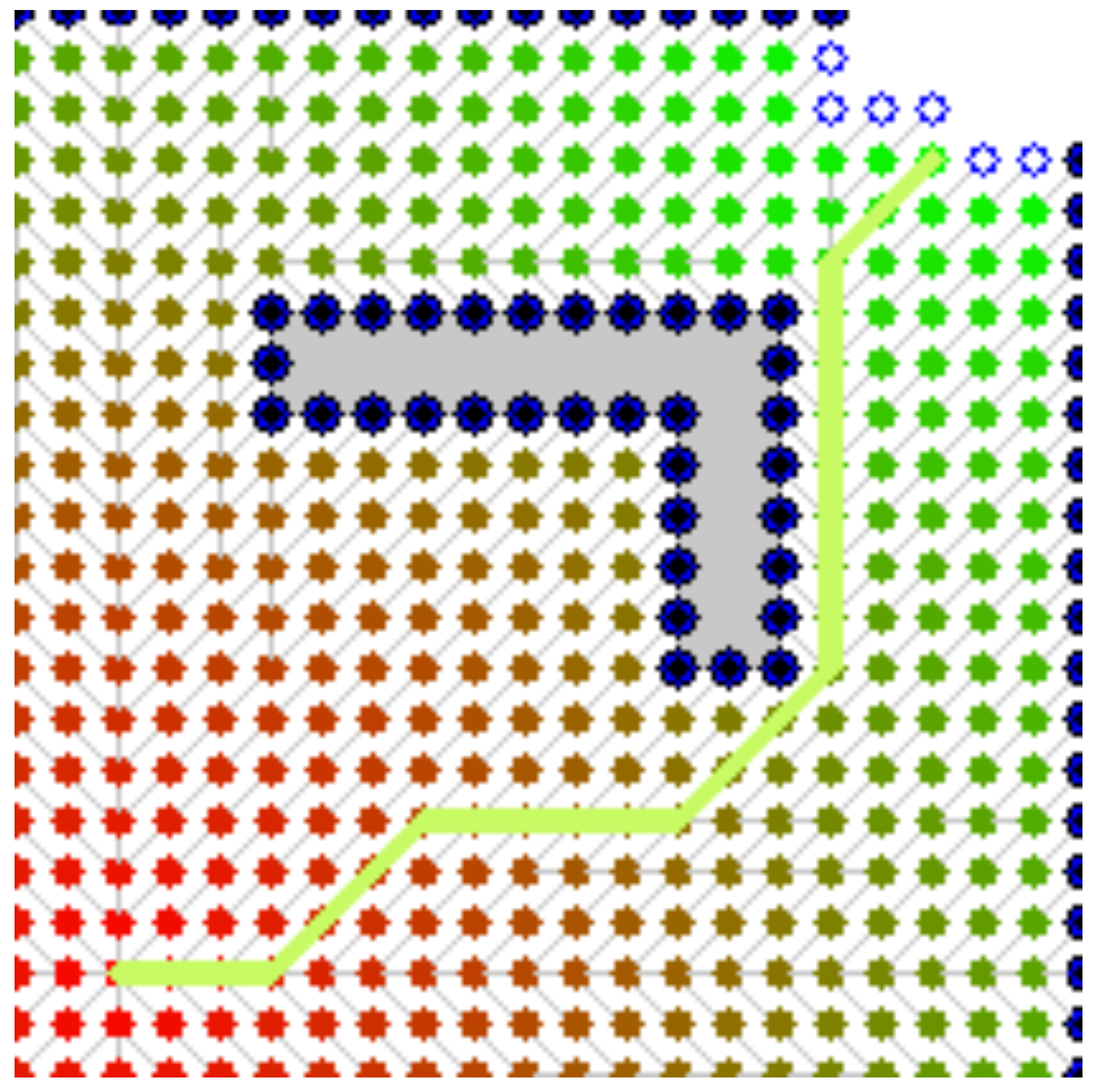
CSE 478 Robot Autonomy
Roadmaps

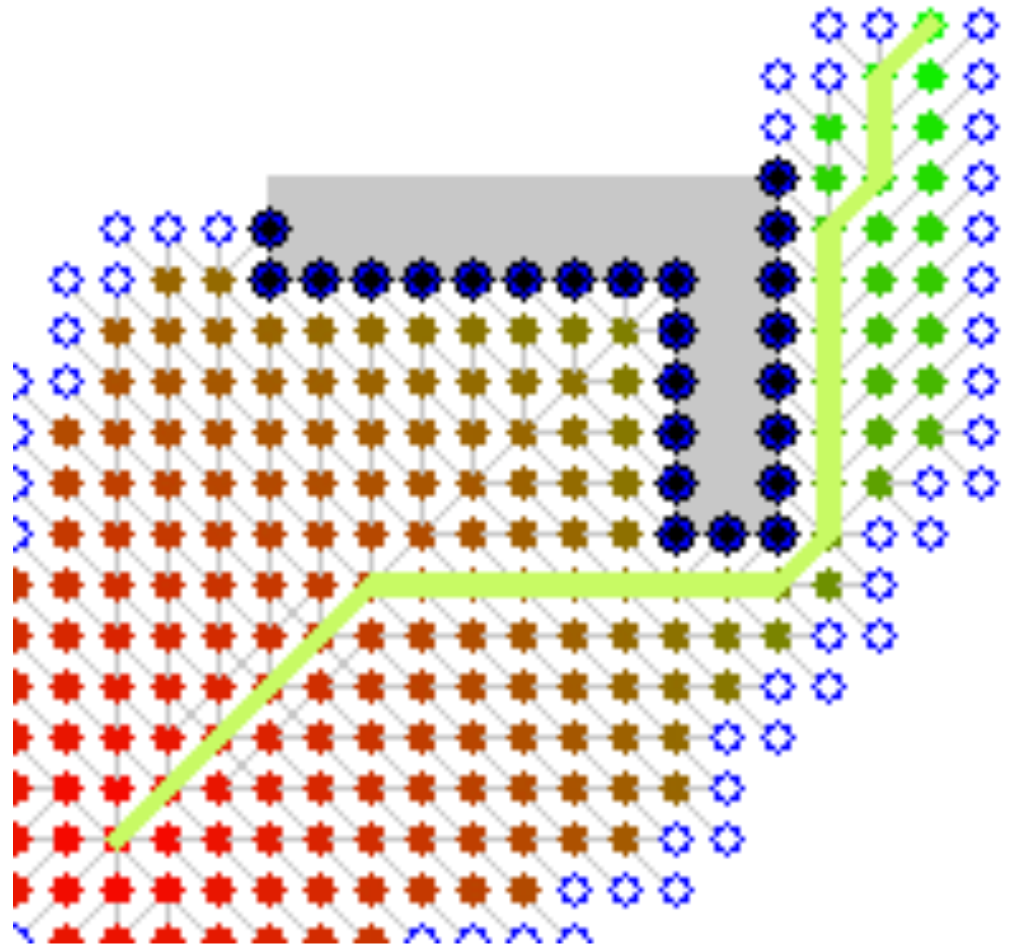
Abhishek Gupta (abhgupta@cs)
Siddhartha Srinivasa (siddh@cs)

TAs:
Carolina Higuera (chiguera@cs)
Rishabh Jain (jrishabh@cs)
Entong Su (ensu@cs)

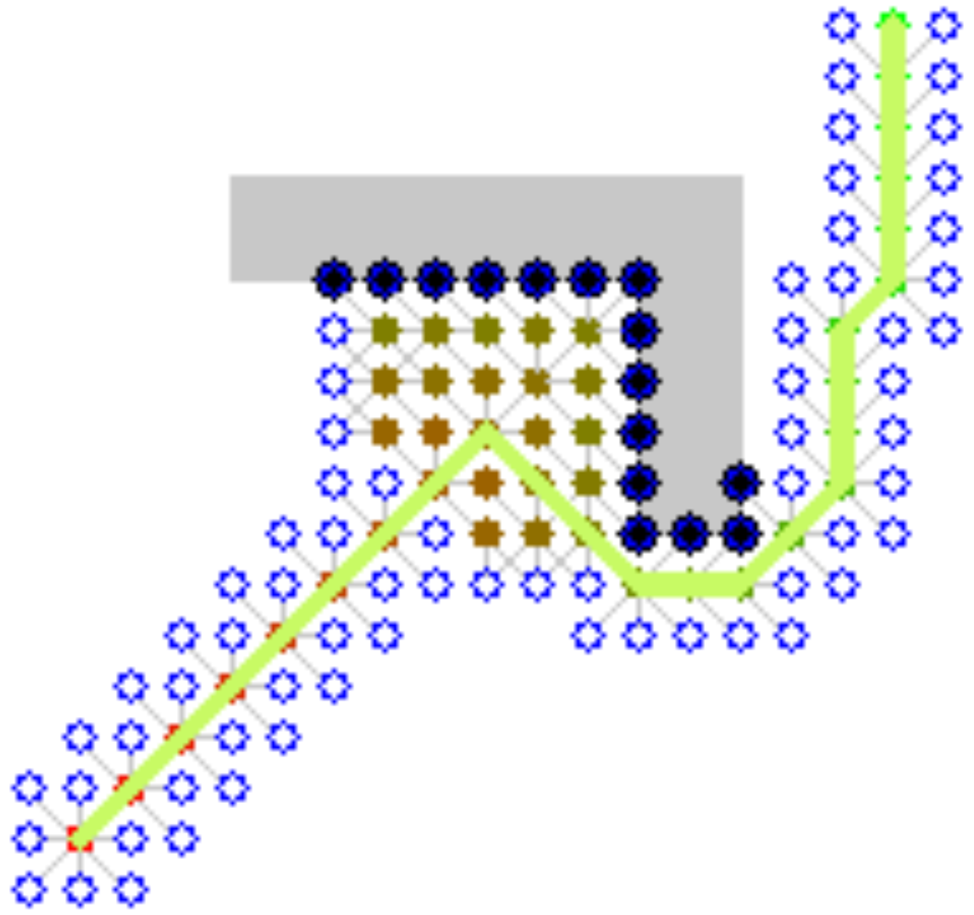
Efficient Heuristic Graph Search



DIJKSTRA



A*

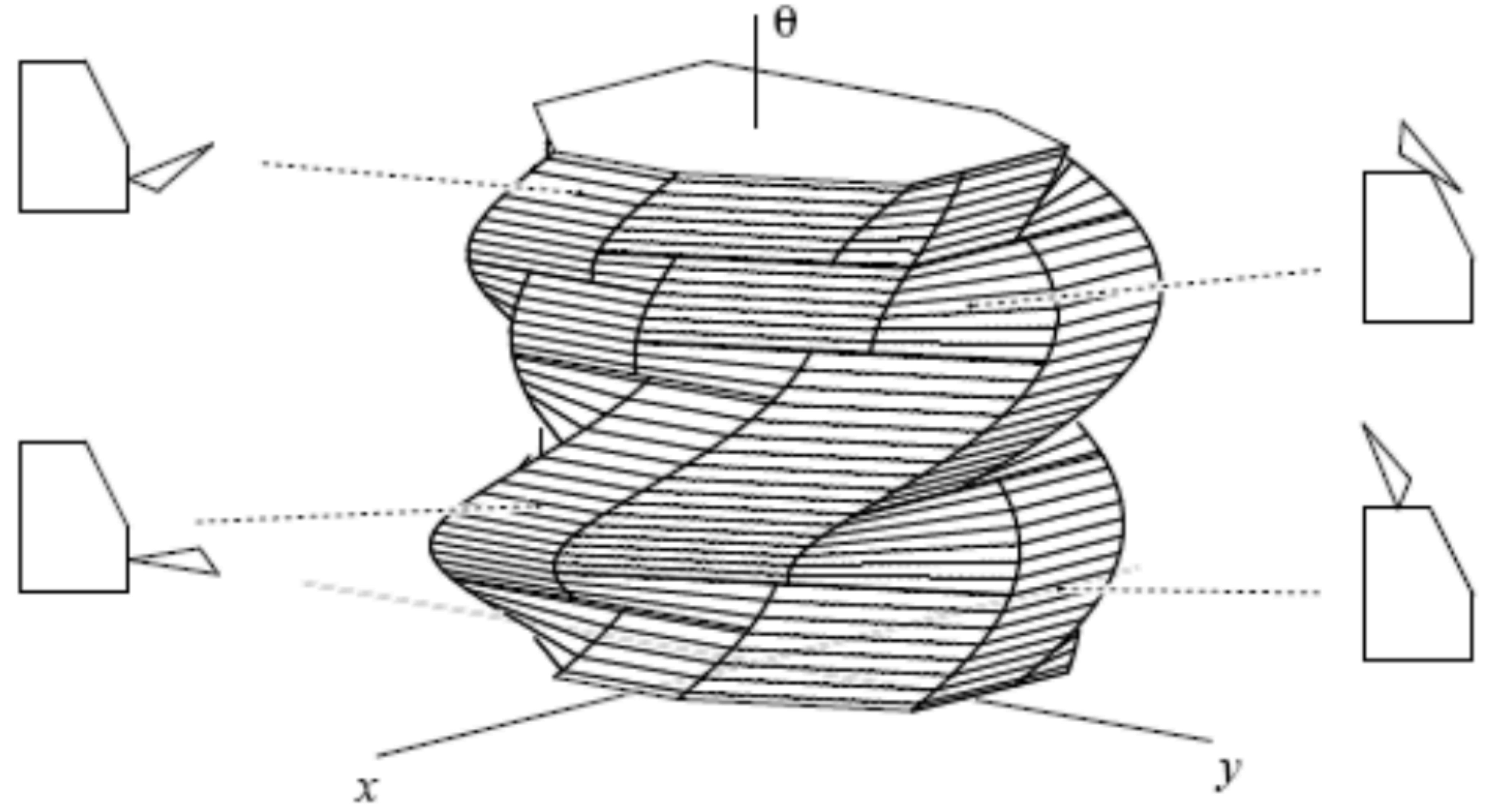


WEIGHTED

(WIKIPEDIA)

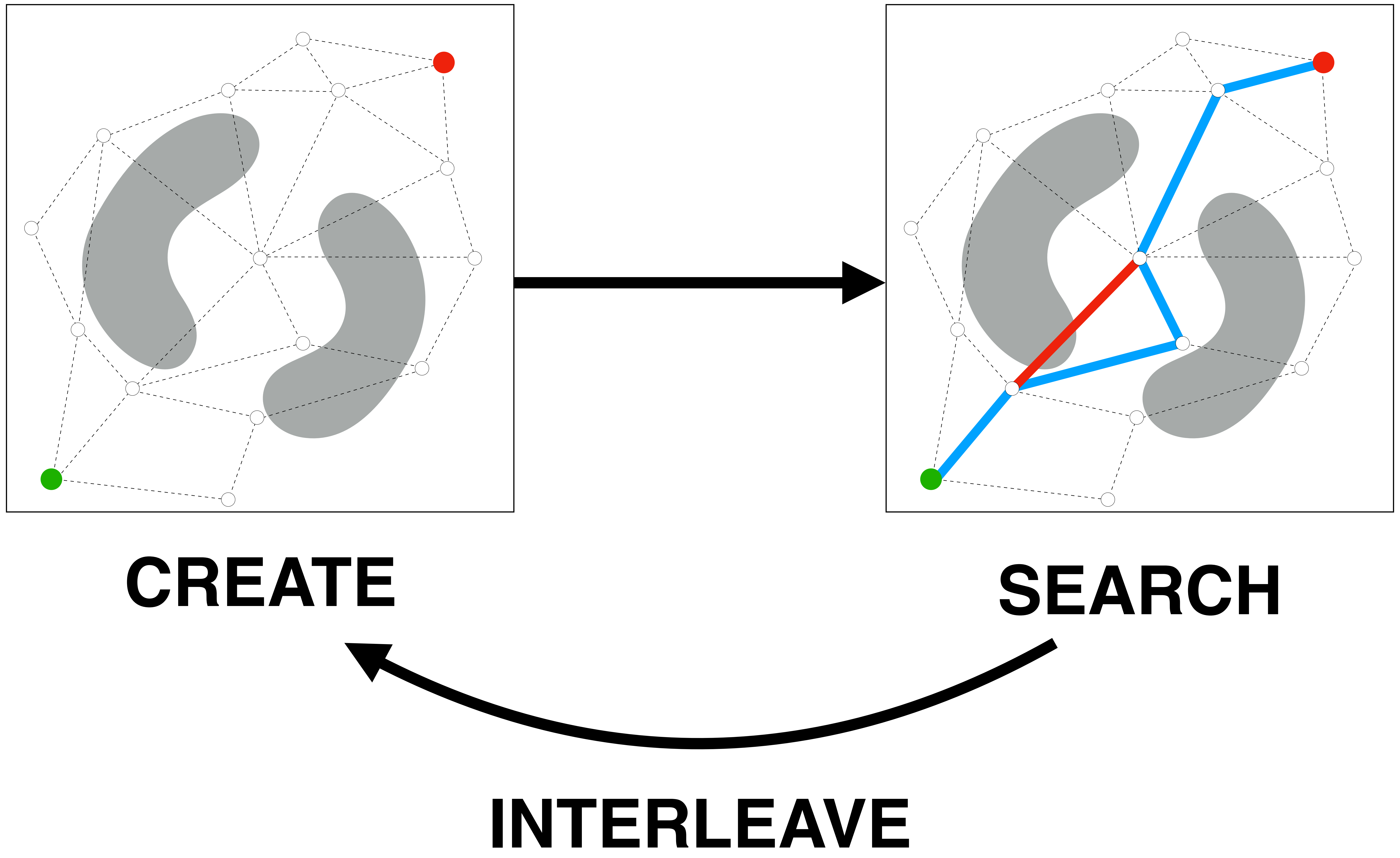
Sampling-Based Motion Planning

- Computing configuration-space obstacles is hard
  - Use a **collision checker** instead!
- Planning in continuous high-dimensional space is hard
  - Construct a **discrete graph approximation** of the continuous space!



(EXAMPLE FROM HOWIE CHOSET)

Sampling-Based Motion Planning



Sampling-Based Motion Planning

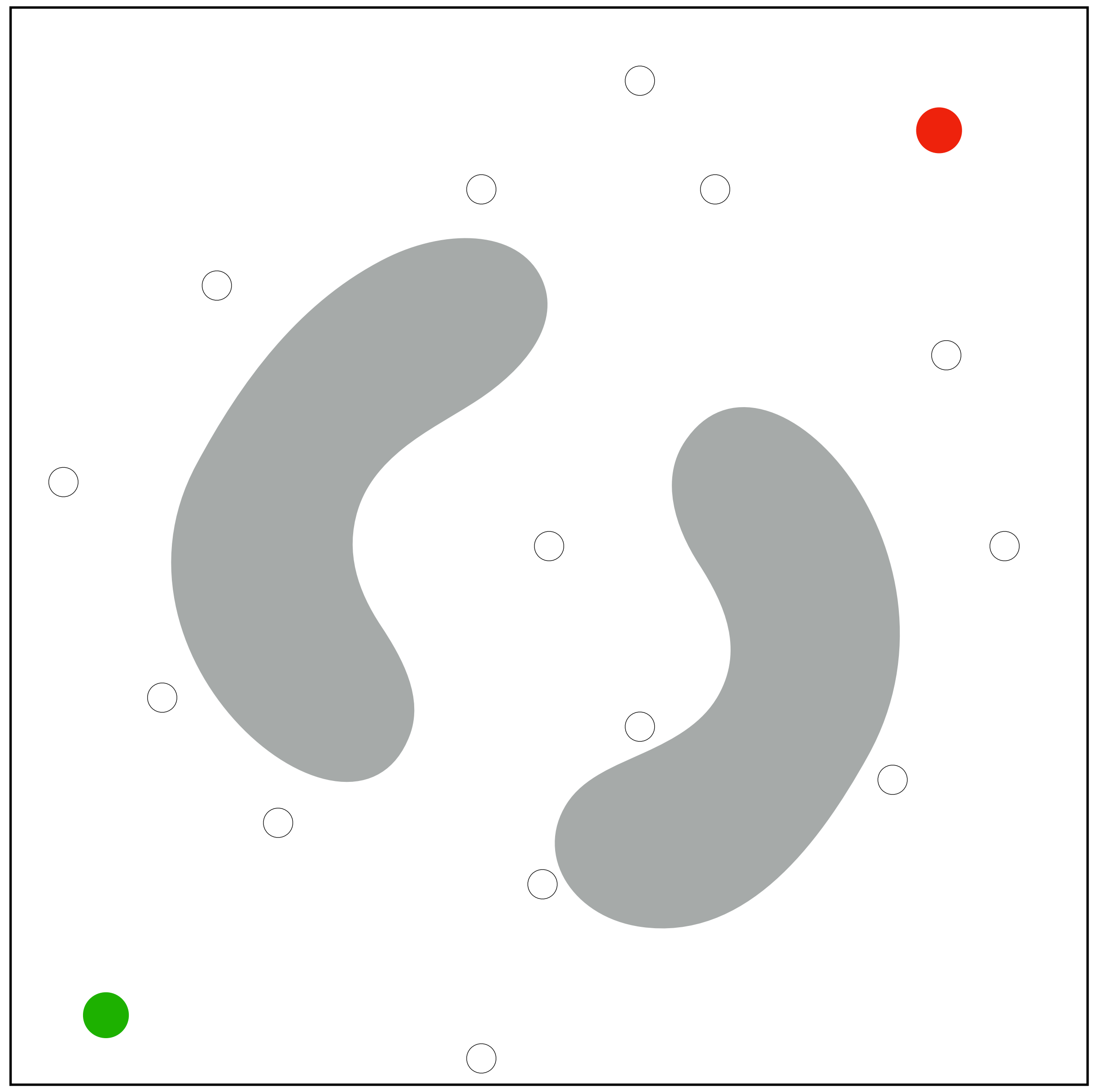
NEW PLANNING ALGORITHM = **FANCY SAMPLER** × **FANCY HEURISTIC** × **FANCY DENSIFICATION**

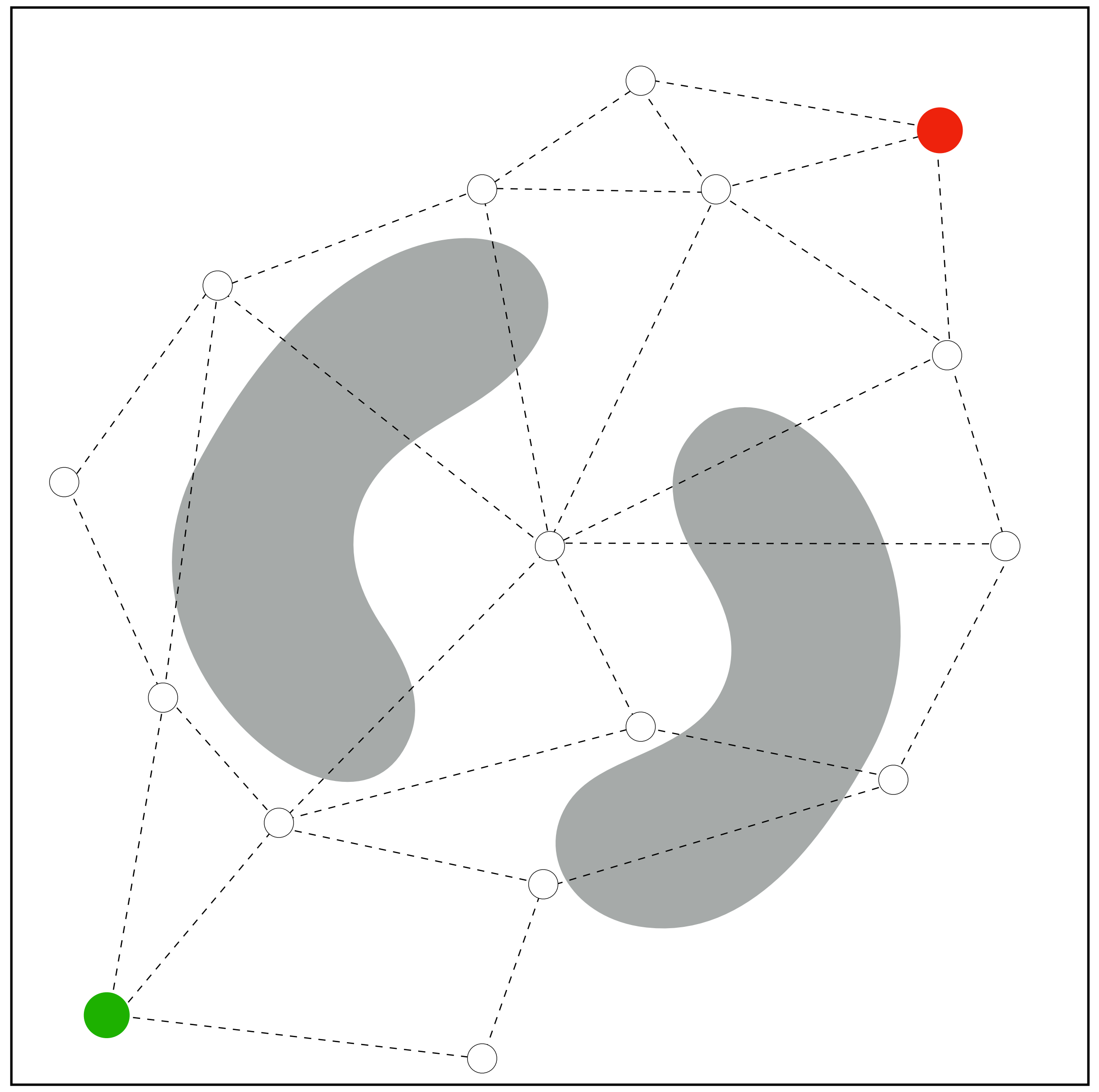
(TODAY) (LAST CLASS) (BONUS)

Creating a Graph

$$G = (V, E)$$

1. Sample collision-free configurations as vertices (including start and goal)
2. Connect neighboring vertices with simple movements as edges





API for motion planning

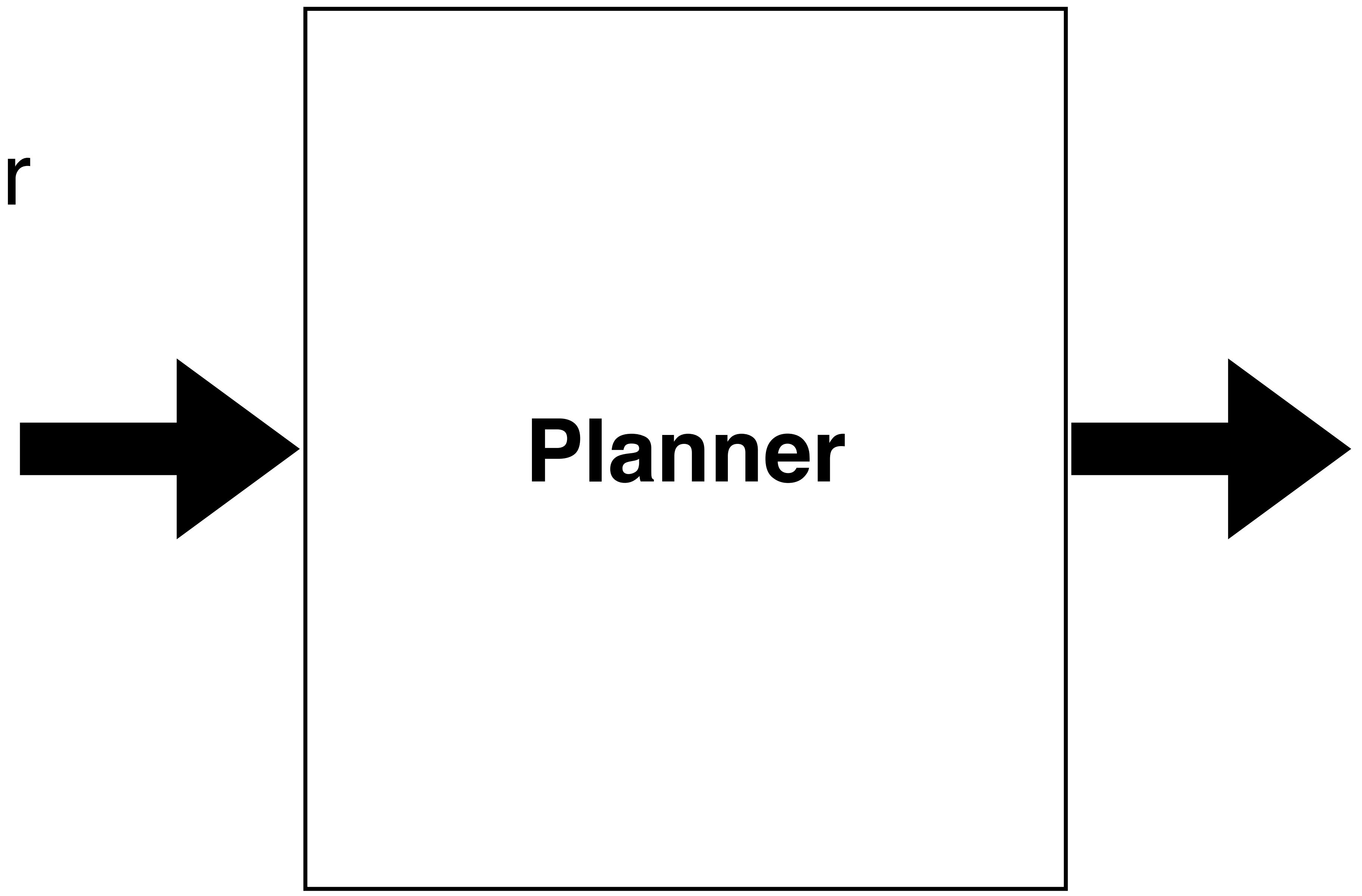
Input

1. A collision checker

$\texttt{coll}(q)$

2. Steering method

$\texttt{steer}(q_1, q_2)$



Output

Collision
free path
joining
start and goal

Let’s take a look at the inputs

We need to give the planner a collision checker

$$\texttt{coll}(q) = \begin{cases} 0 & \text{in collision, i.e. } q \in \mathcal{C}_{obs} \\ 1 & \text{free, i.e. } q \in \mathcal{C}_{free} \end{cases}$$

What work does this function have to do?

Collision checking is expensive!

Let’s take a look at the inputs

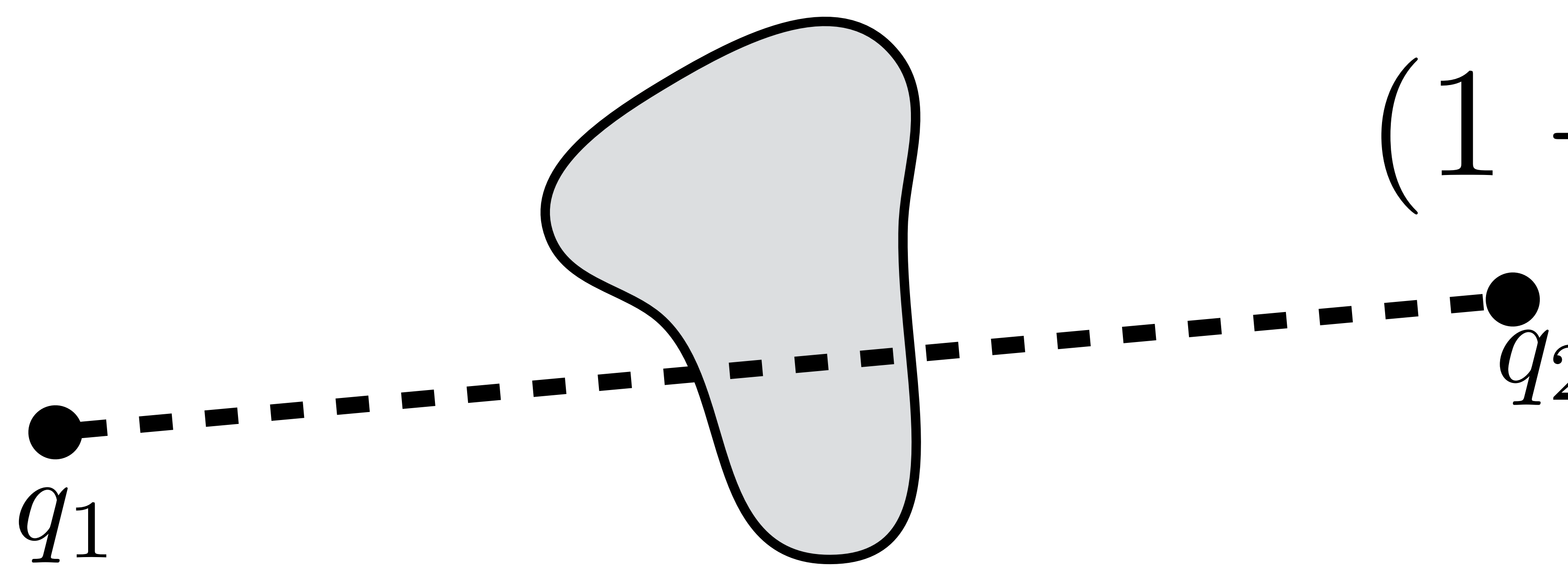
We need to give the planner a steer function

$$\texttt{steer}(q_1, q_2)$$

A steer function tries to join two configurations with a feasible path

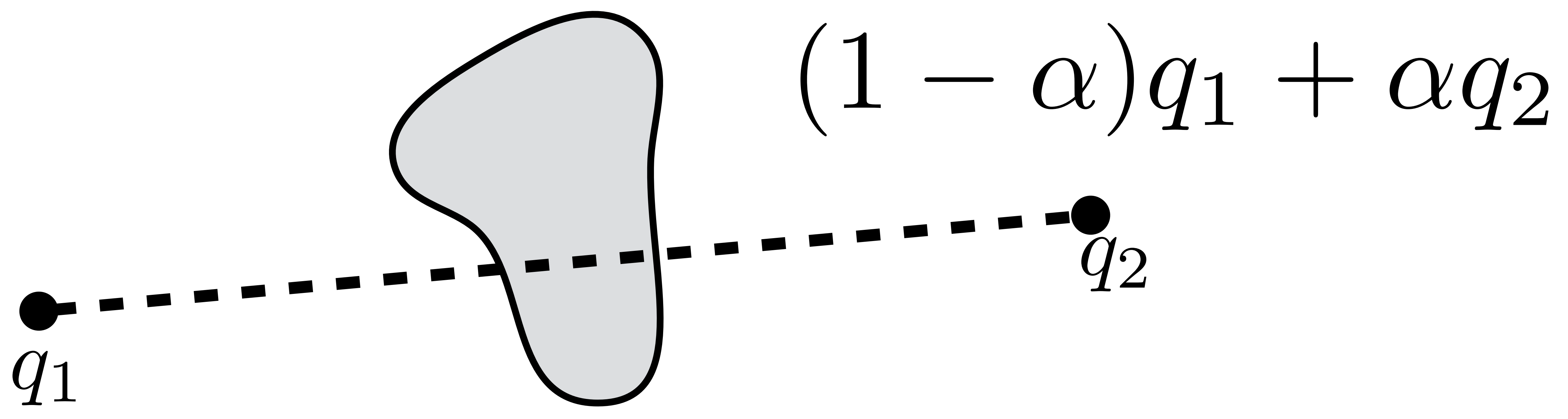
Computes simple path, calls coll(q), and returns success if path is free

$$(1-\alpha)q_1 + \alpha q_2$$



Example: Connect them with a straight line and check for feasibility

Can steer be smart about collision checking?



$\texttt{steer}(q_1, q_2)$ has to assure us line is collision free (upto a resolution)

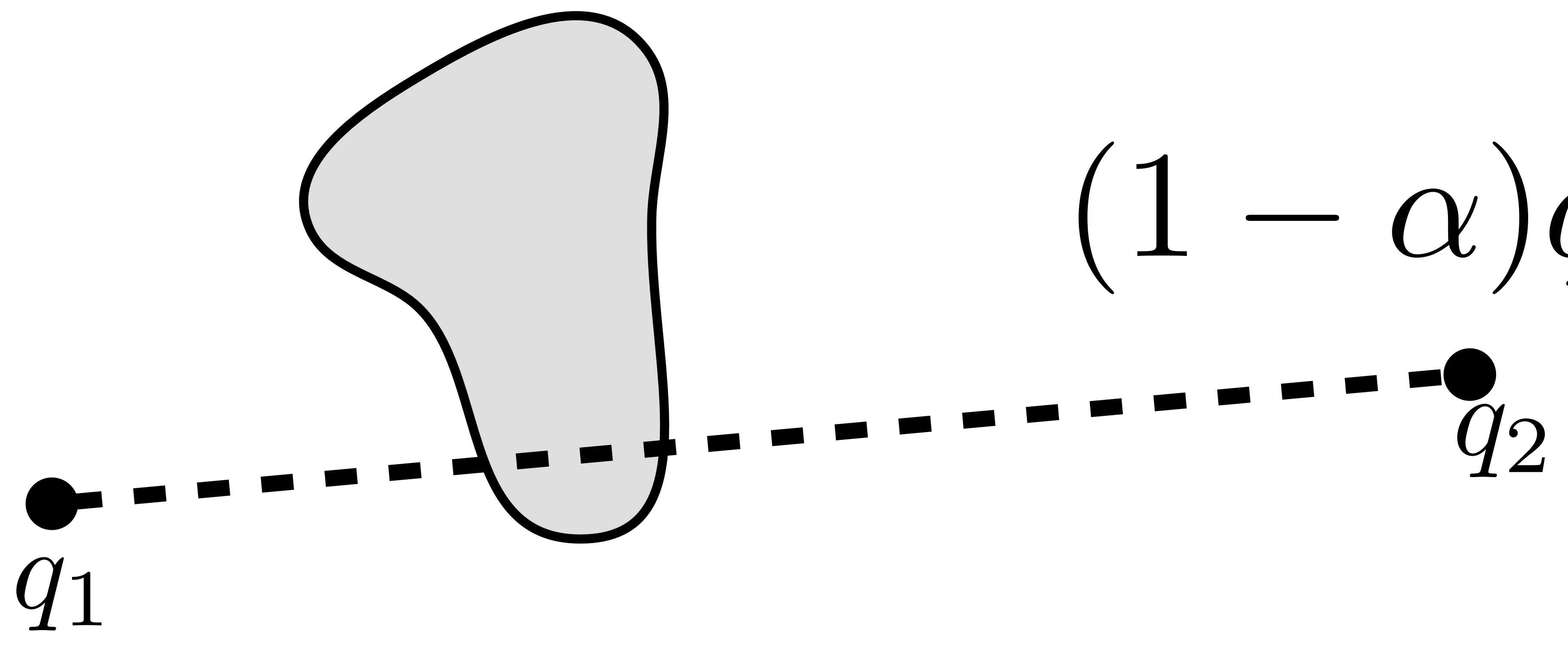
Things we can try:

1. Step forward along the line and check each point
2. Step backwards along the line and check each point

…….

Can steer be smart about collision checking?

Say we chunk the line into 16 parts



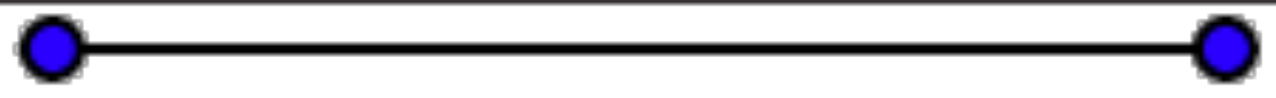
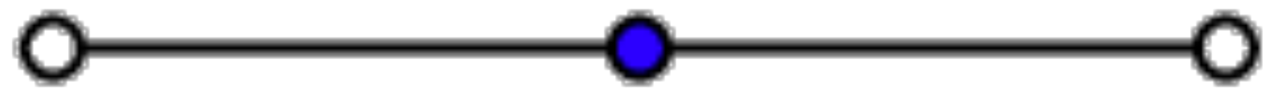
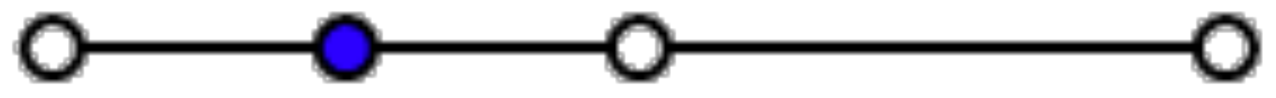
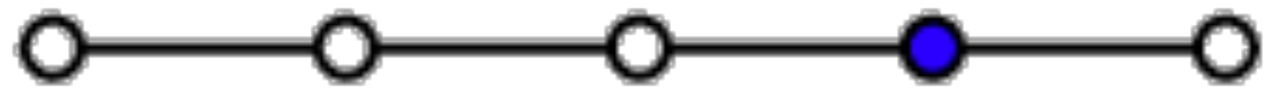
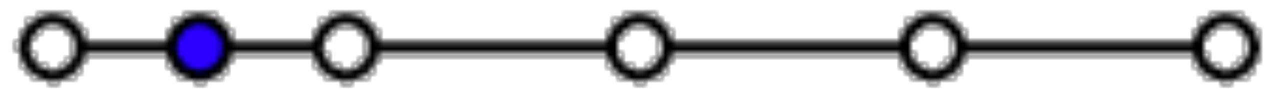
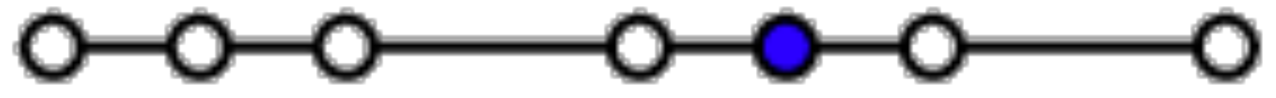
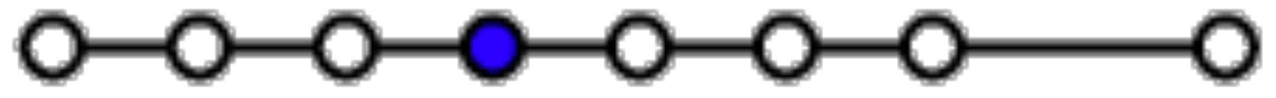
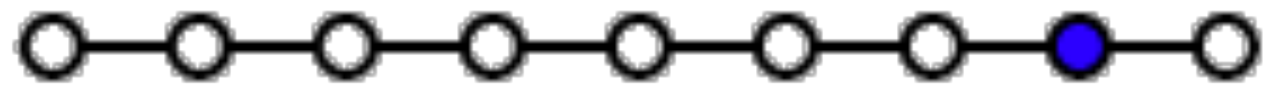
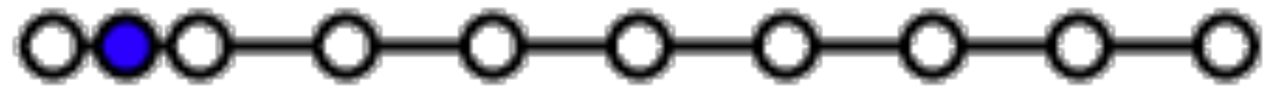
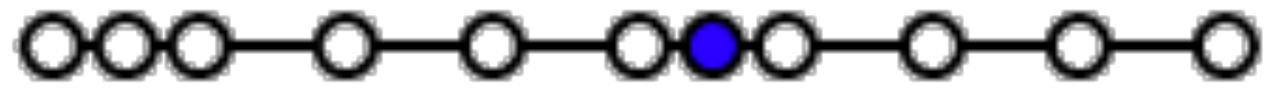
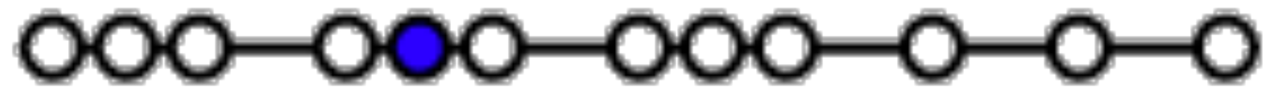
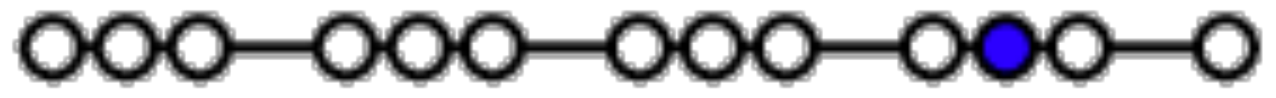
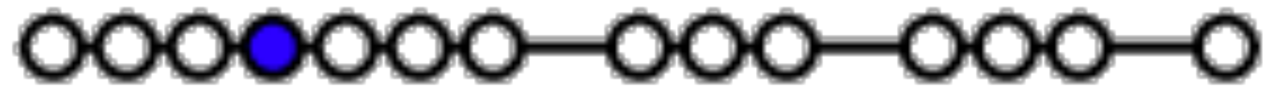
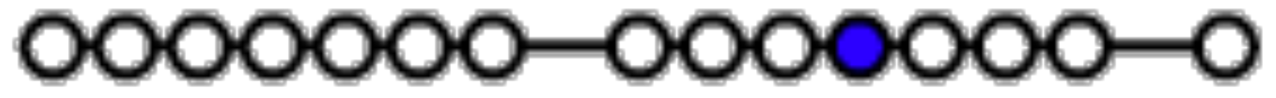
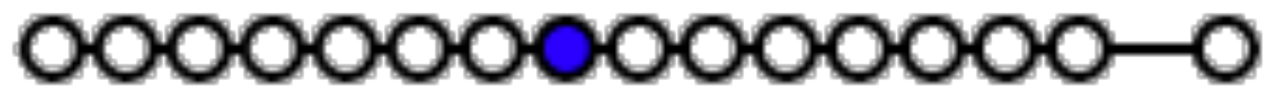
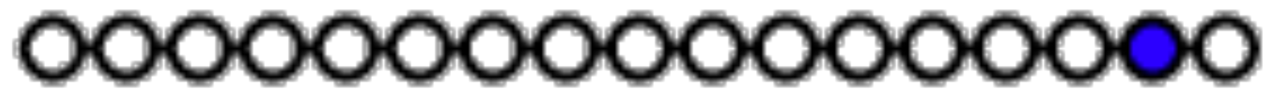
$$(1-\alpha)q_1 + \alpha q_2$$

Any collision checking strategy corresponds to sequence

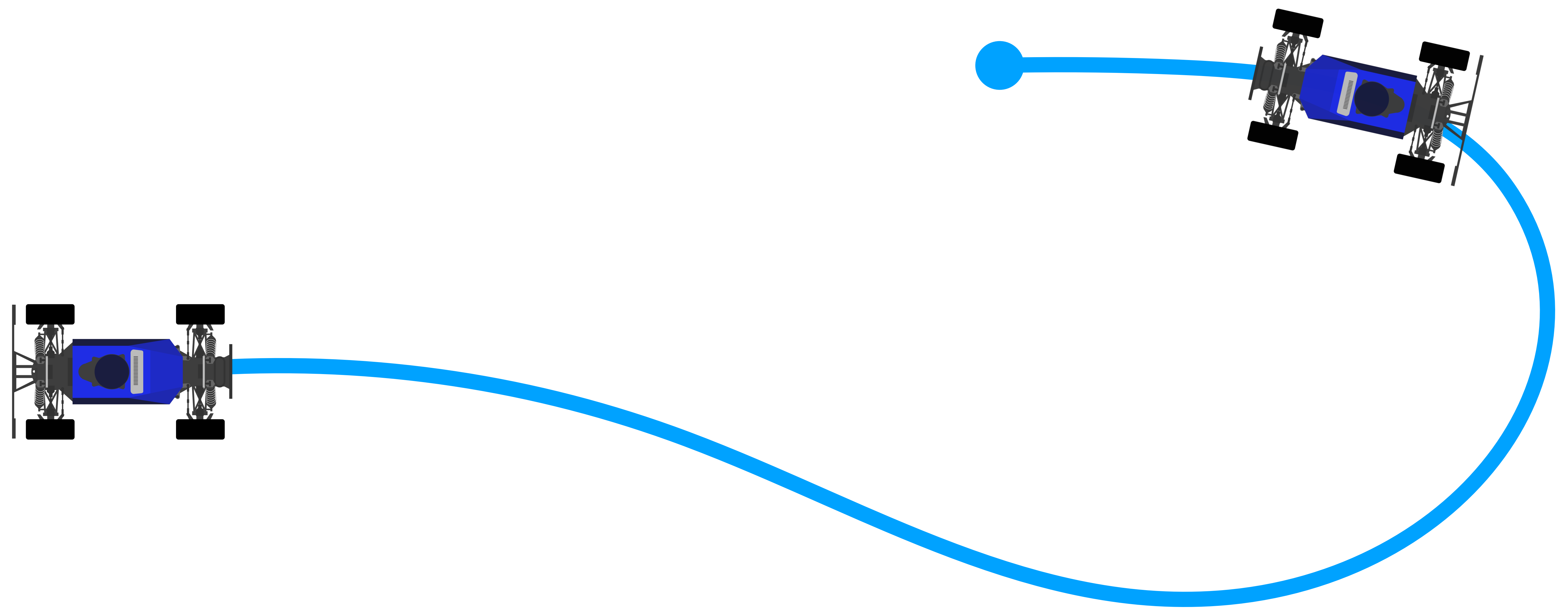
(Naive) $\alpha = 0, \frac{1}{16}, \frac{2}{16}, \frac{3}{16}, \cdots, \frac{15}{16}$

(Bisection) $\alpha = 0, \frac{8}{16}, \frac{4}{16}, \frac{12}{16}, \cdots, \frac{15}{16}$

Ans: Van der Corput sequence

i	Naive Sequence	Binary	Reverse Binary	Van der Corput	Points in $[0,1]/\sim$
1	0	.0000	.0000	0	
2	1/16	.0001	.1000	1/2	
3	1/8	.0010	.0100	1/4	
4	3/16	.0011	.1100	3/4	
5	1/4	.0100	.0010	1/8	
6	5/16	.0101	.1010	5/8	
7	3/8	.0110	.0110	3/8	
8	7/16	.0111	.1110	7/8	
9	1/2	.1000	.0001	1/16	
10	9/16	.1001	.1001	9/16	
11	5/8	.1010	.0101	5/16	
12	11/16	.1011	.1101	13/16	
13	3/4	.1100	.0011	3/16	
14	13/16	.1101	.1011	11/16	
15	7/8	.1110	.0111	7/16	
16	15/16	.1111	.1111	15/16	

Boundary Value Problem

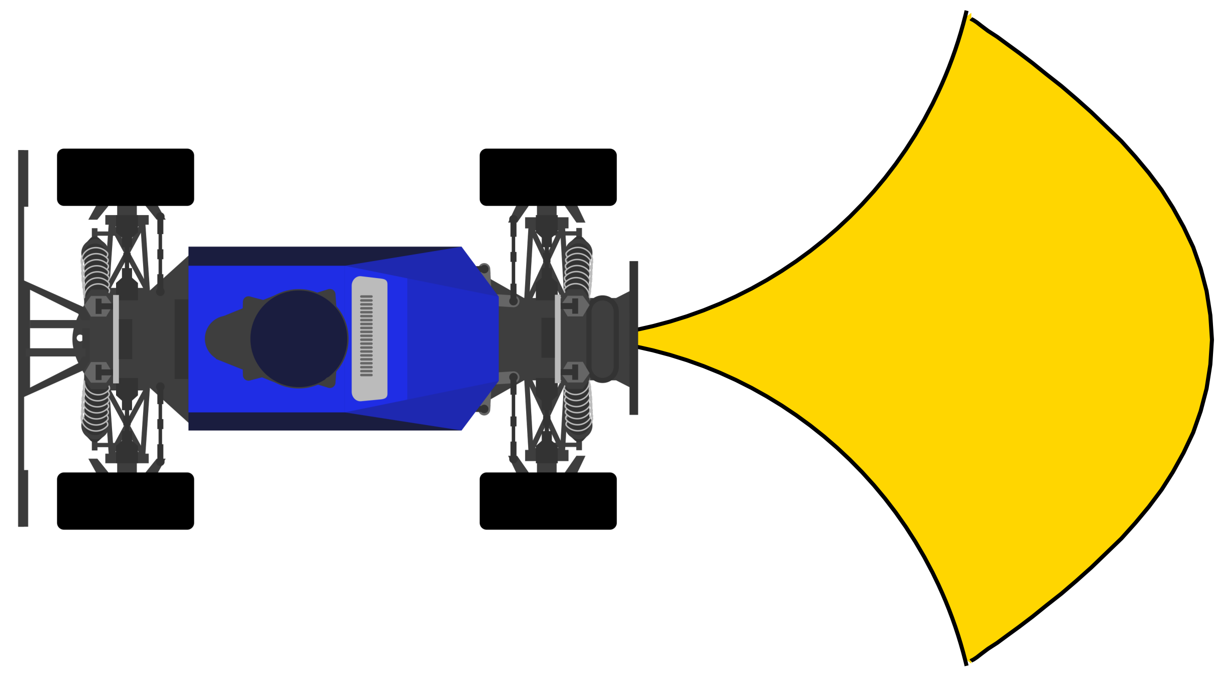


- How can we move from one configuration to another? Hard in general!
- Define a steering function that is tasked with connecting two configurations
- Previously, steering function was trivial (straight line)

Differential Constraints on Graphs

- When expanding a vertex from the priority queue
  - Call successor function to compute neighboring vertices
  - **Solve boundary value problem to compute valid edges and costs**
  - Add neighboring vertices to priority queue
- Otherwise, algorithm remains the same! Can also be lazy

Solving the Boundary Value Problem



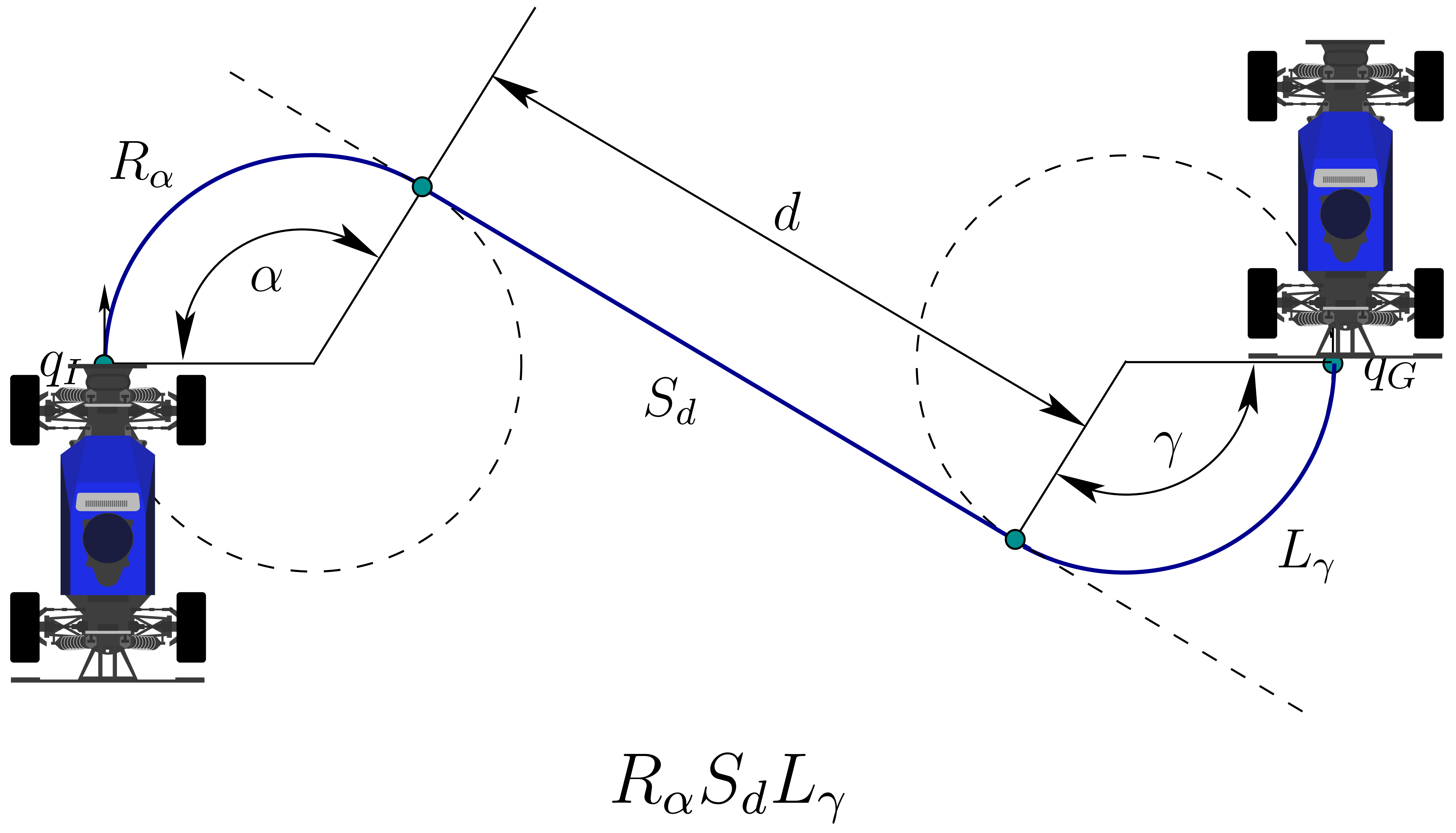
$$q_1 = (x_1, y_1, \theta_1)$$

$$q_2 = (x_2, y_2, \theta_2)$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \frac{v\tan\delta}{L} \end{bmatrix}$$

$$0 \le v \le v_{\max}, |\delta| \le \delta_{\max}$$

Dubins Curves



RIGHT-STRAIGHT-LEFT

- Dubins showed that all solutions had to be one of six classes

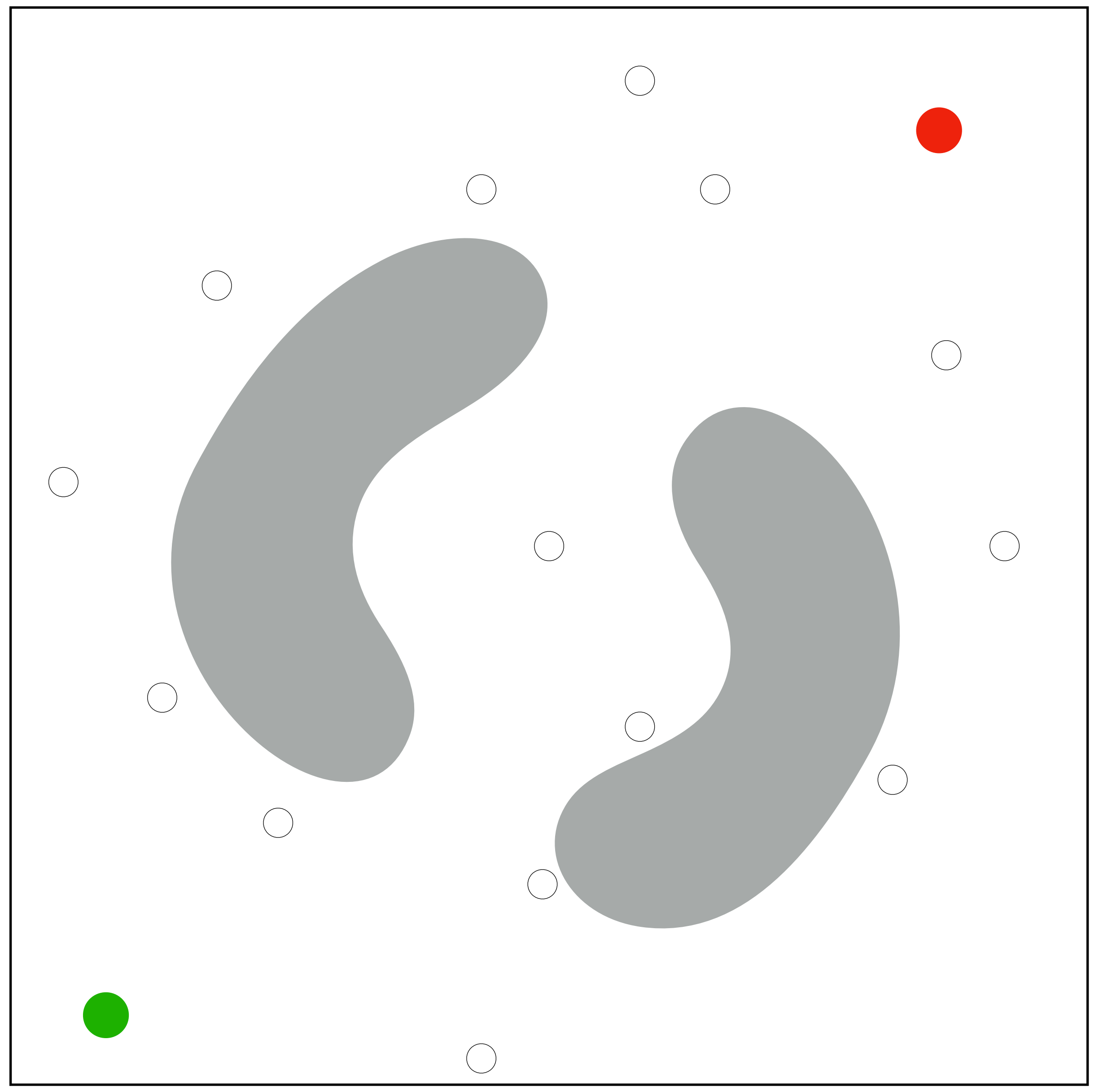
$$\{LRL, RLR, LSL, LSR, RSL, RSR\}$$

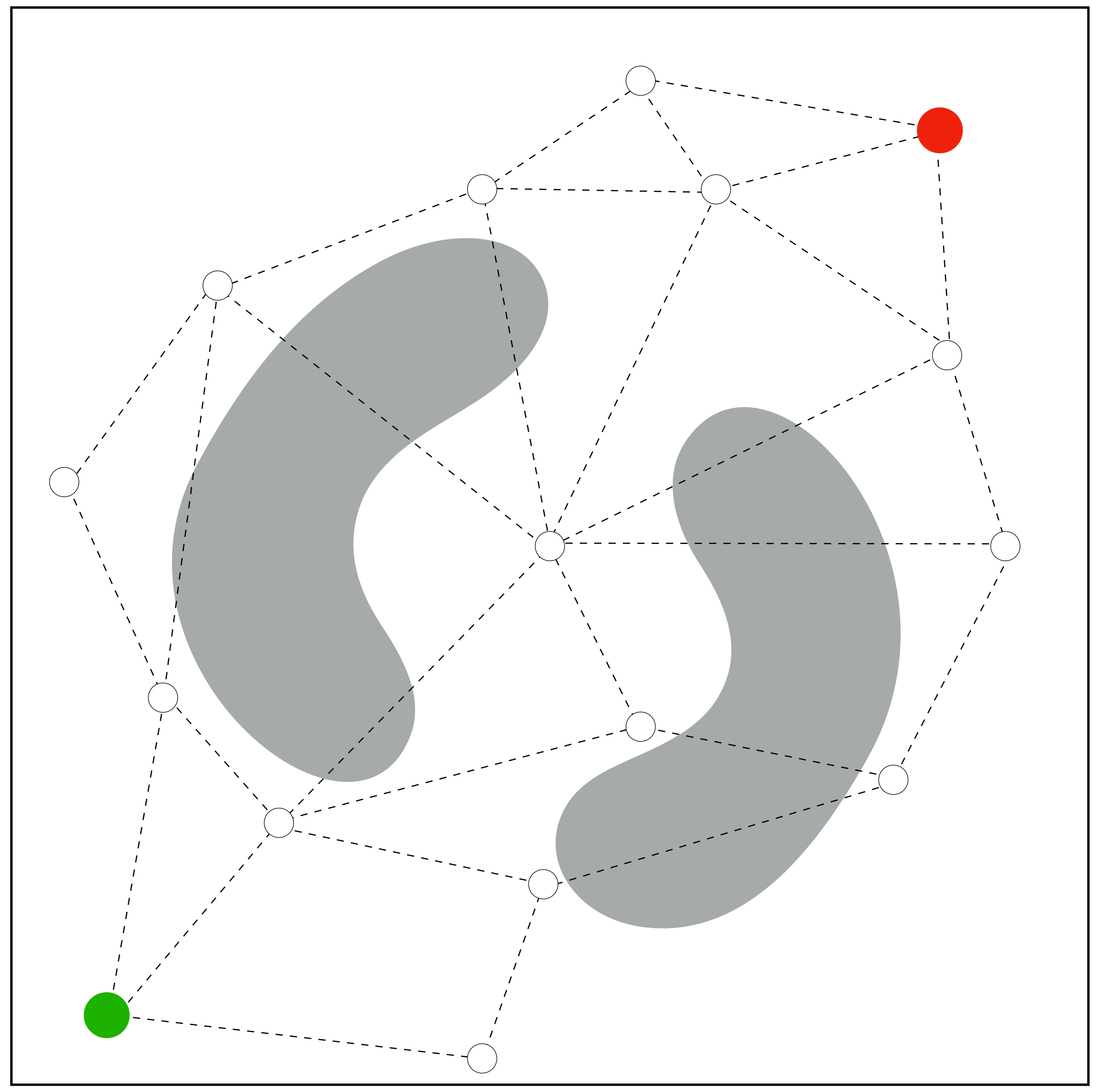
- Given two configurations to connect, evaluate all six options, return shortest one
- Car has fixed forward velocity; Reeds-Shepp curves may include backward velocity

Creating a Graph

$$G = (V, E)$$

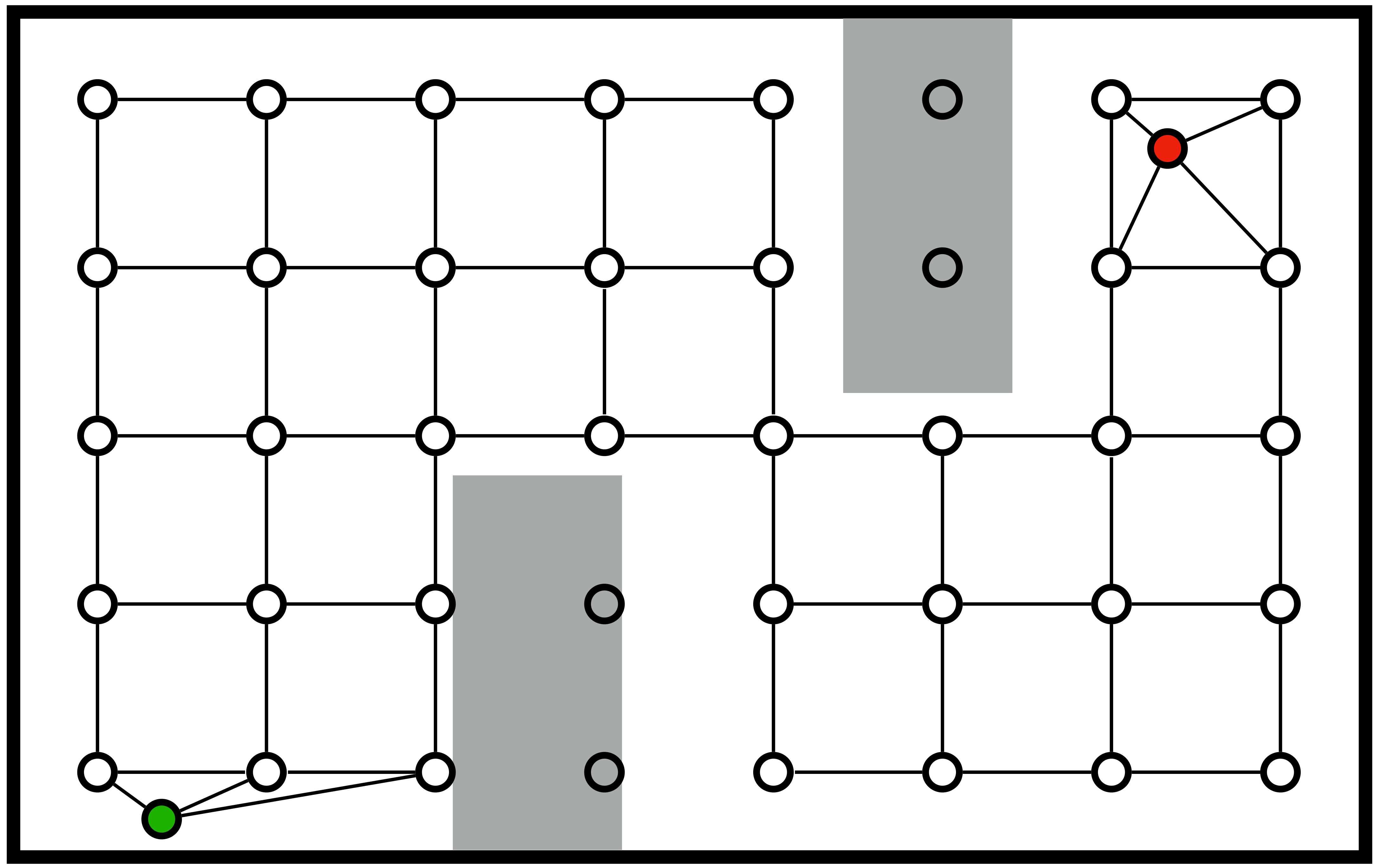
1. Sample collision-free configurations as vertices (including start and goal)
2. Connect neighboring vertices with simple movements as edges





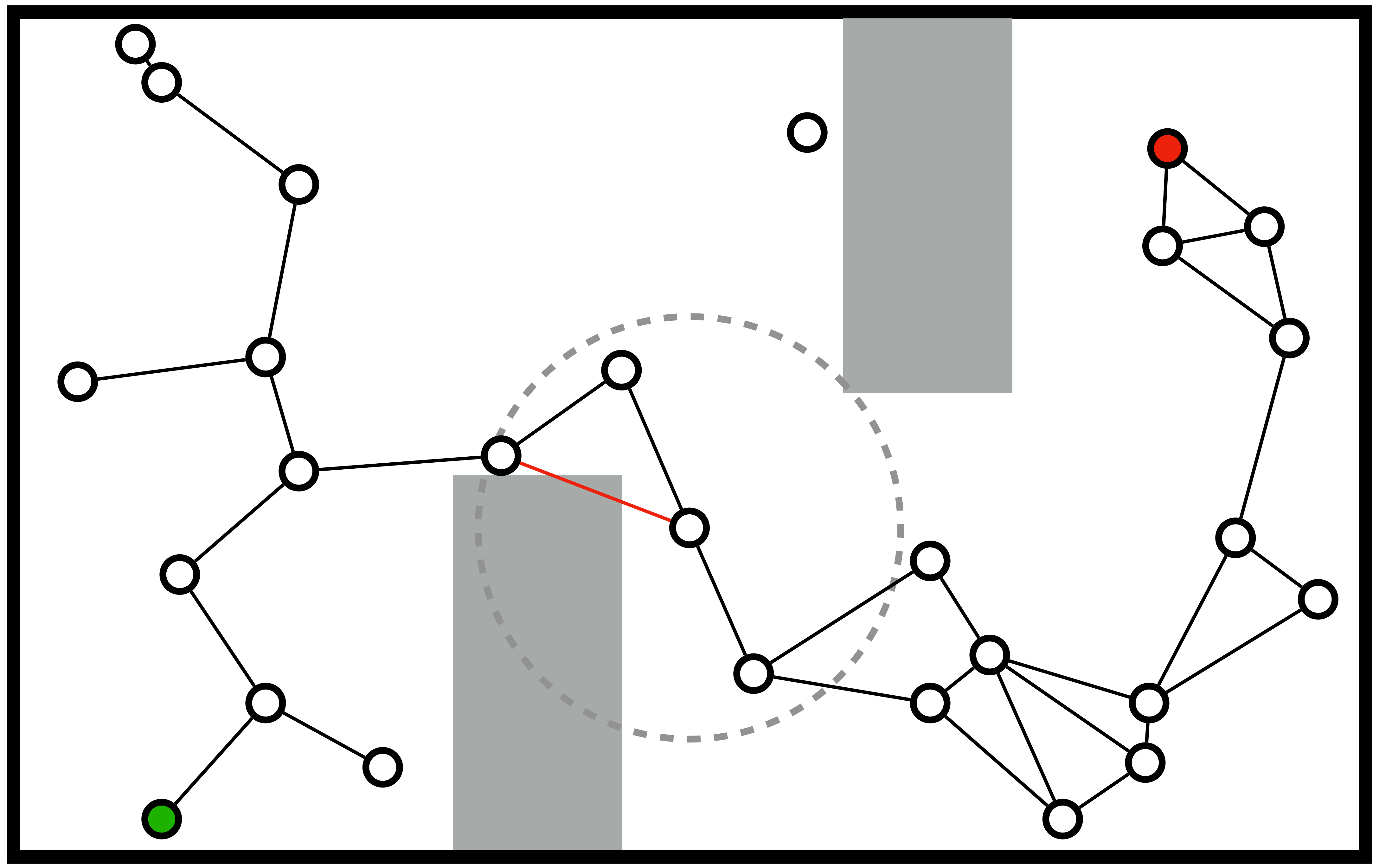
Strategy 1: Lattice Sampling

- Main idea: create a grid, and connect neighboring points (4-conn, 8-conn, …)



Strategy 2: Uniform Random Sampling

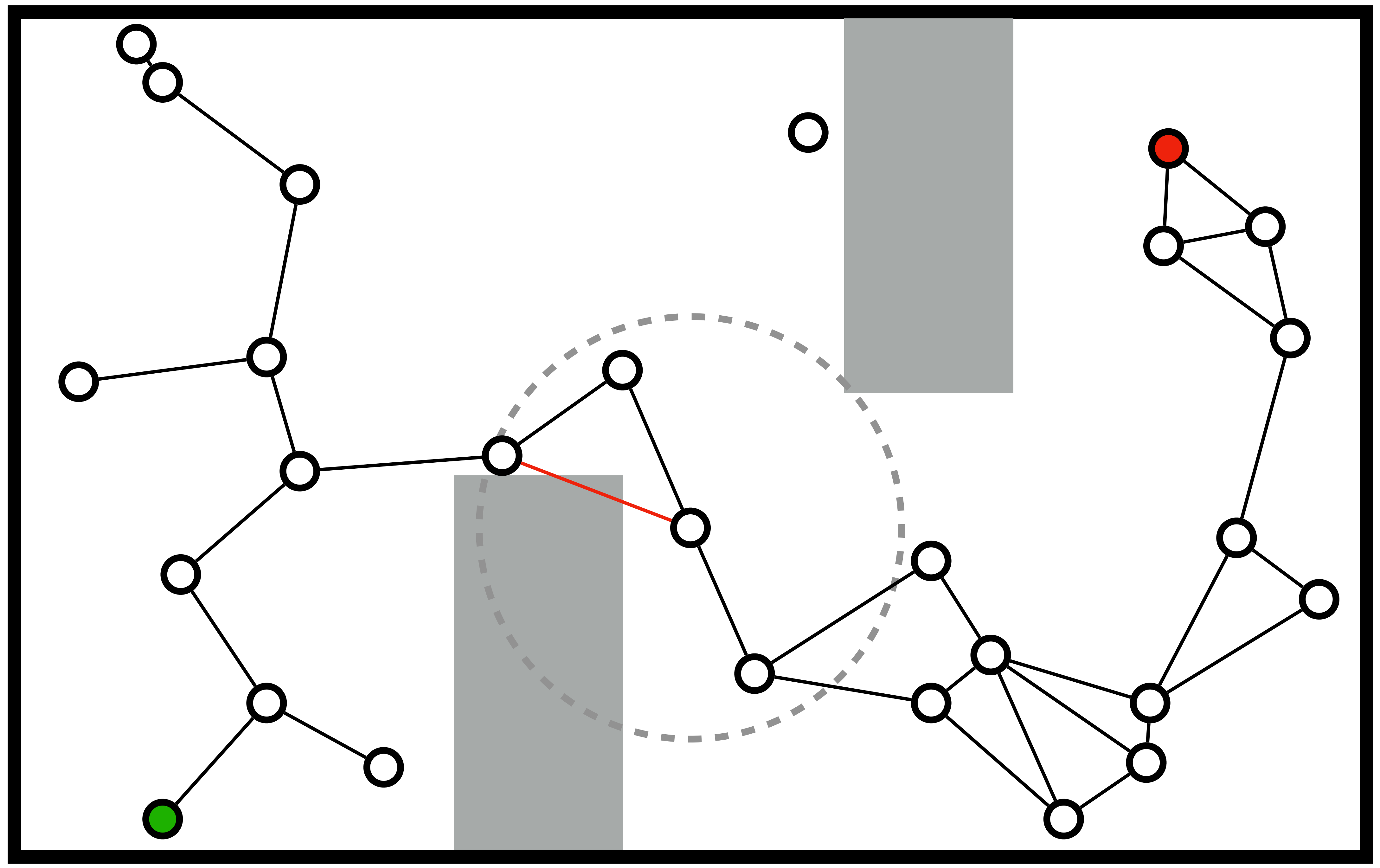
- Main idea: sample uniformly between each dimension's lower/upper bounds
- Connect vertices within radius (r-disc) or k nearest neighbors



KAVRAKI ET AL., 1996

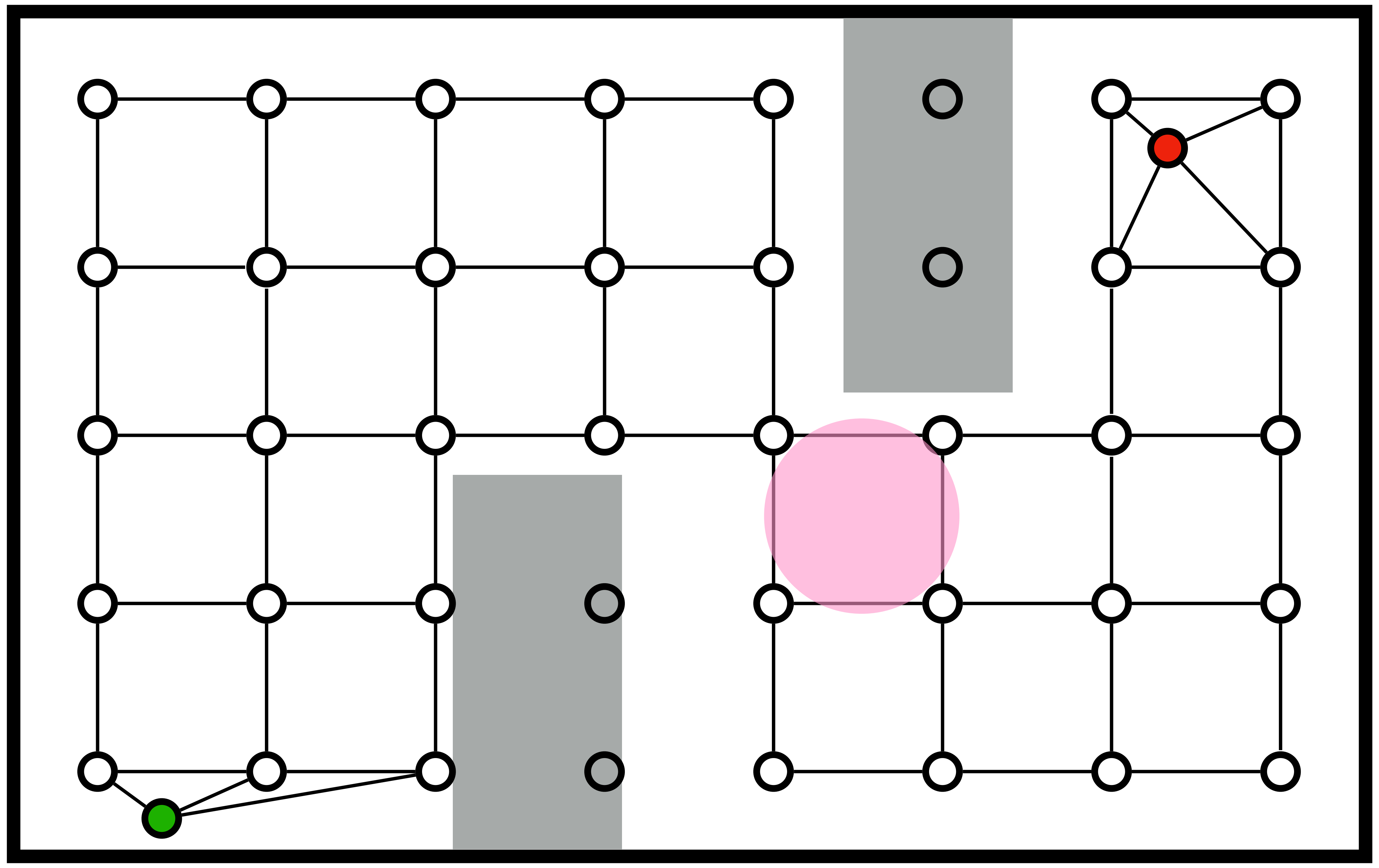
Probabilistic Roadmap (PRM)

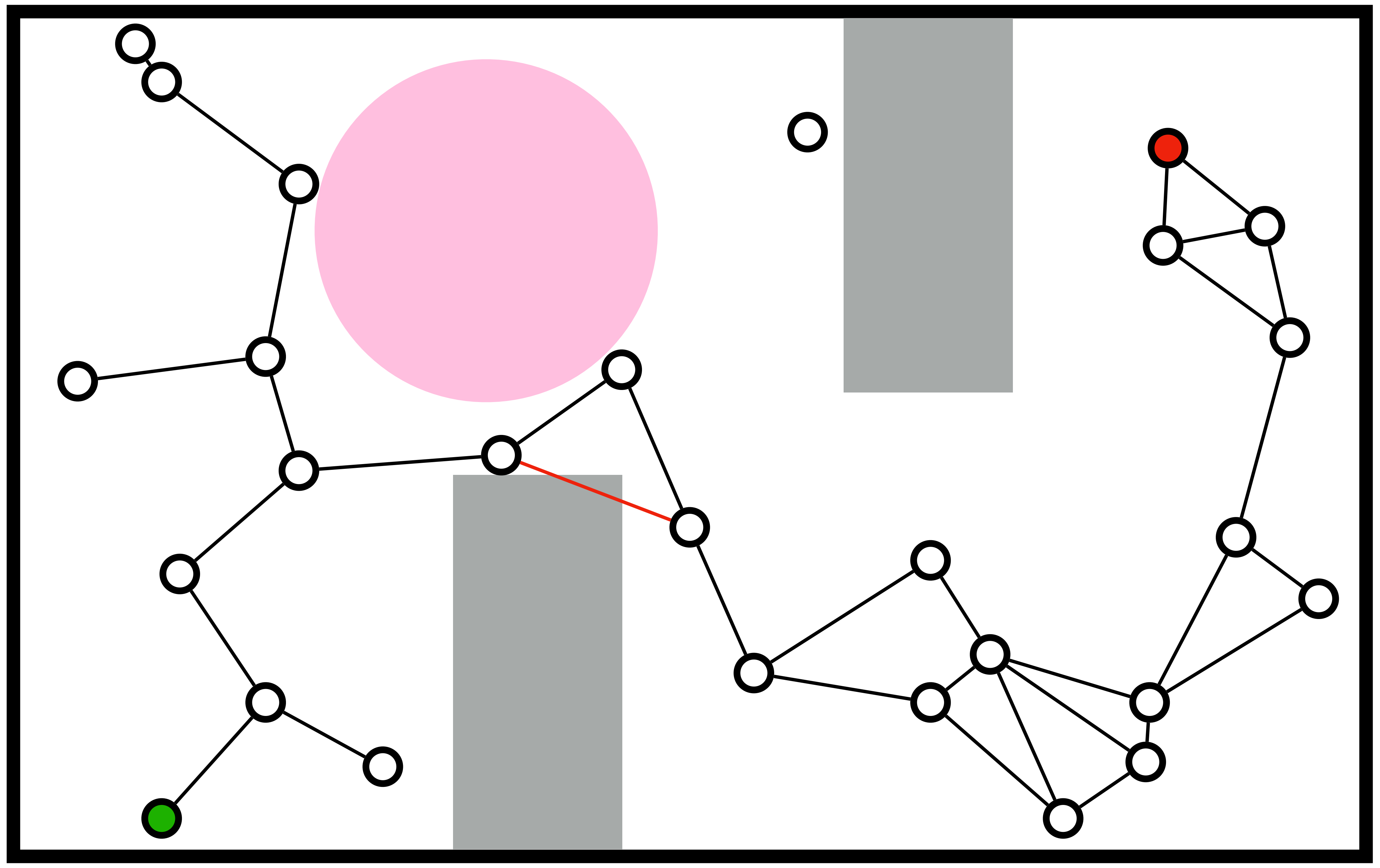
- When should we collision-check edges?
- What is the optimal radius? (PRM with optimal radius = PRM*)



KAVRAKI ET AL., 1996

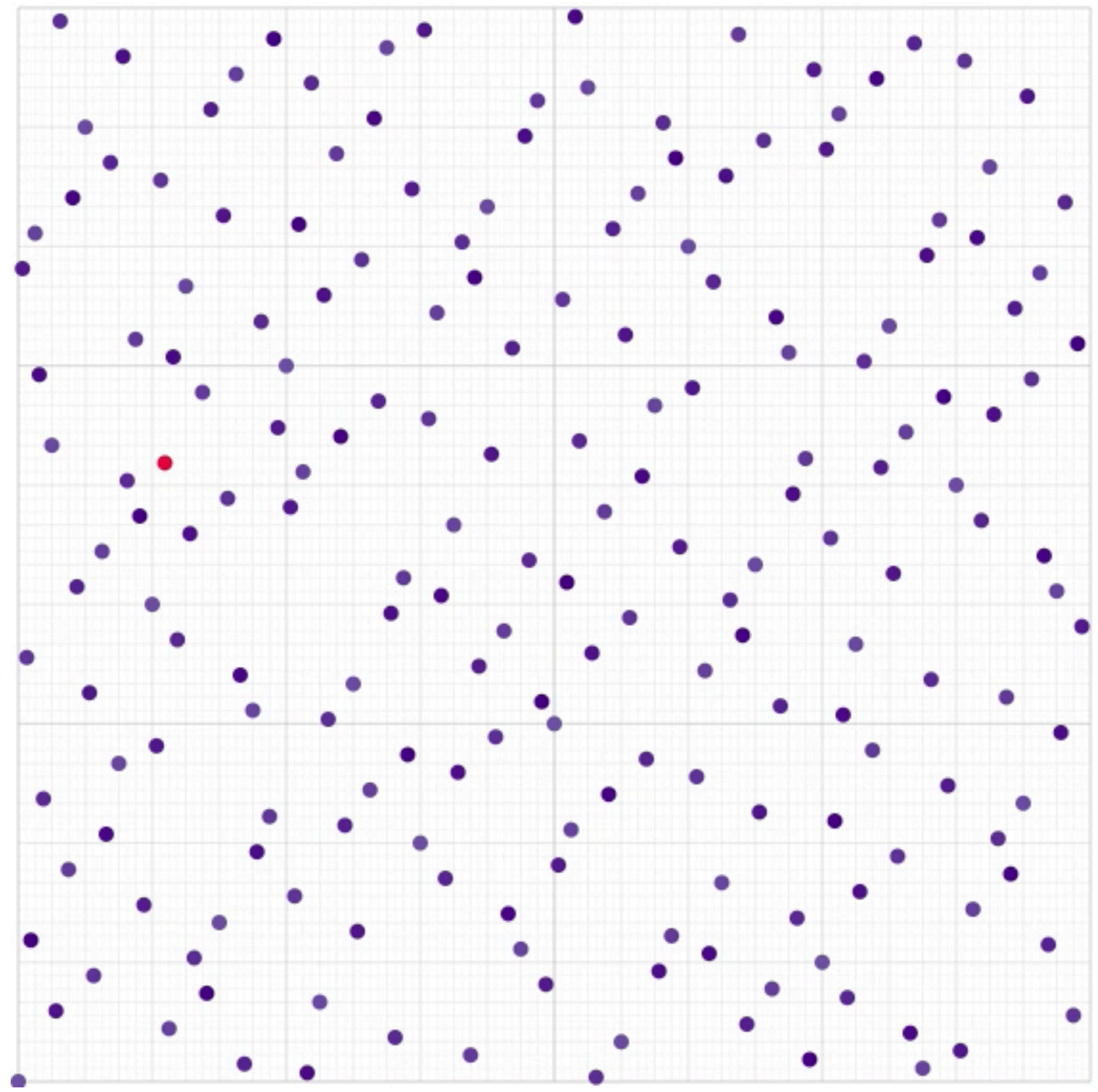
Alternatives to Random Sampling





Strategy 3: Low-Dispersion Sampling

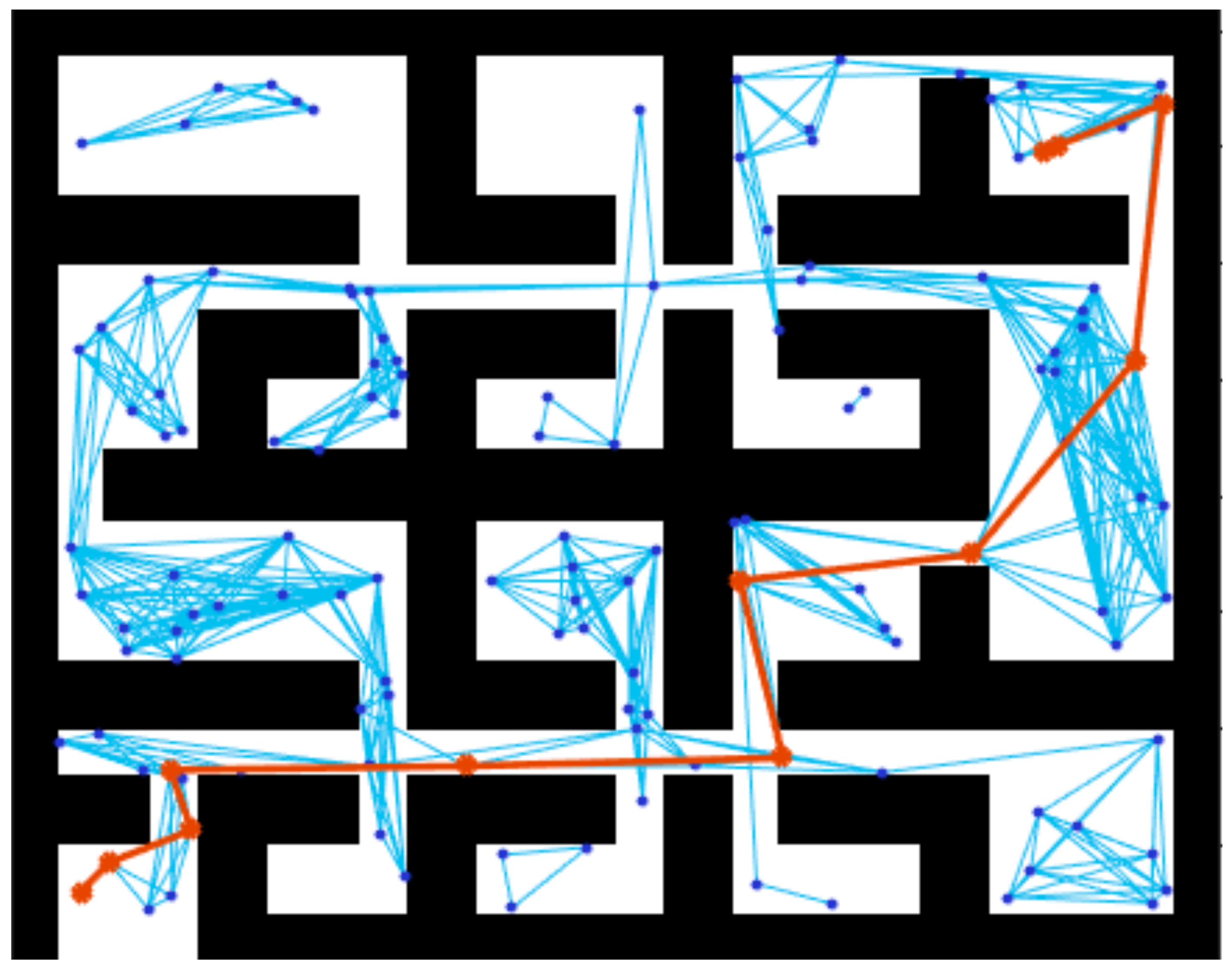
- Main idea: Halton sequence uniformly densifies the space



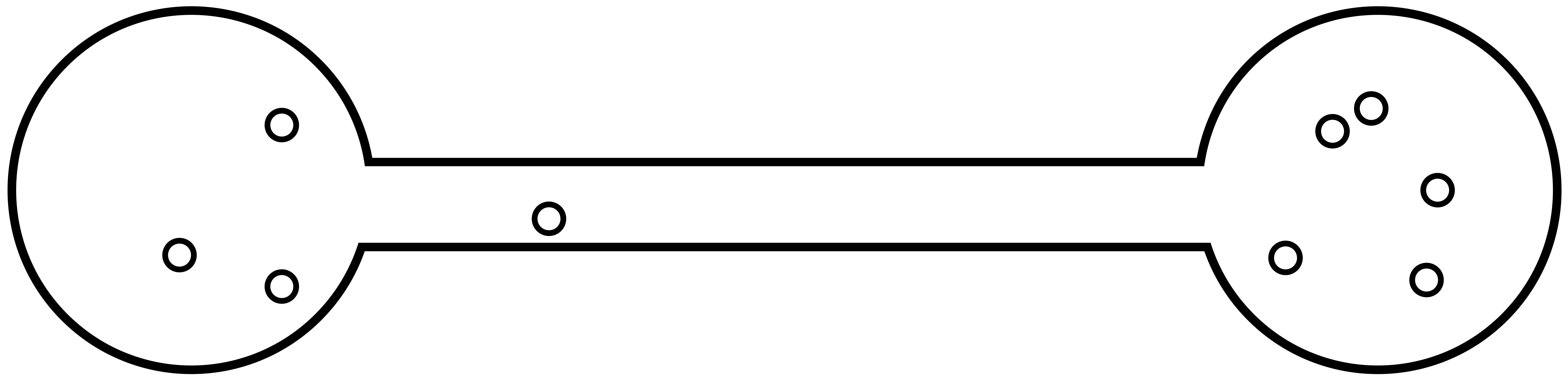
HTTPS://OBSERVABLEHQ.COM/@JRUS/HALTON

What Graphs Are Good?

- A good graph must be **sparse** (both in vertices and edges)
- A good graph must have **good free-space coverage**
  - For every configuration in the free space, there's a vertex in the graph that can be connected to it.
- A good graph must have **good free-space connectivity**
  - For every connected pair of points in the free space, there's a path on the graph between them.



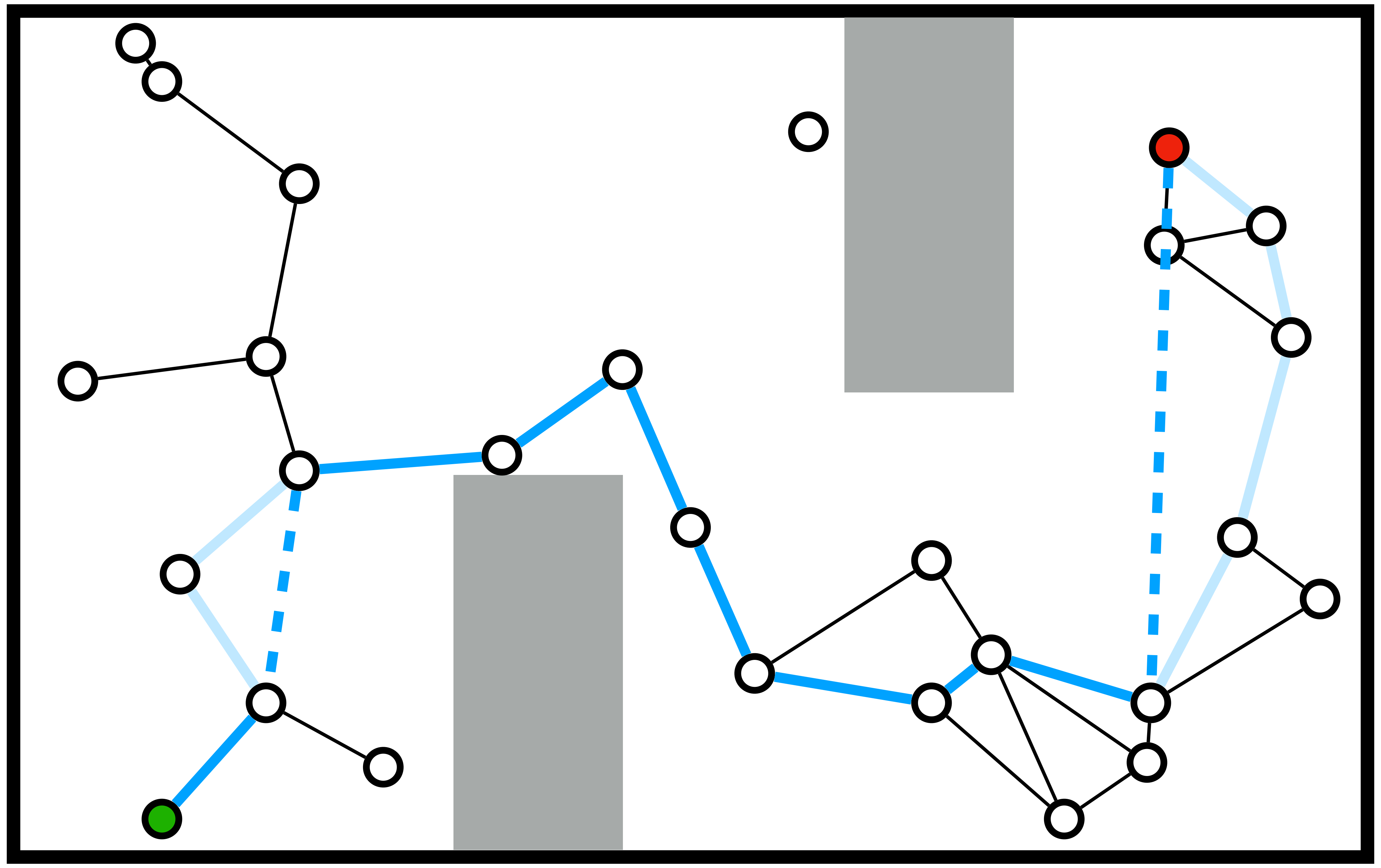
What Environments Are Hard?



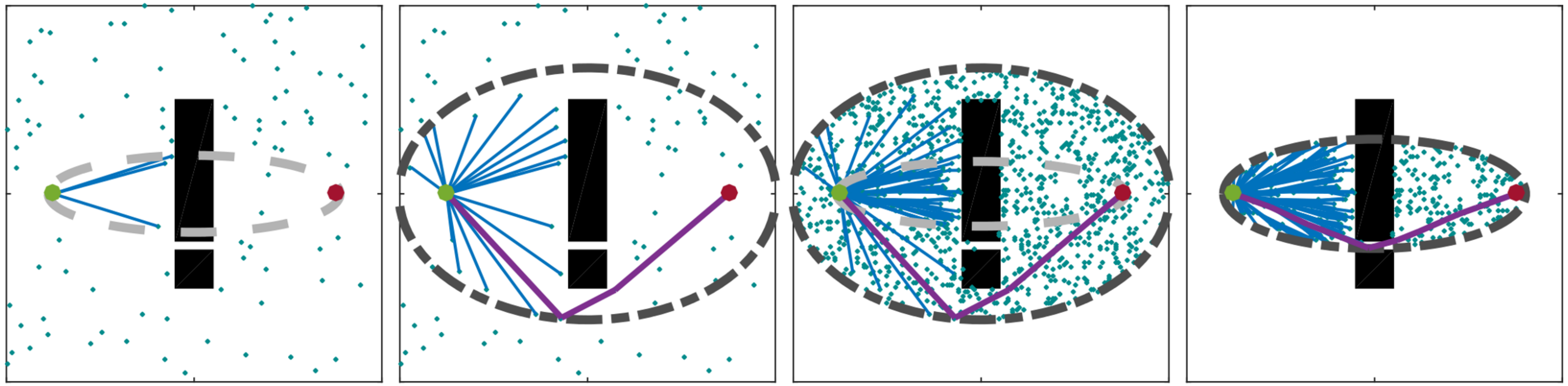
- Sampling-based methods struggle with **narrow passages**
- Probability of sampling an edge in the passage is very small, so with a finite number of samples, the two halves of the roadmap may not be connected
- Practical solutions: sample near obstacle surface, bridge test to add samples between two obstacles, train ML algorithm to detect narrow passages

Post-Processing Planned Paths

- Paths extracted from sampling-based motion planners tend to be jerky
- **Shortcutting**: along the planned path, randomly select two waypoints and try to connect them directly (skipping all intermediate vertices)



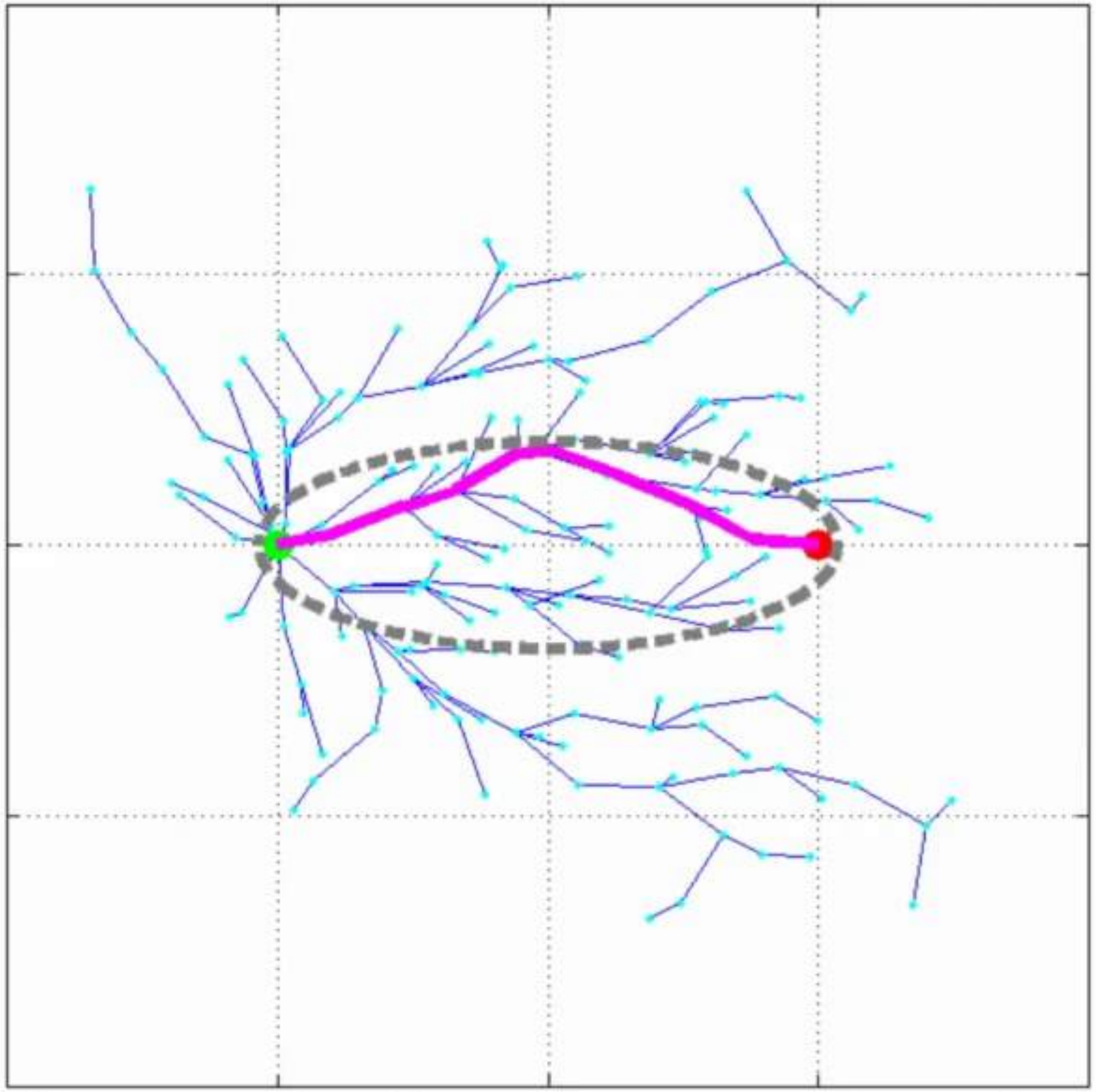
Bonus: Incremental Densification



- Interleave graph construction with graph search
- Euclidean heuristic + current best path length define an ellipse of useful states
- Only sample new configurations from this ellipse! (“informed set”)

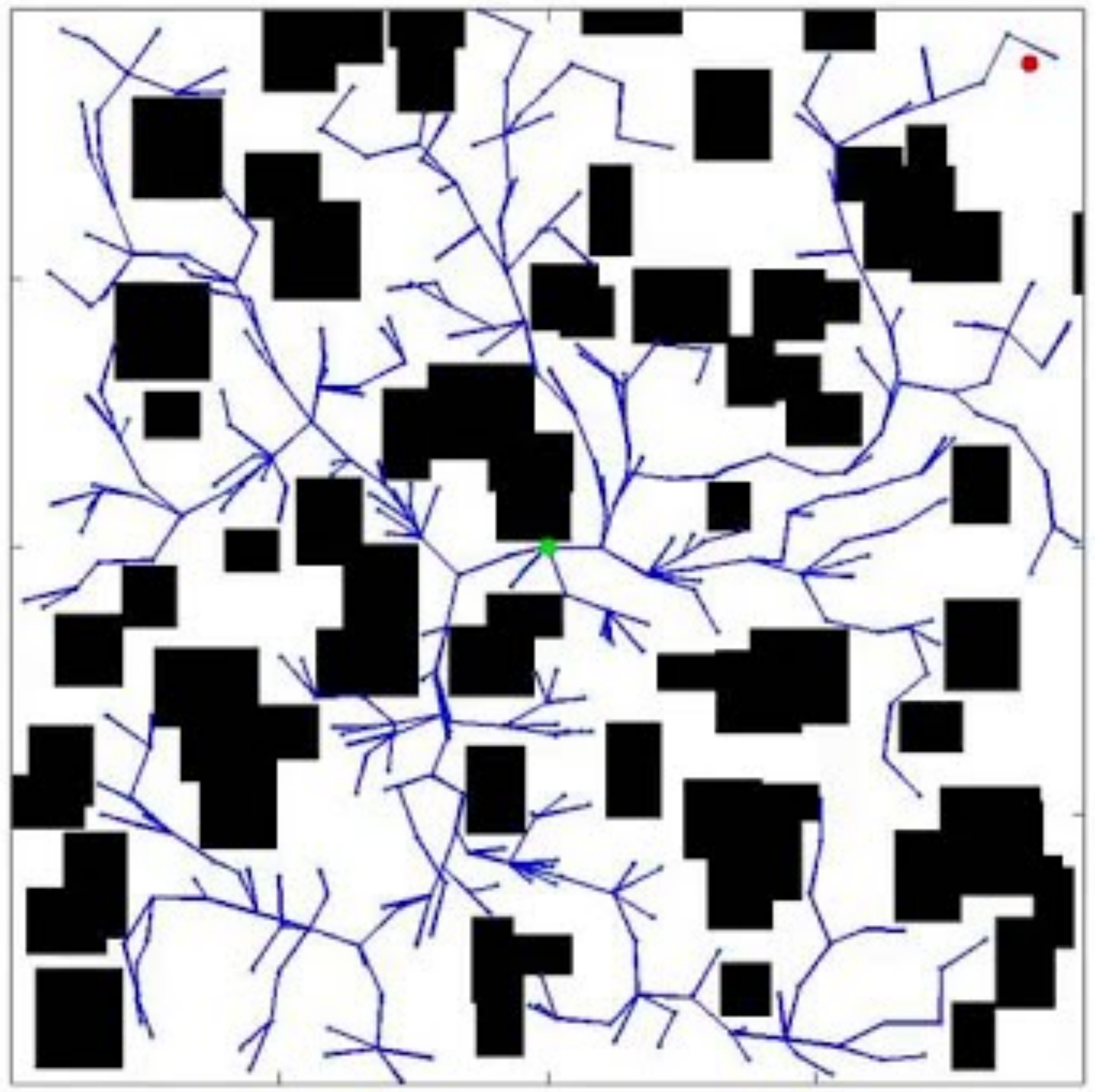
GAMMELL ET AL., 2015

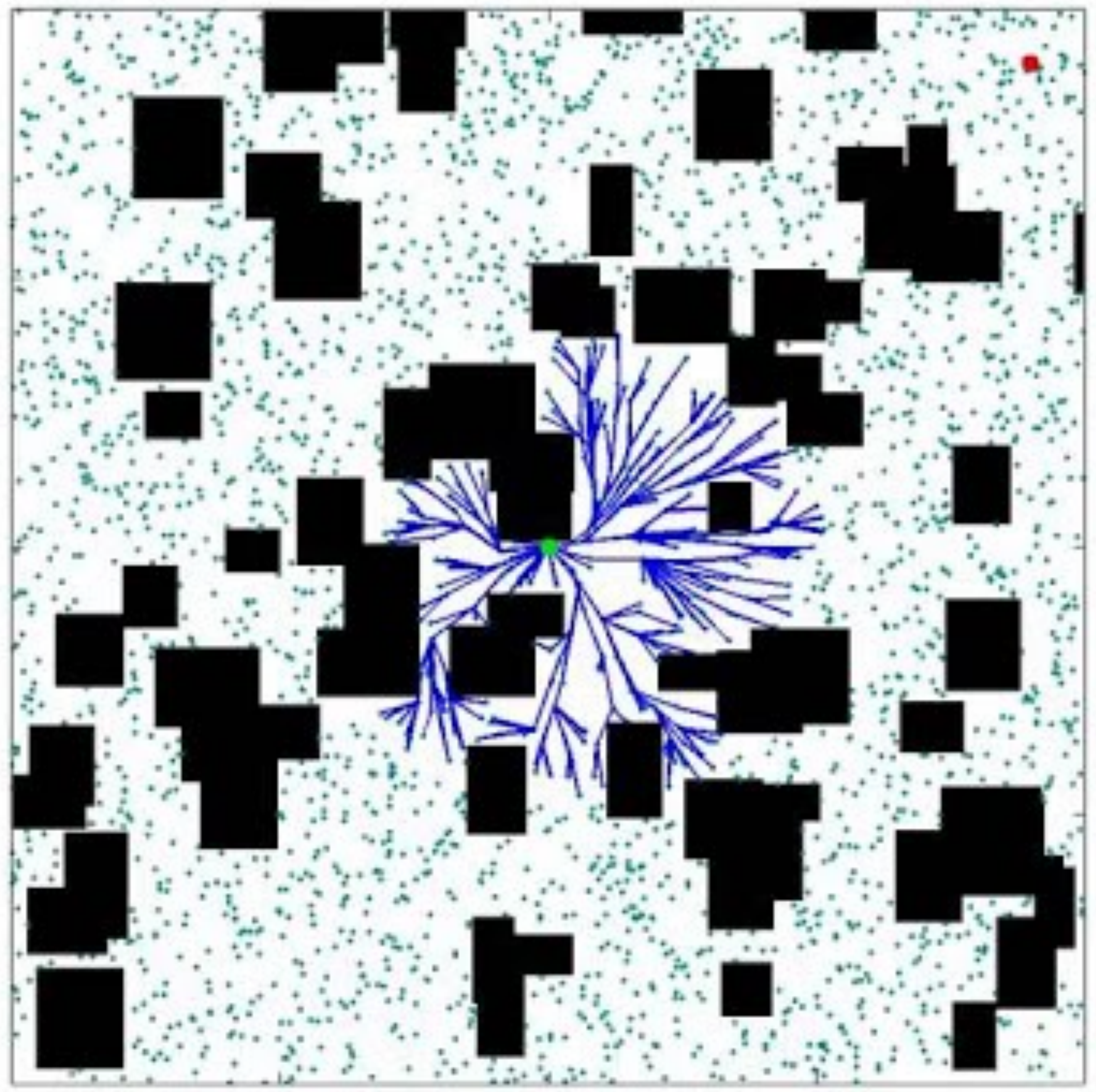
000177



By *directly* sampling the ellipse, we focus the search
to only the states that have the possibility of improving the solution.

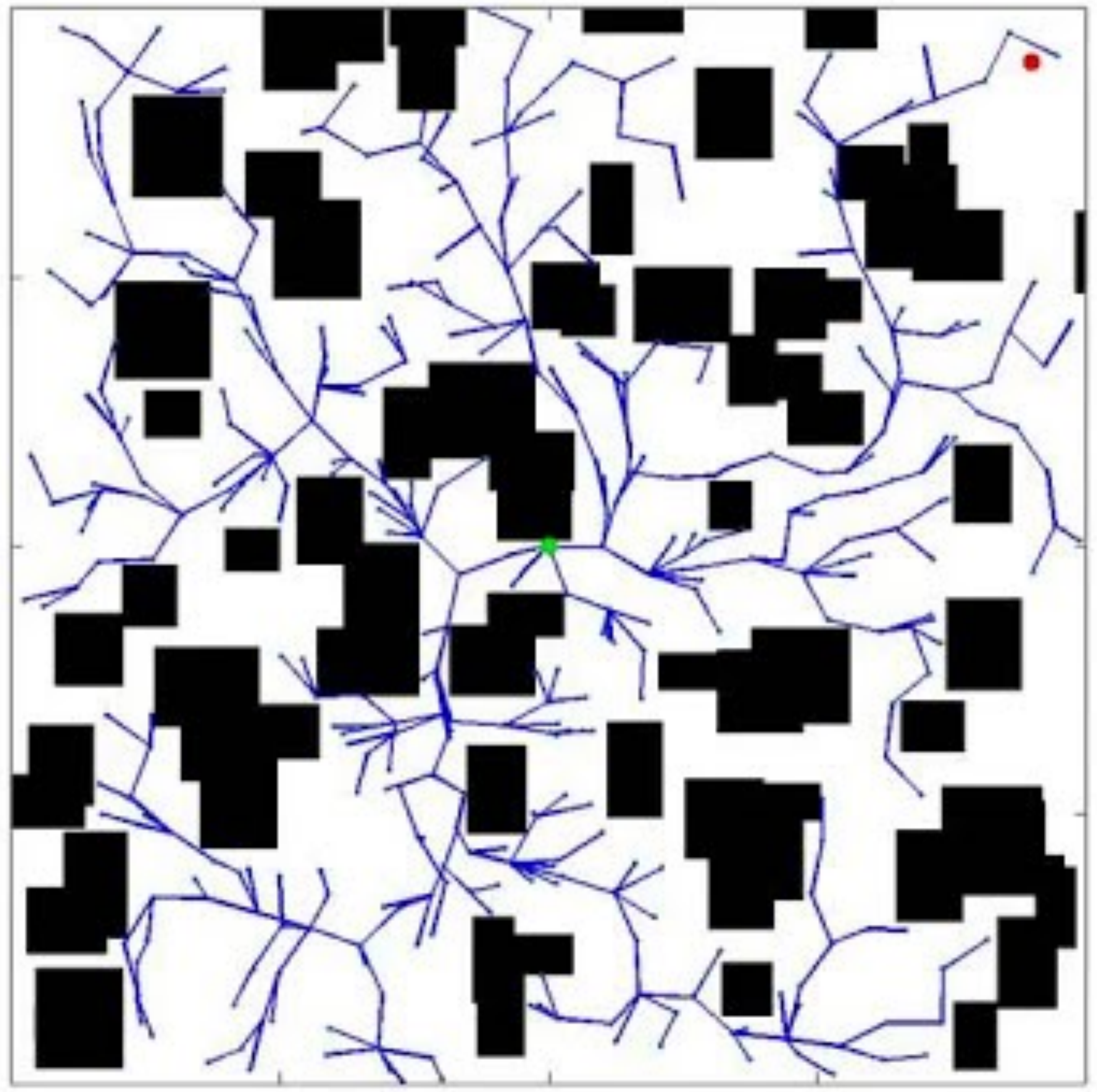
RRT*
$t = 00.031612s$
$c = \infty$

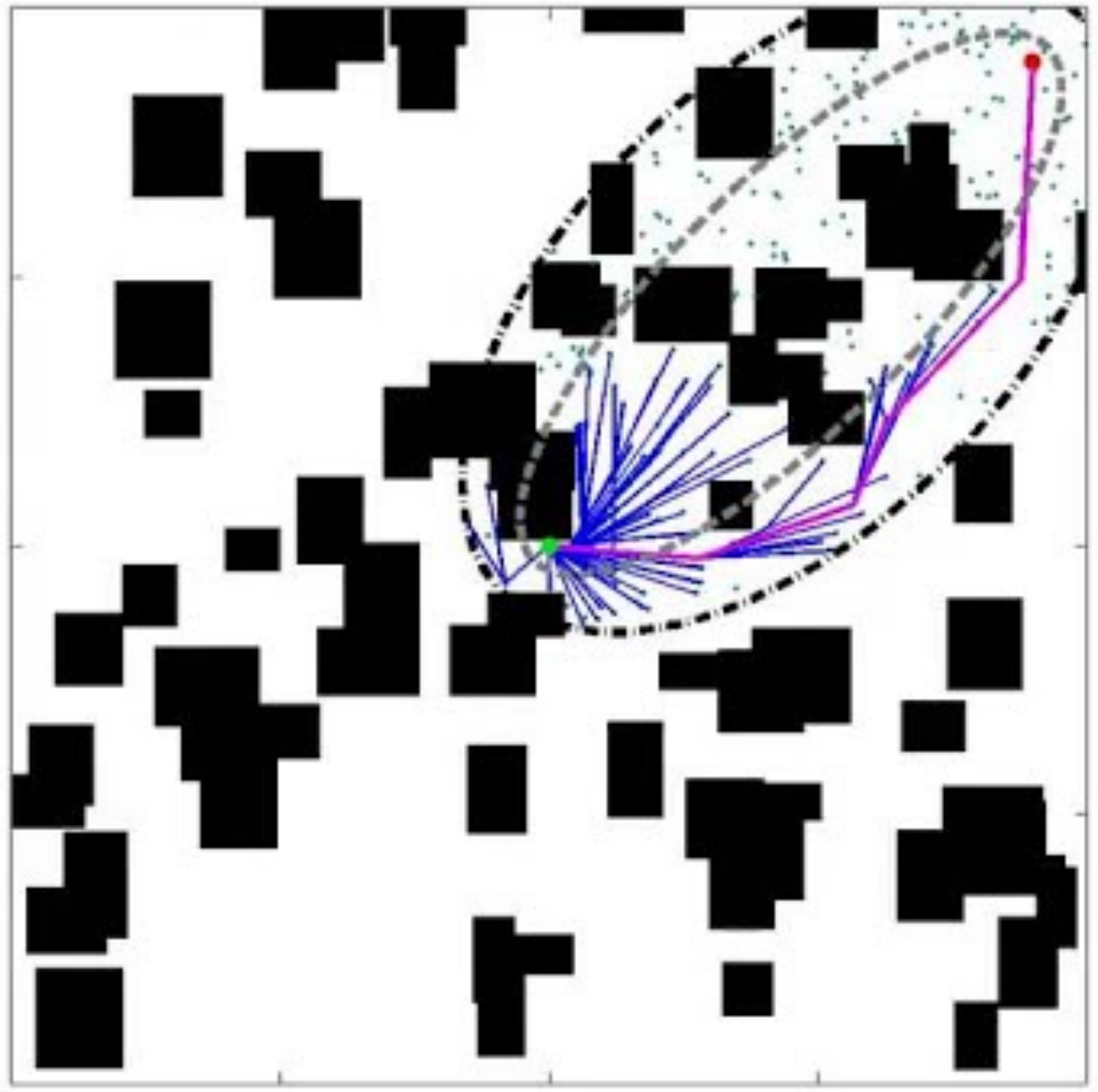




FMT*
$t = 00.031635s$
$c = \infty$

Informed RRT*
$t = 00.031511s$
$c = \infty$





BIT*
$t = 00.031508s$
$c = 01.518589$

CSE 478 Robot Autonomy
Roadmaps

Abhishek Gupta (abhgupta@cs)
Siddhartha Srinivasa (siddh@cs)

TAs:
Carolina Higuera (chiguera@cs)
Rishabh Jain (jrishabh@cs)
Entong Su (ensu@cs)