Sampling-Based Methods for Path Planning

With so many slides and ideas from so many people:

Howie Choset, Nancy Amato, David Hsu, Sonia Chernova, Steve LaValle, James Kuffner, Greg Hager

Difficulty with classic approaches to path planning

- Running time increases exponentially with the dimension of the configuration space.
  - For a d-dimension grid with 10 grid points on each dimension, how many grid cells are there?

$$\boxed{10^d}$$

- Several variants of the path planning problem have been proven to be PSPACE-hard.

Completeness

- Complete algorithm → Slow
  - A **complete** algorithm finds a path if one exists and reports no otherwise in finite time.
  - Example: visibility graph for 2D problems (translation in the plane) and polygonal robot and obstacles
- Heuristic algorithm → Unreliable
  - Example: potential field (we'll see it soon)

- **Probabilistic completeness**
  - Intuition: If there is a solution path, the algorithm will find it with high probability.

The Rise of Monte Carlo Techniques

- KEY IDEA:
  Rather than exhaustively explore ALL possibilities, randomly explore a smaller subset of possibilities while keeping track of progress

- Facilities “probing” deeper in a search tree much earlier than any exhaustive algorithm can

- What’s the catch?
  Typically we must sacrifice both *completeness* and *optimality*
  Classic tradeoff between solution quality and runtime performace

Sampling Based Planning:

Search for collision-free path
only by sampling points.

Probabilistic Roadmaps

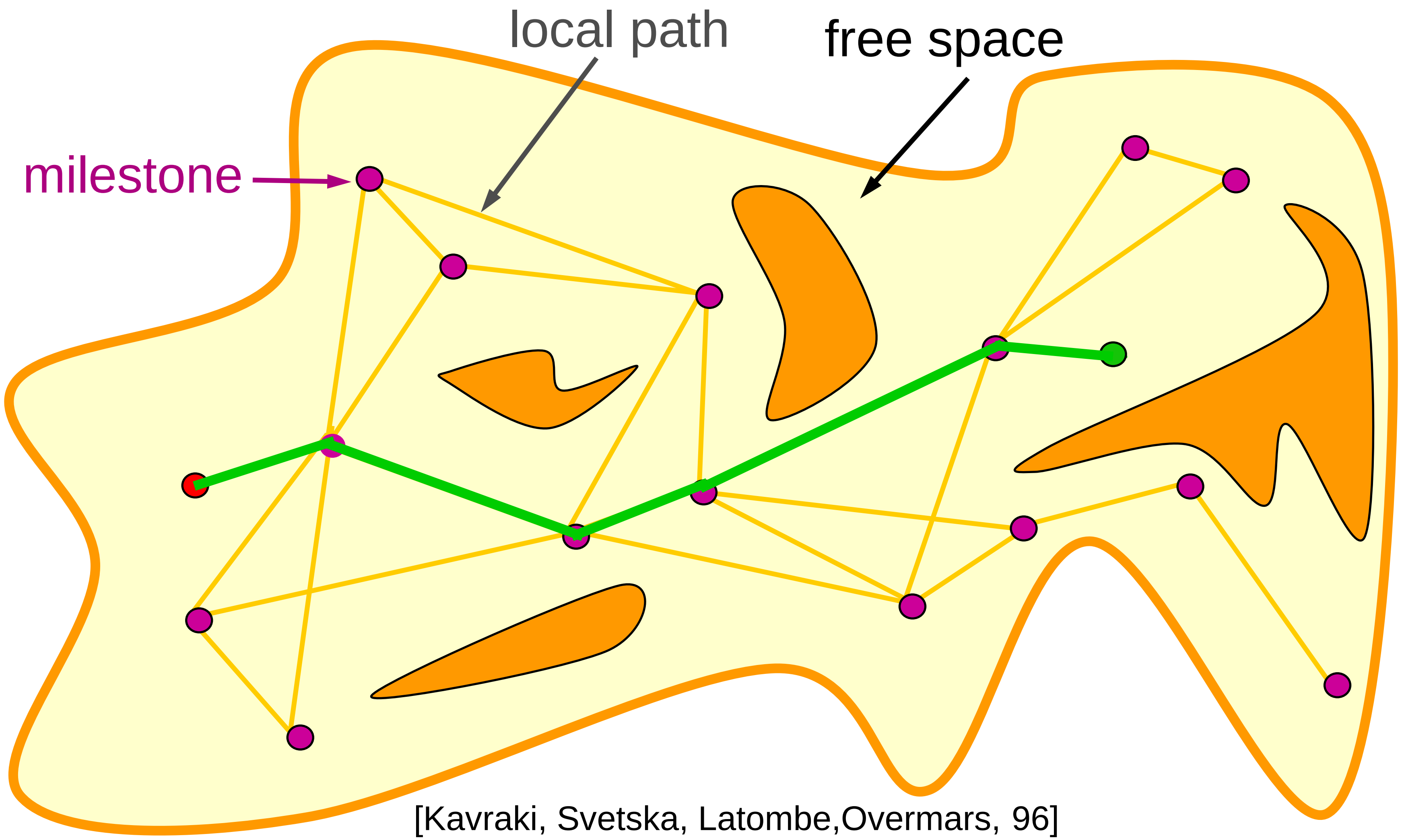
Probabilistic Road Map (PRM)

- Probabilistic Roadmap methods proceed in two phases:

    1. Preprocessing Phase – to construct the roadmap G
    2. Query Phase – to search given q_{init} and q_{goal}

    The roadmap is an undirected graph $G = (N, E)$. The nodes in N are a set of configurations of the robot chosen over C-free. The edges in E correspond to feasible straight-line paths.

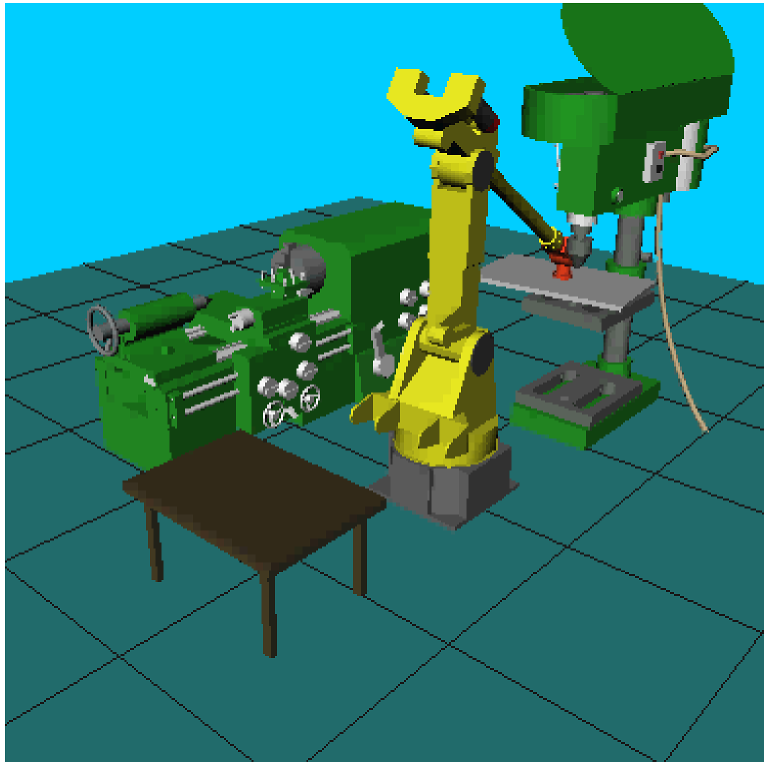
Probabilistic Roadmap (PRM): multiple queries



[Kavraki, Svetska, Latombe,Overmars, 96]

Assumptions

- Static obstacles
- Many queries to be processed in the same environment
- Examples
  - Navigation in static virtual environments
  - Robot manipulator arm in a workcell
- Advantages:
  - Amortize the cost of planning over many problems
  - Probabilistically complete



Overview

- Precomputation: roadmap construction
  - Uniform sampling
  - Resampling (expansion)
- Query processing

Uniform sampling

Input: geometry of the moving object & obstacles
Output: roadmap G = (V, E)

```
V ← ∅ and E ← ∅.
repeat
  q ← a configuration sampled uniformly at random from C.
   if CLEAR(q) then
     Add q to V.
     N_q ← a set of nodes in V that are close to q.
     for each q′ ∈ N_q, in order of increasing d(q,q′)
       if LINK(q′,q) then
         Add an edge between q and q′ to E.
```

Some terminology

- The graph G is called a **probabilistic roadmap**.
- The nodes in G are called **milestones**.

How do we determine a *random free configuration?*

- We want the nodes of V to be a ***uniform*** sampling of Q_{free}

  - Draw each of its coordinates from the interval of values of the corresponding degrees of freedom. (Use the uniform probability distribution over the interval)

  - Check for collision both with robot itself and with obstacles

  - If collision free, add to V, otherwise discard

  - What about rotations? Strategies for sampling orientation are beyond the scope of this class. Since Duckiebots live in the plane, we could merely sample uniformly in the interval $[0, 2\pi]$.

What's the local path planner: Link(q',q) ?

- There are plenty of possibilities
  - Nondeterministic (include a randomized "wandering" component)
    - We'll have to store local paths in roadmap
  - Powerful
    - Slower but maybe we'll need fewer nodes if we do some hard work during roadmap construction?
  - Fast and simple
    - Less powerful, Roadmap will need more nodes

Go with the fast local planner

- Need to make sure start and goal configurations can connect to graph, which requires a somewhat dense roadmap
- Can reuse local planner at query time to connect start and goal configurations
- Don’t need to memorize local paths

Distance Functions: $d(q,q')$

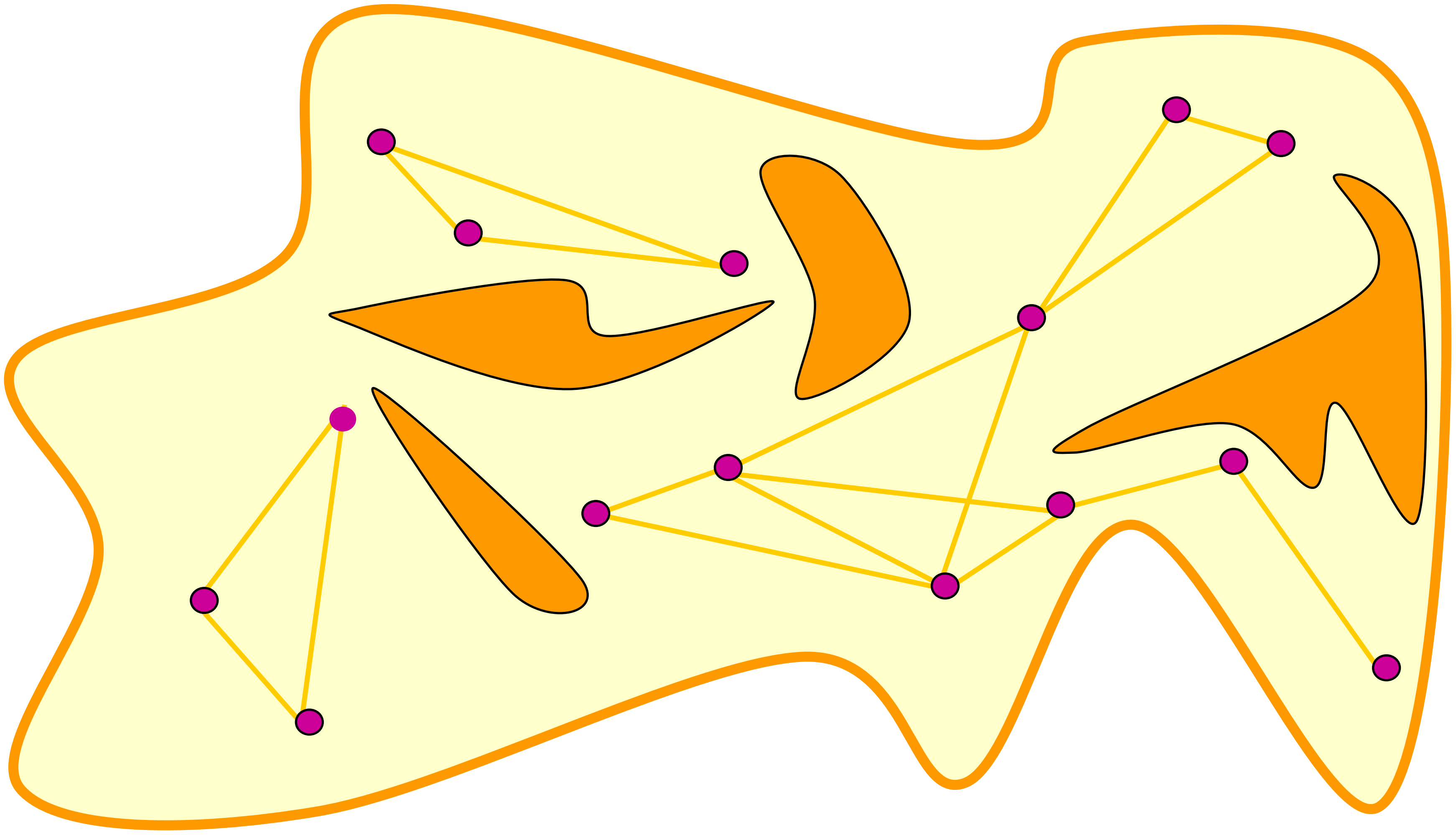
- Really, *d* should reflect the likelihood that the planner will fail to find a path
  - close points, likely to succeed
  - far away, less likely

- This is often related to the area swept out by the robot along the local path:
  - very hard to compute exactly
  - usually heuristic distance is used

- Typical approaches
  - Euclidean distance on some embedding of c-space
  - Create a weighted combination of translation and rotational “distances”
  - Weighted sum of distances for a set of “control points” on the robot

Difficulty

- Many small connected components



Resampling (expansion)

- Failure rate $$r(q) = \frac{f(q)}{n(q)+1}$$

- Weight $$w(q) = \frac{r(q)}{\sum_p r(p)}$$

- Resampling probability $$\Pr(q) = w(q)$$

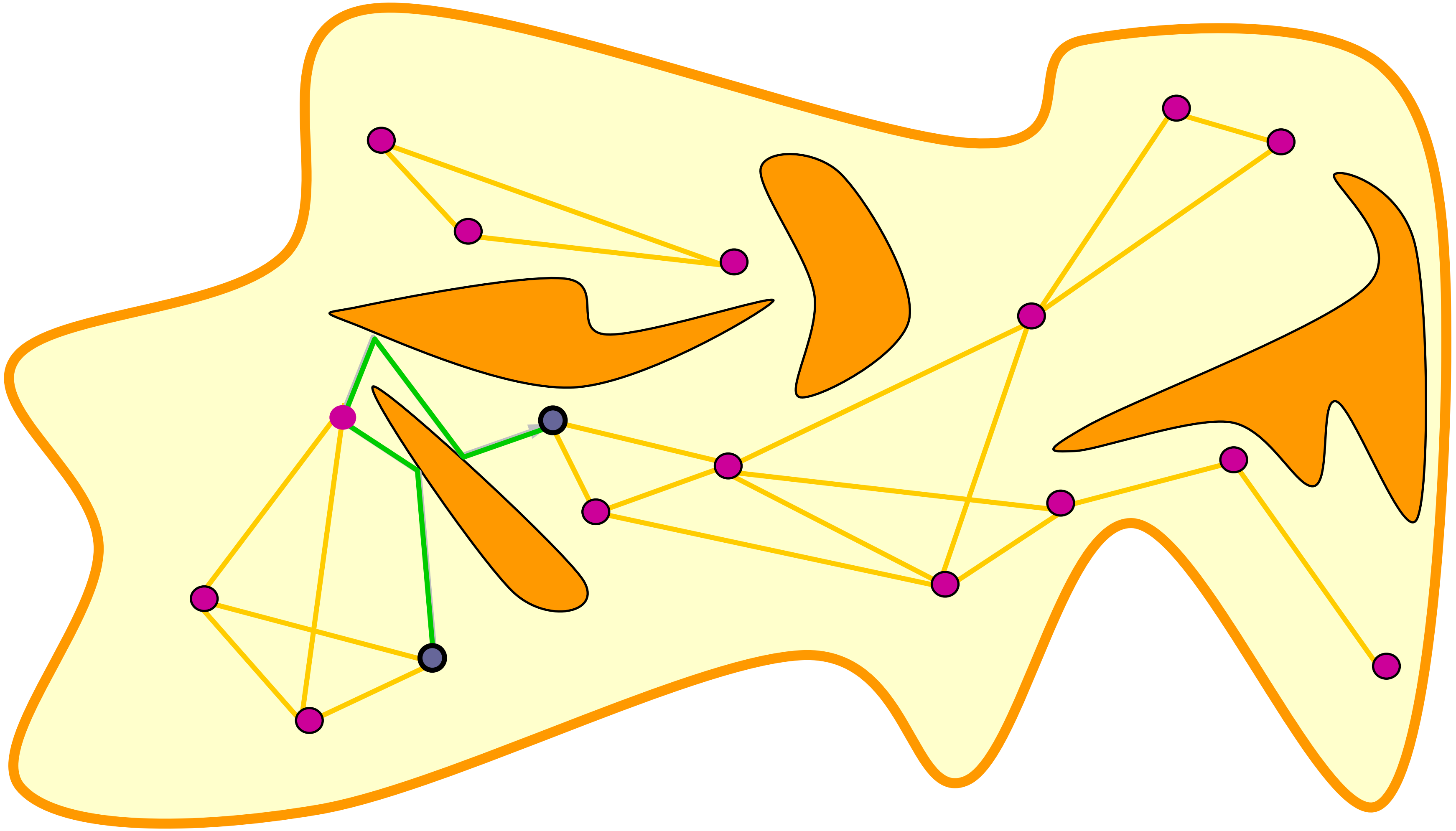
- $f(q)$ =# of failed attempts to connect q to the roadmap
- $n(q)$ = total # of attempts to connect q to the roadmap

Now that we have weights…

- To expand a node c, we compute a short random-bounce walk starting from c.

  This means
  - Repeatedly pick at random a direction of motion in C-space and move in this direction until an obstacle is hit.
  - When a collision occurs, choose a new random direction.
  - The final configuration n and the edge (c,n) are inserted into the roadmap and the path is memorized.
  - Try to connect n to the other connected components like in the construction step.
  - Weights are only computed once at the beginning and not modified as nodes are added to the roadmap.

Resampling (expansion)



Query processing

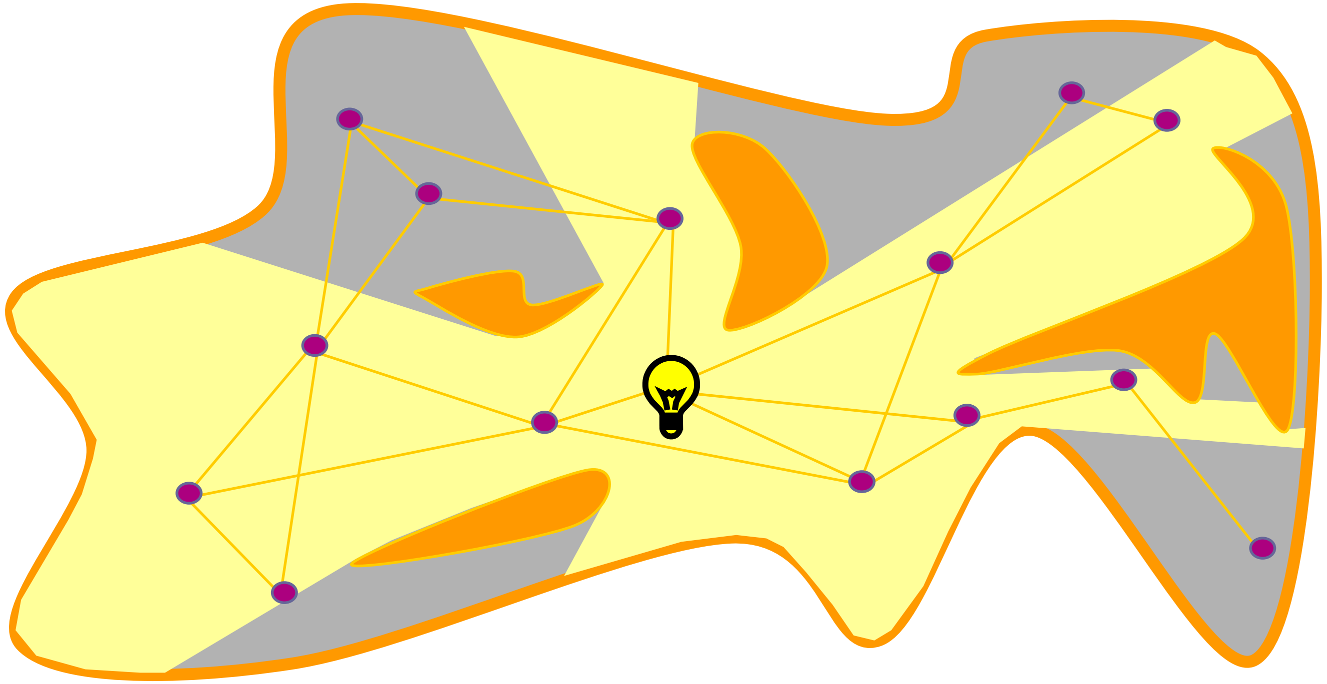
- Connect q_{init} and q_{goal} to the roadmap
- Start at q_{init} and q_{goal}, perform a random walk, and try to connect with one of the milestones nearby
- Try multiple times

Error

- If a path is returned, the answer is always correct.
- If no path is found, the answer may or may not be correct. We hope it is correct with high probability.

Why does it work? Intuition

- A small number of milestones **almost** "cover" the **entire** configuration space.



- Rigorous definitions and exist (of course!)