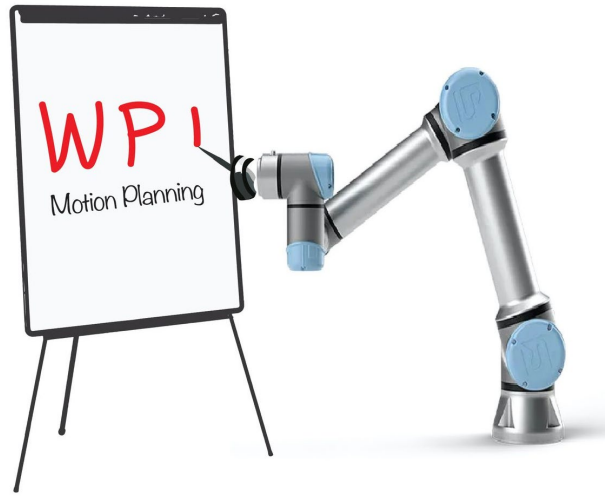
RBE550
Motion Planning
Asymptotically Optimal Planning



Constantinos Chamzas
www.cchamzas.com
www.elpislab.org

Disclaimer and Acknowledgments

The slides are a compilation of work based on notes and slides from Bryce Willey, Lydia Kavraki, Carlos Quintero Pena and Constantinos Chamzas,

Last Time

- Probabilistic Completeness
- Analysis of PRM
- Characterization of space
  - ϵ−good
  - β−lookout
  - $(\epsilon, \alpha, \beta)$−expansive
  - Theoretical results

Overview

- Asymptotic Optimality
- RRT*
- RRT#
- BIT*, FMT*, IRRT*
- Simplification Methods

What can we do so far e.g. solve feasibility

What is optimality?

- **High level:** The property of a planner to return a motion which surpasses all other motions in quality.

- **Mid-level:** From all possible parts return the one which minimizes the objective function.

Rigorus Optimality definition

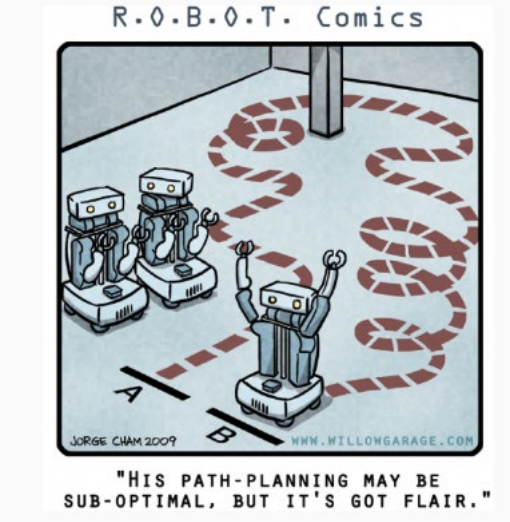
Optimal Planning: The problem of finding a global optimal path.
An optimal path is a path ***σ*** which minimizes a given cost function $s: \Sigma \rightarrow \mathrm{R}$

$$\sigma^* = \underset{\sigma \in \Sigma}{\arg\min} \{ s(\sigma) \mid \sigma(0) = \mathbf{x}_{\text{start}}, \sigma(1) \in \mathbf{x}_{\text{goal}},$$

$$\forall t \in [0,1], \sigma(t) \in X_{\text{free}} \},$$

Why do we need optimality?

Aesthetics



Paths should look good

Efficiency

Paths should be time and energy efficient

Safety

Should keep safety distance

Cost function types

Cost function types

Objective (or cost) function c. Graph $G = (V, E)$ and paths $P = (e_1, \ldots, e_N)$.

- Cost for a configuration $c : V \to \mathbb{R}_{\geq 0}$
- Cost of an edge $c : E \to \mathbb{R}_{\geq 0}$
- Cost of a path $c : P \to \mathbb{R}_{\geq 0}$

Cost function for efficiency (Path Length)

Shortest length

- Configuration cost: Zero
- Edge cost: Length of segment, metric distance
- Path cost: Sum of edge costs

Cost Function for Safety

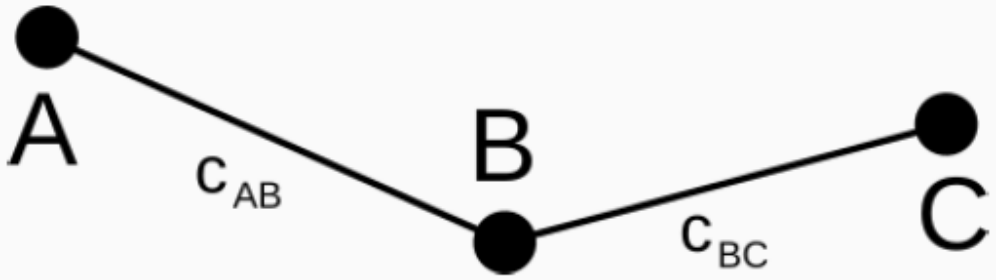
Maximum clearance

- Configuration cost: Distance from robot to environment
- Edge cost: Maximum over all configurations on edge
- Path cost: Maximum over all edges on path

Additive property

Additive costs

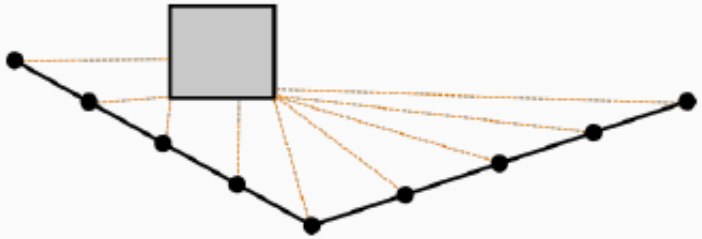
- Note: Most planners like RRT*, BIT* require additive cost!
- Additive cost: cost(A,B,C) = cost(A,B) + cost(B,C)



Non-additive cost Example

Non-additive cost example

Average clearance cost.



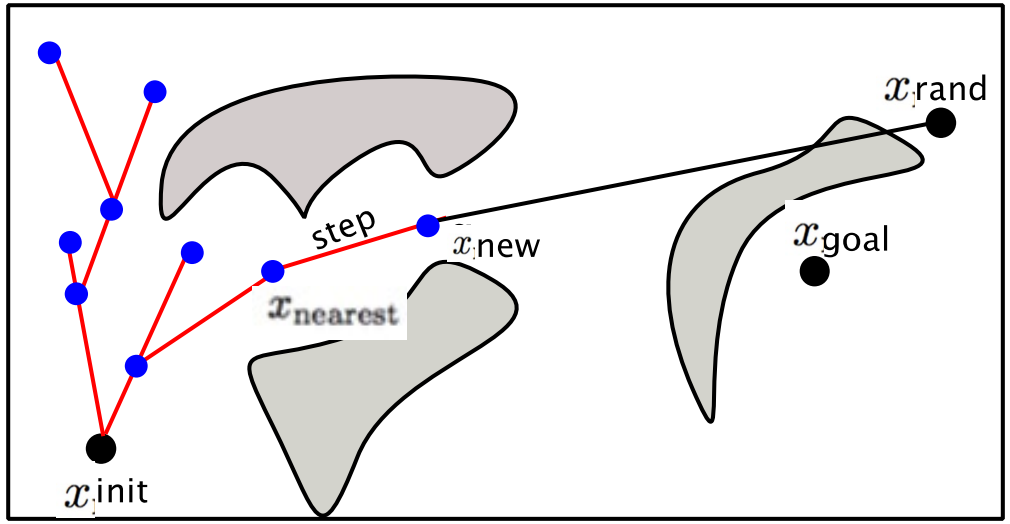
RRT-recap Reminder

Basic:

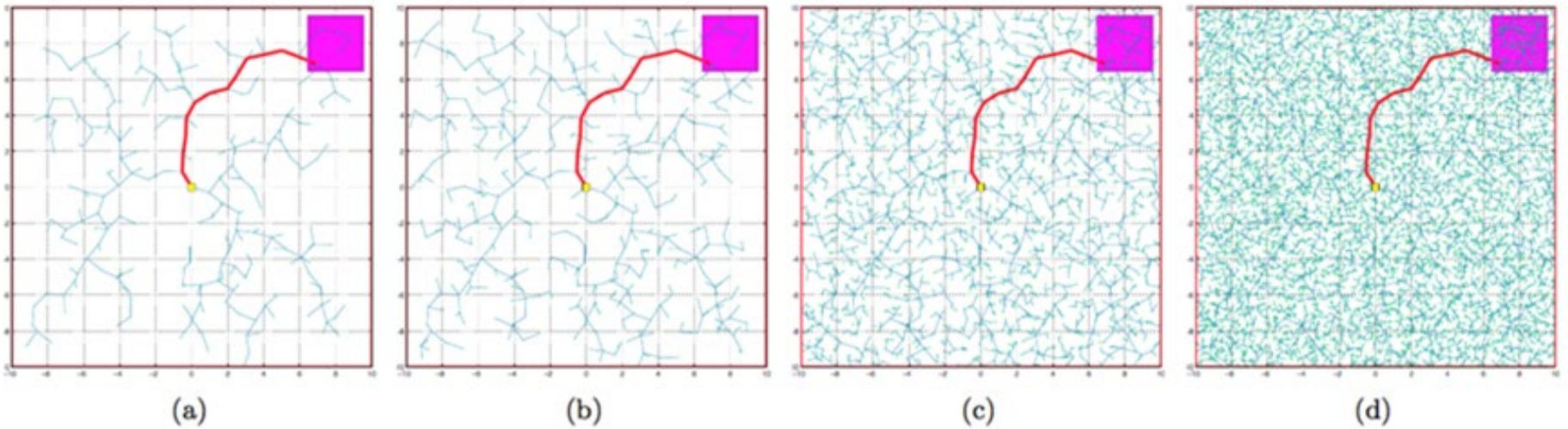
$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $i \leftarrow 0$;
while $i < N$ **do**
  $G \leftarrow (V, E)$;
  $x_{\text{rand}} \leftarrow \texttt{Sample}(i)$; $i \leftarrow i + 1$;
  $(V, E) \leftarrow \texttt{Extend}(G, x_{\text{rand}})$;

ExtendRRT(G,x):

$V' \leftarrow V$; $E' \leftarrow E$;
$x_{\text{nearest}} \leftarrow \texttt{Nearest}(G, x)$;
$x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x)$;
if $\texttt{ObstacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
  $V' \leftarrow V' \cup \{x_{\text{new}}\}$;
  $E' \leftarrow E' \cup \{(x_{\text{nearest}}, x_{\text{new}})\}$;
return $G' = (V', E')$



What happens if we run RRT for more iterations



RRT is Suboptimal [2, Theorem 33]

The cost of the best solution returned by RRT converges to a suboptimal value, with probability one:

$$\mathbb{P}\left(\left\{\lim_{n\to\infty} Y_n^{RRT} > c^*\right\}\right) = 1.$$

Asymptotically Optimal Planning

Probabilistically Complete

Given “long enough”, with probability 1,
a feasible solution is found if one exists

Asymptotically Optimal

Given “long enough”, with probability 1,
an optimal solution is found if one exists

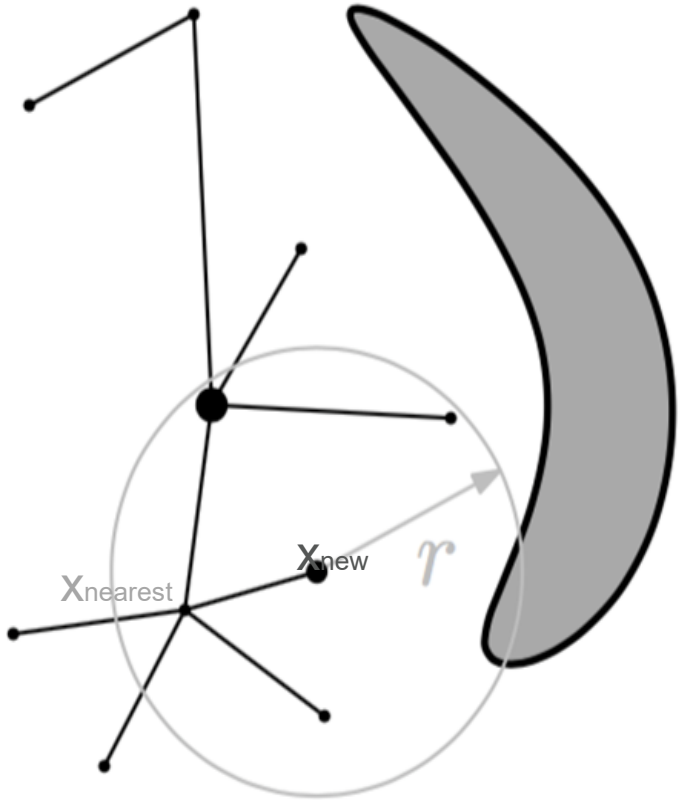
RRT* Extend – Find nearest node on tree

ExtendRRT*(G,x):

$V' \leftarrow V$; $E' \leftarrow E$;
$x_{\text{nearest}} \leftarrow \texttt{Nearest}(G, x)$;
$x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x)$;
if $\texttt{ObstacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
  $V' \leftarrow V' \cup \{x_{\text{new}}\}$;
  $x_{\min} \leftarrow x_{\text{nearest}}$;
  $X_{\text{near}} \leftarrow \texttt{Near}(G, x_{\text{new}}, |V|)$;
  for *all* $x_{\text{near}} \in X_{\text{near}}$ **do**
    if $\texttt{ObstacleFree}(x_{\text{near}}, x_{\text{new}})$ **then**
      $c' \leftarrow \texttt{Cost}(x_{\text{near}}) + c(\texttt{Line}(x_{\text{near}}, x_{\text{new}}))$;
      if $c' < \texttt{Cost}(x_{\text{new}})$ **then**
        $x_{\min} \leftarrow x_{\text{near}}$;
  $E' \leftarrow E' \cup \{(x_{\min}, x_{\text{new}})\}$;
  for *all* $x_{near} \in X_{\text{near}} \setminus \{x_{\min}\}$ **do**
    if $\texttt{ObstacleFree}(x_{\text{new}}, x_{\text{near}})$ *and*
    $\texttt{Cost}(x_{\text{near}}) >$
    $\texttt{Cost}(x_{\text{new}}) + c(\texttt{Line}(x_{\text{new}}, x_{\text{near}}))$ **then**
      $x_{\text{parent}} \leftarrow \texttt{Parent}(x_{\text{near}})$;
      $E' \leftarrow E' \setminus \{(x_{\text{parent}}, x_{\text{near}})\}$;
      $E' \leftarrow E' \cup \{(x_{\text{new}}, x_{\text{near}})\}$;
return $G' = (V', E')$

Karaman, Frazzoli. 2010.

Graphic from Gammell, 2016. Thesis.



define $\texttt{Near}(G, x, n)$ to be the set of all vertices within the closed ball of radius r_n centered at x, where

$$r_n = \min\left\{\left(\frac{\gamma}{\zeta_d}\frac{\log n}{n}\right)^{1/d}, \eta\right\},$$

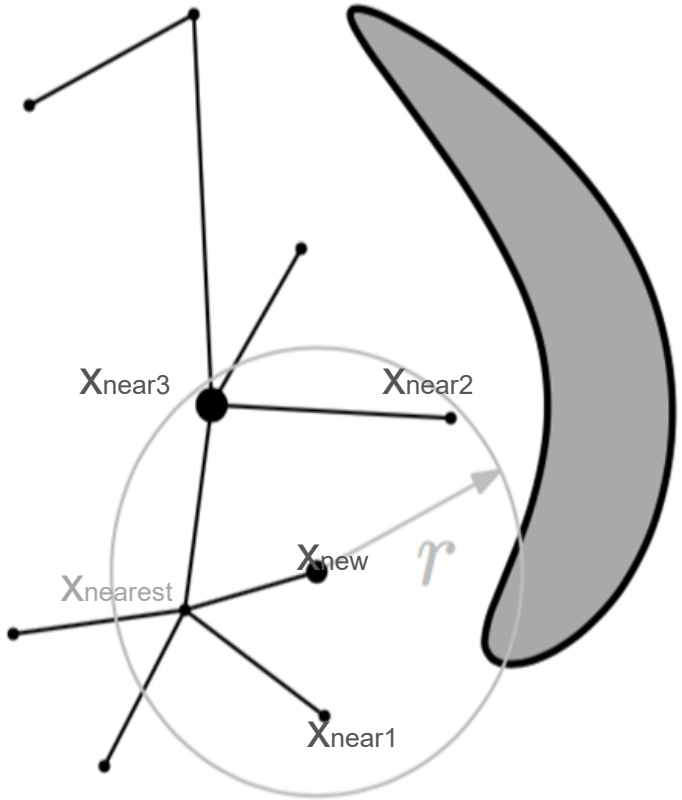
RRT* Extend –find r-neighbors of x new

ExtendRRT*(G,x):

$V' \leftarrow V$; $E' \leftarrow E$;
$x_{\text{nearest}} \leftarrow \texttt{Nearest}(G, x)$;
$x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x)$;
if $\texttt{ObstacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
  $V' \leftarrow V' \cup \{x_{\text{new}}\}$;
  $x_{\min} \leftarrow x_{\text{nearest}}$;
  $X_{\text{near}} \leftarrow \texttt{Near}(G, x_{\text{new}}, |V|)$;
  for *all* $x_{\text{near}} \in X_{\text{near}}$ **do**
    if $\texttt{ObstacleFree}(x_{\text{near}}, x_{\text{new}})$ **then**
      $c' \leftarrow \texttt{Cost}(x_{\text{near}}) + c(\texttt{Line}(x_{\text{near}}, x_{\text{new}}))$;
      if $c' < \texttt{Cost}(x_{\text{new}})$ **then**
        $x_{\min} \leftarrow x_{\text{near}}$;
  $E' \leftarrow E' \cup \{(x_{\min}, x_{\text{new}})\}$;
  for *all* $x_{near} \in X_{\text{near}} \setminus \{x_{\min}\}$ **do**
    if $\texttt{ObstacleFree}(x_{\text{new}}, x_{\text{near}})$ *and*
    $\texttt{Cost}(x_{\text{near}}) >$
    $\texttt{Cost}(x_{\text{new}}) + c(\texttt{Line}(x_{\text{new}}, x_{\text{near}}))$ **then**
      $x_{\text{parent}} \leftarrow \texttt{Parent}(x_{\text{near}})$;
      $E' \leftarrow E' \setminus \{(x_{\text{parent}}, x_{\text{near}})\}$;
      $E' \leftarrow E' \cup \{(x_{\text{new}}, x_{\text{near}})\}$;
return $G' = (V', E')$

Karaman, Frazzoli. 2010.

Graphic from Gammell, 2016. Thesis.



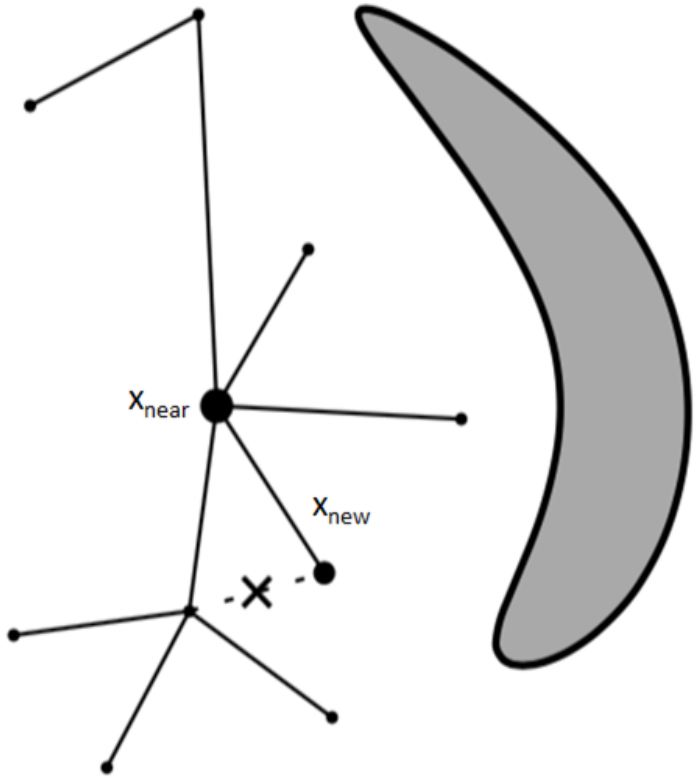
RRT* Extend – check if any neighbor is better and rewire

ExtendRRT*(G,x):

$V' \leftarrow V$; $E' \leftarrow E$;
$x_{\text{nearest}} \leftarrow \texttt{Nearest}(G, x)$;
$x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x)$;
if $\texttt{ObstacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
  $V' \leftarrow V' \cup \{x_{\text{new}}\}$;
  $x_{\min} \leftarrow x_{\text{nearest}}$;
  $X_{\text{near}} \leftarrow \texttt{Near}(G, x_{\text{new}}, |V|)$;
  for *all* $x_{\text{near}} \in X_{\text{near}}$ **do**
    if $\texttt{ObstacleFree}(x_{\text{near}}, x_{\text{new}})$ **then**
      $c' \leftarrow \texttt{Cost}(x_{\text{near}}) + c(\texttt{Line}(x_{\text{near}}, x_{\text{new}}))$;
      if $c' < \texttt{Cost}(x_{\text{new}})$ **then**
        $x_{\min} \leftarrow x_{\text{near}}$;
  $E' \leftarrow E' \cup \{(x_{\min}, x_{\text{new}})\}$;
  for *all* $x_{near} \in X_{\text{near}} \setminus \{x_{\min}\}$ **do**
    if $\texttt{ObstacleFree}(x_{\text{new}}, x_{\text{near}})$ *and*
    $\texttt{Cost}(x_{\text{near}}) >$
    $\texttt{Cost}(x_{\text{new}}) + c(\texttt{Line}(x_{\text{new}}, x_{\text{near}}))$ **then**
      $x_{\text{parent}} \leftarrow \texttt{Parent}(x_{\text{near}})$;
      $E' \leftarrow E' \setminus \{(x_{\text{parent}}, x_{\text{near}})\}$;
      $E' \leftarrow E' \cup \{(x_{\text{new}}, x_{\text{near}})\}$;
return $G' = (V', E')$

Karaman, Frazzoli. 2010.

Graphic from Gammell, 2016. Thesis.

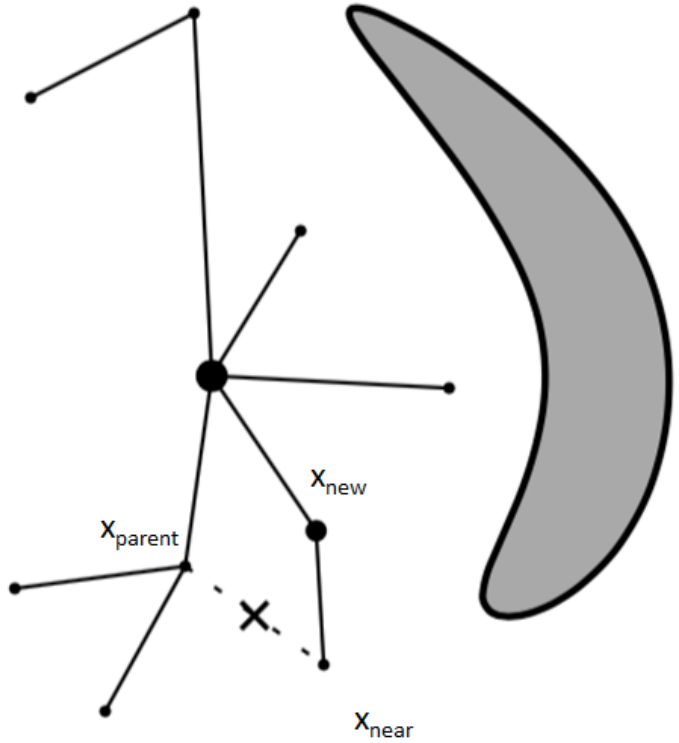


RRT* Extend – Possibly rewire the rest of the neighbors

$V' \leftarrow V$; $E' \leftarrow E$;
$x_{\text{nearest}} \leftarrow \texttt{Nearest}(G, x)$;
$x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x)$;
if $\texttt{ObstacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
  $V' \leftarrow V' \cup \{x_{\text{new}}\}$;
  $x_{\min} \leftarrow x_{\text{nearest}}$;
  $X_{\text{near}} \leftarrow \texttt{Near}(G, x_{\text{new}}, |V|)$;
  for *all* $x_{\text{near}} \in X_{\text{near}}$ **do**
    if $\texttt{ObstacleFree}(x_{\text{near}}, x_{\text{new}})$ **then**
      $c' \leftarrow \texttt{Cost}(x_{\text{near}}) + c(\texttt{Line}(x_{\text{near}}, x_{\text{new}}))$;
      if $c' < \texttt{Cost}(x_{\text{new}})$ **then**
        $x_{\min} \leftarrow x_{\text{near}}$;
  $E' \leftarrow E' \cup \{(x_{\min}, x_{\text{new}})\}$;
  for *all* $x_{near} \in X_{\text{near}} \setminus \{x_{\min}\}$ **do**
    if $\texttt{ObstacleFree}(x_{\text{new}}, x_{\text{near}})$ *and*
    $\texttt{Cost}(x_{\text{near}}) >$
    $\texttt{Cost}(x_{\text{new}}) + c(\texttt{Line}(x_{\text{new}}, x_{\text{near}}))$ **then**
      $x_{\text{parent}} \leftarrow \texttt{Parent}(x_{\text{near}})$;
      $E' \leftarrow E' \setminus \{(x_{\text{parent}}, x_{\text{near}})\}$;
      $E' \leftarrow E' \cup \{(x_{\text{new}}, x_{\text{near}})\}$;
return $G' = (V', E')$

Karaman, Frazzoli. 2010.

Graphic from Gammell, 2016. Thesis.

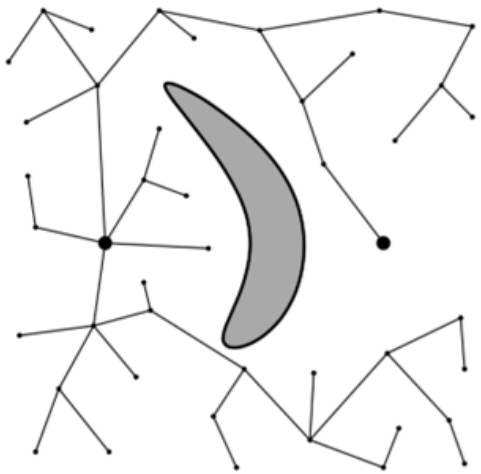


RRT* Extend vs RRT Extend

$V' \leftarrow V$; $E' \leftarrow E$;
$x_{\text{nearest}} \leftarrow \texttt{Nearest}(G, x)$;
$x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x)$;
if $\texttt{ObstacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
  $V' \leftarrow V' \cup \{x_{\text{new}}\}$;
  $x_{\min} \leftarrow x_{\text{nearest}}$;
  $X_{\text{near}} \leftarrow \texttt{Near}(G, x_{\text{new}}, |V|)$;
  for *all* $x_{\text{near}} \in X_{\text{near}}$ **do**
    if $\texttt{ObstacleFree}(x_{\text{near}}, x_{\text{new}})$ **then**
      $c' \leftarrow \texttt{Cost}(x_{\text{near}}) + c(\texttt{Line}(x_{\text{near}}, x_{\text{new}}))$;
      if $c' < \texttt{Cost}(x_{\text{new}})$ **then**
        $x_{\min} \leftarrow x_{\text{near}}$;
  $E' \leftarrow E' \cup \{(x_{\min}, x_{\text{new}})\}$;
  for *all* $x_{near} \in X_{\text{near}} \setminus \{x_{\min}\}$ **do**
    if $\texttt{ObstacleFree}(x_{\text{new}}, x_{\text{near}})$ *and*
    $\texttt{Cost}(x_{\text{near}}) >$
    $\texttt{Cost}(x_{\text{new}}) + c(\texttt{Line}(x_{\text{new}}, x_{\text{near}}))$ **then**
      $x_{\text{parent}} \leftarrow \texttt{Parent}(x_{\text{near}})$;
      $E' \leftarrow E' \setminus \{(x_{\text{parent}}, x_{\text{near}})\}$;
      $E' \leftarrow E' \cup \{(x_{\text{new}}, x_{\text{near}})\}$;
return $G' = (V', E')$

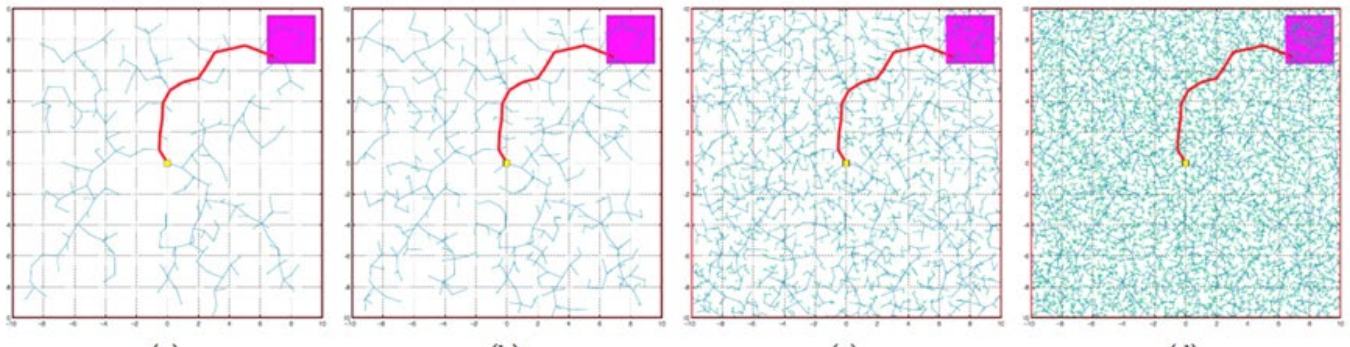
Karaman, Frazzoli. 2010.

$V' \leftarrow V$; $E' \leftarrow E$;
$x_{\text{nearest}} \leftarrow \texttt{Nearest}(G, x)$;
$x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x)$;
if $\texttt{ObstacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
  $V' \leftarrow V' \cup \{x_{\text{new}}\}$;
  $E' \leftarrow E' \cup \{(x_{\text{nearest}}, x_{\text{new}})\}$;
return $G' = (V', E')$

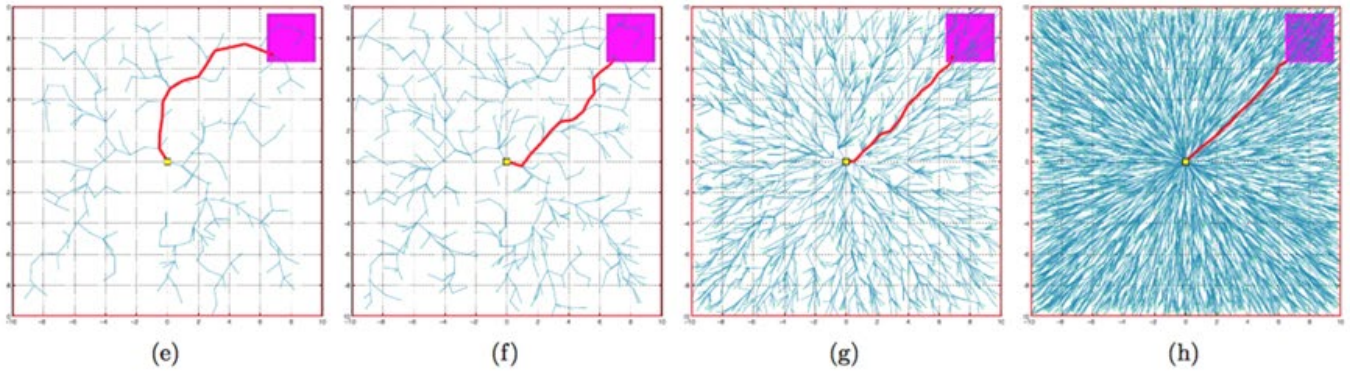


RRT*

RRT



RRT*

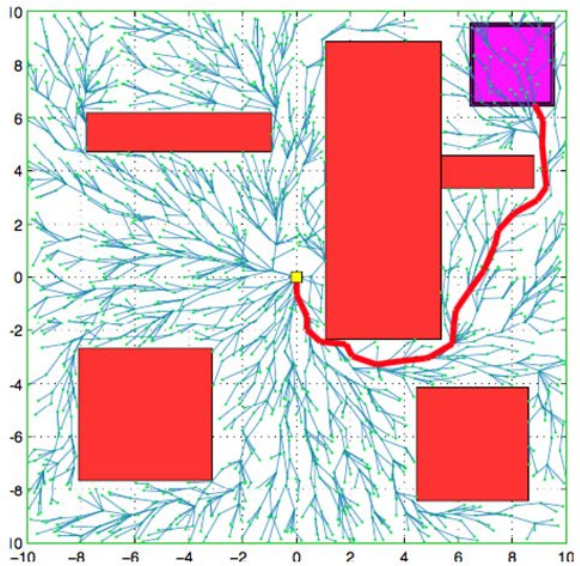


Karaman, Frazzoli. 2010.

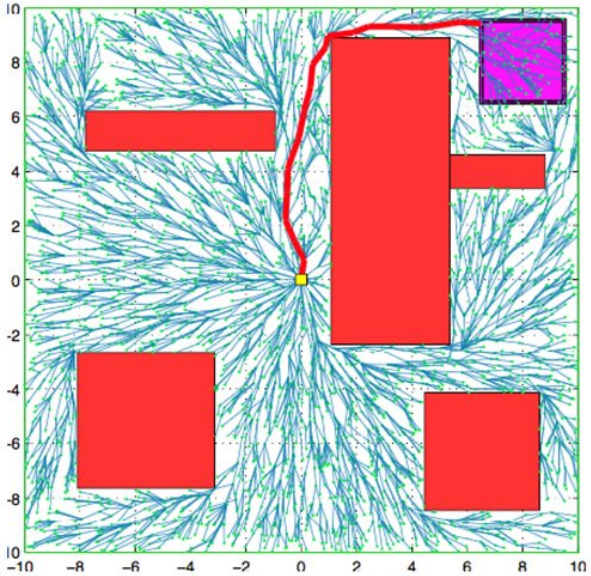
RRT*

RRT*

2500 Iterations



5000 Iterations



Karaman, Frazzoli. 2010.

PRM*

- "Simple" PRM

$V \leftarrow \{x_{\text{init}}\} \cup \{\texttt{SampleFree}_i\}_{i=1,\dots,n}$; $E \leftarrow \emptyset$;
foreach $v \in V$ **do**
  $U \leftarrow \texttt{Near}(G = (V, E), v, r) \setminus \{v\}$;
  foreach $u \in U$ **do**
    if $\texttt{CollisionFree}(v, u)$ **then** $E \leftarrow E \cup \{(v, u), (u, v)\}$
return $G = (V, E)$;

- PRM*
  - Radius shrinks with more vertices
  - One change, and its' optimal!

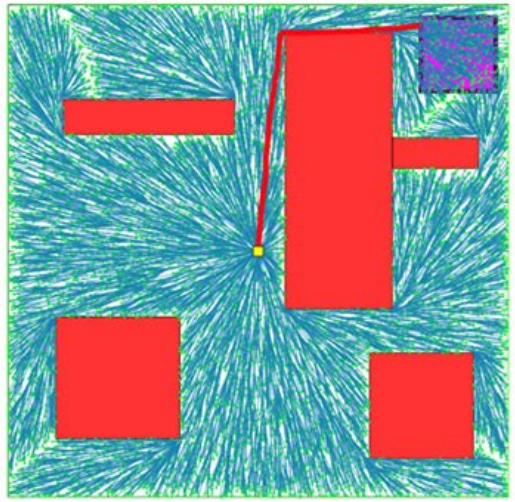
$V \leftarrow \{x_{\text{init}}\} \cup \{\texttt{SampleFree}_i\}_{i=1,\dots,n}$; $E \leftarrow \emptyset$;
foreach $v \in V$ **do**
  $U \leftarrow \texttt{Near}(G = (V, E), v, \gamma_{\text{PRM}}(\log(n)/n)^{1/d}) \setminus \{v\}$;
  foreach $u \in U$ **do**
    if $\texttt{CollisionFree}(v, u)$ **then** $E \leftarrow E \cup \{(v, u), (u, v)\}$
return $G = (V, E)$;

RRT*: Practical?

There are 2 issues with RRT* convergence.

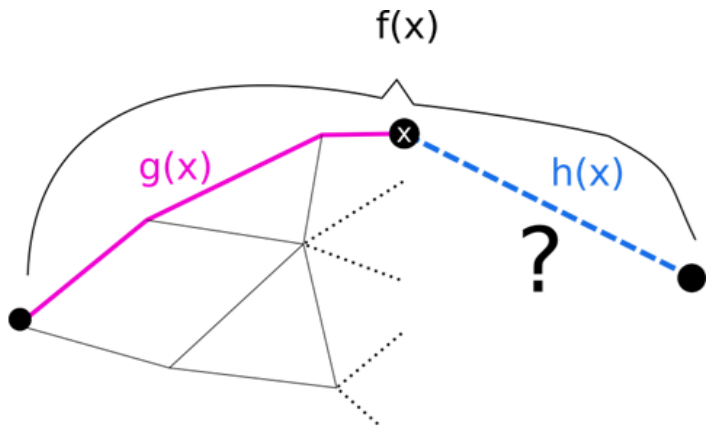
1. It does not use a heuristic, like A* to prioritize paths
2. It keeps sampling in regions when that will not improve the solution

How can we make it better?



A*

- Graph path-planning algorithm
- Similar to Dijkstra's, but adds a 'heuristic' cost from the current node to the goal
- Explore nodes in order of best cost (f(x))

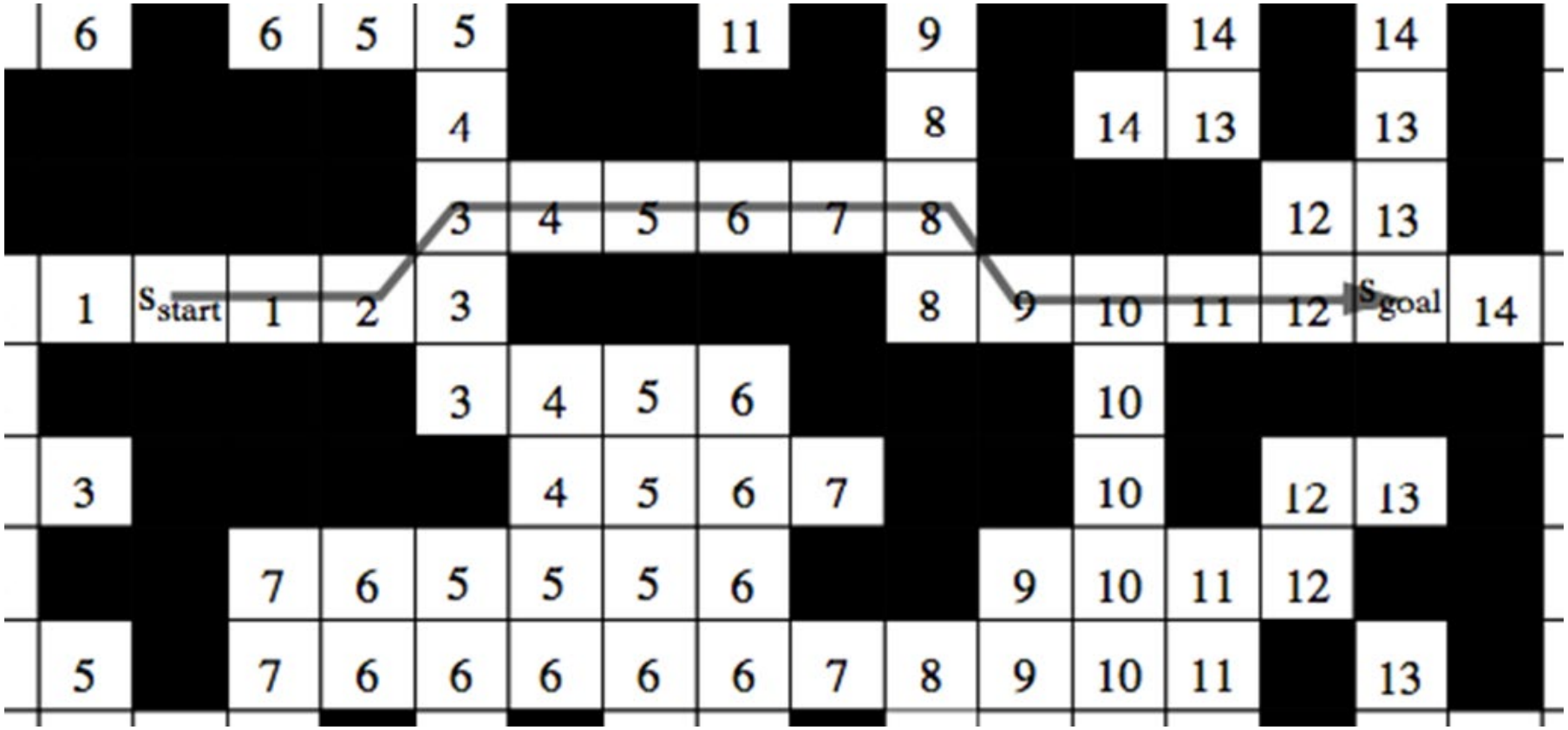


Lifelong Planning A* (LPA*)

- "Lifelong" because it saves and reuses information from past searches
- Designed for edge additions/deletions and changing weights
- Consistent nodes
  - saved g(x) from previous query is still the shortest distance among neighbors
- Inconsistent
  - edges were added, saved g(x) is no longer the shortest distance

Koenig S, Likhachev M, Furcy D. Lifelong planning A∗. Artificial Intelligence. 2004 May 1;155(1-2):93-146.

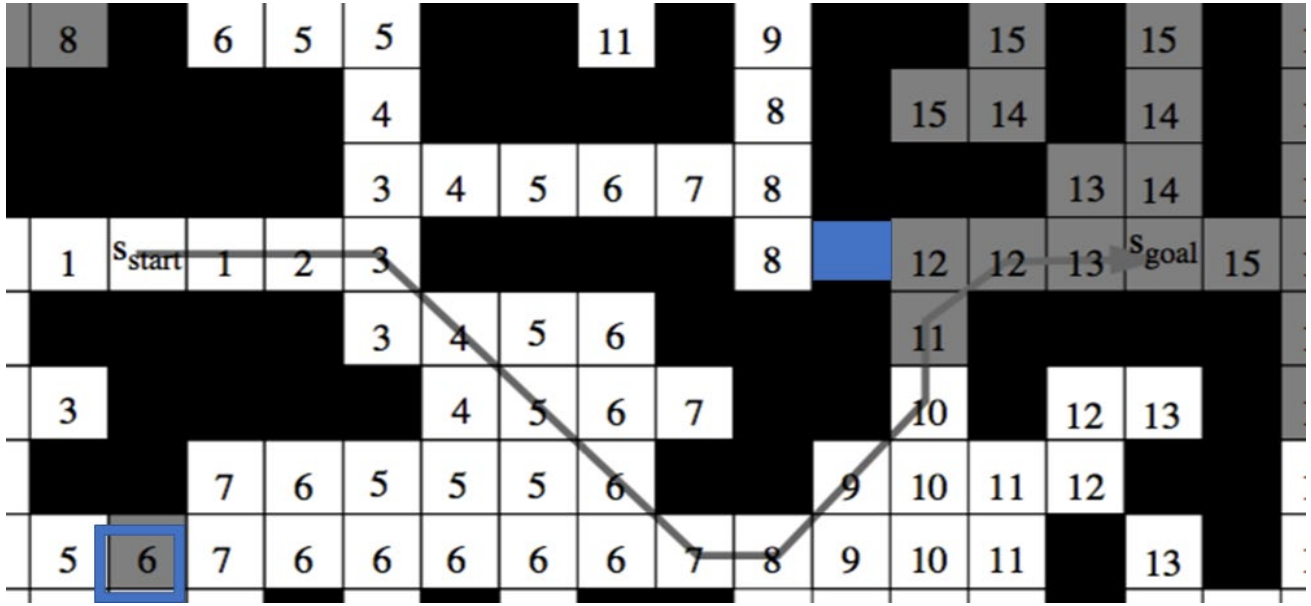
LPA* Example –



Recover path by starting at the goal and decreasing cost-to-come g(x) distance

Koenig S, Likhachev M, Furcy D. Lifelong planning A∗. Artificial Intelligence. 2004 May 1;155(1-2):93-146.

LPA* Example



Keep two estimates of start distance

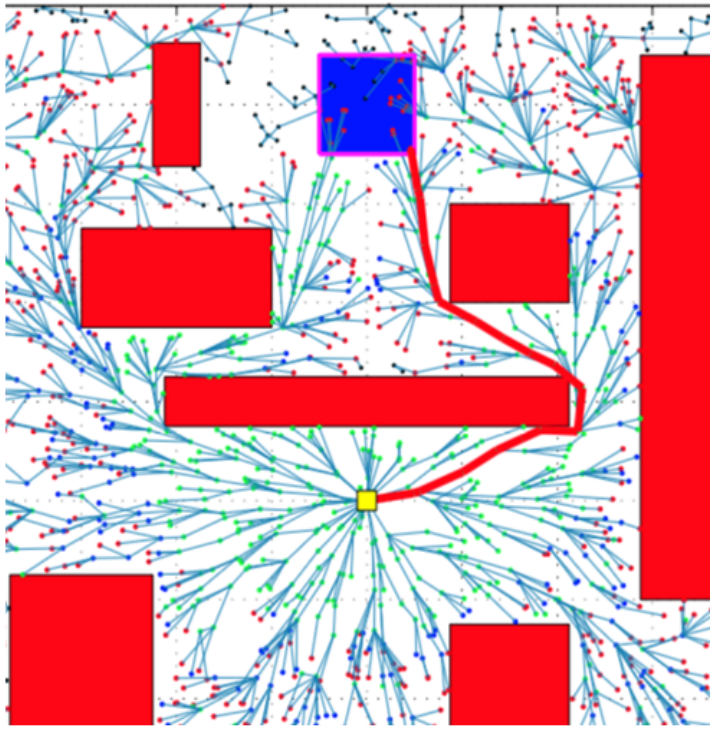
- **G-VALUE:** Cost-to-Go from previous (A*) search
- **RHS**: 1-step look-ahead [min(G value of neighbors)+cost]

If RHS and G-VALUE are the same the nodes are consistent, otherwise recalculation is needed (inconsistent)

Koenig S, Likhachev M, Furcy D. Lifelong planning A∗. Artificial Intelligence. 2004 May 1;155(1-2):93-146.

RRT#

- RRT with LPA* ensuring that the tree is always consistent
  - Promising nodes: where the estimated cost of a path through that node is less than the current best path
  - After adding a new node to the tree, updates parents and cost-to-come for promising nodes
  - You only have to rewire a small fraction of all nodes in the graph

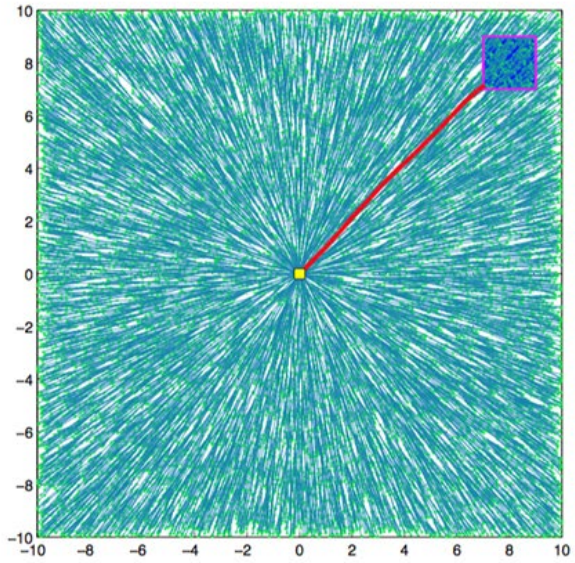


Arslan, Tsiotras. 2012

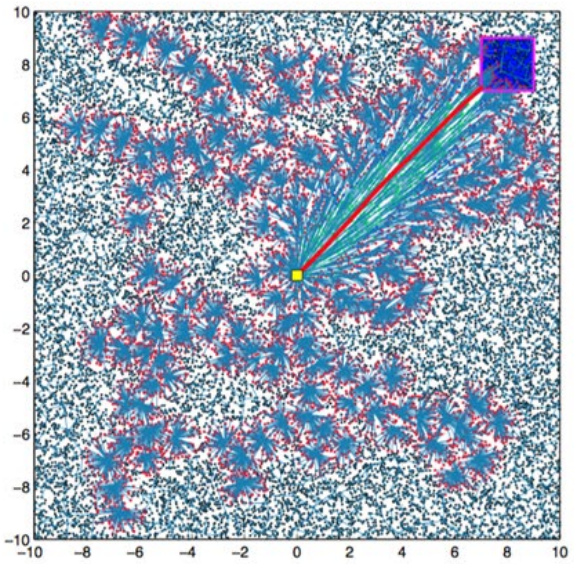
As this works only with cost-to-come costs, it does not have to be additive

RRT#

RRT* Tree Growth

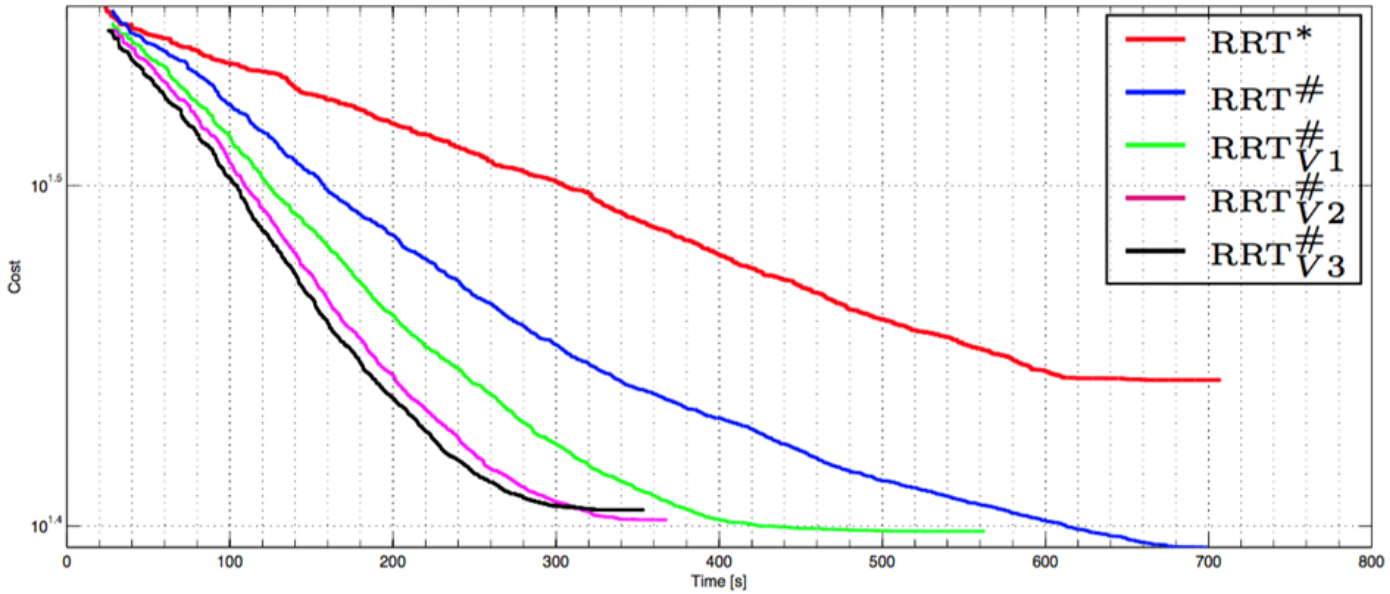


RRT# Tree Growth



Arslan, Tsiotras. 2012

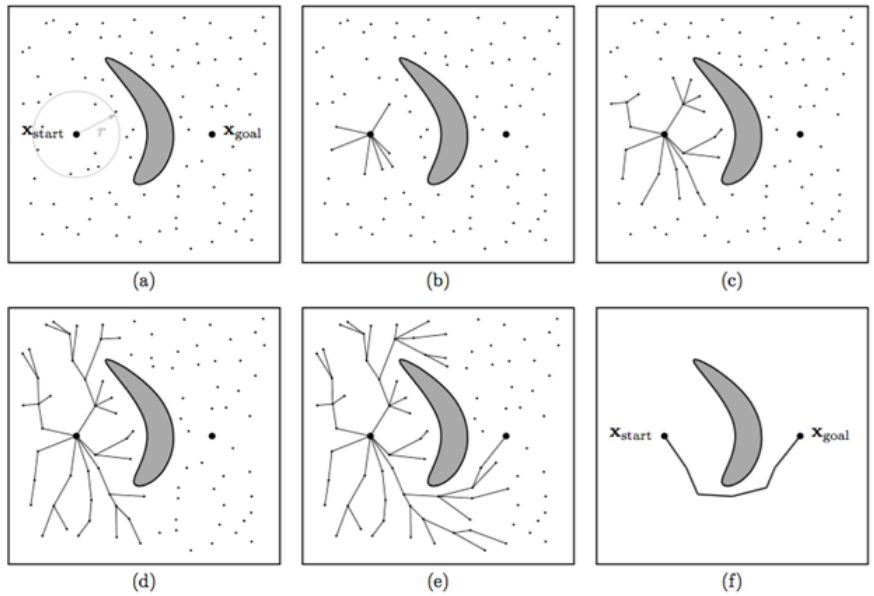
RRT# 5D space with Random Obstacles



Variants on RRT# only add new vertices to graph if:
1 = new could be promising
2 = parent is promising
3 = new is explicitly promising

Fast Marching Trees (FMT*)

- Samples all points upfront
  - Similar to Djikstra's
  - "Fast-marching": expands from the start node outwards with increasing 'cost-to-come' (g(x))
- Without obstacles, the result is identical to PRM*
  - FMT* will build a minimum spanning tree
  - All nodes/edges in this tree will be in PRM*'s roadmap

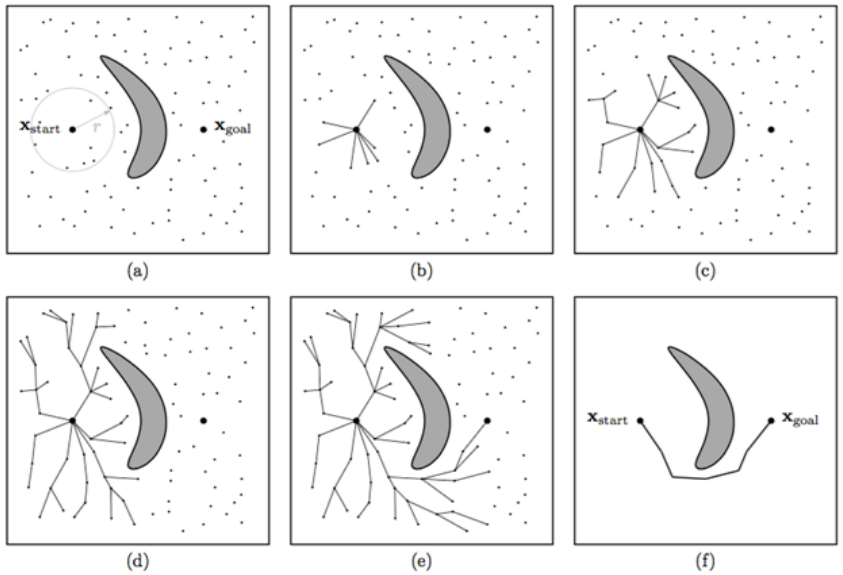


Graphic by Gammell, 2016

Jason, Pavone. 2013.

Fast Marching Trees (FMT*)

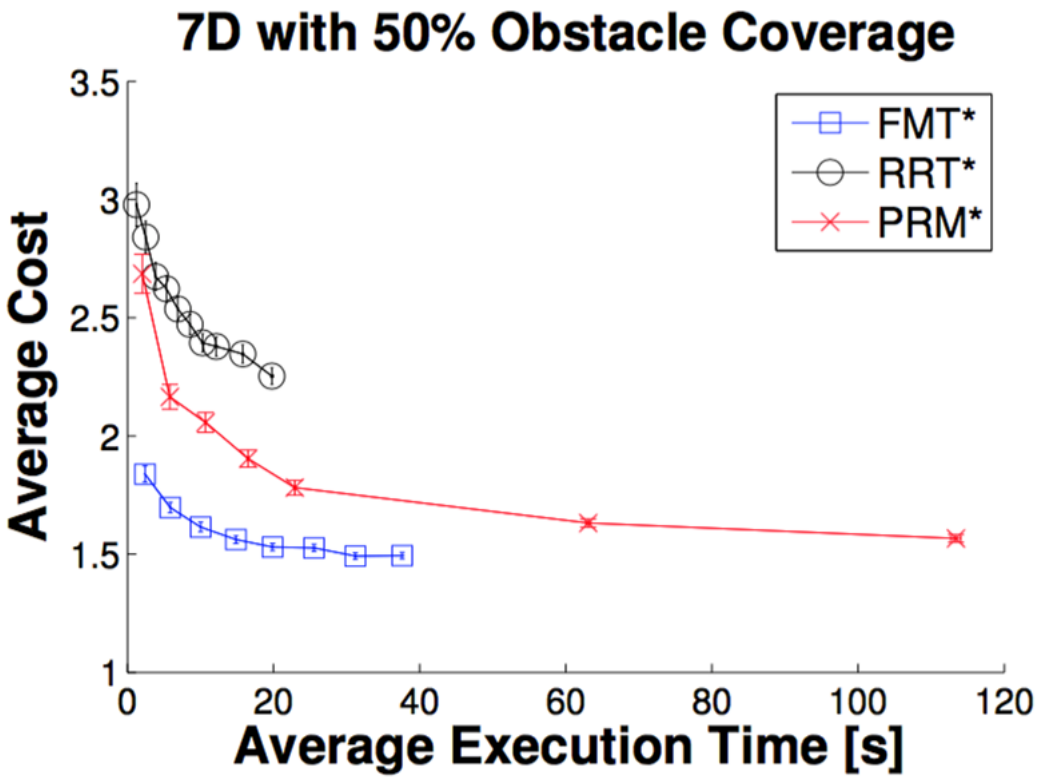
- Has a very small chance of making suboptimal connections
- Always terminates
- Asymptotically Optimal in terms of the number of samples, not running time



Graphic by Gammell, 2016

Jason, Pavone. 2013.

FMT* Results


7D with 50% Obstacle Coverage
FMT*
RRT*
PRM*
Average Cost
Average Execution Time [s]
3.5
3
2.5
2
1.5
1
0
20
40
60
80
100
120

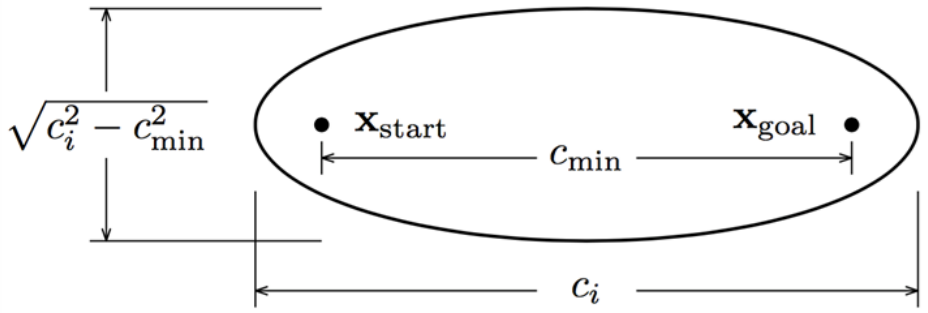
Informed RRT*

- Starts just like RRT* until a first solution is found
- Then, it only samples from a subset from an admissible heuristic
- More focused search
- RRT* finds optimal paths to every state in the tree. Not consistent with single-query nature.

Inform RRT*: L2 norm as admissible heuristic

- Informed RRT* tries to reduce the cost of rejection sampling
- To minimize path length in R^n, L^2-Norm can be used

$$X_{\hat{f}} = \left\{ \mathbf{x} \in X_{\text{free}} \;\middle|\; \|\mathbf{x} - \mathbf{x}_{\text{start}}\|_2 + \|\mathbf{x}_{\text{goal}} - \mathbf{x}\|_2 < c_i \right\}$$

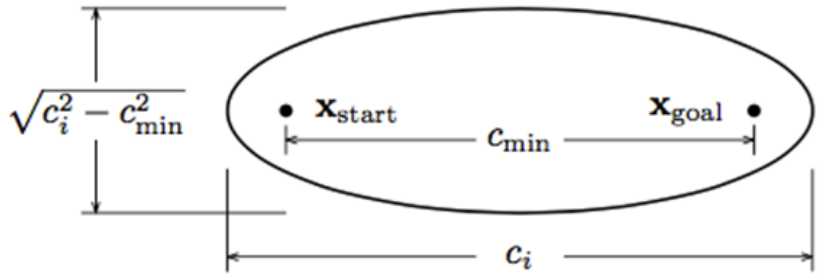


Gammell, 2016.

Inform RRT*: L2 norm and PHSs

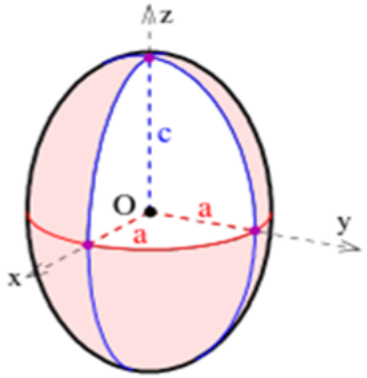
- To minimize path length in R^n, L^2-Norm can be used

$$\begin{aligned}\widehat{f} &= \widehat{g}(x) + \widehat{h}(x) \\ &= \|x - x_{start}\|_2 + \|x_{goal} - x\|_2\end{aligned}$$



Inform RRT*: L2 norm and PHSs

- L^2-Norm is the intersection of Free Space and a prolate hyper-spheroid (PHS)
- An ellipse in multiple dimensions



Prolate

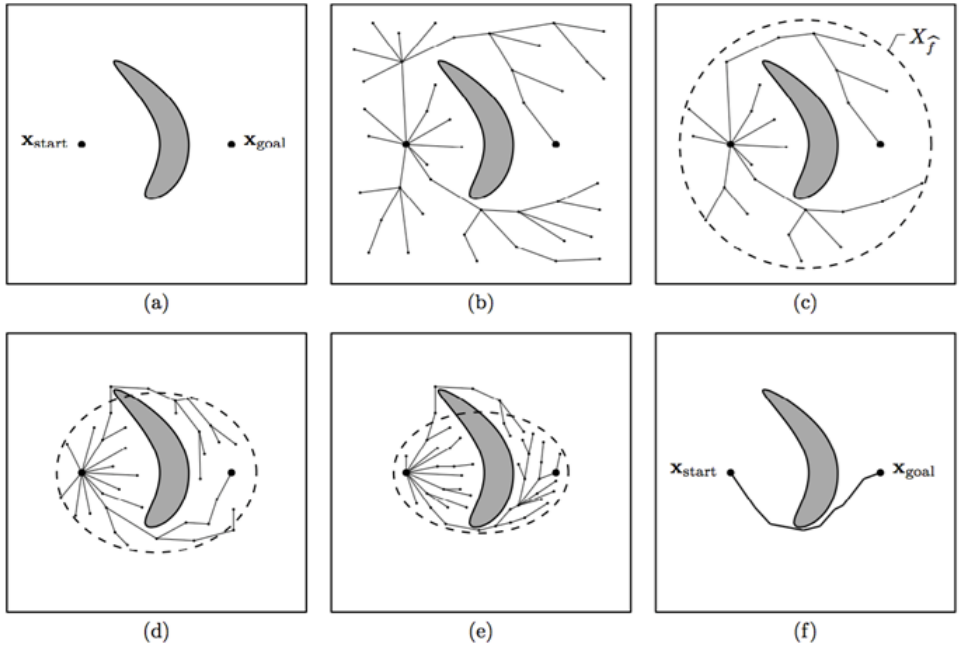
- The PHS can be sampled from directly: no rejection sampling, affected less by dimensionality increase

Gammell, 2016.

Image source: https://commons.wikimedia.org/wiki/File:Ellipsoid-rot-ax.svg

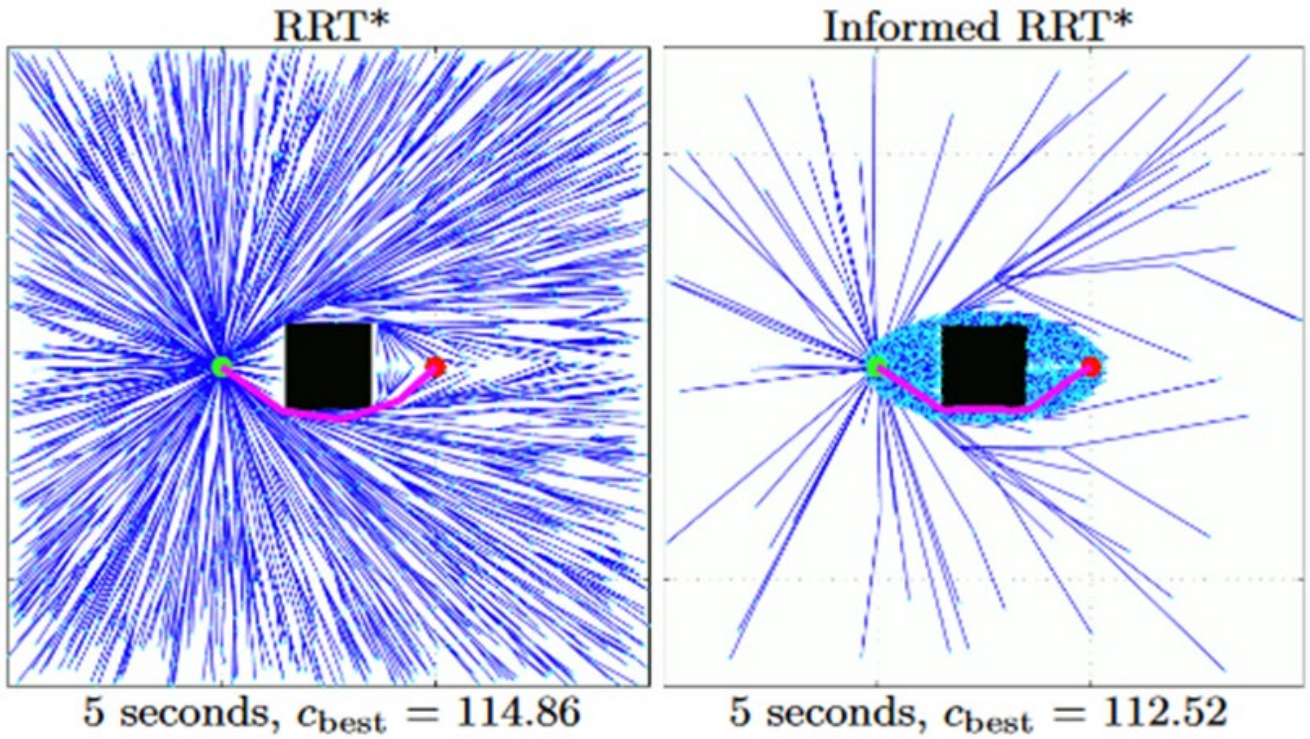
Informed RRT*

a) Start and Goal
b) RRT* until you find an initial path
c) Bound the problem with the L^2-Norm, prune nodes outside the bound
d) Sample from the PHS defining that L^2-Norm
e) Repeat until convergence
f) Final path

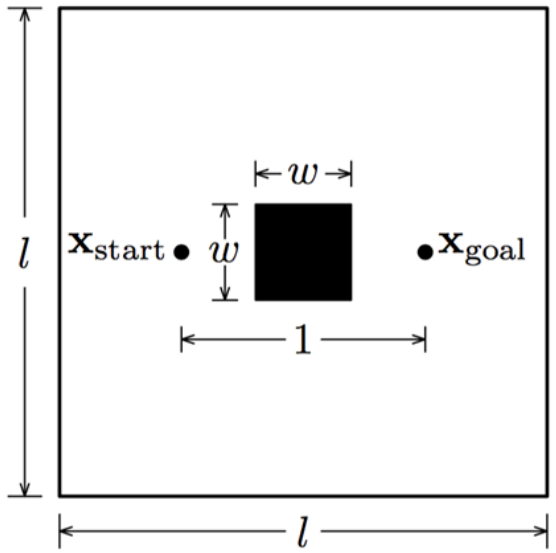


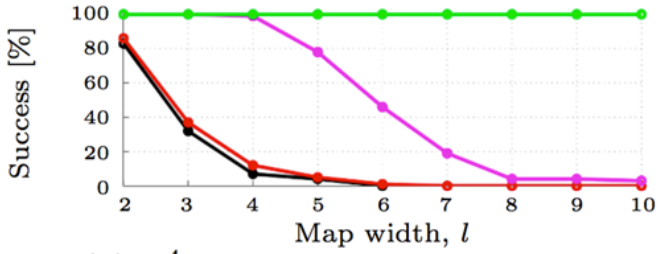
Gammell, 2016.

Informed RRT*

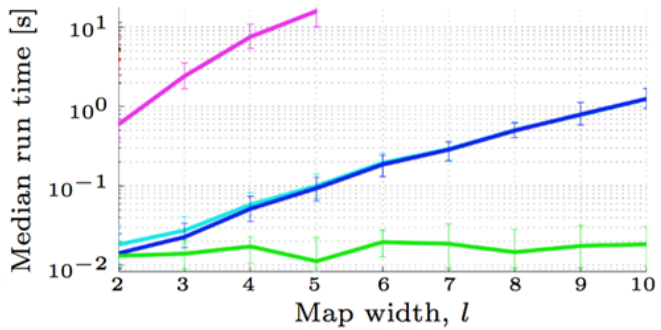


Informed RRT* Results





(b) $\mathbb{R}^4$: Success vs. map width



Informed RRT*

RRT*

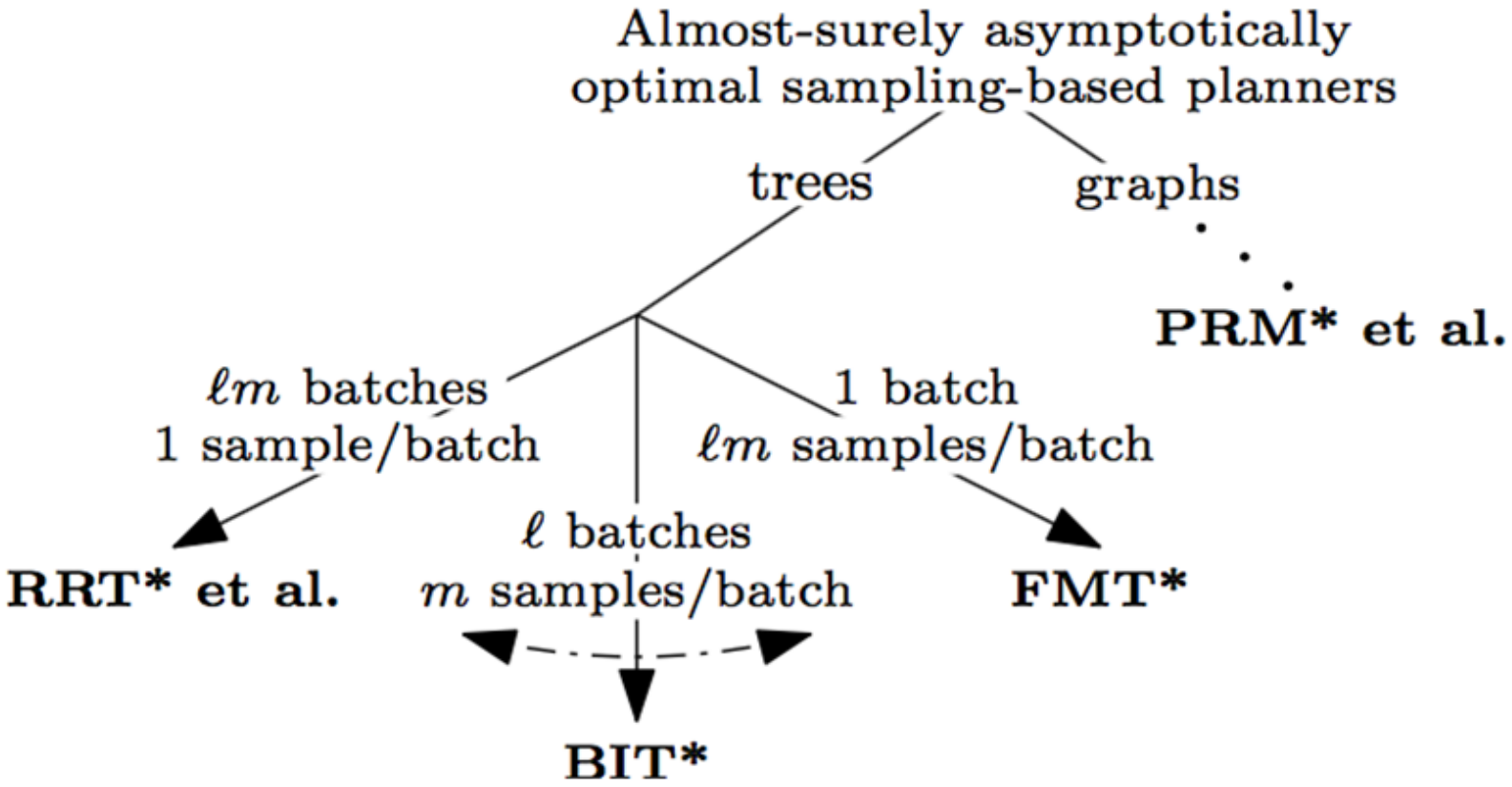
RRT* with Pruning

RRT* with Rejection Sampling

RRT* with Pruning + Rejection Sampling

Gammell, 2016.

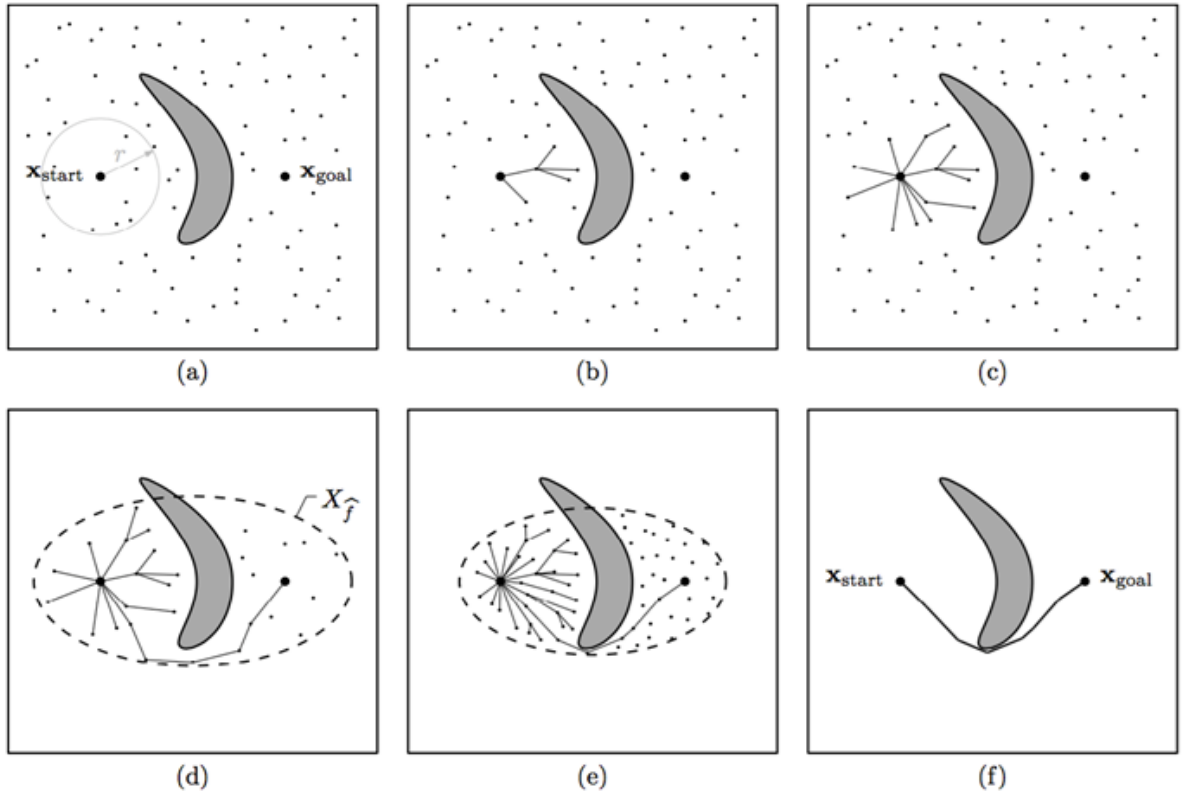
BIT*: The Most Generalized Tree



Gammell, 2016.

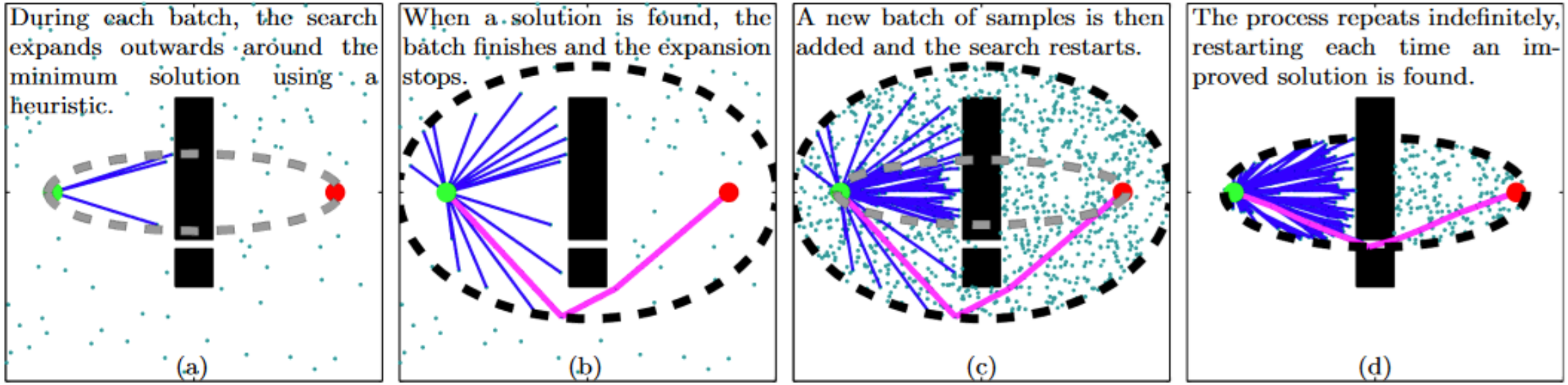
BIT*: Batch Informed Trees

- Creates an edge-implicit graph in *batches*
- Searches the graph in order of potential solution quality until no more samples or the solution can't be improved.
- Adds a batch from the informed set
  - old samples are pruned or reused

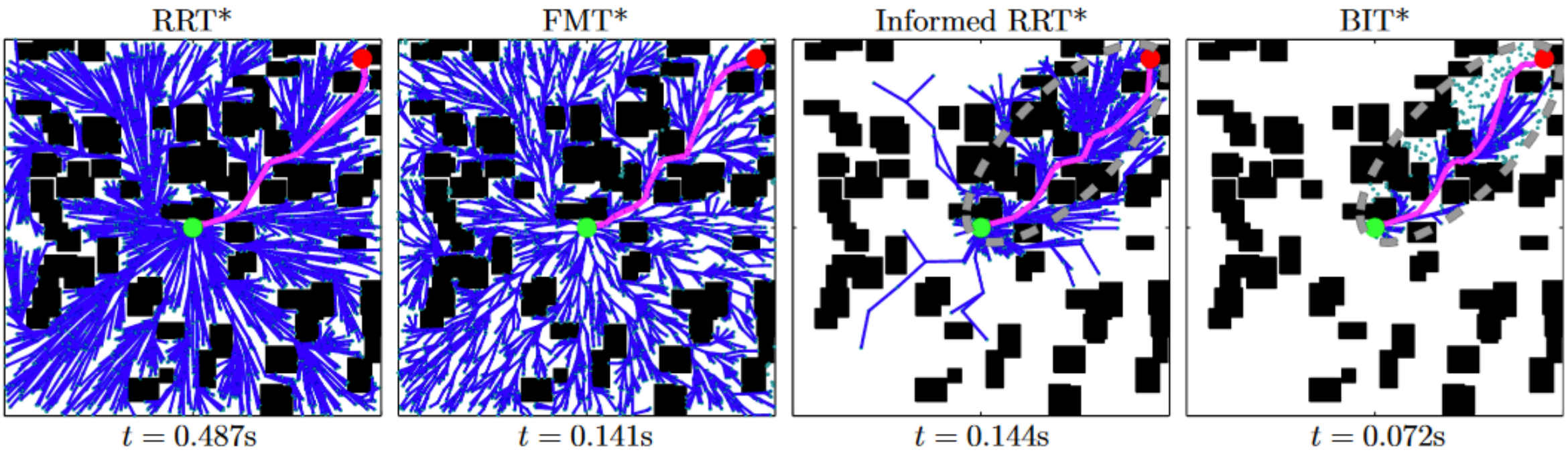


Gammell, 2016.

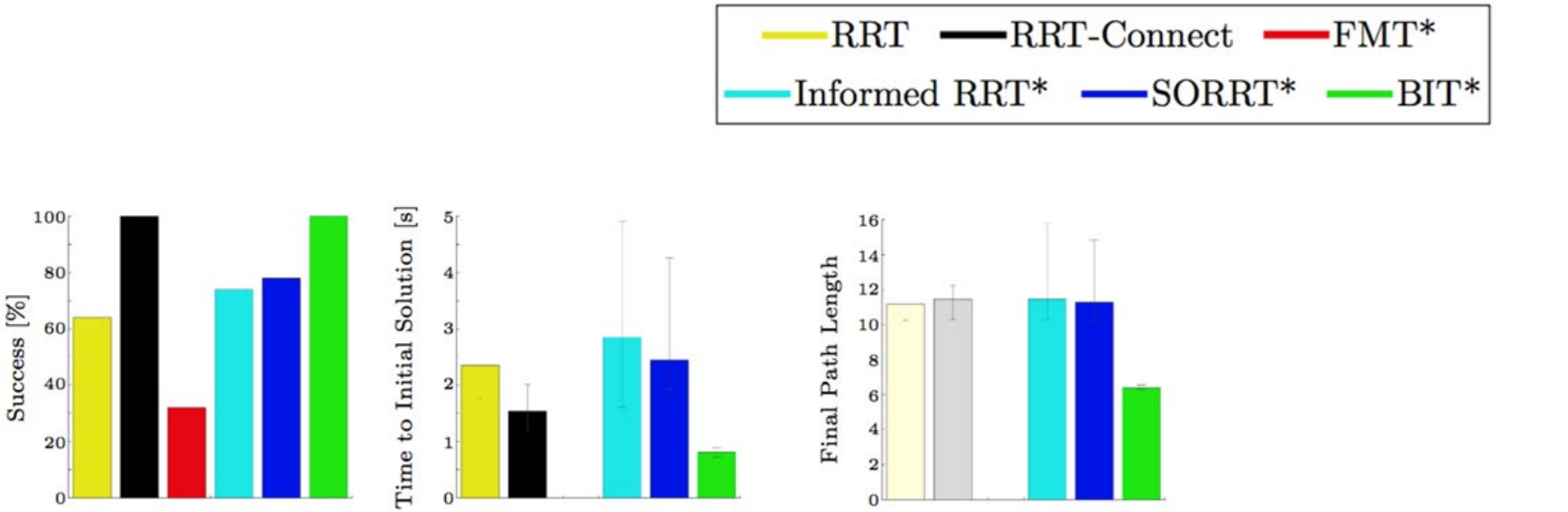
BIT*

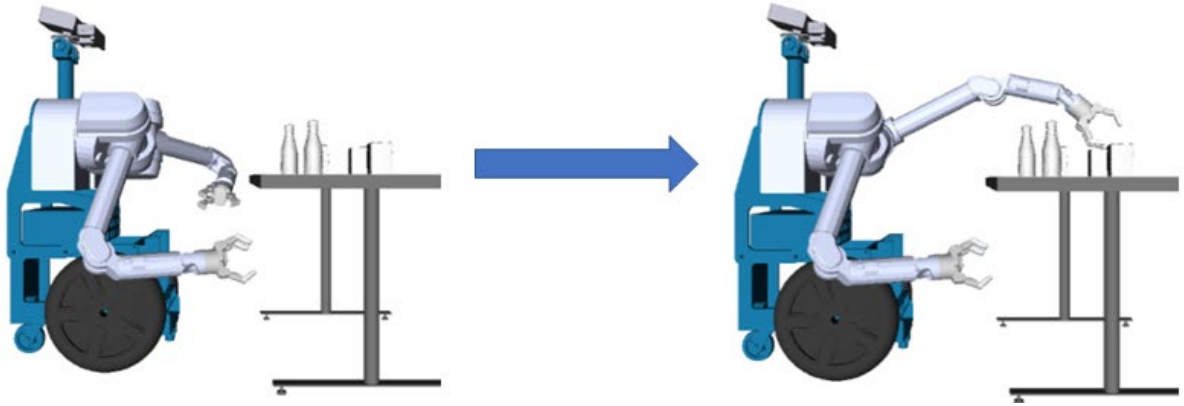

During each batch, the search expands outwards around the minimum solution using a heuristic.
(a)
When a solution is found, the batch finishes and the expansion stops.
(b)
A new batch of samples is then added and the search restarts.
(c)
The process repeats indefinitely, restarting each time an improved solution is found.
(d)

BIT*



BIT*: Results





Gammell, 2016.

Overview

- Asymptotic Optimality
- RRT*
- RRT#
- BIT*, FMT*, IRRT*
- Simplification Methods

This Lecture is based on

- [1]
- [2]
- [3]
- [4]