UC San Diego

Introduction to Deep Reinforcement Learning

Model-based Methods

Zhiao Huang

Model-free Methods

- The model $p(s'|s,a)$ is unknown
  - we solve Q, V and the policy from the sampled trajectories/transitions

Model-based Method

- Learn the environment model directly

$$p(s'|s,a)$$

  - Learn $R(s,a,s')$ if it's unknown

Model-based Method

- Learn the environment model directly by supervised learning

$$p(s'|s,a)$$

- **Search** the solution with the model directly

Model-based Methods

- Model and search have broad meanings

Model

Physics
Geometry
Probability model
Inverse Dynamics
Game Engine
….

Search

MCTS
CEM
RL
iLQR
RRT/PRM
……

Model-Predictive Control

- Forward model with parameters θ

$$f_\theta(s, a): S \times A \to S$$

  - Predicts what will happen if we execute the action a at the state s

Forward Model

- We may have a forward model in mind

Learning the Forward Model

- Sample transitions (s, a, s') from the replay buffer and train the model with **supervised learning**

$$\min_{\theta} E[\|f_\theta(s, a) - s'\|^2]$$

Rollout

- Predict a short trajectory $s_1, s_2, \ldots, s_T$ if we start at s_0 and execute $a_0, a_1, a_2, \ldots, a_{T-1}$ sequentially

$$s_0 \xrightarrow{f_\theta(s_0, a_0)} s_1 \xrightarrow{f_\theta(s_1, a_1)} s_2 \xrightarrow{f_\theta(s_2, a_2)} \cdots \xrightarrow{f_\theta(s_{T-1}, a_{T-1})} s_T$$

"rollout" the forward model

Model-Predictive Control

- Given the forward model f_θ and the current state s_0, find a sequence of action $a_0, a_1, a_2, \ldots, a_{T-1}$ such that has the maximum reward

$$\max_{a_{0:T-1}} \sum_{i=1}^{T} R(s_i, a_i, s_{i+1}) \text{ s.t. } s_{i+1} = \mathrm{f}_\theta(s_i, a_i)$$

Random Shooting

- Sample N random action sequences:

$$a_0^1, a_1^1, a_2^1, \dots, a_{T-1}^1$$

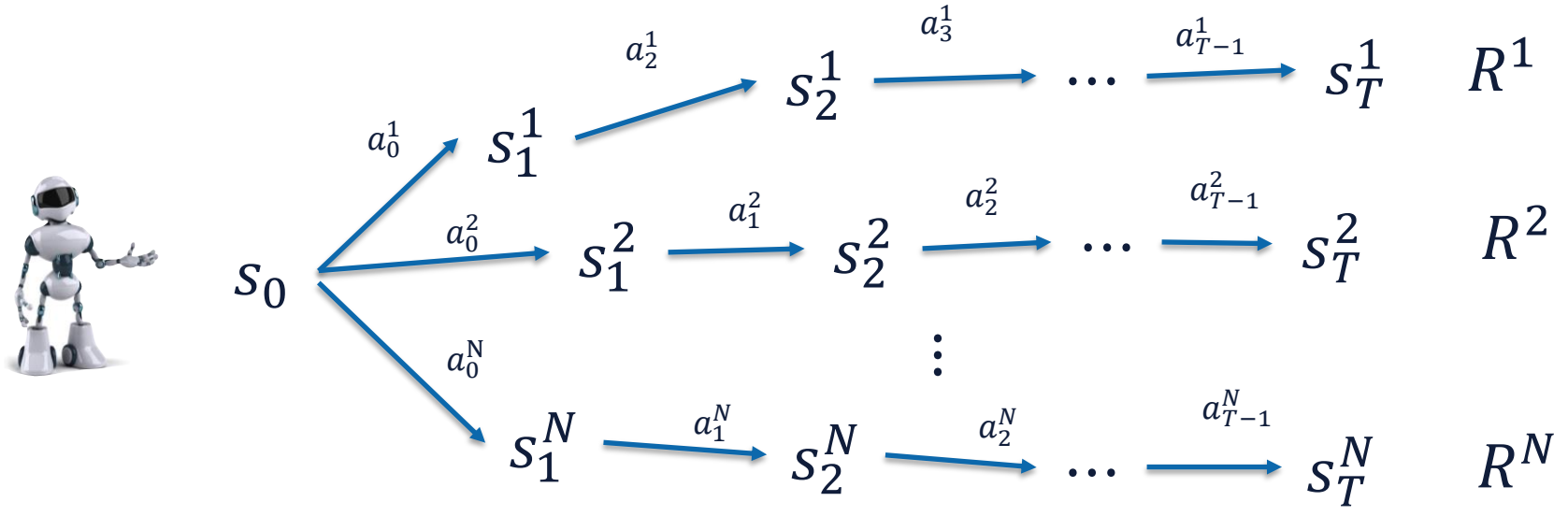
$$a_0^2, a_1^2, a_2^2, \dots, a_{T-1}^2$$

$$\vdots$$

$$a_0^1, a_1^1, a_2^1, \dots, a_{T-1}^1$$

Random Shooting

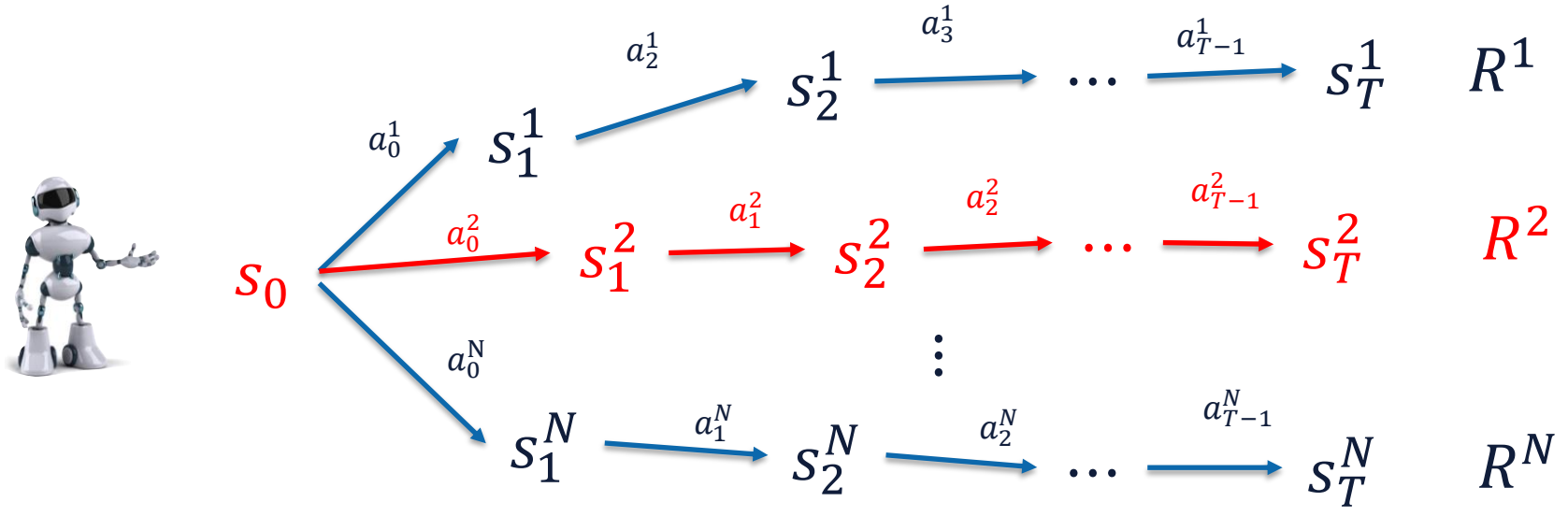
- Evaluate the reward of each action sequence by simulating the model f_θ



"rollout" the forward model $f_\theta(s, a)$

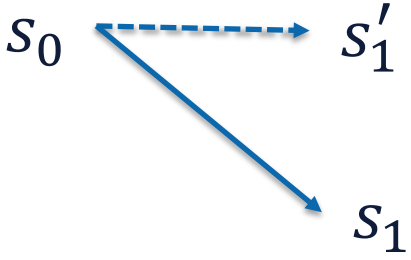
Random Shooting

- Return the best action sequence $a^*_{0:T-1}$ and execute in the real environment



Planning at Each Step

If we execute the searched action sequence $a_0, a_1, a_2, \ldots$ in the environment



Model

Reality

- The action sequence $a_1, a_2, \ldots, a_{T-1}$ maximize the reward from $s_1' = f_\theta(s_0, a_0)$ but not s_1
- Search new actions for state s_1 again!

Model Predictive Control

- Repeat
  - Observe the current state s
  - Sample N random action trajectories
  - Evaluate the reward of each action sequence from s with the model f_θ; Find the best action sequence $\{a_0^k, a_1^k, \ldots, a_{T-1}^k\}$
  - Execute a_0^k in the environment

Cross-Entropy Method

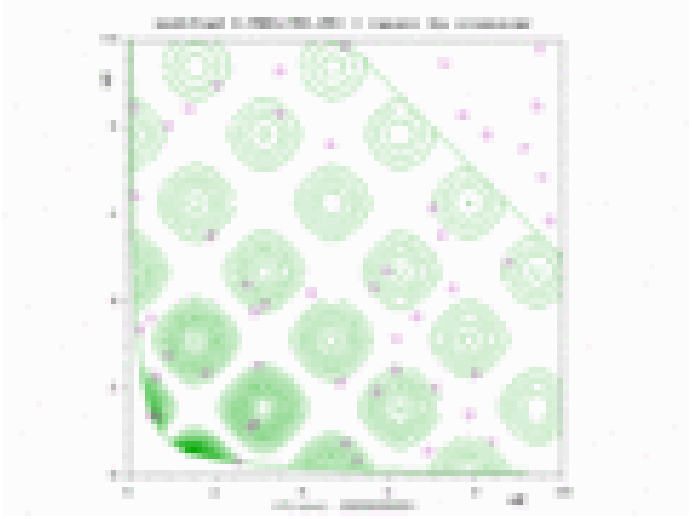
- Black-box Function Optimization

$$\max_{x} f(x)$$

- We have many other choices
  - Cross Entropy Method

Cross-Entropy Method

- Basically the simplest evolutionary algorithm
- Maintain the distribution of solutions



Cross-Entropy Method

- Initialize $\mu \in \mathbb{R}^d, \sigma \in \mathbb{R}^d_{>0}$
- For iteration = 1,2, ...
  - Sample n candidates $x_i \sim N(\mu, \mathrm{diag}(\sigma^2))$
  - For each x_i evaluate its value $f(x_i)$
  - Select the top k of x as elites
  - Fit a new diagonal Gaussian to those samples and update μ, σ

Cross-Entropy Method (in Python)

```python
def cem(f, mean, std, num_iter=10, population_size=100, elite_size=20):
    for i in range(num_iter):
        populations = np.array([np.random.normal() * std + mean for j in range(population_size)])
        values = np.array([f(j) for j in populations])
        elites = populations[values.argsort()[-elite_size:]]
        mean, std = elites.mean(), elites.std()
    return mean
```

Model Predictive Control

- Hyper parameters

$$\mu_a, \sigma_a, n_{\text{iter}}, n_{\text{pop}}, n_{\text{elite}}$$

- Initialize an action sequence $\mu = \{a_i = \mu_a\}_{i<T}$
- Repeat
  - Observe the current state s
  - Search the new action sequence with CEM

$$\{a'_0, a'_1, \dots, a'_{T-1}\} = \text{CEM}(\mu, \{\sigma_a\}_{i<T})$$

  - Execute a'_0 in the environment
  - Update $\mu \leftarrow \{a'_1, a'_2, \dots, a'_{T-1}, \mu_a\}$

Model Predictive Control

- Hyper parameters

$$\mu_a, \sigma_a, n_{\text{iter}}, n_{\text{pop}}, n_{\text{elite}}$$

- Initialize
- Repeat
  - Obse
  - Search the new ac equence with CEM

$$\{a'_0, a'_1, \ldots, a'_{T-1}\} = \text{CEM}(\mu, \{\sigma_a\}_{i<T})$$

  - Execute a'_0 in the environment
  - Update $\mu \leftarrow \{a'_1, a'_2, \ldots, a'_{T-1}, \mu_a\}$

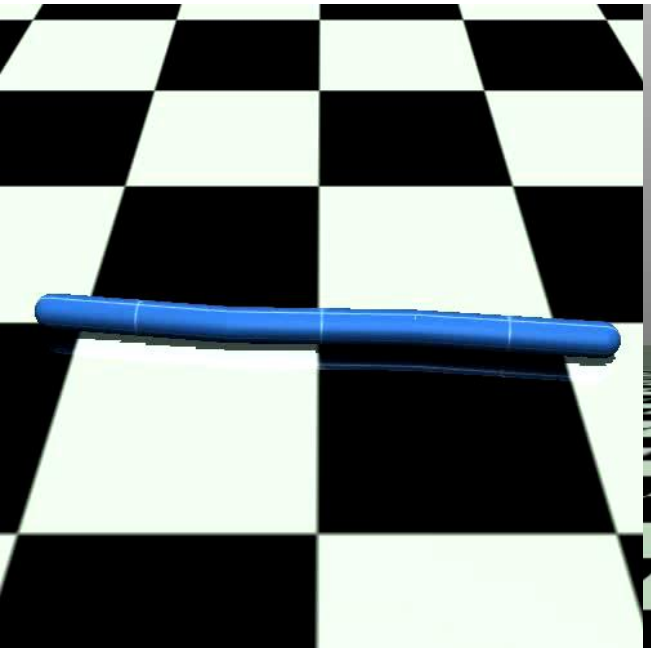
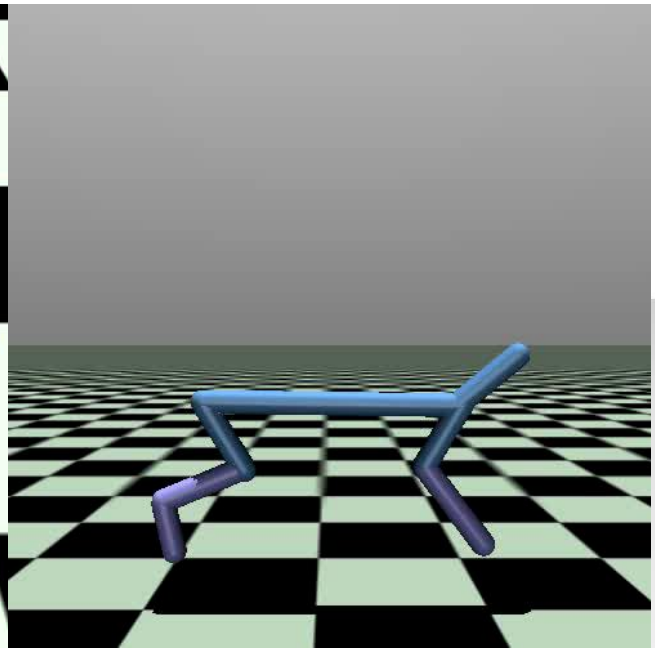
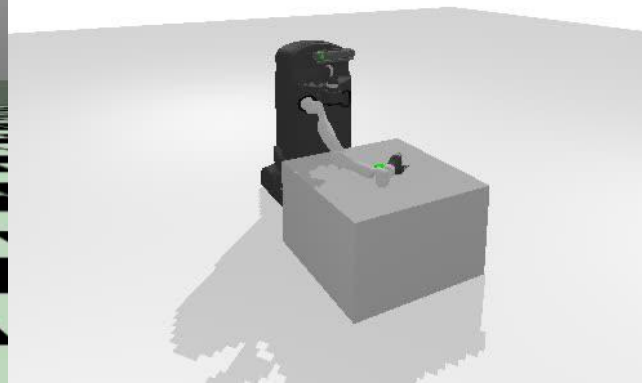
```python
def cem(f, mean, std, num_iter=10, population_size=100, elite_size=20):
    for i in range(num_iter):
        populations = np.array([np.random.normal() * std + mean for j in range(population_size)])
        values = np.array([f(j) for j in populations])
        elites = populations[values.argsort()[-elite_size:]]
        mean, std = elites.mean(), elites.std()
    return mean
```

Notes

- CEM performs well for most control tasks
- Instead of searching for action sequence, we can also search for the parameters of the network
- General “Gradient Descent”
- CEM and Random shooting work poorly for very long horizons T or dense reward

Performance of CEM

- When the model is known

Comparisons

- On-policy methods: Policy
- Off-policy methods: Value/Q
- Model-based methods: Model

$$\text{Model} \Rightarrow \text{Value} \Rightarrow \text{Policy}$$

Comparisons

- How difficult to model it
  - Q Value > Policy
  - Model depends on the priors
- Robustness
  - Model < Q Value < Policy
- Time complexity
  - Model > Q/Policy
- Data-efficient/Generalization
  - Model > Q Value > Policy

Conclusion

- Very few data / We know the model well
  - Model-based methods
- We can’t model the environment and we don’t want to sample too much
  - Off-policy methods
- We have enough time/money
  - Off-policy + On-policy methods

Sample-based Motion Planning

Topics

- **Problem formulation**
- **Probabilistic roadmap method (PRM)**
- **Rapidly exploring random trees (RRT)**

Topics

- **Problem formulation**
- Probabilistic roadmap method (PRM)
- Rapidly exploring random trees (RRT)

Configuration Space

- Configuration space(C-space) C is a subset of $\mathbb{R}^n$ containing all possible states of the system(state space in RL).
- $C_{free} \subseteq C$ contains all valid states.
- $C_{obs} \subseteq C$ represents obstacles.
- Examples:
  - All valid poses of a robot.
  - All valid joint values of a robot.
  - …

Motion Planning

- Problem:
  - Given a configuration space C_{free}
  - Given start state q_{start} and goal state q_{goal} in C_{free}
  - Calculate a sequence of actions that leads from start to goal
- Challenge:
  - Need to avoid obstacles
  - Long planning horizon
  - High-dimensional planning space

Motion Planning

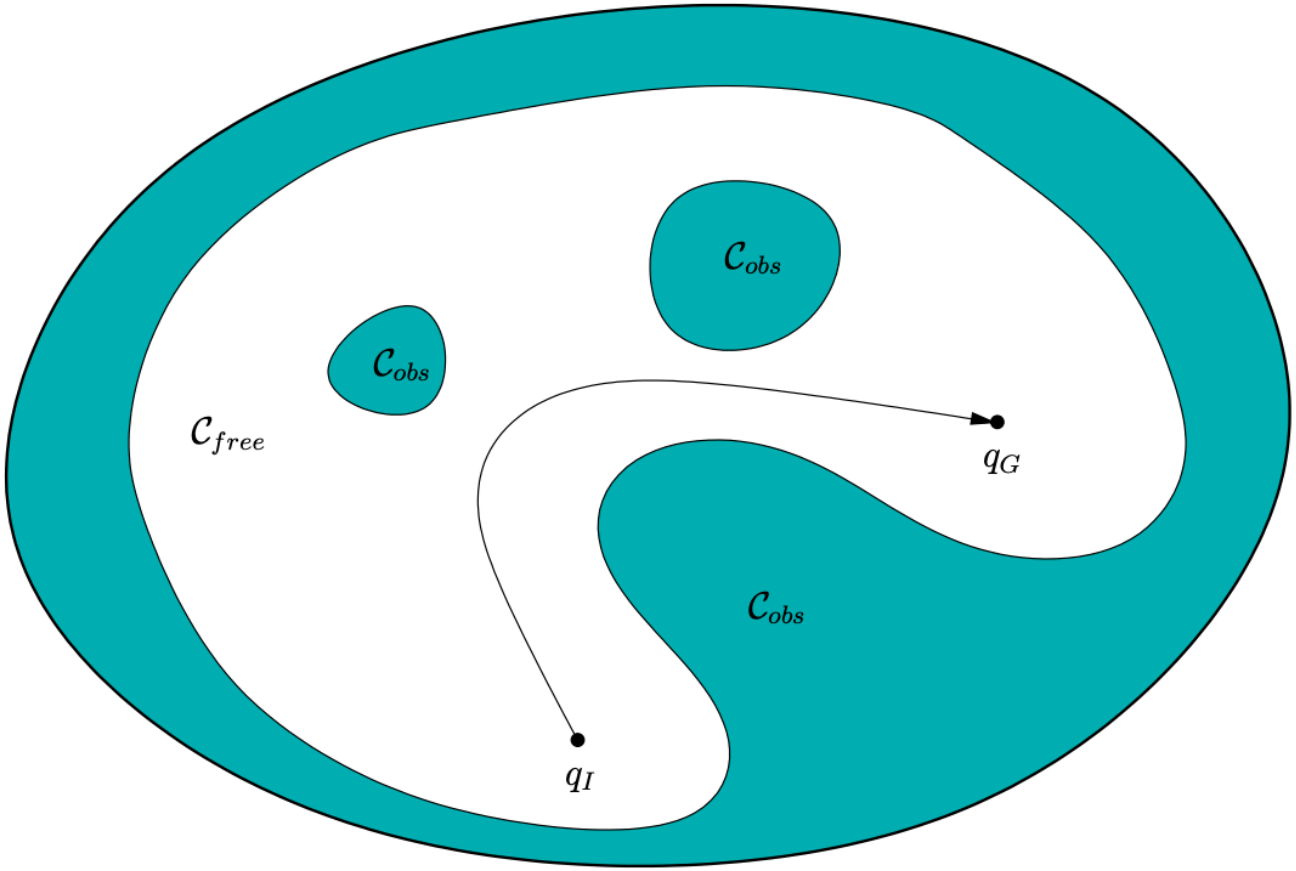
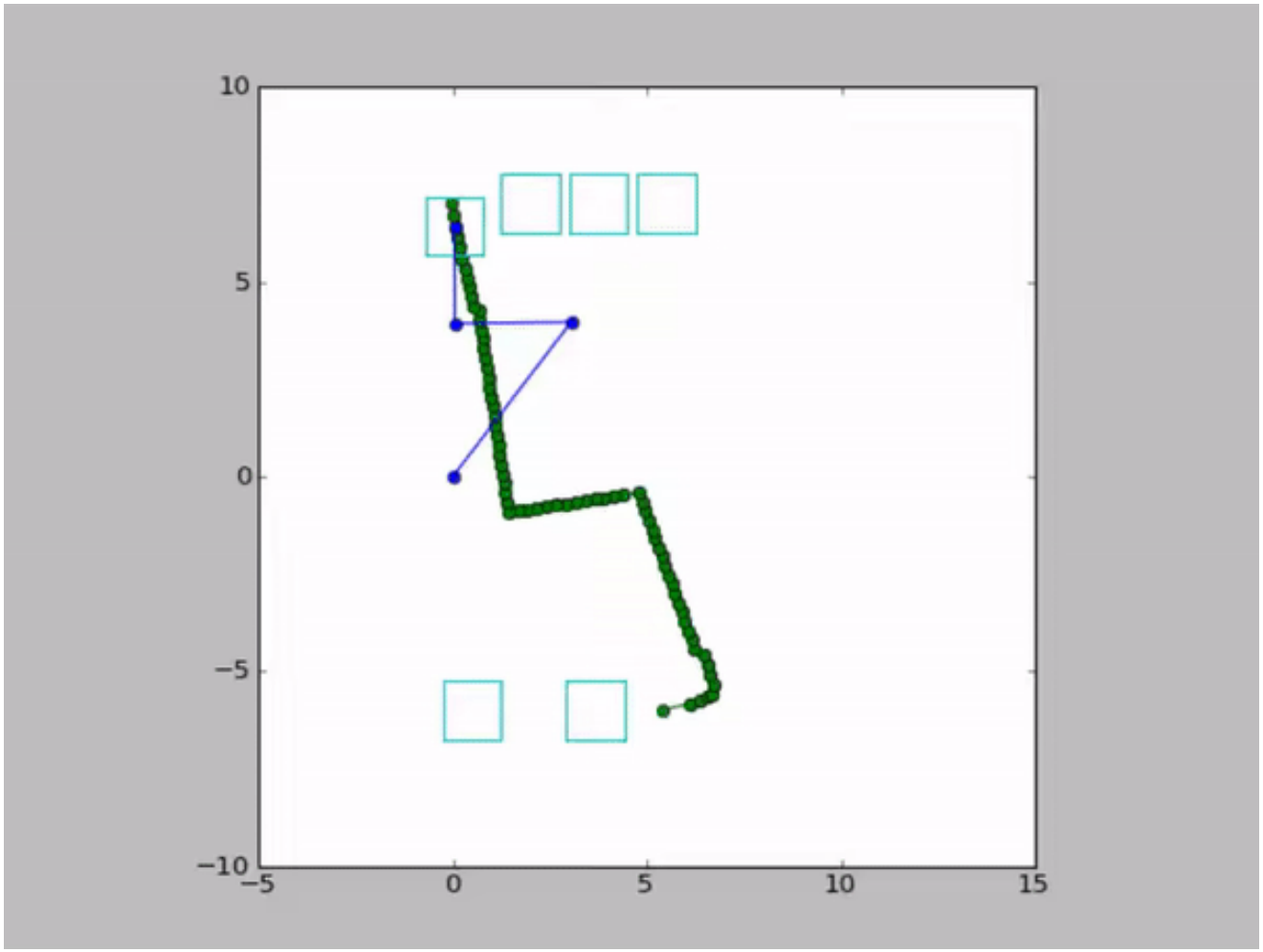


Figure 4.11: The basic motion planning problem is conceptually very simple using C-space ideas. The task is to find a path from q_I to q_G in $\mathcal{C}_{free}$. The entire blob represents $\mathcal{C} = \mathcal{C}_{free} \cup \mathcal{C}_{obs}$.

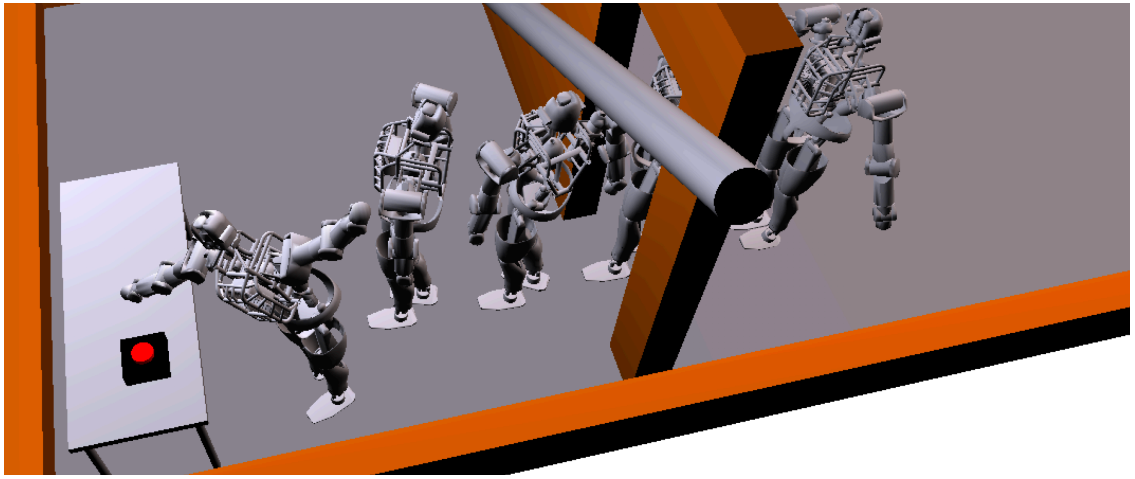
LaValle, Steven M. *Planning algorithms*. Cambridge university press, 2006.

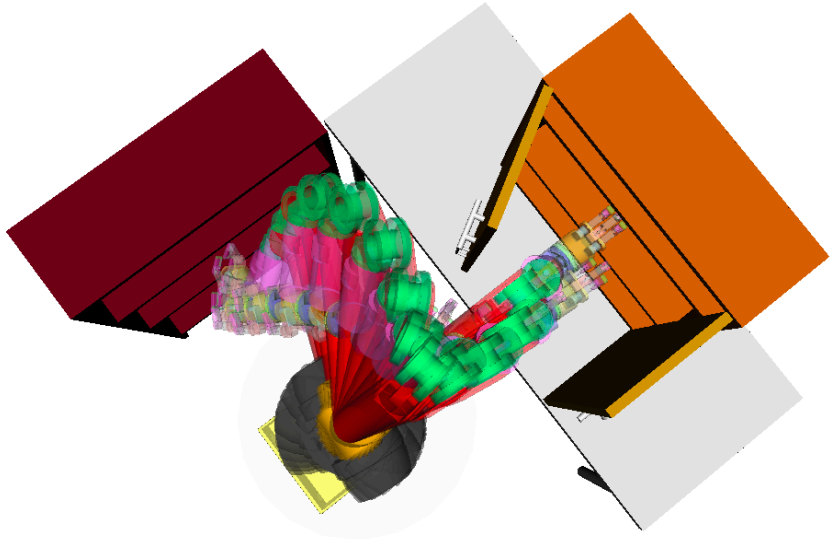
Examples



https://medium.com/@theclassytim/robotic-path-planning-rrt-and-rrt-212319121378

Examples





Ratliff N, Zucker M, Bagnell J A, et al. CHOMP: Gradient optimization techniques for efficient motion planning, ICRA 2009
Schulman, John, et al. Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization, RSS 2013

Sample-based algorithm

- The key idea is to explore a smaller subset of possibilities randomly without exhaustively exploring all possibilities.
- Pros:
  - Probabilistically complete
  - Solve the problem after knowing partial of C_{free}
  - Apply easily to high-dimensional C-space
- Cons:
  - Requires find path between two close points
  - Does not work well when the connection of C_{free} is bad
  - Never optimal

Topics

- Problem formulation
- **Probabilistic roadmap method (PRM)**
- Rapidly exploring random trees (RRT)

Probabilistic roadmap(PRM)

- The algorithm contains two stages:
  - Construction phase
    - Randomly sample states in C_{free}
    - Connect every sampled state to its neighbors
    - Connect the start and goal state to the graph
  - Query phase
    - Run path finding algorithms like Dijkstra

Kavraki, Lydia E., et al. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces." *IEEE transactions on Robotics and Automation* 12.4 (1996): 566-580.

Rejection Sampling

- Aim to sample uniformly in C_{free}.
- Method
  - Sample uniformly over C.
  - Reject the sample not in the feasible area.

Pipeline

Input: n: number of sampled nodes in the roadmap, k: number of closest neighbours to examine for each configuration, q_{start}, q_{goal}.

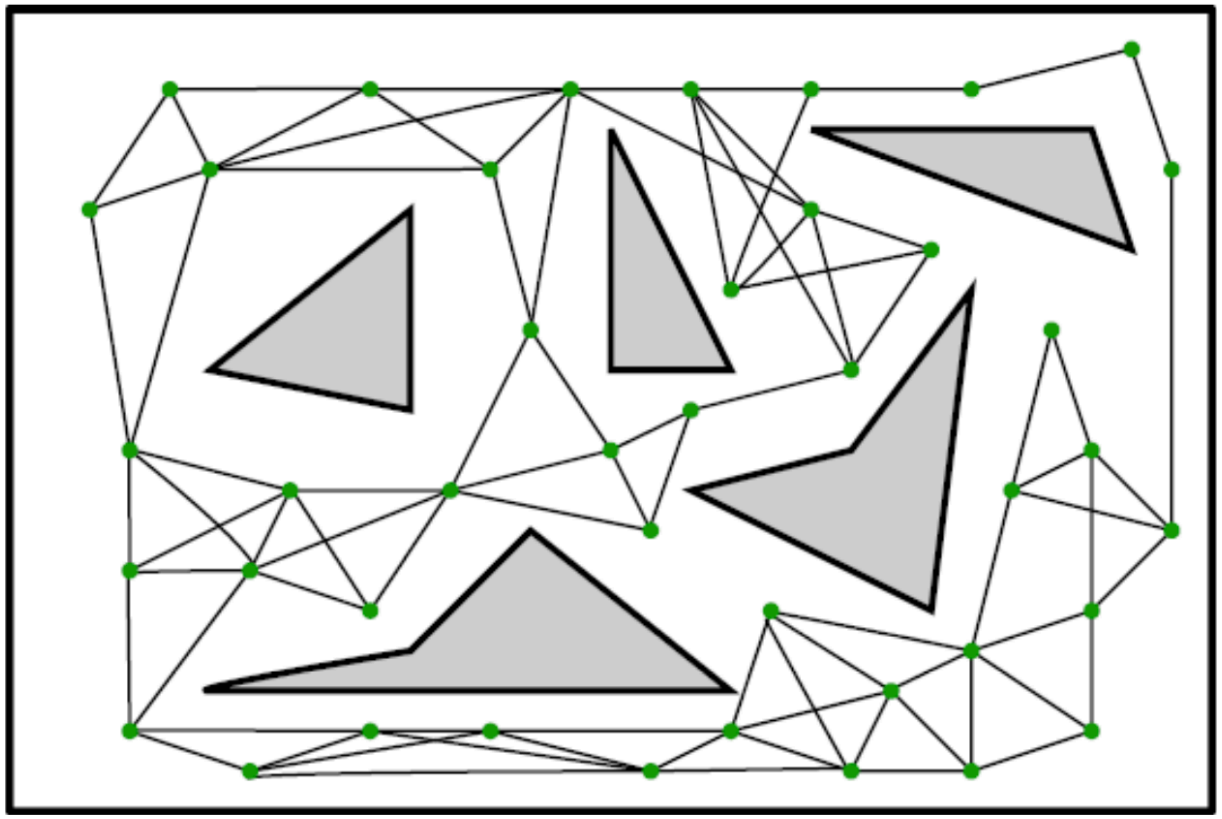
$V \leftarrow \{q_{start}, q_{goal}\}$;
$E \leftarrow \emptyset$;
while $|V| < n$ **do**
    repeat
        $q \leftarrow$ a random configuration in C.
    until q *is in* C_{free};
end
foreach $q \in V$ **do**
    $N_q \leftarrow$ the k closest neighbours of q chosen from V according to a distance function;
    foreach $q' \in N_q$ **do**
        if $(q, q') \notin E$ *and* $(q, q') \in C_{free}$ **then**
            $E \leftarrow E \cup \{(q, q')\}$
        end
    end
end
Find a path from q_{start} to q_{goal} with Dijkstra algorithm;

Challenges

- Connect neighboring points:
  - In general it requires solving boundary value problem.
- Collision checking:
  - It takes a lot of time to check if the edges are in the configuration space.

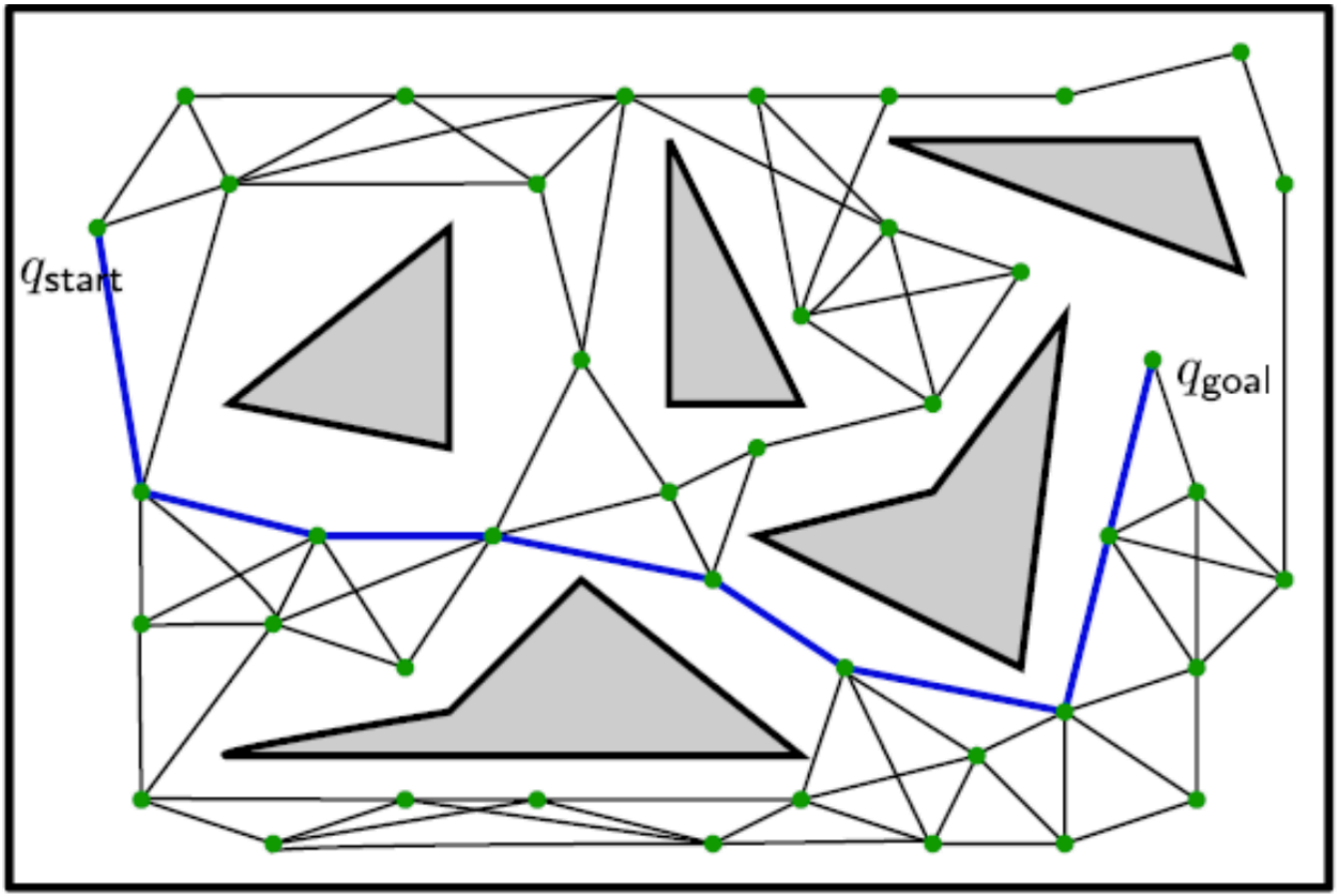
Example

- PRM generates a graph $G = (V, E)$ such that every edge is in the configuration space without colliding with obstacles.



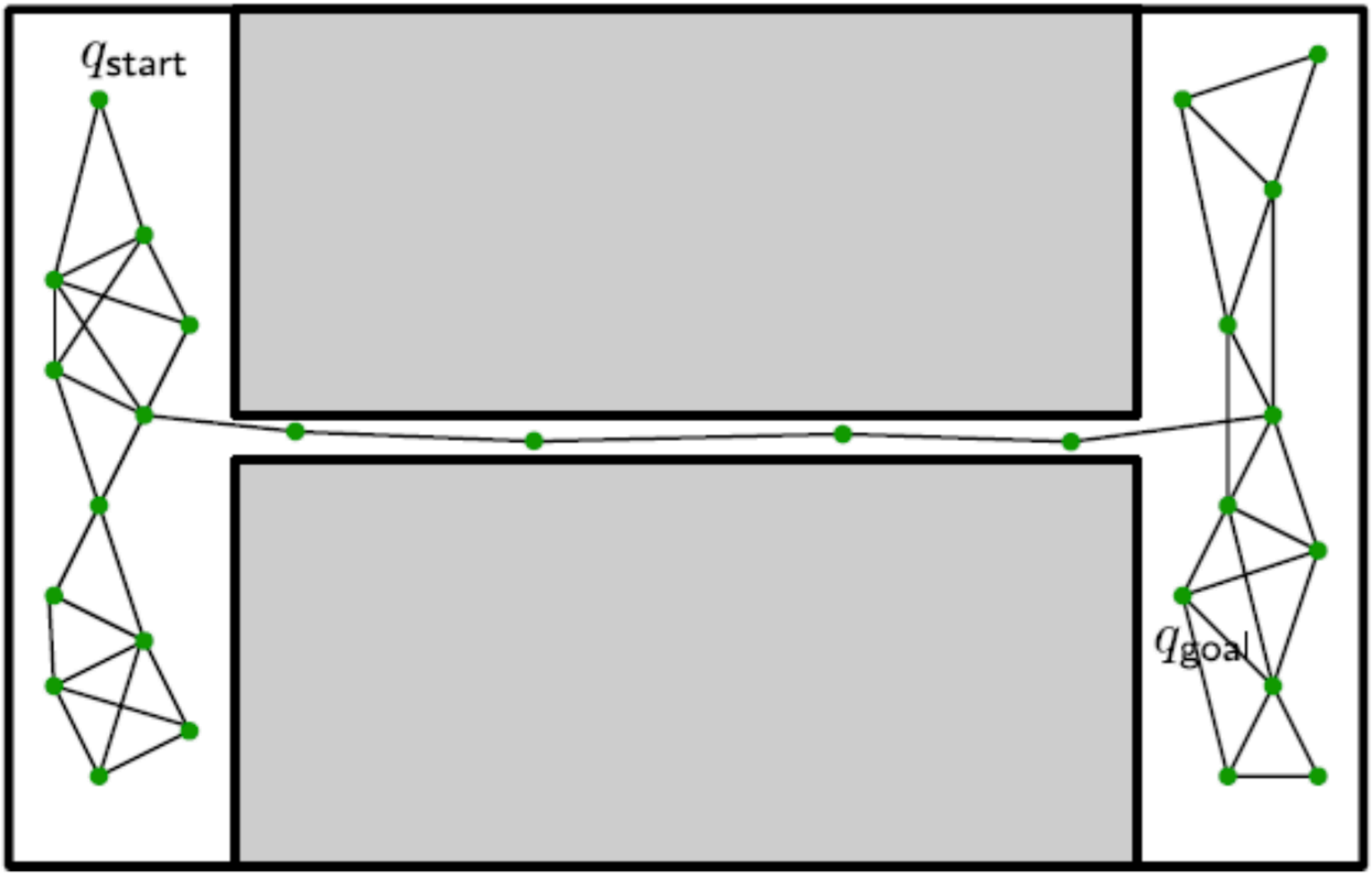
Example

- Find the path from start state q_{start} to goal state q_{goal}



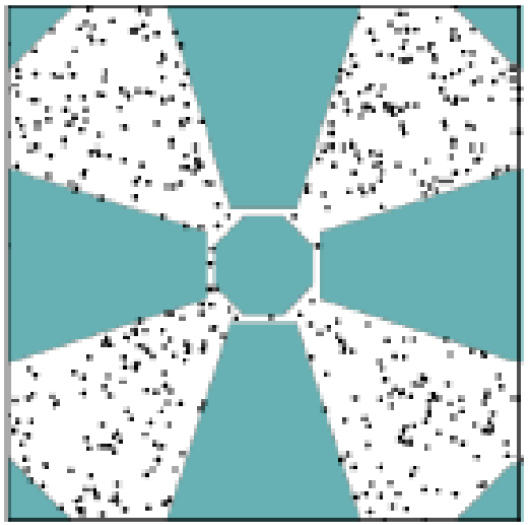
Limitations:narrow passages

- It is unlikely to sample the points in the narrow bridge

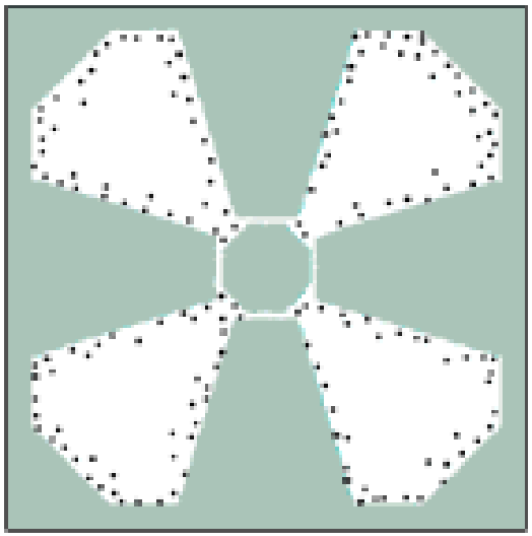


Gaussian Sampling

- Generate one sample q_1 uniformly in the configuration space
- Generate another sample q_2 from a Gaussian distribution $\mathcal{N}(q_1, \sigma^2)$
- If $q_1 \in C_{free}$ and $q_2 \notin C_{free}$ then add q_1



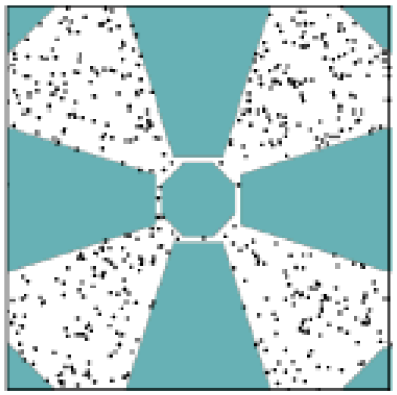
Uniform sampling



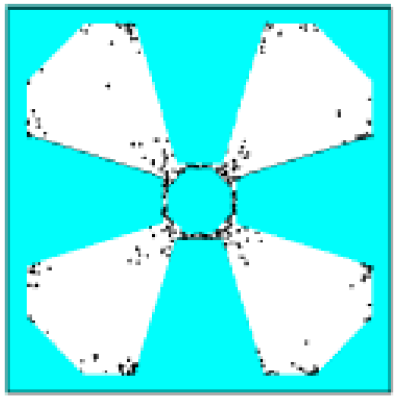
Gaussian sampling

Bridge sampling

- Generate one sample q_1 uniformly in the configuration space
- Generate another sample q_2 from a Gaussian distribution $\mathcal{N}(q_1, \sigma^2)$
- $q_3 = \frac{q_1 + q_2}{2}$
- If q_1, q_2 are not in C_{free} then add q_3



Uniform sampling



Bridge sampling

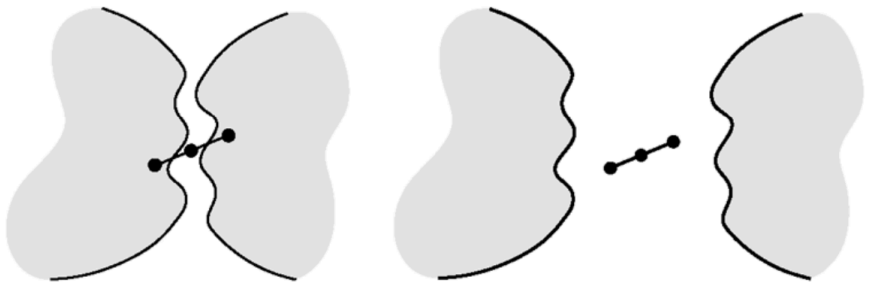


Fig. 2. Building short bridges is much easier in narrow passages (left) than in wide-open free space (right).

Sun, Zheng, et al. "Narrow passage sampling for probabilistic roadmap planning." IEEE Transactions on Robotics 21.6 (2005): 1105-1115.

Topics

- Problem formulation
- Probabilistic roadmap method (PRM)
- **Rapidly exploring random trees (RRT)**

Rapidly-exploring random tree(RRT)

- RRT grows a tree rooted at the start state by using random samples from configuration space.
- As each sample is drawn, a connection is attempted between it and the nearest state in the tree. If the connection is in the configuration space, this results in a new state in the tree.

Extend operation

Figure 2: The basic RRT construction algorithm.

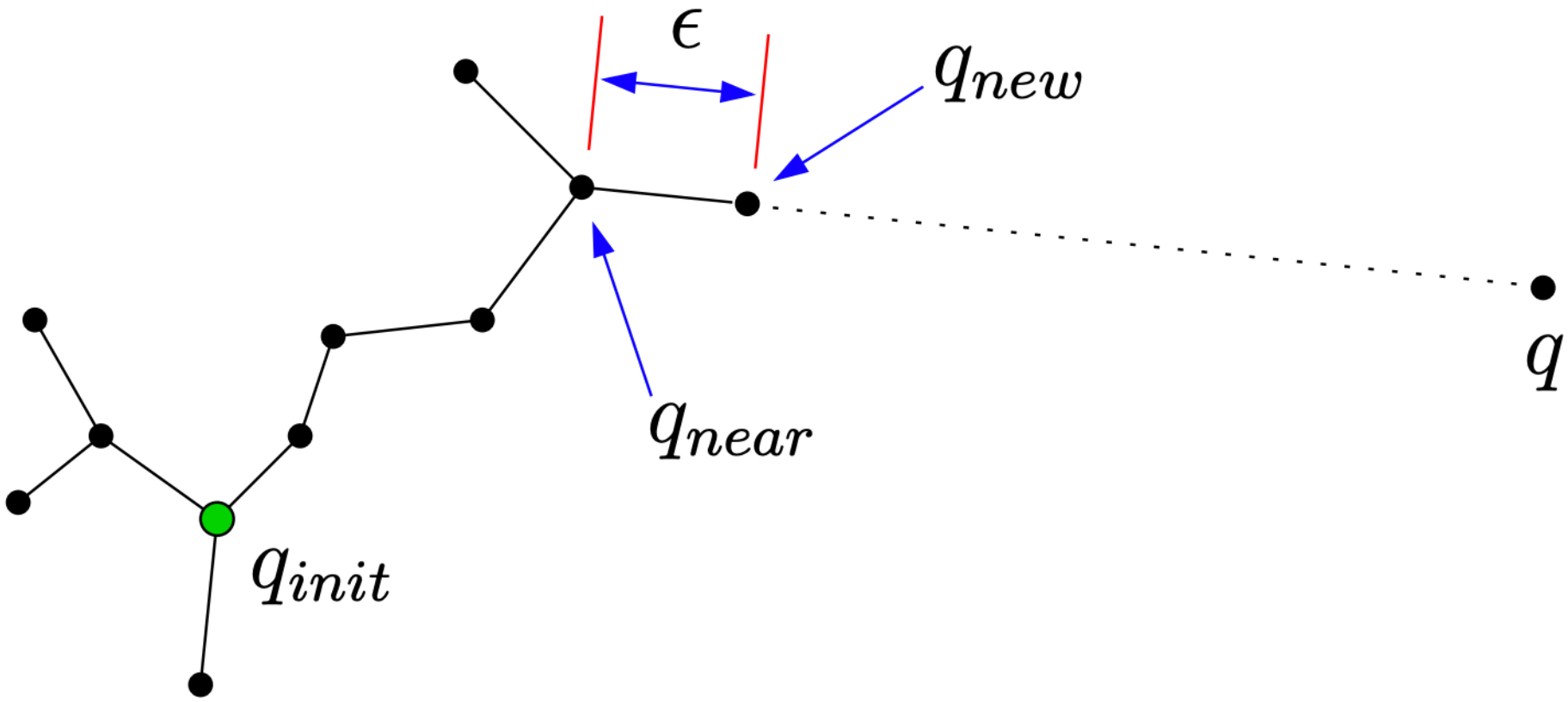


Figure 3: The EXTEND operation.

Pipeline

Input: n: number of sampled nodes in the tree, ϵ is the stepsize, β is the probability of sampling q_{goal}, q_{start}, q_{goal}.

$V \leftarrow \{q_{start}\}$;

$E \leftarrow \emptyset$;

for $i = 1 \rightarrow n$ **do**

  if *rand*$(0, 1)$ < β **then**

    $q_{target} \rightarrow q_{goal}$

  end

  else

    $q_{target} \rightarrow$ uniformly random sample from C_{free}

  end

  $q_{near} \rightarrow$ nearest neighbor of q_{target} in V ;

  $q_{new} \rightarrow q_{near} + \frac{\epsilon}{|q_{near} - q_{target}|} (q_{near} - q_{target})$;

  if $q_{new} \in C_{free}$ *and* $(q_{near}, q_{new}) \in C_{free}$ **then**

    $V \rightarrow V \cup \{q_{new}\}$;

    $E \rightarrow E \cup \{(q_{near}, q_{new})\}$;

  end

end

Find a path from q_{start} to q_{goal} with Dijkstra algorithm;

Challenges

- Find nearest neighbor in the tree
  - We need to support online quick query
  - Examples: KD Trees
- Need to choose a good ϵ to expand the tree efficiently
  - Large ϵ: hard to generate new samples
  - Small ϵ: too many samples in the tree

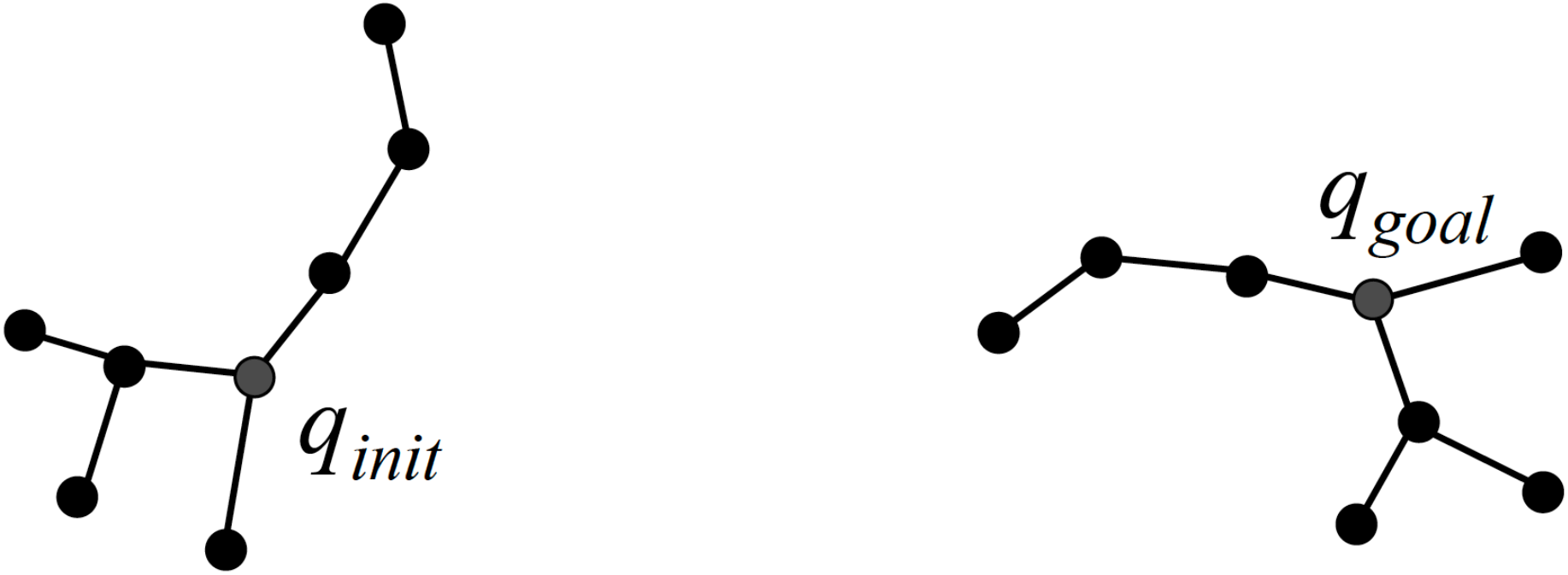
Examples

RRT-Connect

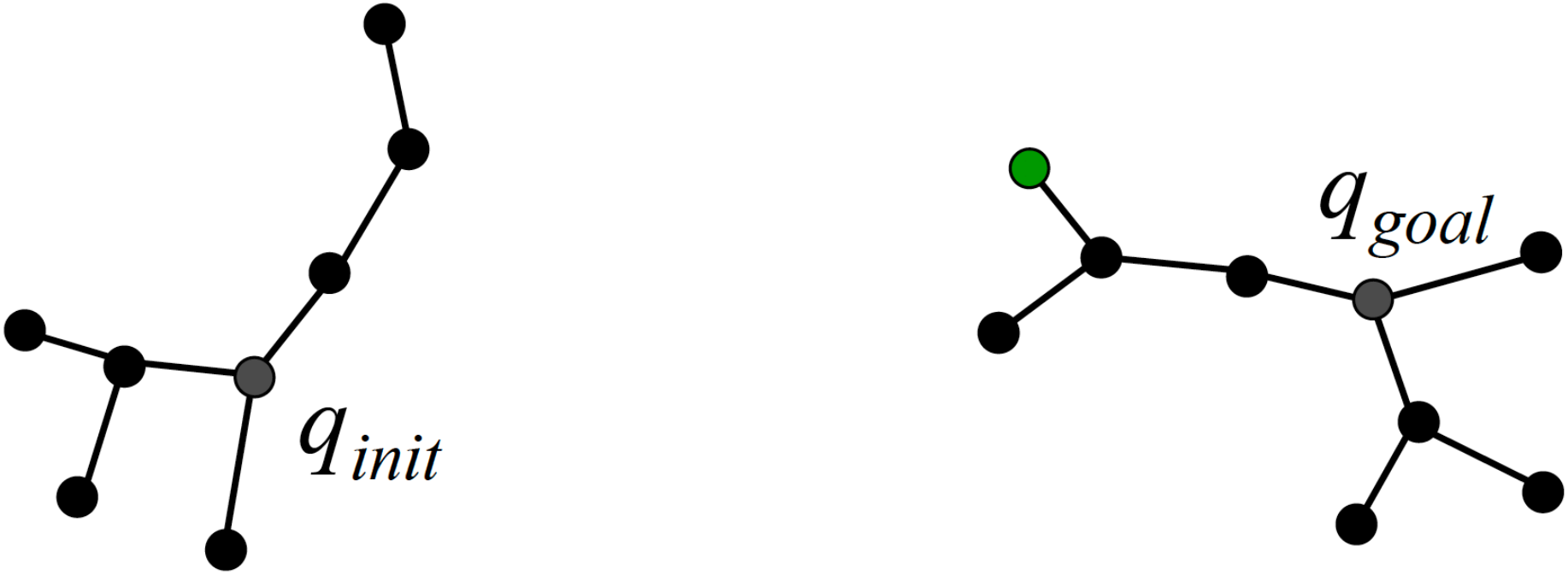
- Grow two trees starting from q_{start} and q_{goal} respectively instead of just one.
- Grow the trees towards each other rather than random configurations
- Use stronger greediness by growing the tree with multiple epsilon steps instead of a single one.

Kuffner, James J., and Steven M. LaValle. "RRT-connect: An efficient approach to single-query path planning." Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065). Vol. 2. IEEE, 2000.

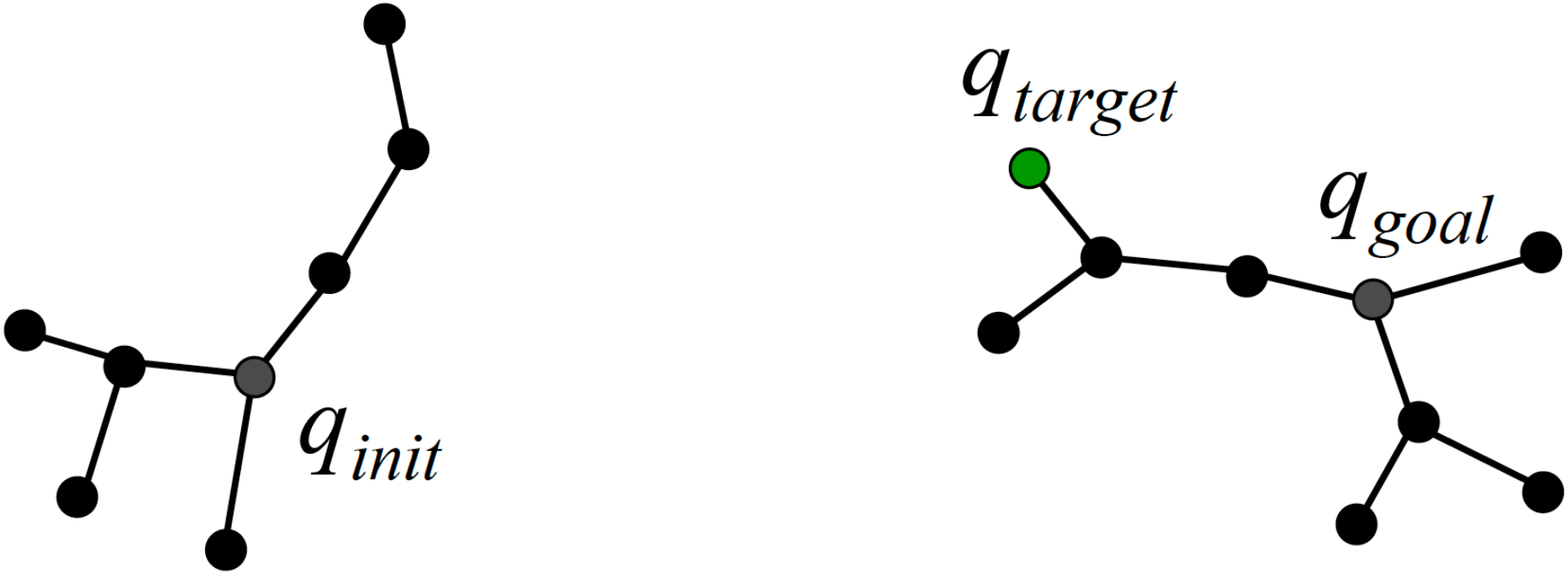
A single RRT-Connect iteration



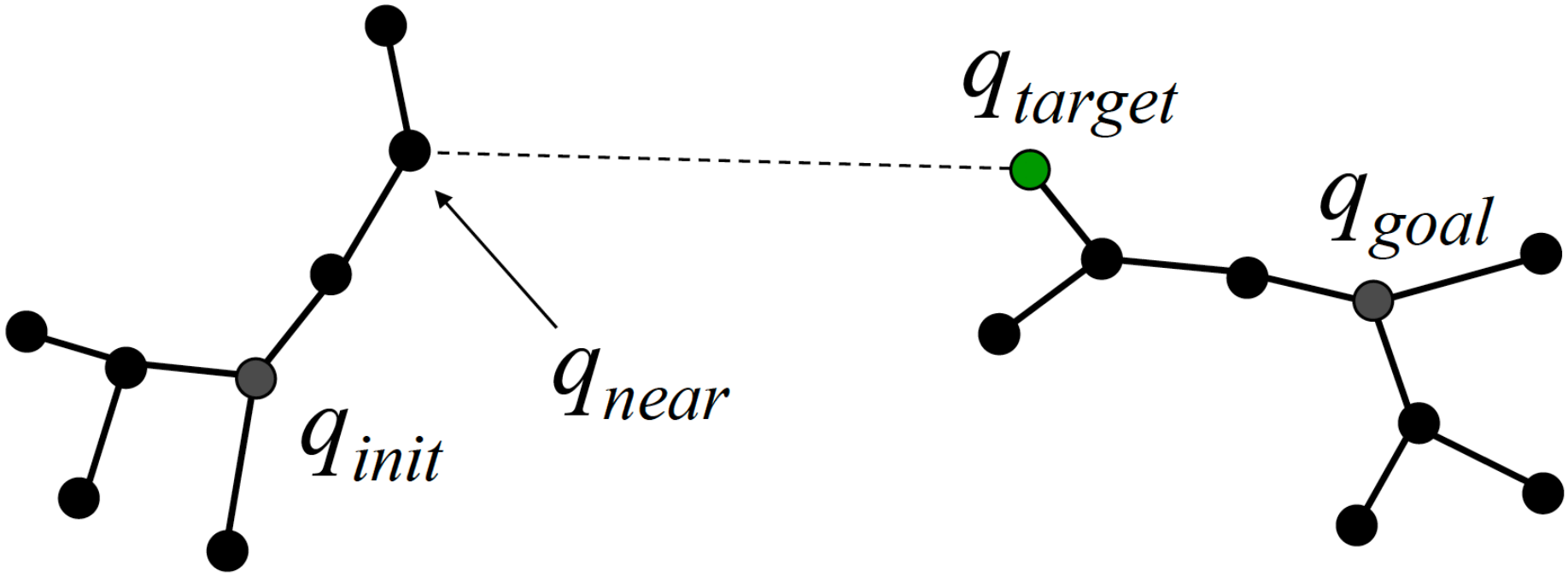
One tree grown using a random target



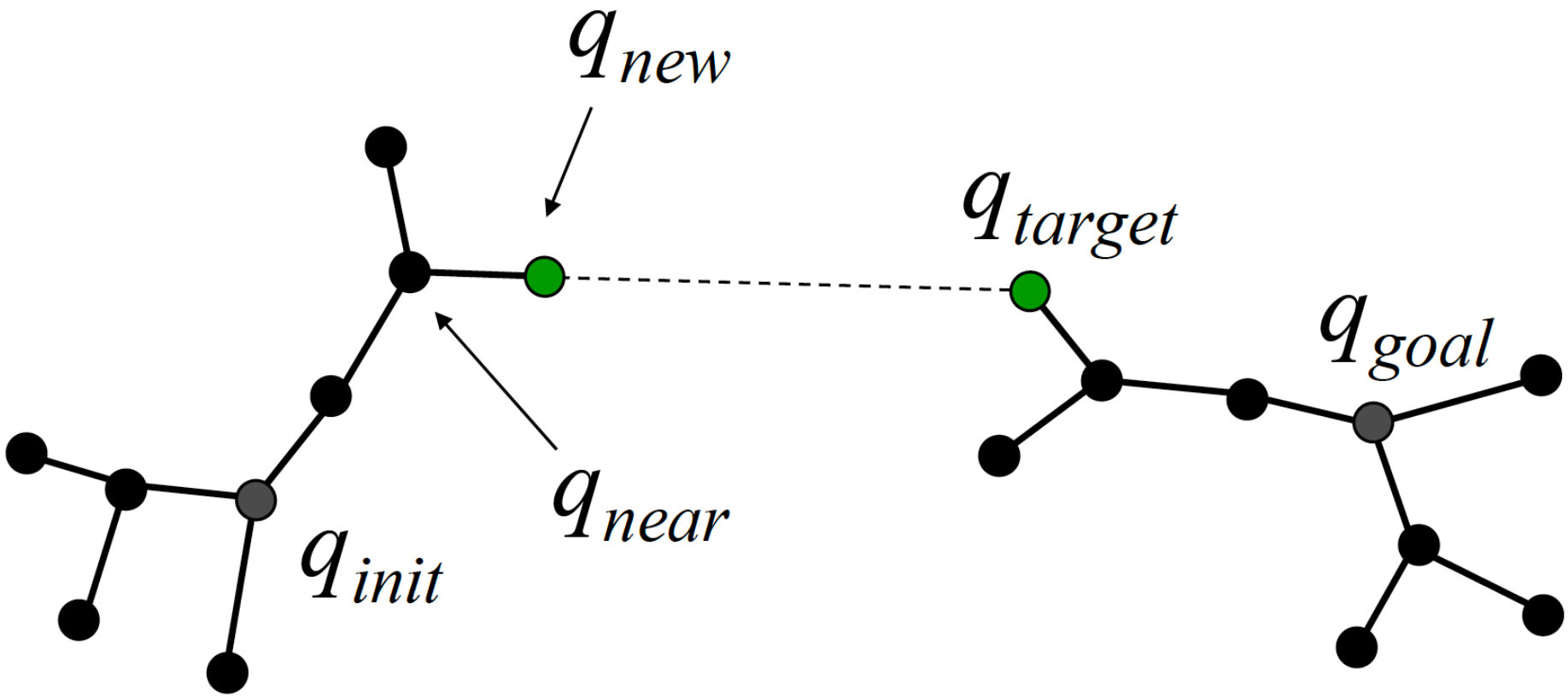
New node becomes target for the other tree



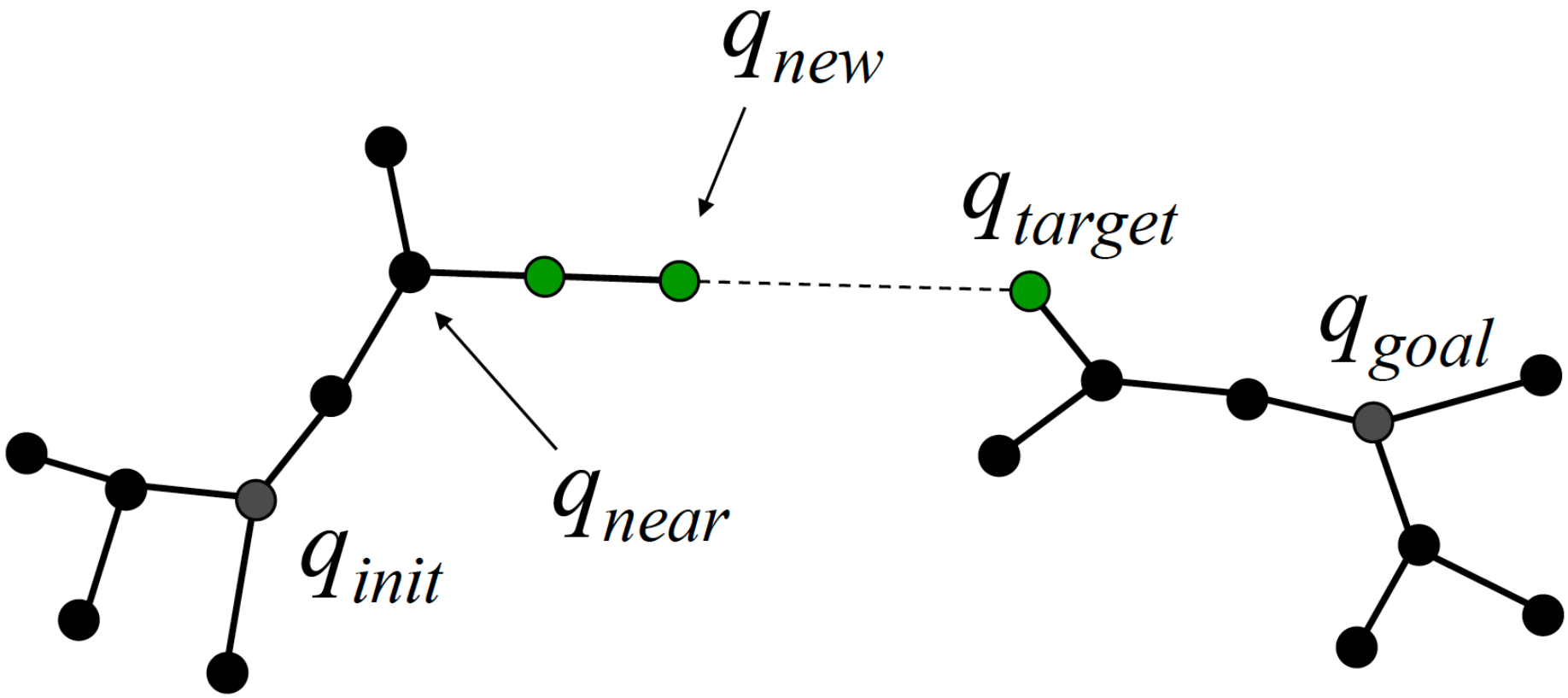
Calculate nearest node to target



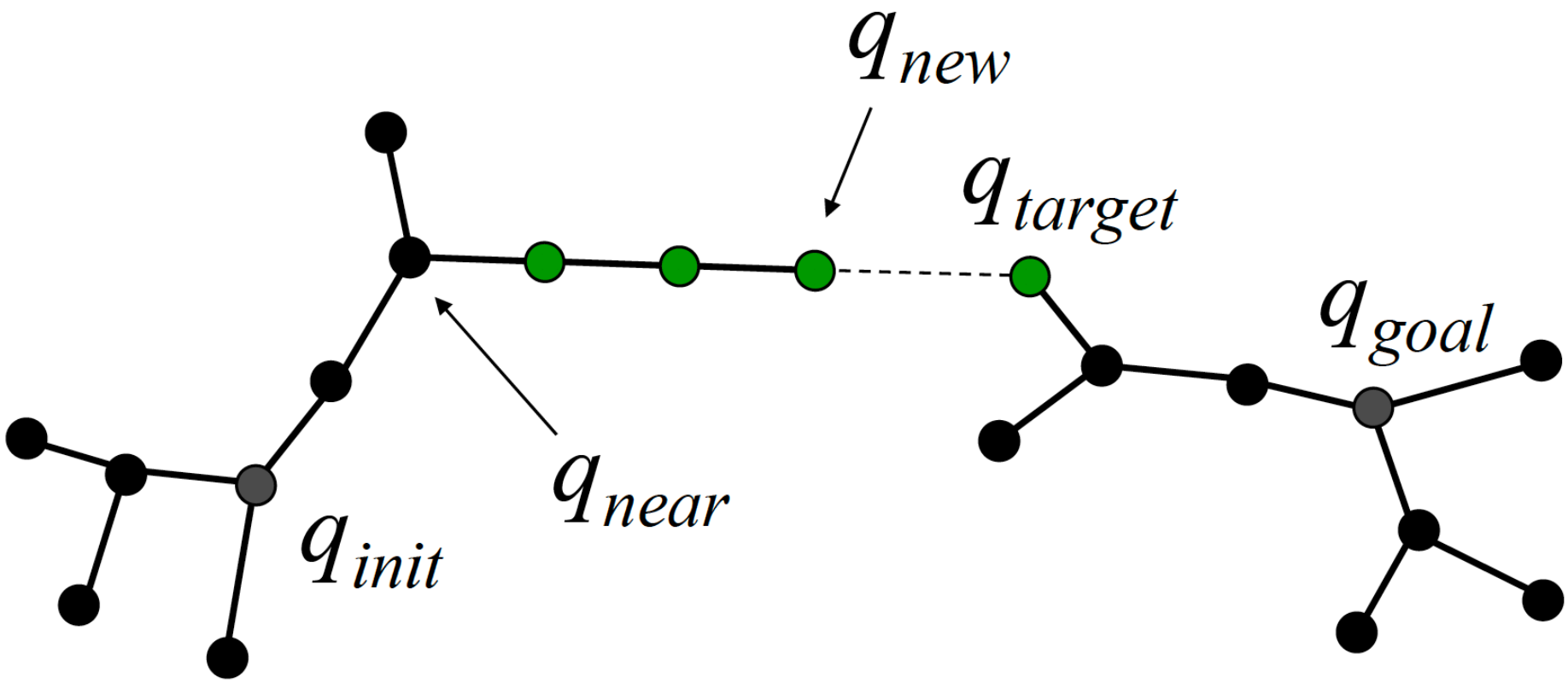
Try to add the new node



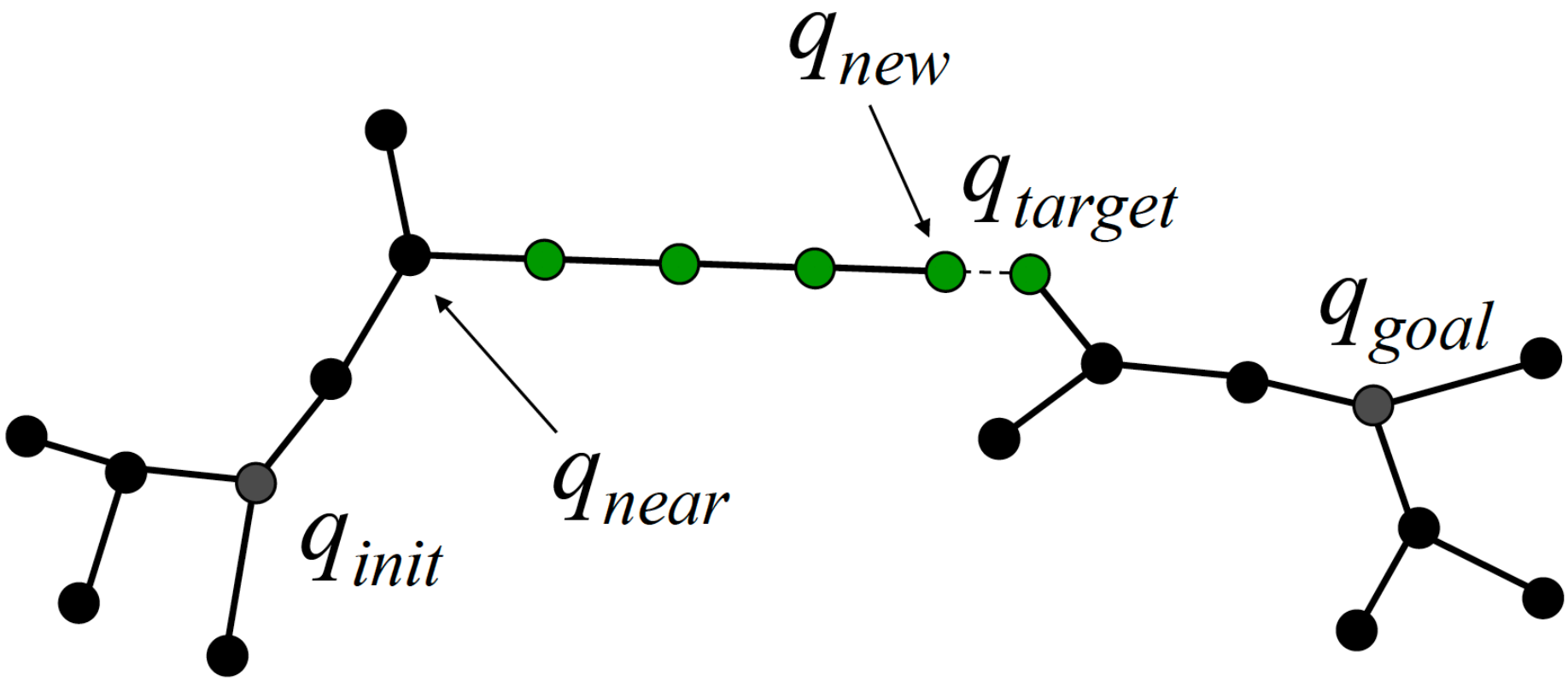
If successful, keep extending



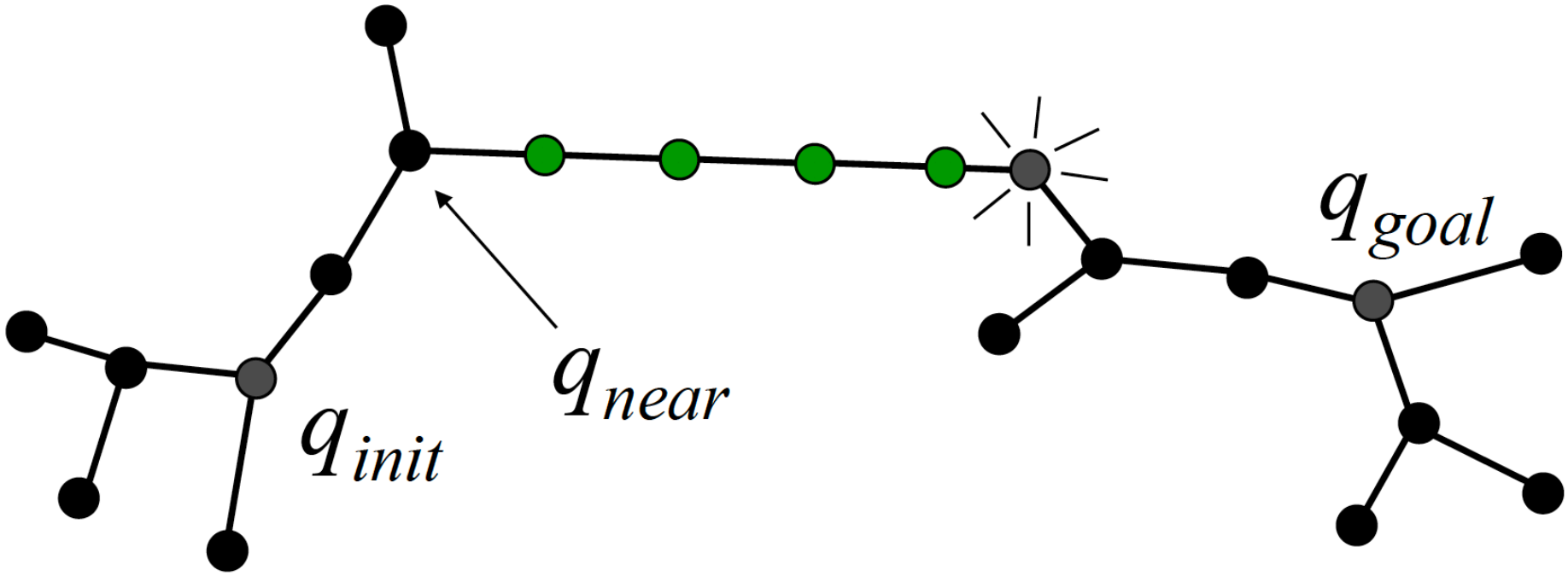
If successful, keep extending



If successful, keep extending



Path found if branch reaches target

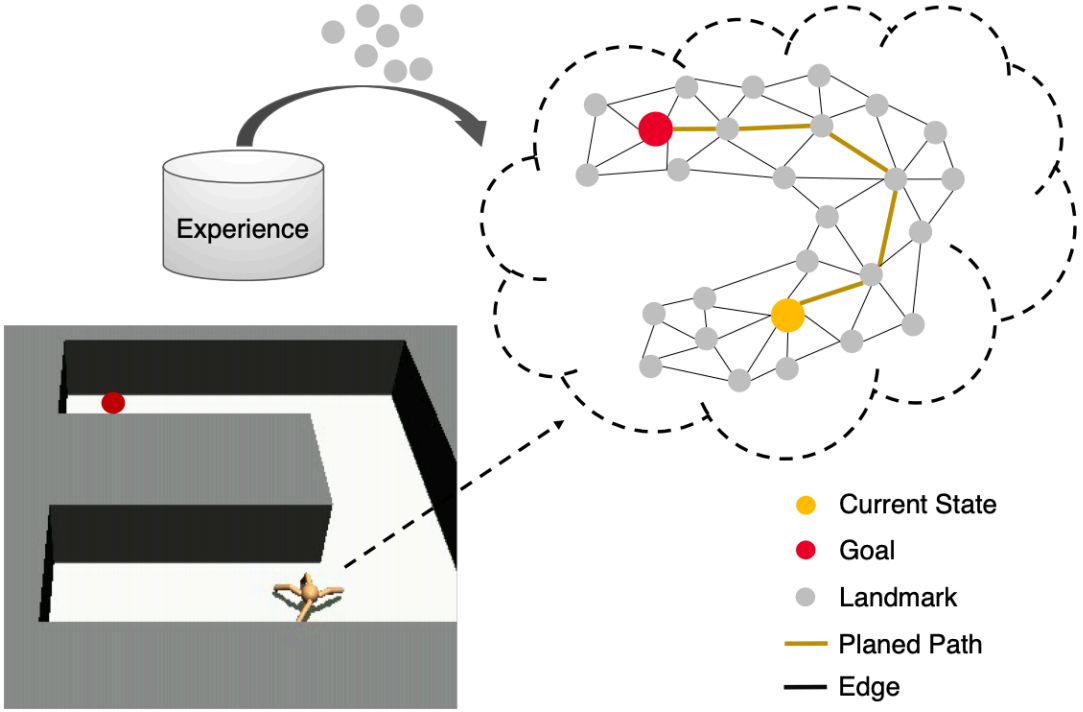


Local planners

- Both RRT and PRM rely on a local planner which gives a sequence of action to connect two states.
- RL can give local planners without solving the dynamics equations explicitly.
- RL cannot solve long term high dimensional planning problem efficiently due to the exploration.

PRM+RL

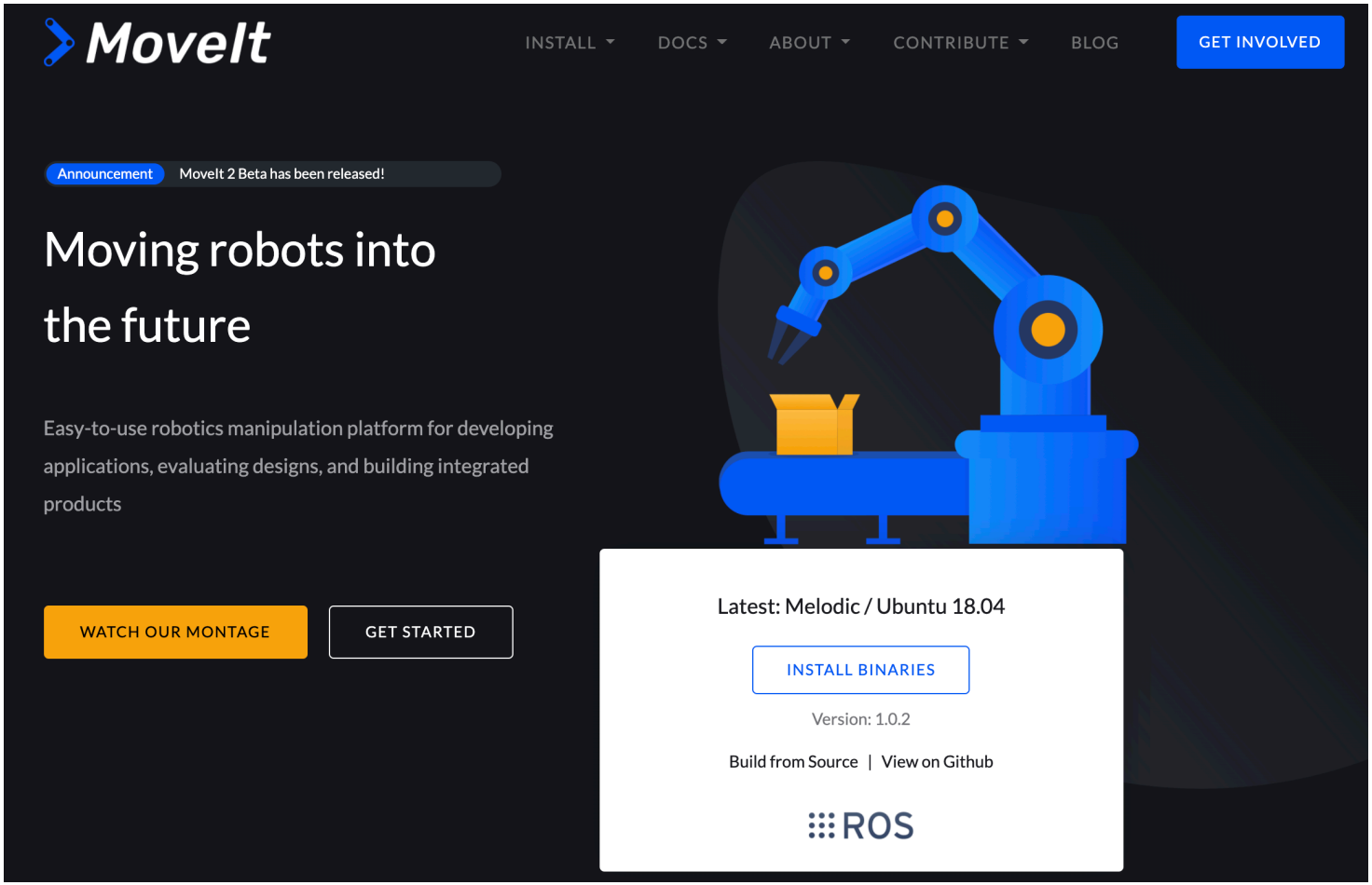
- Sample some landmarks and build a graph in the configuration space like PRM.
- Find the planning path.
- Action sequence is found by RL algorithm like DDPG.



Huang, Zhiao, Fangchen Liu, and Hao Su. "Mapping state space using landmarks for universal goal reaching." Advances in Neural Information Processing Systems. 2019.

Moveit

- Moveit is package on ROS
- A lot of useful motion planning algorithm can be found



Summary

- Motion planning
- PRM
  - Sampling: Gaussian sampling, bridge sampling
- RRT
  - RRT-Connect
- Motion planning + RL
  - PRM + RL