Robotics Planning

Henrik I Christensen

Path Planning

- A path is a continuous mapping in C

$$\pi : [0, L] \rightarrow S_{free}$$

- Where L is the length of the path
- The path is collision from if for all t

$$\pi(t) \in S_{free}$$

Query / problem definition

- A problem or query is
  - Given two states q_0 and q_f
- Determine if there is a collision-free path between q_0 and q_f

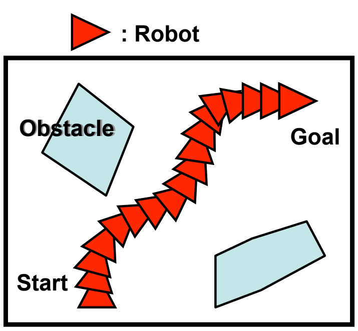
Path Planning

<u>Given:</u>

- World geometry
- Robot's geometry
- Start and goal configuration

<u>Compute:</u>

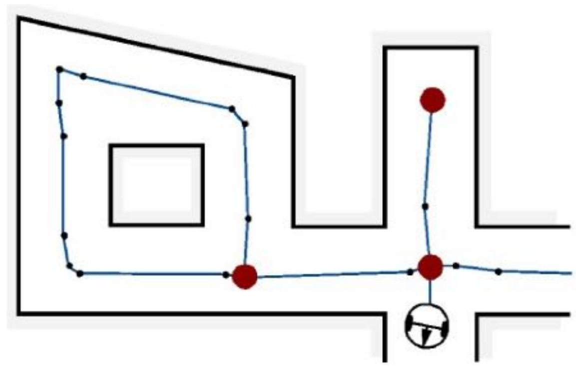
A collision-free, feasible path to the goal



Source: F. Moss, Rice

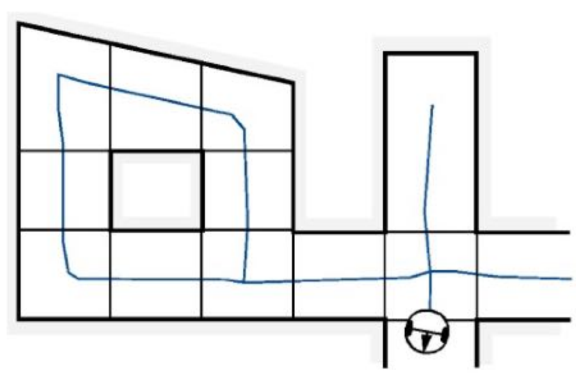
Roadmaps

- Roadmaps / Graphs
- How do we select the key nodes?



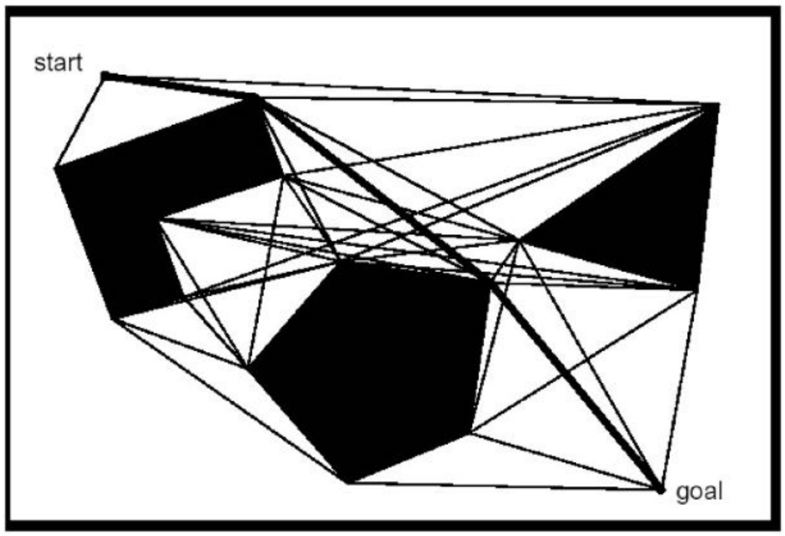
Space decomposition

- Partition
  - Free-space
  - Obstacles
- Generation of a tessellation



Visibility Graph

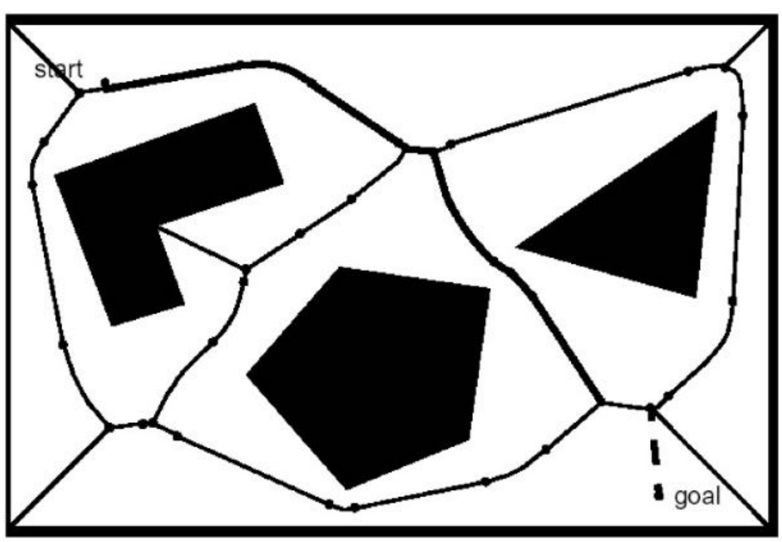
- Connect visible vertices in space
- Generate a search across the resulting graph



(c) Henrik I Christensen

Voronoi Graph

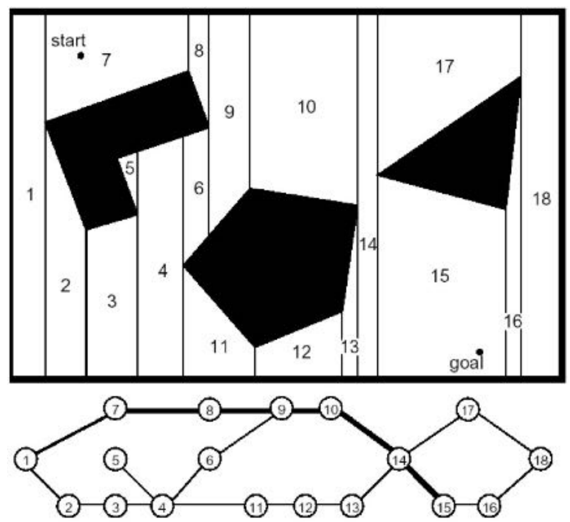
- Compute maximum distance to boundary points
- Search for shortest path along the graph



(c) Henrik I Christensen

Cell decomposition

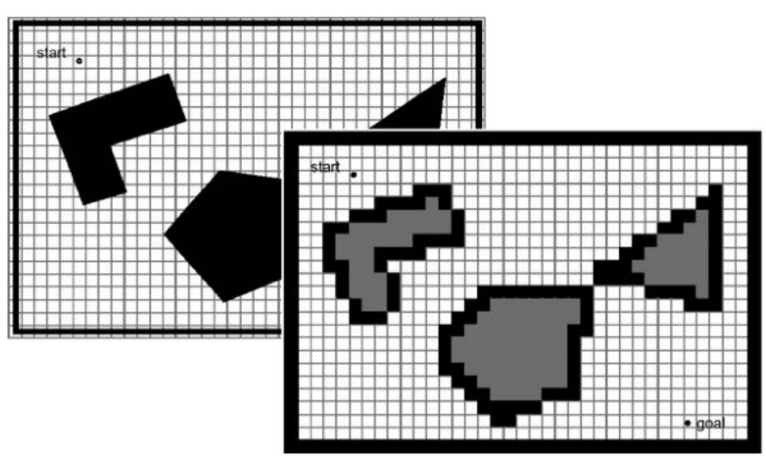
- Divide free space (S_{free}) into single connected regions termed cells
- Generate connectivity graph
- Local Goal and Start cells
- Search the graph
- Generation a motion strategy



(c) Henrik I Christensen

Approximate cell decomposition

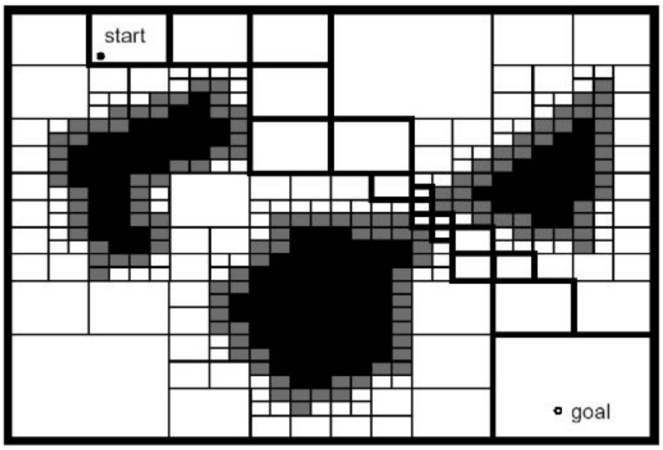
- Easy to implement
- Use of standard methods for search such as wavefront & distance
- Widely used in simple environments



(c) Henrik I Christensen

Adaptive cell decomposition

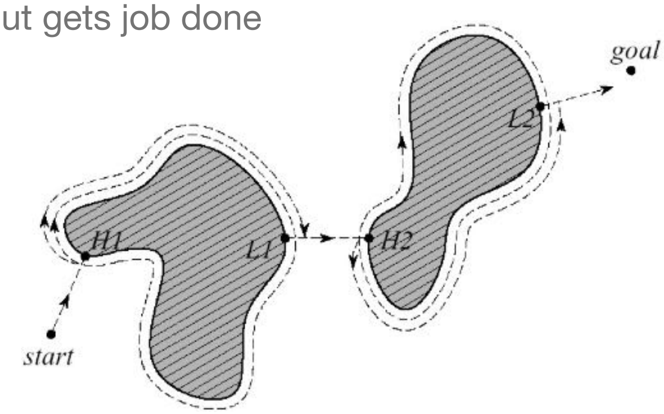
- Efficient representation of space
- Quad-trees are well-known from computational geometry
- Suited for sparse workspaces



(c) Henrik I Christensen

Planning basics

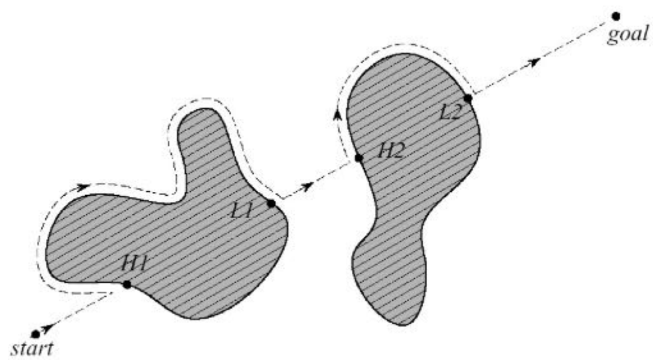
- The Bug family of algorithms [Lumelski]
- Simplest possible strategy
- Traverse obstacles
- Leave a point of minimum distance
- Inefficient but gets job done



(c) Henrik I Christensen

Bug-2 the obvious improvement

- Do traversal but leave at point to goal point
- Efficient in open spaces
- Mazes a challenge



Potential fields

- Consider the robot a particle in a potential field
- Goal serves as an attractor
- Obstacles represents repellors
- When the potential field is differentiable the force is

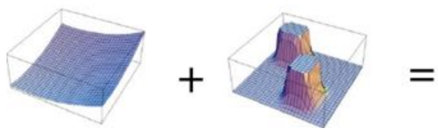
  $$F(q) = -\nabla U(q)$$

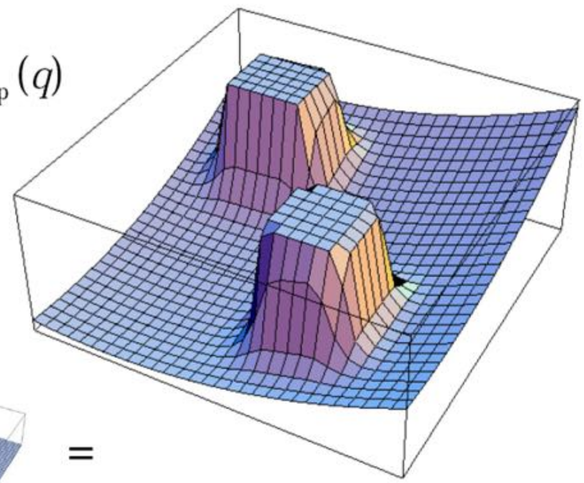
  specifies locally the direction of motion
- Potential fiedls can be represented by harmonics
  - Superposition principle specifies
  - Goal dynamics can be represented by a potential field
  - Each obstacle is a potential field
  - The sum of the parts is still a potential field

Potential fields

$$U(q) = U_{\text{att}}(q) + U_{\text{rep}}(q)$$

$$F(q) = -\nabla U(q)$$

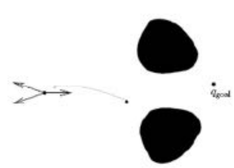




Source: IROS tutorial, 2011

Potential fields

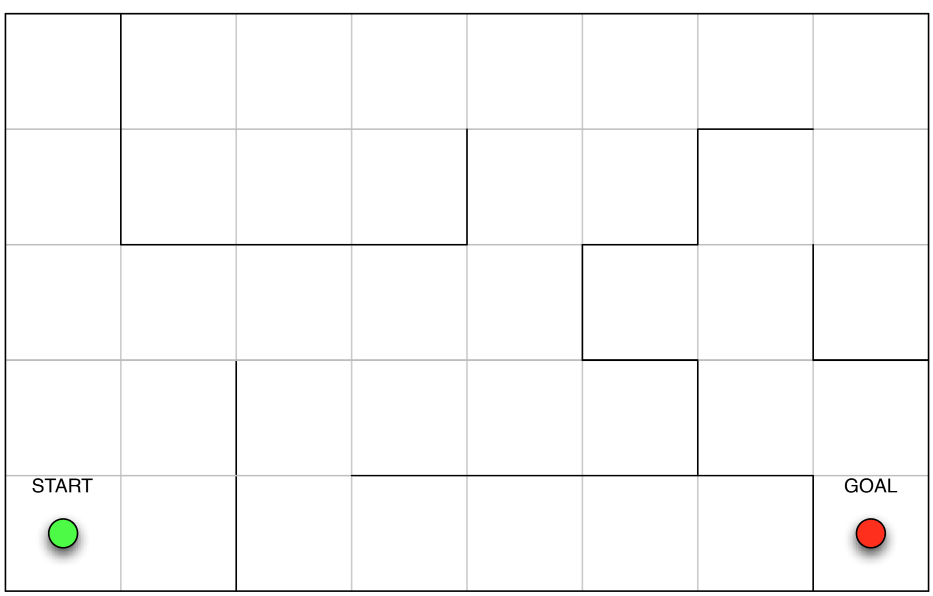
- Potential fields are known to experience local minima



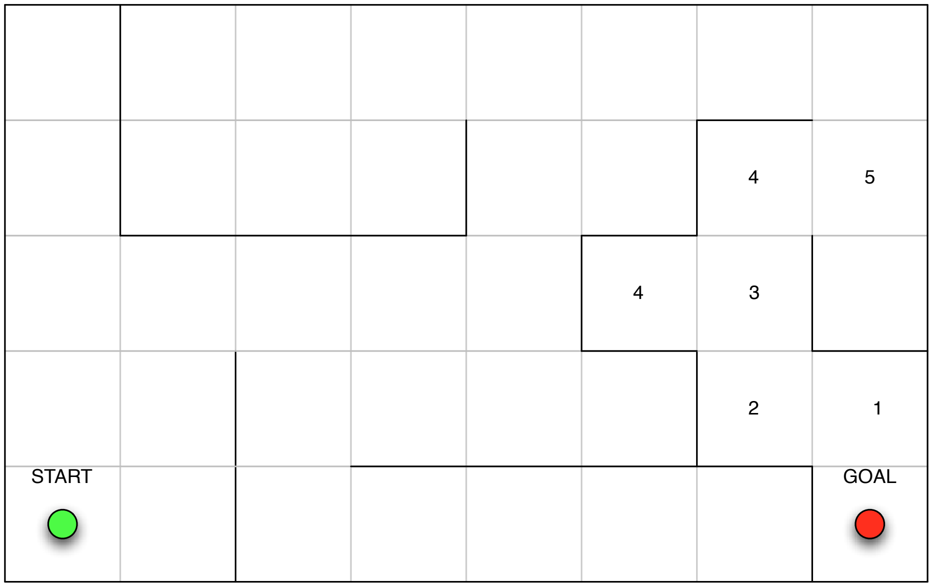
Wavefront propagation

- Consider the world as a an homogenous grid
- Cells are free or occupied / or with walls
- Search from start to goal
- Neighbor metrics can be used to define behavior

Wavefront propagation

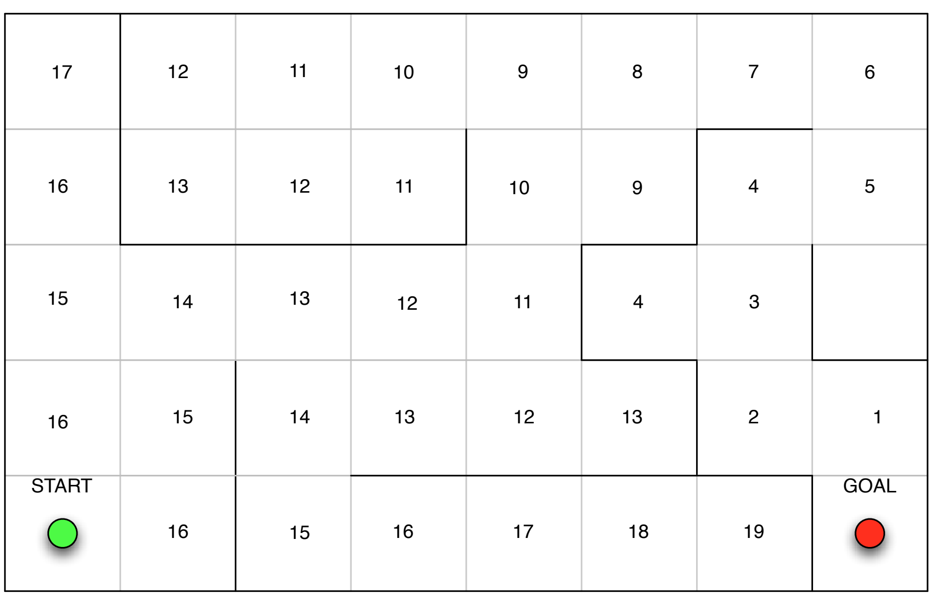


Wavefront propagation



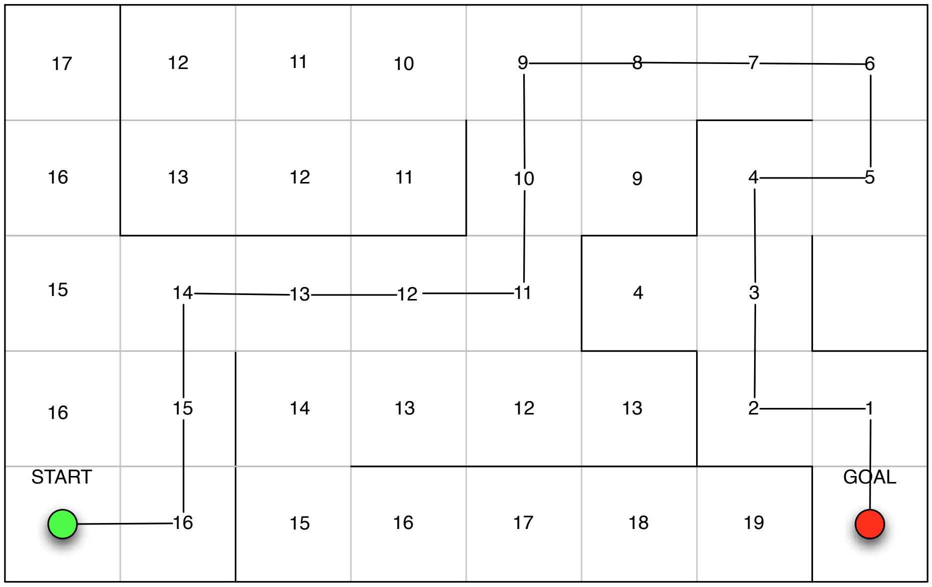
(c) Henrik I Christensen

Wavefront propagation



(c) Henrik I Christensen

Wavefront propagation



Graph search using A*

- A* is well known as a graph search heuristic based on estimated and actual cost

$$c(\vec{p}) = \alpha \; cc(\vec{s}, \vec{p}) + \beta \; gc(\vec{p}, \vec{g})$$

- where
  - p is present position
  - s is the start position
  - g is the goal position
  - cc is current cost
  - gc is an estimate of the cost of achieving the goal position
  - α, β represents weight factors

Basic tree-search

```
function TREE-SEARCH( problem, fringe) returns a solution, or failure
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST[problem] applied to STATE(node) succeeds return node
        fringe ← INSERTALL(EXPAND(node, problem), fringe)
```

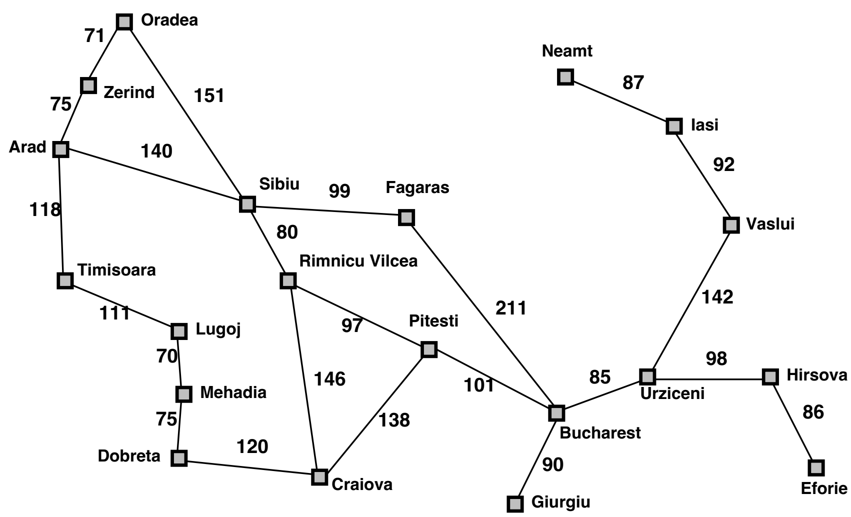
- The challenge is the design of the expand funtion

Source: Russell & Norvig, Artificial Intelligence

Informed search strategies

- The ideal - Best First Search
- Selection of an evaluation function f(n)
- Expand low-cost nodes before higher cost nodes
- Design a heuristic function h(n)
    - Estimated cost of the cheapest path from n to goal

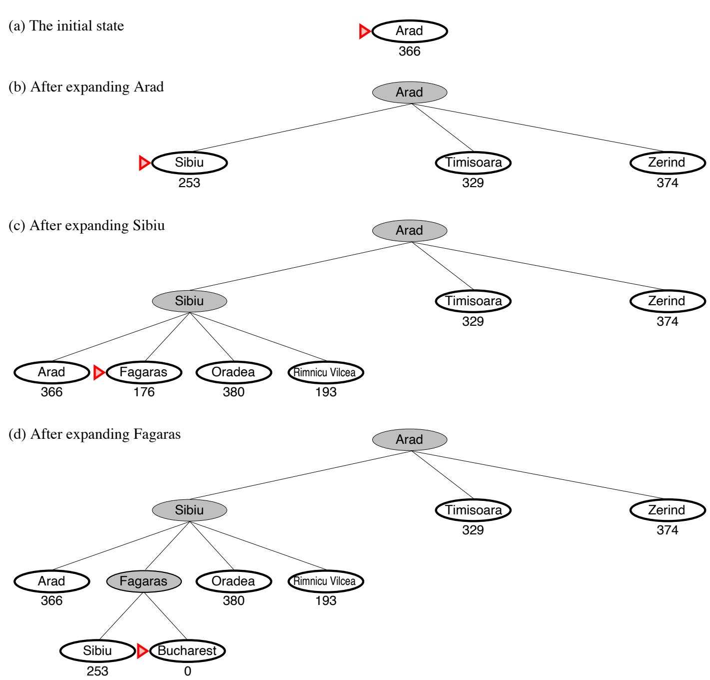
Example of navigation in maps



Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

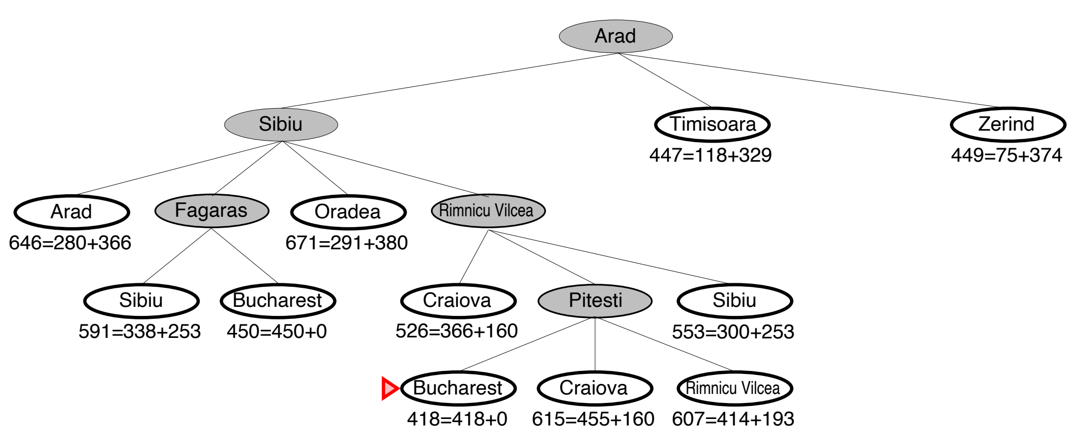
Doing it greedily



Properties of greedy search

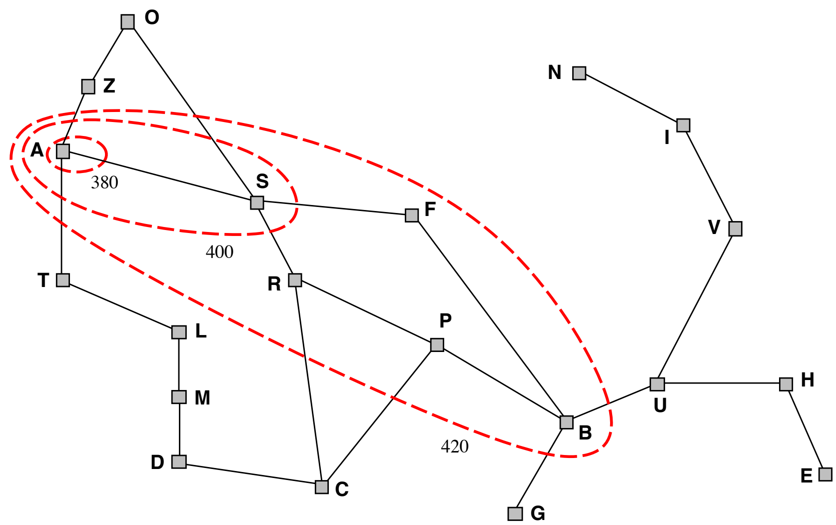
- Completeness: might get stuck in loops
  - Repeated state check needed to break loops
- Time: $O(b^m)$ - a good heuristic can improve performance
- Space $O(b^m)$ - keep all nodes in memory
- Optimal? no greedy is not always optimal

A* search



A* optimality?

- Increase nodes according to f value
- Gradually adds f contours to nodes (a la breadth first w. layers)



A* properties

- Complete? Yes
- Time: exponential in h - accuracy * h*(start)
- Space: all nodes in memory
- Optimal: Yes

- Variation of A*
  - Iterative deepening A* (IDA)
  - Recursive best first (RBDF)
  - Memory bounded A* (MA)
  - Simple MA (SMA)

(c) Henrik I Christensen

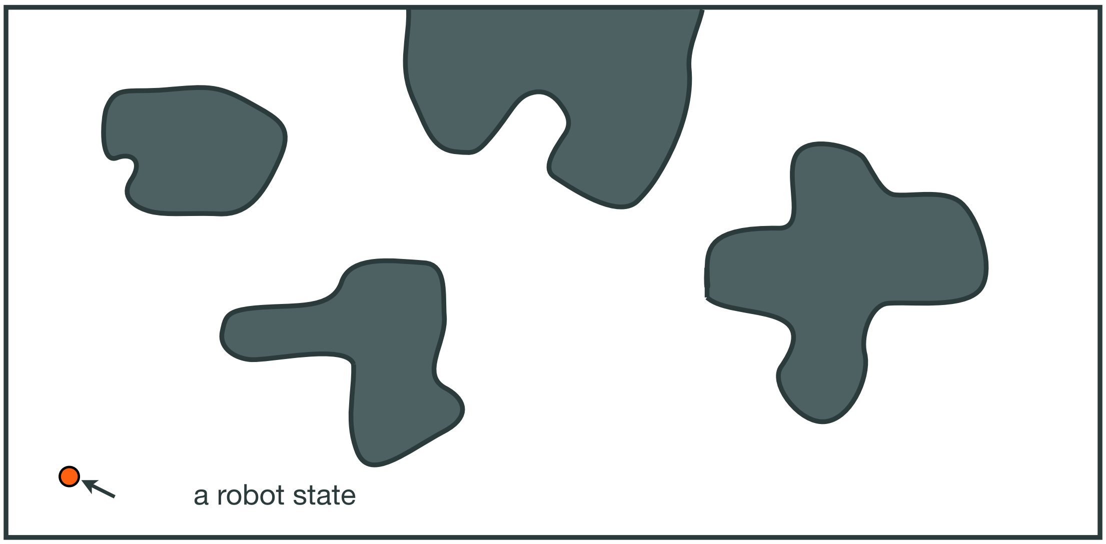
What are our options?

Method	Advantage	Disadvantage
Exact	theoretically insightful	impractical
Cell Decomposition	easy	does not scale
Control-Based	online, very robust	requires good trajectory
Potential Fields	online, easy	slow or fail
Sampling-based	fast and effective	cannot recognize impossible query

Why randomized planners?

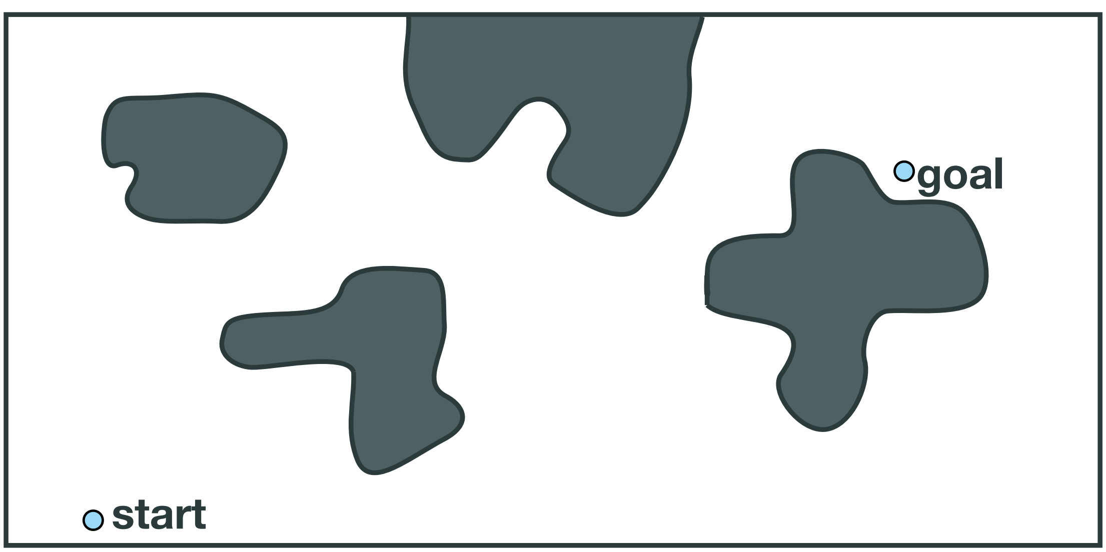
- The structure of the C-space can be highly complex
- The space can be high dimensional 6+

Point robot in 2-D



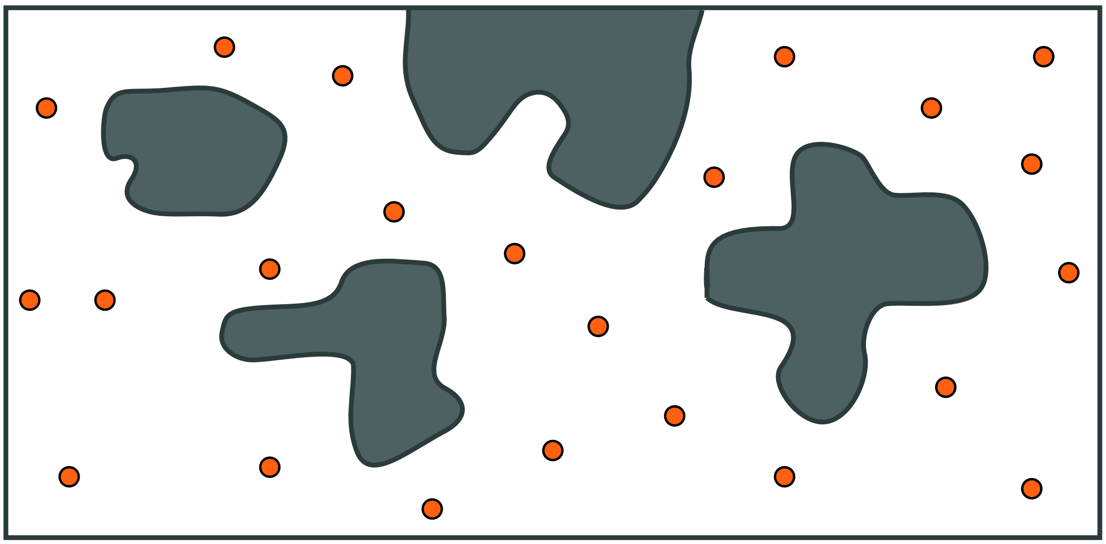
Source: L. Kavraki, RICE - Tutorial

Sample based tree planner



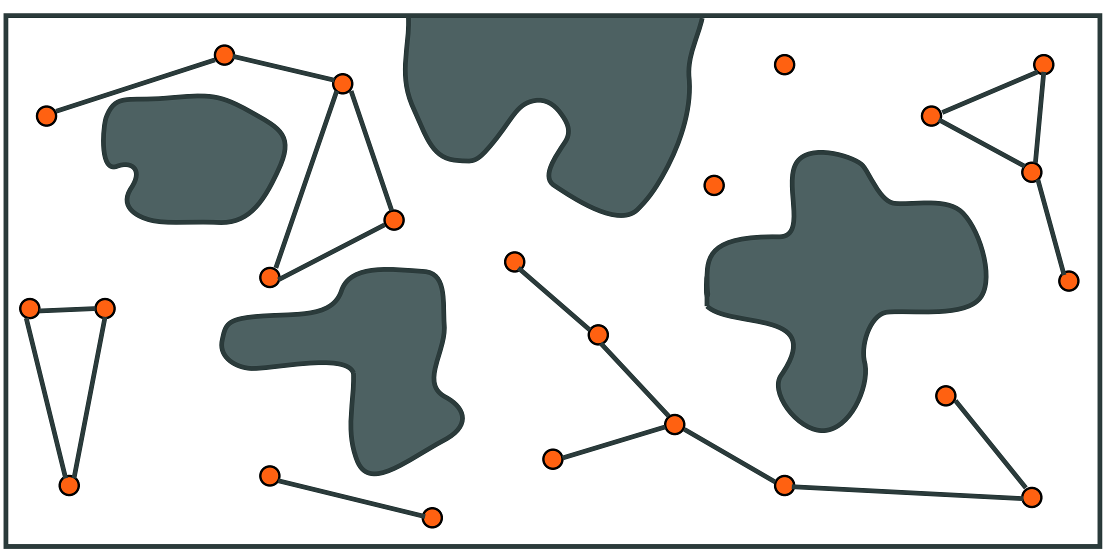
Source: L. Kavraki, RICE - Tutorial

Probabilistic Roadmaps (PRM)



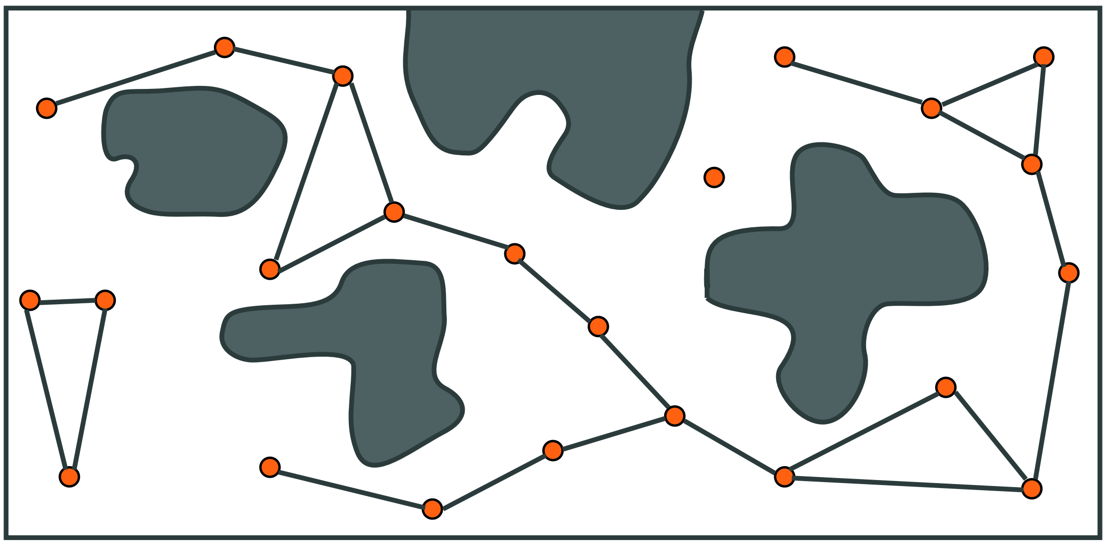
Source: L. Kavraki, RICE - Tutorial

Probabilistic Roadmaps (PRM)



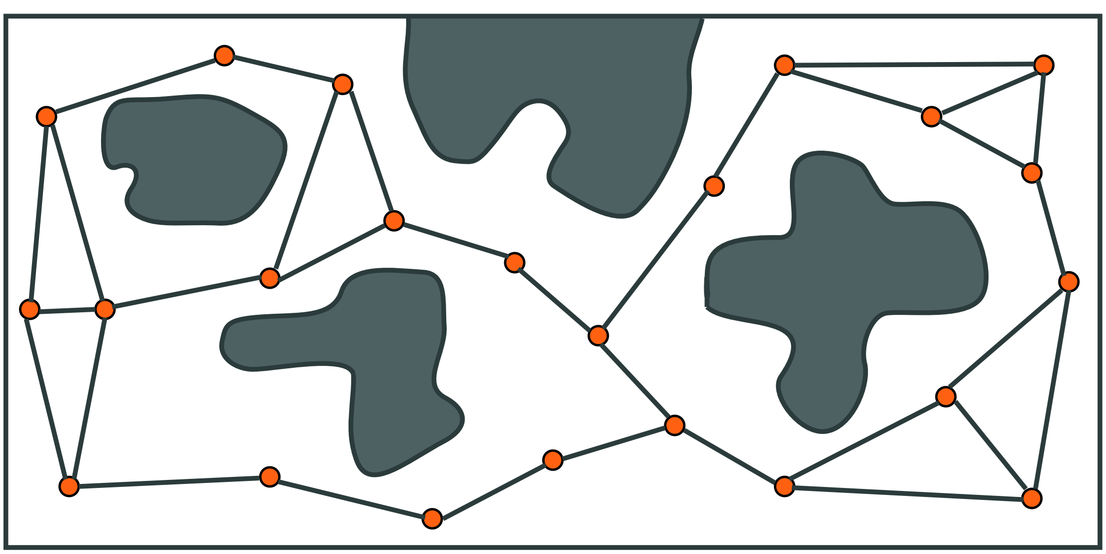
Source: L. Kavraki, RICE - Tutorial

Probabilistic Roadmaps (PRM)



Source: L. Kavraki, RICE - Tutorial

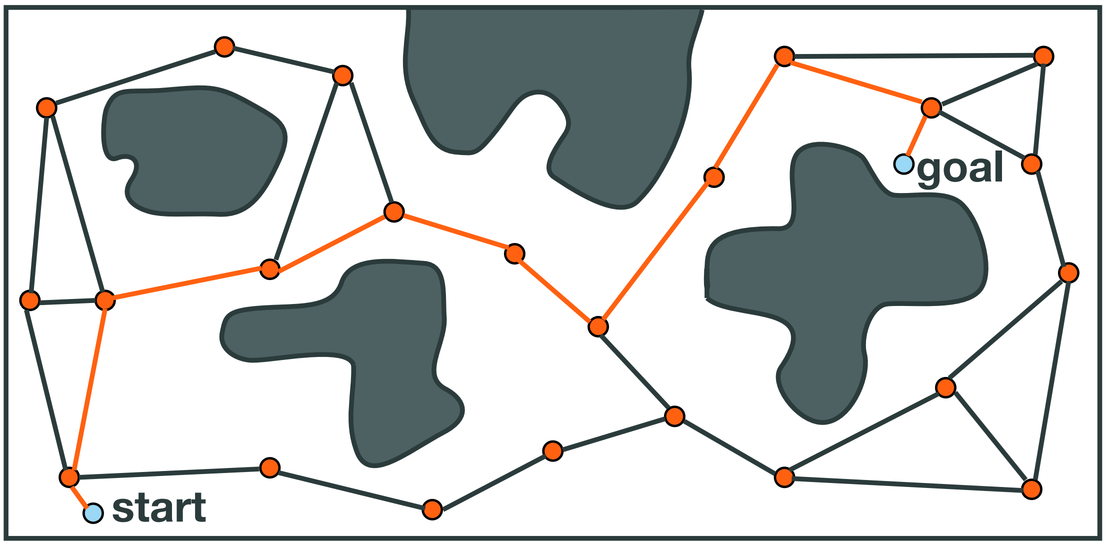
Probabilistic Roadmaps (PRM)



Source: L. Kavraki, RICE - Tutorial

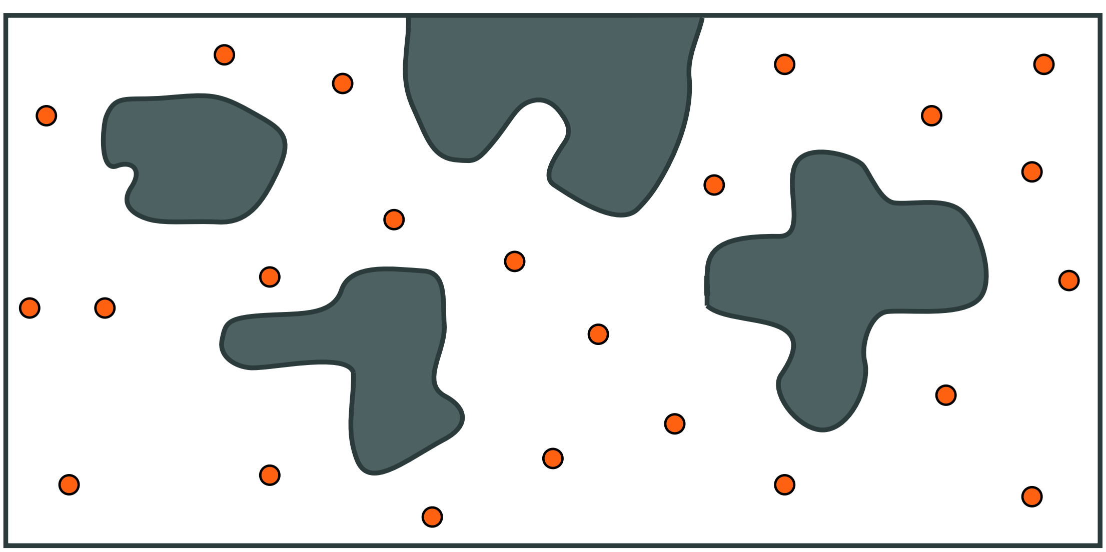
(c) Henrik I Christensen

Answering queries



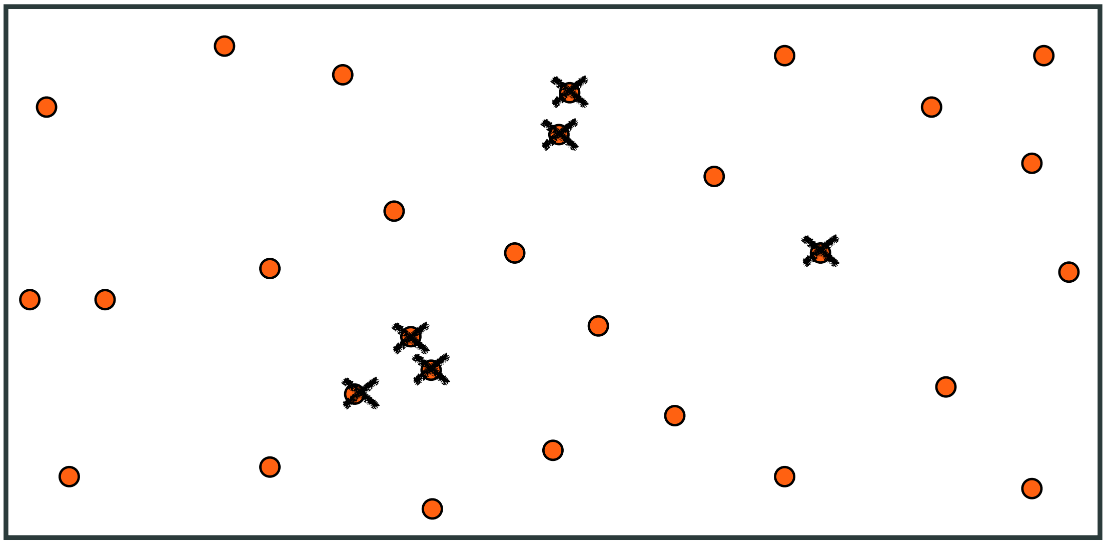
Source: L. Kavraki, RICE - Tutorial

Operations of a PRM



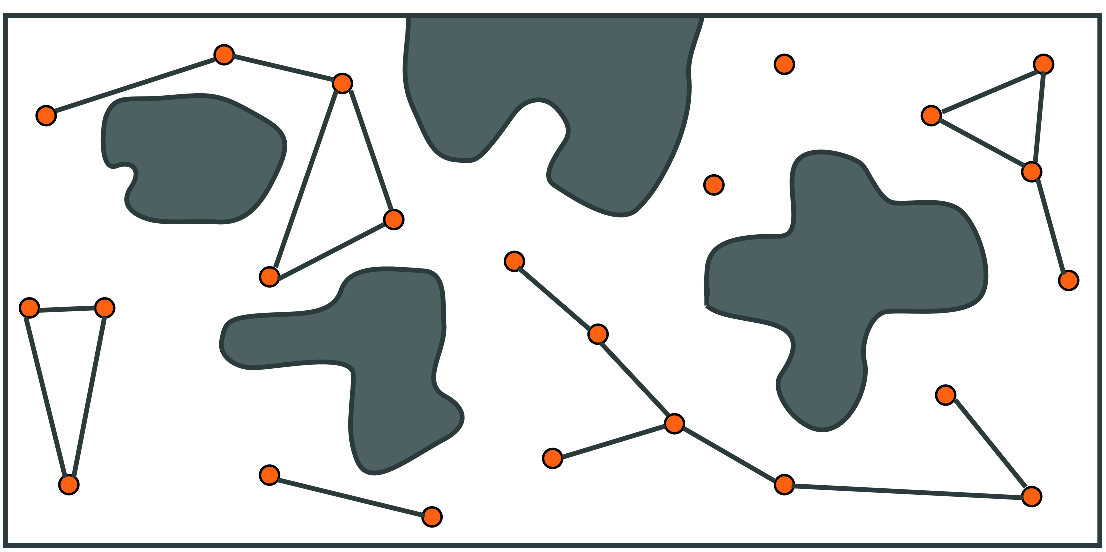
Source: L. Kavraki, RICE - Tutorial

Operations of a PRM



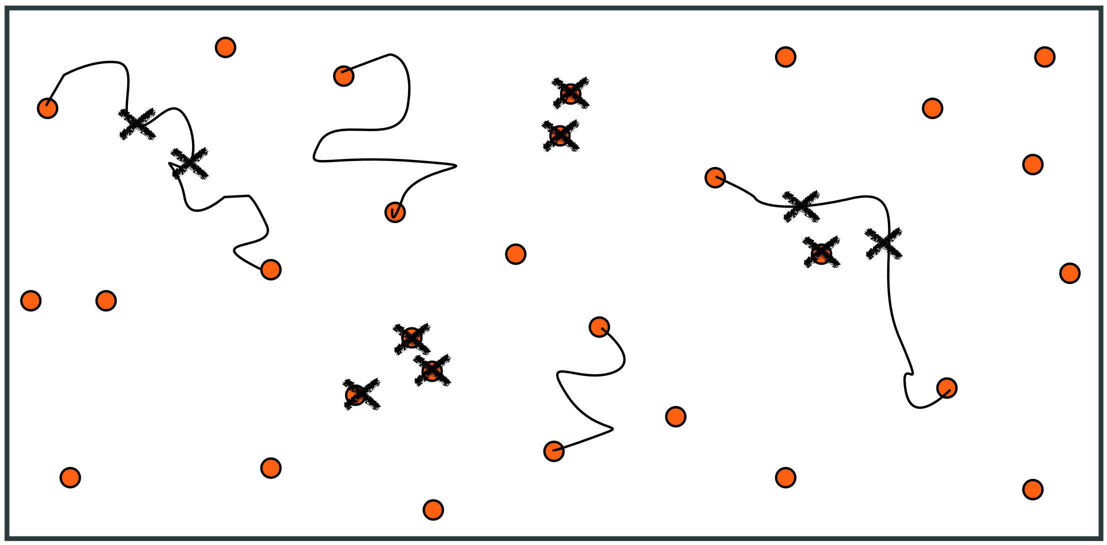
Source: L. Kavraki, RICE - Tutorial

Operations of a PRM



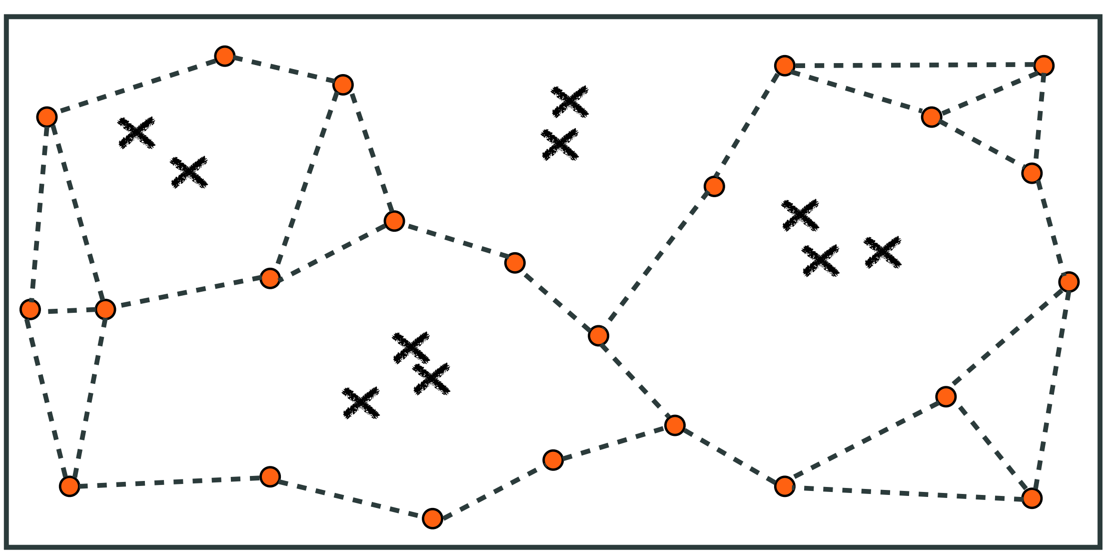
Source: L. Kavraki, RICE - Tutorial

Operations of a PRM



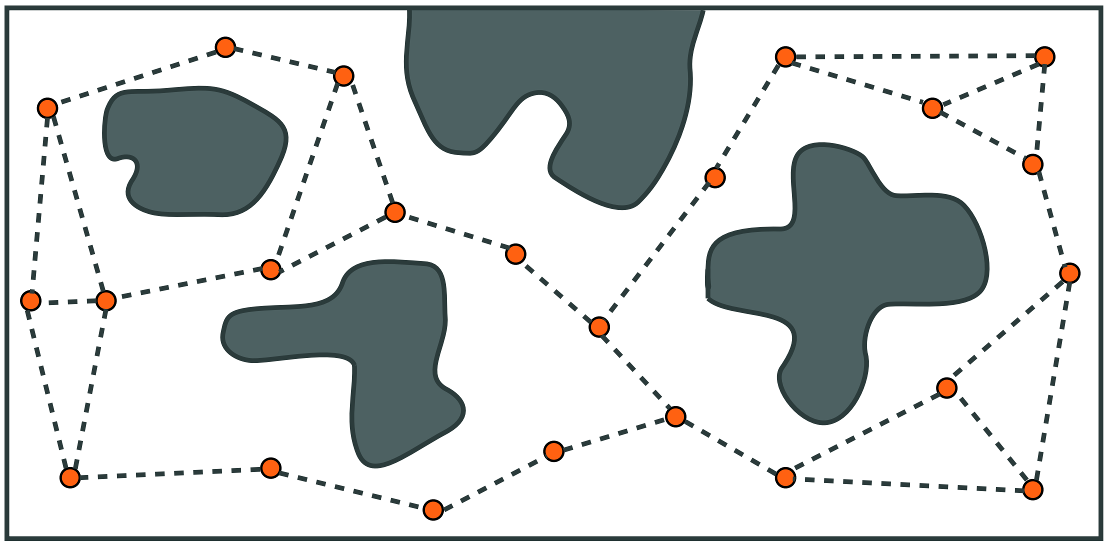
Source: L. Kavraki, RICE - Tutorial

Operations of a PRM



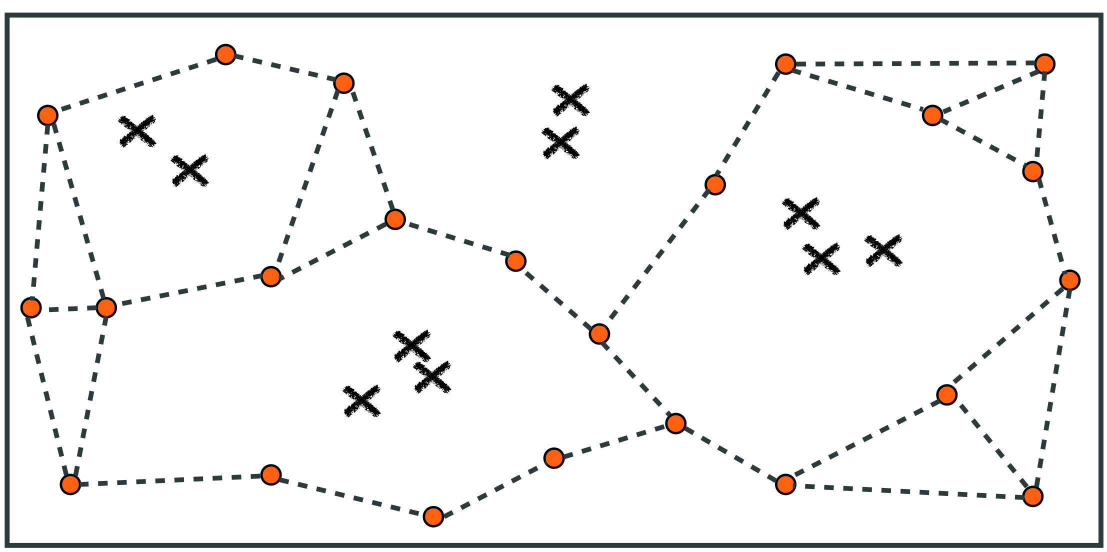
Source: L. Kavraki, RICE - Tutorial

Operations of a PRM



Source: L. Kavraki, RICE - Tutorial

Operations of a PRM



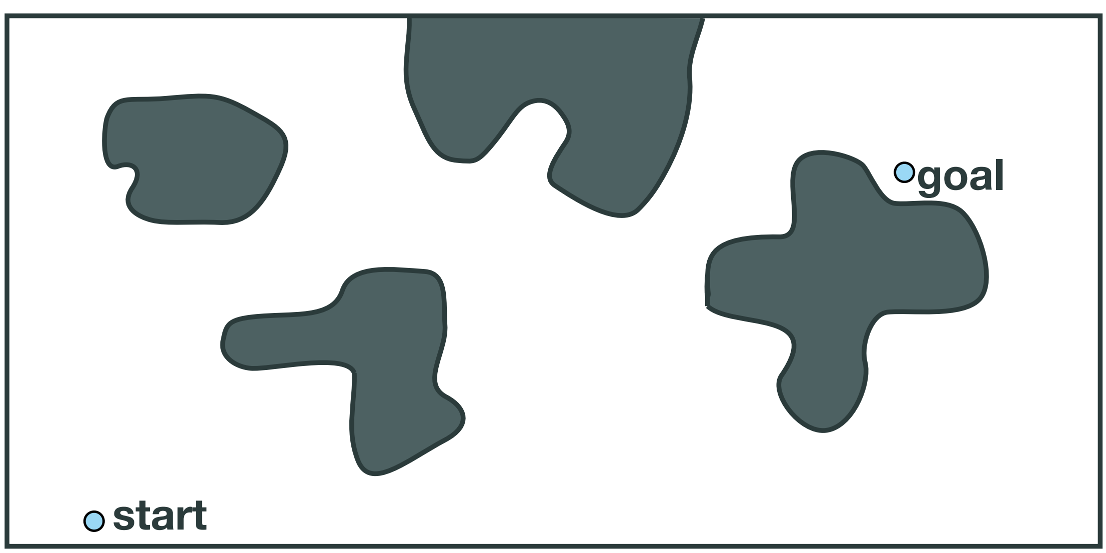
Source: L. Kavraki, RICE - Tutorial

(c) Henrik I Christensen

Rapid Random Trees

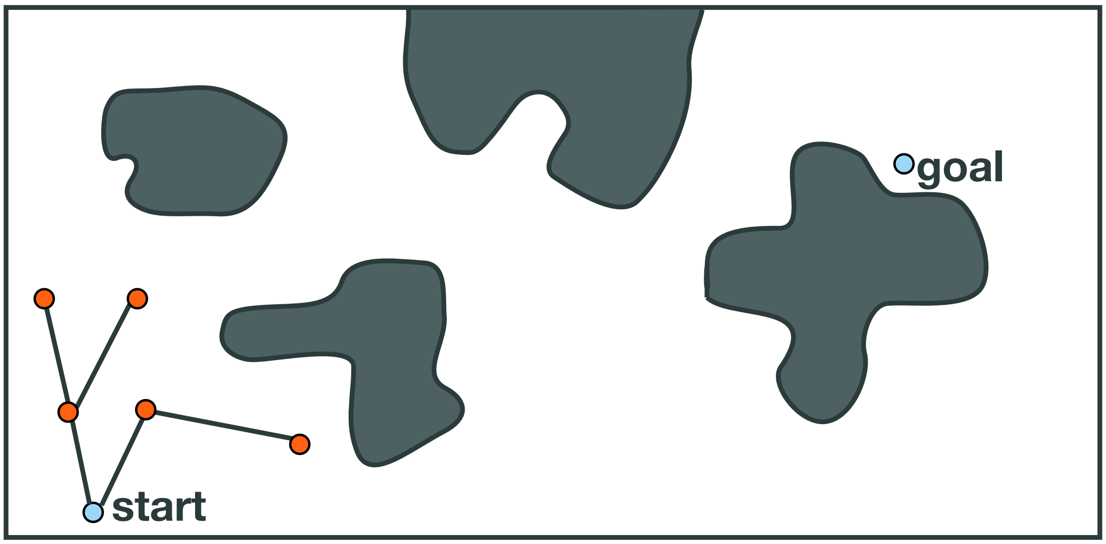
- Could tree search be randomized to achieve some of the same functionality?
- There has been two recent approaches to randomized C-space search
  - Probabilistic Roadmaps (PRM)
  - Rapid Exploring Random Trees (RRT)

Operations of a tree based planner



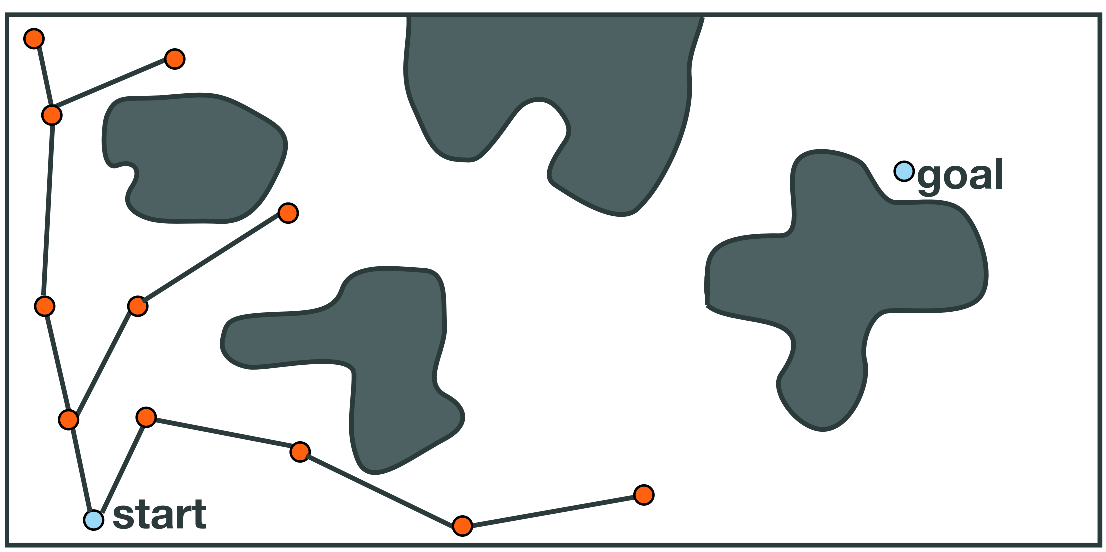
Source: L. Kavraki, RICE - Tutorial

Sampling based tree planner



Source: L. Kavraki, RICE - Tutorial

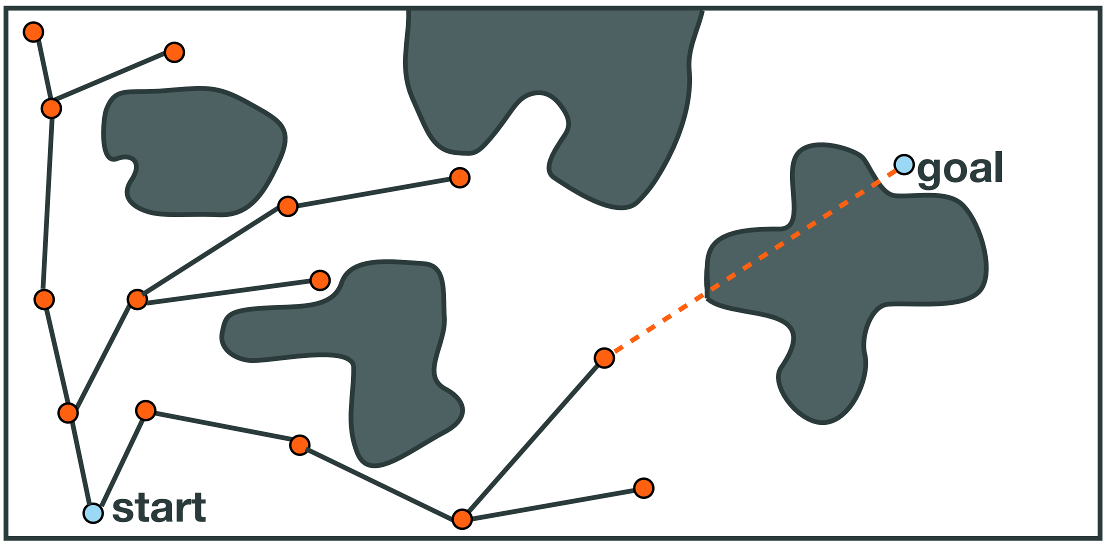
Sampling based tree planner



Source: L. Kavraki, RICE - Tutorial

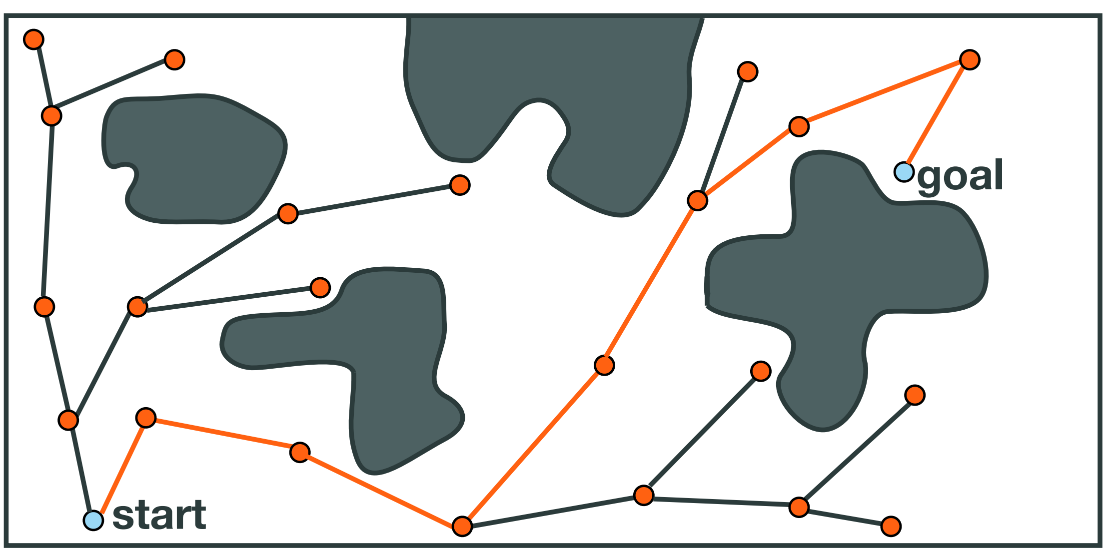
(c) Henrik I Christensen

Sampling based tree planner



Source: L. Kavraki, RICE - Tutorial

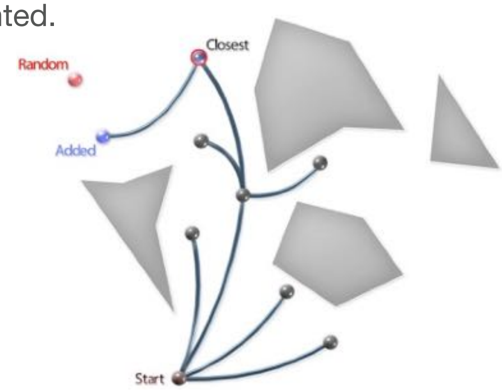
Sampling based tree planner



Source: L. Kavraki, RICE - Tutorial

Randomly Exploring Random Trees (RRT)

- Uses proximity query to guide construction (Voronoi Bias).
- Uses propagation instead of connection.
- Powerful heuristic for single-query planning.
- Bi-directional search can be implemented.



[Lavalle, Kuffer 1999, 2000]

Planning

- There are a rich variety of planning methods
- Consideration of the characteristics
  - Complexity of the configuration space?
  - Can domain constraints be imposed?
  - Can we design deterministic search strategies?
  - What are memory requirements?
  - Do we need real-time response?
- Repositories for planner benchmarking are emerging
- Great literature
  - Choset et al, Principles of Robot Motion, MIT Press
  - Lavalle, Planning Algorithms, Cambridge University Press