Motion Planning Lecture 9

Sampling-Based Motion Planning: More Theory and Planners (EST, RRT-Connect, PRM*, LazyPRM, FMT*); Intro to Optimization

Wolfgang Hönig (TU Berlin) and Andreas Orthey (Realtime Robotics)

June 19, 2024

Recap: Sampling-based Motion Planning

Geometric

Basic: PRM, RRT, (EST, LazyPRM)
(Bidirectional: RRT-Connect)
Optimizing: RRT*, BIT*, SST*, (PRM*, FMT*)

Kinodynamic

Basic: kinodynamic RRT
Optimizing: AO-x, SST*

OMPL

C++ Library with Python bindings

Theoretical Insights

Robustly Feasible Motion-Planning Problem: Definitions

Robust Path/Trajectory

Let $\mathcal{B}_\delta(\mathbf{q})$ be the $d-$dimensional ball of radius $\delta > 0$ around configuration $\mathbf{q} \in \mathcal{Q}$. A path/trajectory $\mathbf{q} : [0, T] \rightarrow \mathcal{Q}$ is robust, if there exists $\delta > 0$ such that:

$$\mathcal{B}_\delta(\mathbf{q}(t)) \in \mathcal{Q}_{free} \; \forall t \in [0, T].$$

Robust Feasibility

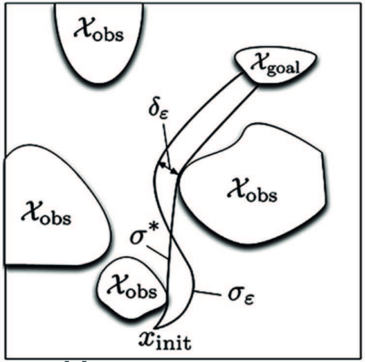
A motion planning problem is robustly feasible if a robust solution trajectory exists.

Robust Optimum

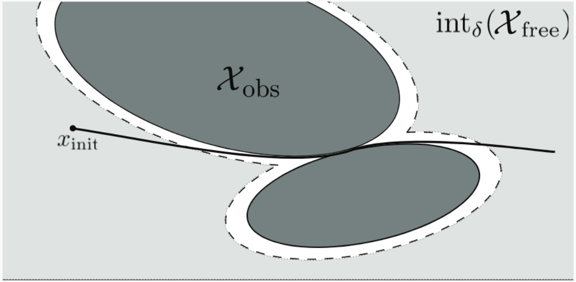
The robust optimum of a motion planning problem is:

$$J^* = \inf\{J(\mathbf{q}(p)) \,|\, \mathbf{q}(p) \text{ is a robust solution}\}.$$

Robustly Feasible Motion-Planning Problem: Examples



Source: [1]



Source: [2]

Why is this useful?

Probability of sampling a specific point is zero!

Recap: Completeness [3]

Completeness

An algorithm A is complete if in a finite amount of time, A always finds a solution if a solution exists or otherwise A determines that a solution does not exist. E.g., A*

Resolution Completeness

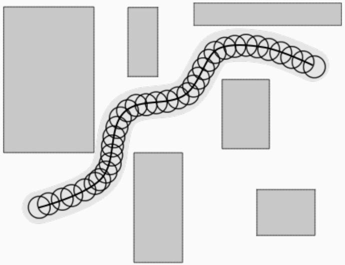
An algorithm A is resolution complete if in a finite amount of time and for some small resolution step $\epsilon > 0$, A always finds a solution if a solution exists or otherwise A determines that a solution does not exist. E.g., State-Lattice A*

Probabilistic Completeness

An algorithm A is probabilistically complete if the probability of finding a solution, if a solution exists, converges to 1, when the running time approaches infinity. E.g., RRT

Completeness of (geometric and kinodynamic) RRT [4]

def RRT$(\mathcal{Q}, \mathcal{W}_{free}, \mathcal{B}(\cdot), d(\cdot,\cdot), \mathbf{q}_{start}, \mathcal{Q}_{goal})$:
    $\mathcal{T} = (\mathcal{V}, \mathcal{E}) = (\{\mathbf{q}_{start}\}, \emptyset)$
    while True:
        $\mathbf{q}_{rand} = SAMPLE(\mathcal{Q}_{free})$
        $\mathbf{q}_{near} = NEAREST(\mathbf{q}_{rand}, \mathcal{V})$
        $\mathbf{q}_{new} = STEER(\mathbf{q}_{near}, \mathbf{q}_{rand})$
        if path $\mathbf{q}_{near}$ to $\mathbf{q}_{new}$ feasible:
            $\mathcal{V} = \mathcal{V} \cup \{\mathbf{q}_{new}\}$
            $\mathcal{E} = \mathcal{E} \cup \{\text{path } \mathbf{q}_{near} \text{ to } \mathbf{q}_{new}\}$
            if $\mathbf{q}_{new} \in \mathcal{Q}_{goal}$:
                return solution



Assume there exists a solution, i.e., a discrete sequence of configurations $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{k+1}$ and controls $\mathbf{u}_0, \mathbf{u}_2, \ldots, \mathbf{u}_k$.

Completeness of (geometric and kinodynamic) RRT [4]

Induction for $i = 0, \dots, k+1$ with hypothesis that a solution to $\mathbf{q}_i$ is found:

- For $i = 0$, this is trivially true since we add $\mathbf{q}_{start}$ in line 2
- Now assume it holds for i; we need to show the property for $i+1$:
- Assume the volume of each Voronoi region of the vertices is lower-bounded: $\mu(Voronoi(v_j)) > c_1 \, \forall v \in \mathcal{V}$
- The probability of selecting $\mathbf{q}_i$ for extension is $\mathbb{P}_s = \frac{\mu(Voronoi(\mathbf{q}_i))}{\mu(\mathcal{Q}_{free})} \geq \frac{c_1}{\mu(\mathcal{Q}_{free})}$
- Assume the probability to sample $\mathbf{u}_i$ such that we reach $\mathbf{q}_{i+1}$ from $\mathbf{q}_i$ is lower-bounded by c_2
- Probability to connect in one step : $\mathbb{P}_c \geq c_2 \cdot \mathbb{P}_s$
- Probability to connect after j iterations: $\mathbb{P}_j \geq 1 - (1 - \mathbb{P}_c)^j$
- $\lim_{j \to \infty} \mathbb{P}_j = 1$

Completeness of (geometric and kinodynamic) RRT [4]

Why does this proof not work for problems that are not robustly feasible?

Depending on the case, c_1 or c_2 might be zero.

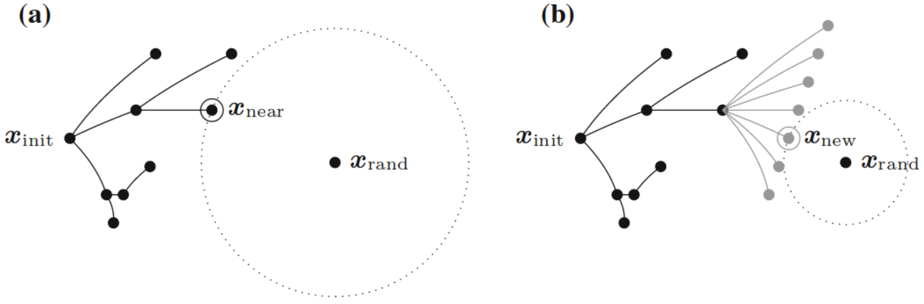
Why does this proof not work to show convergence to optimal solution?

(Assume $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{k+1}$ is a optimal sequence.)

Induction over pairs $\mathbf{q}_i, \mathbf{q}_{i+1}$ not sufficient. We would need to show that $\mathbf{q}_0, \ldots, \mathbf{q}_i$ leads to adding $\mathbf{q}_{i+1}$ with $\mathbf{q}_i$ as parent.

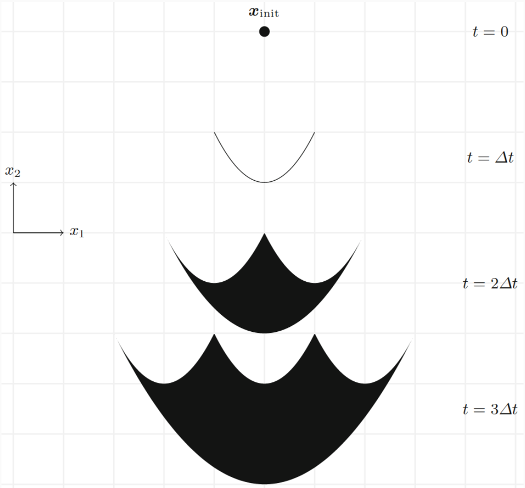
Not All RRTs are Complete (1) [5]

Consider the RRT variant with fixed Δt and best-input (analytical version of guided monte carlo)



Source: [5]

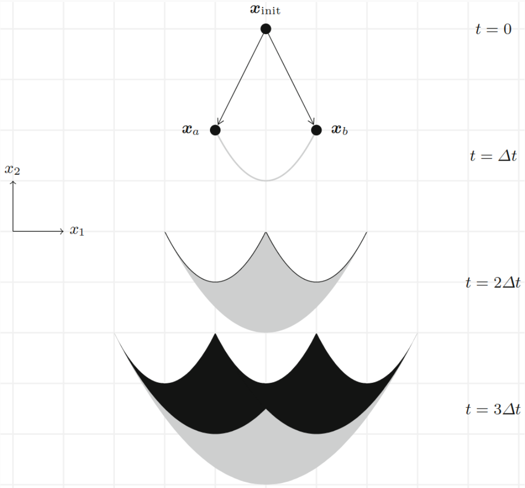
Not All RRTs are Complete (2) [5]



Source: [5]

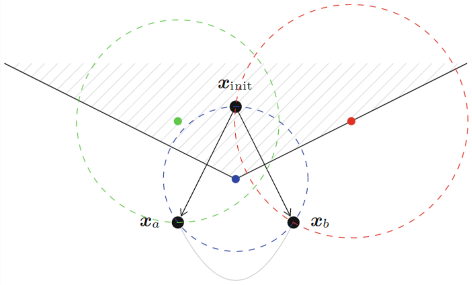
- Dynamics: $\dot{x}_1 = u$; $\dot{x}_2 = u^2 - 3$; $-1 \leq u \leq 1$
- We have $-3 \leq \dot{x}_2 \leq -2 \Rightarrow$ always moves in a negative direction; impossible to revisit an earlier state

Not All RRTs are Complete (3) [5]



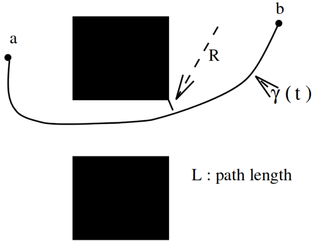
Source: [5]

- Consider the intermediate tree with 3 nodes (complete algorithm has to “recover” from any intermediate tree)
- Some space will never be explored (need to select x_{init} in first step; but then best-input would always pick x_a or x_b)



Completeness Convergence (geometric PRM) [6]

```
def GenPRM(𝒬, 𝒲_free, ℬ(·), N):
    𝒢 = (𝒱, ℰ) = (∅, ∅)
    # Generate N vertices
    while |𝒱| < N:
        q = Sample(𝒬)
        if ℬ(q) ⊂ 𝒲_free:
            𝒱 = 𝒱 ∪ {q}
    # Connect vertices
    for q in 𝒱:
        for p in {p ∈ 𝒱 : isNeighbor(p, q)}:
            if path q to p feasible:
                ℰ = ℰ ∪ {path q to p}
    return 𝒢
```



Source: [6]

- R: minimum clearance of solution $\mathbf{q}(p)$
- L: Total path length ($J(\mathbf{q}(p))$)

Completeness Convergence (geometric PRM) [6]

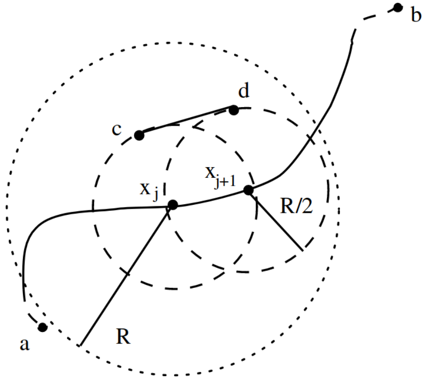
- Distribute configurations $\mathbf{q}_0, \mathbf{q}_1, \ldots \mathbf{q}_k$ on solution, with $k = \lceil 2L/R \rceil$, s.t.:

$$d(\mathbf{q}_i, \mathbf{q}_{i+1}) \leq R/2 \ \forall i$$

- Then we have (triangle inequality):

$$\mathcal{B}_{R/2}(\mathbf{q}_{i+1}) \subset \mathcal{B}_R(\mathbf{q}_i) \ \forall j = 0, \ldots, k-1$$

- For any $c \in \mathcal{B}_{R/2}(\mathbf{q}_i)$ and $d \in \mathcal{B}_{R/2}(\mathbf{q}_i)$, we know that $\overline{cd} \in \mathcal{Q}_{free}$, because $c, d \in \mathcal{B}_R(\mathbf{q}_i)$



Source: [6]

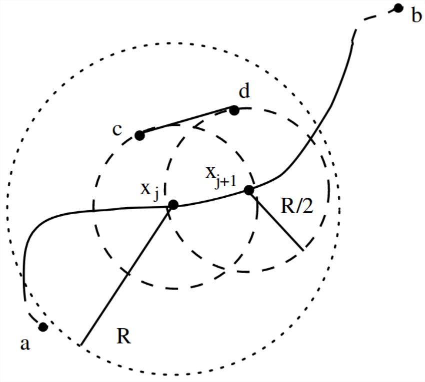
Completeness Convergence (geometric PRM) [6]

- Assume PRM generated N configurations $\mathbf{p}_1, \ldots, \mathbf{p}_N$
- If we have at least one $\mathbf{p}_j$ inside each ball $\mathcal{B}_{R/2}(\mathbf{q}_i)$, PRM will find the solution (using the geometric observations from last slide)
- Probability that sample $\mathbf{p}_j$ is in $\mathcal{B}_{R/2}(\mathbf{q}_i)$:

$$\mathbb{P}[\text{Sample is in Ball } i] = \frac{\mu(\mathcal{B}_{R/2}(\mathbf{q}_i))}{\mu(\mathcal{Q}_{free})}$$

- Probability that sample $\mathbf{p}_j$ is not in $\mathcal{B}_{R/2}(\mathbf{q}_i)$:

$$\mathbb{P}[\text{Sample is not in Ball } i] = 1 - \frac{\mu(\mathcal{B}_{R/2}(\mathbf{q}_i))}{\mu(\mathcal{Q}_{free})}$$



Source: [6]

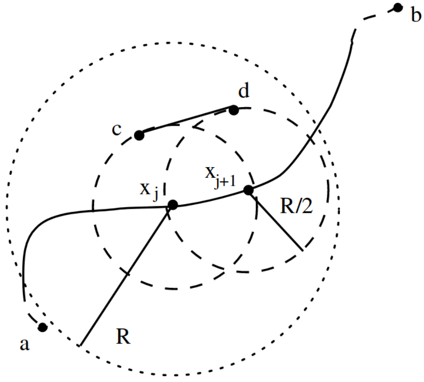
Completeness Convergence (geometric PRM) [6]

- Probability that no sample is in $\mathcal{B}_{R/2}(\mathbf{q}_i)$:

$$\mathbb{P}[\text{No Sample in Ball } i] = \left(1 - \frac{\mu(\mathcal{B}_{R/2}(\mathbf{q}_i))}{\mu(\mathcal{Q}_{free})}\right)^N$$

- Probability of not finding a solution:

$$\begin{aligned}\mathbb{P}[\text{Failure}] &\leq \mathbb{P}[\text{Some ball is empty}] \\ &\leq \sum_{i=1}^{k-1} \mathbb{P}[\text{No Sample in Ball } i] \\ &= \left(\left\lceil \frac{2L}{R} \right\rceil - 1\right)\left(1 - \frac{\mu(\mathcal{B}_{R/2}(\mathbf{q}_i))}{\mu(\mathcal{Q}_{free})}\right)^N\end{aligned}$$



Source: [6]

Completeness Convergence (geometric PRM) [6]

PRM Completeness Bound

The probability of PRM with N vertices not finding a solution in a d-dimensional space is bounded by:

$$\mathbb{P}[\text{Failure}] \leq \frac{2L}{R} e^{-\alpha_d R^d N},$$

where L is the path length, R is the minimum clearance, and α_d is a constant.

This is exponential in N, i.e., very fast convergence!

Similar results for RRT, see [4, 7].

Completeness Convergence (geometric PRM) [6]

What happens if I change $R = 1\,\mathrm{m}$ to $R = 0.5\,\mathrm{m}$?

Probability of failure increases (not linear).

What happens if problem is not robustly feasible?

R is zero, i.e., undefined.

How can we show probabilistic completeness using this result?

$$\lim_{N\to\infty} \mathbb{P}[\text{Failure}] \leq 0$$

Recap: Optimality

Optimality

An algorithm A is optimal if in a finite amount of time, A always finds the solution with the lowest cost c^* if a solution exists. E.g., A*

Bounded Suboptimality

An algorithm A is bounded suboptimal if in a finite amount of time, A always finds the solution of cost c that is at most a factor of ϵ larger than the optimal cost c^* if a solution exists: E.g., Weighted-A*

$$c \leq \epsilon c^*.$$

Recap: Optimality

Asymptotic Optimality

An algorithm is asymptotically optimal, if the probability that the solution cost c_t approaches c^* is 1, when the running time approaches infinity: E.g., RRT*

$$\lim_{t\to\infty} \mathbb{P}[\{c_t - c^* > \epsilon\}] = 0,\ \forall \epsilon > 0.$$

Asymptotic Near-Optimality

An algorithm is asymptotically near-optimal, if the probability that the solution cost c_t is at most a factor of ϵ larger than the optimal solution c^* is 1, when the running time approaches infinity: E.g., SST*

$$\lim_{t\to\infty} \mathbb{P}[\{c_t > \epsilon c^*\}] = 0.$$

Recap: Optimality

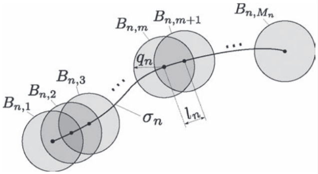
Does Optimality Imply Completeness?

No (an optimal algorithm may never terminate if no solution exists).

Does Asymptotic Optimality Imply Probabilistic Completeness?

Yes.

Asymptotic Optimality of PRM* [2]



1. Cover the trajectory with balls, as before
2. For each number of iterations use a different "robustness" δ_n, i.e., the optimal δ_n-robust solution is c_n^* (such that $\lim_{n\to\infty} \delta_n = 0$)
3. Analyze:

$$\sum_{n=1}^{\infty} \mathbb{P}[\{c_n - c_n^* > \epsilon\}]$$

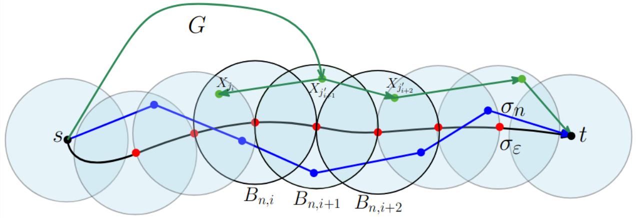
4. This is bounded, i.e., $< \infty$ (Intuition: $\mathbb{P}[\{c_n - c_n^* > \epsilon\}]$ is exponential in n)
5. Use Borel–Cantelli lemma (if the sum is probabilities of events is finite, then the probability that infinite many of them occur is zero)

Asymptotic Optimality of RRT* [1, 2]

Challenges Over PRM*

- There is an order between vertices in the tree
- Sampling is not uniform i.i.d. anymore

- Still use the same sequence of balls idea



- Now adds another dimension (time) to deal with ordering and need to show that *neighboring* balls can be connected

Asymptotic Optimality of AO-x [8]

Assumptions

- Underlying algorithm is complete (termination in finite time)
- Non-negligible improvement in each iteration

- Show that, in expectation, $c_n - c_n^*$ reduces exponential in n and use Markov inequality ($P(X > \epsilon) \leq E[X]/\epsilon$)
- Details, see Lecture 7

Convergence rate not useful, since it assumes that improvement in non-negligible factor ω.

Asymptotic Optimality Convergence [9]

Convergence Rate of PRM*

PRM* converges to the optimal solution with rate:

$$O(N^{-1/d+\rho}),$$

where N is the number of vertices, d the dimension of the configuration space, and ρ is a small positive constant.

This is much slower than the convergence to a feasible solution (which was exponential in N).

Do We Need True Randomness? [10]

Using deterministic sequences (e.g., Halton sequence) has many advantages:

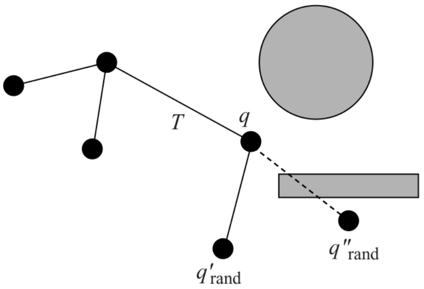
- Remains asymptotically optimal
- Works with smaller connection radii
- Known suboptimality convergence rate (i.e., for a user-specified suboptimality, we can compute number of iterations)
- Empirically (slightly) better

More Sampling-based Planners

EST: Expansive Space Trees (1) [11]

- Key insight: use explicit function rather than Voronoi bias for exploration

def EST$(\mathcal{Q}, \mathcal{W}_{free}, \mathcal{B}(\cdot), \mathbf{q}_{start}, \mathcal{Q}_{goal})$:
    $\mathcal{T} = (\mathcal{V}, \mathcal{E}) = (\{\mathbf{q}_{start}\}, \emptyset)$
    while True:
        $\mathbf{q}$ =randomly choose from $\mathcal{V}$ with
        $\hookrightarrow$ probability $\pi_{\mathcal{T}}(\mathbf{q})$
        $\mathbf{p} =$ random configuration near $\mathbf{q}$
        if path $\mathbf{q}$ to $\mathbf{p}$ feasible:
            $\mathcal{V} = \mathcal{V} \cup \{\mathbf{p}\}$
            $\mathcal{E} = \mathcal{E} \cup \{\text{path } \mathbf{q} \text{ to } \mathbf{p}\}$
            if $\mathbf{p} \in \mathcal{Q}_{goal}$:
                return solution



Source: [3]

EST: Expansive Space Trees (2)

- Choice of probability density function $\pi_{\mathcal{T}}(\mathbf{q})$: Good exploration of $\mathcal{Q}_{free}$, e.g., proportional to dispersion
- $\pi_{\mathcal{T}}(\mathbf{q})$ often changes during the search

Online Dispersion Estimation

- Discretize $\mathcal{Q}$ in a grid
- Count the number of $\mathbf{q} \in \mathcal{V}$ that belong to each grid cell
- Probability $\pi_{\mathcal{T}}(\mathbf{q})$ is inverse proportional to the number corresponding to the grid cell of $\mathbf{q}$

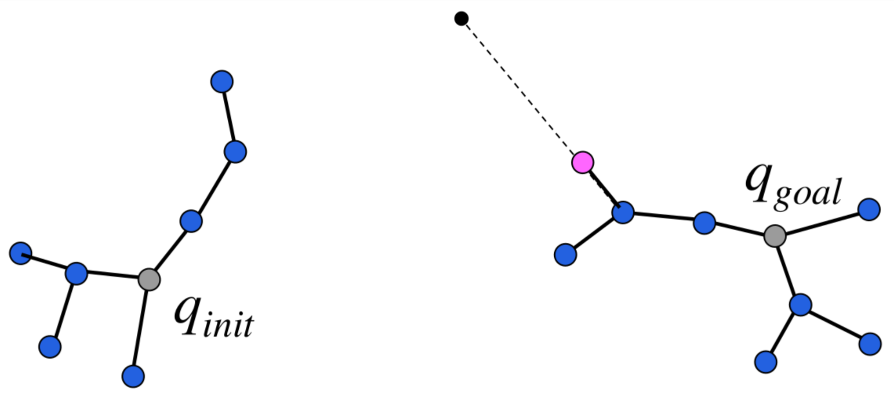
EST Main Challenge

Difficult to define $\pi_{\mathcal{T}}(\mathbf{q})$ efficiently.

RRT-Connect (1) [12]

- Bidirectional search: Use two trees: one rooted at $\mathbf{q}_{start}$, one rooted at $\mathbf{q}_{goal}$
- Try to connect both trees

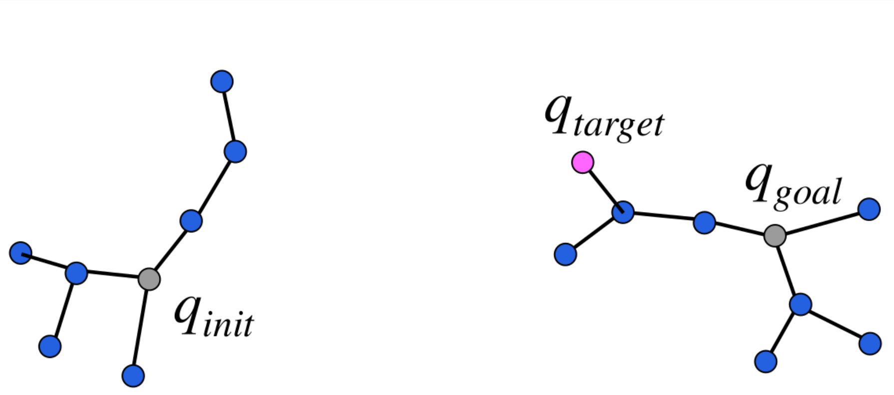
RRT-Connect (2)



Source: [13]

- Sample $\mathbf{q}_{rand}$ and Extend the goal tree (right side)

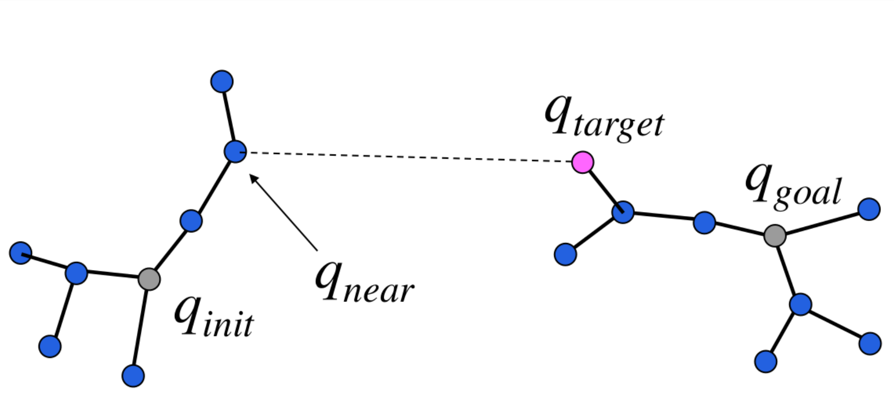
RRT-Connect (3)



Source: [13]

- $\mathbf{q}_{target}$ is now the goal for the init tree (left side)

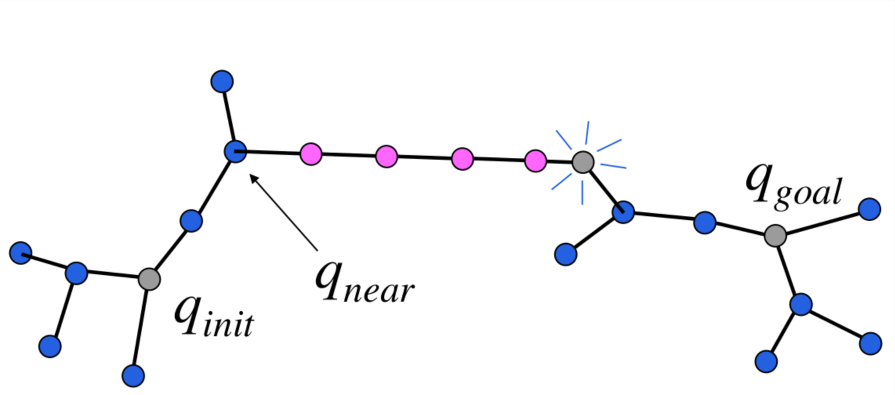
RRT-Connect (4)



Source: [13]

- Calculate $\mathbf{q}_{near}$ (closest node to $\mathbf{q}_{target}$ in init tree)

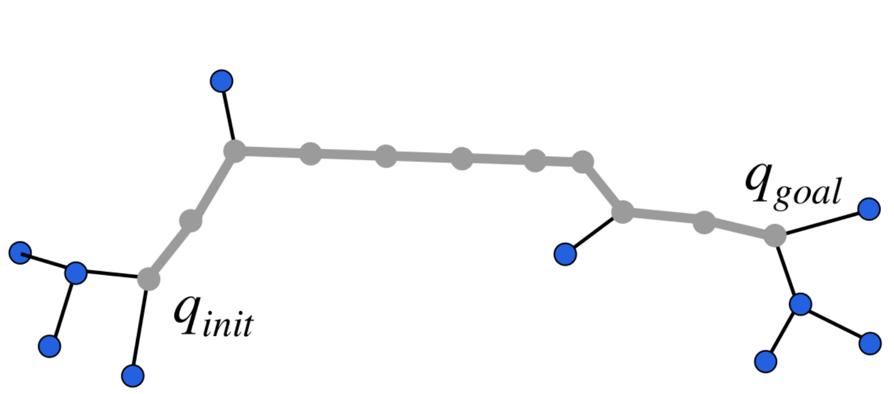
RRT-Connect (5)



Source: [13]

- Try to connect $\mathbf{q}_{near}$ and $\mathbf{q}_{target}$

RRT-Connect (6)



Source: [13]

- Solution is the path connecting $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$

RRT-Connect (7)

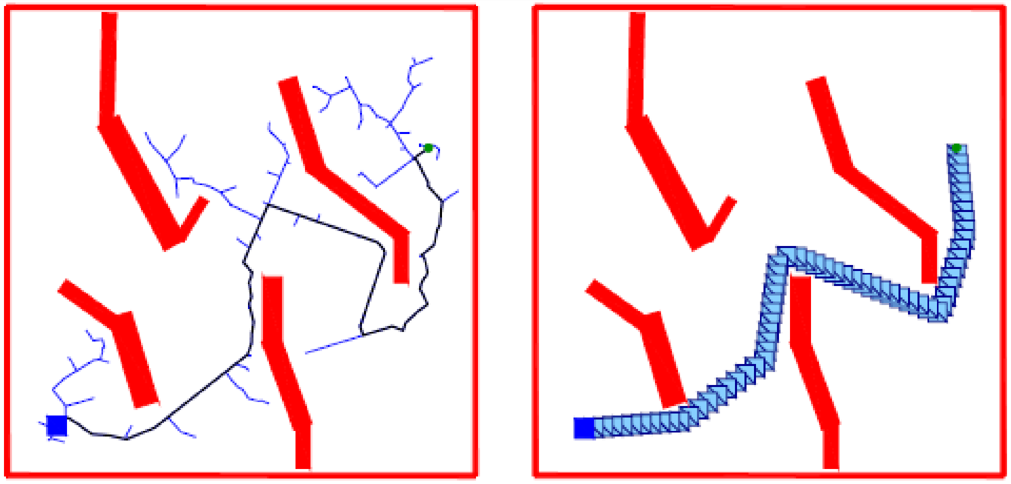
Pseudo-Code from the original paper:

```
RRT_CONNECT_PLANNER(q_init, q_goal)
    T_a.init(q_init); T_b.init(q_goal);
    for k = 1 to K do
        q_rand ← RANDOM_CONFIG();
        if not (EXTEND(T_a, q_rand) = Trapped) then
            if (CONNECT(T_b, q_new) = Reached) then
                Return PATH(T_a, T_b);
        SWAP(T_a, T_b);
    Return Failure
```

Source: [12]

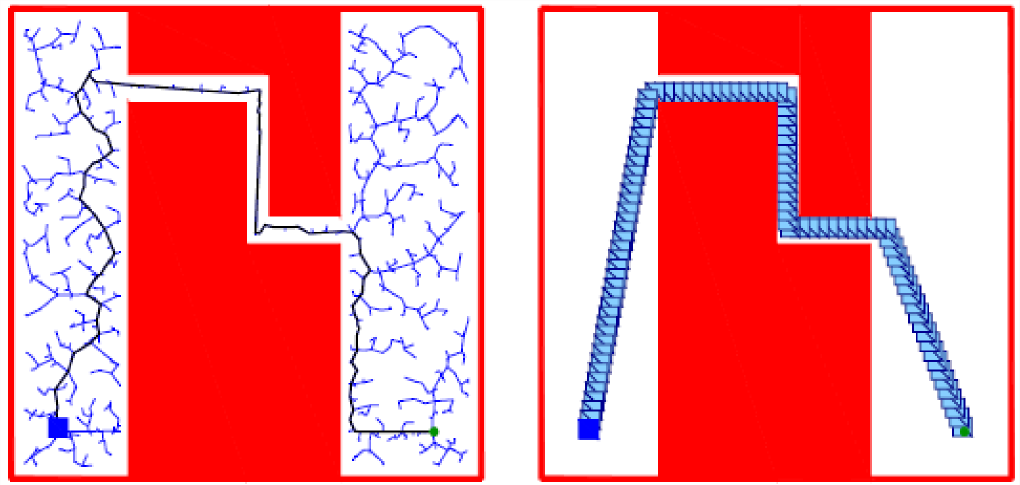
What is the purpose of SWAP here?

RRT-Connect Examples (1)



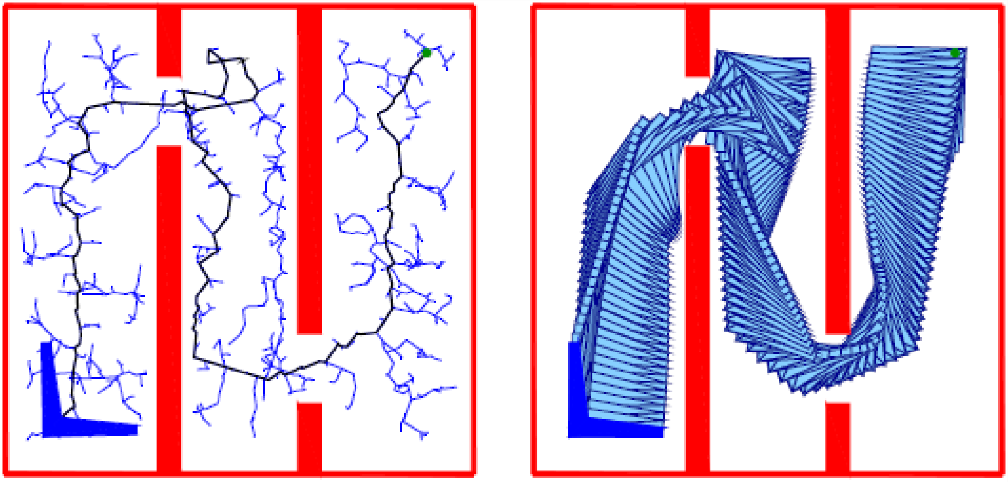
Source: [12]

RRT-Connect Examples (2)



Source: [12]

RRT-Connect Examples (3)



Source: [12]

PRM*

Algorithm 4: PRM*

```
V ← {x_init} ∪ {SampleFree_i}_{i=1,...,n}; E ← ∅;
foreach v ∈ V do
    U ← Near(G = (V, E), v, γ_PRM(log(n)/n)^{1/d}) \ {v};
    foreach u ∈ U do
        if CollisionFree(v, u) then E ← E ∪ {(v, u), (u, v)}
return G = (V, E);
```

```
def GenPRM(𝒬, 𝒲_free, ℬ(·), N):
  # ...
  for q in 𝒱:
    for p in {p ∈ 𝒱 : isNeighbor(p, q)}:
      if path q to p feasible:
        ℰ = ℰ ∪ {path q to p}
  return 𝒢
```

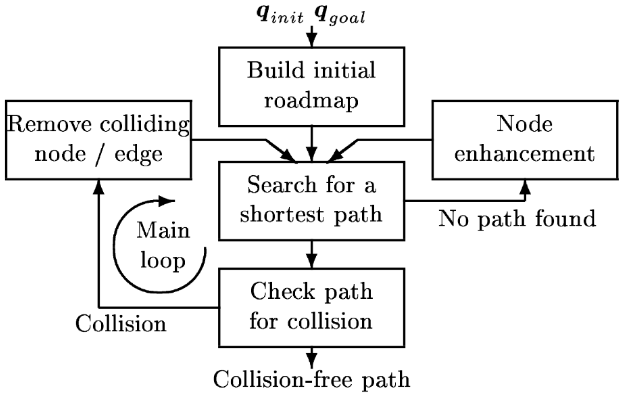
- Pseudo code from [2]
- Consistent with our previous pseudo code of PRM (lecture 5)
- Neighbors are computed using the dynamic radius, depending on $|\mathcal{V}|$

How does this work for parallel pre-processing and query?

LazyPRM (1) [15]

LazyPRM Insight

Collision checking takes majority of the time ⇒ delay as much as possible



Source: [14]

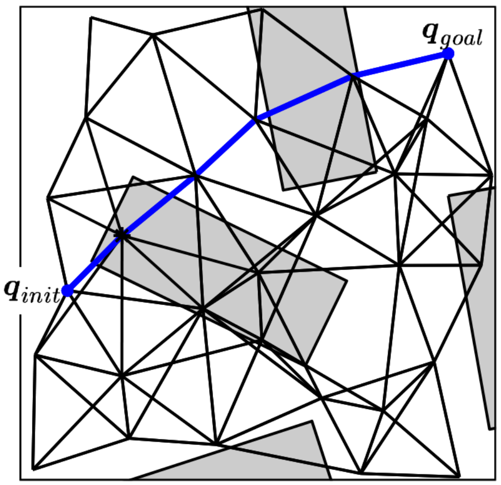
LazyPRM (2) [15]

What is a good strategy for collision checking here? (Goal: Minimize collision checks.)

1. Check vertices starting from $\mathbf{q}_{start}$ and $\mathbf{q}_{goal}$ towards the center.
2. Check edges with a coarse granularity (i.e., start with midpoint of each edge) following the same edge order (edges close to start/goal first).
3. Iteratively refine edge-checking granularity.

LazyPRM (2) [15]

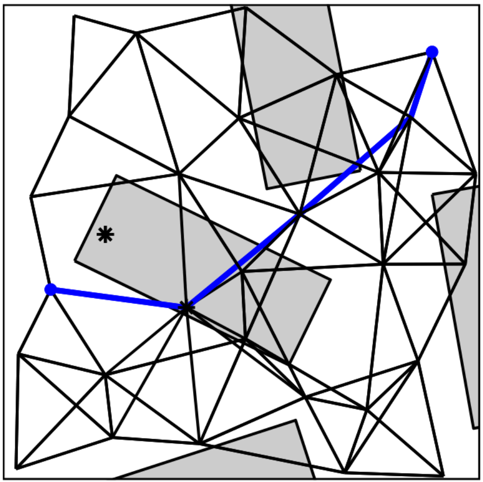
Solution path found on the initial roadmap; 1 vertex in collision (*)



Source: [14]

LazyPRM (3) [15]

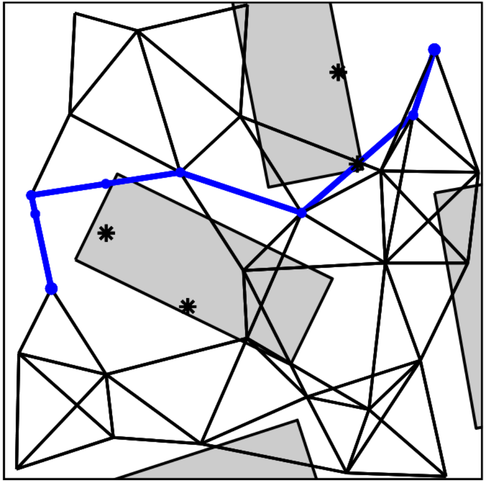
Vertex (and edges) in roadmap deleted; New solution path found (1 vertex collision)



Source: [14]

LazyPRM (4) [15]

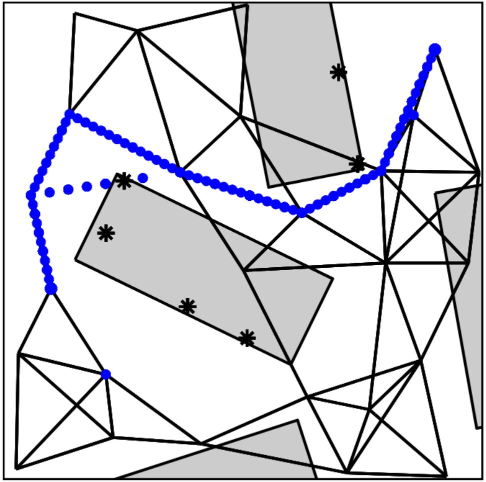
Vertex (and edges) in roadmap deleted; New solution path found (1 edge collision)



Source: [14]

LazyPRM (5) [15]

Final solution path found after fine-grained edge checking



Source: [14]

LazyPRM (6) [15]

Node Enhancement:

- Add additional vertices, once roadmap becomes disconnected
- Select seeds, i.e., vertices that are close to the boundary (e.g., midpoints of edges that were discovered to be in collision)
- Sample new points close to the seed vertices (e.g., normal distribution)

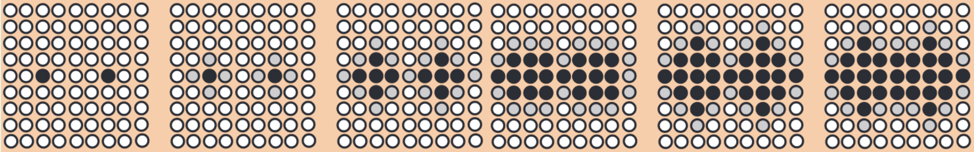
OMPL

LazyPRM in OMPL does not include all changes as in paper [15]. LazyPRM* and LazyRRT follow the same key insight, but are not described in a paper.

Fast marching tree (FMT*) (1) [9]

Background: Fast Marching Method

- Numerical method to track the front of a propagating wave [16]
- Example: Throw a stone in a pond with different fluids (water; oil); This method can track the wavefront over time
- Related to Dijkstra: build the solution in an *outward* direction, without backtracking



Source: [17]

Fast marching tree (FMT*) (2) [9]

Algorithm 1. Fast Marching Tree algorithm (FMT*): Basics.

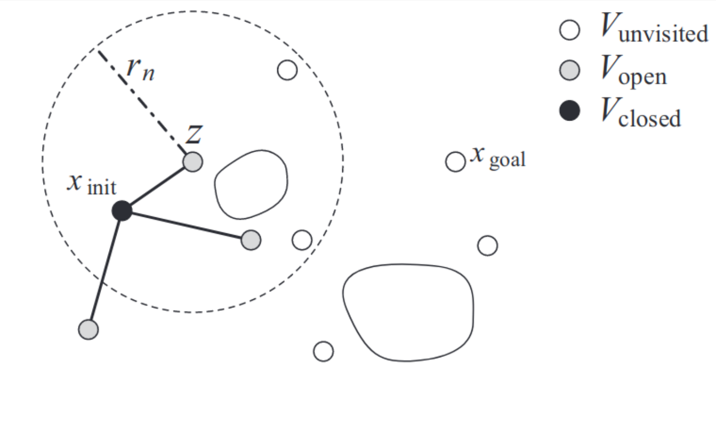
Require: Sample set V comprised of x_{init} and n samples in X_{free}, at least one of which is also in X_{goal}

```
Place x_init in V_open and all other samples in V_unvisited; initialize tree with root node x_init
Find lowest-cost node z in V_open
    For each of z's neighbors x in V_unvisited:
        Find neighbor nodes y in V_open
        Find locally optimal one-step connection to x from among nodes y
        If that connection is collision-free, add edge to tree of paths
    Remove successfully connected nodes x from V_unvisited and add them to V_open
    Remove z from V_open and add it to V_closed
    Repeat until either:
  (1) V_open is empty ⇒ report failure
  (2) Lowest-cost node z in V_open is in X_goal ⇒ return unique path to z and report success
```

Source: [9]

Fast marching tree (FMT*) (3) [9]

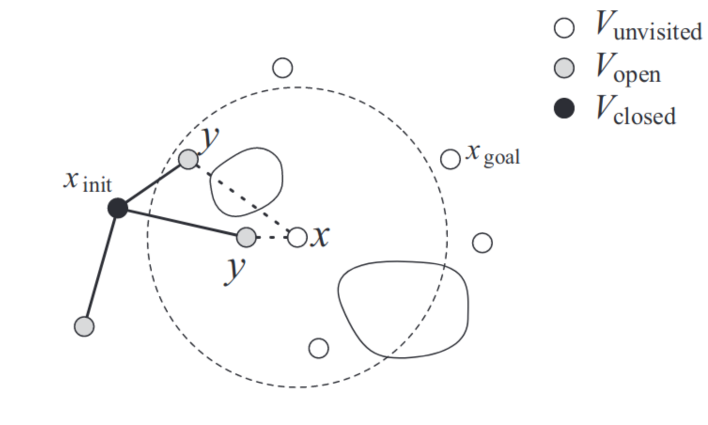
Select the lowest-cost configuration z from set V_{open} and find neighbors in $V_{unvisited}$ within radius r_n (similar to RRT*/PRM*). [Lines 2 – 3]



Source: [9]

Fast marching tree (FMT*) (4) [9]

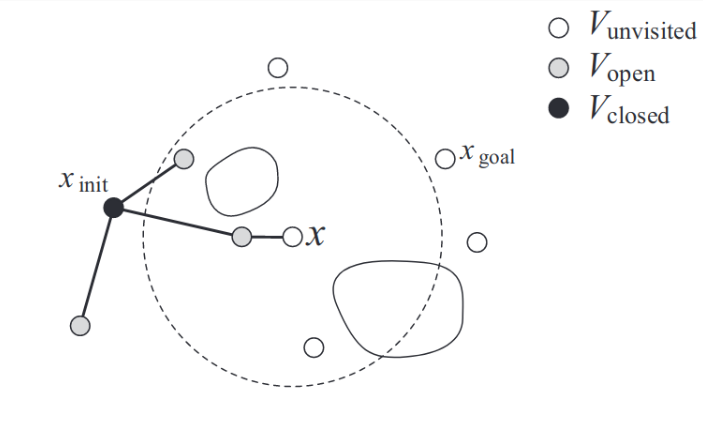
For a neighbor x, find nearby configurations (within r_n of x) in V_{open}. [Lines 4 – 5]



Source: [9]

Fast marching tree (FMT*) (5) [9]

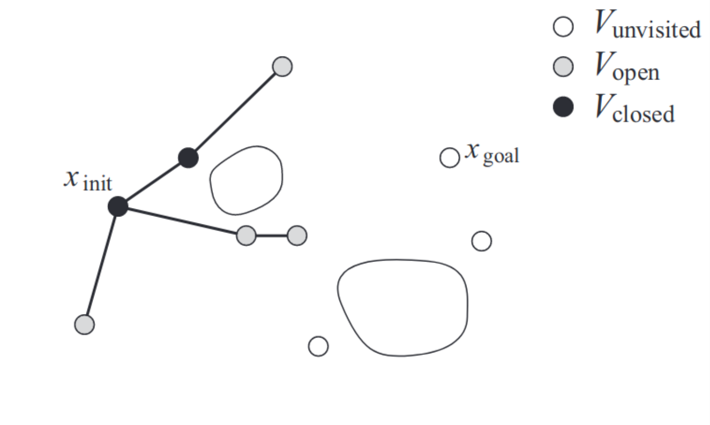
Pick the lowest-cost neighbor (ignoring obstacles) and add if edge is collision free. [Line 6]



Source: [9]

Fast marching tree (FMT*) (6) [9]

After z is explored: put it in V_{closed}, add new configurations to V_{open} [Lines 7 – 8]



Source: [9]

Fast marching tree (FMT*) (7) [9]

How does this differ from RRT*?

- Vertices are pre-sampled (i.e., more similar to PRM than RRT) $\Rightarrow$ (vanilla) FMT* is not anytime
- Usage of different sets (unvisited, open, closed), which allows local Bellman optimality

Properties

- Asymptotically Optimal with known convergence rate
- Time: $O(n \log n)$, Collision checks: $O(n)$, Space: $O(n \log n)$

Convex Optimization

Optimization Problems

Unconstrained Optimization

Let $\mathbf{x} \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$. Find:

$$\operatorname*{argmin}_{\mathbf{x}} f(\mathbf{x}).$$

Constrained Optimization

Let $\mathbf{x} \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$, $h : \mathbb{R}^n \to \mathbb{R}^l$. Find:

$$\operatorname*{argmin}_{\mathbf{x}} f(\mathbf{x}) \text{ s.t. } g(\mathbf{x}) \leq \mathbf{0}, h(\mathbf{x}) = \mathbf{0}.$$

- Blackbox: only $f(x)$ can be evaluated
- Gradient: $\nabla f(x)$ can be evaluated
- 2nd order: $\nabla^2 f(x)$ can be evaluated

Optimization Algorithms

- Gradient Descent (unconstrained)
- Augmented Lagrangian (converts constrained in unconstrained problem)

General Optimization is Local

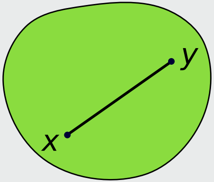
Most algorithms are local and only provide a (refined) solution around the initial guess.

Convex Optimization (1)

Convex Set

A set $\mathcal{X}$ is convex iff (if and only if):

$$\forall x, y \in \mathcal{X}, a \in [0, 1] : ax + (1 - a)y \in \mathcal{X}$$



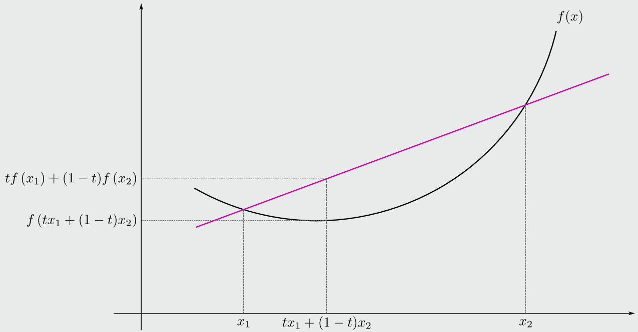
Source: Wikipedia

Convex Optimization (2)

Convex Function

A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex iff:

$$\forall x, y \in \mathbb{R}^n, a \in [0,1] : f(ax + (1-a)y) \leq af(x) + (1-a)f(y)$$



Source: Wikipedia

Convex Optimization (3)

Convex Set

A set $\mathcal{X}$ is convex iff (if and only if):

$$\forall x, y \in \mathcal{X}, a \in [0, 1] : ax + (1 - a)y \in \mathcal{X}$$

Convex Function

A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex iff:

$$\forall x, y \in \mathbb{R}^n, a \in [0, 1] : f(ax + (1 - a)y) \leq af(x) + (1 - a)f(y)$$

Convex Program / Optimization

A constrained optimization problem is convex iff f and g are convex and h is linear.

Convex Optimization (4)

Convex Optimization Is Global

Every local minima is a global minimum!

But still difficult to find such minima in predictable time.

Convex Optimization (5)

Linear Program (LP)

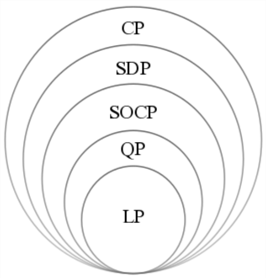
$$\underset{\mathbf{x}}{\operatorname{argmin}}\, c^\top \mathbf{x} \text{ s.t. } G\mathbf{x} \leq h, A\mathbf{x} = \mathbf{b}.$$

Can be solved in polynomial time!

Quadratic Program (QP)

$$\underset{\mathbf{x}}{\operatorname{argmin}}\, \frac{1}{2}\mathbf{x}Q\mathbf{x}^\top + c^\top x \text{ s.t. } G\mathbf{x} \leq h, A\mathbf{x} = \mathbf{b}.$$

If Q is positive definite, can be solved in polynomial time!



Source: Wikipedia

Conclusion

- Different forms of completeness
- Proofs use robust feasibility to compute probabilities of not sampling (or connecting) vertices
- Different forms of optimality
- Proofs are similar in style and keep track of cost-reductions over time
- New variants for geometric planners: EST, RRT-Connect, PRM*, Lazy PRM, FMT*
- Intro to (convex) optimization

Next Time

- Part 3: Optimization-based Motion Planning

Suggested Reading

1. Howie Choset, Kevin M. Lynch, Seth Hutchinson, George A. Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Intelligent Robotics and Autonomous Agents Series. Cambridge, MA, USA: A Bradford Book, 2005. 630 pp. ISBN: 978-0-262-03327-5, Section 7.4
2. Papers referenced in the slide titles

[1] Kiril Solovey, Lucas Janson, Edward Schmerling, Emilio Frazzoli, and Marco Pavone. "Revisiting the Asymptotic Optimality of RRT*". In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 2020, pp. 2189–2195. DOI: 10.1109/ICRA40945.2020.9196553.

[2] Sertac Karaman and Emilio Frazzoli. "Sampling-Based Algorithms for Optimal Motion Planning". In: *International Journal of Robotics Research (IJRR)* 30.7 (2011), pp. 846–894. DOI: 10.1177/0278364911406761.

References ii

[3] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George A. Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Intelligent Robotics and Autonomous Agents Series. Cambridge, MA, USA: A Bradford Book, 2005. 630 pp. ISBN: 978-0-262-03327-5.

[4] Steven M. LaValle and James J. Kuffner. "Randomized Kinodynamic Planning". In: *The International Journal of Robotics Research* 20.5 (May 1, 2001), pp. 378–400. ISSN: 0278-3649. DOI: 10.1177/02783640122067453.

[5] Tobias Kunz and Mike Stilman. "Kinodynamic RRTs with fixed time step and best-input extension are not probabilistically complete". In: *Algorithmic foundations of robotics XI*. Springer, 2015, pp. 233–244.

[6] L.E. Kavraki, M.N. Kolountzakis, and J.-C. Latombe. “Analysis of Probabilistic Roadmaps for Path Planning”. In: *IEEE Transactions on Robotics and Automation* 14.1 (Feb. 1998), pp. 166–171. ISSN: 2374-958X. DOI: 10.1109/70.660866.

[7] Michal Kleinbort, Kiril Solovey, Zakary Littlefield, Kostas E. Bekris, and Dan Halperin. “Probabilistic Completeness of RRT for Geometric and Kinodynamic Planning With Forward Propagation”. In: *IEEE Robotics and Automation Letters* 4.2 (Apr. 2019), pp. x–xvi. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2018.2888947.

[8] Kris Hauser and Yilun Zhou. "Asymptotically Optimal Planning by Feasible Kinodynamic Planning in a State-Cost Space". In: *IEEE Trans. Robotics* 32.6 (2016), pp. 1431–1443. DOI: 10.1109/TRO.2016.2602363.

[9] Lucas Janson, Edward Schmerling, Ashley A. Clark, and Marco Pavone. "Fast Marching Tree: A Fast Marching Sampling-Based Method for Optimal Motion Planning in Many Dimensions". In: *I. J. Robotics Res.* 34.7 (2015), pp. 883–921. DOI: 10.1177/0278364915577958.

[10] Lucas Janson, Brian Ichter, and Marco Pavone. "Deterministic Sampling-Based Motion Planning: Optimality, Complexity, and Performance". In: *The International Journal of Robotics Research* 37.1 (Jan. 1, 2018), pp. 46–61. ISSN: 0278-3649. DOI: 10.1177/0278364917714338.

[11] D. Hsu, J.-C. Latombe, and R. Motwani. "Path Planning in Expansive Configuration Spaces". In: *International Conference on Robotics and Automation*. Vol. 3. Apr. 1997, 2719–2726 vol.3. DOI: 10.1109/ROBOT.1997.619371.

[12] J.J. Kuffner and S.M. LaValle. "RRT-Connect: An Efficient Approach to Single-Query Path Planning". In: *International Conference on Robotics and Automation*. Vol. 2. Apr. 2000, 995–1001 vol.2. DOI: 10.1109/ROBOT.2000.844730.

[13] Dmitry Berenson. "Motion Planning: Robotics and Beyond". In: (2021). URL: https://web.eecs.umich.edu/~dmitryb/courses/winter2021motionplanning/index.html.

[14] Robert Bohlin and Lydia E. Kavraki. "A Randomized Approach to Robot Path Planning Based on Lazy Evaluation". In: *Handbook of Randomized Computing* 1 (), pp. 221–253.

[15] R. Bohlin and L.E. Kavraki. "Path Planning Using Lazy PRM". In: *IEEE International Conference on Robotics and Automation*. Vol. 1. Apr. 2000, 521–528 vol.1. DOI: 10.1109/ROBOT.2000.844107.

[16] Stanley Osher and James A Sethian. "Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations". In: *Journal of Computational Physics* 79.1 (Nov. 1, 1988), pp. 12–49. ISSN: 0021-9991. DOI: 10.1016/0021-9991(88)90002-2.

[17] Alberto Valero-Gomez, Javier V. Gomez, Santiago Garrido, and Luis Moreno. "The Path to Efficiency: Fast Marching Method for Safer, More Efficient Mobile Robot Trajectories". In: *IEEE Robotics & Automation Magazine* 20.4 (Dec. 2013), pp. 111–120. ISSN: 1558-223X. DOI: 10.1109/MRA.2013.2248309.