Robot Autonomy

Lecture 10: Planning with Costs

Oliver Kroemer

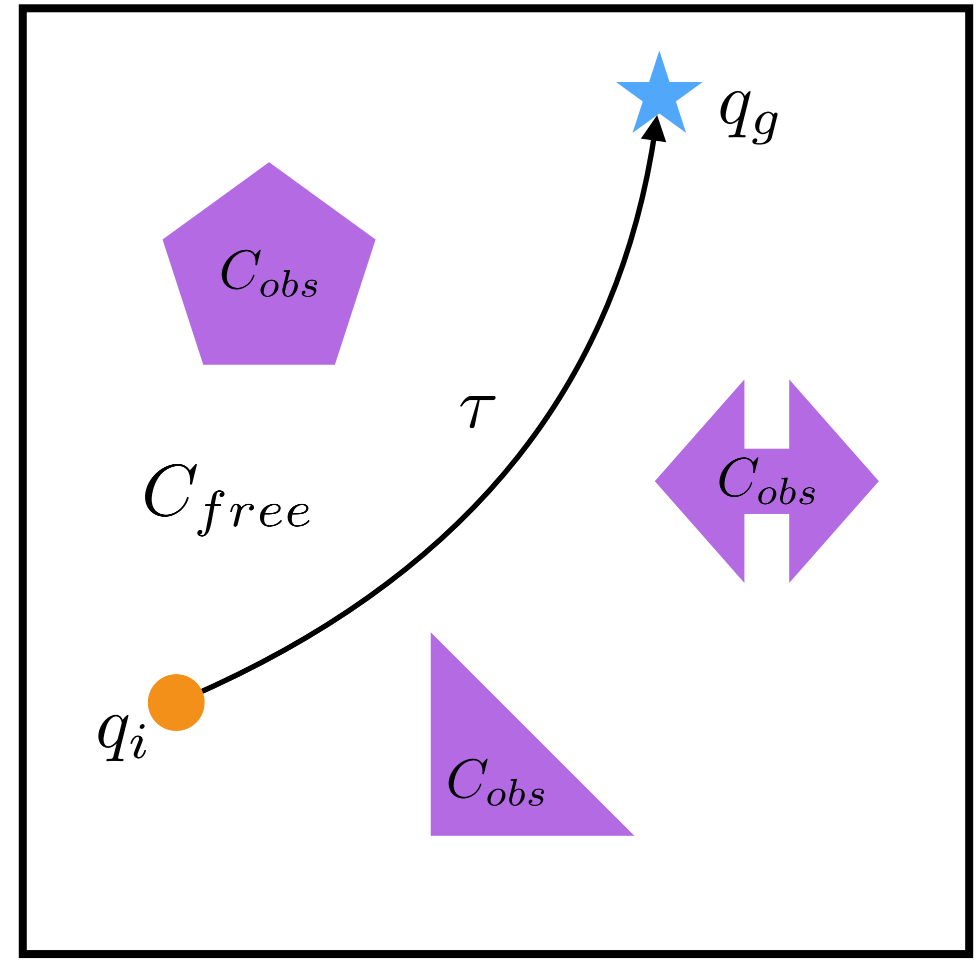
Based on lecture boards of Siddhartha Srinivasa

- Cost of a trajectory

$$C(\tau) = \int_0^1 c(\tau(s)) \left| \frac{d\tau}{ds} \right| ds$$

configuration cost

reparameterization invariant



- Goal is to minimize cost

$$\tau^* = \arg\min_{\tau \in H} C(\tau)$$

- Shortest feasible path

$$c(q) = 1 \text{ if } q \in C_{free}$$

$$c(q) = \infty \text{ if } q \in C_{obs}$$

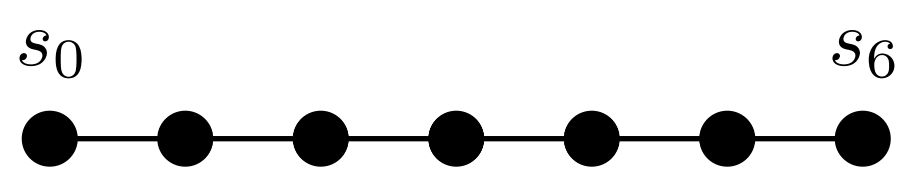
arc/path length

$$C(\tau) = \int_0^1 \left| \frac{d\tau}{ds} \right| ds$$

Discretized Costs for Trajectories

- Approximate trajectory by a set of equal spaced points

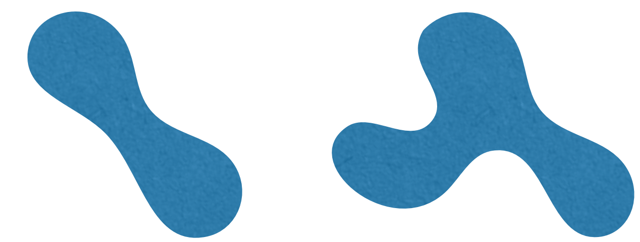
$$C(\tau) \approx \frac{L}{N} \sum_{i=0}^{N-1} c(\tau(s_i)) \qquad s_i = \frac{i}{N-1}$$



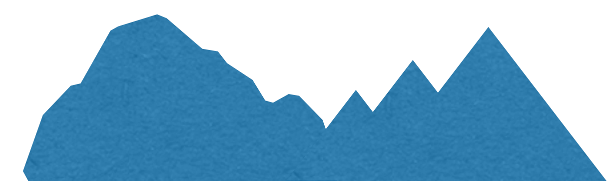
- Segment has cost of initial starting point in segment
- Can also have variable lengths for segments
  - Be careful of skipping over costly regions with large segments

Example Costs

- Distance to obstacles

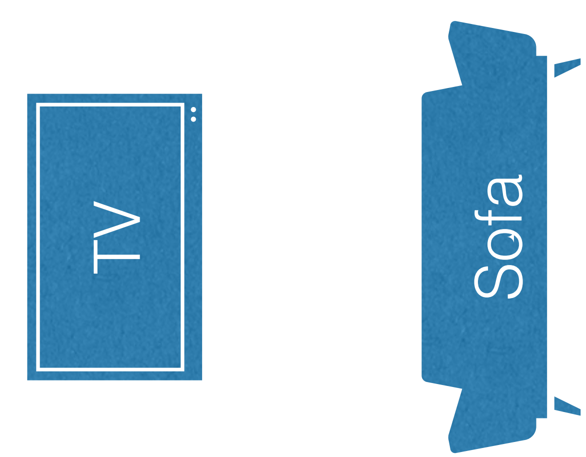


- Traversability



- Effort

- Human preferences



Time-dependent Trajectory Costs

- Time dependent trajectories may penalize derivatives

- Velocity

$$\dot{\tau}(t)$$

- Acceleration

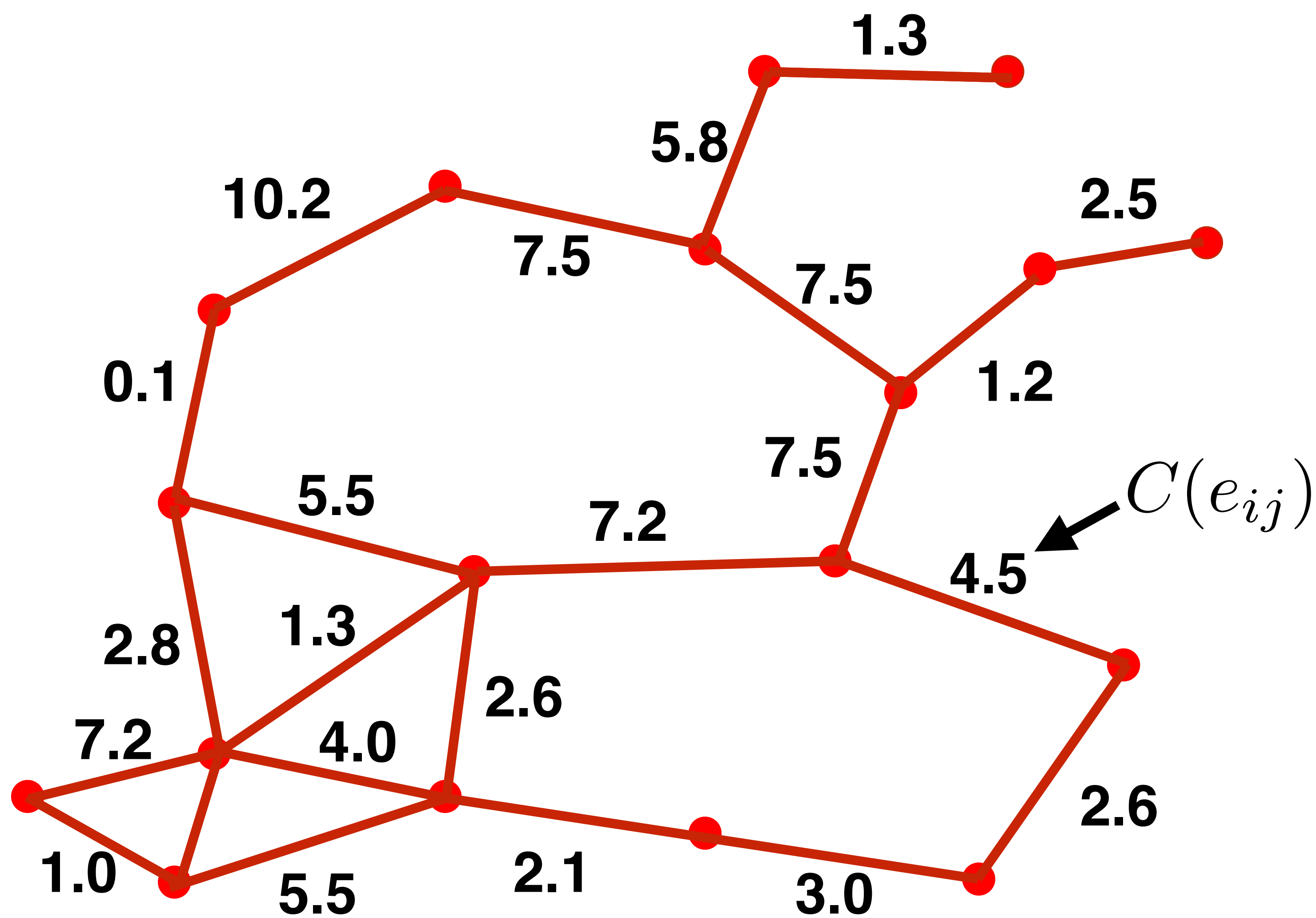
$$\ddot{\tau}(t)$$

- Jerk

$$\dddot{\tau}(t)$$

PRM Lowest Cost

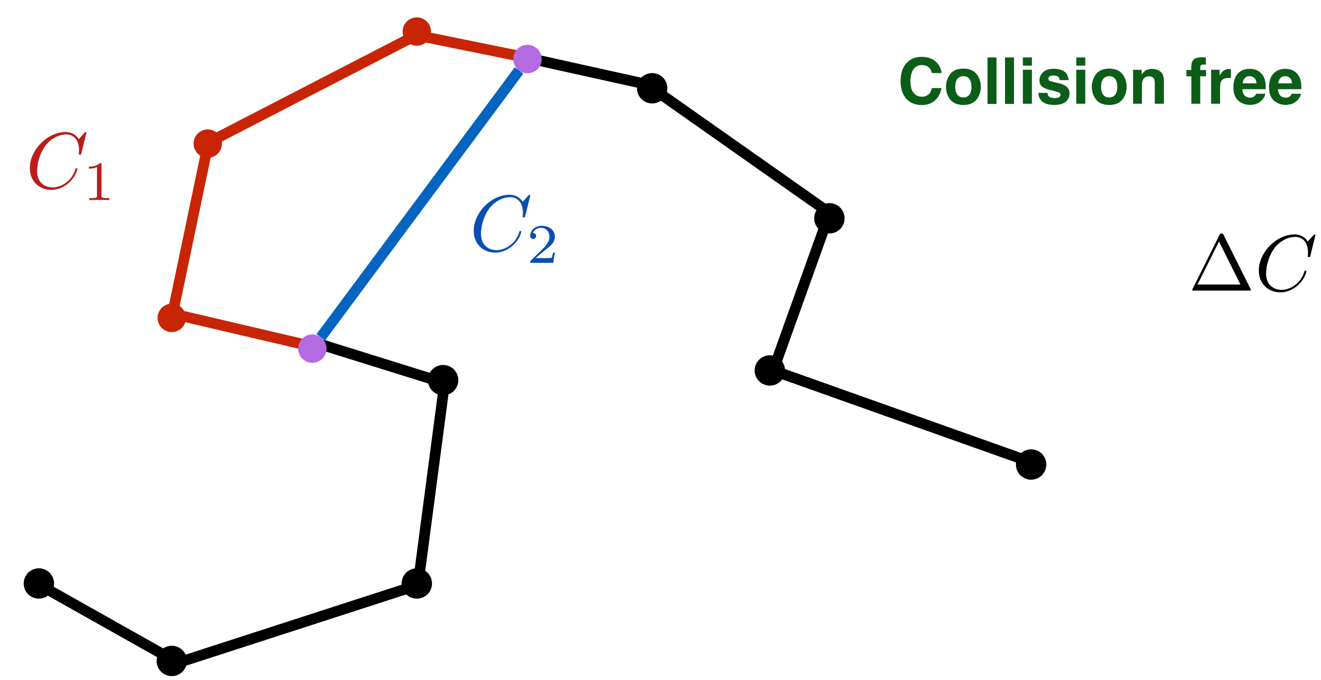
- Compute cost for each edge of PRM



- Search for path with minimum cost (discrete search)
- Limited to paths along PRM (could potentially do better)

Path Shortening with Costs

- Assume we have an initial trajectory to goal (PRM,RRT)
- How can we improve the trajectory?



$$\Delta C = C_2 - C_1$$

- Select two points along the trajectory
- Create straight connection and compute corresponding cost
- Select new path with probability $p(\Delta C)$

Acceptance Probability

- Always accept if cost is reduced:

$$p(\Delta C) = 1 \text{ if } \Delta C < 0$$

- If cost is increased: $\Delta C > 0$
  - Hill climbing

$$p(\Delta C) = 0$$

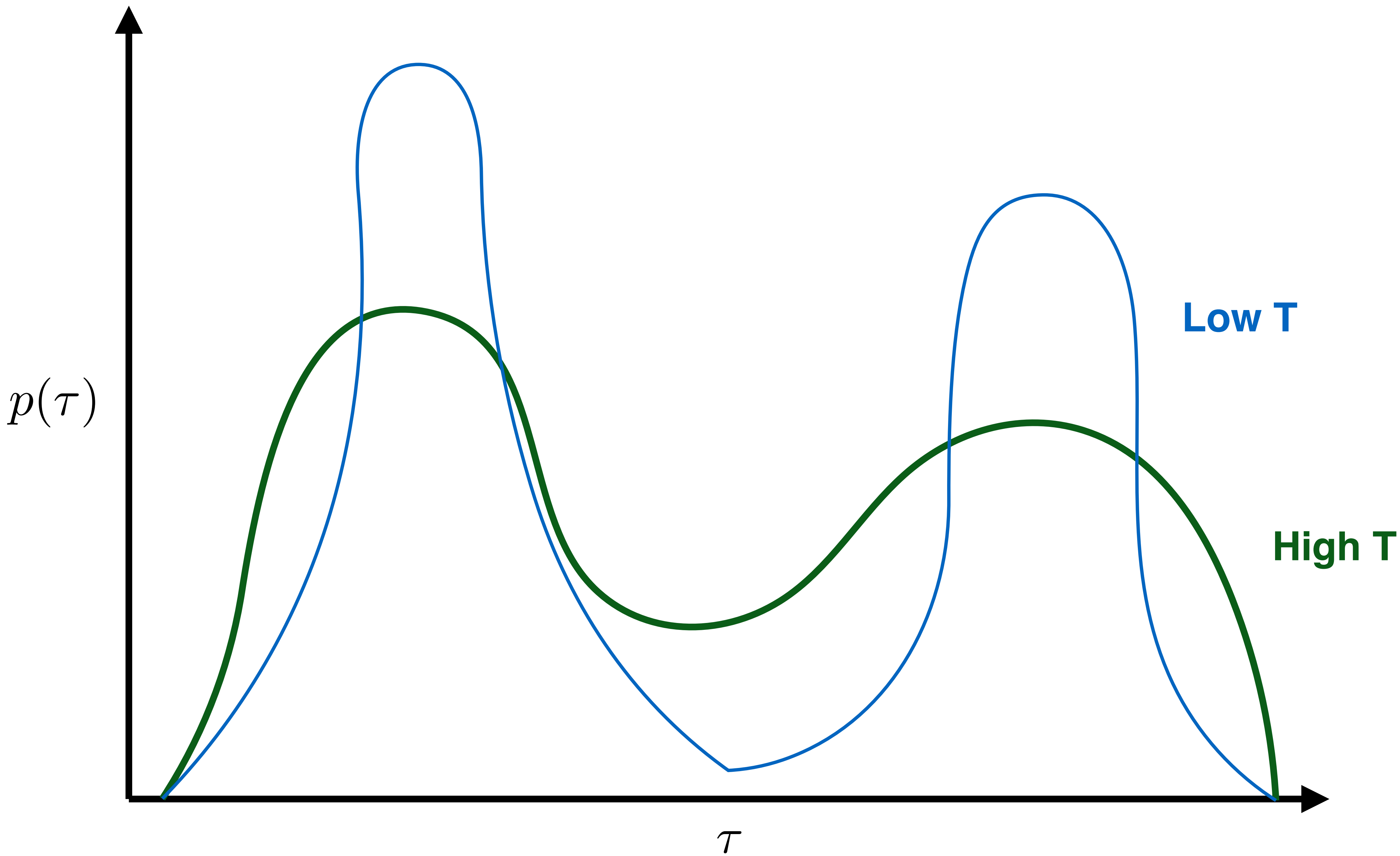
  - Metropolis Hasting

$$p(\Delta C) = \exp(-\Delta C / T) \text{ with } T \text{ fixed}$$

  - Simulated Annealing

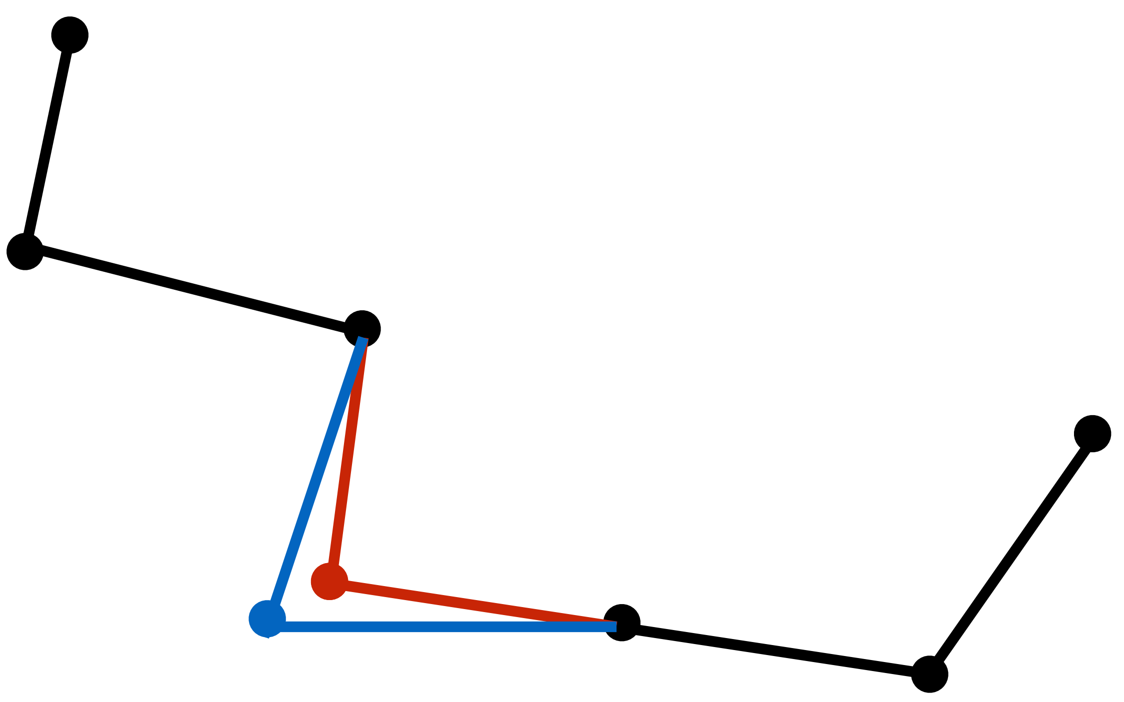
$$p(\Delta C) = \exp(-\Delta C / T) \text{ with } T \text{ decreasing}$$

Metropolis-Hasting (MCMC) Sampling



Optimizing Point Locations

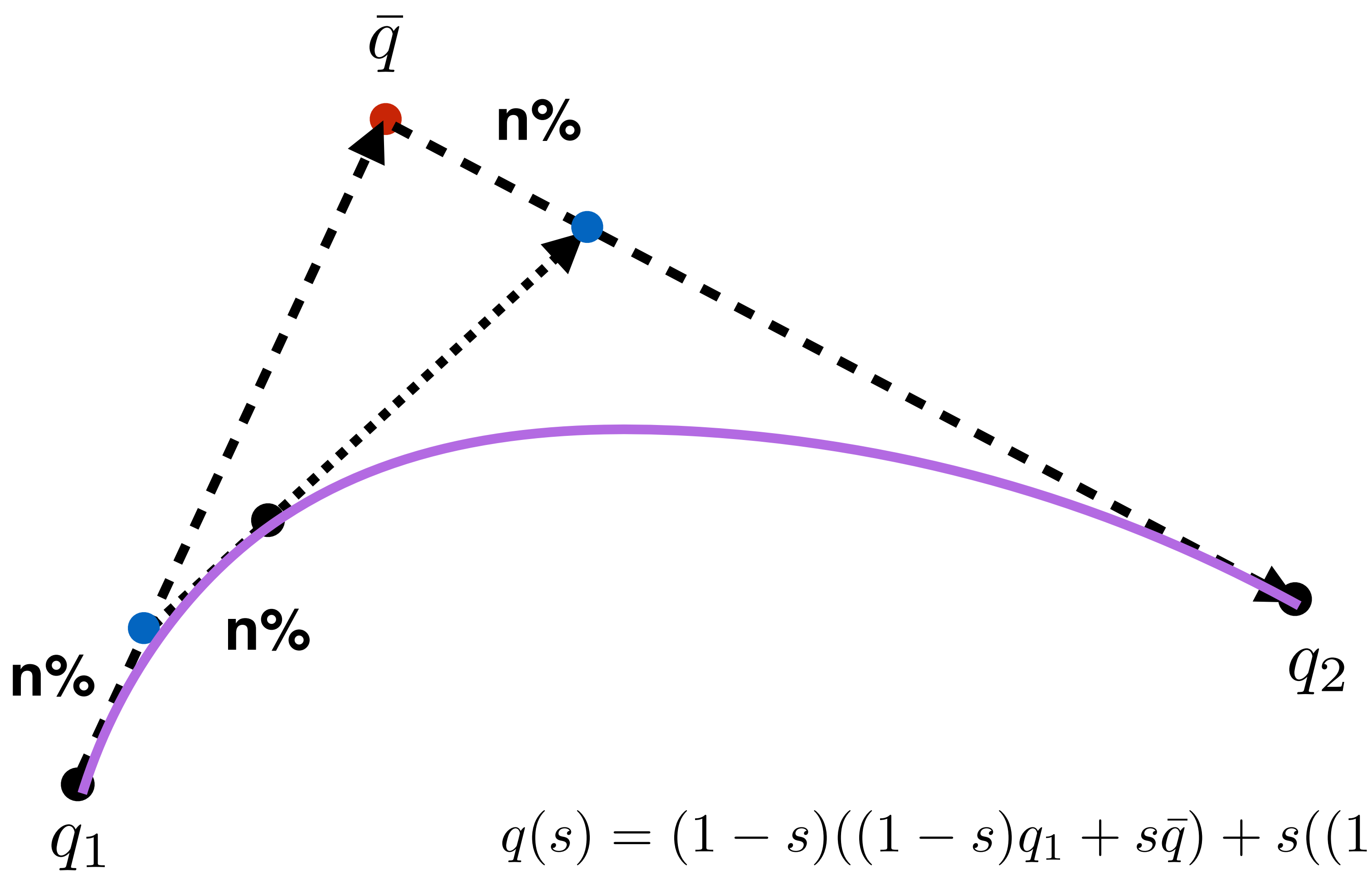
- Can also shift points locally to attempt to improve cost



  - Locally sample shifted point, e.g., Gaussian
  - Shift individual point or multiple points at a time
- What about smooth curves for trajectories?

Bezier Curves

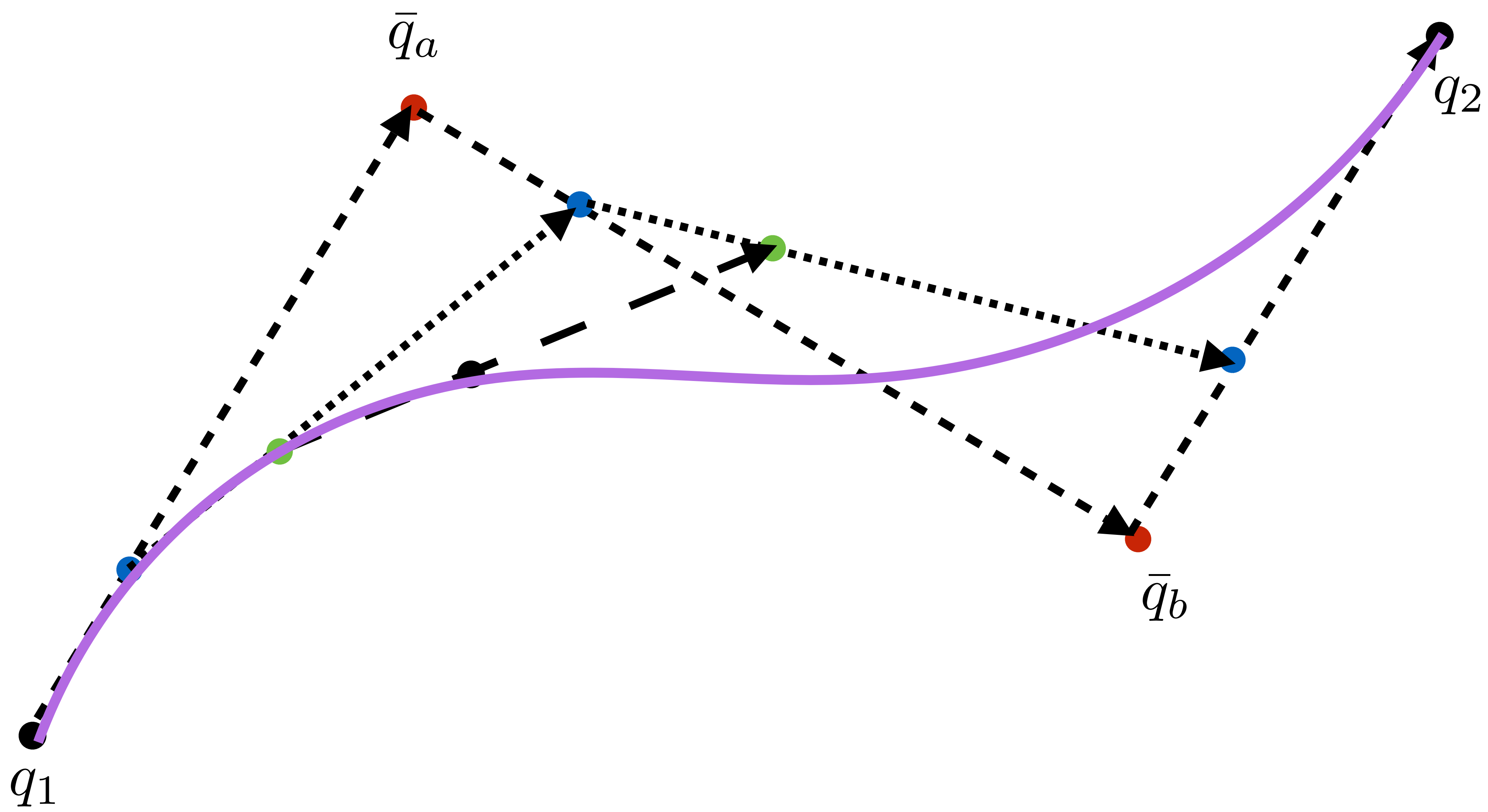
- Parameterized curves between points (e.g., spline)



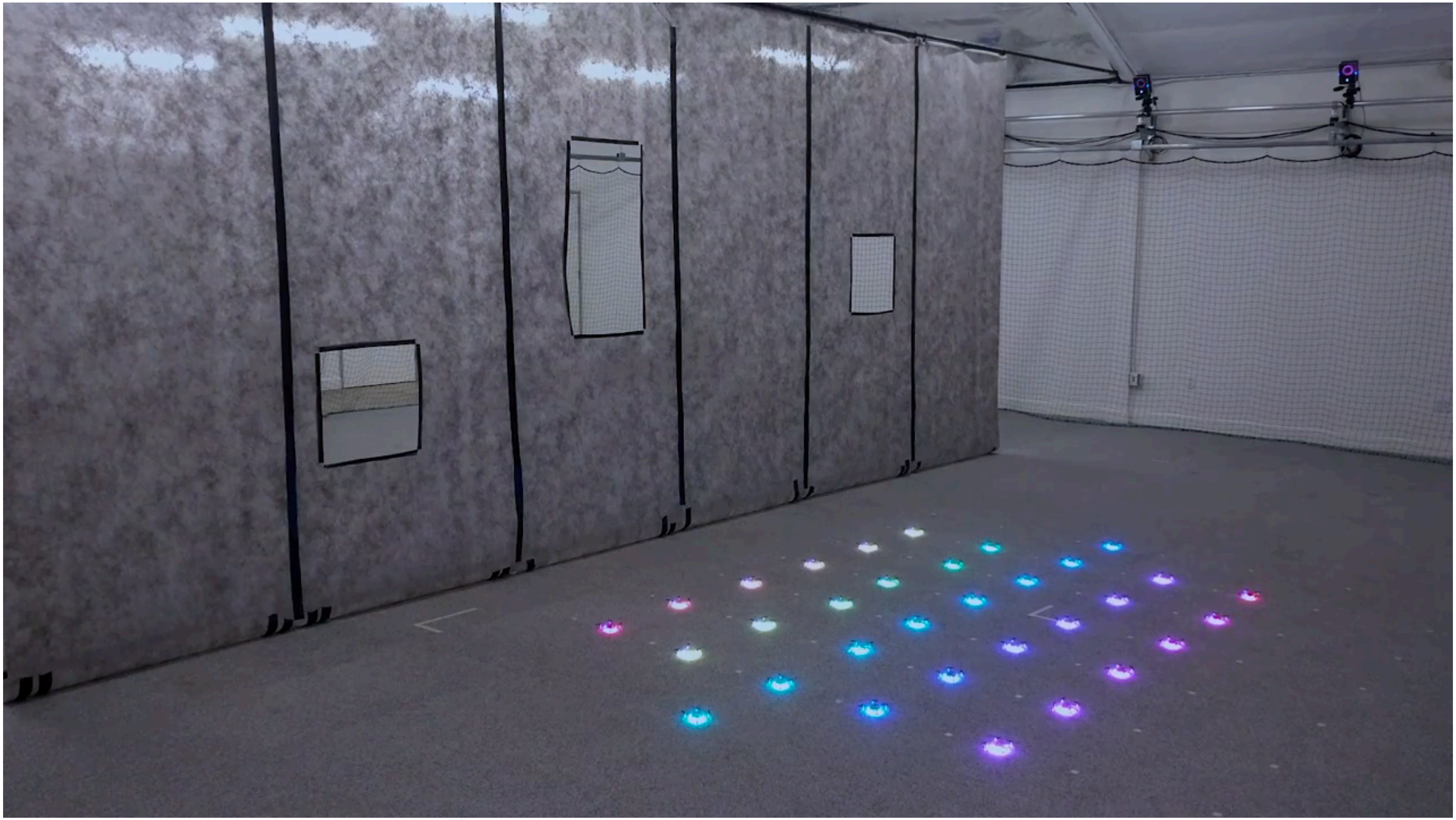
$$q(s) = (1-s)((1-s)q_1 + s\bar{q}) + s((1-s)\bar{q} + sq_2)$$

  - Shape of curve defined by control points
  - Optimize locations of control points
  - Curve within convex hull of control points

Bezier Curves



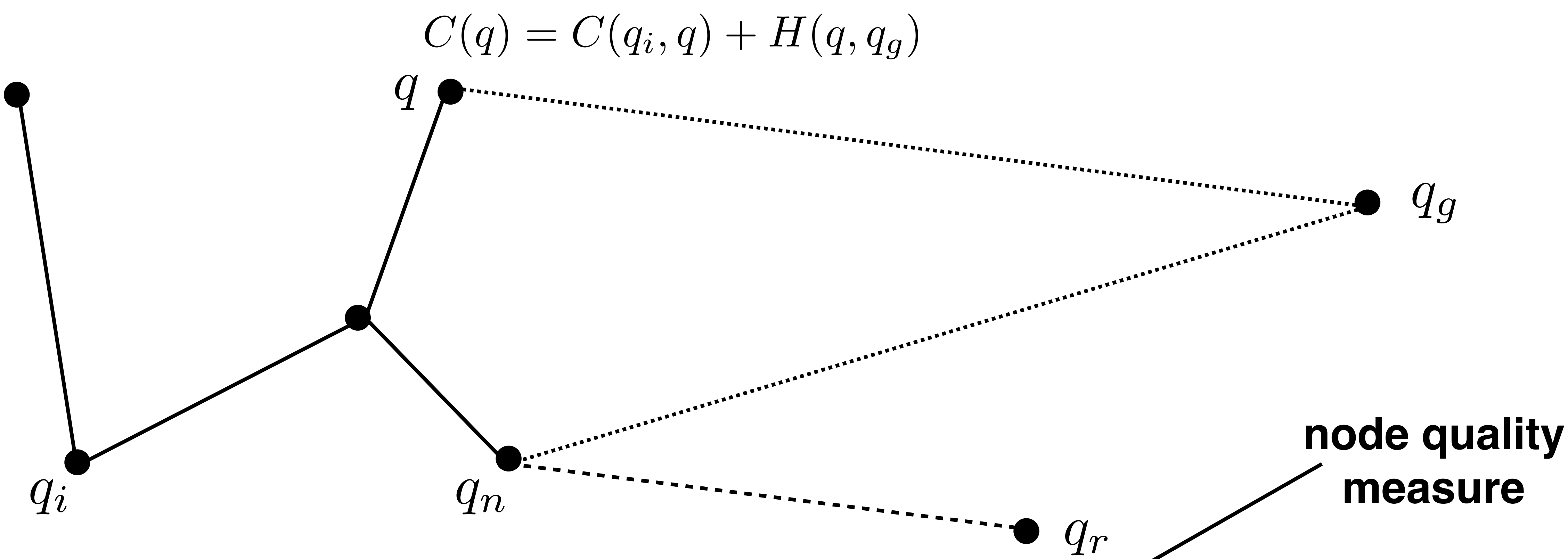
Crazy Fly Example



J. Preiss et al., Downwash-Aware Trajectory Planning for Large Quadcopter Teams, 2017

Heuristically Guided RRT

- Want to bias RRT growth towards lower cost paths

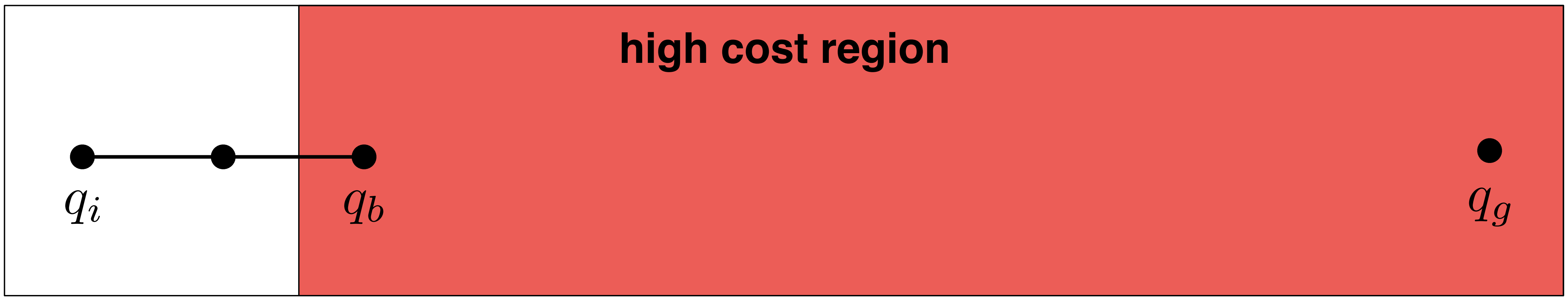


$$C_{opt} = H(q_i, q_g) \qquad C_{max} = \max_{q \in V} C(q) \qquad m(q) = 1 - \frac{C(q) - C_{opt}}{C_{max} - C_{opt}}$$

- Sample q_r, find q_n, compute $m(q_n)$
- Resample if $\mathrm{rand} > \max(m(q_n), p_{min})$, else expand q_n
- Tends to rejects samples that would expand poor nodes

Heuristically Guided RRT

- HRRT can reject a lot if poor nodes near open space



  - q_b has high probability of being NN, but low quality (high reject)

- Solution is to consider K nearest neighbours:
  - Keep iterating between neighbours until one is accepted
  - Alternatively, select the neighbour with with highest quality

Anytime RRT

- Anytime algorithms refine solutions over time
  - Can get a solution quickly, but get a better solution if you wait
- Anytime RRT iteratively creates multiple RRTs
- Cost of path from j-th iteration is given by

$$C^j$$

- Want to find trajectory with next RRT that is better

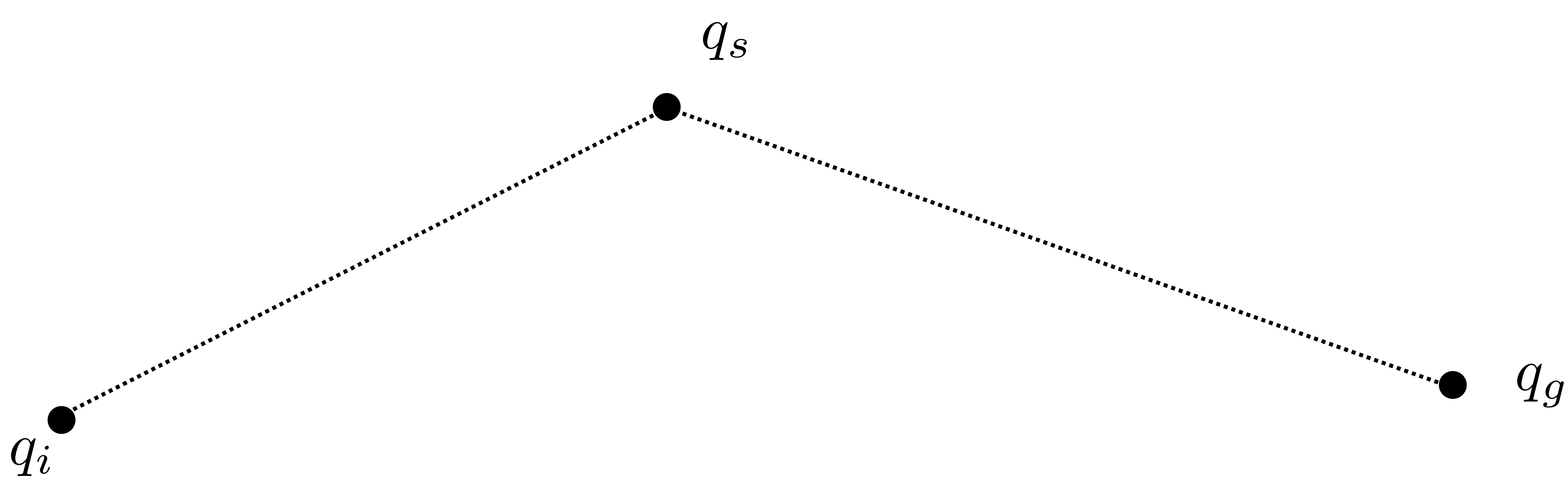
$$C^{j+1} < (1 - \epsilon)C^j = C_{target}$$

improvement factor

- How do we achieve the target cost?
  Clever node sampling, node selection, and node extension

Node Sampling

- Only accept nodes that may be better than target

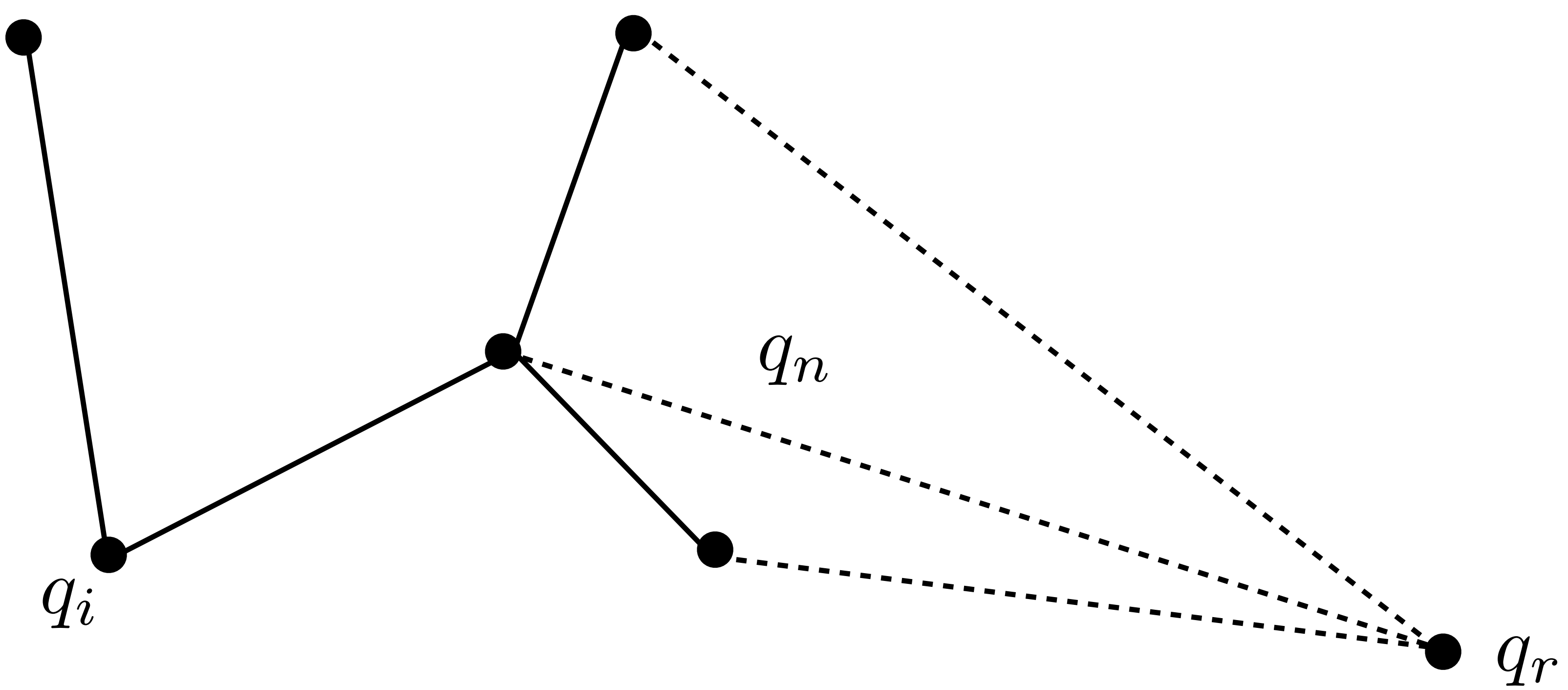


- Accept if

$$h(q_i, q_s) + h(q_s, q_g) < C_{target}$$

Node Selection

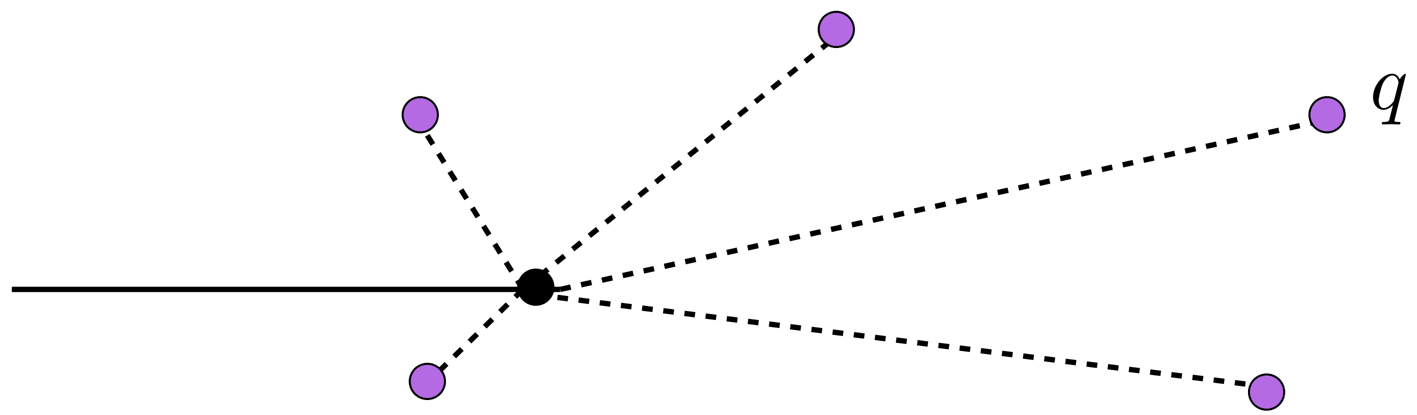
- Increasing favour building on nodes with low cost



- Select neighbour with min $\delta_d dist(q, q_r) + \delta_c cost(q_i, q)$
- Initially $\delta_d = 1 \text{ and } \delta_c = 0$
- With each iteration, increase δ_c and decrease δ_d

Node Extension

- Generate multiple potential extensions (shooting)



- Select cheapest extension that satisfies

$$C(q_i, q) + H(q, q_g) < C_{target}$$

and doesn't collide with obstacles

Questions?