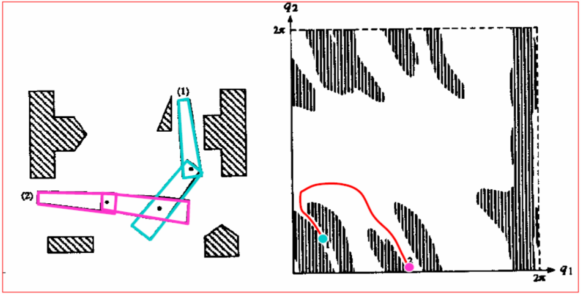
CS 689: Robot Motion Planning

Roadmaps

Amarda Shehu

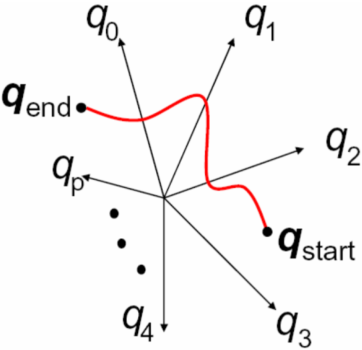
Department of Computer Science
George Mason University

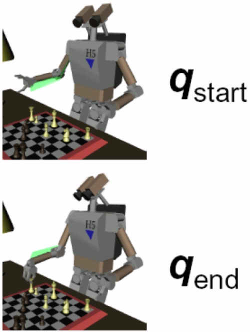
- Application of search approaches, such as A*, stochastic search, and more.
- Search in geometric structures (constrained configuration space)
- Spatial Reasoning
- Challenges
  - Continuous state space
  - Vast, high-dimensional configuration space for searching



- The problem is reduced to finding the path of a point robot through configuration space by expanding obstacles.

Motion Planning Problem





- A = robot with p dofs in 2D or 3D workspace
- CB = set of obstacles
- A configuration q is legal if it does not cause the robot to intersect the obstacles
- Given start and goal configurations, q_{start}, q_{goal}, find a continuous sequence of legal configurations from q_{start} to q_{goal}.
- Report failure if no valid path is found.

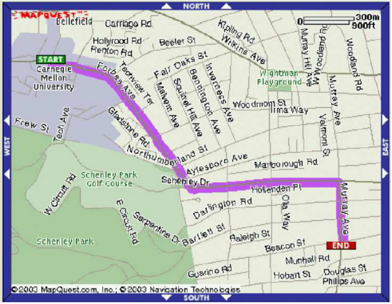
From Formal Guarantees to Practical Algorithms

- Formal result not useful for practical algorithms[1]: A path (if it exists) can be found in time exponential in p and polynomial in m and d.
  - p: dimension of c-space
  - m: number of polynomials describing free c-space
  - d: maximum degree of the polynomials

- *In practical approaches: reduce intractable problem in continuous c-space into tractable problem in a discrete space, where then one can use all standard techniques for path finding, such as A*, stochastic search, and more.*

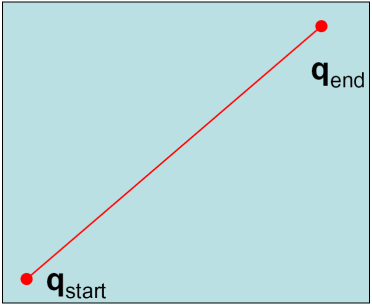
- Basic Approaches:
  - Roadmaps: Visibility graphs vs. Voronoi diagrams
  - Cell decomposition
  - Potential fields
- Extensions
  - Sampling techniques
  - Online algorithms

[1] J. Canny. "The complexity of Robot Motion Planning Plans." MIT Ph.D. Dissertation, 1987.

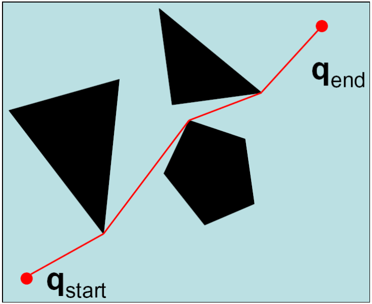


General Idea:

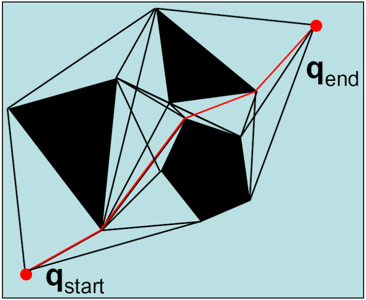
- Avoid searching entire space
- Pre-compute a (hopefully small) graph (the roadmap) such that staying on the "roads" is guaranteed to avoid the obstacles.
- Find a path between q_{start} and q_{goal} by using the roadmap.



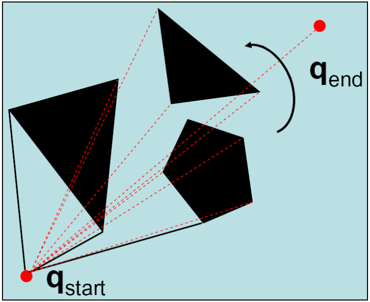
In the absence of obstacles, the best path is the straight line between q_{start} and q_{goal}.



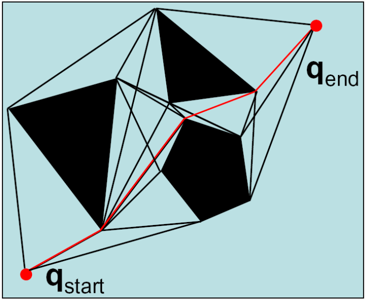
- Assuming polygonal osbtacles, it looks like the shortest path is a sequence of straight lines joining the vertices of the obstacles.
- Is this always true?



- Visibility graph G = set of unblocked lines between vertices of the obstacles, q_{start}, and q_{goal}
- A node P is lined to a node P' if P' is visible from P
- Solution = shortest path in visibility graph G.



- Sweep a line originating at each vertex
- Record those lines that end at visible vertices.

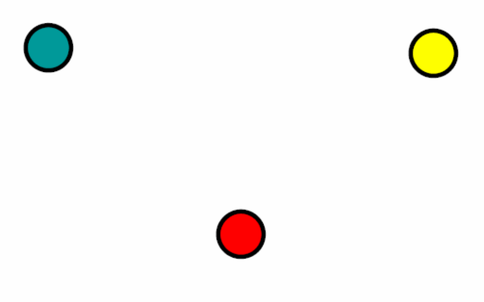


- Let N = total number of vertices of the obstacle polygons
- Naive: $O(N^3)$
- Sweep: $O(N^2 \cdot lg(N))$
- Optimal: $O(N^2)$

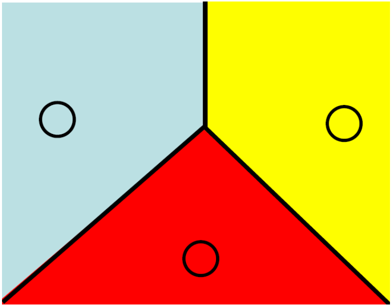
Visibility Graphs: Weaknesses

- Shortest path but:
  - Tries to stay as close as possible to obstacles
  - Any execution error will lead to a collision
  - Complicated in more than 2 dimensions
- We may not care about strict optimality so long as we find a safe path. Staying away from obstacles is more important than finding the shortest path
- Need to define other types of "roadmaps"

Voronoi Diagrams

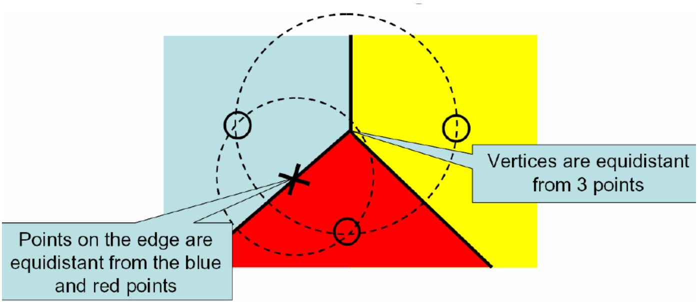


- Given a set of data points in the plane:
  - Color the entire plane such that the color of any point in the plane is the same as the color of its nearest neighbor

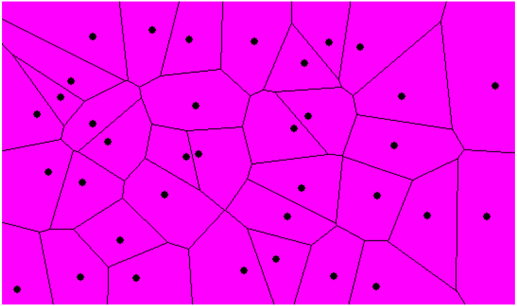


- Voronoi diagram = Set of line segments separating regions corresponding to different colors
  - Line segment = points equidistant from 2 data points
  - Vertices = points equidistant from more than 2 data points

Voronoi Diagrams

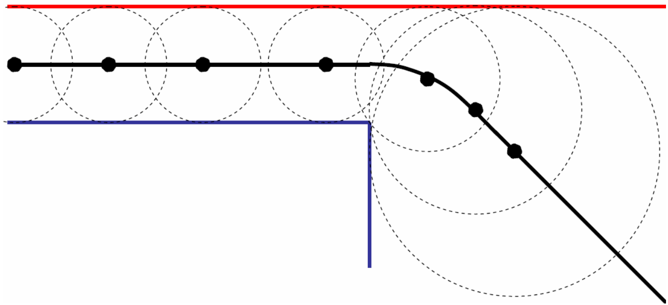


- Voronoi diagram = Set of line segments separation regions corresponding to different colors
  - Line segment = points equidistant from 2 data points
  - Vertices = points equidistant from more than 2 data points

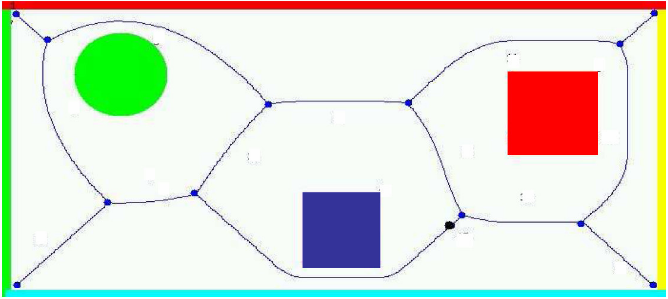


- Complexity (in the plane):
- $O(N \cdot logN)$ time
- $O(N)$ space
- See htpp://www.cs.cornell.edu/Info/People/chew/Delaunay.html for interactive demo

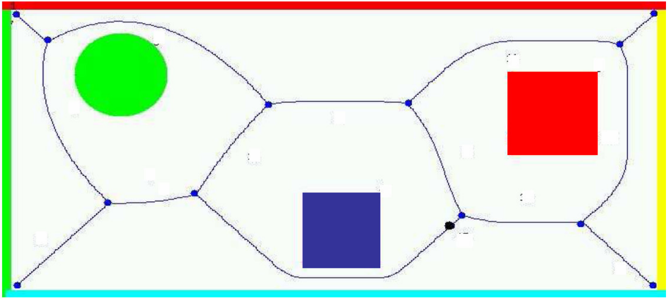
Voronoi Diagrams: Beyond Points



- Edges are combinations of straight line segments and segments of quadratic curves
- Straight edges: Points equidistant from 2 lines
- Curved edges: Points equidistant from one corner and one line



- Key property: Points on edges of Voronoi diagram are furthest from obstacles
- Idea: Construct a path between q_{start} and q_{goal} by following edges on Voronoi diagram
- Use Voronoi diagram as roadmap graph instead of visibility graph



- Find point q^*_{start} of the Voronoi diagram closest to q_{start}
- Find point q^*_{goal} of the Voronoi diagramn closest to q_{goal}
- Compute shortest path from q^*_{start} to q^*_{goal} on the Voronoi diagram

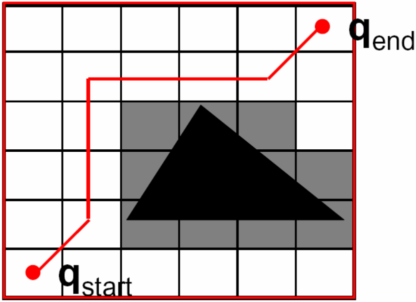
Voronoi Diagrams: Weaknesses

- Difficult to compute in higher dimensions or non-polygonal worlds
- Approximate algorithms exist
  Use of Voronoi is not necessarily best heuristic (stay away from obstacles)
- It can lead to paths that are much too conservative
- Can be unstable: that is, small changes in obstacle configuration can lead to large changes in the diagram

Cell Decomposition

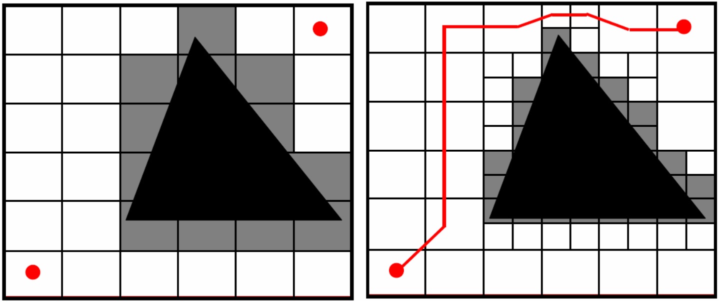
- Key Idea: Decompose c-space into cells so that any path inside a cell is obstacle-free
- Approximate vs. Exact Cell Decomposition

Approximate Cell Decomposition



- Define discrete grid in c-space
- Mark any cell of the grid that intersects configuration space obstacles as blocked
- Find path through remaining cells by using, for instance, A* (using Euclidean distance as heuristic)
- Cannot be complete as described. Why?

Approximate Cell Decomposition

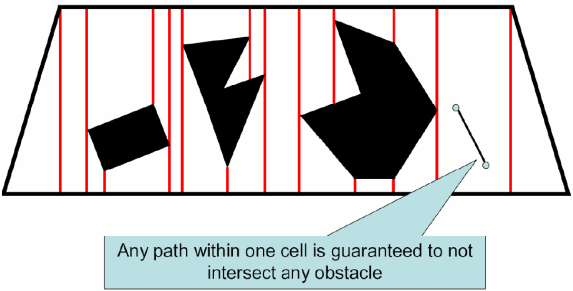


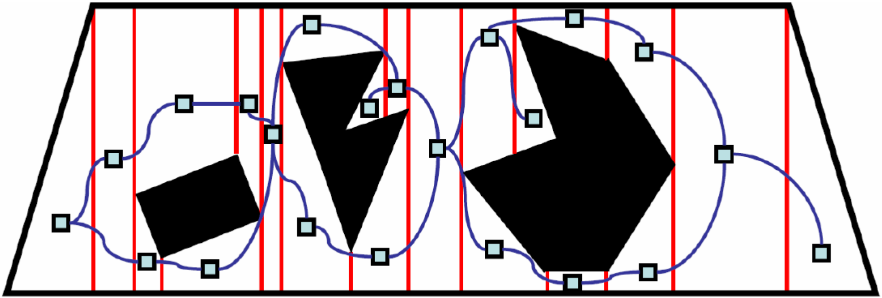
- Cannot find path in this case, even though one exists
- Solution:
- Distinguish between
  - Cells that are entirely contained in some configuration space obtacle (FULL) and
  - Cells that partially intersect configuration space obstacles (MIXED)
- Try to find path using current set of cells
- If no path found:
  - Subdivide MIXED cells again and try with new set of cells
  - UNTIL some reasonable cell size and then stop with failure

Approximate Cell Decomposition: Limitations

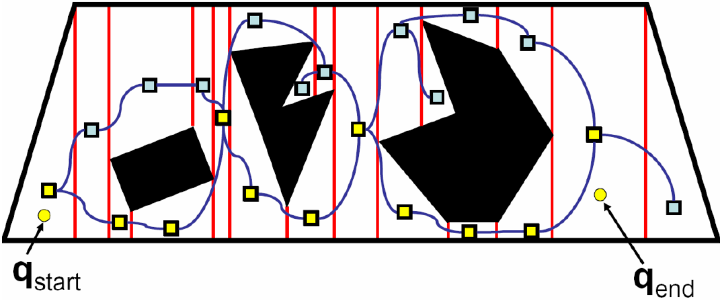
- Good:
  - Limited assumption on obstacle configuration
  - Approach used in practice
  - Finds obvious solutions quickly
- Bad:
  - No clear notion of optimality ("best" path)
  - Trade-off completeness/computation
  - Still difficult to employ in high dimensions

Exact Cell Decomposition



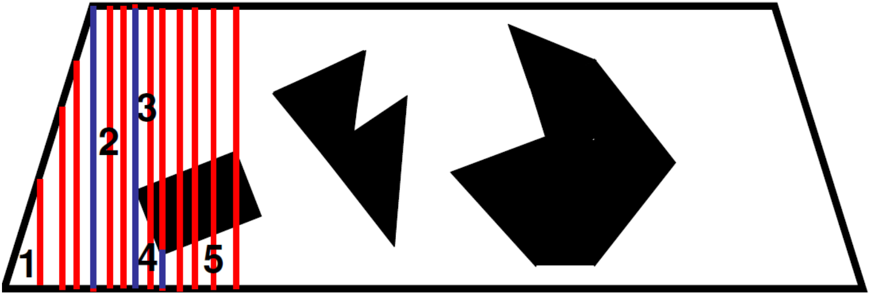


- Graph of cells defines a roadmap



- Graph can be used to find a path between any two configurations

Exact Cell Decomposition

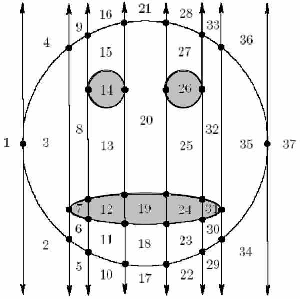


- Critical Event 1: Create new cell
- Critical Event 2: Split cell

Plane Sweep Algorithm

- Initialize current list of cells to empty
- Order vertices of configuration space obstacles along the x direction
- For every vertex:
  - Construct plane at corresponding x location
  - Depending on type of event:
    - Slit current cell into 2 new cells OR
    - Merge two current cells
  - Create a new cell
- Complexity in 2D:
  - Time: $O(N \cdot logN)$
  - Space: $O(N)$

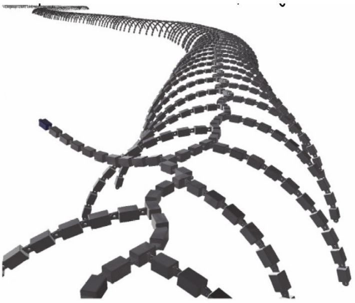
Exact Cell Decomposition



- A version of exact cell decomposition can be extended to higher dimensions and non-polygonal boundaries (cylindrical cell decomposition)
- Provides exact solution; thus, completeness
- Expensive and difficult to implement in higher dimensions

Potential Fields

See previous lecture.



Millipede-like robot (S. Redon) has close to 13,000 dofs.

Dealing with C-space Dimension

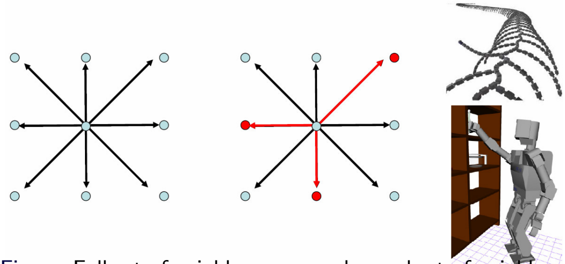


Figure: Full set of neighbors vs. random subset of neighbors

- We should evaluate all neighbors of current state, but:
- Size of neighborhood grows exponentially with dimension
- Very expensive in high dimensions

Solution:

- Evaluate on random subset of K neighbors
- Move to lowest potential neighbor

Draw away:

- Completely describing and optimally exploring C-space is too hard in high dimensions and not necessary
- Focus on finding a good sampling of C-space. So, probabilistic motion planning!