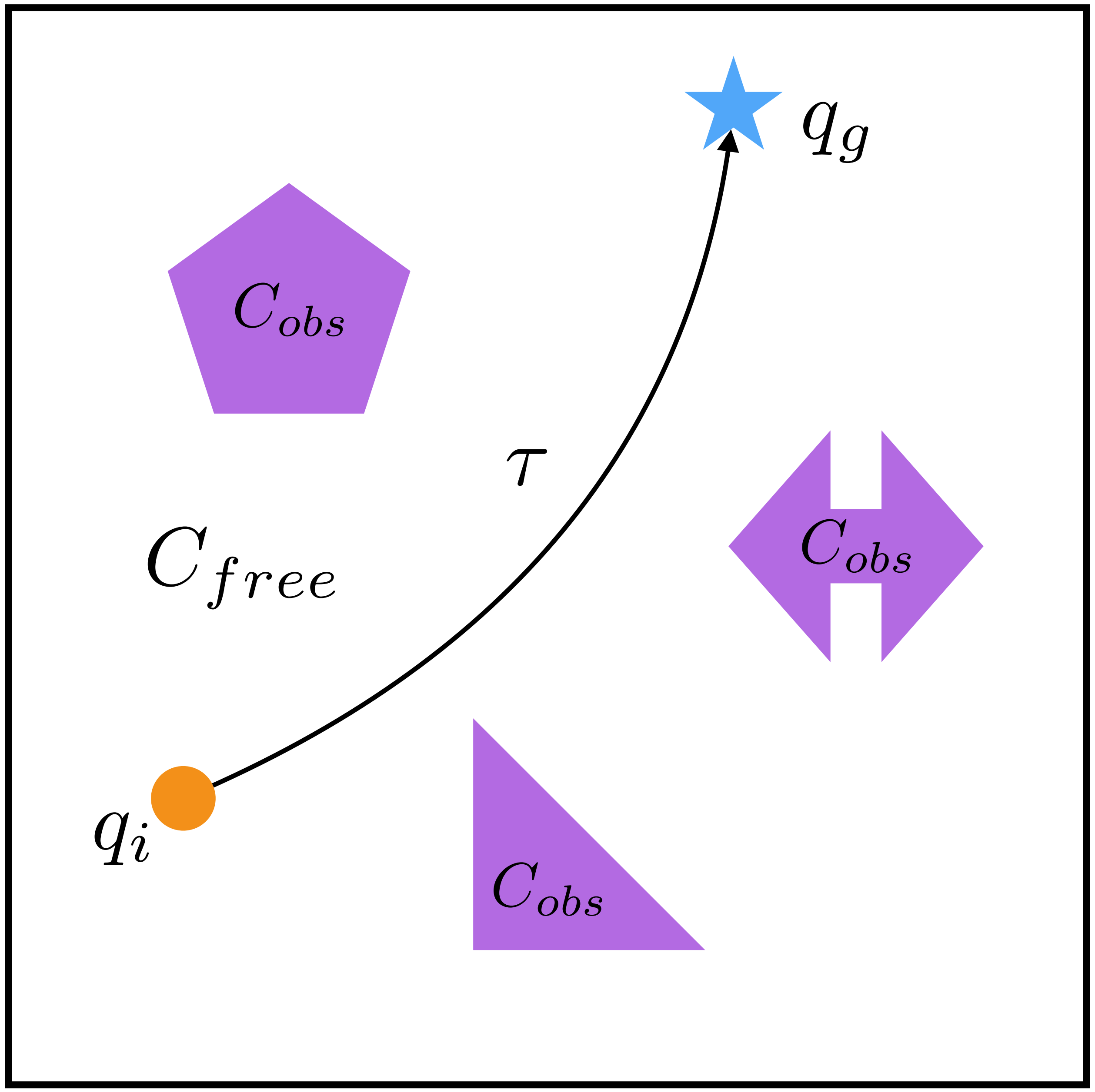
Robot Autonomy

Lecture 8: Motion Planning III

Oliver Kroemer

Piano Mover Problem



$$\tau : [0, 1] \to C_{free}$$

$$\tau(0) = q_i$$

$$\tau(1) = q_g$$

Topological Graphs

- Use graph to turn continuous problem into discrete one
- Define a topological graph $G(V, E)$ as:
  - Vertices

$$v_i \in C_{free}$$

  - Edges

$$e_{ij} \equiv \tau : [0, 1] \rightarrow C_{free}$$

Completeness

- Complete:
  Planner is complete if 1) it always returns a solution if one exists and 2) otherwise returns a failure in bounded time

- Probabilistic Complete:
  Planner is probabilistic complete if the probability of a solution existing tends to zero as the number of sampled points increases and no solution is found. No time bound.

- Focus on feasibility rather than optimality

Multi- vs Single- Query Sample-Based Planners

- Multi-query Planner
  - Build a roadmap for multiple planning queries
  - Assumes fixed obstacles across multiple queries
  - Cover C_{free} and capture its connectivity

- Single-query Planner
  - Build a graph for individual planning queries
  - Environment may change between queries
  - Connect the initial and goal nodes as directly as possible

General Procedure for Sample-based Planning

- Initialization:
  initialize graph $G = (V, E)$ as empty or with $V = \{q_i, q_g\}$
- Vertex Selection:
  Sample q_{new} in C_{free} to expand G
- Local Planning:
  Attempt to create paths between q_{new} and existing V
- Check if Done:
  Check termination conditions or goto vertex selection

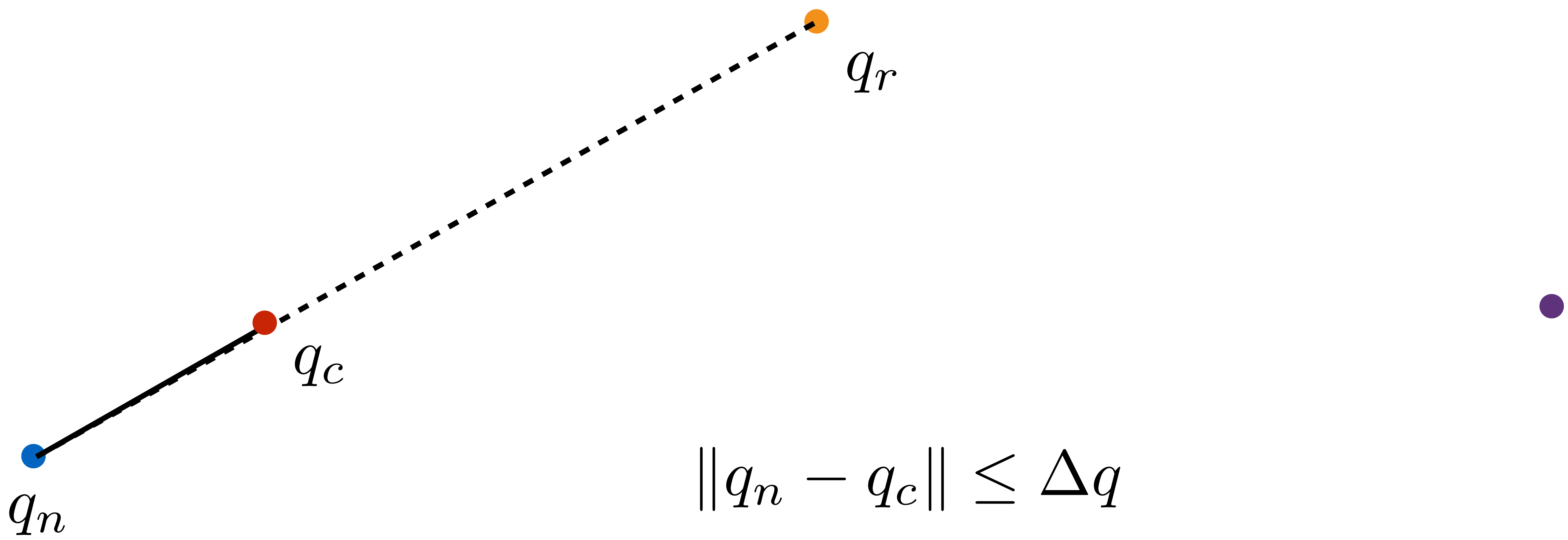
- Search graph for to find a path for the query q_i to q_g

Rapidly-Exploring Random Trees

- Initialize graph with $V = \{q_i, q_g\}$
  - Goal is to find a connecting path between initial and goal
- Add vertices to components that include either q_i or q_g
  - No additional components in c-space
- Grow the components in a tree structure (no loops)
  - More computationally efficient, but less optimal
- Grow quickly towards large unexplored regions
  - Explores the c-space rapidly

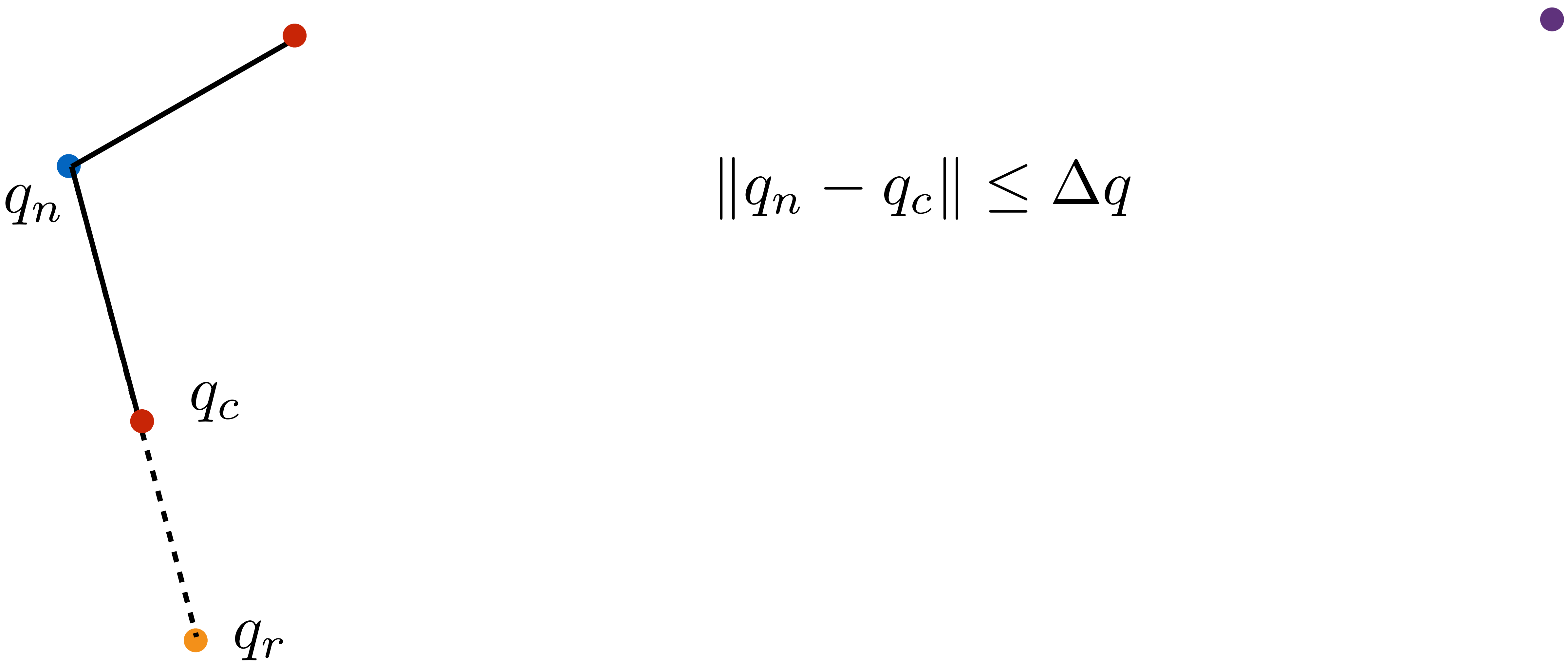
Rapidly-Exploring Random Trees

Rapidly-Exploring Random Trees



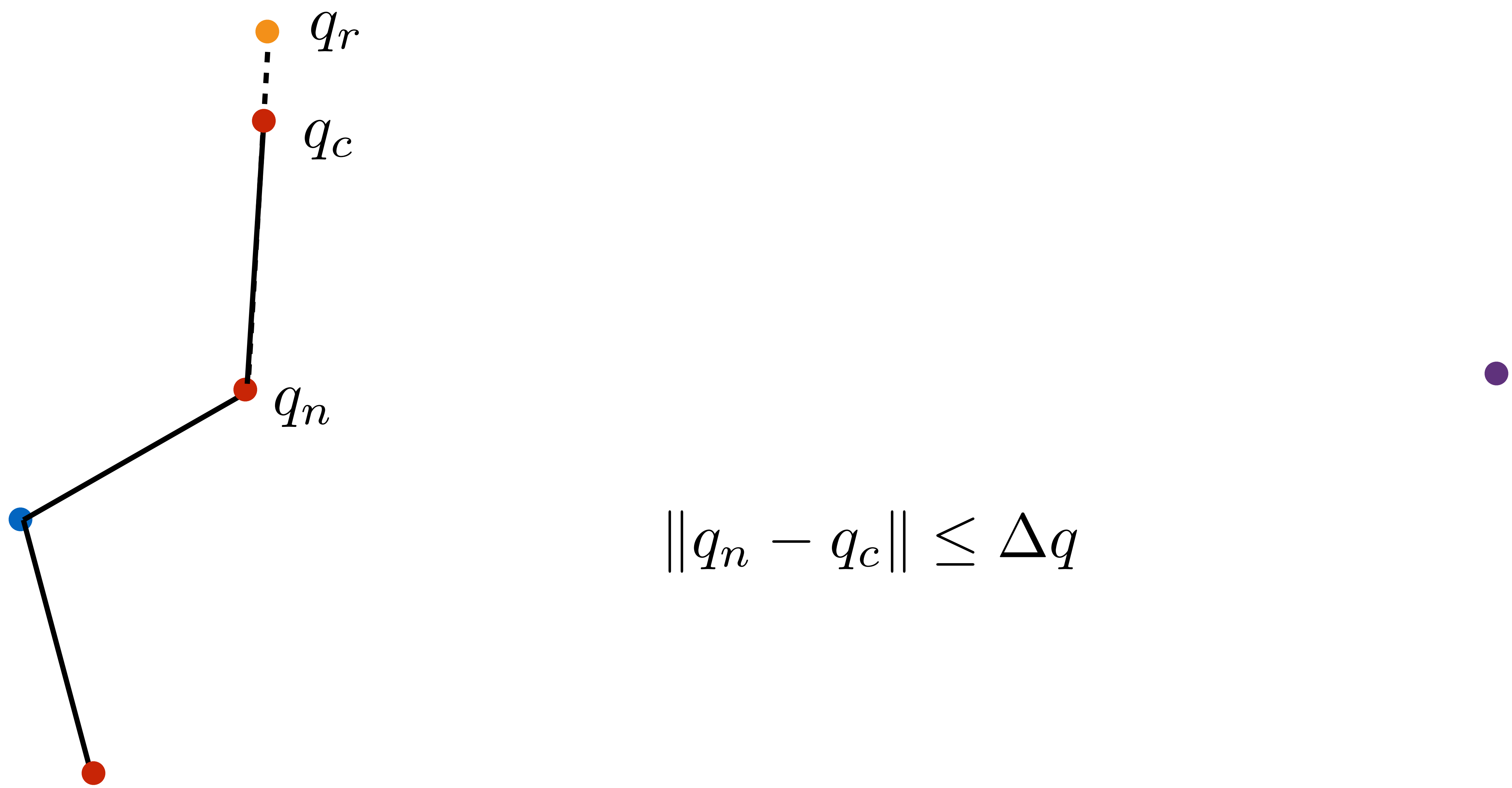
$$\|q_n - q_c\| \leq \Delta q$$

Rapidly-Exploring Random Trees



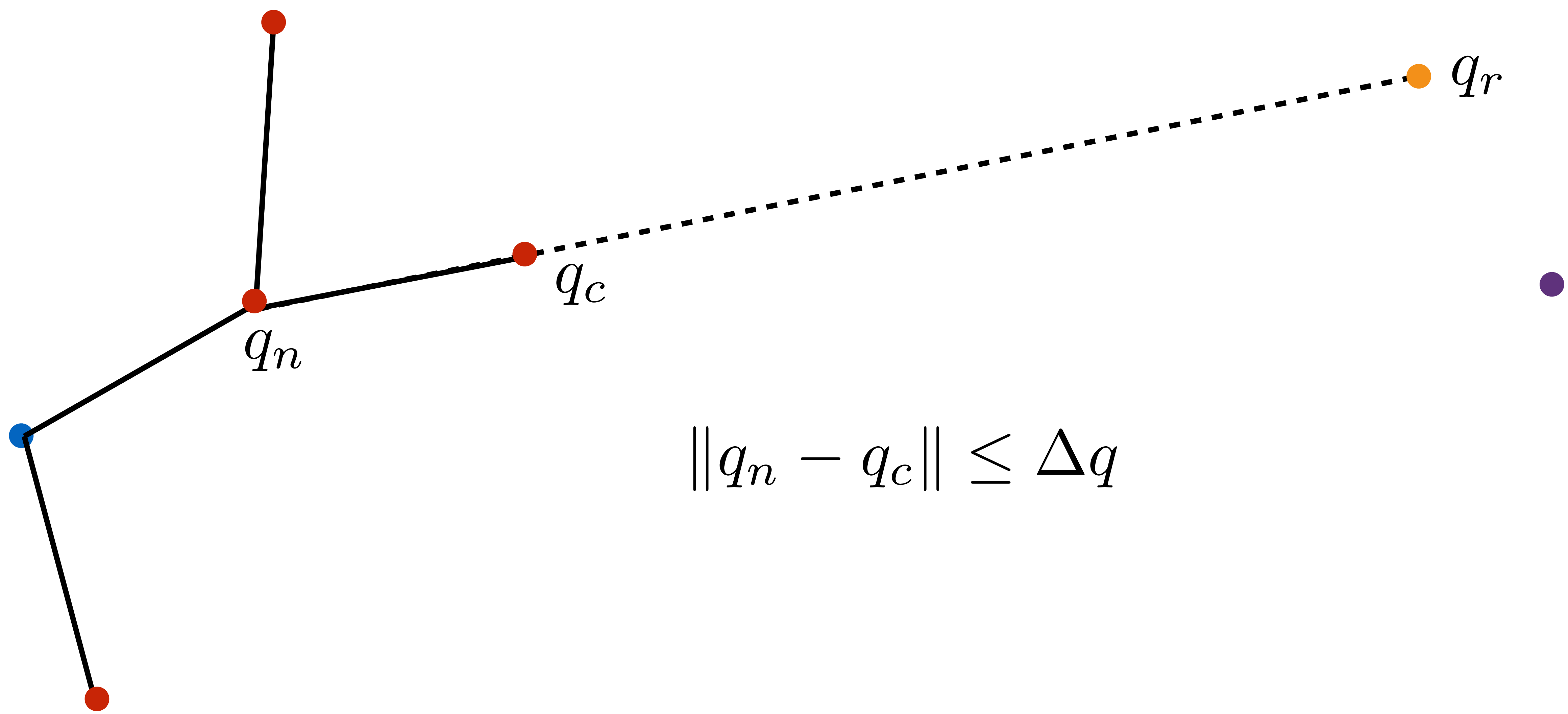
$$\|q_n - q_c\| \leq \Delta q$$

Rapidly-Exploring Random Trees



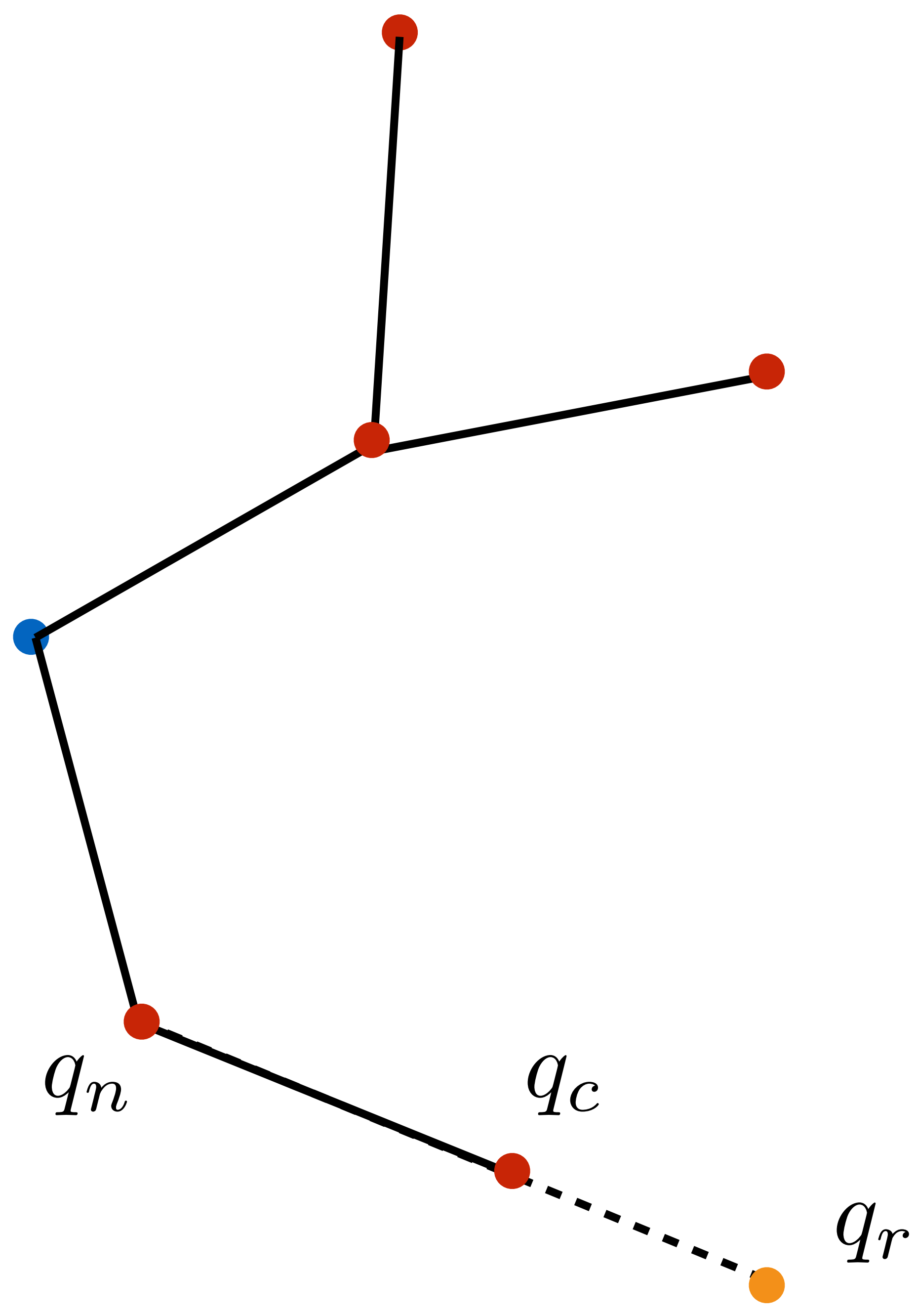
$$\|q_n - q_c\| \leq \Delta q$$

Rapidly-Exploring Random Trees



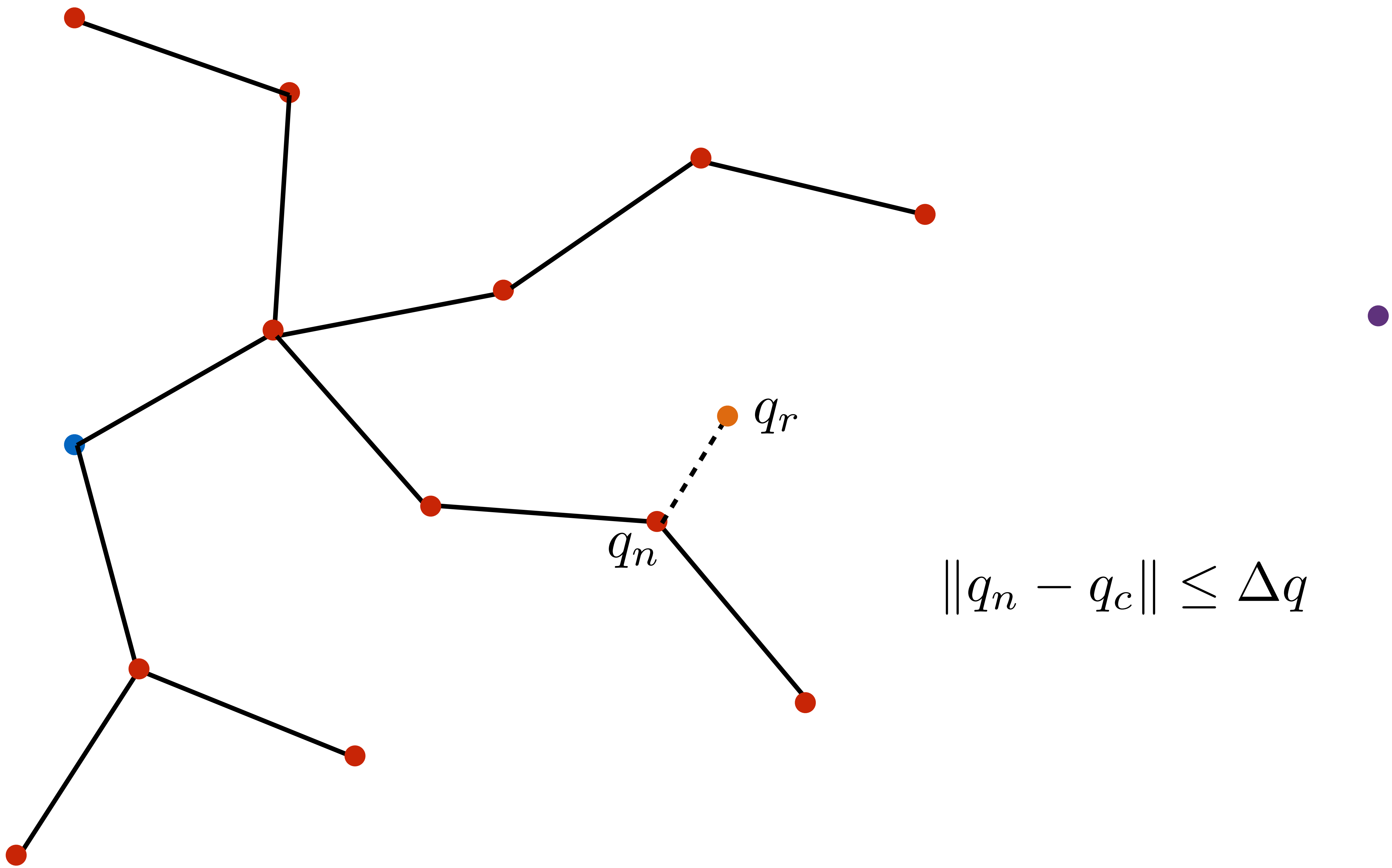
$$\|q_n - q_c\| \leq \Delta q$$

Rapidly-Exploring Random Trees



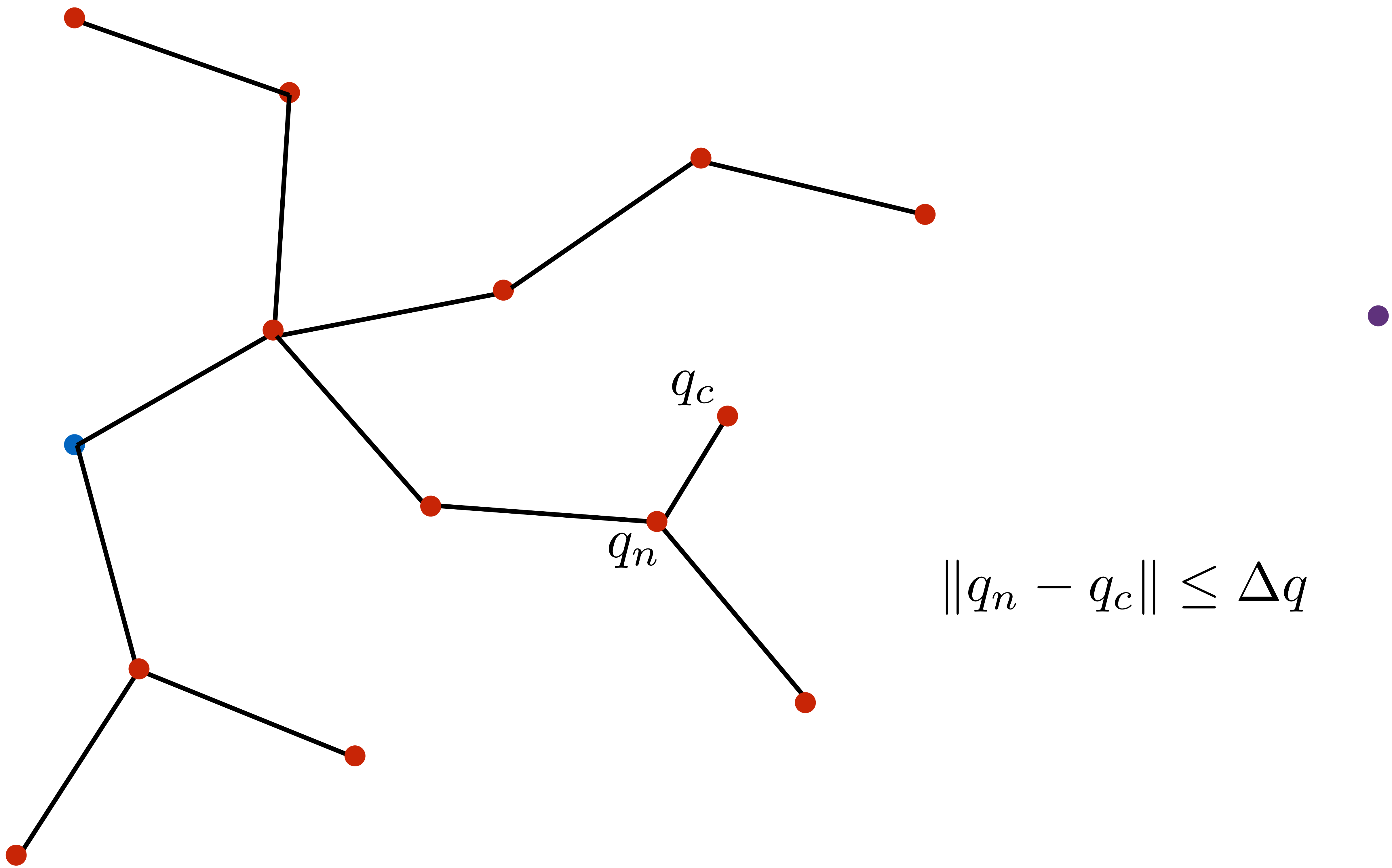
$$\|q_n - q_c\| \leq \Delta q$$

Rapidly-Exploring Random Trees



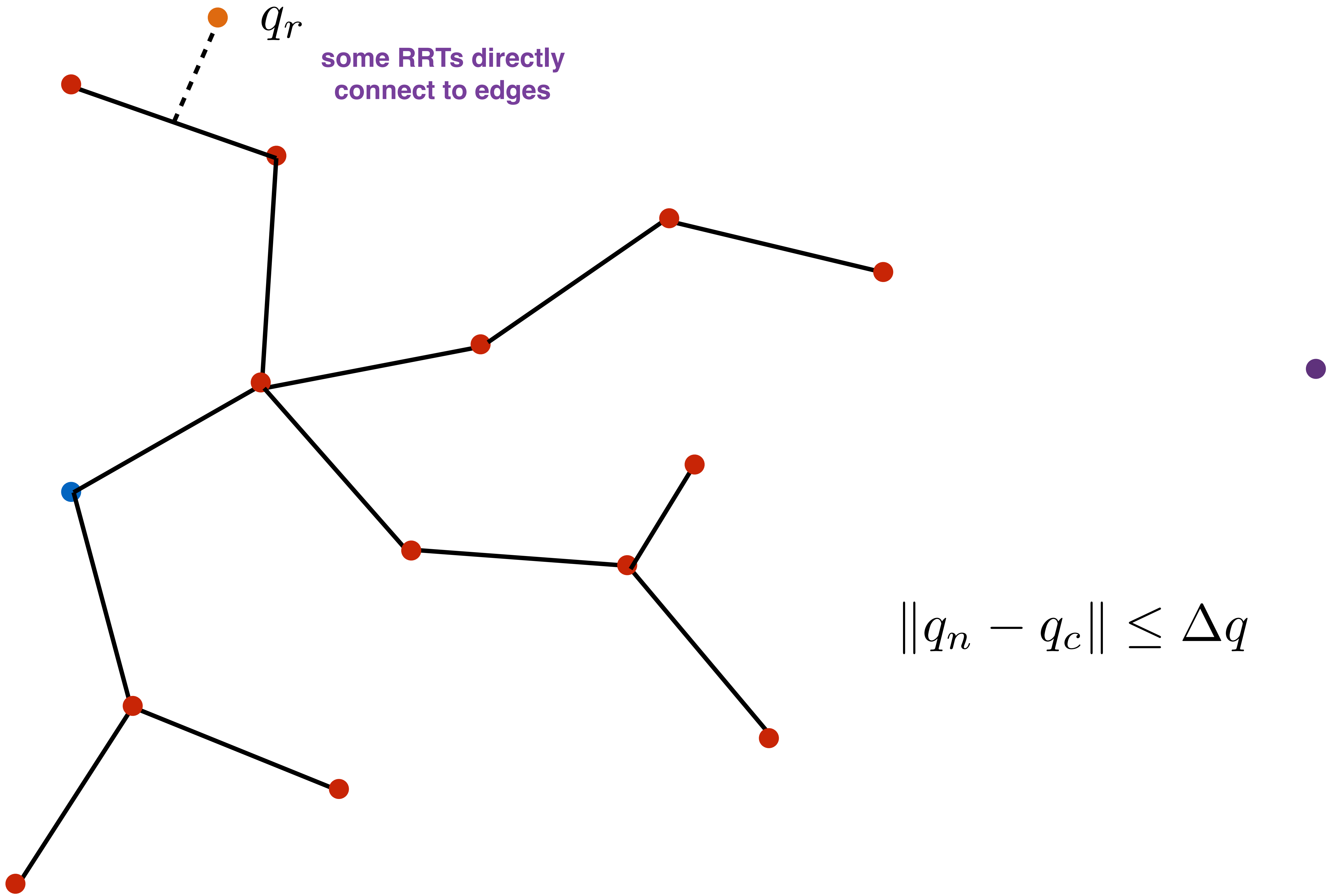
$$\|q_n - q_c\| \leq \Delta q$$

Rapidly-Exploring Random Trees



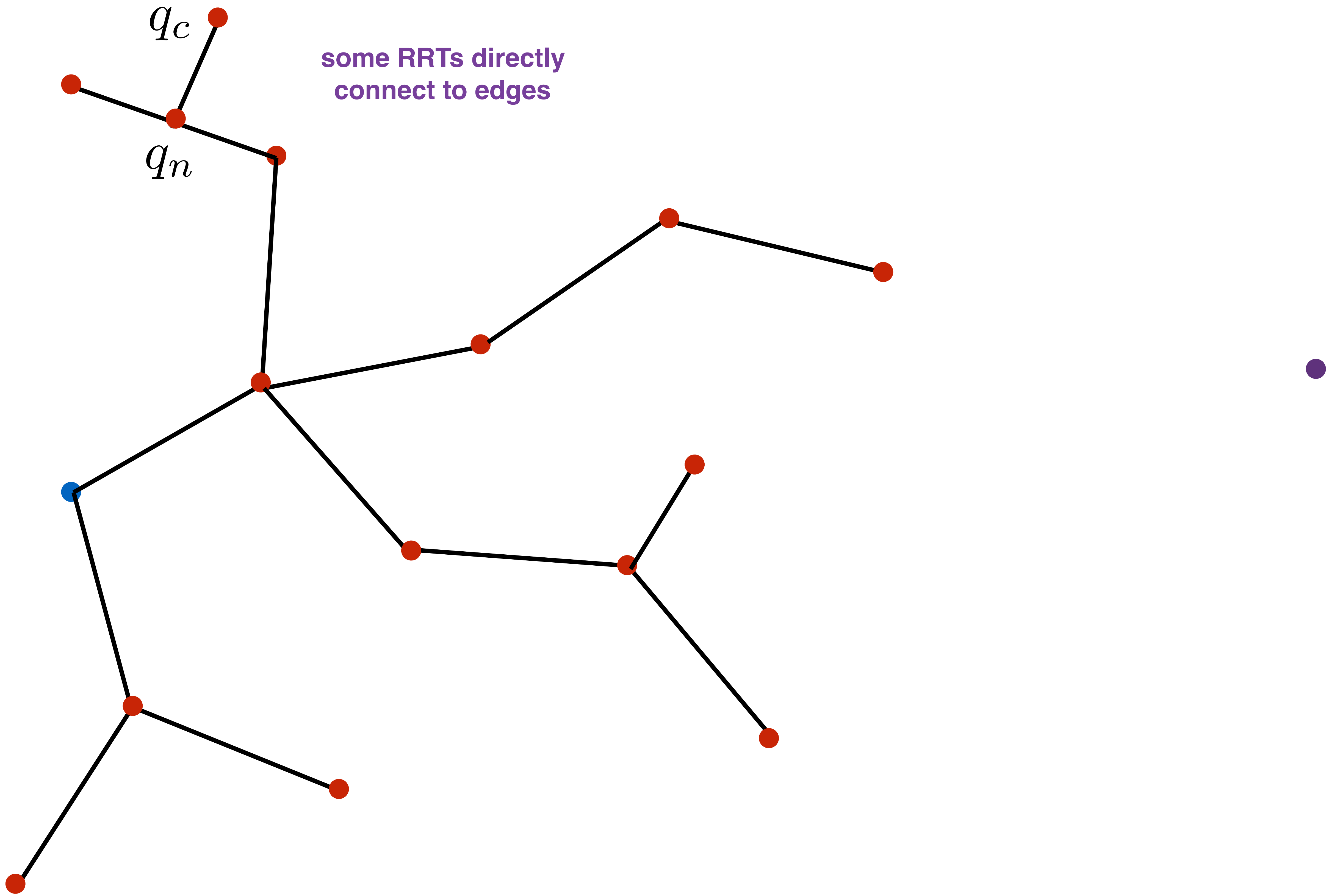
$$\|q_n - q_c\| \leq \Delta q$$

Rapidly-Exploring Random Trees

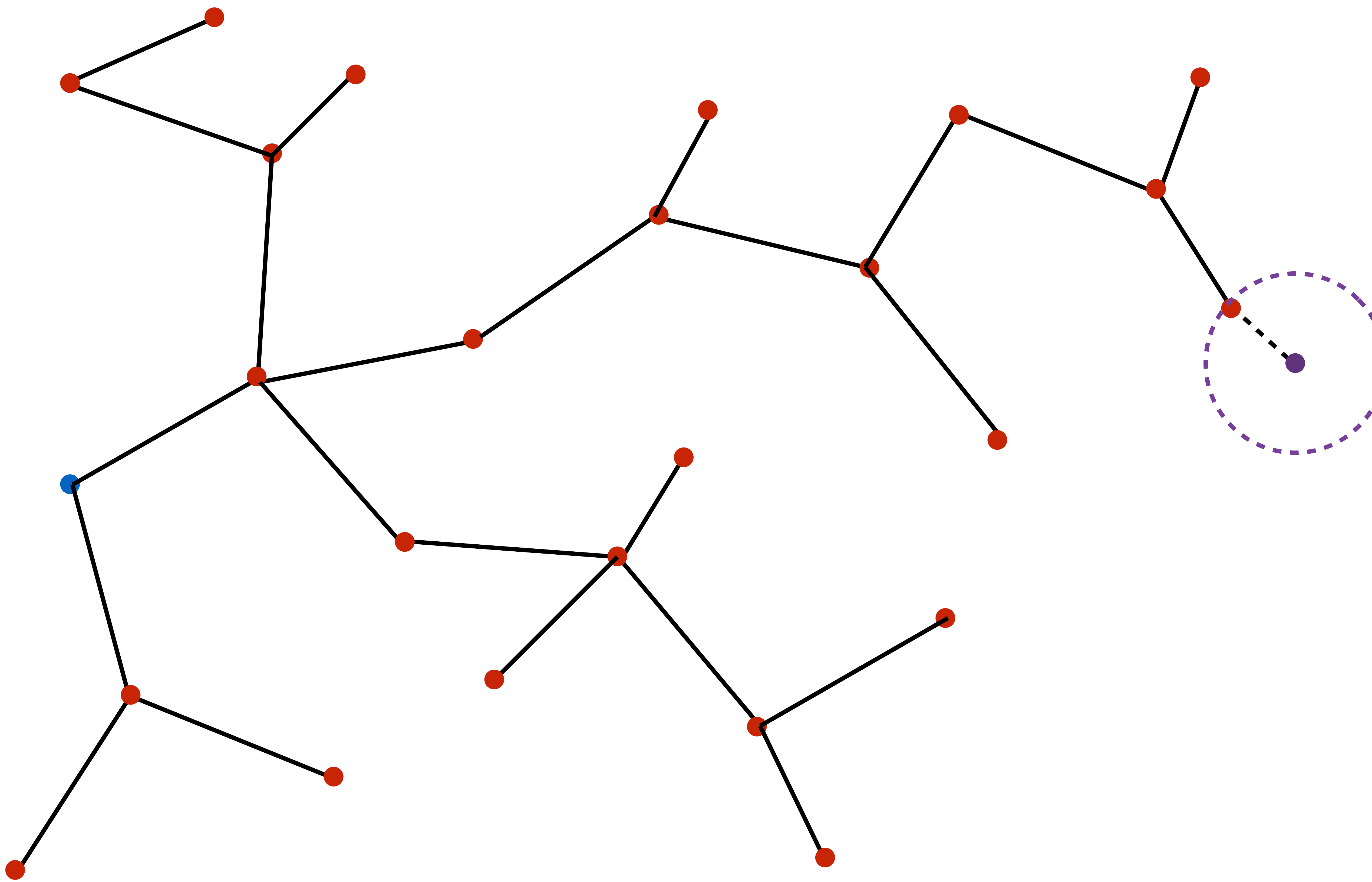


$$\|q_n - q_c\| \leq \Delta q$$

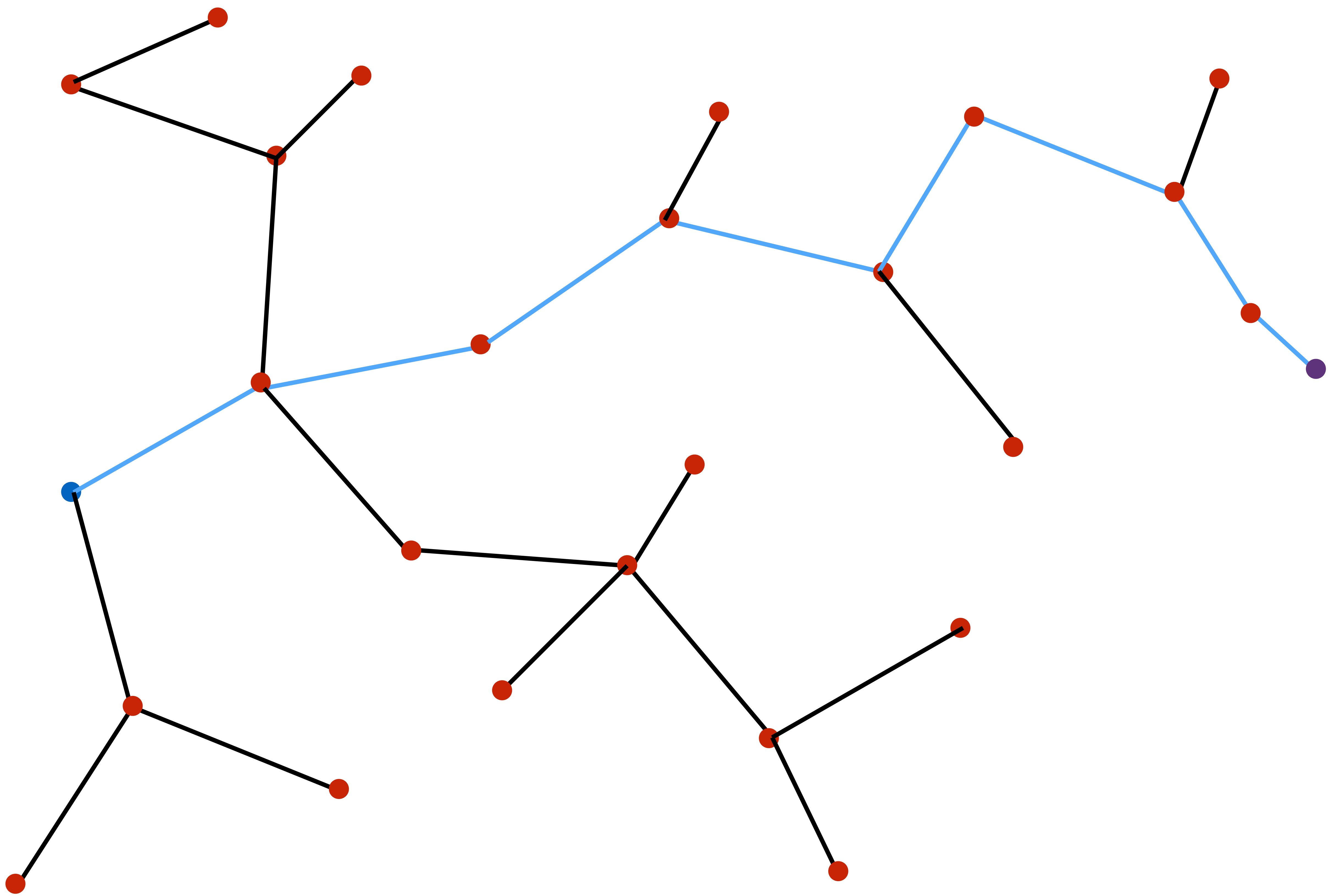
Rapidly-Exploring Random Trees



Rapidly-Exploring Random Trees



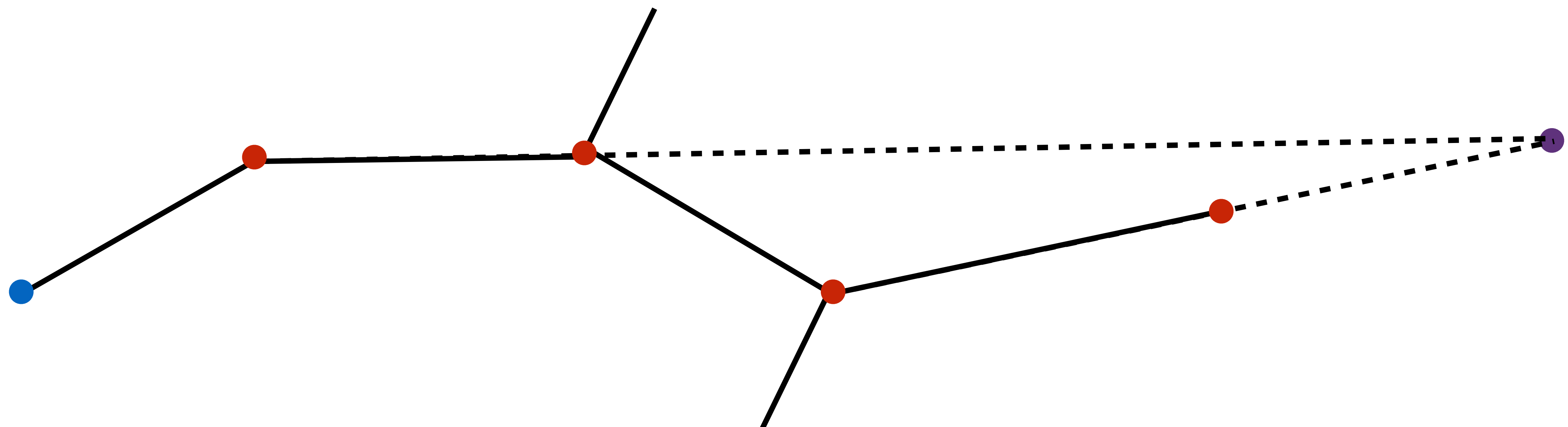
Rapidly-Exploring Random Trees



Goal Bias

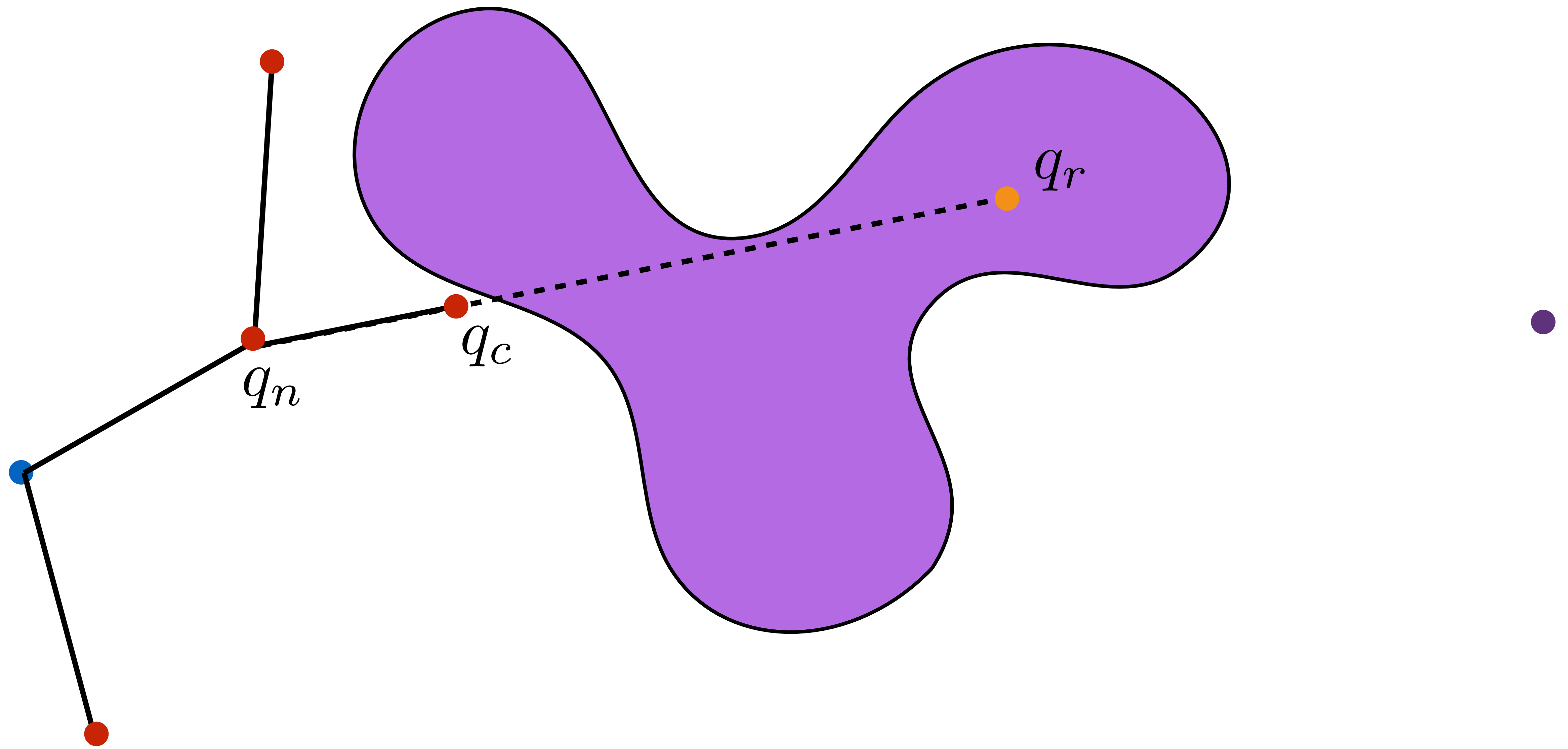
- Idea:
  - Bias sampling towards goal to quickly create a path to q_g

- How:
  - With probability p use q_g as q_r
  - Set p small, e.g., 0.01 to 0.1



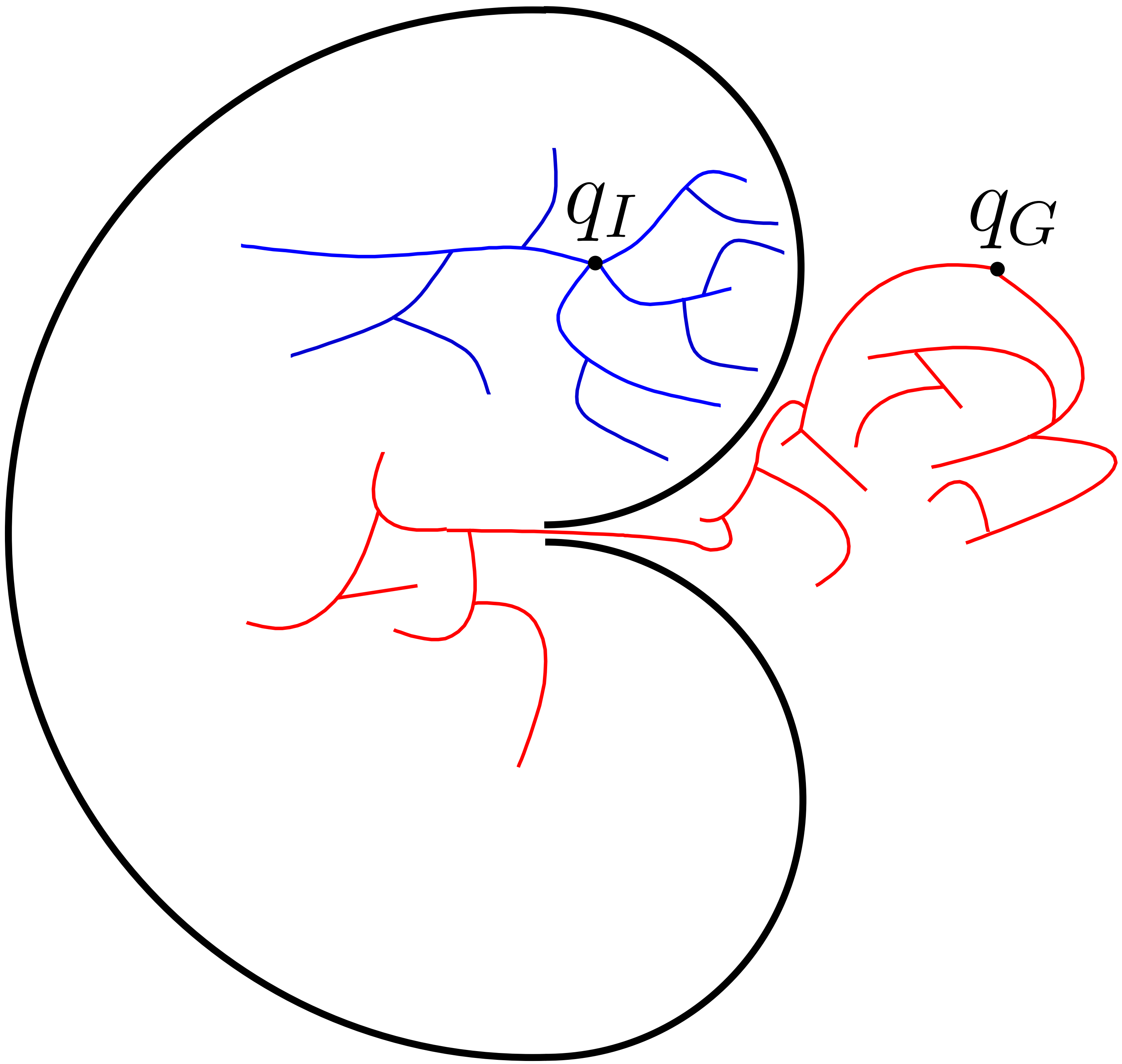
Obstacles

- Reject, or move towards q_r until a collision is detected



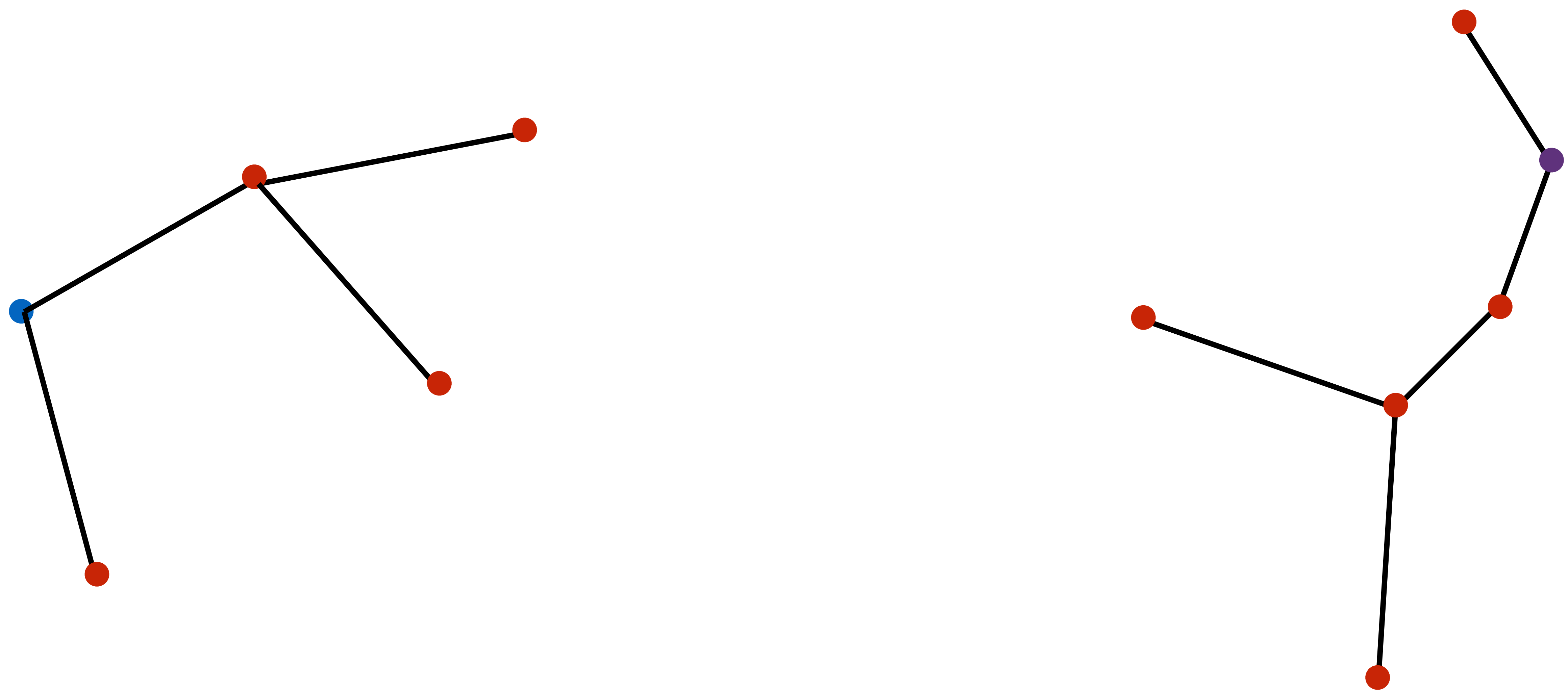
- Do not add vertex if q_n is already near collision
- May also resample q_r if it is not in C_{free}

Bidirectional Search

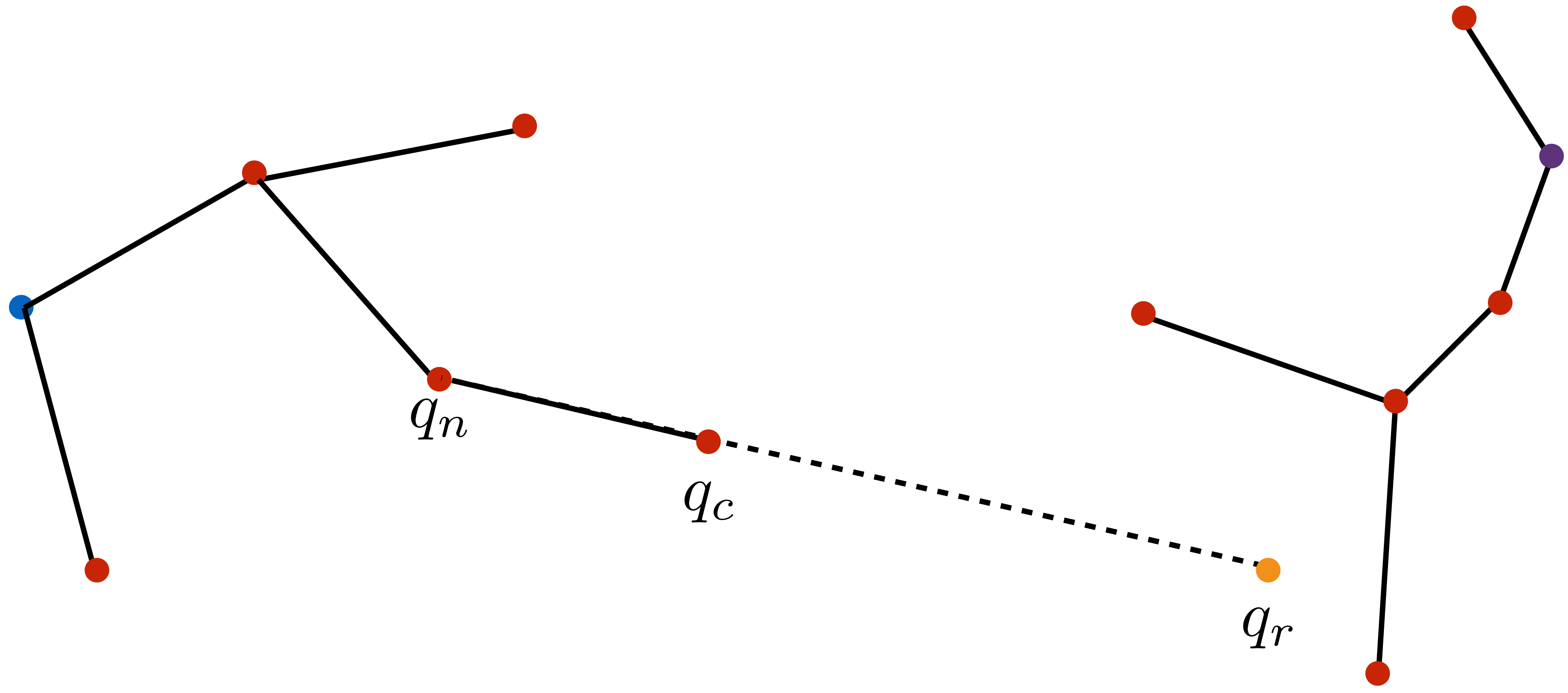


LaValle: Planning Algorithms, 2006

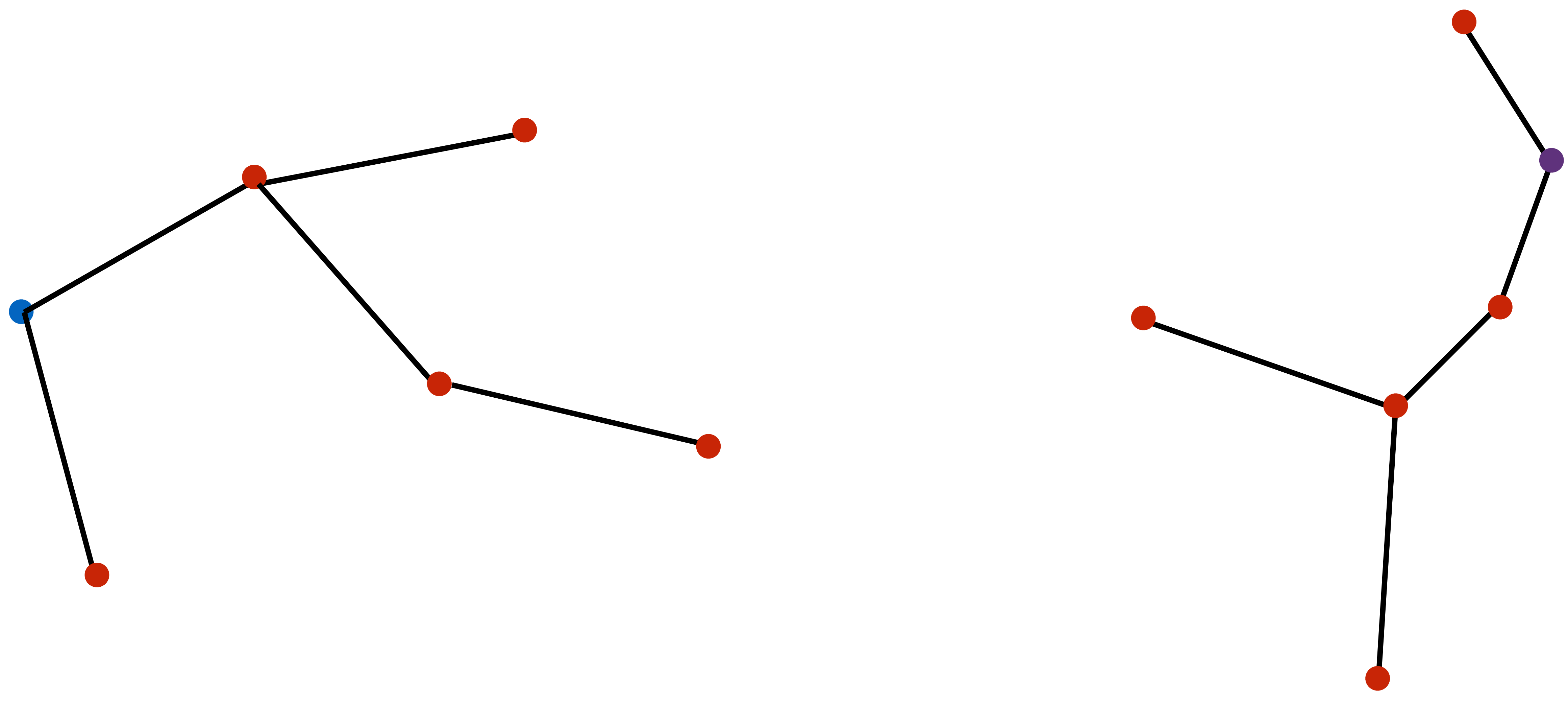
Bidirectional Search



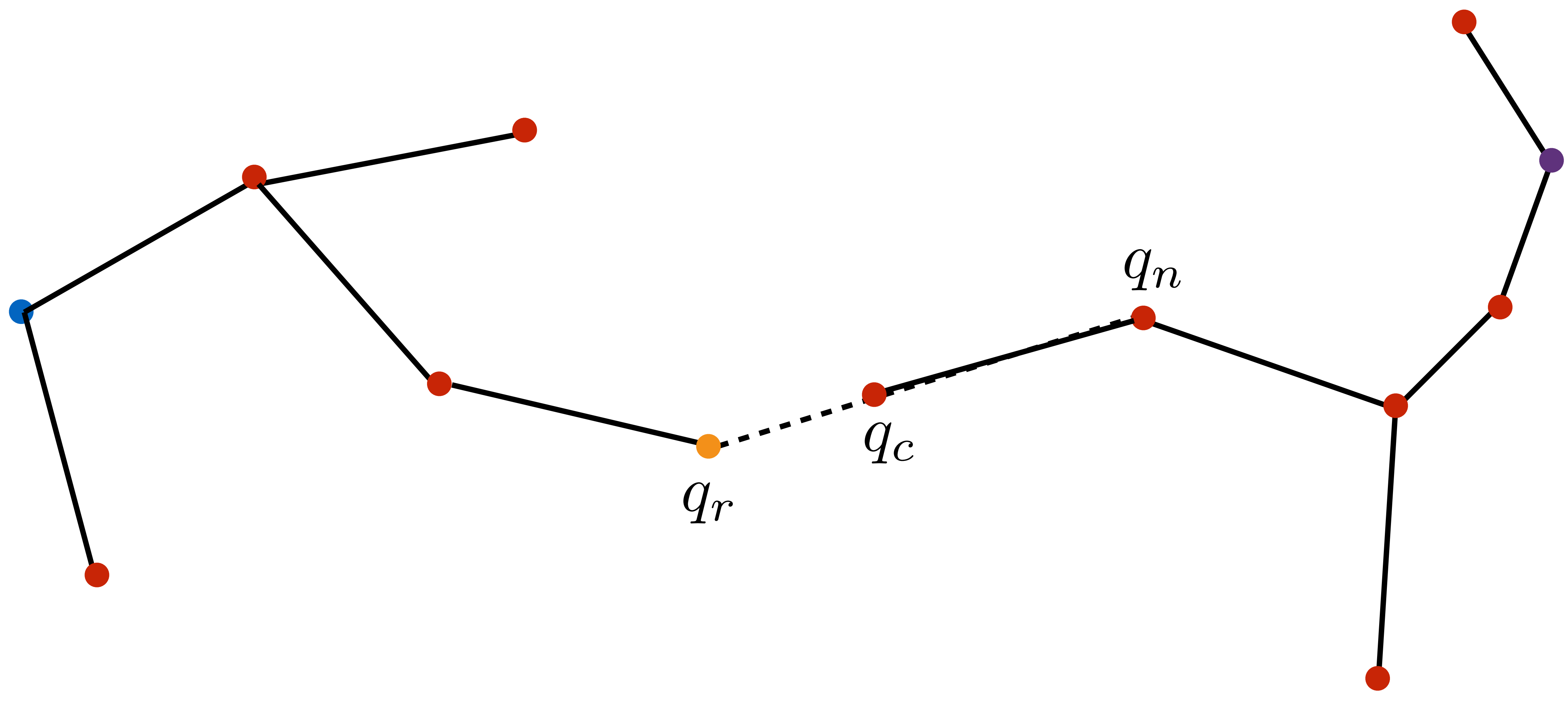
Bidirectional Search



Bidirectional Search

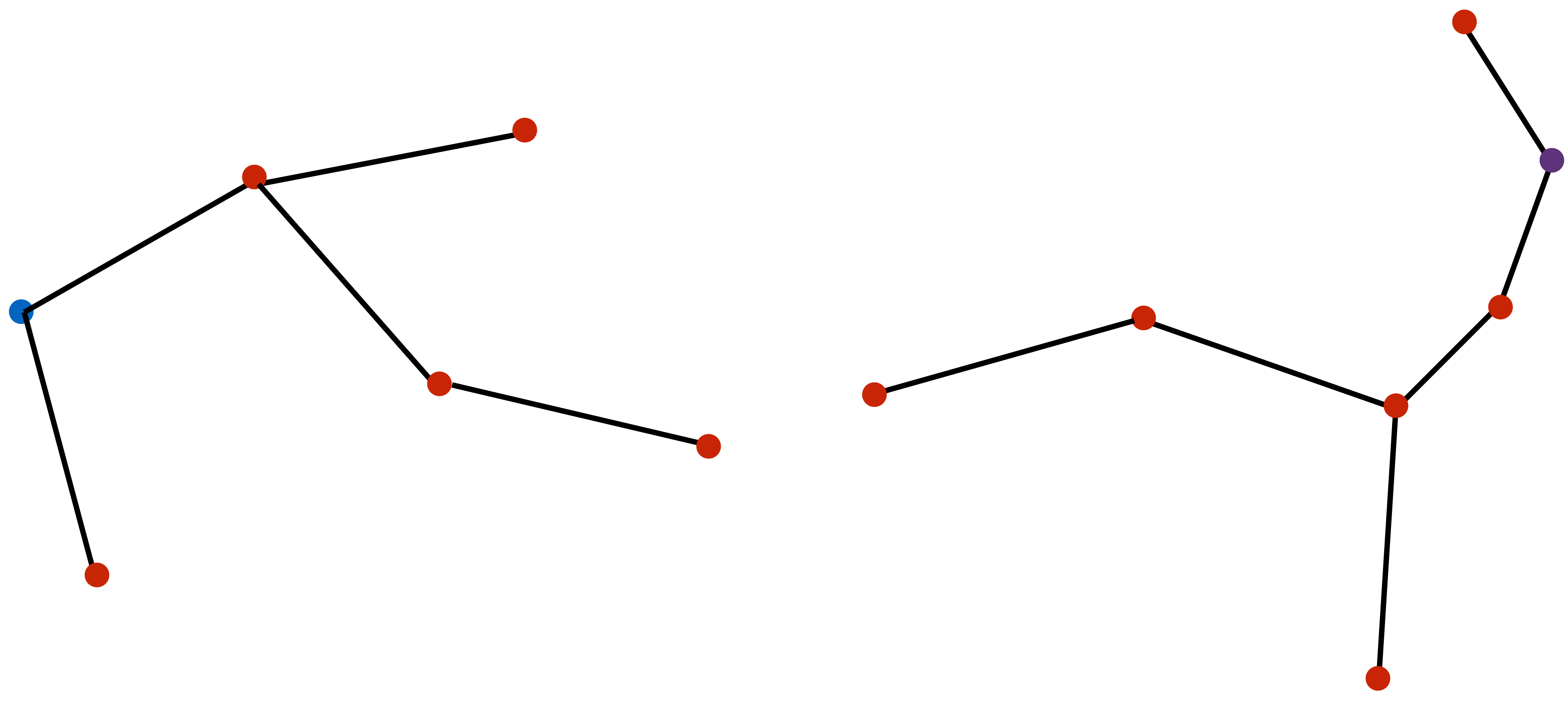


Bidirectional Search



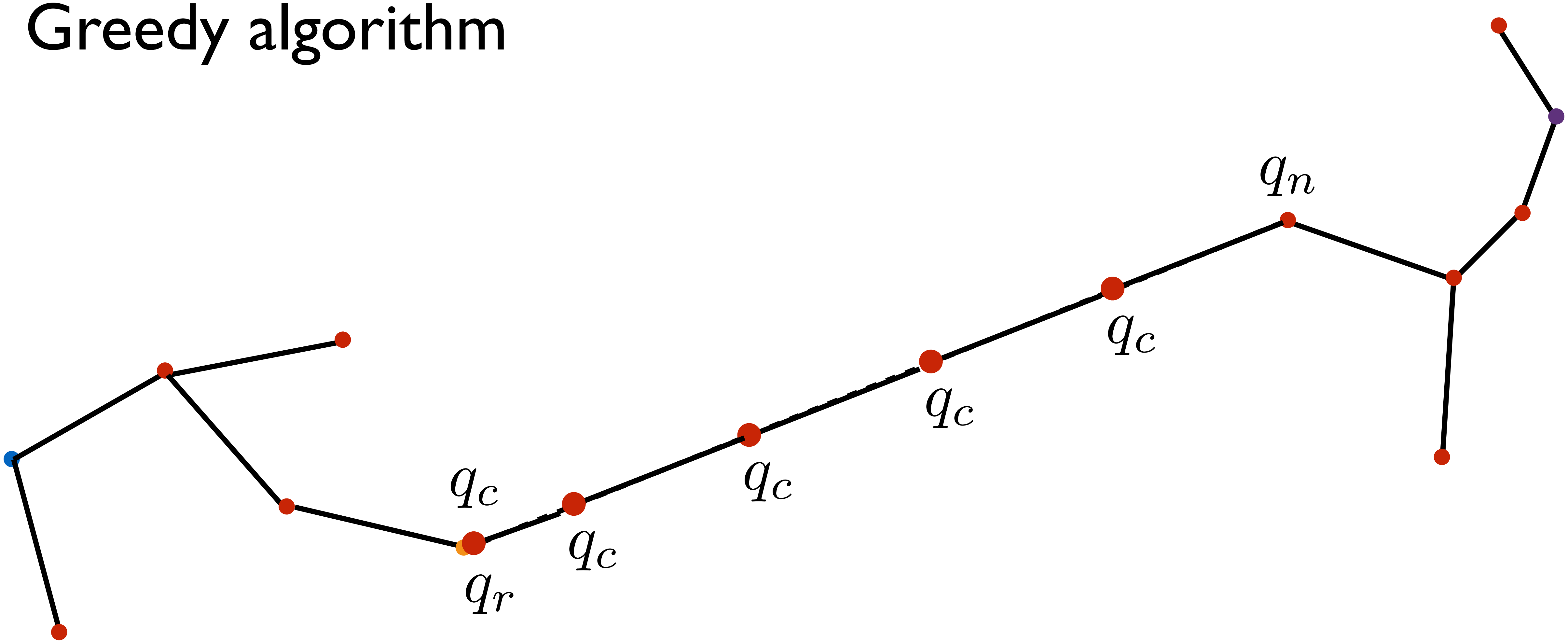
Balanced Bidirectional Search

- Every few iterations switch between tree to expand first
- Alternatively, expand the tree with fewer vertices first

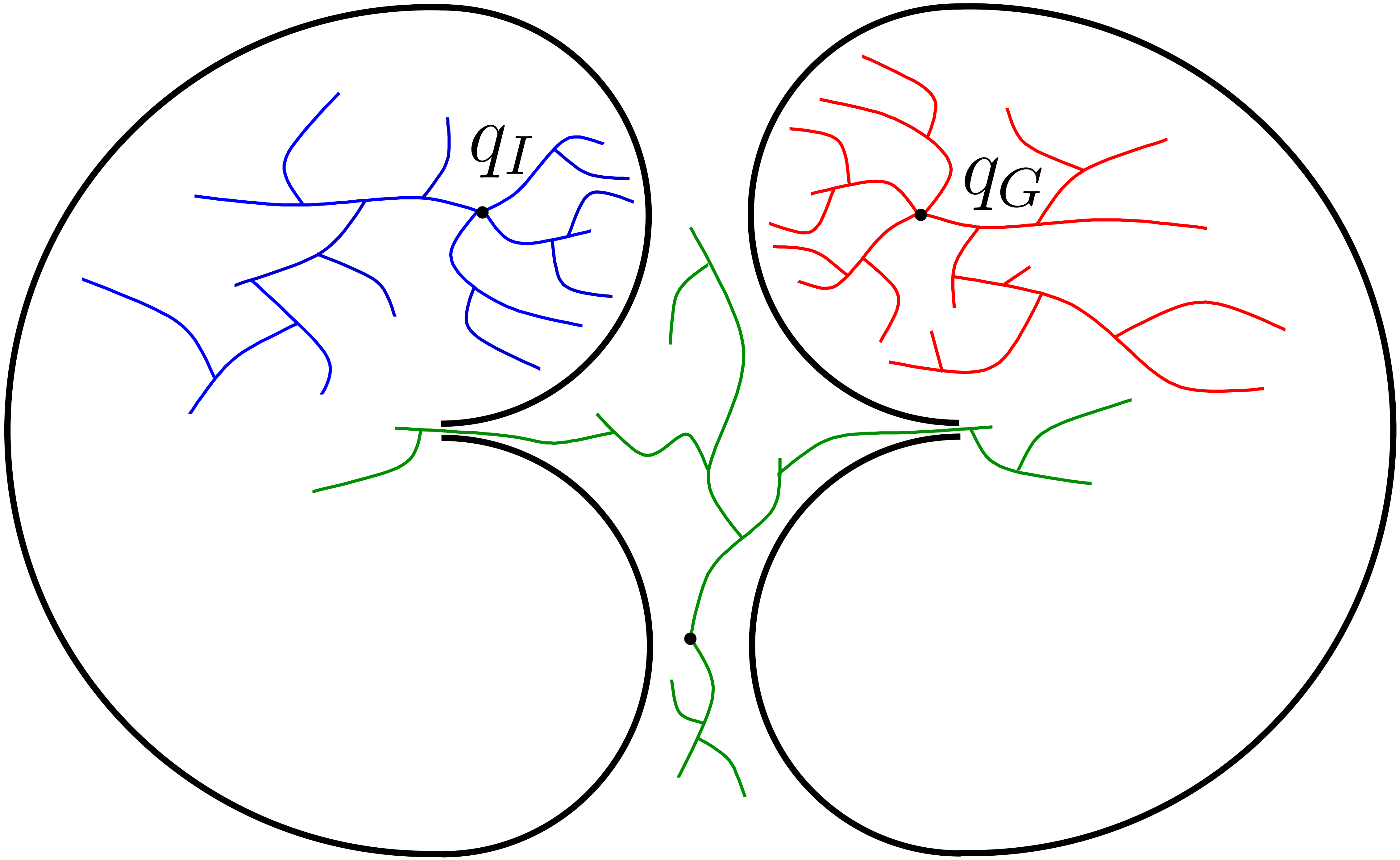


RRT-Connect

- Similar to balanced bidirectional search
- Attempt to connect second tree to new node
- Connect: keep expanding until reach first tree or collision
- Greedy algorithm

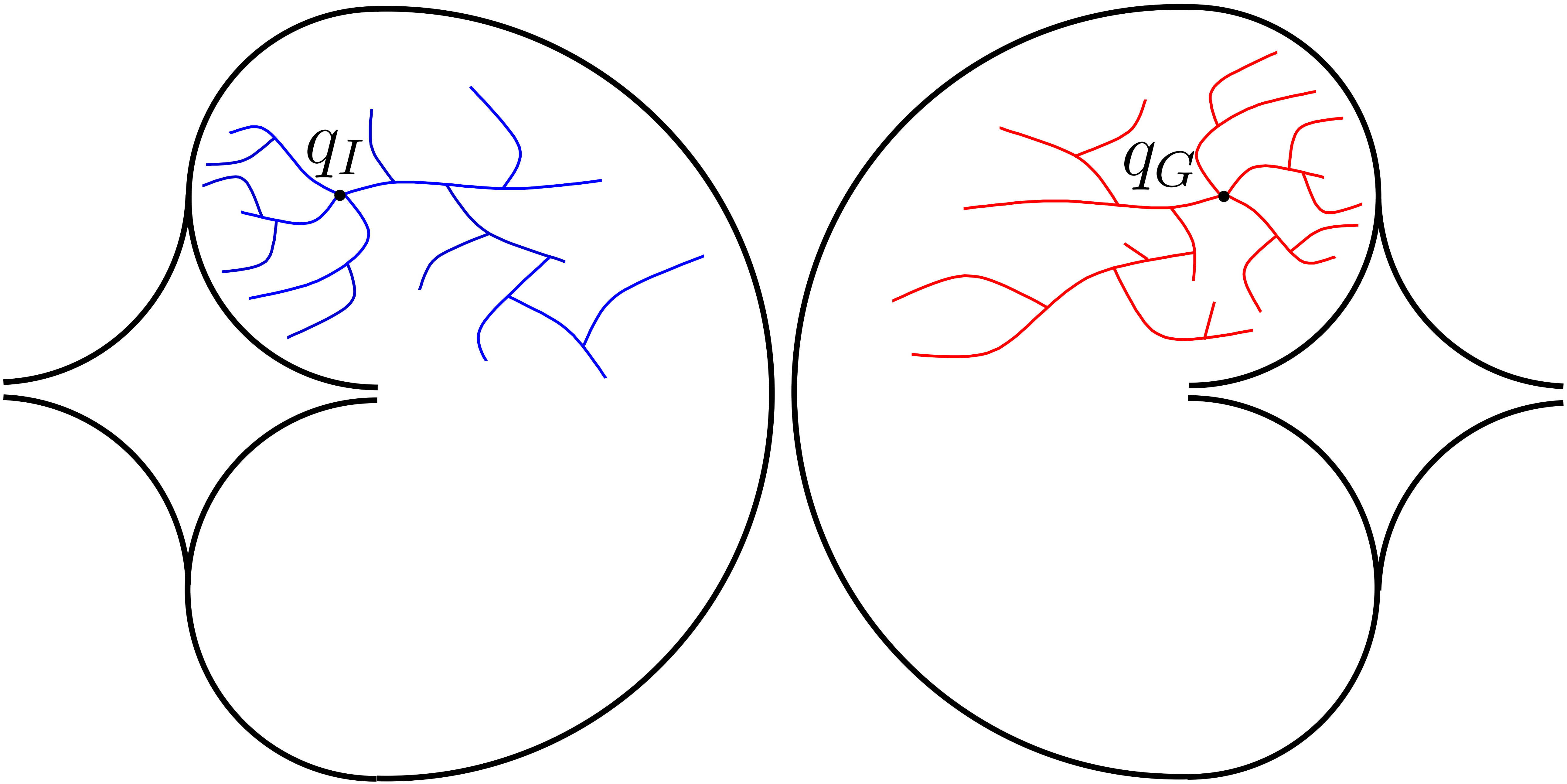


Multi-Directional Search



LaValle: Planning Algorithms, 2006

Some Problems are Simply Difficult



LaVallee: Planning Algorithms, 2006

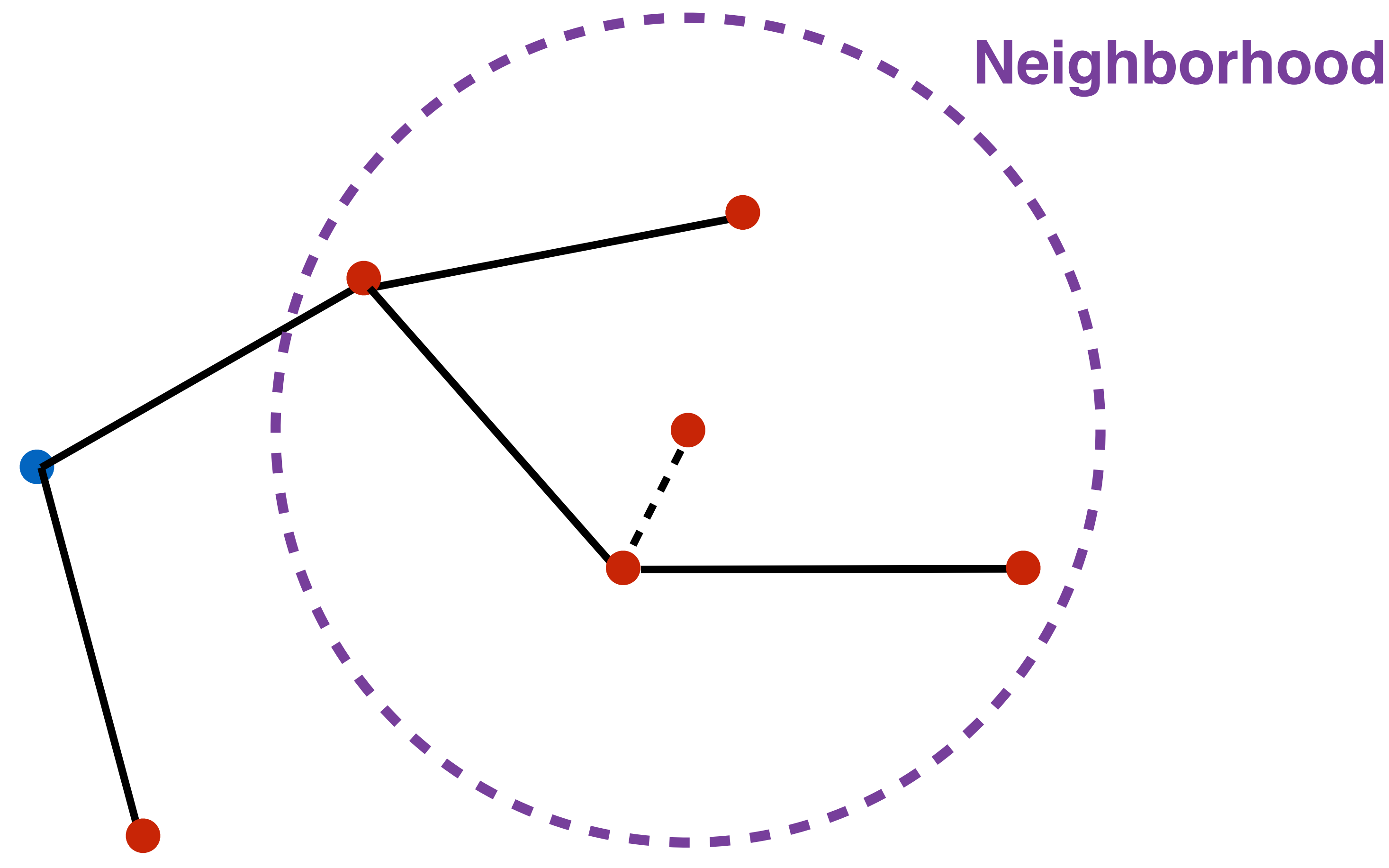
RRT*

- RRT sacrifices optimality of paths for efficiency
- Want to maintain tree structure and have short paths

- RRT* updates local neighborhoods
  - Attach new vertex to the neighbour with shortest path
  - Rewire local vertices' parents to create shortest paths

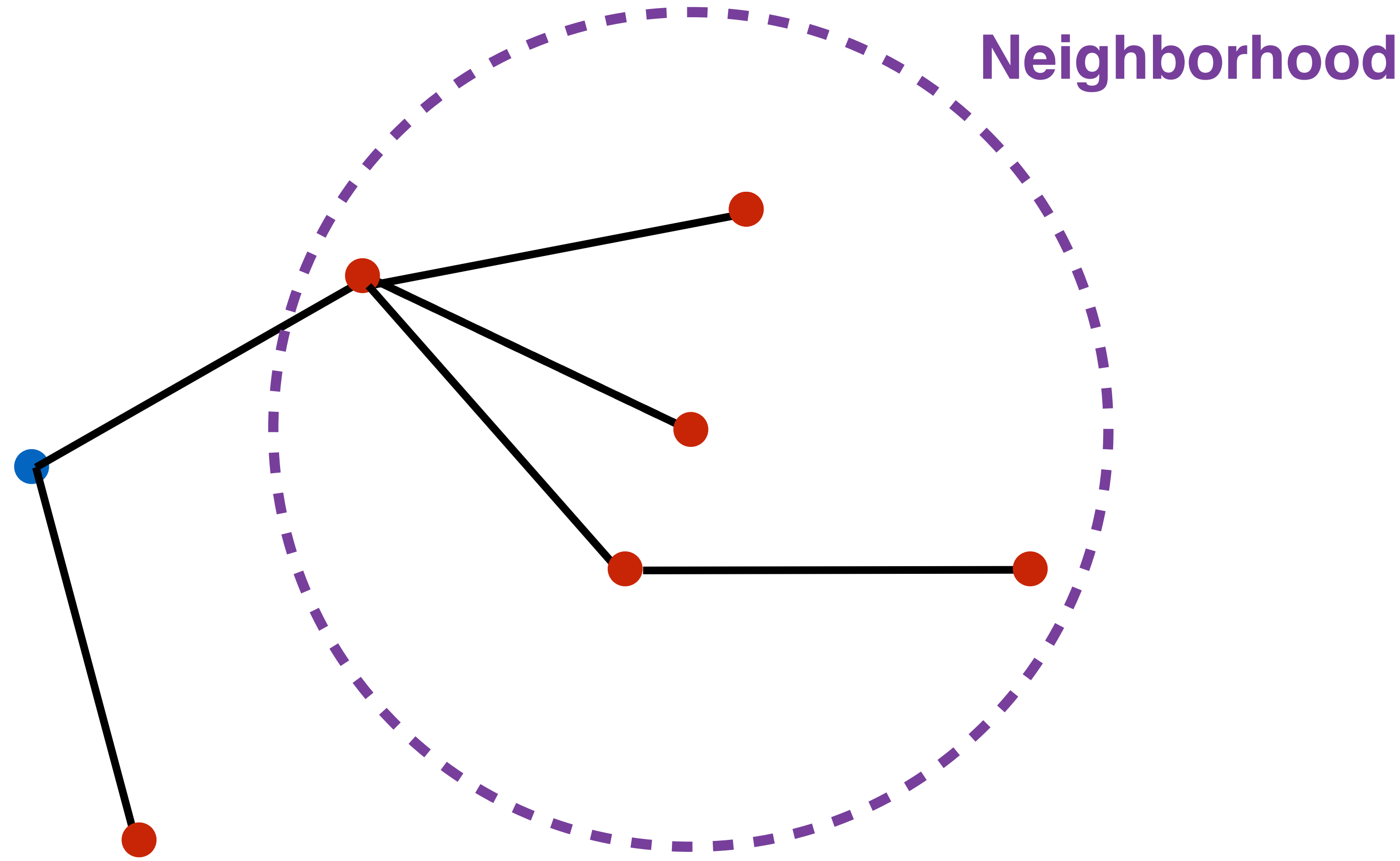
RRT*

Start by sampling new vertex as before



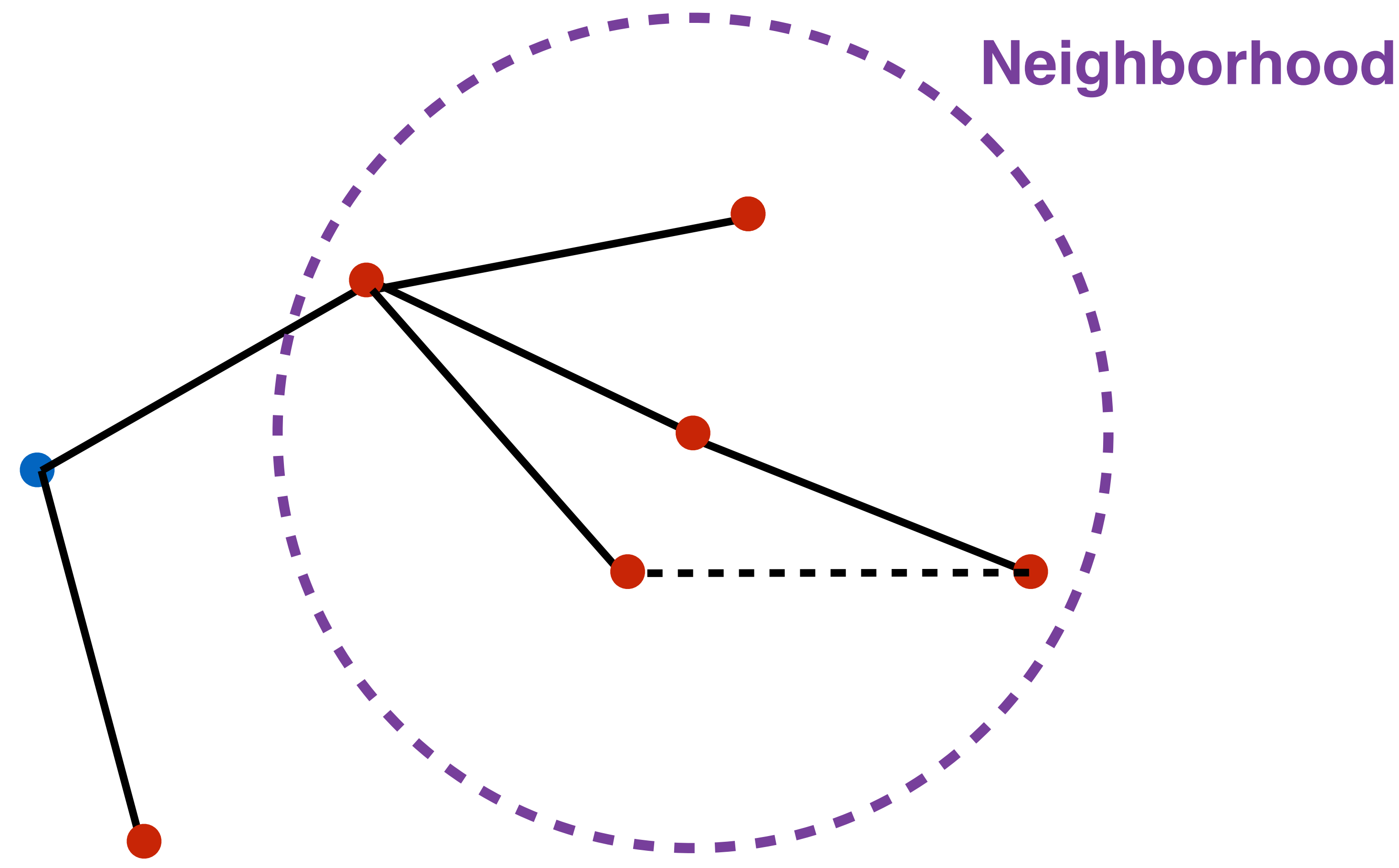
RRT*

Connect to neighbour with shortest path to root



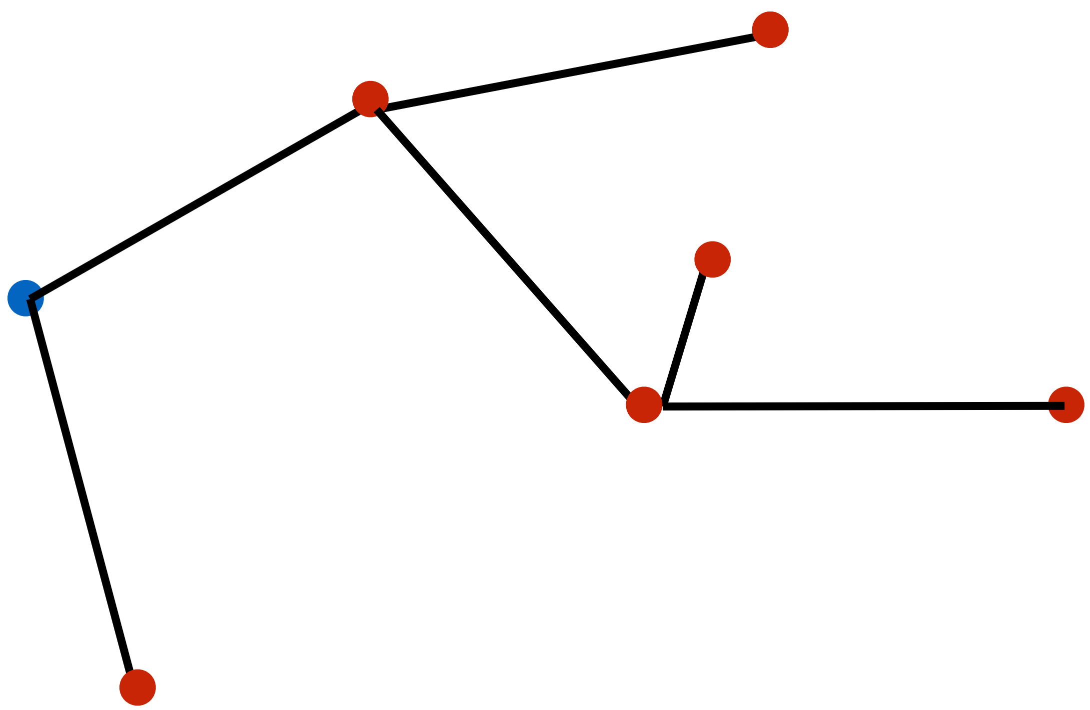
RRT*

Rewire neighbours to get shortest paths to root

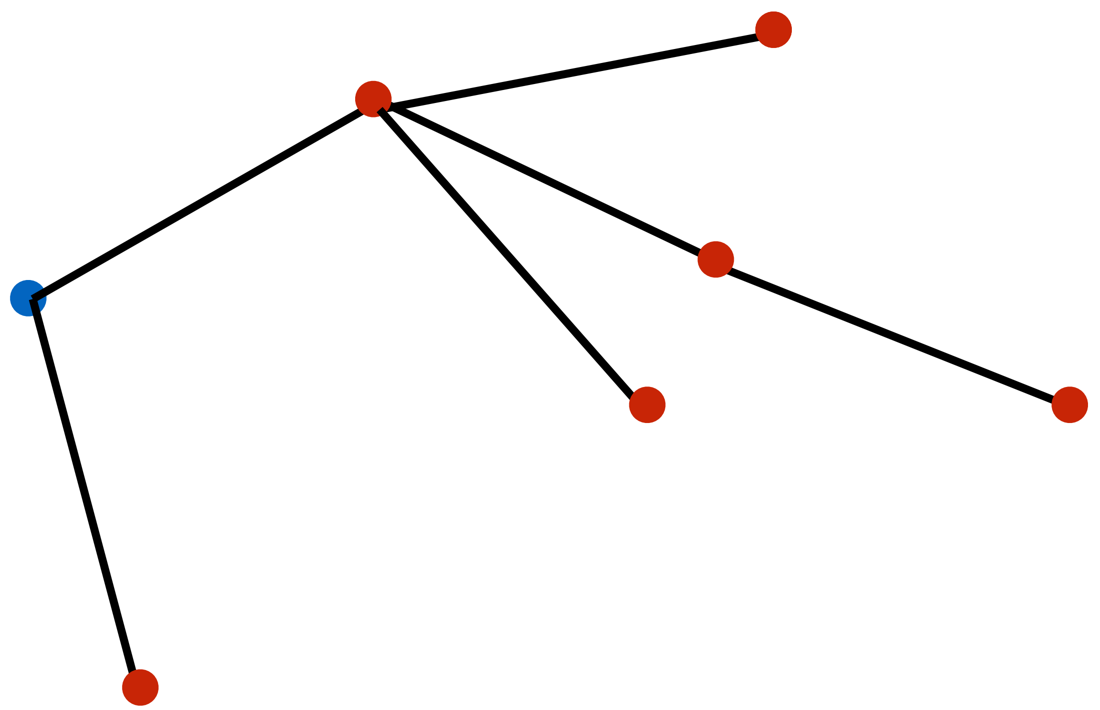


RRT vs RRT*

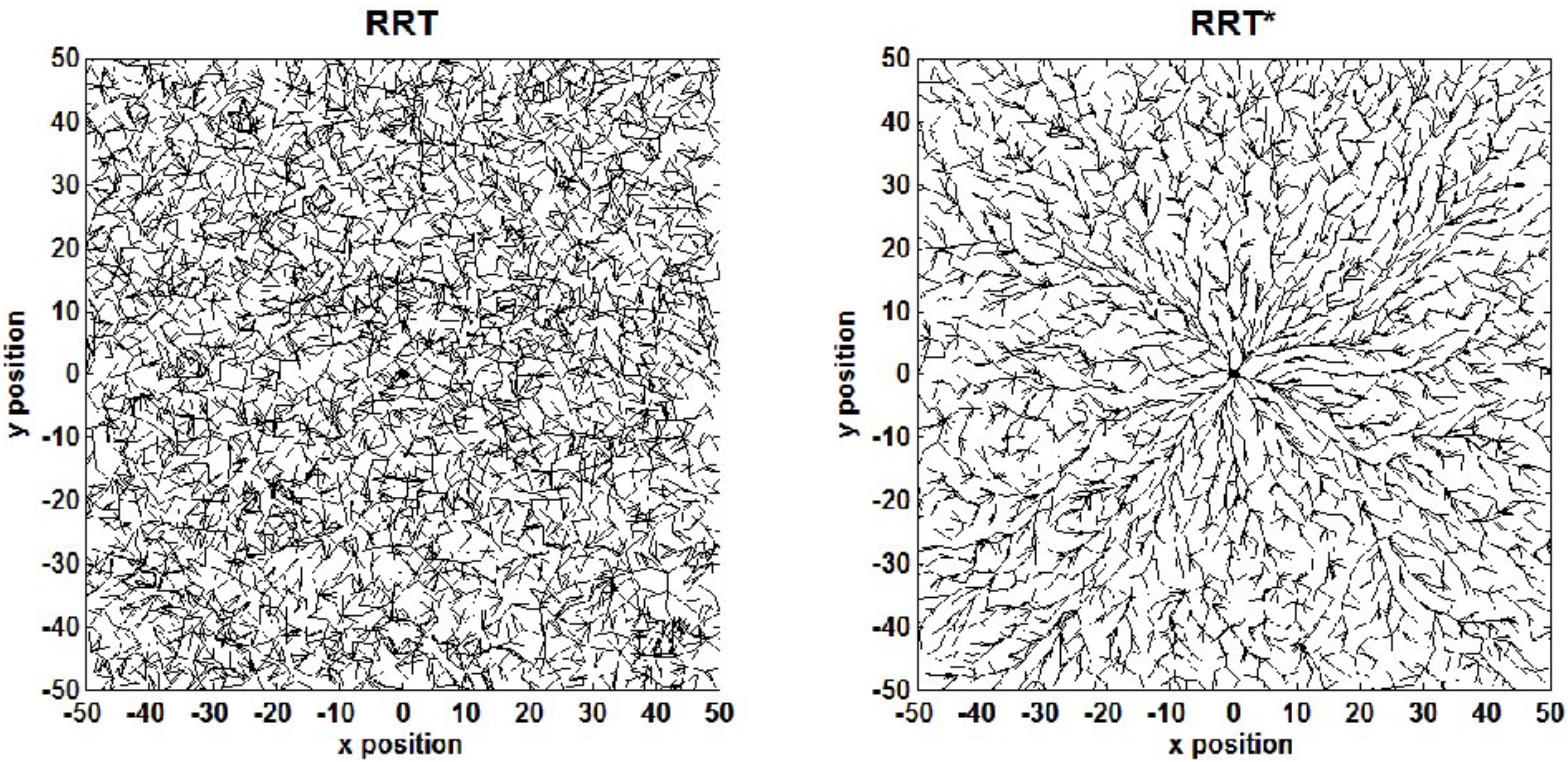
RRT



RRT*



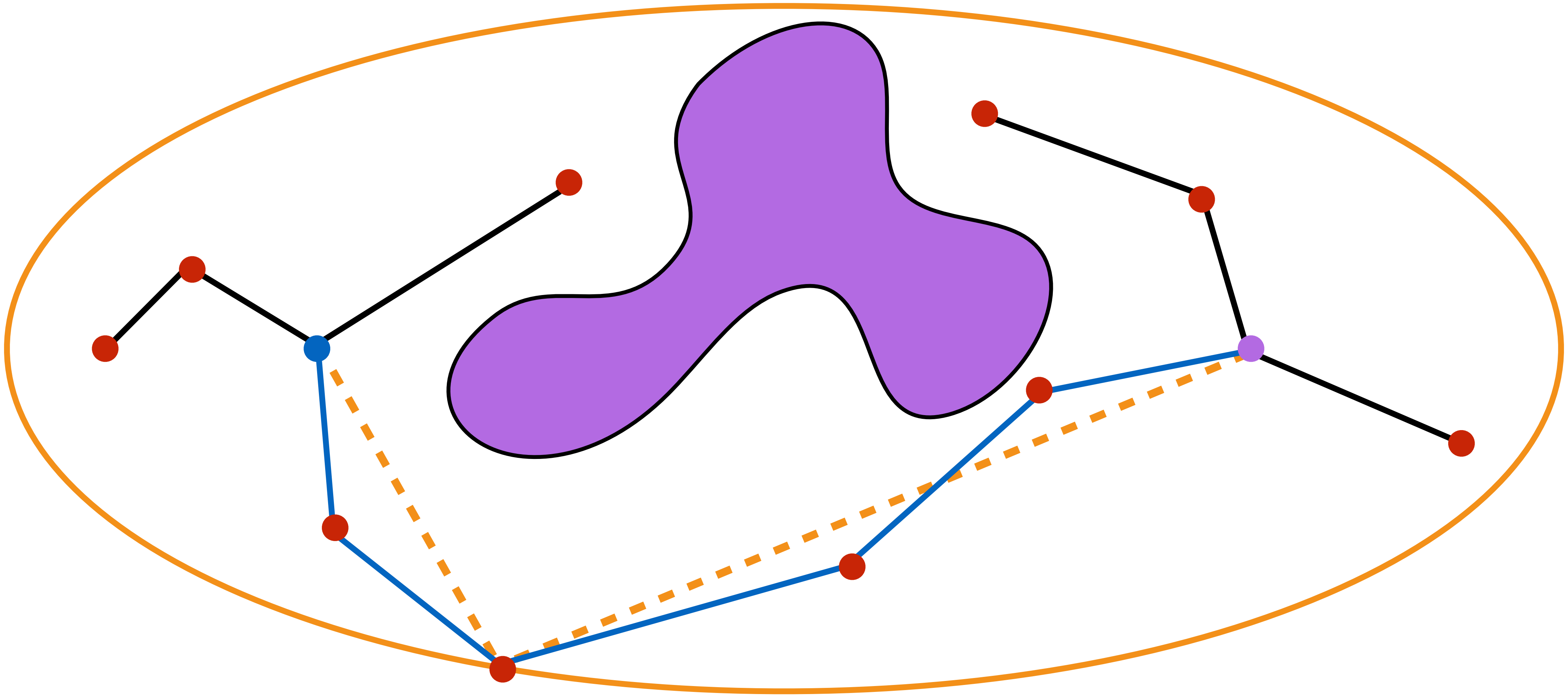
RRT vs RRT*



i.ytimg.com

Incremental Densification

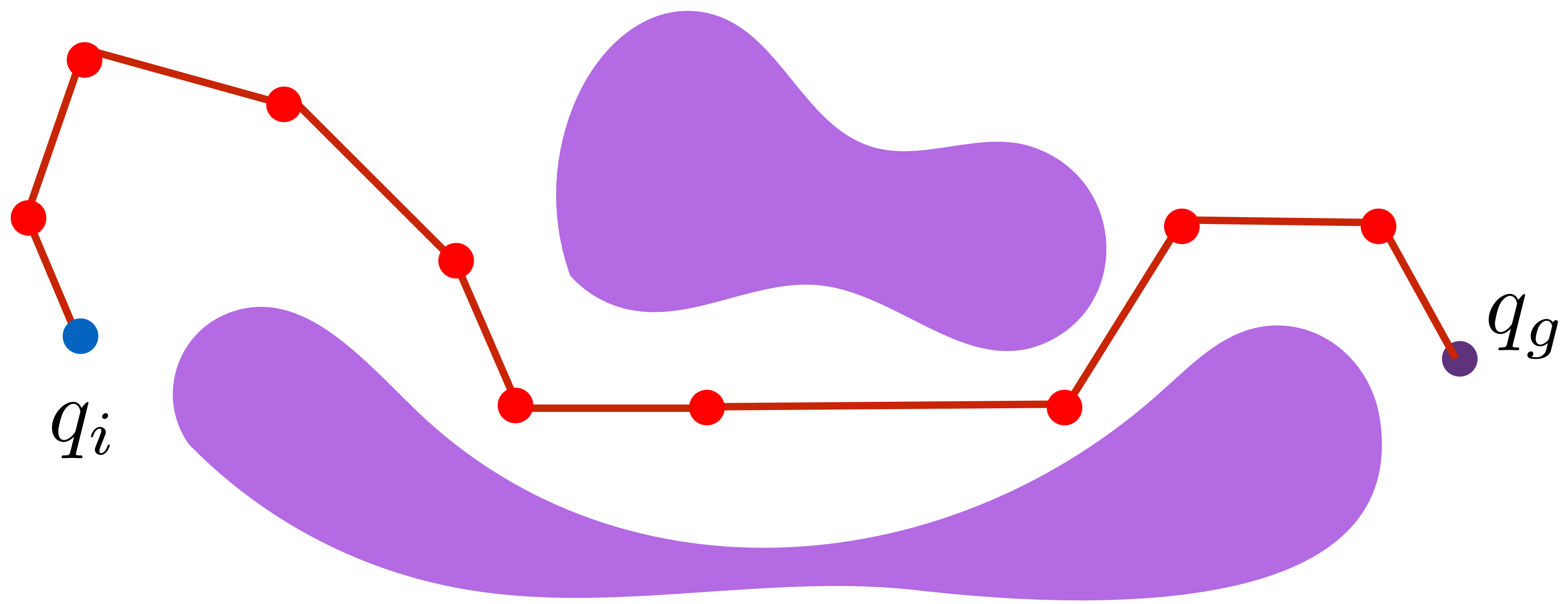
- More vertices are more likely to result in shorter paths
- Unlikely that shorter path will have vertices further away



- Sample additional vertices only within the ellipse

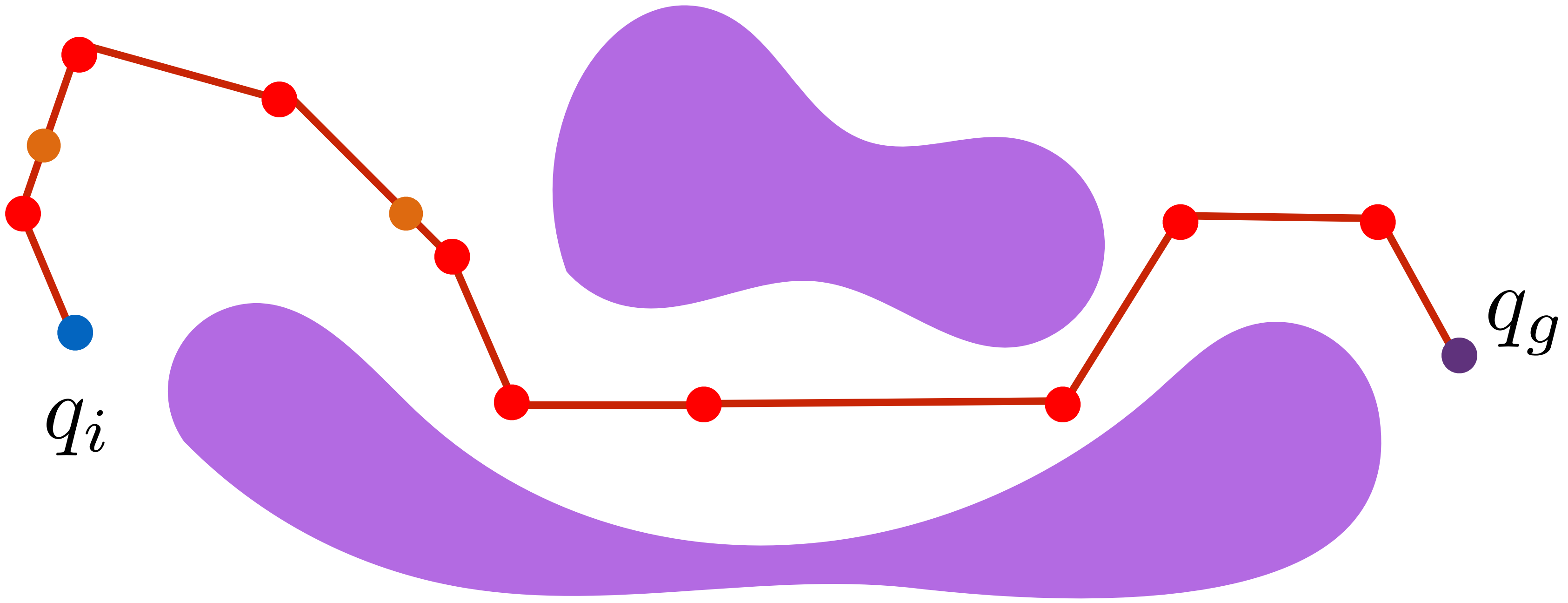
Path Shortening

Given a shortest path



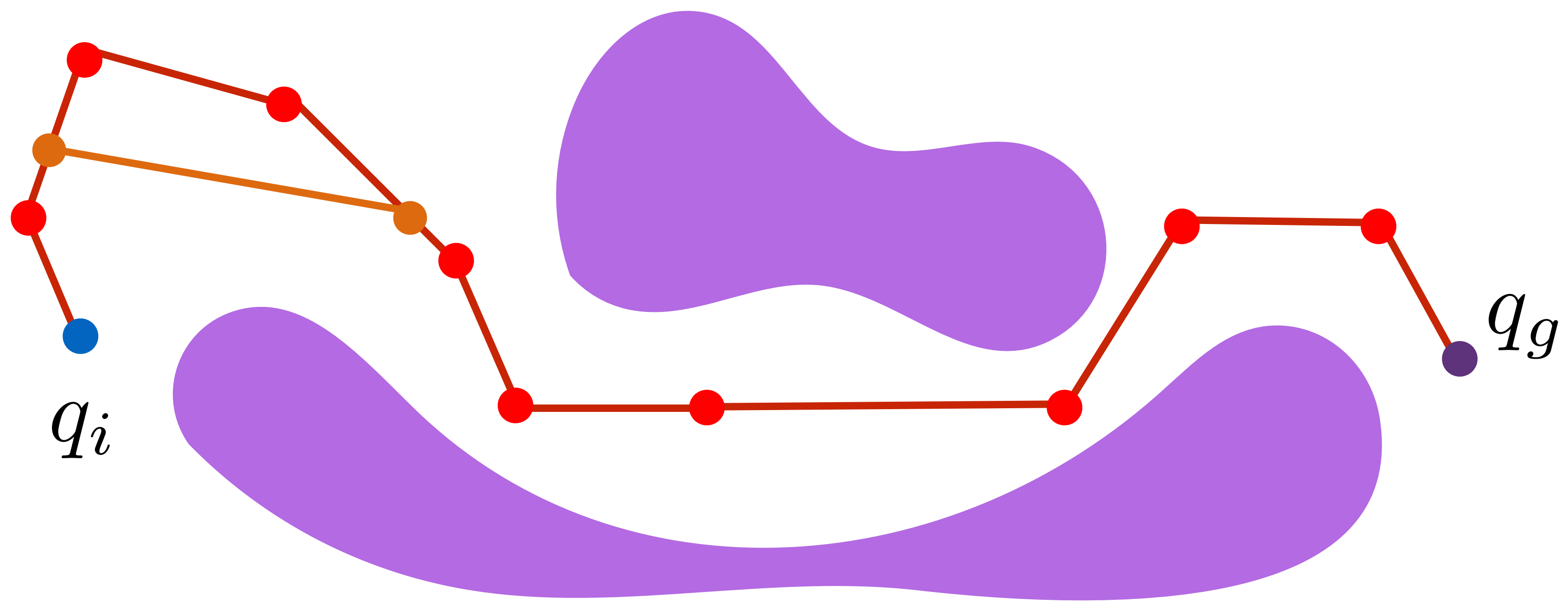
Path Shortening

Sample two points along the path



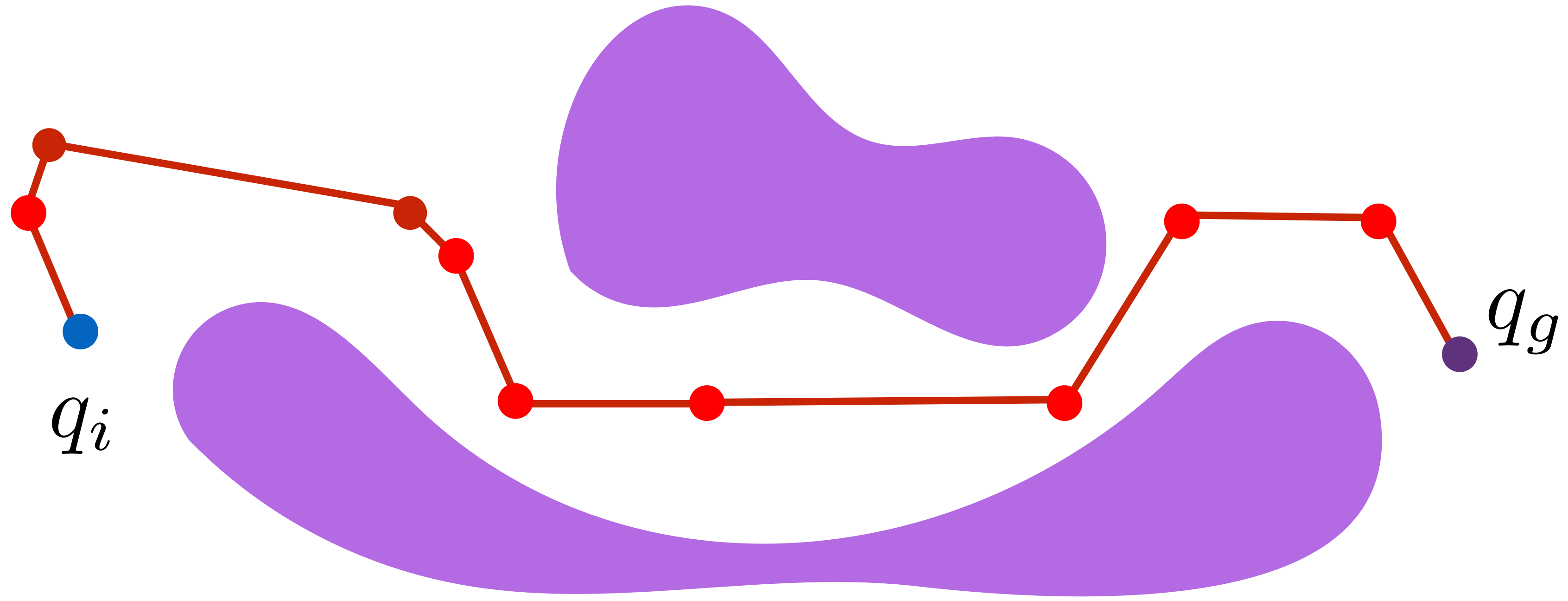
Path Shortening

Attempt to connect points



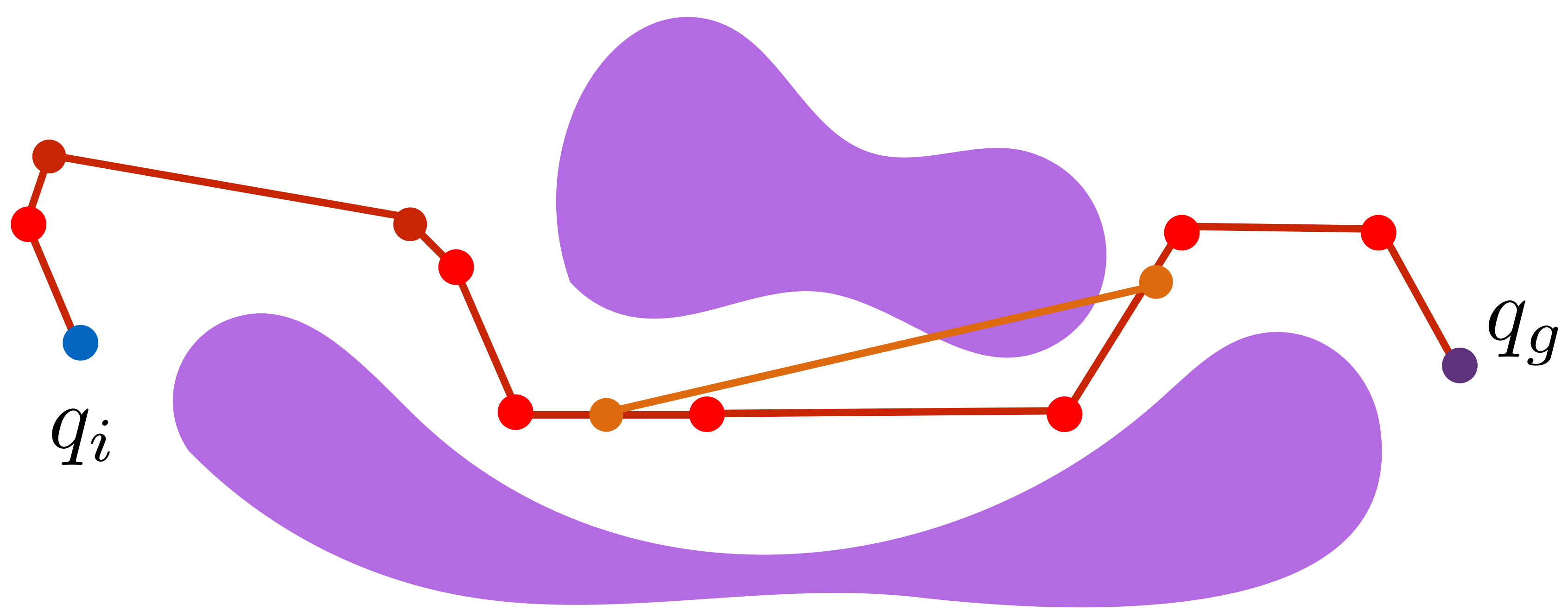
Path Shortening

Remove longer path segment



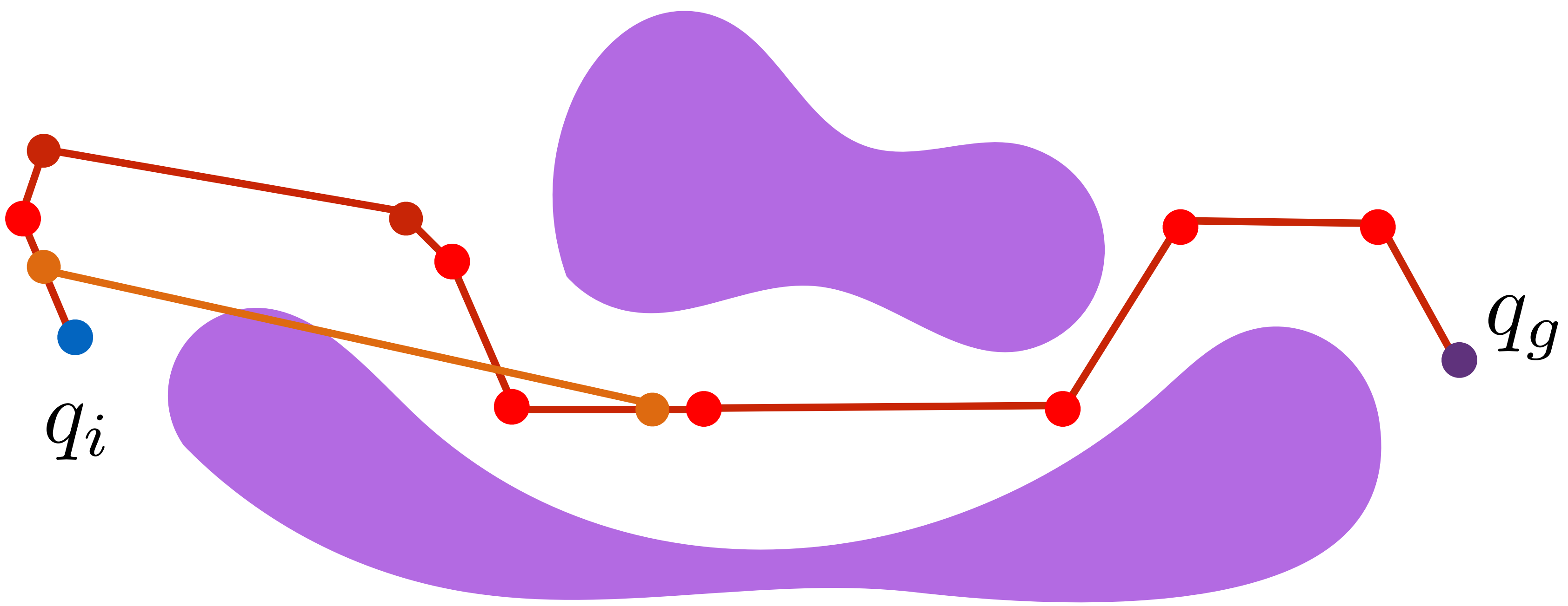
Path Shortening

Sample two points along the path and attempt to connect



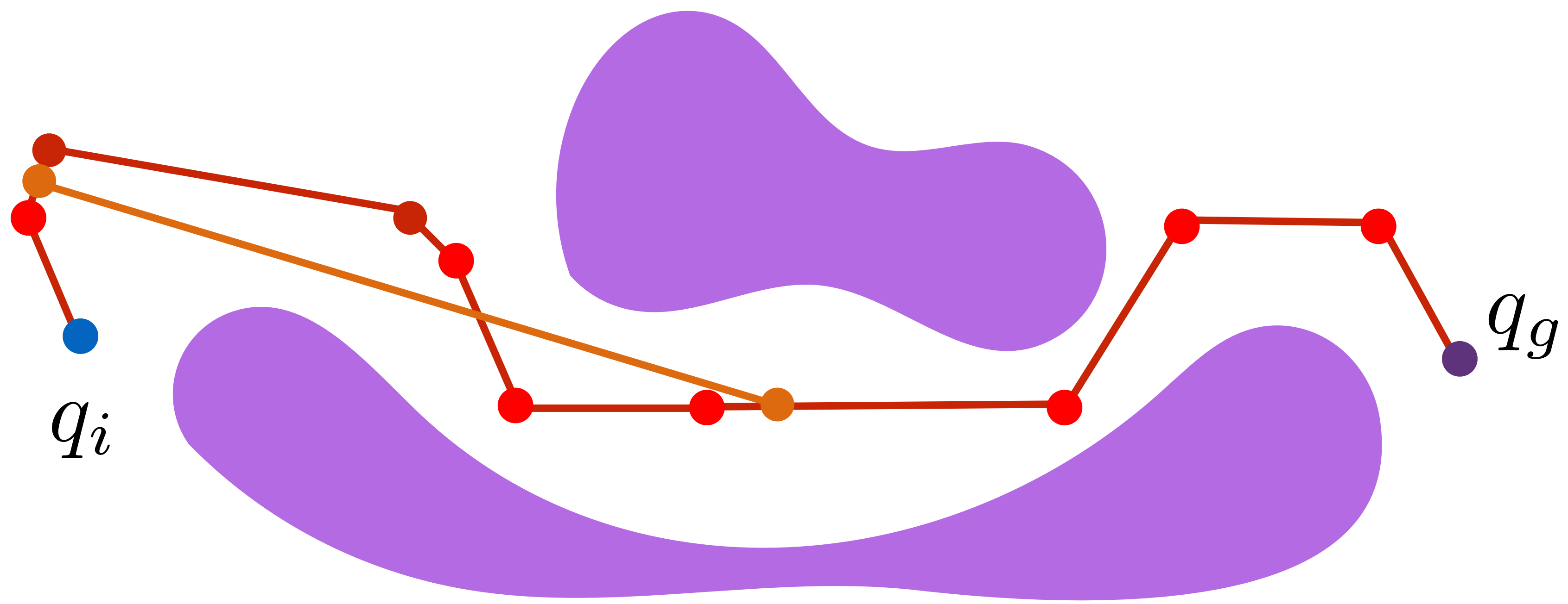
Path Shortening

Sample two points along the path and attempt to connect



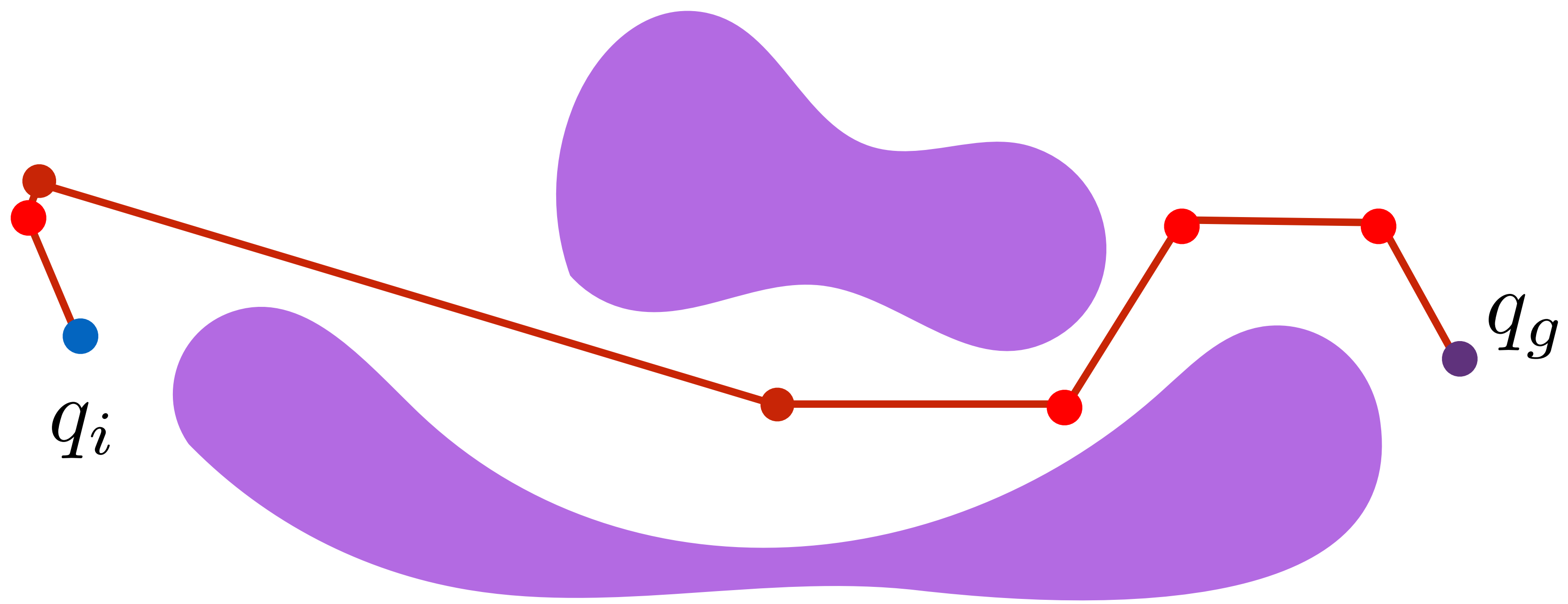
Path Shortening

Sample two points along the path and attempt to connect



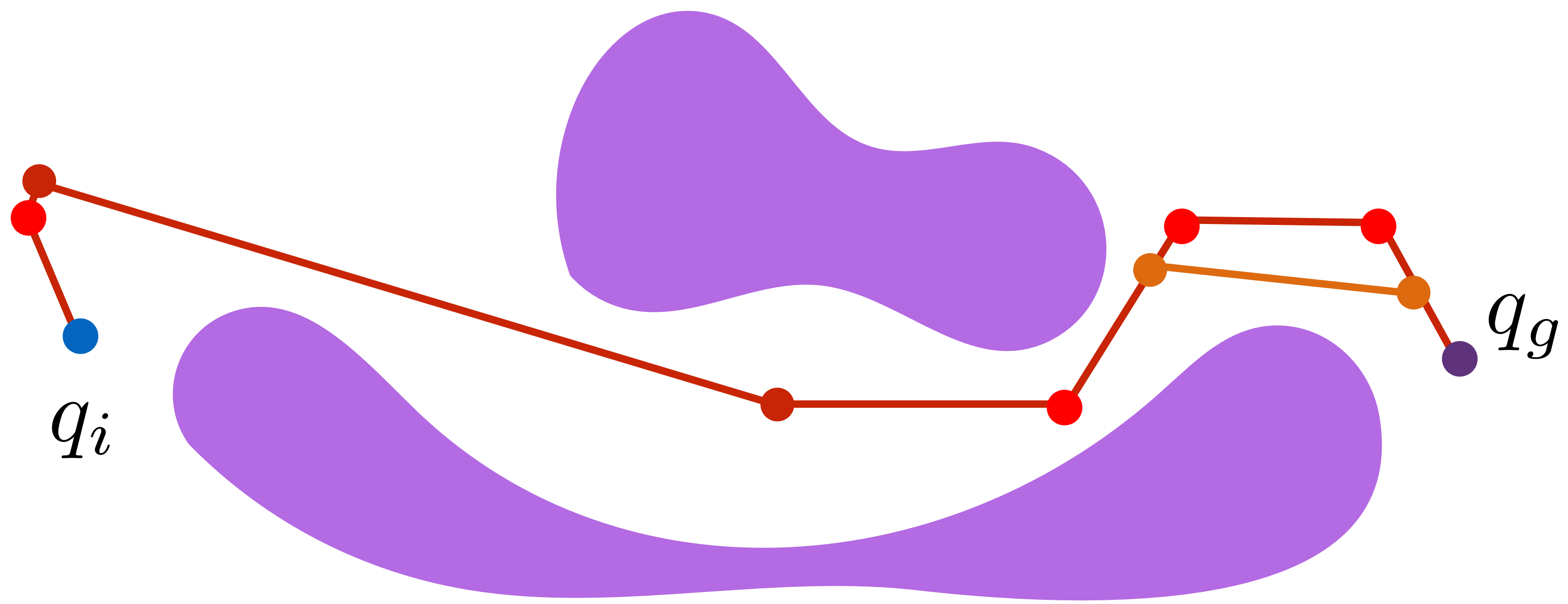
Path Shortening

Sample two points along the path and attempt to connect



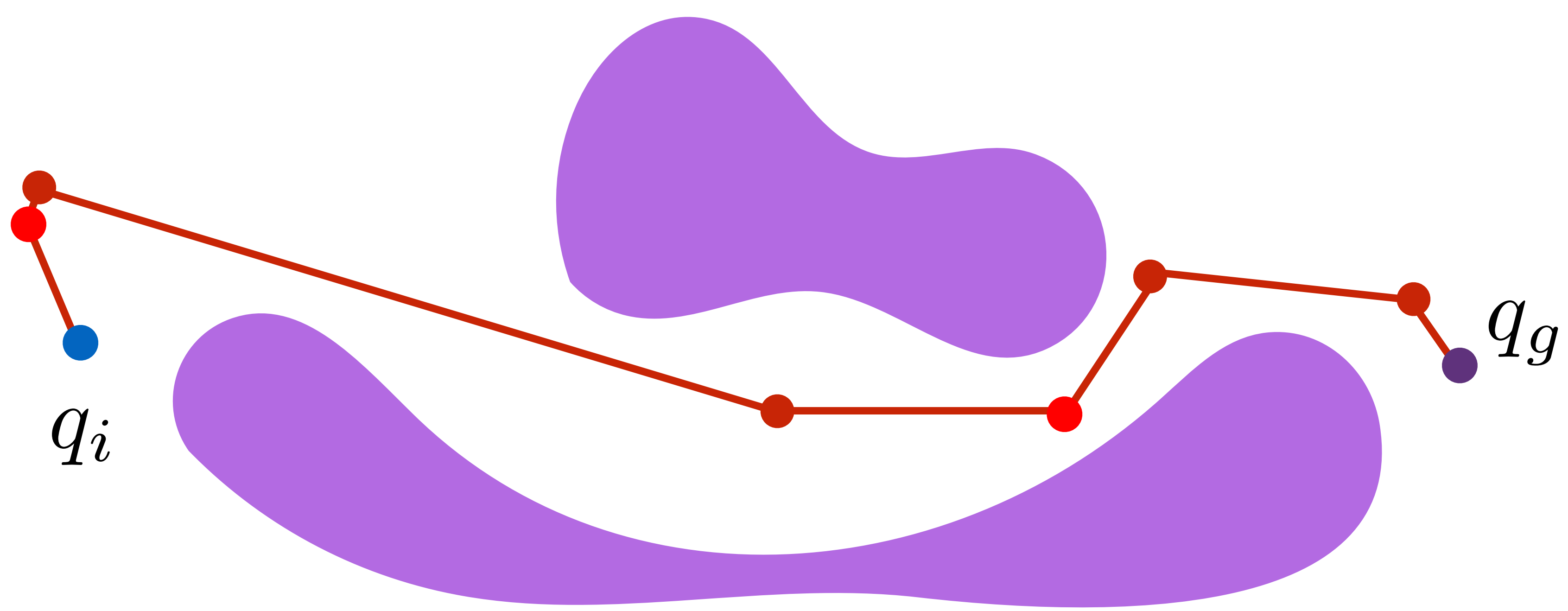
Path Shortening

Sample two points along the path and attempt to connect



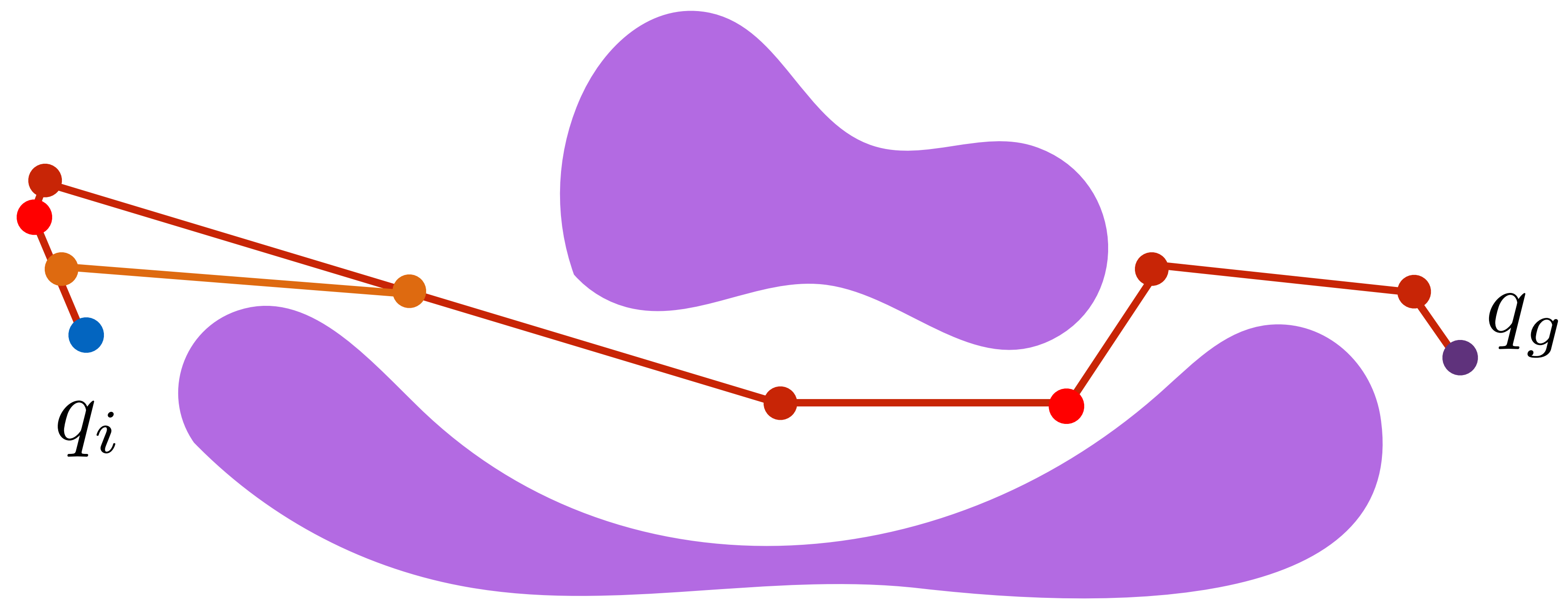
Path Shortening

Sample two points along the path and attempt to connect



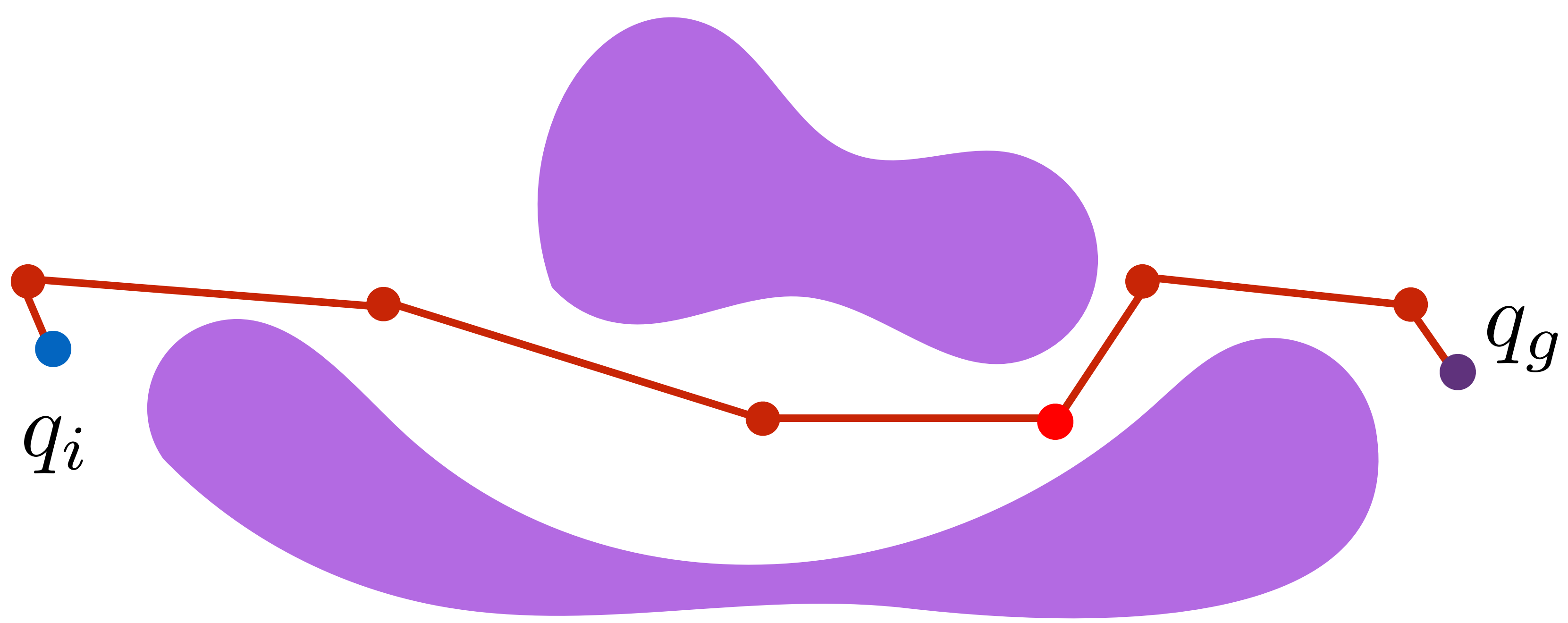
Path Shortening

Sample two points along the path and attempt to connect



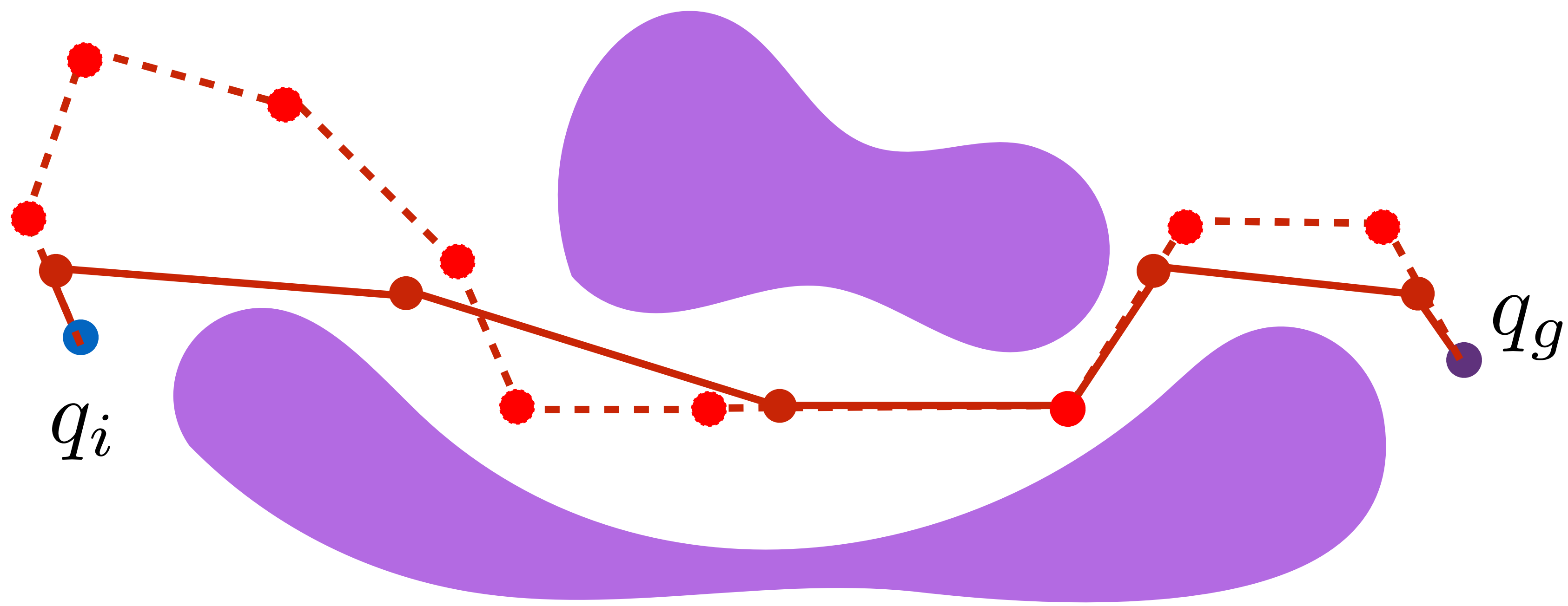
Path Shortening

Sample two points along the path and attempt to connect



Path Shortening

Straight line connections ensure full path is shorter



Sample-Based Motion Planning

- Multiple queries: PRM
- Single query: RRT

- Algorithmically simple
- Highly explorative
- Applicable to high-dimensional spaces
- Probabilistic completeness

Questions?