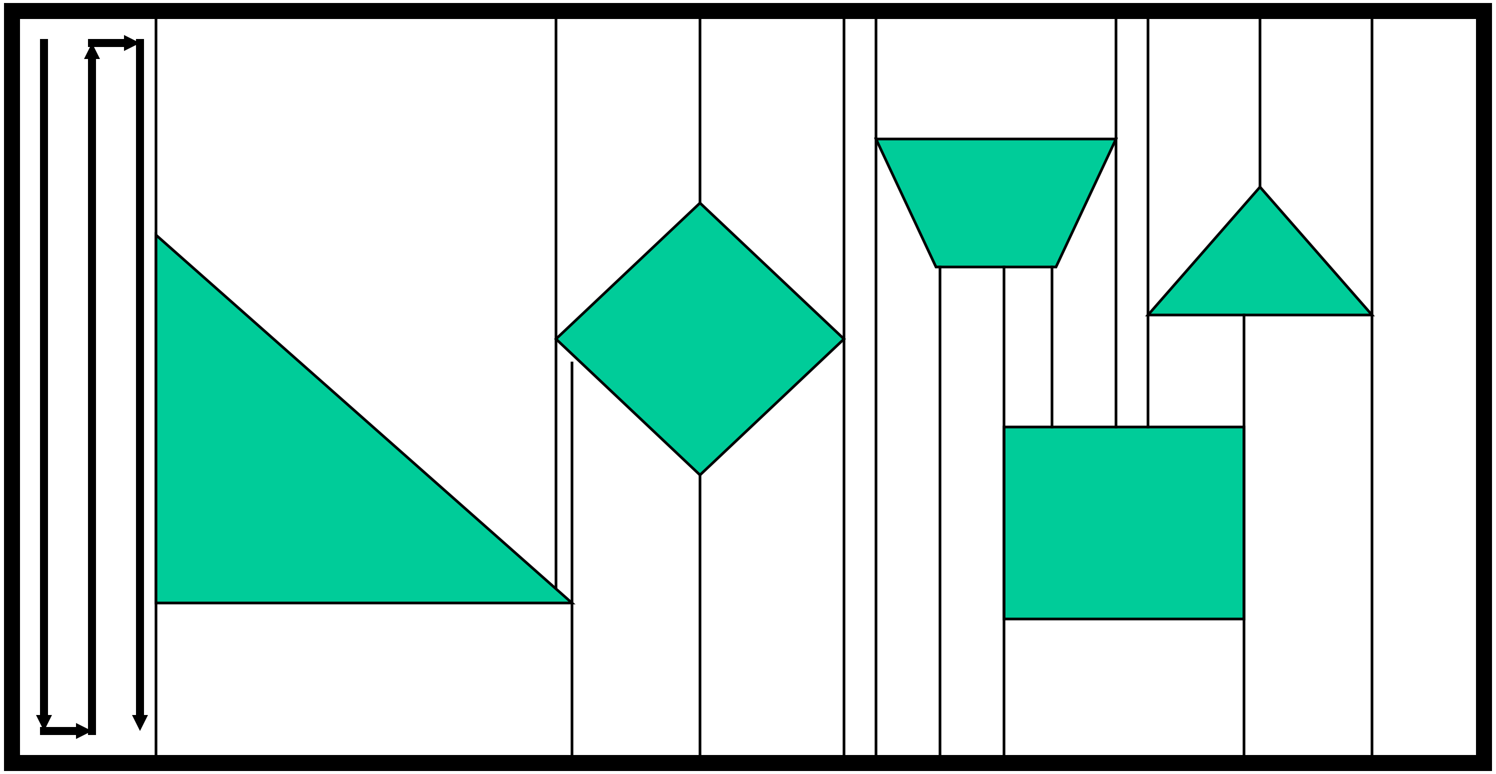
Howie Choset
Robotics Institute



Coverage
(snake robots)

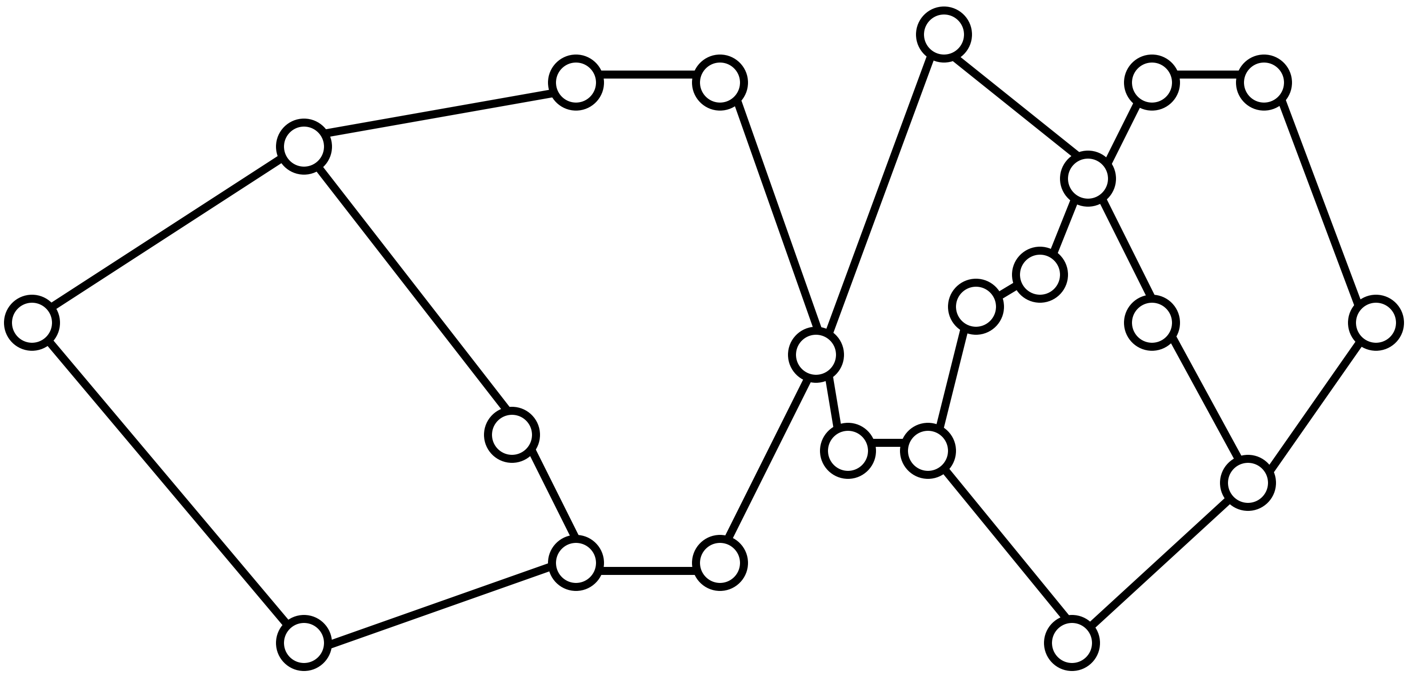
Cell Decompositions: Trapezoidal Decomposition

- How is this useful? Well, trapezoids can easily be covered with simple back-and-forth sweeping motions. If we cover all the trapezoids, we can effectively cover the entire "reachable" world.

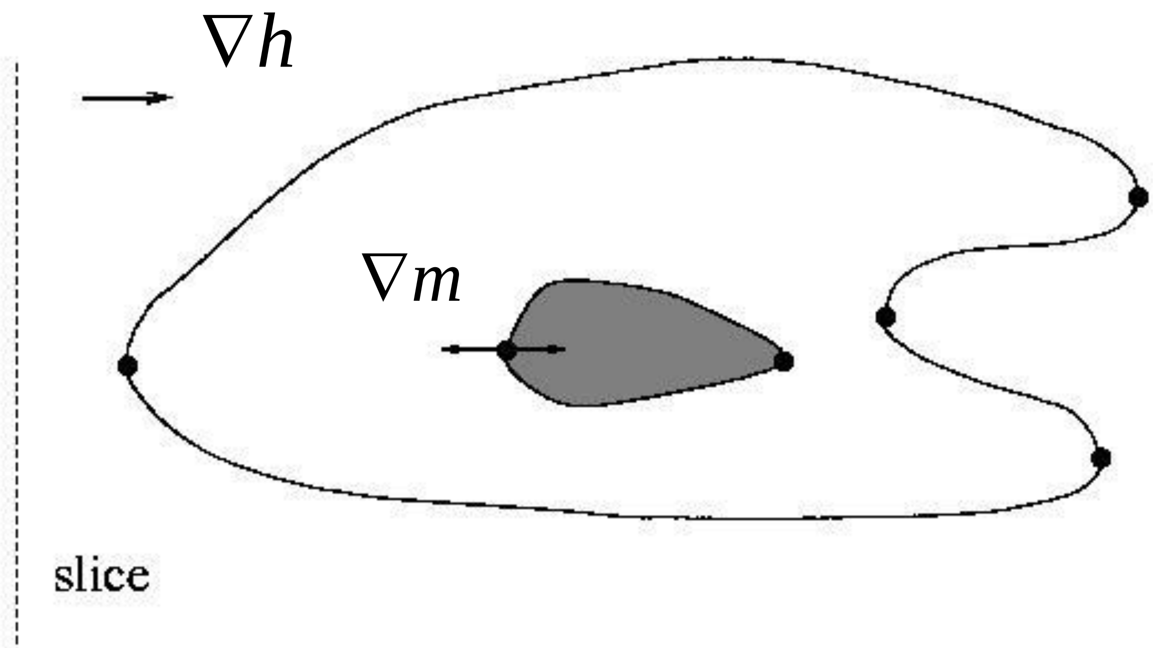


Applications: Coverage

- Simply visit all the nodes, performing a sweeping motion in each, and you're done.



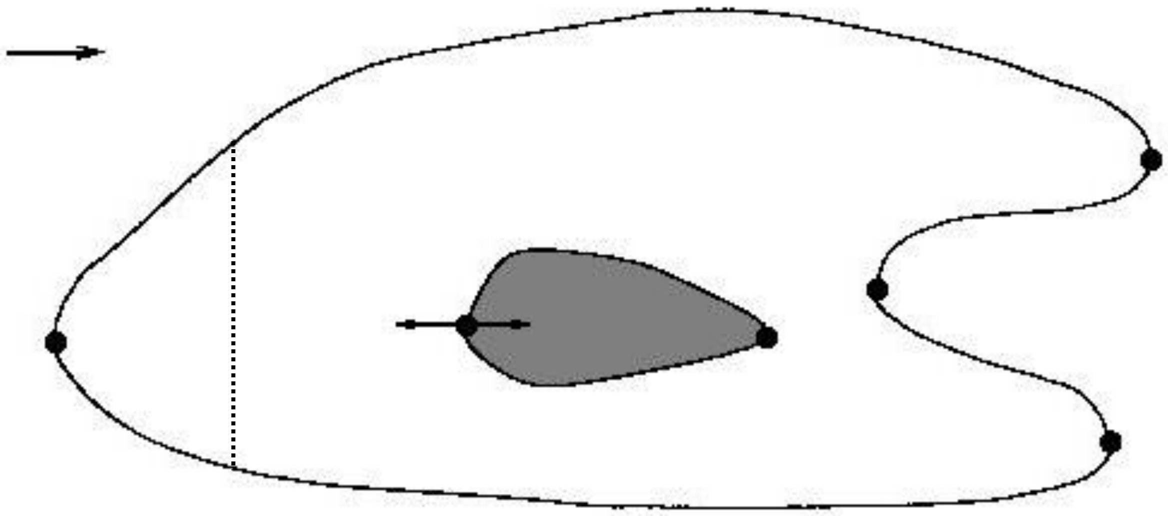
Cell Decomp. in Terms of Critical Points



- *Slice is a level set*

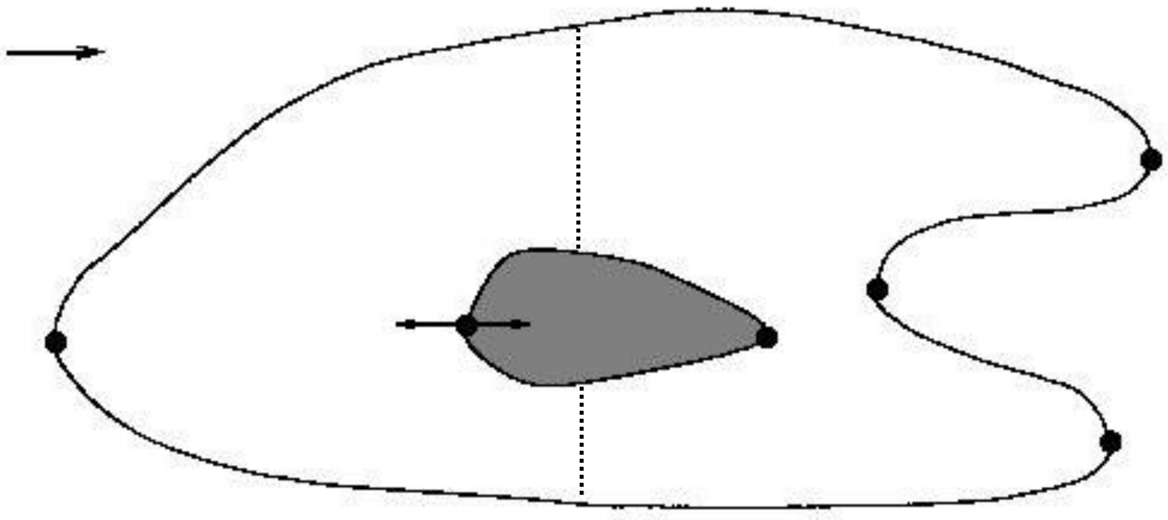
- *Slice function: h(x,y)= x, slice={(x,y)|h(x,y)=λ}*

- *At a critical point x of* $h|_M, \nabla h(x) = \nabla m(x)$ *where M = {x|m(x)=0}*



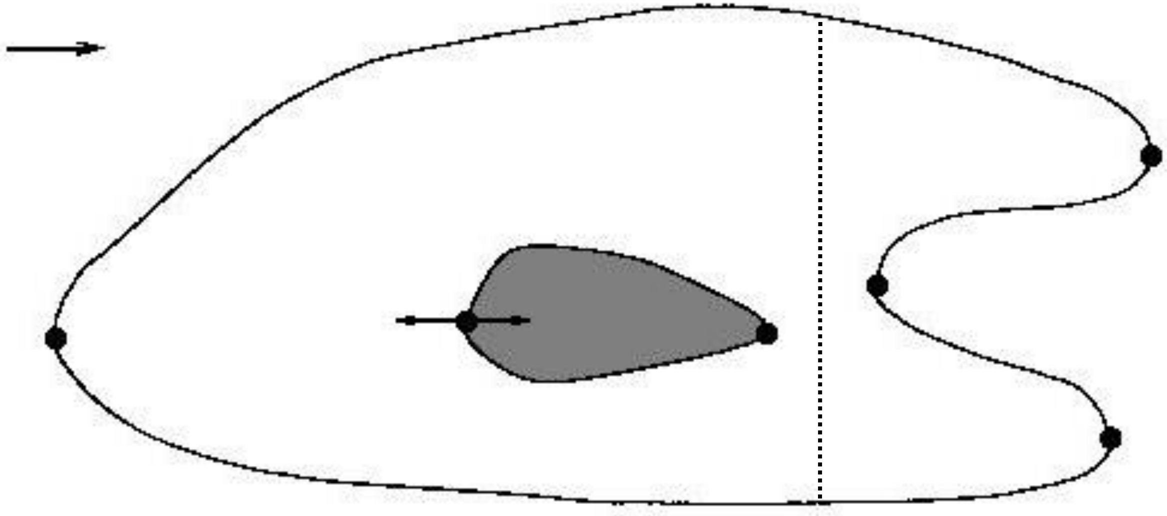
1-connected

$h(x,y) = a1$



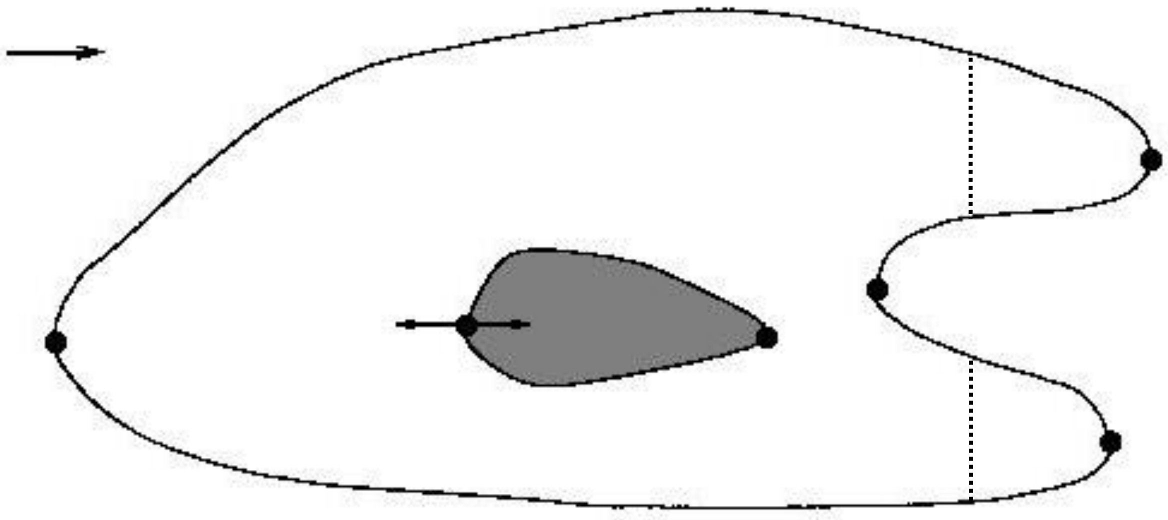
2-connected

h(x,y) = a2



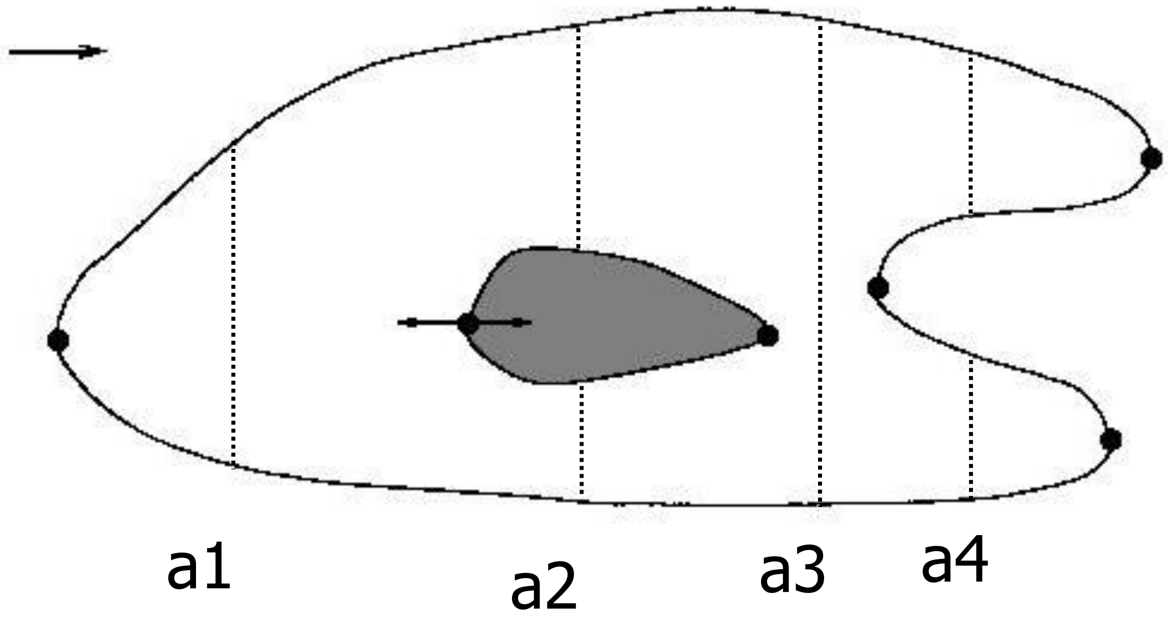
1-connected

h(x,y) = a3

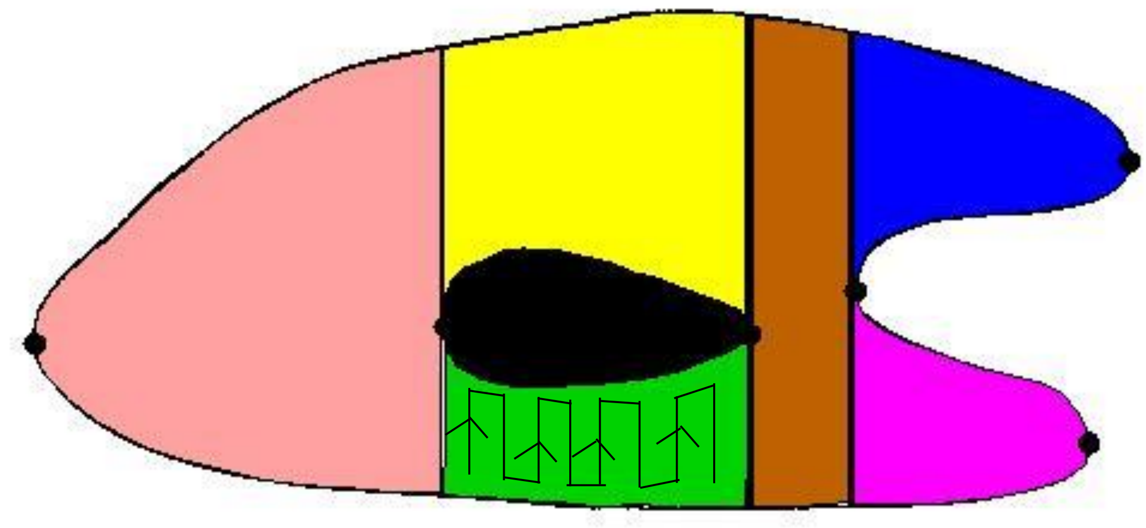


2-connected

h(x,y) = a4

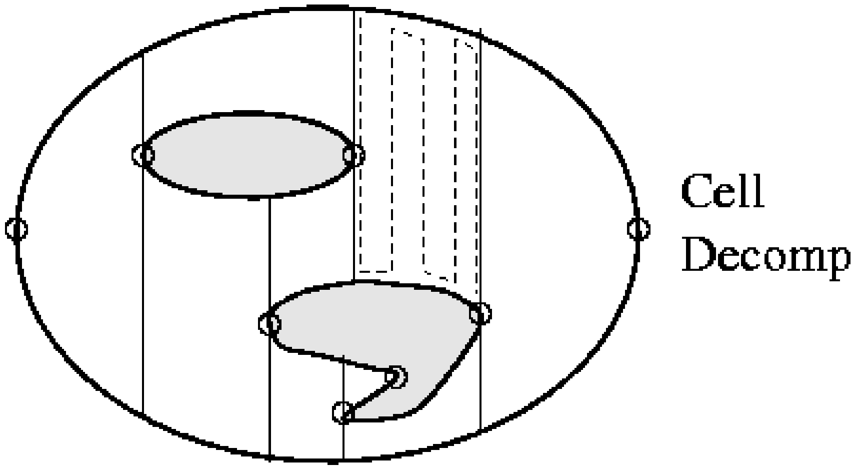


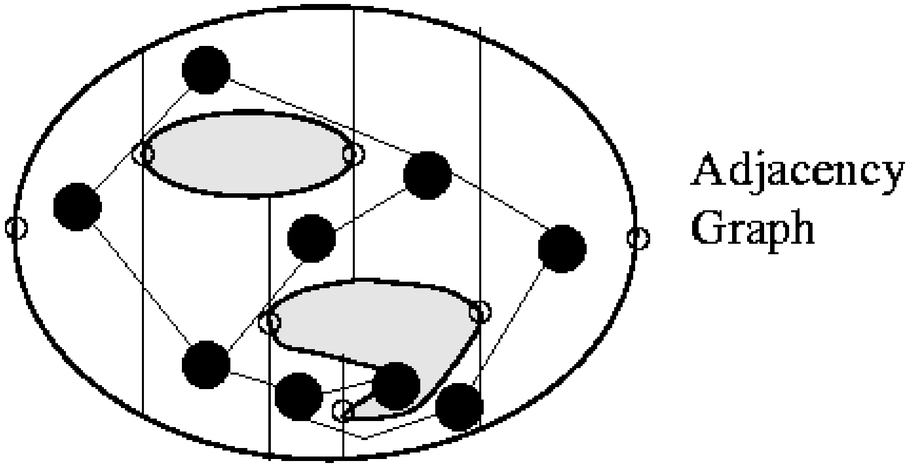
- *Connectivity of the slice in the free space changes at the critical points (Morse theory)*



- *Each cell can be covered by back and forth motions*

Cell-Decomposition Approach





Define Decomposition

- Completeness

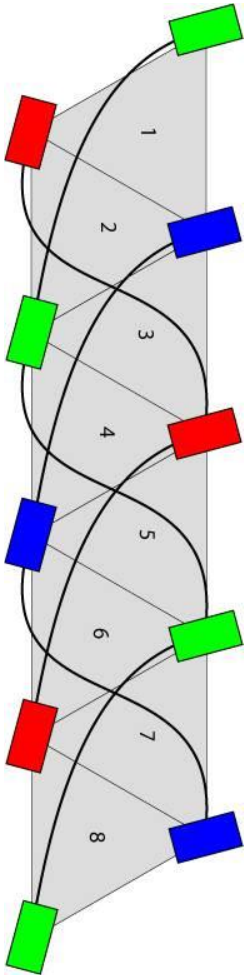
Sensor-based Construction

Define Other Decompositions

- Other patterns
- Extended detector

Provably completeness ≠ guaranteed completeness

Simultaneous Coverage* and Localization



*mapping too

25m x 30m

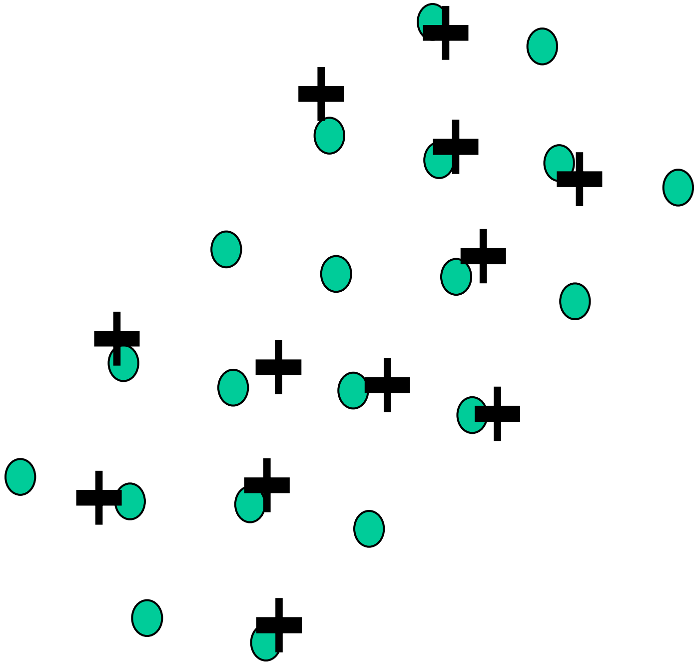
Successful Experiment:
Stopped because of robot battery limitations

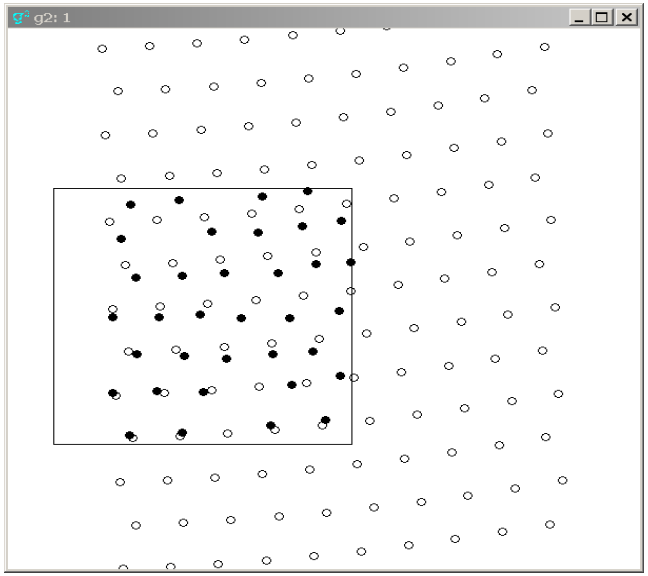
Operational Hierarchy

Path Planning	Where to cover and map
Position Planning	How to collaboratively move and maintain low position error (w/o GPS)
Local Planning	Obstacle detection/avoidance Terrain features to inform Path Planner

Calibrate robots' initial location and go

Probabilistic Coverage





Surface Deposition

- Process Variables
  - Uniformity
  - Waste
  - Positioning
- Cycle-time
  - Time-to-completion
  - Programming time

Conclusion: Complete Overview

- ☑ The Basics
  - Motion Planning Statement
  - The World and Robot
  - Configuration Space
  - Metrics
- ☑ Path Planning Algorithms
  - Start-Goal Methods
    - Lumelsky Bug Algorithms
    - Potential Charge Functions
    - The Wavefront Planner
  - Map-Based Approaches
    - Generalized Voronoi Graphs
    - Visibility Graphs
  - Cellular Decompositions => Coverage
- ☑ *Done with Motion Planning!*