Last lecture …

Plan the motion of a car in a dynamic environment by combining the following ideas, if the future environment changes are known or predictable:

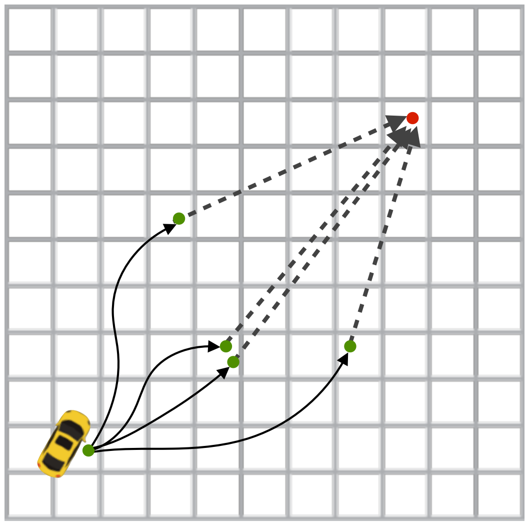
- Represent the motion constraints

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= (v/L)\tan\phi \end{aligned}$$

  More generally,

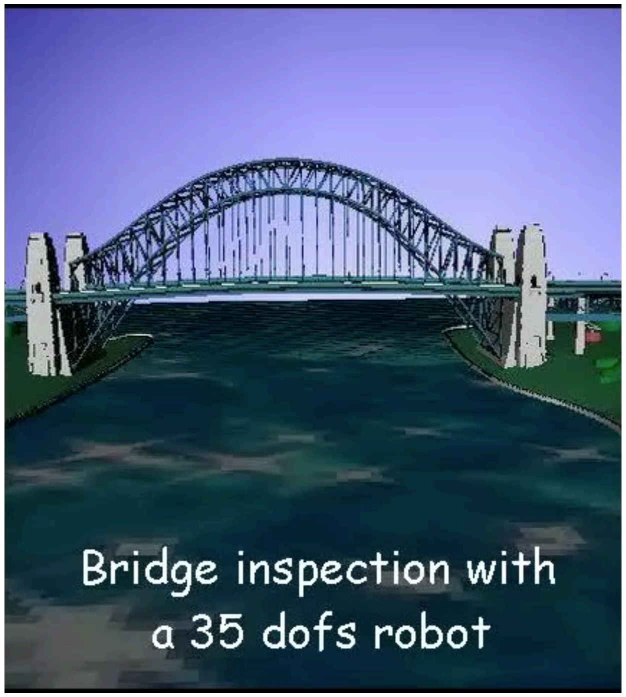
$$\dot{q} = f(q, u),$$

$$q_{t+1} = q_t + \dot{q}_t \Delta t$$



- Configuration space-time $q = (x, y, \theta, t)$
- Hybrid A*

“Curse of dimensionality”. Hybrid A* cannot handle high-dimensional configuration spaces.



The mobile manipulator has 35 DoFs, i.e., a 35-dimensional configuration space. Placing a grid over this space results in 10^{35} grid cells, if each dimension has 10 discrete value.

The running time increases exponentially with the dimension of the configuration space.

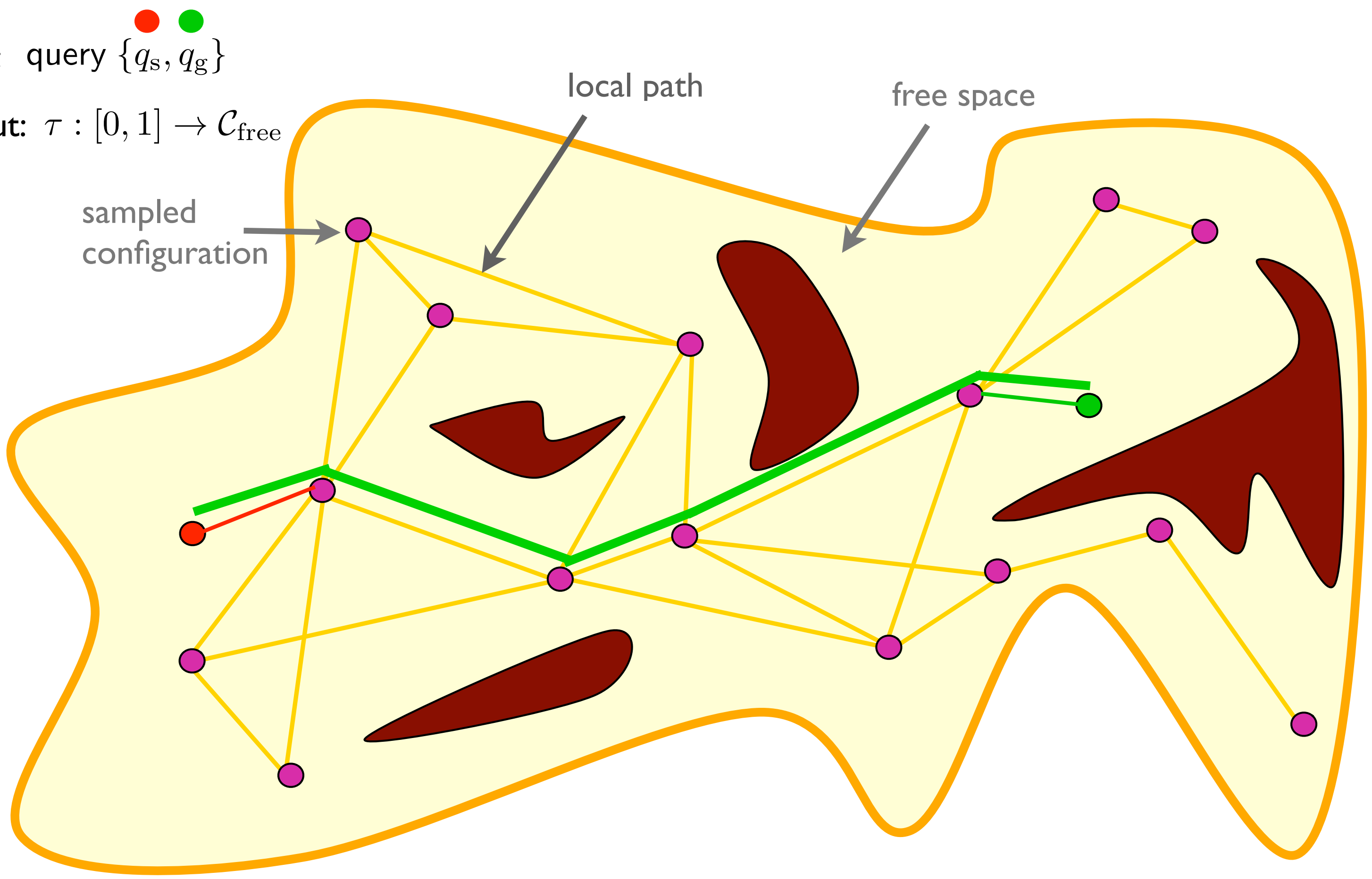
Theoretically, many variants of the path planning problem are PSPACE-hard.

Further, there is geometric complexity, in addition to C-space dimensionality. The rigid body in the example has only 6 DoFs, the required motion to separate the two bodies is intricate and difficult to produce or reproduce.

Probabilistic roadmap (PRM) planning. PRM is a remarkably simple idea that conquers the "curse of dimensionality" through **random sampling**.

Input: query $\{q_{\rm s}, q_{\rm g}\}$

Output: $\tau : [0,1] \to \mathcal{C}_{\rm free}$



PRM planning consists of two phases, offline and online:

- In the offline pre-computation phase, construct a roadmap graph G that captures the connectivity of the underlying configuration space.
- In the online query phase, search G for a path that connects the start and the goal configurations.

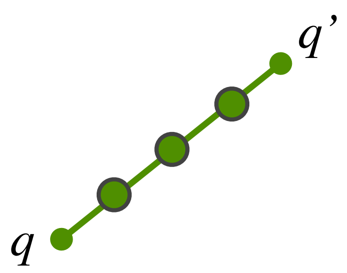
A probabilistic roadmap G is a graph.

- A node of G represents a randomly sampled collision-free configuration.
- An edge between two nodes of G represents a collision-free "simple" local path (e.g., straight-line path) between the two corresponding configurations.

PRM precomputation.

```
Input:
    N: the number of roadmap nodes
    geometry of a robot and obstacles
Output:
  roadmap G = (V, E)

BasicPRM
1: V ← ∅ and E ← ∅.
2:  while |V| < N
3:   q ← a configuration sampled uniformly at random from C.
4:    if CLEAR(q) = TRUE then
5:      Add q to V.
6:      N_q ← a set of nodes in V such that each is close to q
             according to distance d(q,q')
7:      for each q'∈ N_q
8:        if LINK(q',q) = TRUE then
9:          Add an edge between q and q' to E.
13:return G = (V,E)
```

The PRM algorithm relies on two primitives: **CLEAR** and **LINK**. **CLEAR** checks whether a configuration q is collision-free. **LINK** checks whether two configurations q and q' can be connected via a simple path, in particular, a straight-line path. By discretizing the straight-line path, **LINK** can be implemented as a sequence of calls to **CLEAR**. So, **LINK** is computationally much more expensive than **CLEAR**.



PRM query.

- Add the start configuration s as a new node in the roadmap G and connect s to other nodes that are close.
- Do the same for the goal configuration g.
- Search in G for a path between s and g. Return the path if one is found. Otherwise, report NONE.

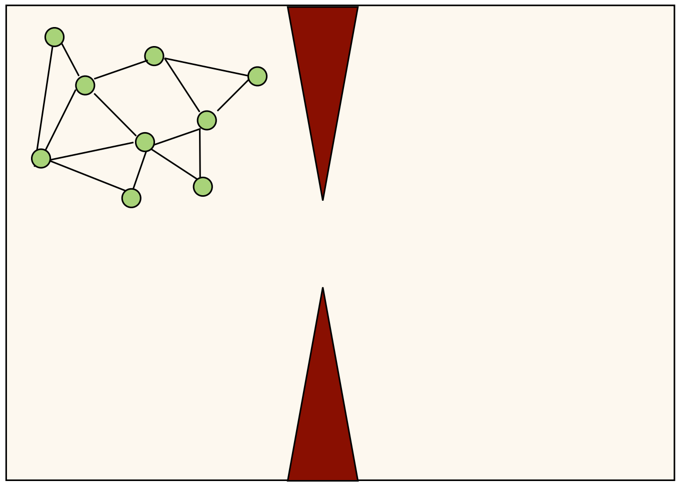
PRM is a major breakthrough in motion planning. It is deceptively simple. Does it work? More importantly, why?

First, we must define what it means to "work".

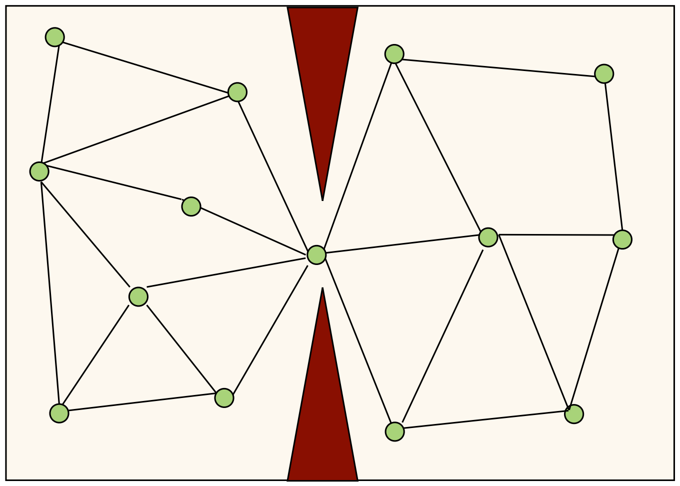
Completeness. The algorithm finds the path if it exists, and reports no otherwise.

Consider the two roadmaps below. The one on the left fails to **cover** the space well.

bad coverage

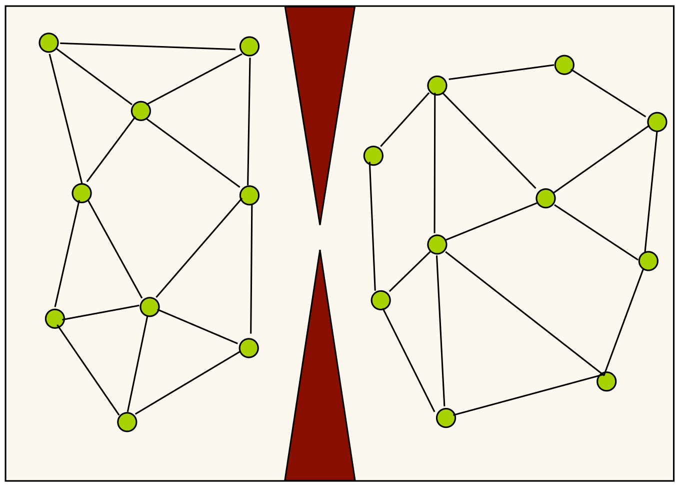


good coverage

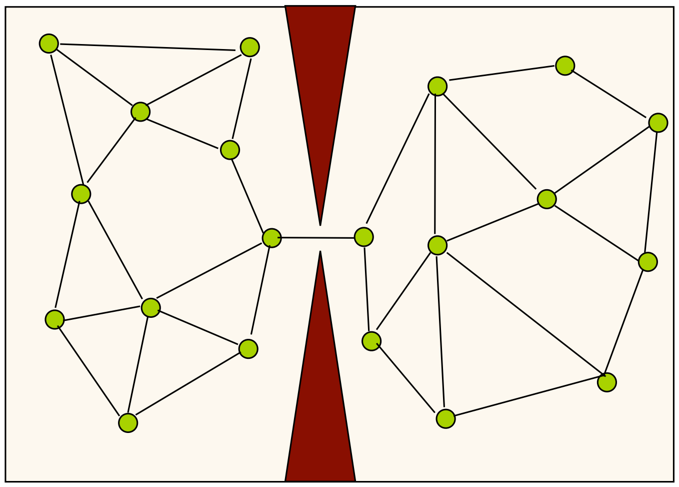


Both roadmaps below cover the space. The one on the left fails to **connect** the space properly.

bad connectivity



good connectivity



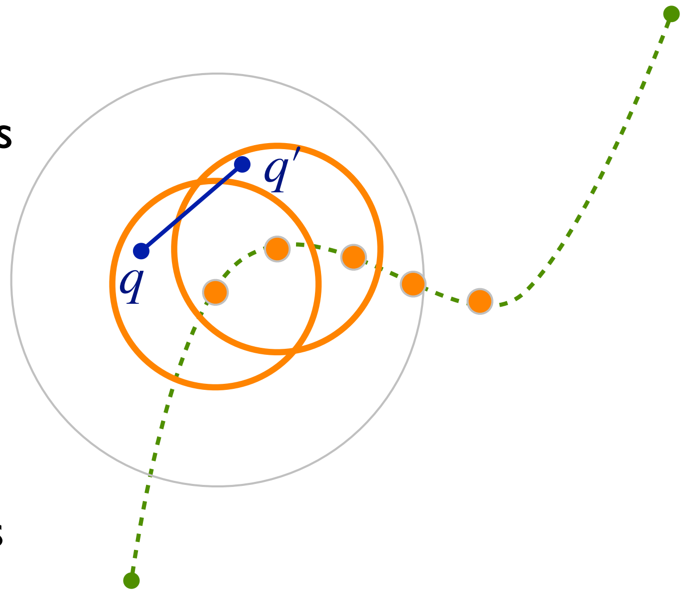
BasicPRM is not complete, for either bad coverage or bad connectivity.

Probabilistic completeness. A PRM algorithm P is probabilistically complete if P satisfies the two conditions below:

- If no solution path exists, P reports NONE.
- If a solution path exists, P finds it with high probability after a finite amount of computation time.

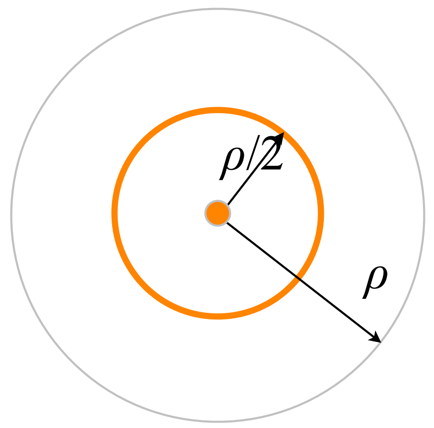
The PRM algorithm is **probabilistically complete**.

Intuitively, the coverage and connectivity of a roadmap improves with the increasing number of samples.



Proof.

- Assume a path of clearance ρ and length L.
- Place balls of radius $\rho/2$ along the path so that the centers of any two adjacent balls are within distance $\rho/2$ along the path.
- Let q be a point in a ball. Let q' be any point in an adjacent ball. The straight-line path between q and q' must be collision-free.
- After sampling a point from every ball, we have a collision-free path.



- Sample n points uniformly at random from the configuration space.

- The failure probability $p(F_i)$ that a given ball i does not get a sample is

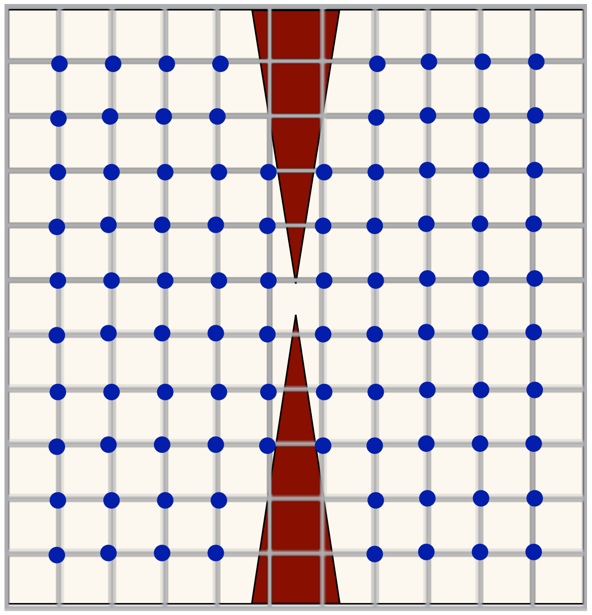
$$p(F_i) = \left(1 - \mu(B_{\rho/2})\right)^n$$

where $\mu(B_{\rho/2})$ is the volume of a ball with radius $\rho/2$.

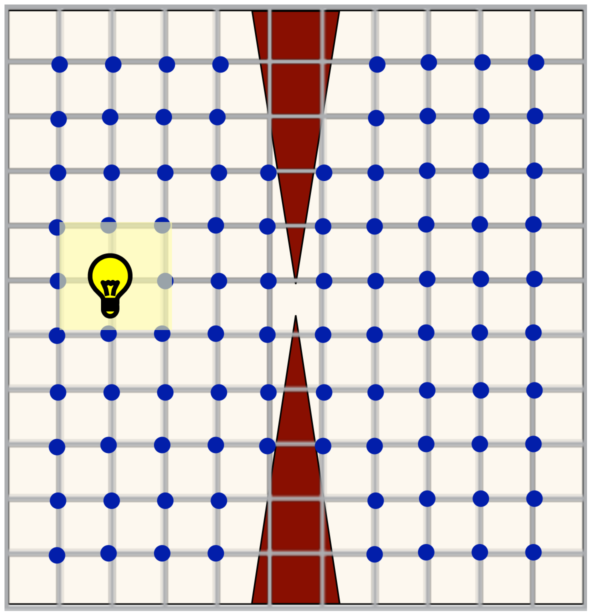
- There are roughly $2L/\rho$ balls. Apply the union bound and obtain the overall failure probability that some ball does not get a sample:

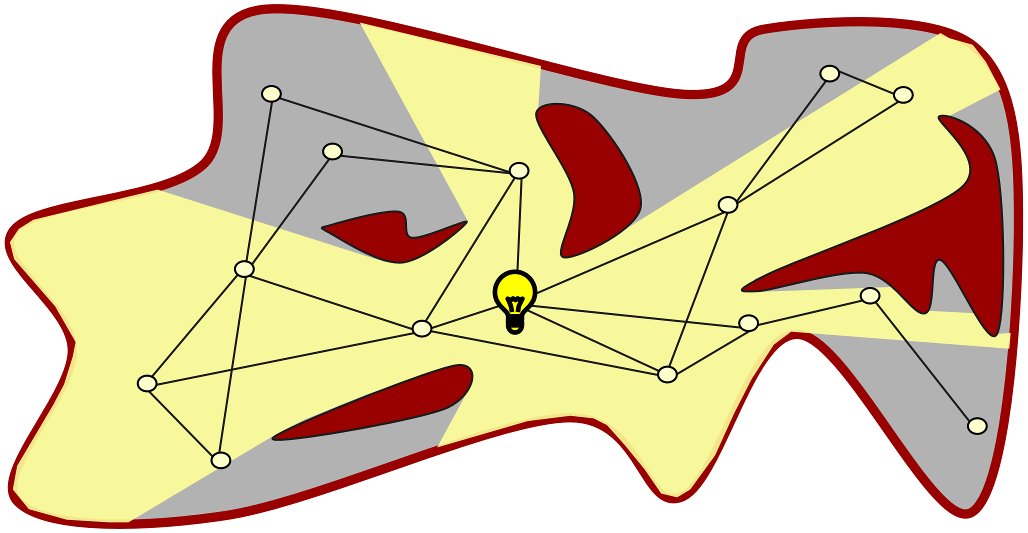
$$\begin{aligned} p(F) &= p(\bigcup_i F_i) \\ &\le \sum_i p(F_i) && \text{Apply the union bound.} \\ &= \frac{2L}{\rho}\left(1 - \mu(B_{\rho/2})\right)^n \\ &\le \frac{2L}{\rho} \mathrm{e}^{-n\mu(B_{\rho/2})} && 1 - x \le \mathrm{e}^{-x} \end{aligned}$$

Comparison. Compare the PRM roadmap with the cell decomposition graph. The cell decomposition graph "samples" configurations on a regular grid. How do they differ?



In the cell decomposition graph, each node is connected to nodes of immediately neighboring cells. In contrast, in a PRM roadmap, a node may be connected to a node far away. The "visibility" region of a node in the roadmap is much bigger than that of a node in the cell decomposition graph. As a result, the roadmap covers the space with much fewer samples.


A dense graph


A very sparse graph

Importance (re-)sampling. Suppose that we have a roadmap consisting of configurations sampled uniformly at random. We can improve the connectivity of the roadmap by sampling in "difficult" regions with poor connectivity.

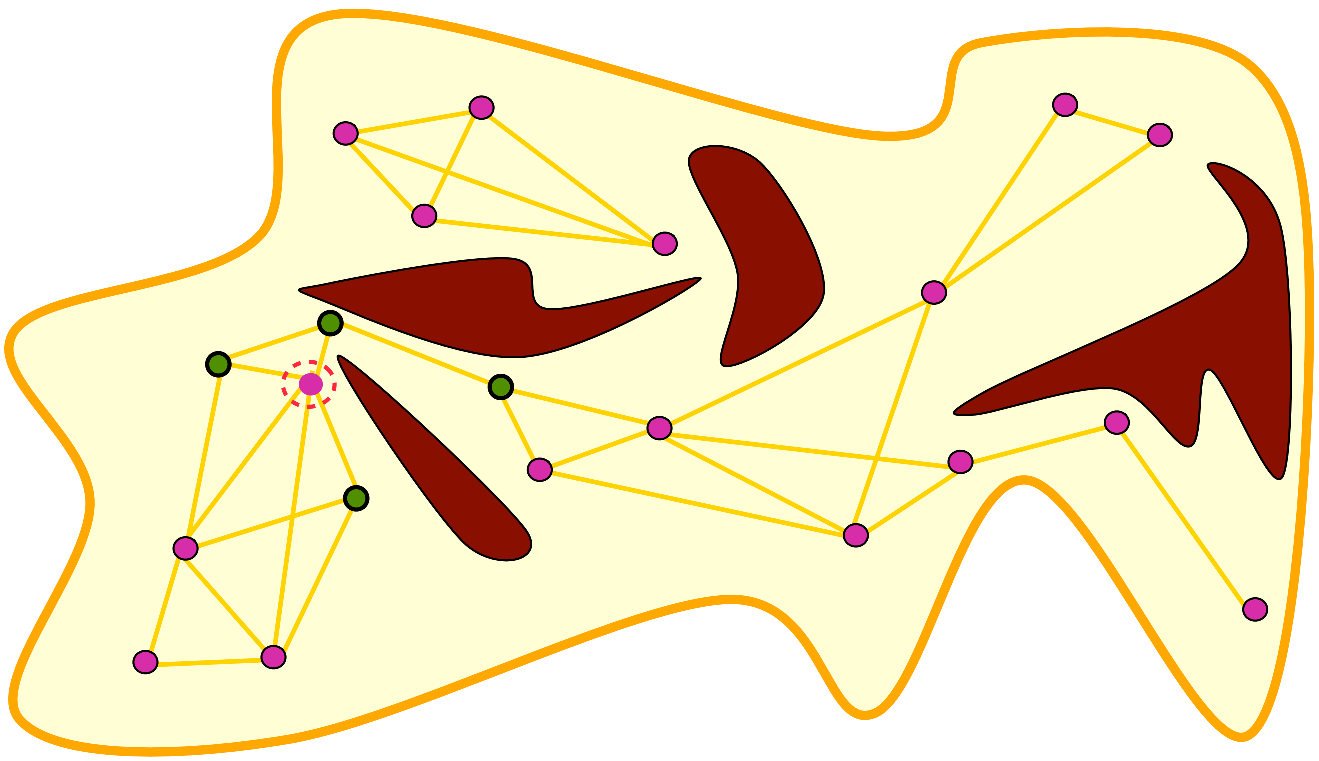
- For each node q in a roadmap, define the importance weight

$$\pi(q) = \frac{1/(\deg(q)+1)}{\sum_{p\in V} 1/(\deg(p)+1)}$$

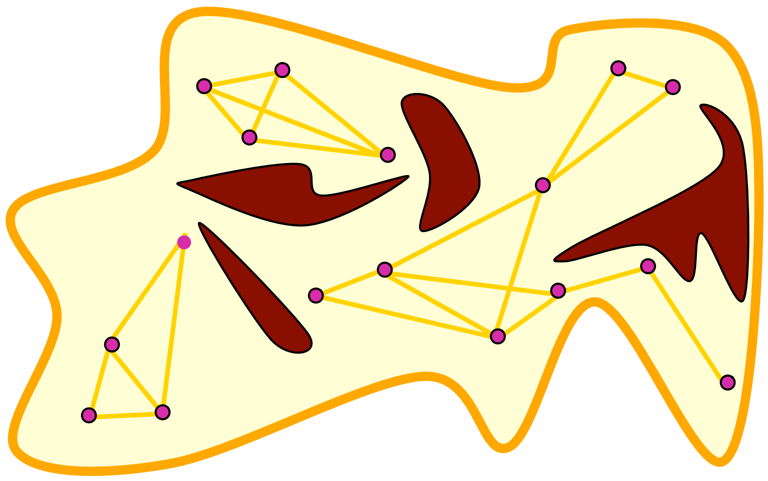
  where $\deg(q)$ is the degree of the node q in the roadmap.

- Choose a node q from V with probability $\pi(q)$.
- Sample new nodes in the neighborhood of q.
- Connect the new nodes to existing nodes in the roadmap.

If a node has low degree, it is not well connected with its neighbors. The connections are blocked by obstacles. It is a reasonable heuristic to add more samples in the "difficult" regions.



Question. According to the importance sampling criteria, which node is most likely to be chosen? Is it a good choice?

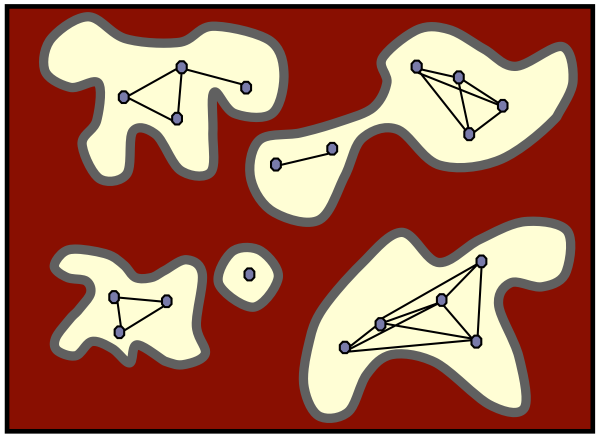


The node at the lower-right corner has degree 1 and is most likely to be chosen. If we indeed choose it, sampling in its neighborhood is unlikely to help improve the connectivity of the roadmap.

The node chosen has degree 2. The probability of choosing the node is proportional to 1/3. So it is quite probable.

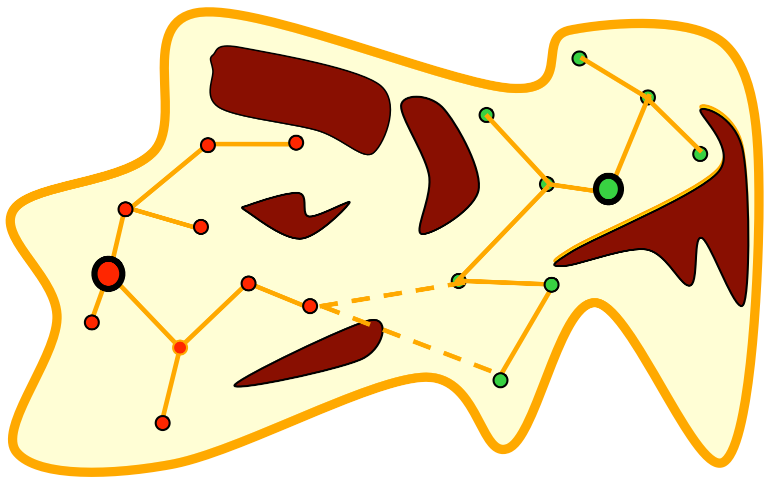
The importance function is merely a heuristic, though often reasonable.

Tree search with random sampling. If we want to answer a single path planning query from the current configuration to a goal. It seems "overkill" to build an entire roadmap. In the configuration space, there are at most two connected components relevant to the query, one containing the start configuration and one containing the goal configuration. The motivation here is similar to that of forward search.



Here we must combine tree search with random sampling.

- Incrementally "grow" two trees T_s and T_g, rooted at s and g, respectively.
- **Expand** a tree by choosing a node from a tree and sample in its neighborhood.
- Connect the two trees.



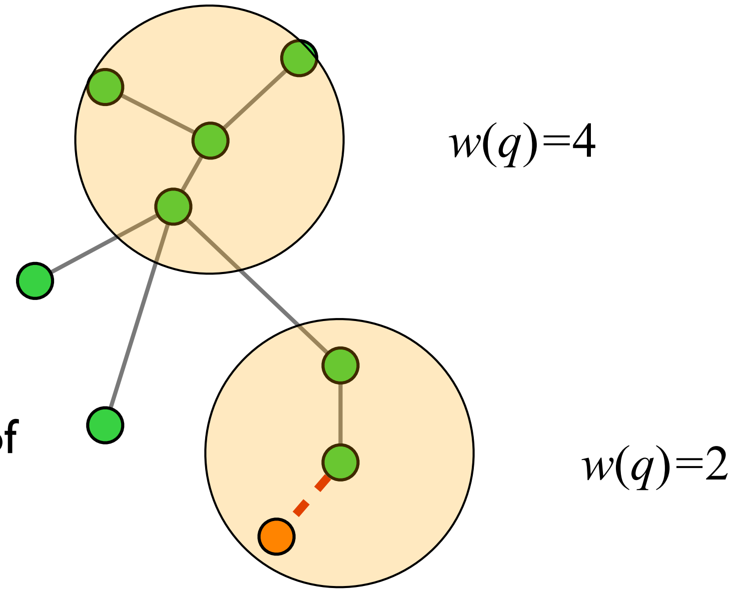
There are different ways of expanding tree nodes.

Expansive-space tree (EST). The main idea is to expand tree nodes in low-density regions.

- For a node q, define the weight $w(q)$ to be the number of nodes in a sphere with the center q and some chosen radius ρ.
- Choose a node q from T with probability

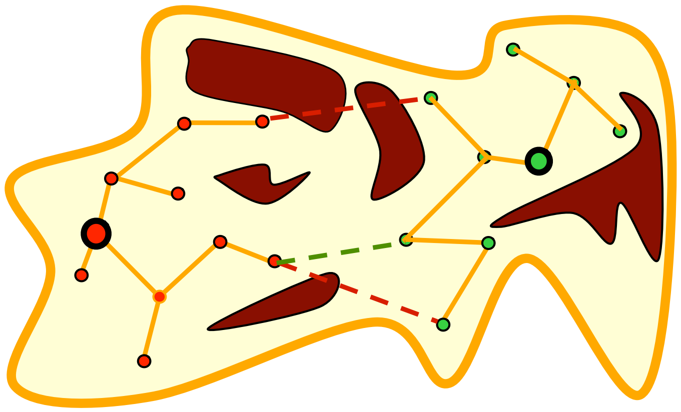
$$\pi(q) = \frac{1/(w(q))}{\sum_{p \in V} 1/w(p)}.$$

- Sample a new configuration q' from the neighborhood of q. Add q' to T if q' is collision-free and there is a collision-free straight-line path between q and q'.



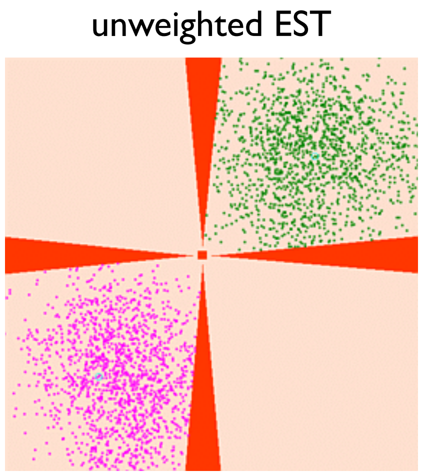
```
Input:
  q0: the configuration for the root of the tree
  N : the number of tree nodes
  geometry of a robot and obstacles
Output:
  tree T = (V, E)

EST-Expand
1: V ← {q0} and E ← ∅.
2: while |V| <= N
3:   Choose a node q from V with probability $\pi$(q).
4:   Sample a configuration q′ from the neighborhood of q.
5:      if CLEAR(q′) = TRUE and LINK(q′,q) = TRUE then
6:         Add q′ to V.
7:         Add an edge between q and q′ to E.
8:         $N_{q'}$ ← a set of nodes in V whose distance to q′ is
9:               no greater than $\delta$.
10:        w(q′) ← |$N_{q'}$|+1
11:        for each p ∈ $N_{q'}$
12:          w(p) ← w(p)+1.
13:return T = (V,E)
```

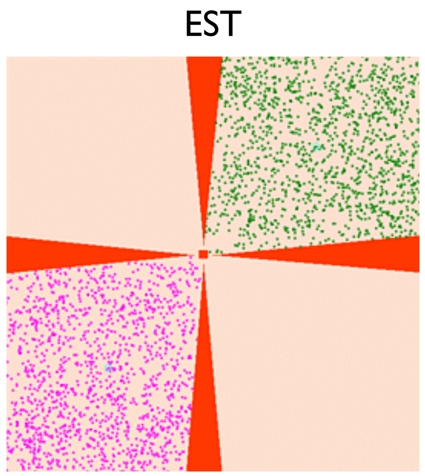
Expand two EST trees, rooted at the start and the goal configurations, and connect the two trees. For each node q in T and q' in T', if the distance $d(q, q')$ is small, call **`LINK`**(q, q') to check whether there is a collision-free straight-line path between q and q'.



Question.

- What is the difference between the two sets of samples below? Which one is obtained with the weighted sampling?





- Why do we use weighted sampling when choosing the nodes from V?

Clustered samples do not cover the space well.

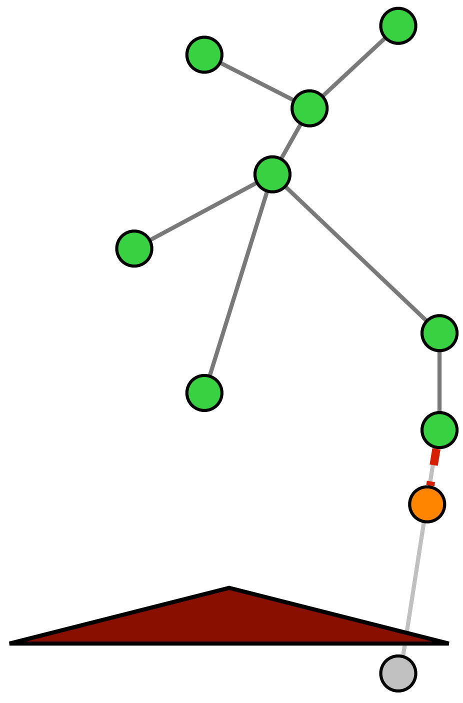
Rapidly-exploring random tree. Approach samples from a target sampling distribution, e.g., the uniform distribution.

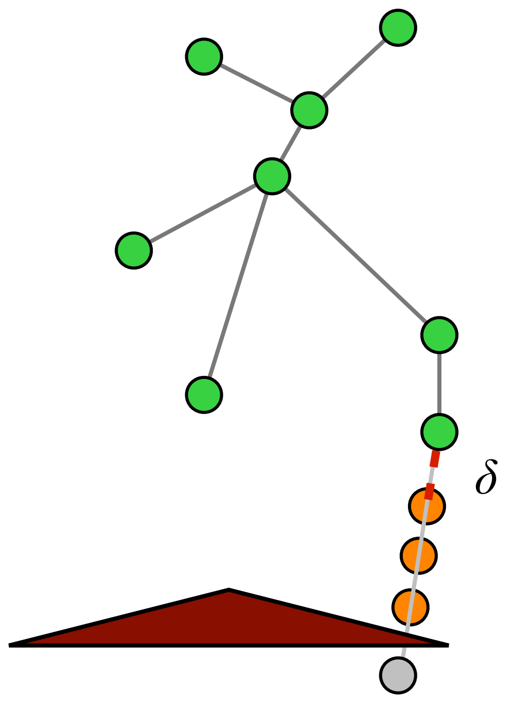
- Sample a collision-free configuration p uniformly at random.
- Find a node q from T, which is closest to p.
- Set q' to be the configuration at a fixed distance δ from q along the straight line from q to p. Add q' to T if q' is collision-free and there is a collision-free straight-line path between q and q'.

We can also approach the target sample in a more aggressive manner by the replacing the last step above with

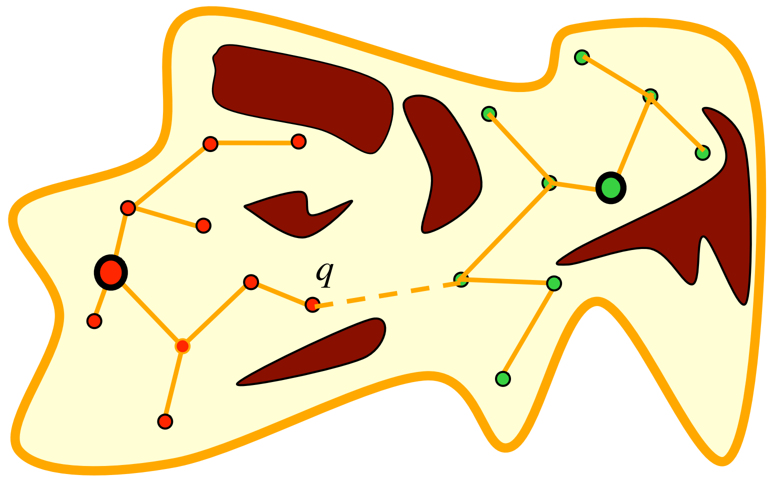
- Set q' to be the configuration a fixed distance δ from q along the straight line from q to p. Add q' to T if q' is collision-free and there is a collision-free straight-line path between q and q'.

Set $q = q'$. Repeat until blocked by obstacles.





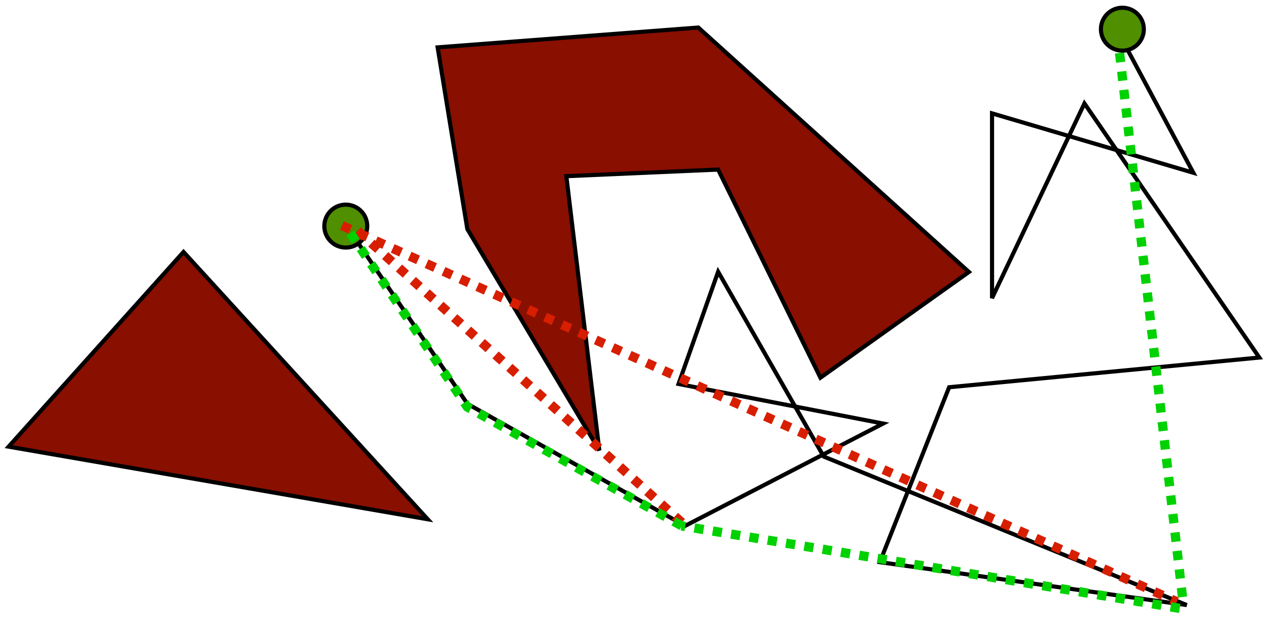
Expand two RRT trees T and T', rooted at the start and the goal configurations. For each new node q of a tree T, (greedily) extend T' towards q instead of a randomly sampled configuration p. Terminate if the two trees are connected.



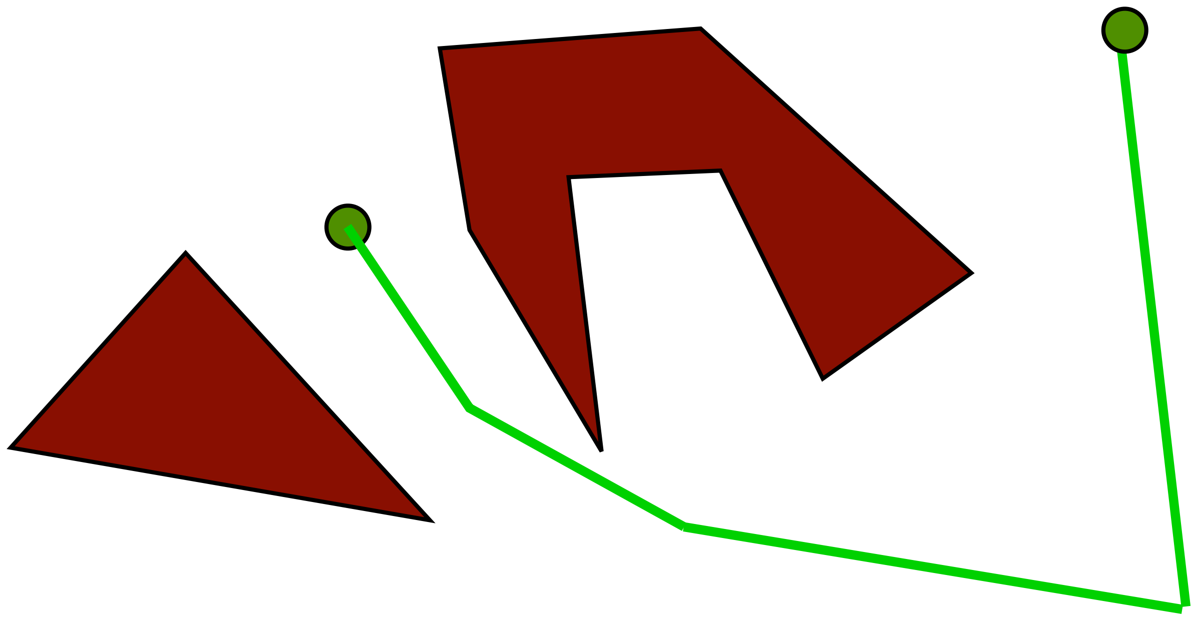
Comparison of EST and RRT.

- Both EST and RRT can be shown to be probabilistically complete.
- They are in fact very similar, but differ mainly in how new configurations are sampled. RRT places strong confidence in the sampling heuristic that there are straight-line paths between nodes of the two trees and is more **greedy** in trying to sample nodes that connect the two trees togethers. This is often good, but gets in trouble when the heuristic fails. As a rough analogy, EST behaves more like breadth-first search, and RRT behaves more like depth-first search.
- EST and RRT find valid collision-free path, but **not** the shortest path.

Path smoothing. Random sampling often results in a jagged path, which is undesirable in practice. To smooth the path, perform "shortcutting" recursively.



Path smoothing. Random sampling often results in a jagged path, which is undesirable in practice. To smooth the path, perform "shortcutting" recursively.



Online and offline planning. There are two common planning and execution architectures, offline and online.

Offline planning consists of two phases:

- Planning. Plan a path for every possible start and goal configuration pairs.
- Plan execution. Execute a pre-plannned path, given the current configuration and the goal configuration.

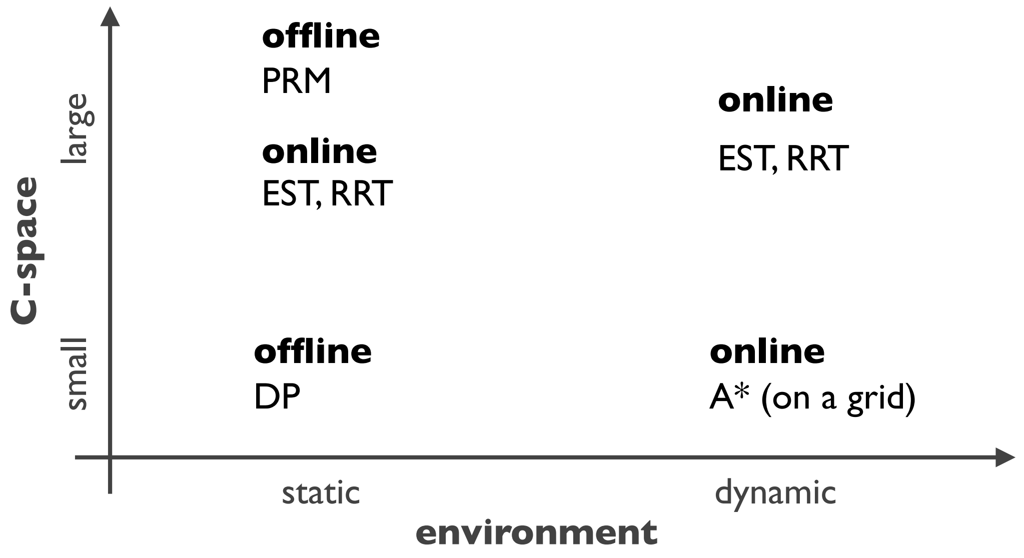
If we have sufficient information about the execution environment and sufficient computational resources, then we can pre-compute all possible solutions in advance. The execution phase is then very simple and efficient. The robot looks up the pre-computed solution based on the current configuration and executes it.

Alternatively, online planning repeats the following steps:

- Plan a path for the current configuration.
- Execute one step along the path.
- Update the current configuration and repeat

Online planning is common in dynamic environments, where pre-computed solutions quickly become invalid because of environment changes.

Summary.

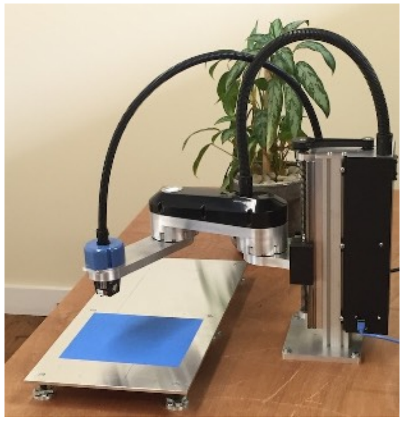


- Dynamic environments requires online planning. Extensive pre-computation is wasteful, as pre-computed solutions or partial solutions quickly become invalid because of environment changes. Forward search algorithms such as A*, EST, RRT, … are suitable candidates.
- In static environments, we can pre-compute all solutions, using, e.g., dynamic programming, if the C-space size is moderate and sufficient computational resources are available.

- In a high-dimensional C-space, e.g., a robot manipulator with many DoFs, PRM planning helps to alleviate the difficulty of "curse of dimensionality". However, forward search algorithms, such as EST or RRT, become the preferred choices with increasing C-space dimensions.

Question. What algorithms shall we choose for the following motion planning tasks?

- A 3-DoF robot manipulator on a factory assembly line.
- A self-driving car driving on public roads.
- A warehouse robot fetching items in a big warehouse.


Flx.arm

Fetch

A factory assembly line is engineered to be a static environment.

The robot has only 3 DoFs, thus a small C-space.

The car has only 3 DoFs, but operates in a highly dynamic environment. It's not realistic to engineer an open environment, such as public roads.

Fetch (the one with the arm) is a robot manipulator with a mobile base. It has more than 11 DoFs.

The warehouse is a static environments, except for the other robots.

Required readings.

- [PRM-Cho01] Sect 7.1.1, 7.1.2, 7.2

Supplementary readings.

- [PRM-Cho01] Sect 7.1.3, 7.1.4, 7.4.1

Key concepts.

- PRM
- EST
- RRT
- Completeness, probabilistic completeness
- Offline and online planning

Tools.

- MoveIt
- OMPL