Dynamic Region-biased Rapidly-exploring Random Trees

by

Jory Denny, Read Sandstrom, Andrew Bregger, and Nancy M. Amato

University of Richmond, Richmond VA, USA

Texas A&M University, College Station, TX, USA

Presenter: Jae Won Choi

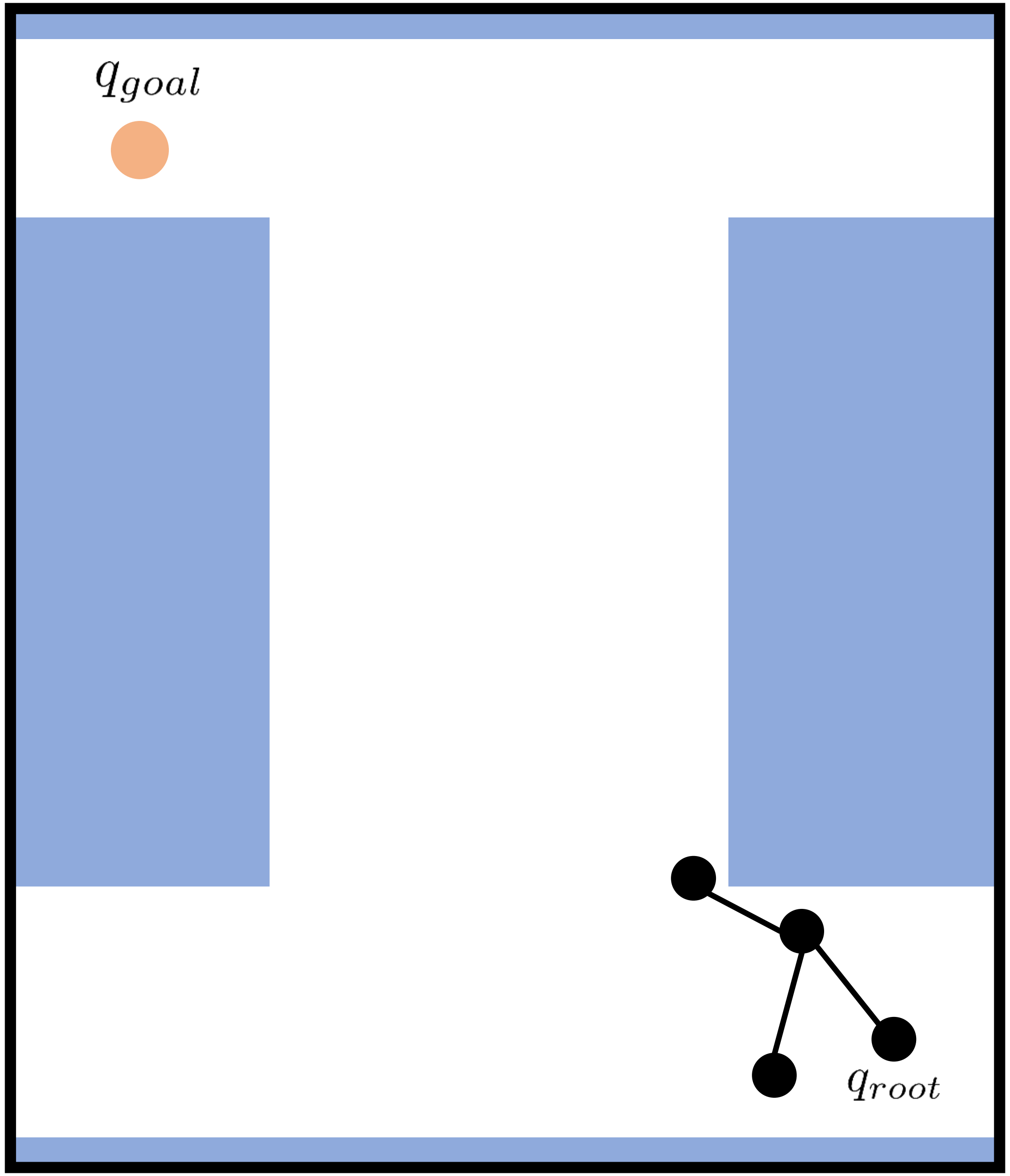
RRT Review

RRT Review

- RRT - Randomized Sampling

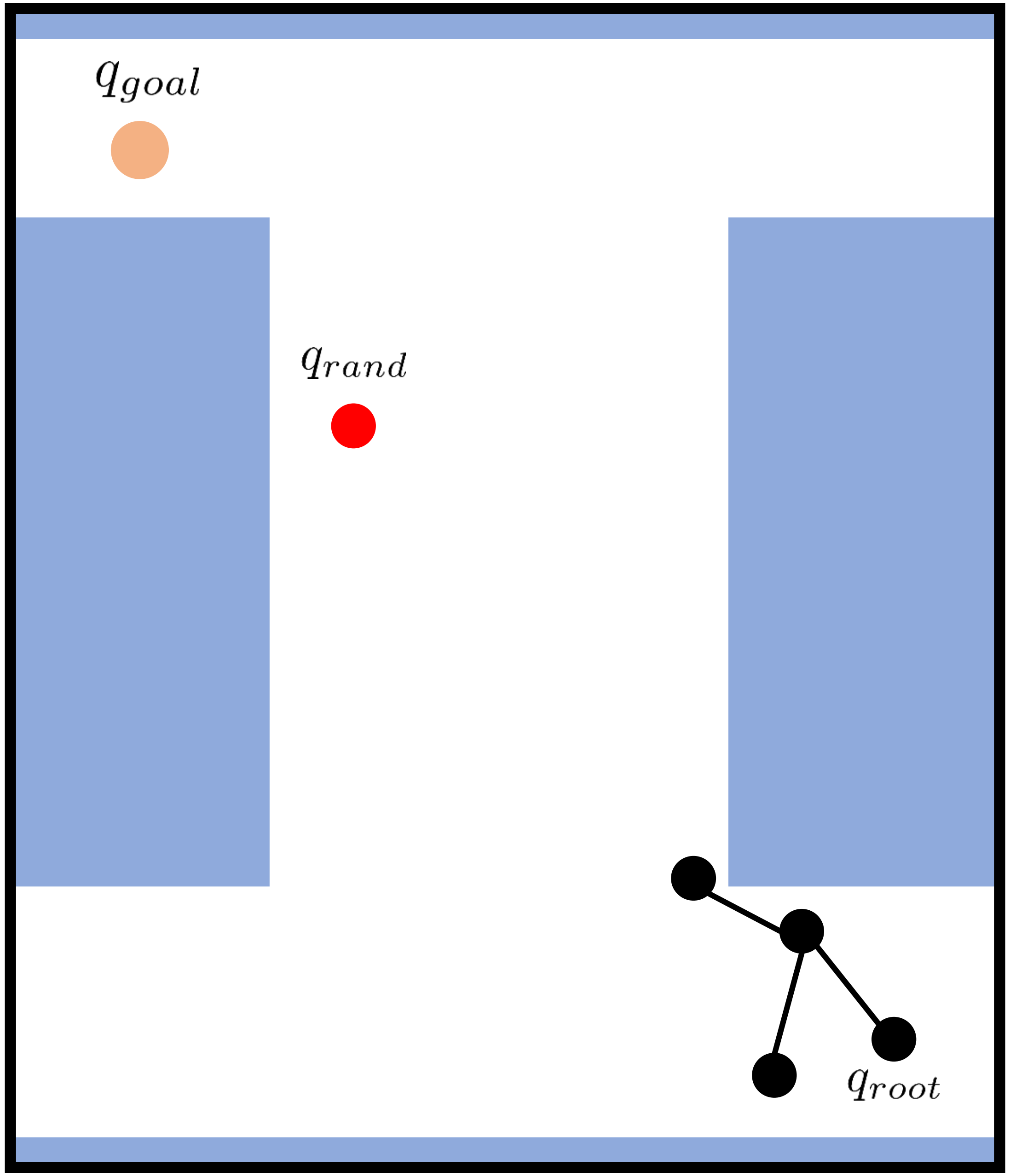
RRT Review

- RRT - Randomized Sampling



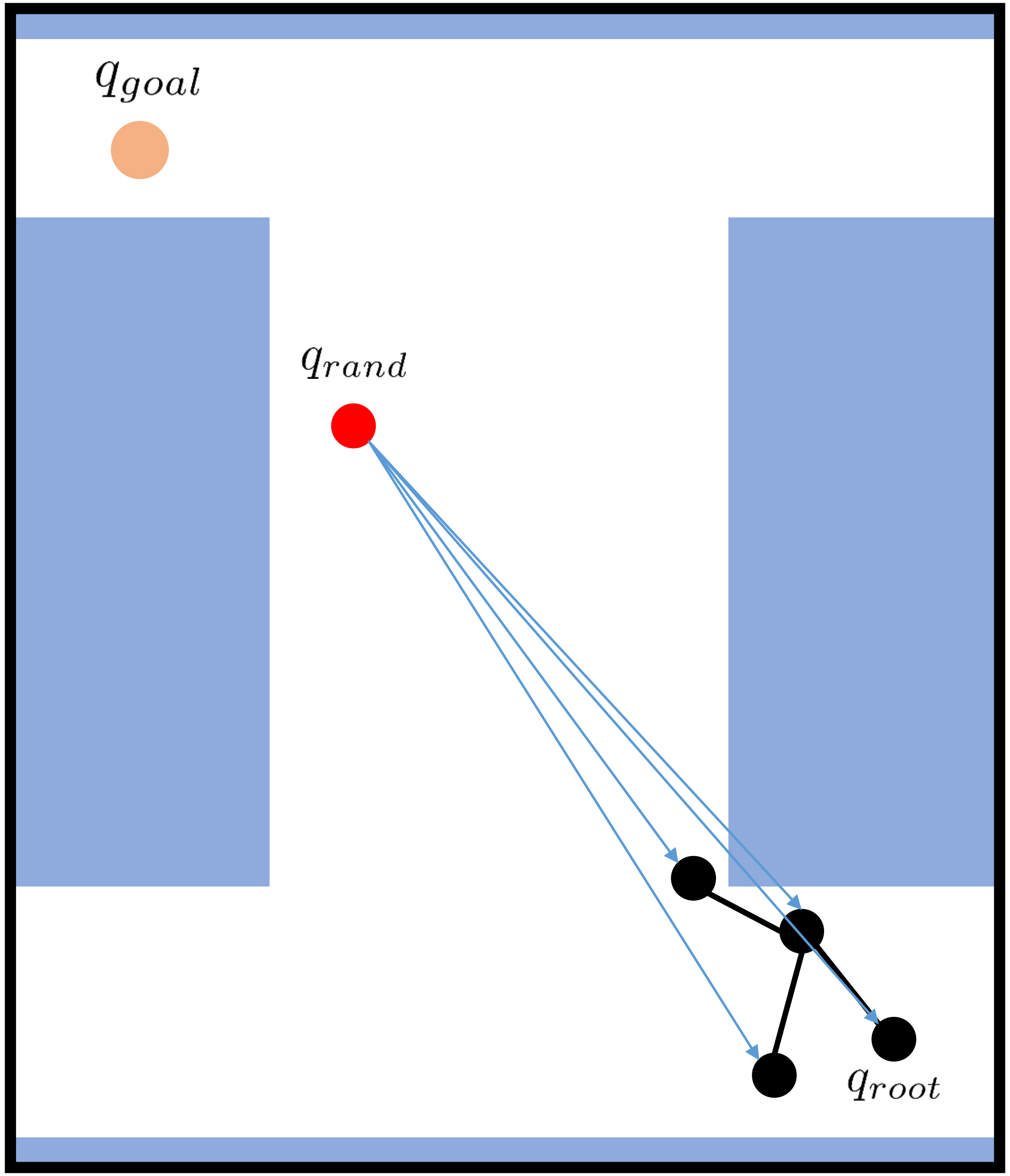
RRT Review

- RRT - Randomized Sampling



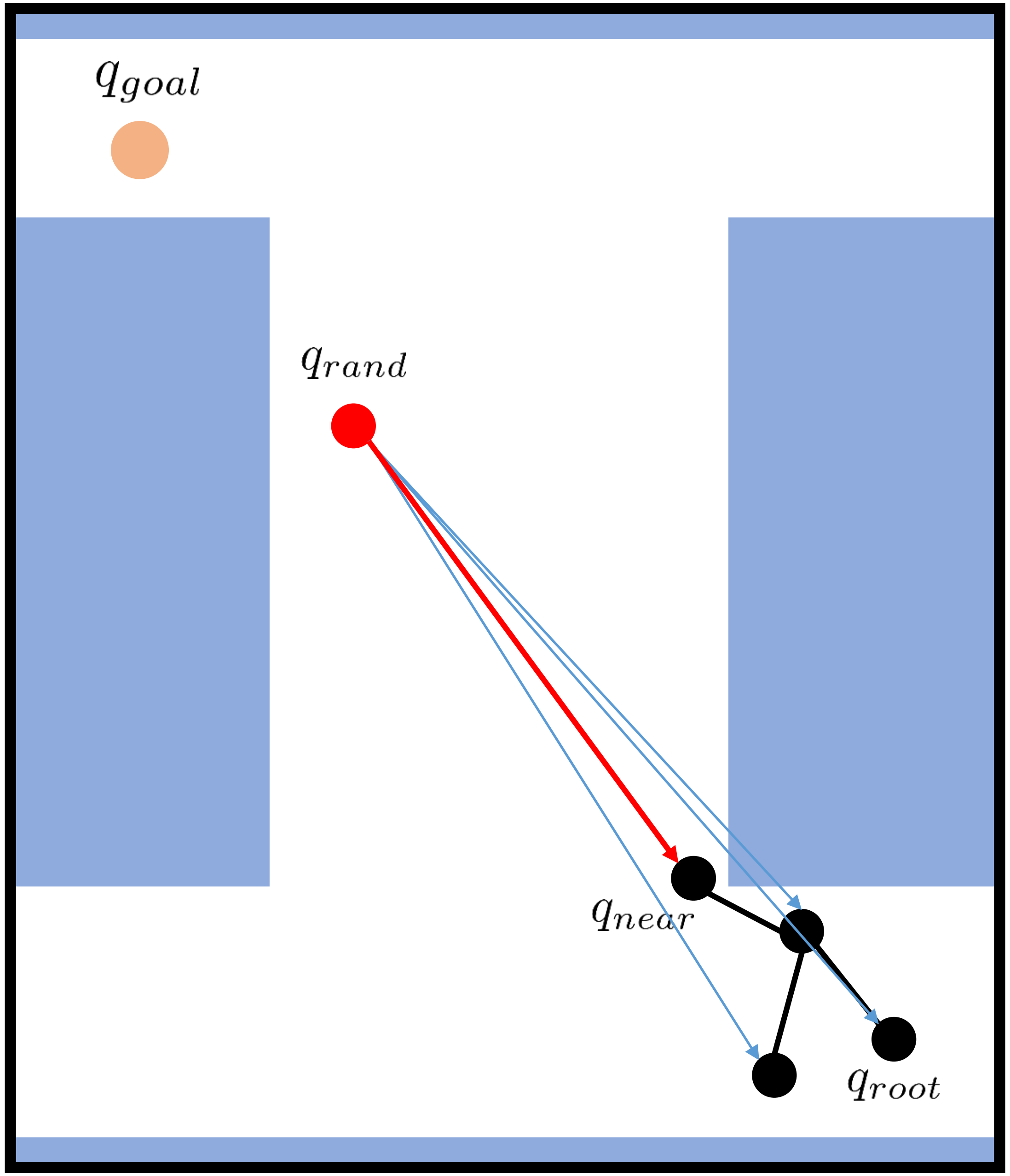
RRT Review

- RRT - Randomized Sampling



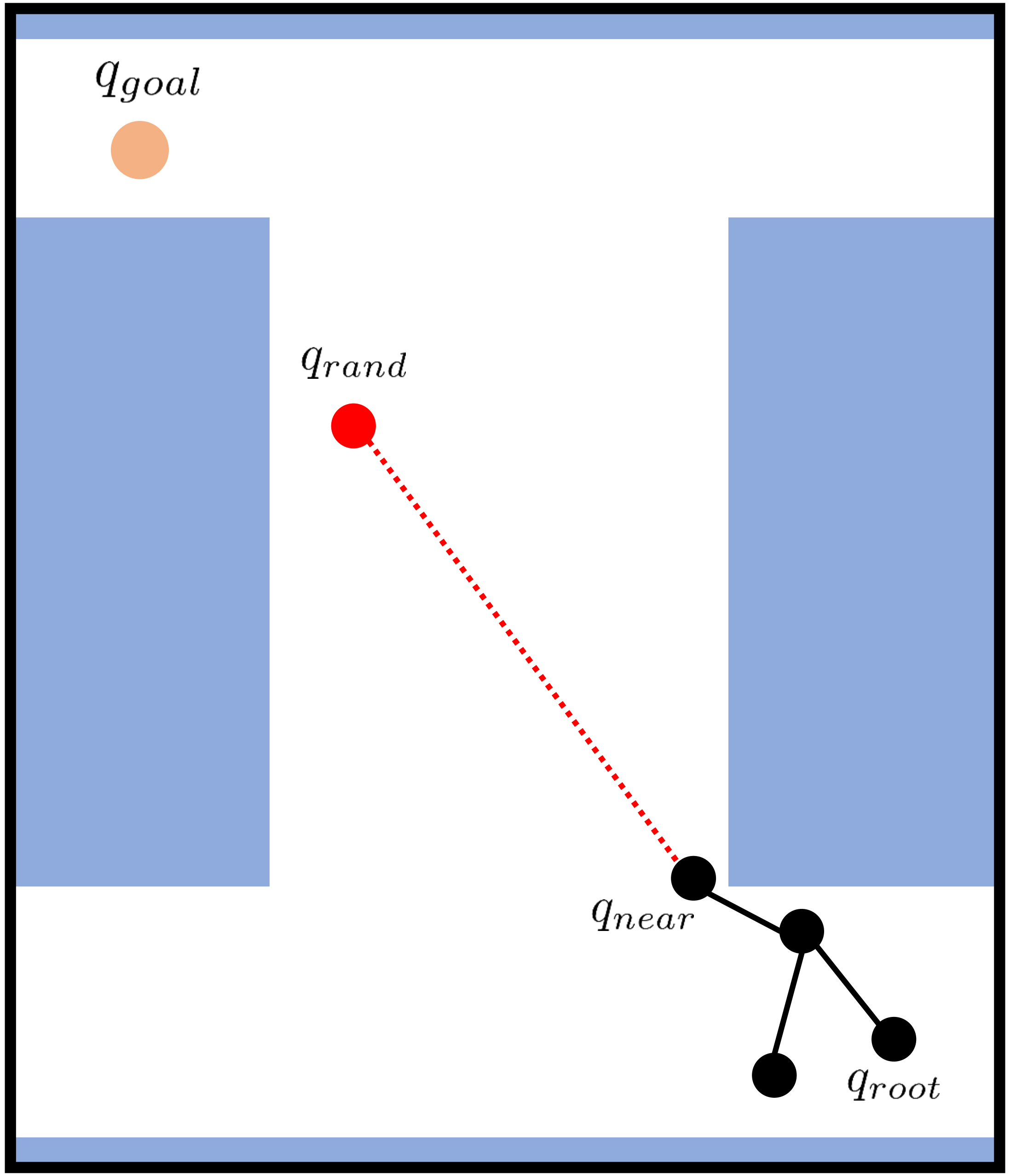
RRT Review

- RRT - Randomized Sampling



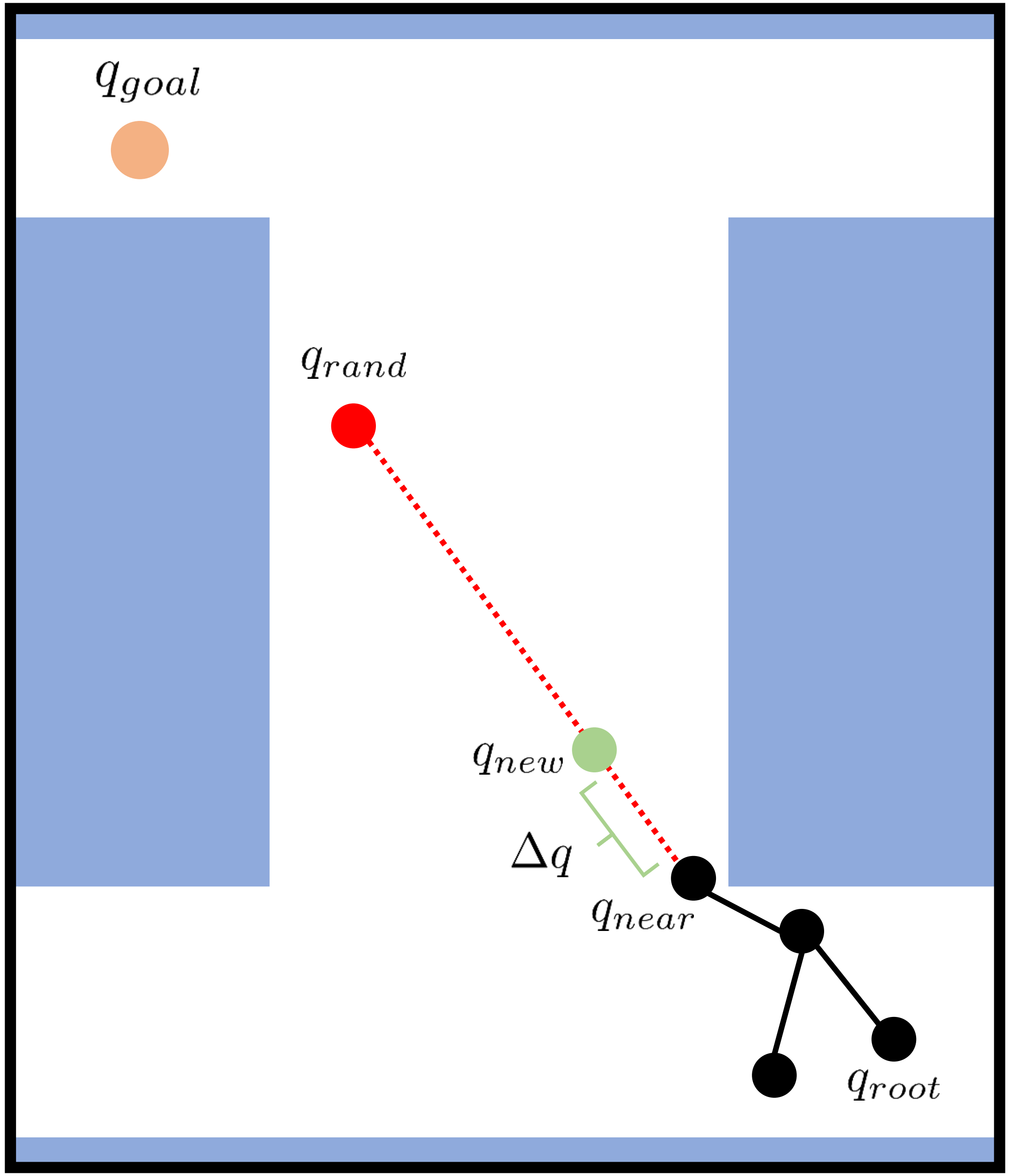
RRT Review

- RRT - Randomized Sampling



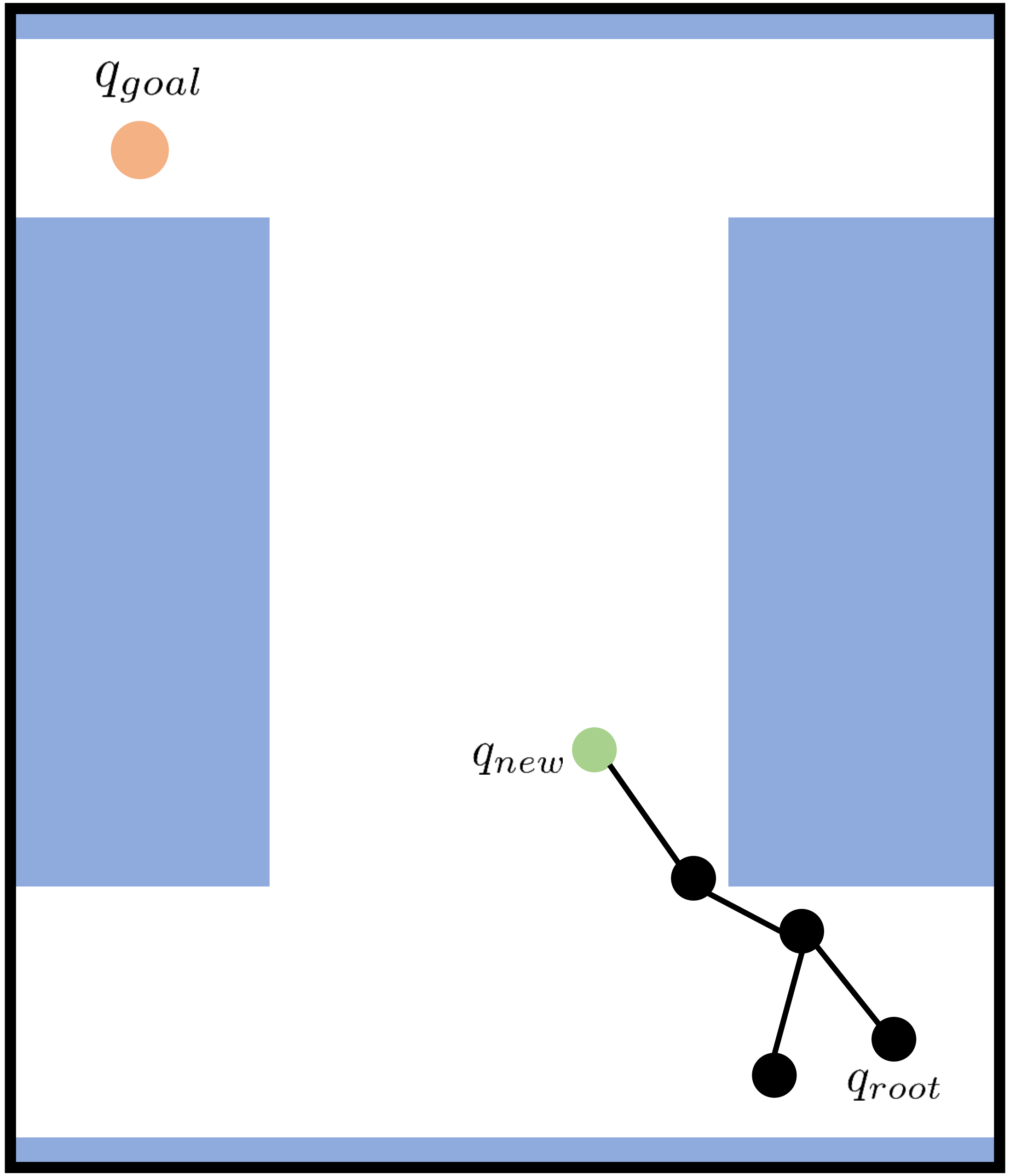
RRT Review

- RRT - Randomized Sampling



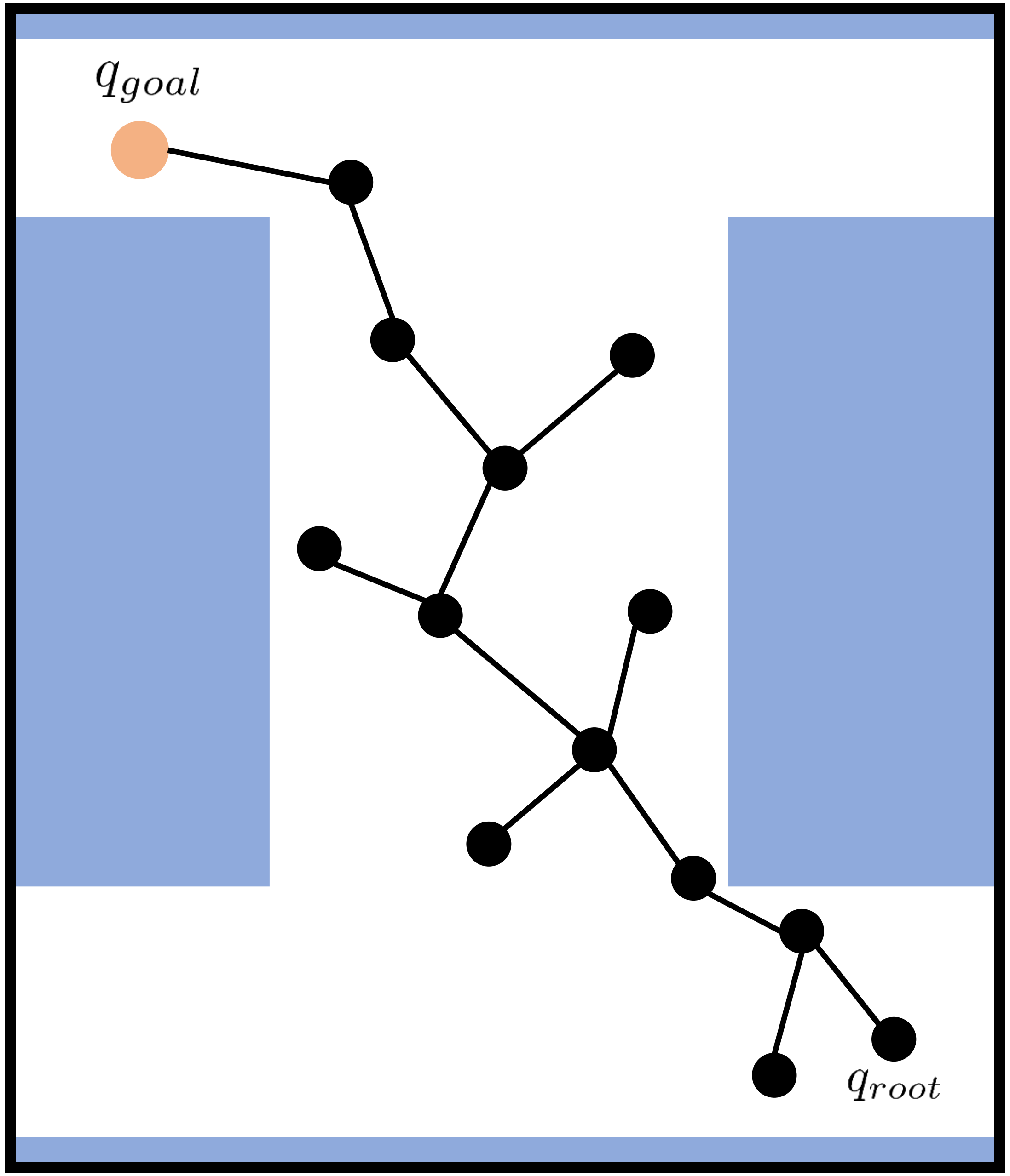
RRT Review

- RRT - Randomized Sampling



RRT Review

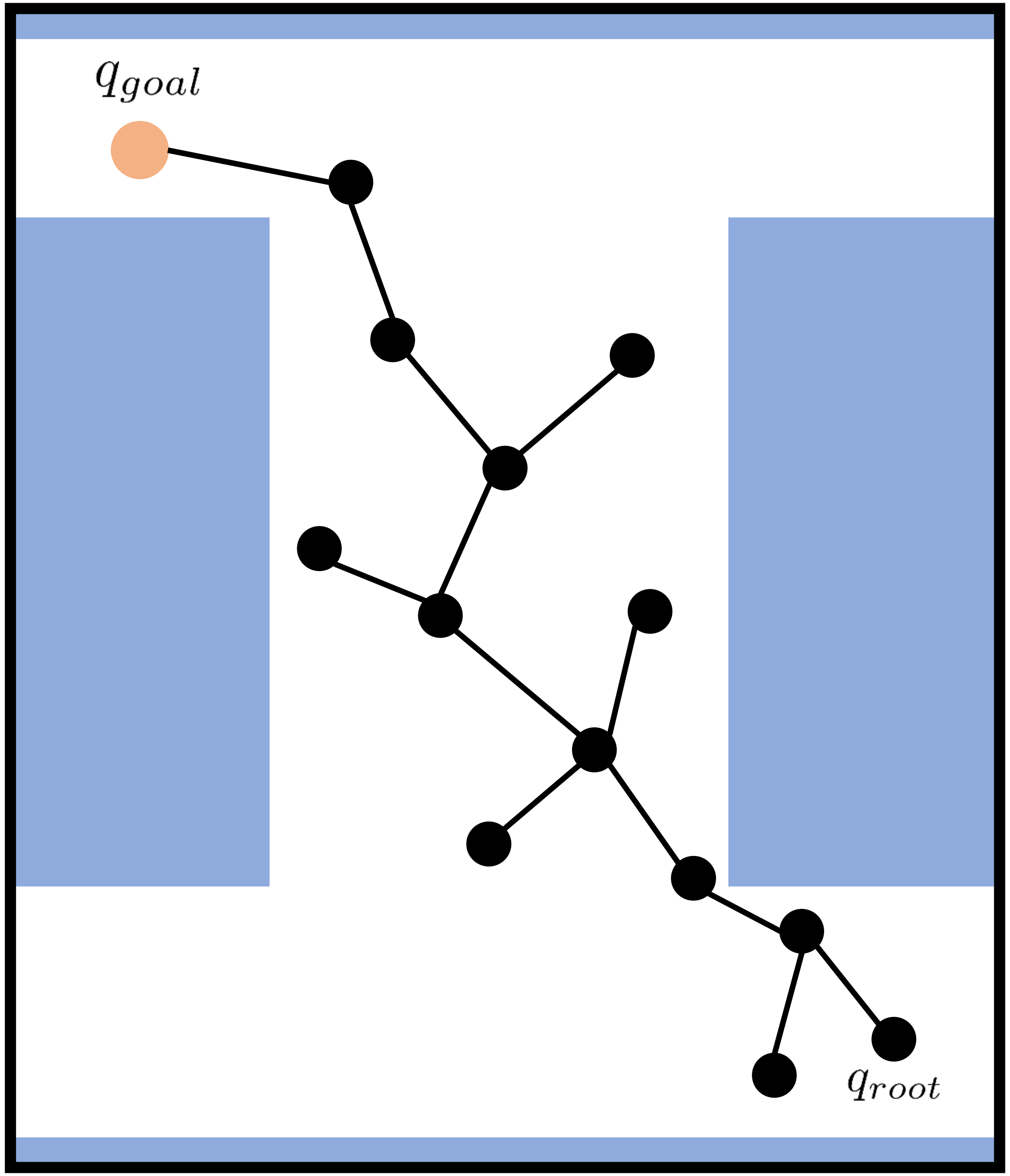
- RRT - Randomized Sampling



RRT Review

- RRT - Randomized Sampling

+ Simple way to construct an approximate model of problem space

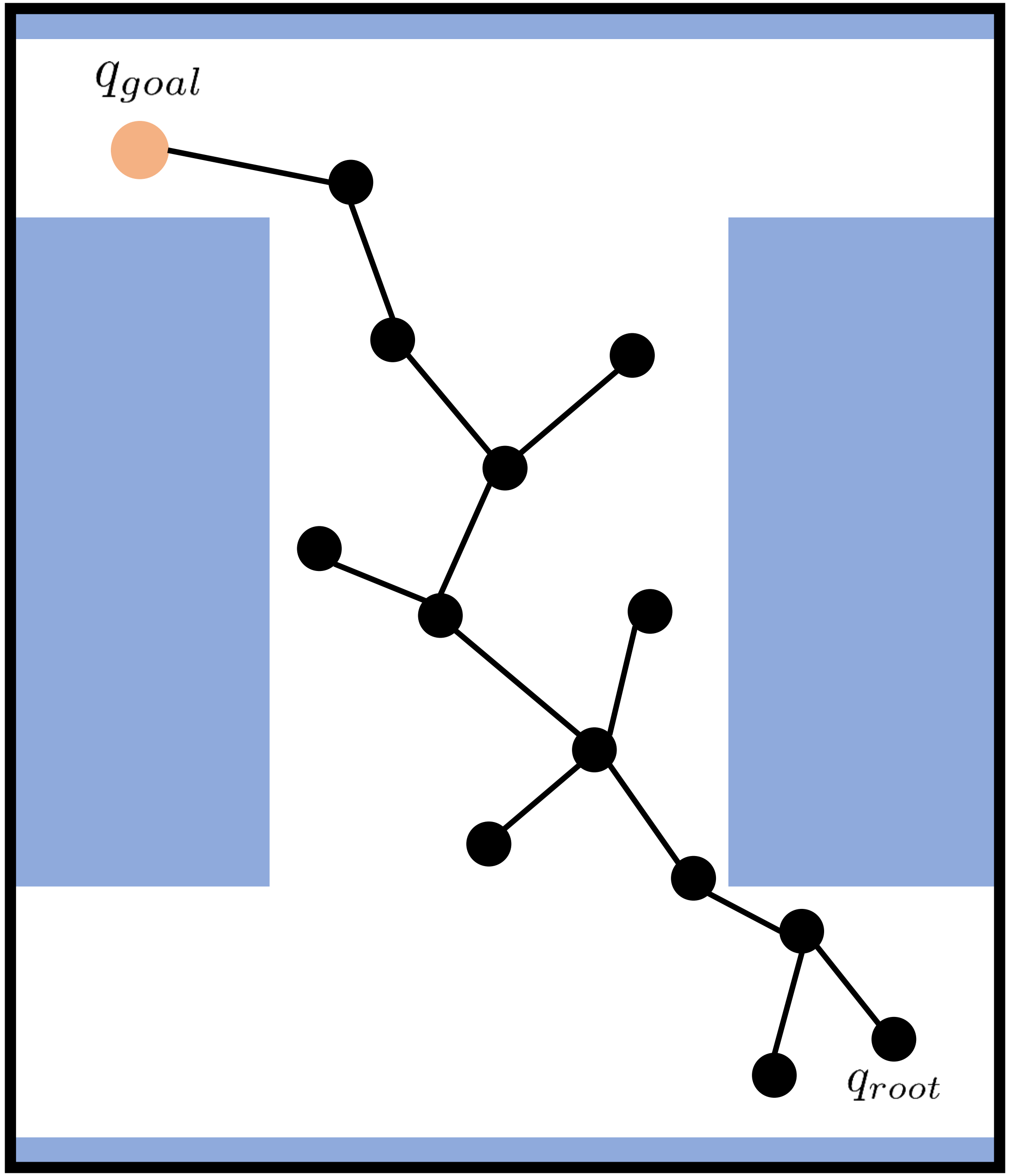


RRT Review

- RRT - Randomized Sampling

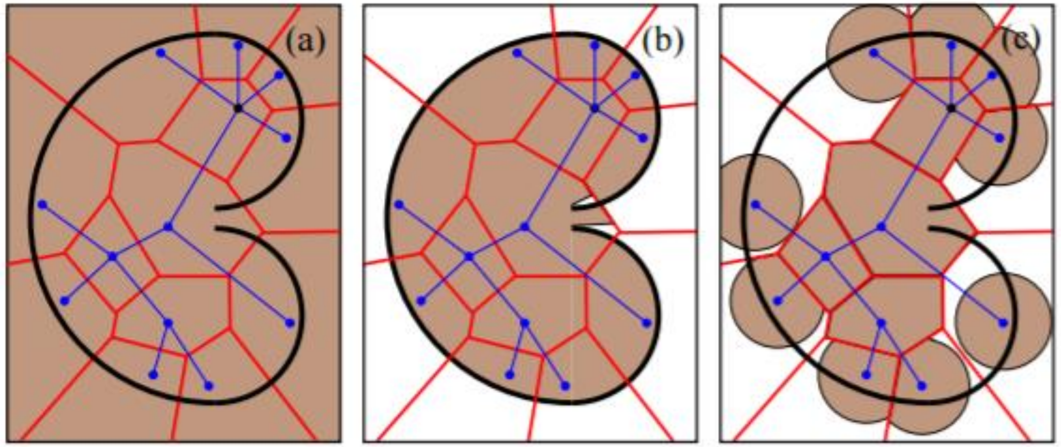
+ Simple way to construct an approximate model of problem space

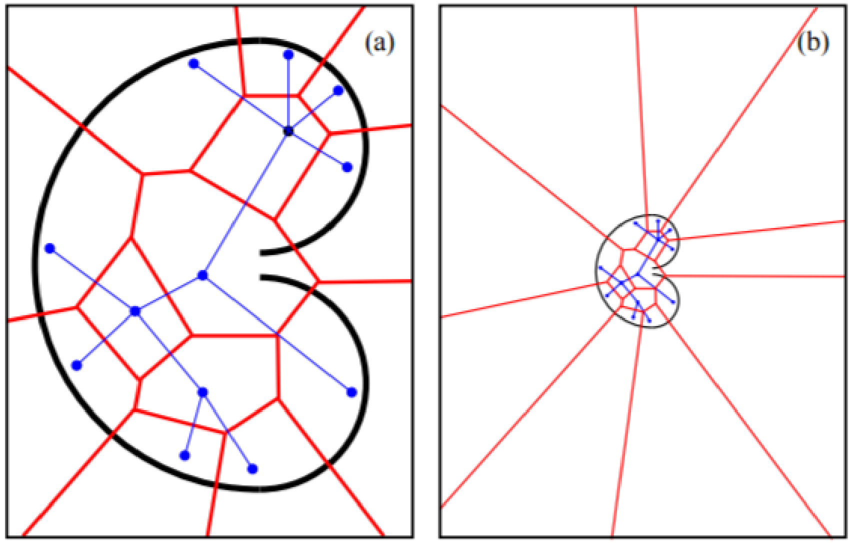
- Weak with narrow and cluttered spaces



Related Work 1

- Dynamic Domain RRT
  - \+ Reduces unnecessary samples from boundary regions
  - \+ High probability of sampling narrow passage
  - \- Worst case same as Regular RRT





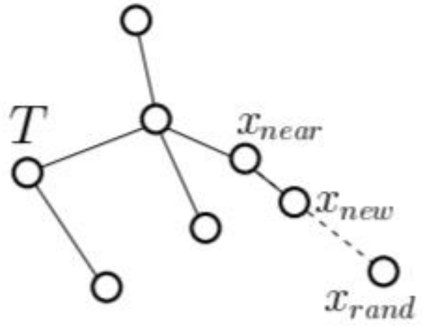
(a) Regular RRT sampling domain
(b) Visible Voronoi region
(c) Dynamic Domain

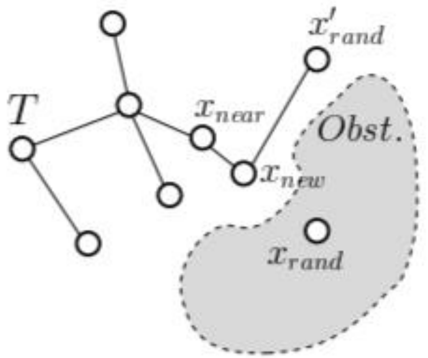
Dynamic-Domain RRTs: Efficient Exploration by Controlling the Sampling Domain by Yershova, Jaillet, Simeon, and La Valle.

Related Work 2

- Obstacle-based RRT (OBRRT) :
  - Growing tree based on obstacle hints

1. Choose a node to grow from – x_{near}
2. Choose a growth method G_i
3. Generate target configuration $x'_{rand'}$
4. Extend from source configuration x_{near} toward target configuration x'_{rand}





G0: Basic Extension
G1: Random position, Same orientation
G2: Random obstacle vector, Random Orientation
G3: Random Obstacle Vector, Same Orientation
G4: Rotation followed by Extension
G5: …
G6: …
…
G9

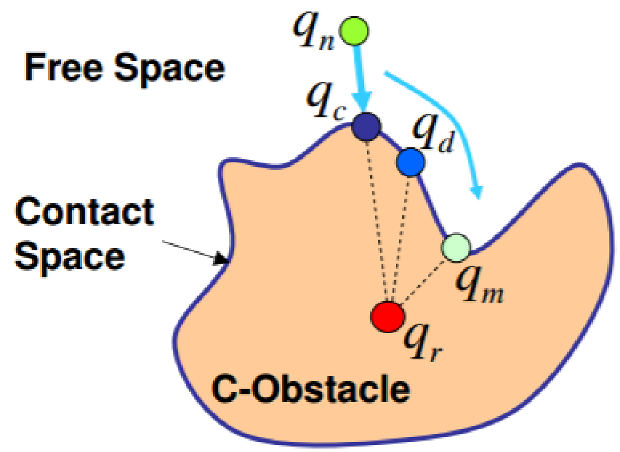
An Obstacle-Based Rapidly-Exploring Random Tree by Samuel Rodriguez, Xinyu Tang, Jyh-Ming Lien and Nancy M. Amato

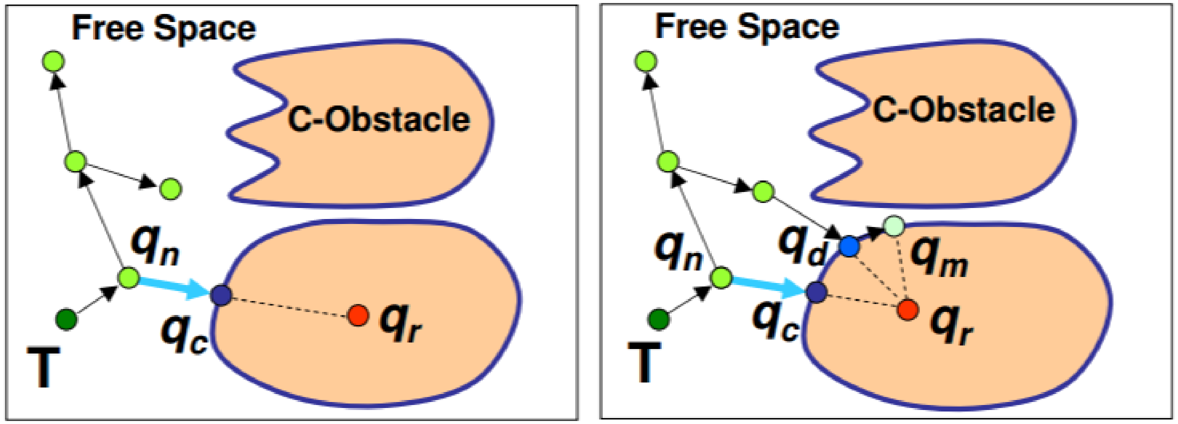
Related Work 3

- Retraction-based RRT

  \+ Improve performance of RRT in narrow passages by sampling near the boundary of C-obstacle

  \- Slower than Regular RRT when there are no narrow passages



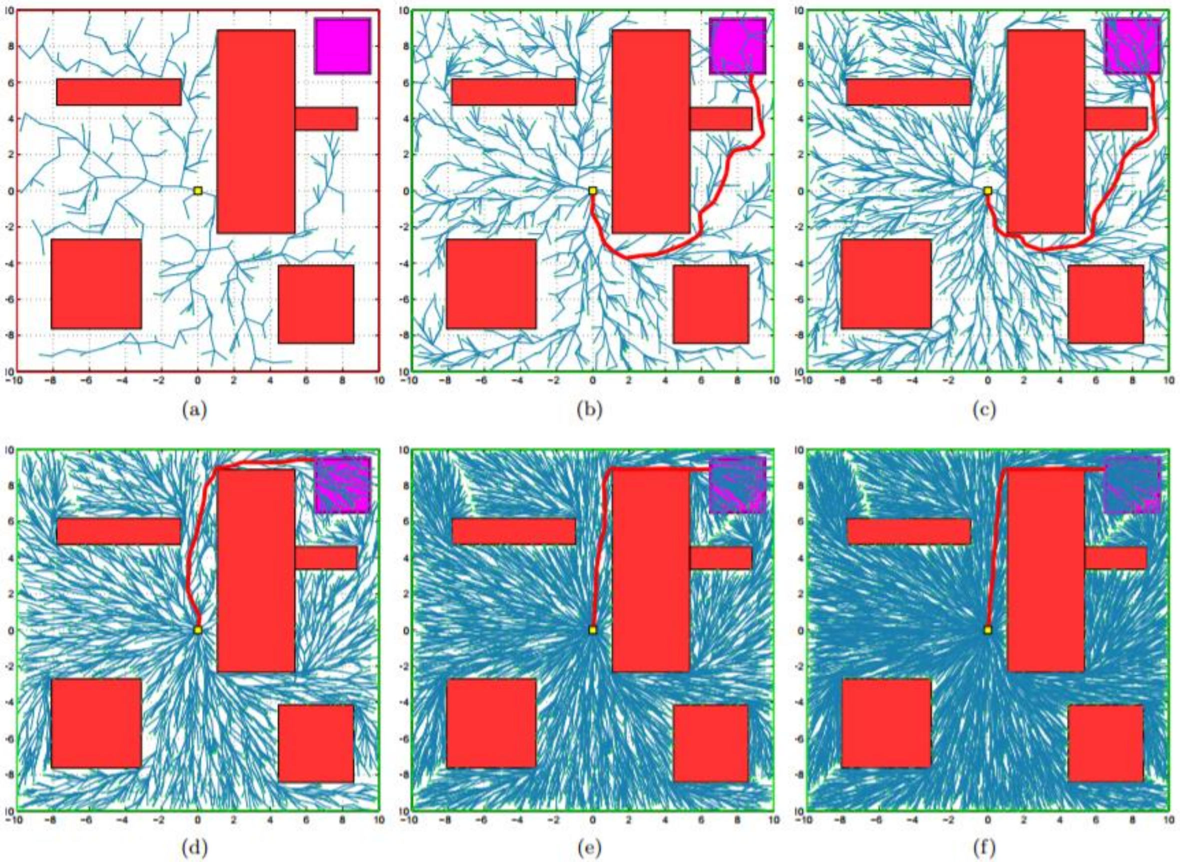


Related Work 4

- RRT*
  - Tree locally rewires itself to ensure optimization of a cost function

  + Effective in finding shortest path

  - In practice, it requires many iterations to produce near optimal solutions



(a) 500, (b) 1500, (c) 2500, (d) 5000,
(e) 10,000, (f) 15,000 iterations

KAIST

More Related Works

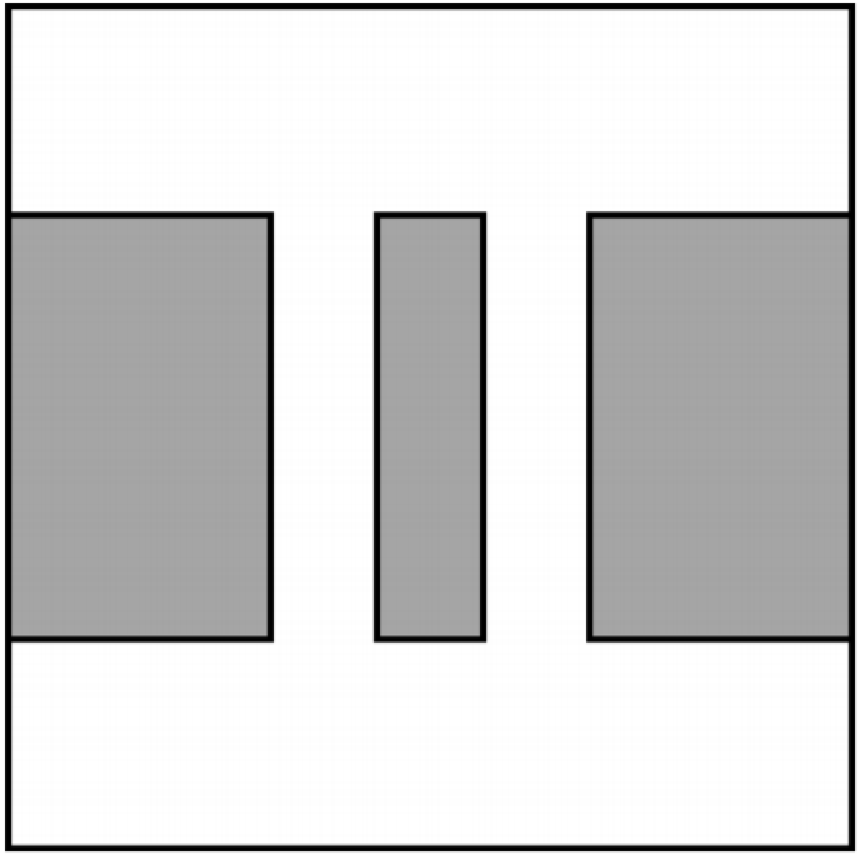
- RRT-Blossom
- Stable Sparse-RRT

…

Dynamic Region RRT

Input: Environment *e* and a query (q_s, q_g)

1. G <- Compute Embedding Graph(e)
   [pre computation]
2. F <- Compute Flow Graph (G, q_s, q_g)
3. R <- Initialize Regions (F, q_s)
4. While not done do
5. Region Biased RRT Growth (F, R)

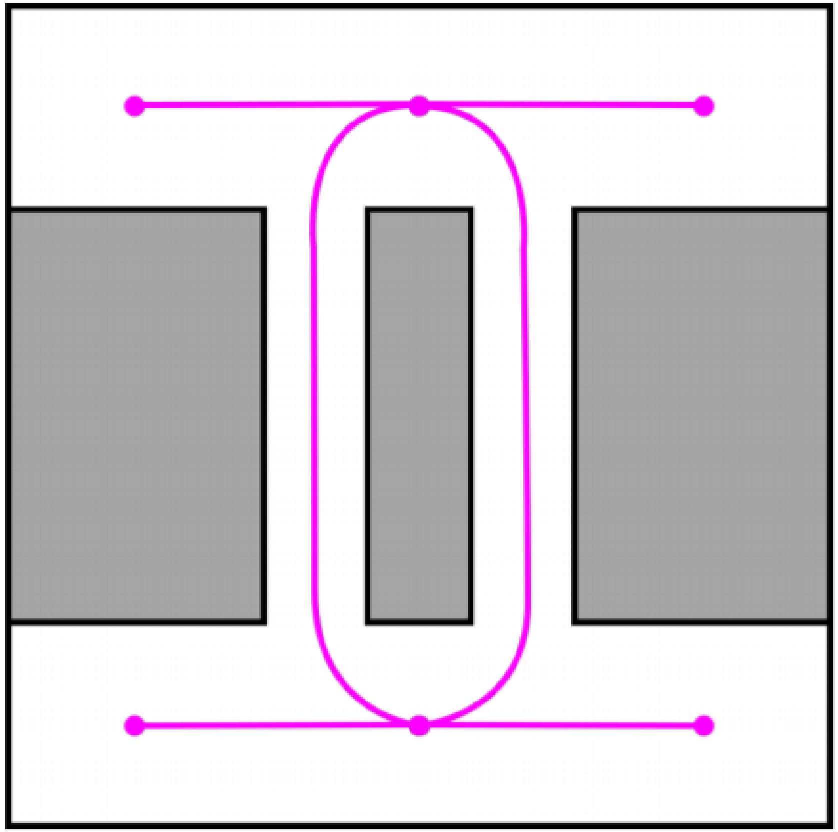


(a) Environment

Dynamic Region RRT

Input: Environment *e* and a query (q_s, q_g)

1. G <- Compute Embedding Graph(e) [pre computation]
2. F <- Compute Flow Graph (G, q_s, q_g)
3. R <- Initialize Regions (F, q_s)
4. While not done do
5. Region Biased RRT Growth (F, R)

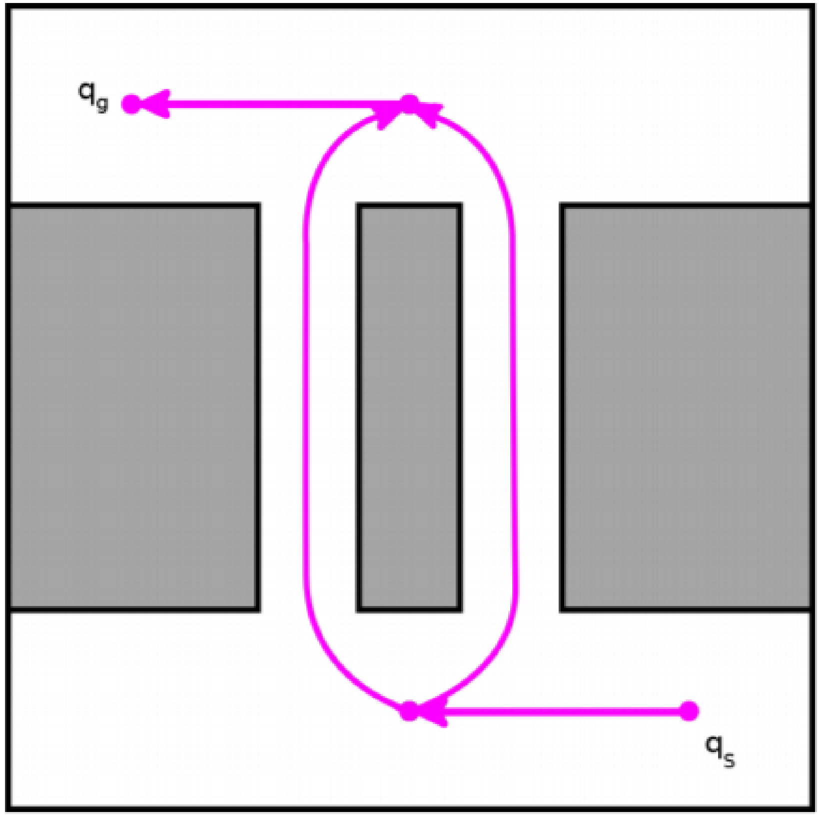


(b) Embedded Graph

Dynamic Region RRT

Input: Environment *e* and a query (q_s, q_g)

1. G <- Compute Embedding Graph(e)
   [pre computation]
2. $\boxed{\text{F <- Compute Flow Graph (G, } q_s\text{, } q_g\text{)}}$
3. R <- Initialize Regions (F, q_s)
4. While not done do
5. Region Biased RRT Growth (F, R)

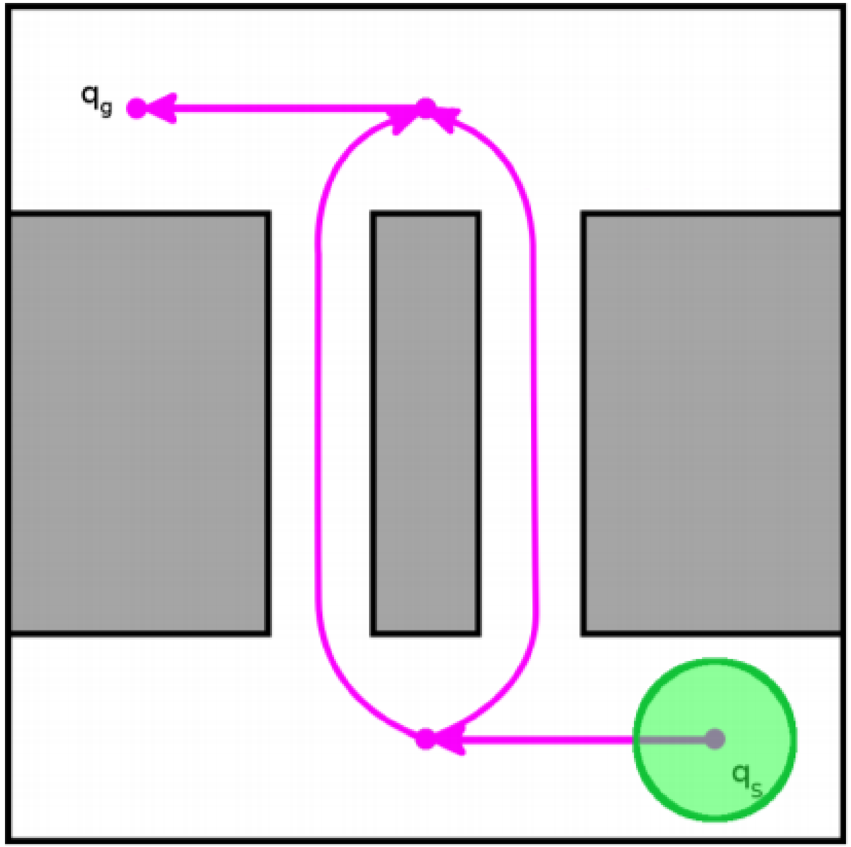


(c) Flow Graph

Dynamic Region RRT

Input: Environment *e* and a query (q_s, q_g)

1. G <- Compute Embedding Graph(e)
   [pre computation]
2. F <- Compute Flow Graph (G, q_s, q_g)
3. R <- Initialize Regions (F, q_s)
4. While not done do
5. Region Biased RRT Growth (F, R)

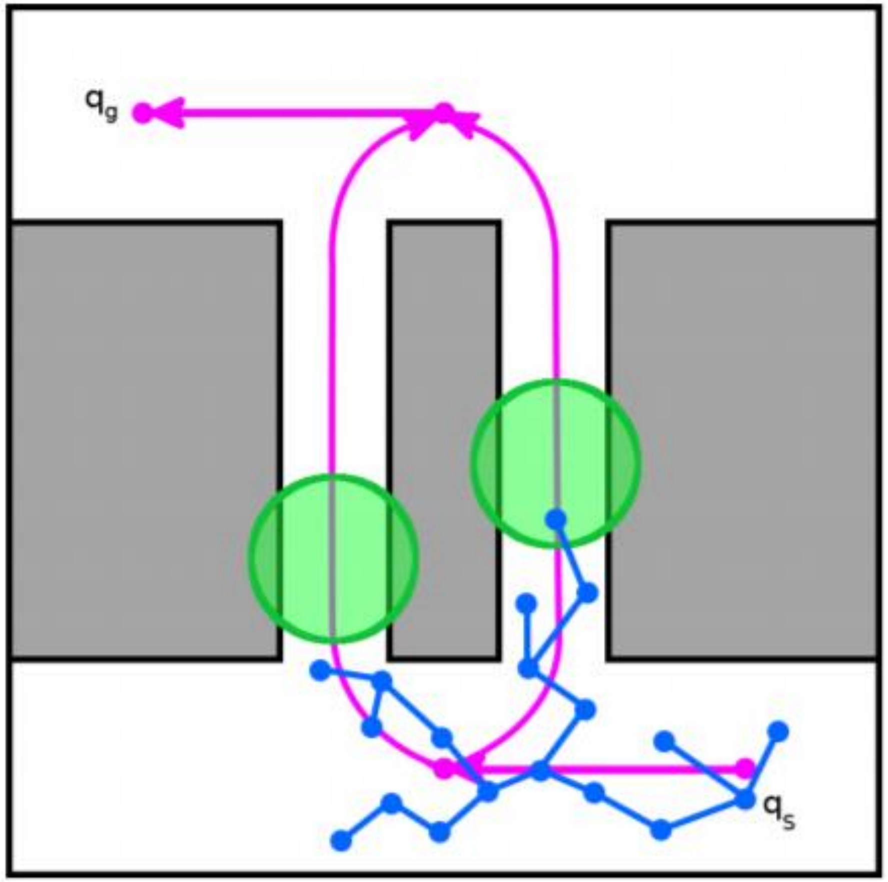


(d) Initial Regions

Dynamic Region RRT

Input: Environment *e* and a query (q_s, q_g)

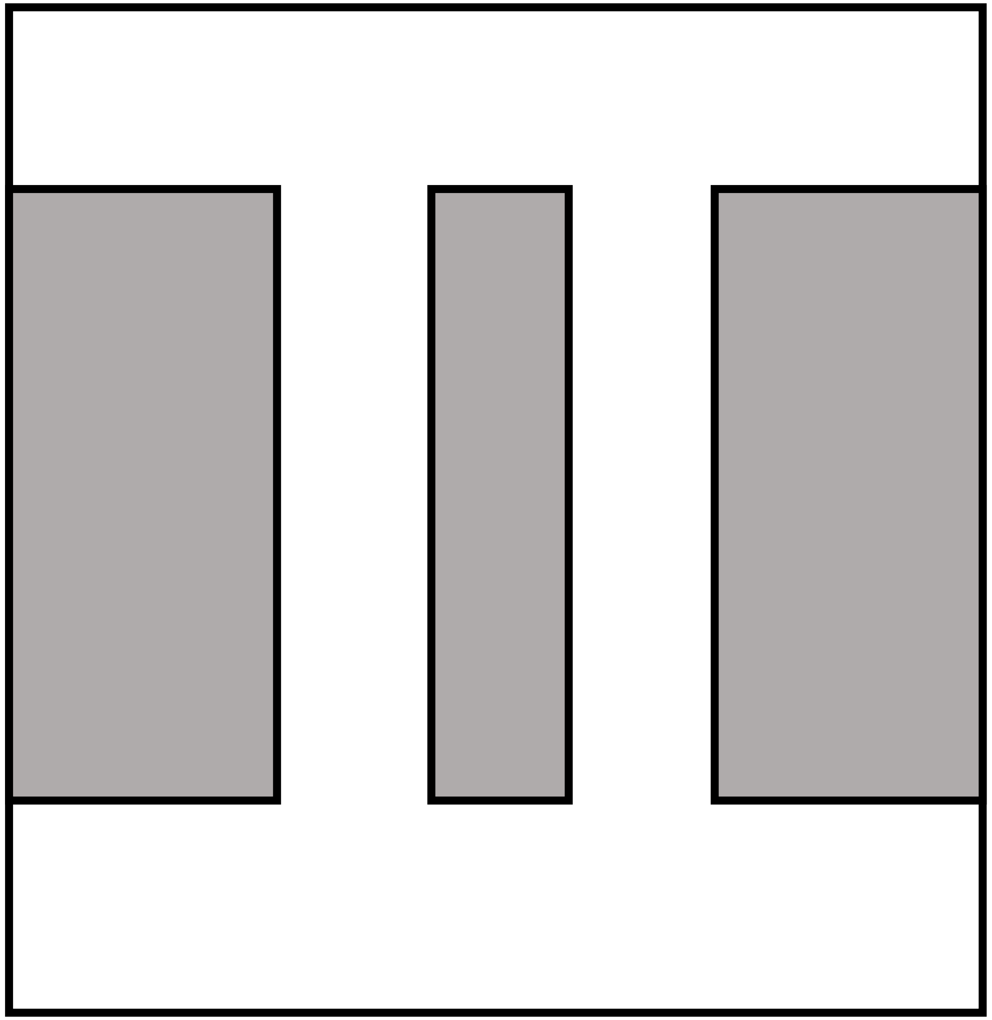
1. G <- Compute Embedding Graph(e) [pre computation]
2. F <- Compute Flow Graph (G, q_s, q_g)
3. R <- Initialize Regions (F, q_s)
4. While not done do
5. Region Biased RRT Growth (F, R)



(f) Multiple Regions

1. Embedding Graph

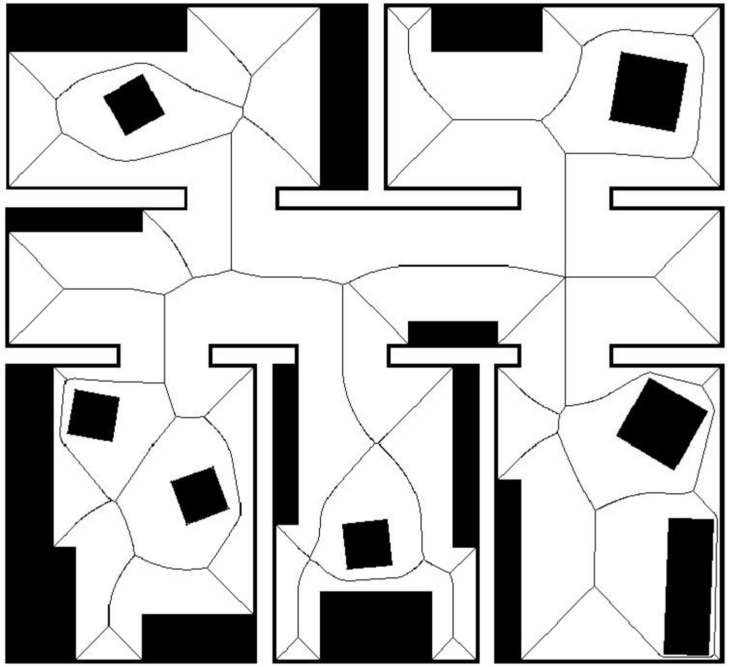
- Computing Embedding Graph

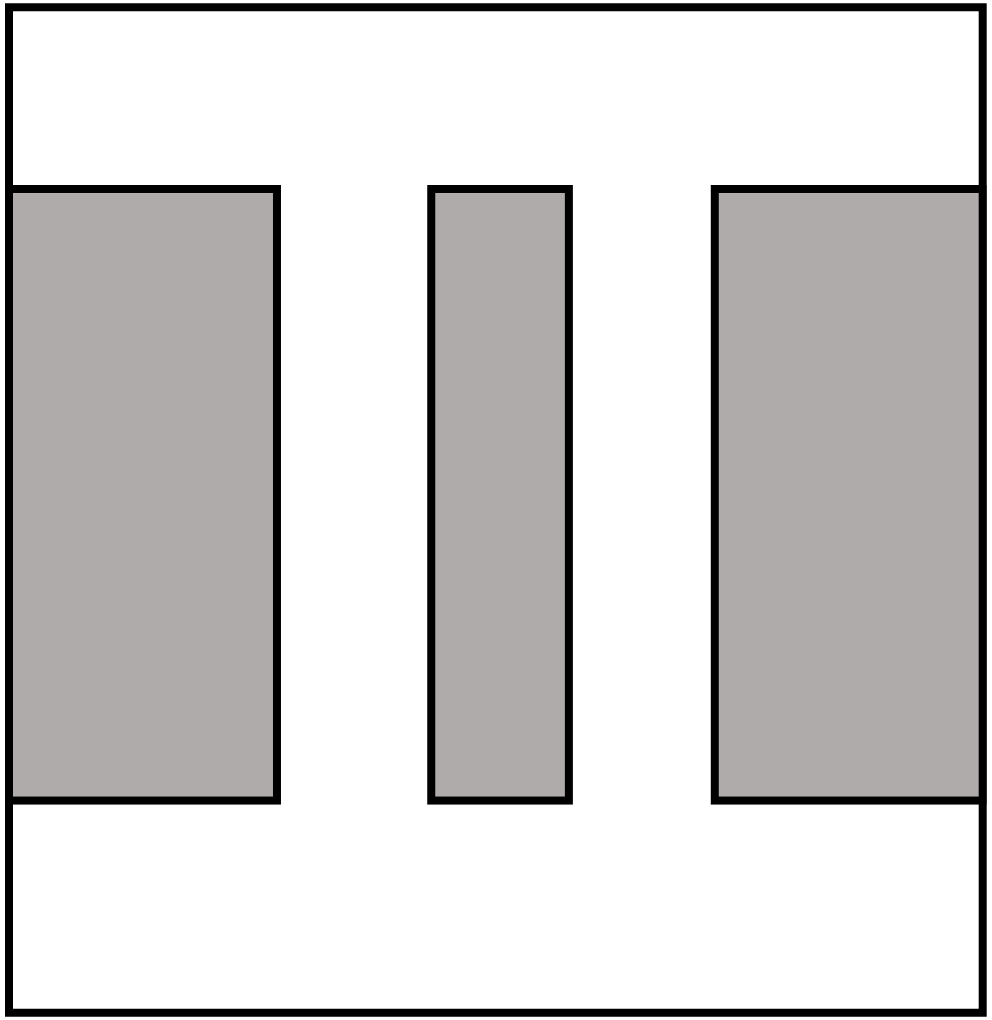


1. Embedding Graph

- Computing Embedding Graph

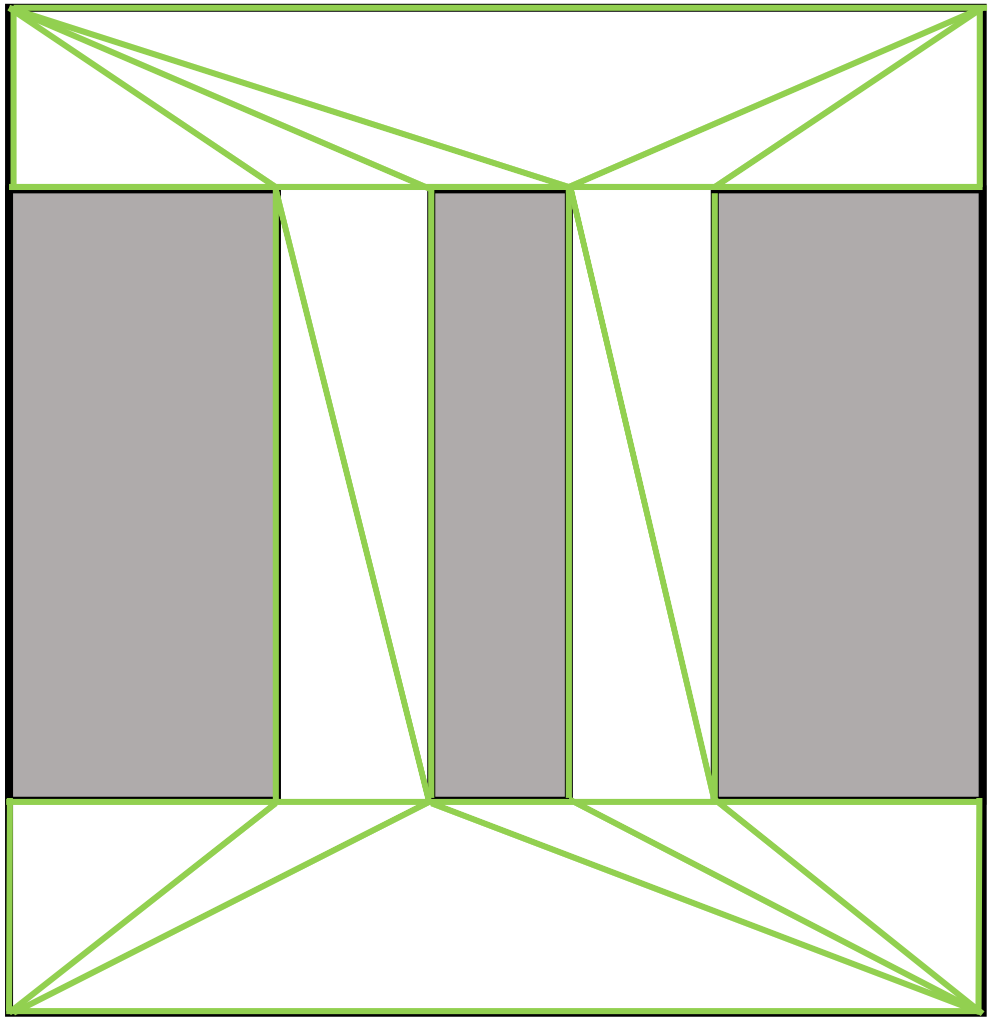
Generalized Voronoi Graph





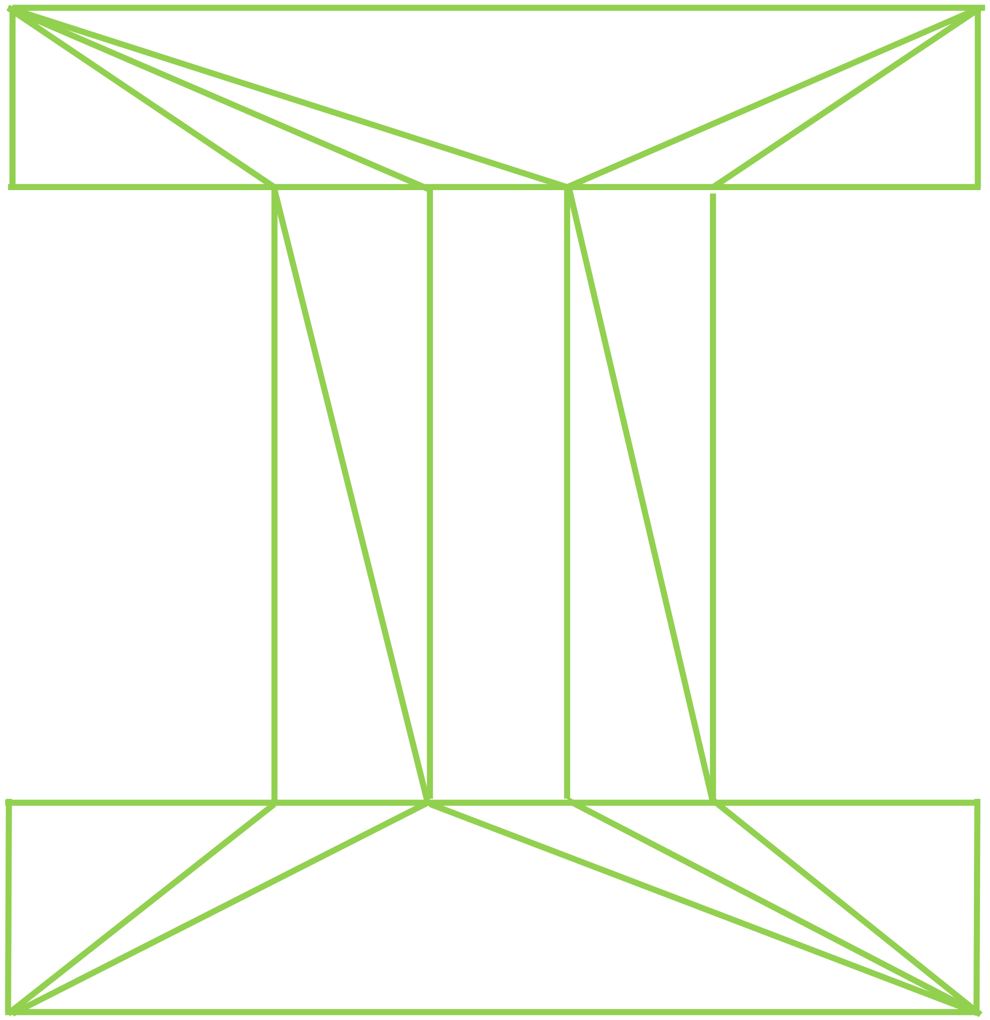
1. Embedding Graph

- Computing Embedding Graph
  1. Compute Tetrahedralization of the environment



1. Embedding Graph

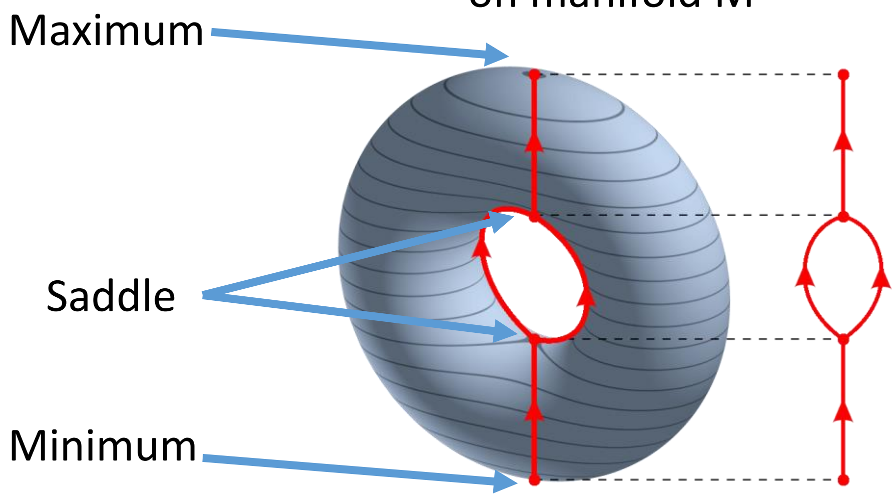
- Computing Embedding Graph
  1. Compute Tetrahedralization of the environment
  2. Construct a Reeb Graph from the Tetrahedralization

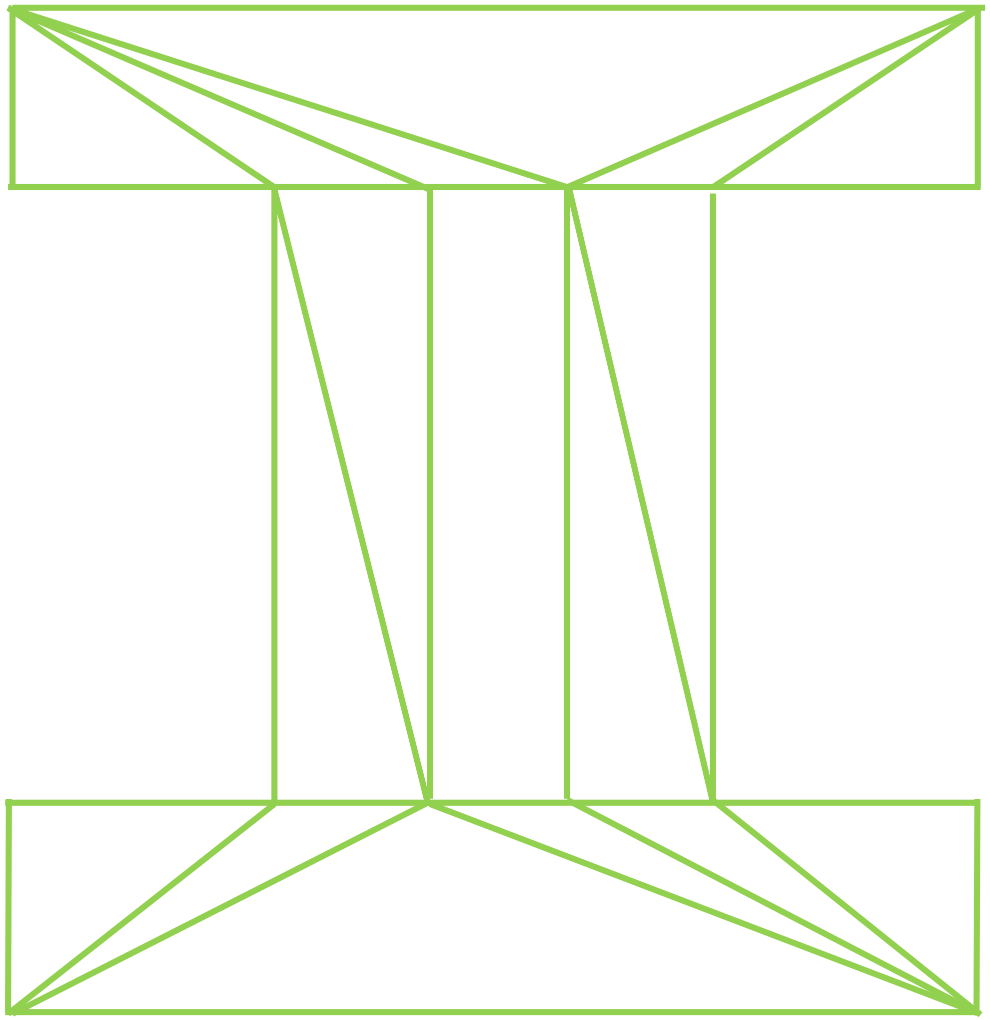


1. Embedding Graph

- Computing Embedding Graph
  1. Compute Tetrahedralization of the environment
  2. Construct a Reeb Graph from the Tetrahedralization

F = z coordinate of a point on manifold M

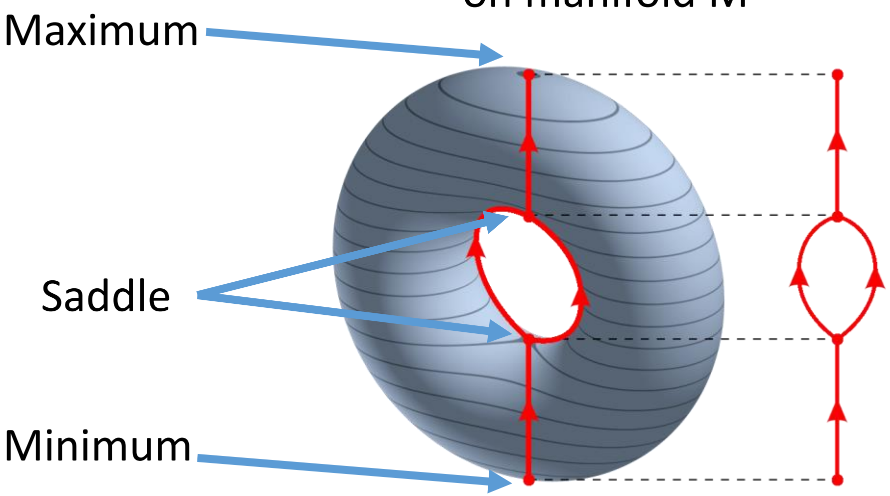




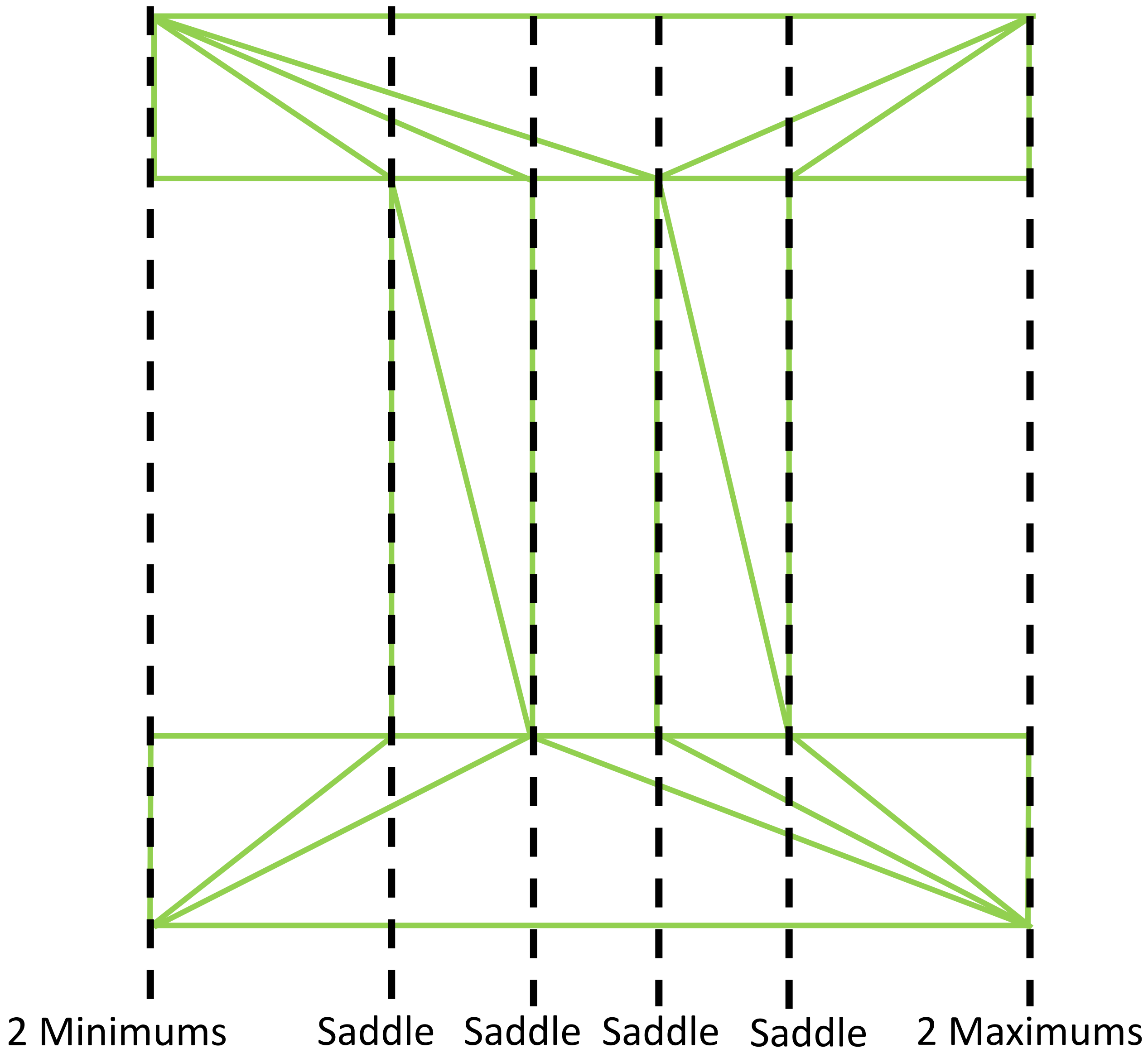
1. Embedding Graph

- Computing Embedding Graph
  1. Compute Tetrahedralization of the environment
  2. Construct a Reeb Graph from the Tetrahedralization

F = z coordinate of a point on manifold M



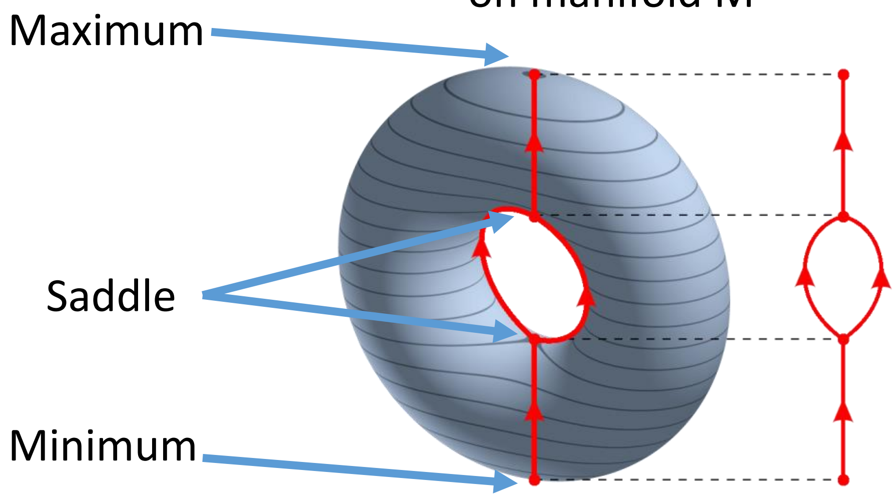
F = y coordinate of a point on manifold M

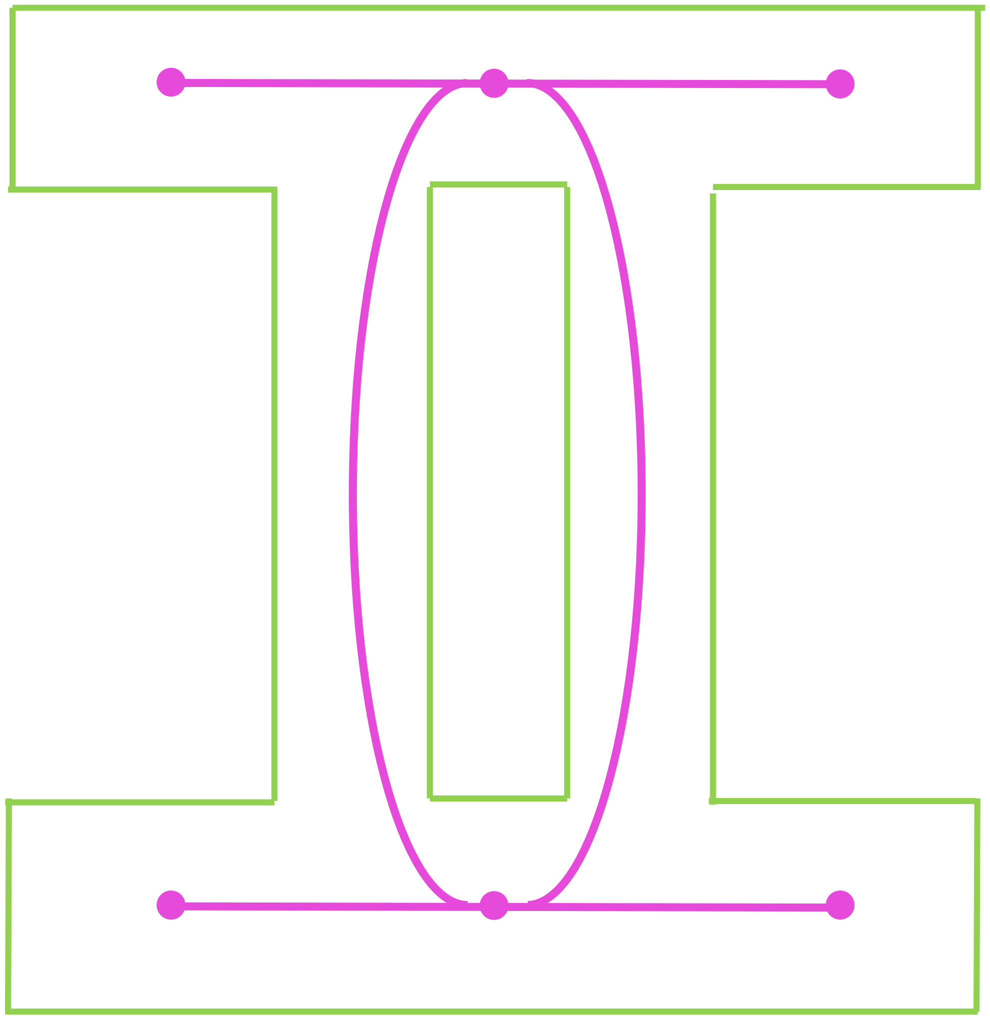


1. Embedding Graph

- Computing Embedding Graph
  1. Compute Tetrahedralization of the environment
  2. Construct a Reeb Graph from the Tetrahedralization

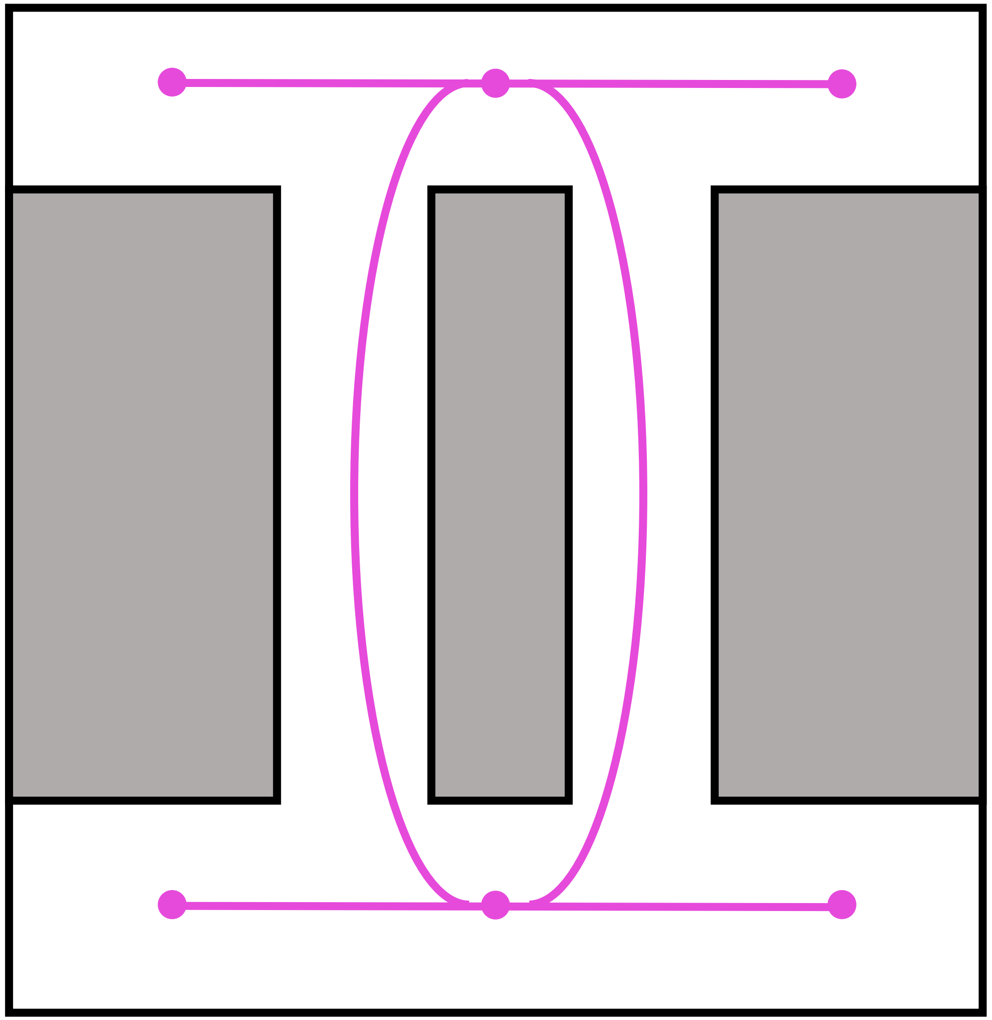
F = z coordinate of a point on manifold M





1. Embedding Graph

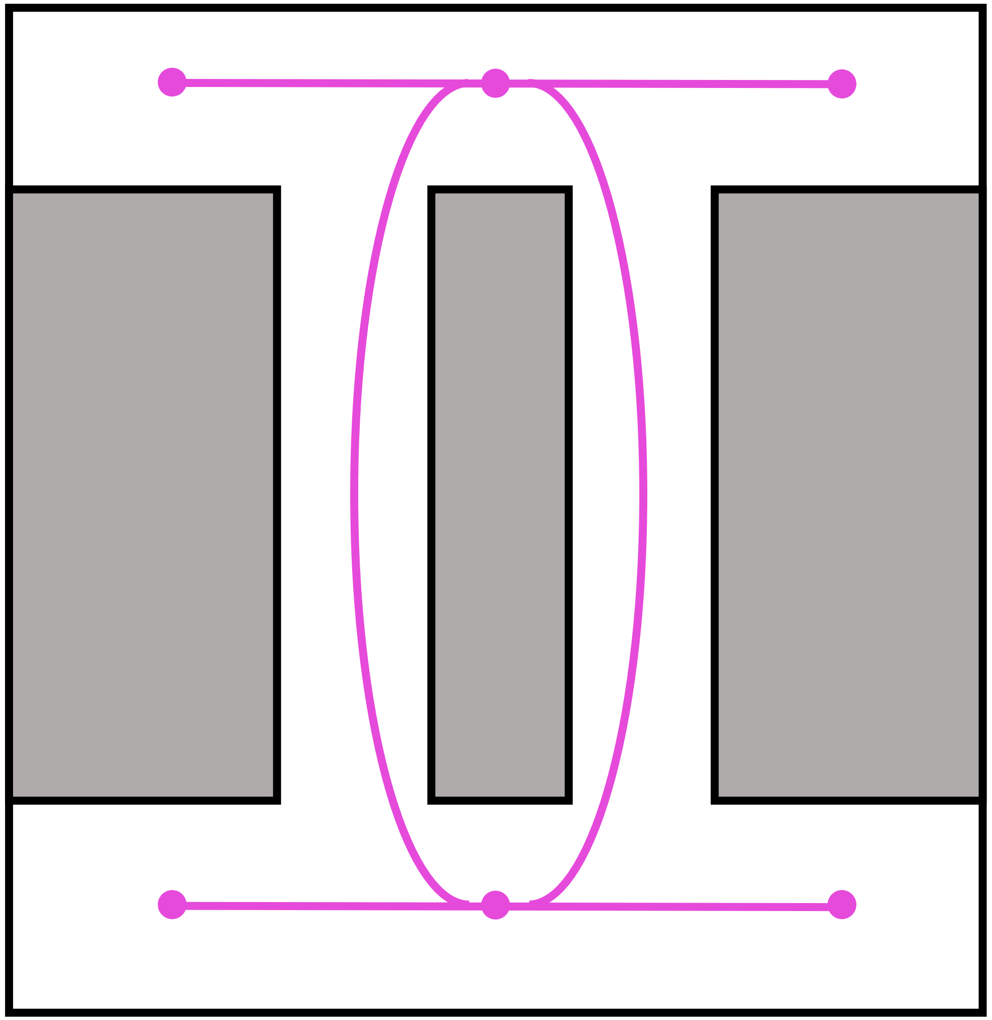
- Computing Embedding Graph
  1. Compute Tetrahedralization of the environment
  2. Construct a Reeb Graph from the Tetrahedralization
  3. Embed the Reeb graph back to the Environment



1. Embedding Graph

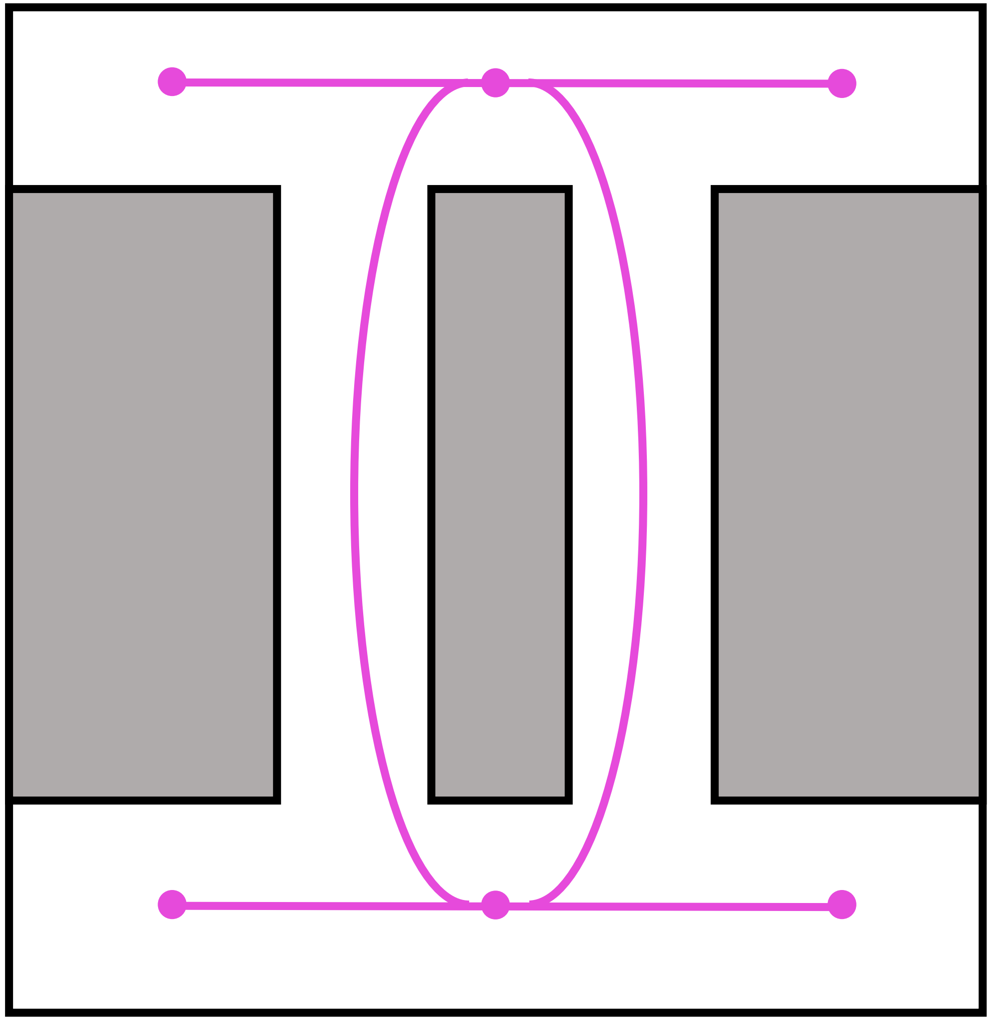
- Computing Embedding Graph
    1. Compute Tetrahedralization of the environment
    2. Construct a Reeb Graph from the Tetrahedralization
    3. Embed the Reeb graph back to the Environment

Naïve Reeb Graph Algorithm: $O(n^2)$



1. Embedding Graph

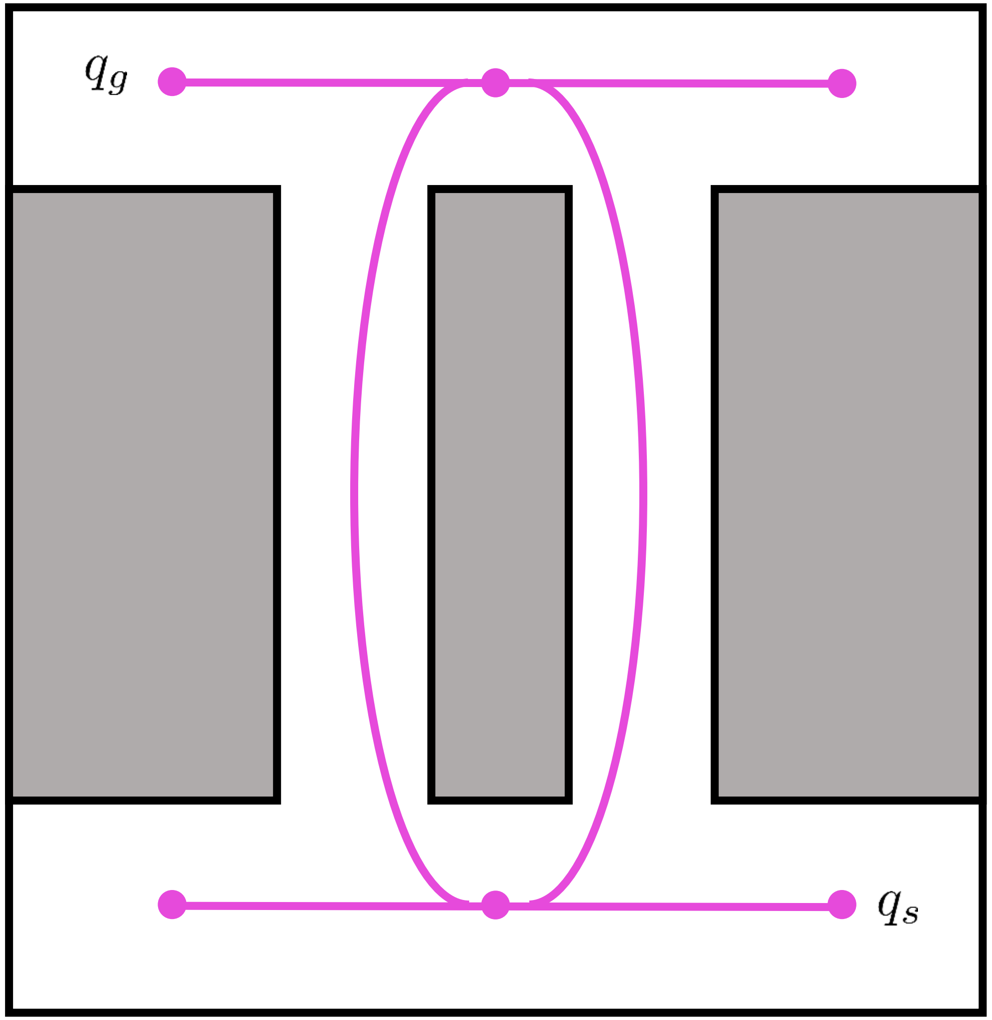
- Computing Embedding Graph
  1. Compute Tetrahedralization of the environment
  2. Construct a Reeb Graph from the Tetrahedralization
  3. $\boxed{\text{Embed the Reeb graph back to the Environment}}$



Naïve Reeb Graph Algorithm: $O(n^2)$
Fast Reeb Graph Algorithm: $O(n \log(n))$

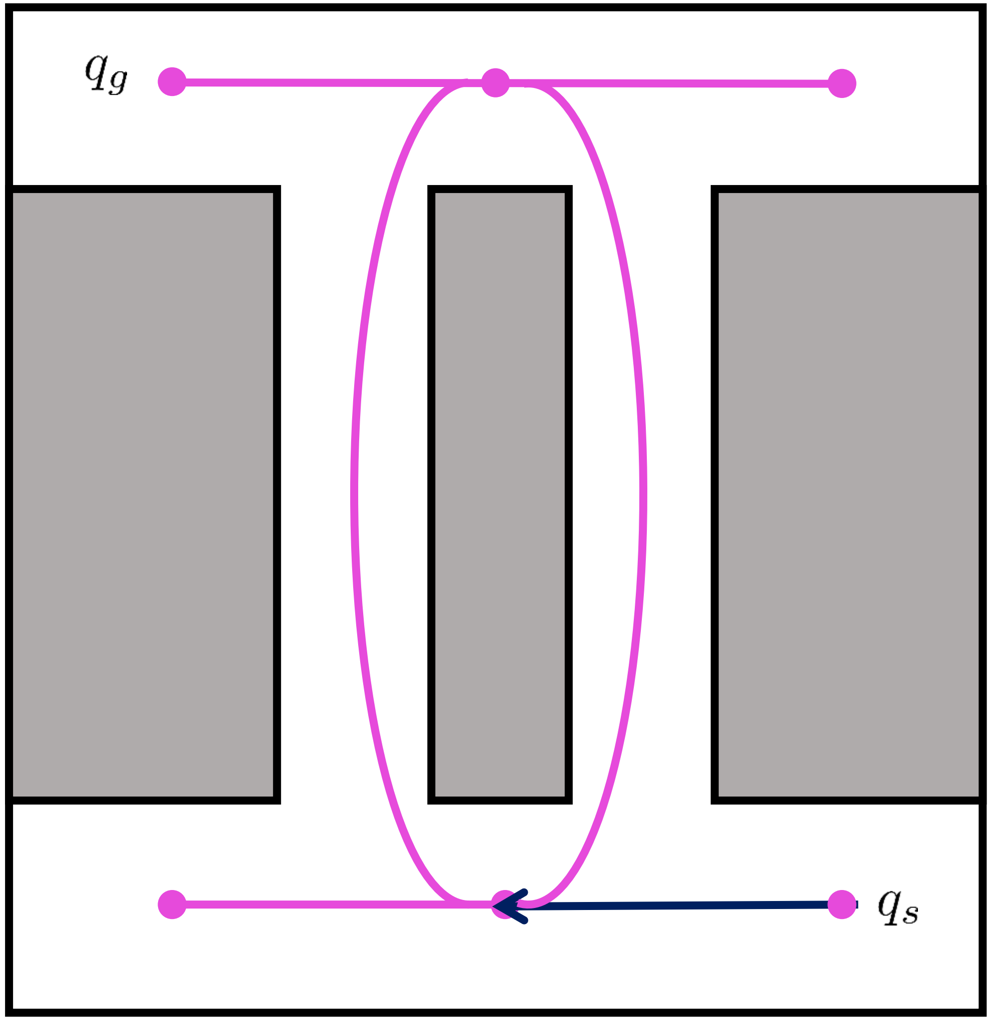
2. Flow Graph

- Computing Flow Graph



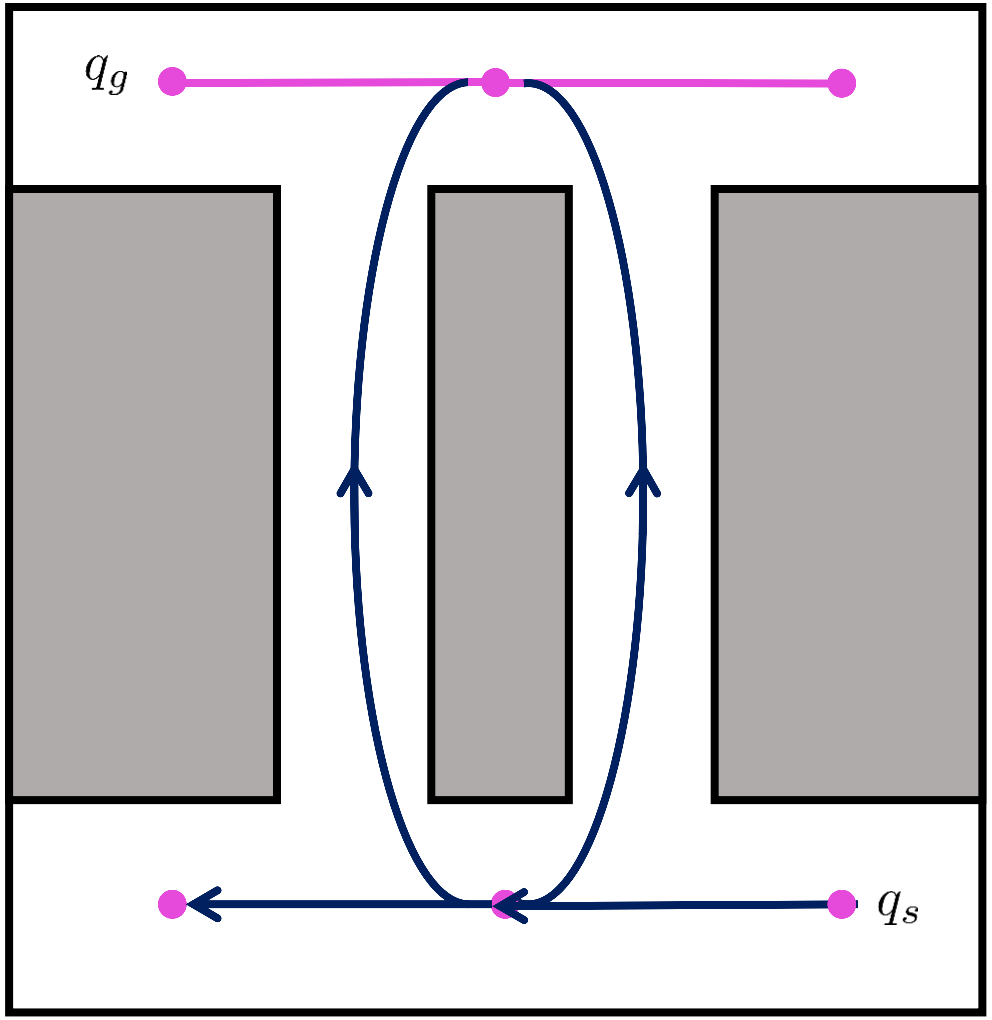
2. Flow Graph

- Computing Flow Graph
  1. Perform BFS from the nearest node q_s



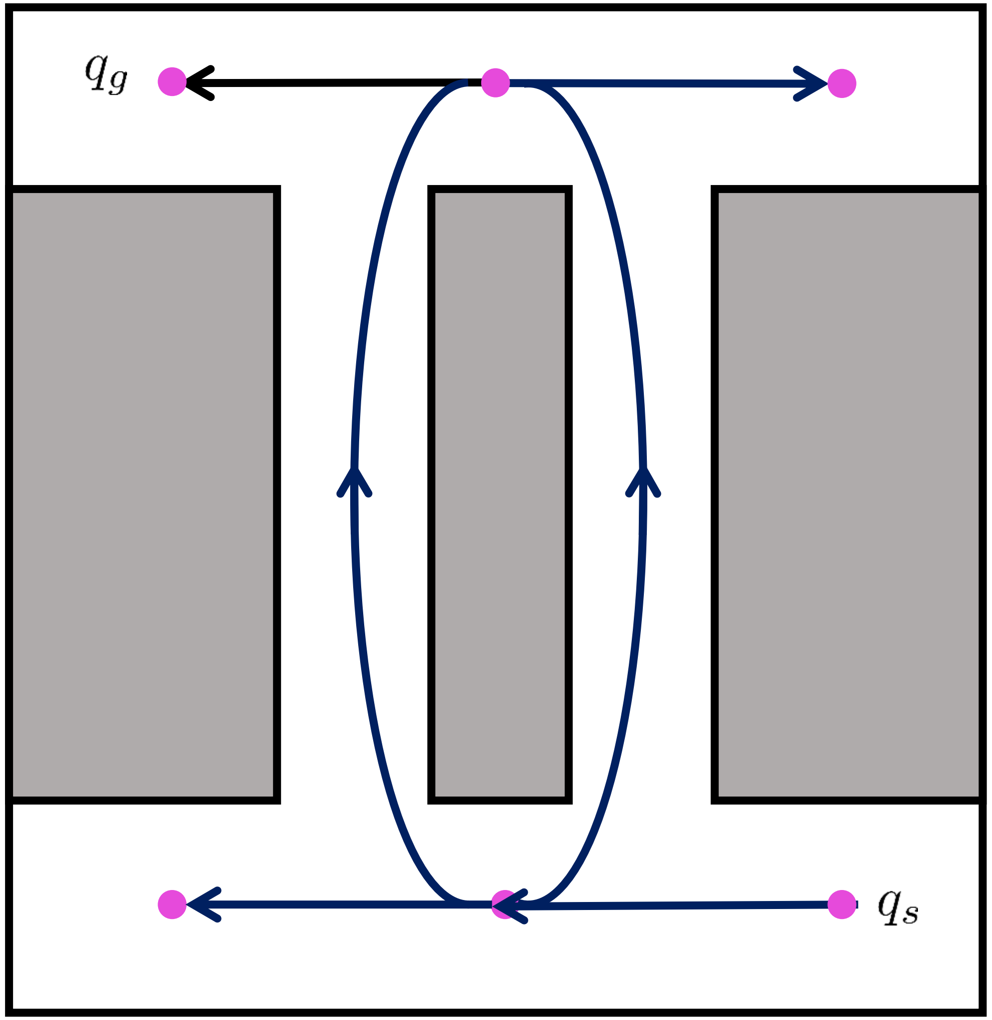
2. Flow Graph

- Computing Flow Graph
  1. Perform BFS from the nearest node q_s



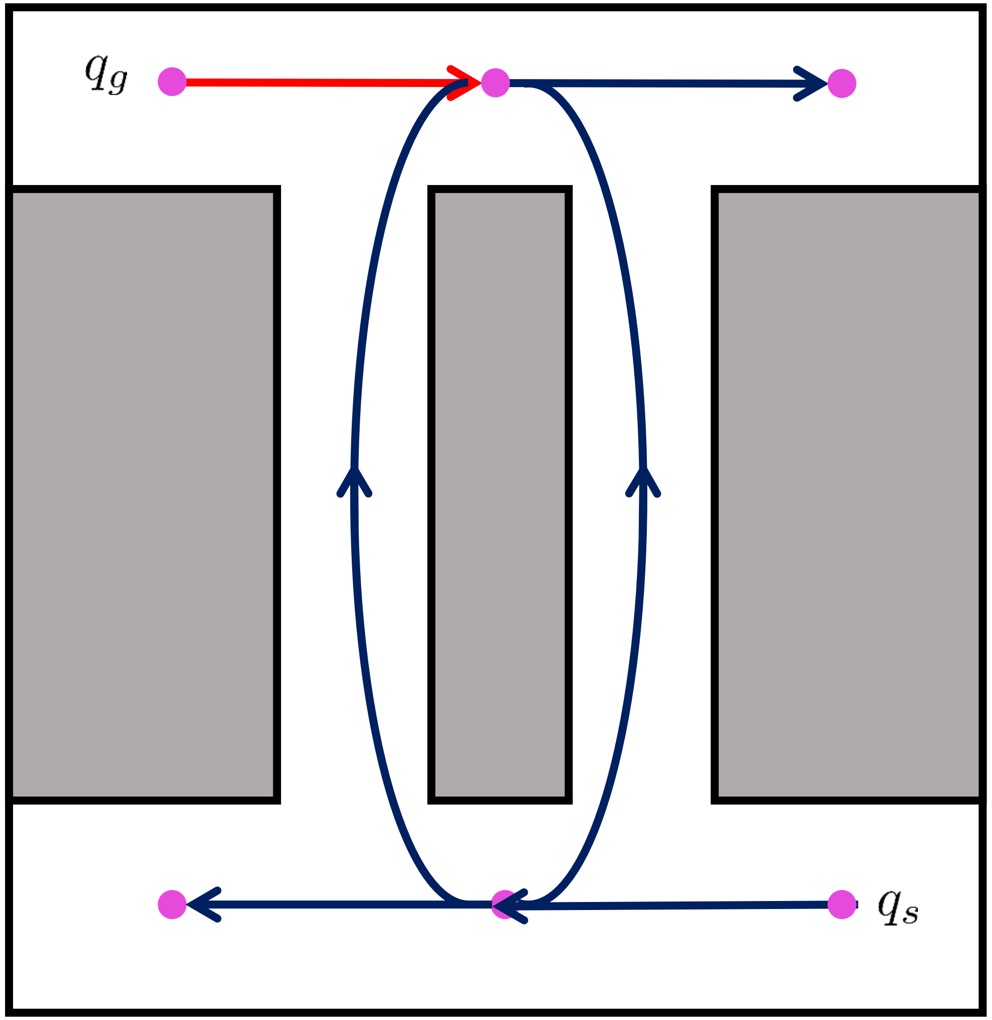
2. Flow Graph

- Computing Flow Graph
  1. Perform BFS from the nearest node q_s



2. Flow Graph

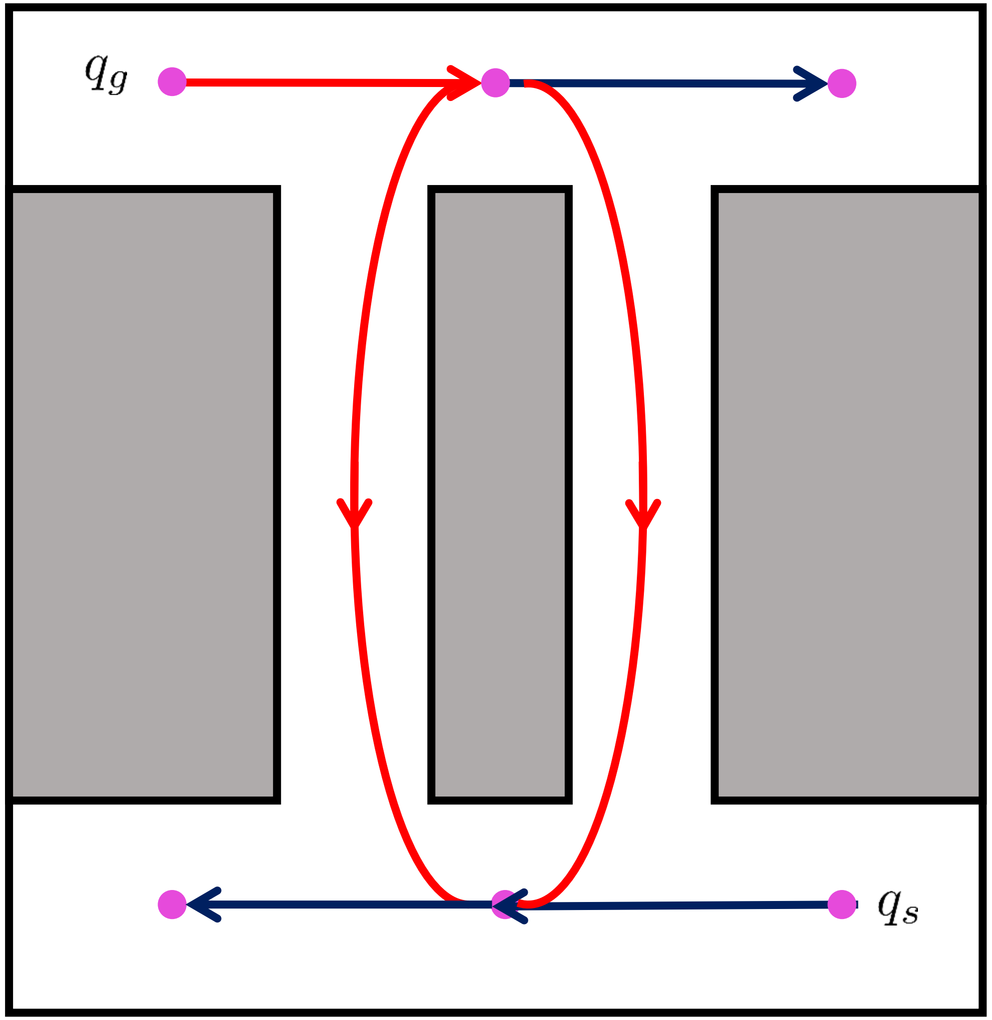
- Computing Flow Graph
  1. Perform BFS from the nearest node q_s
  2. Backtrack from the nearest node to q_g to trim unrelated edges to a solution path (pruning)



2. Flow Graph

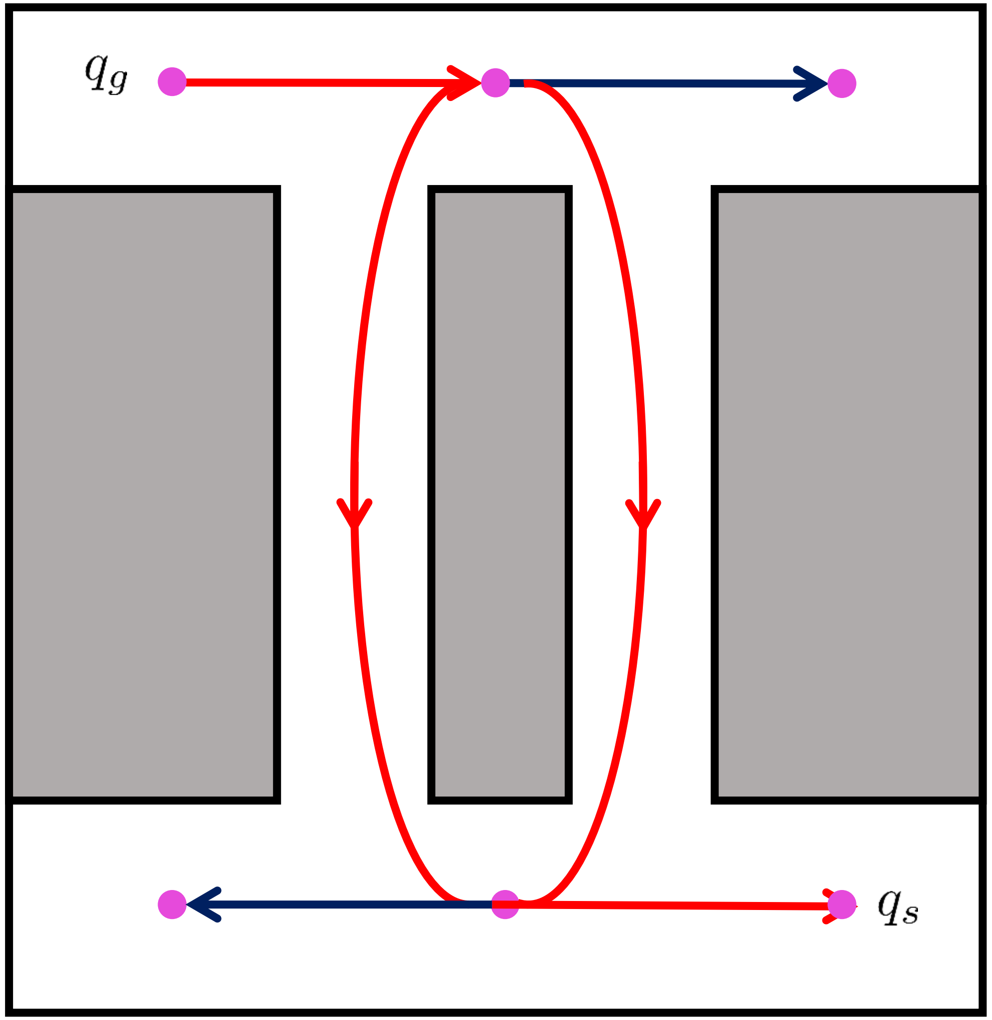
- Computing Flow Graph
  1. Perform BFS from the nearest node q_s

  2. Backtrack from the nearest node to q_g to trim unrelated edges to a solution path (pruning)



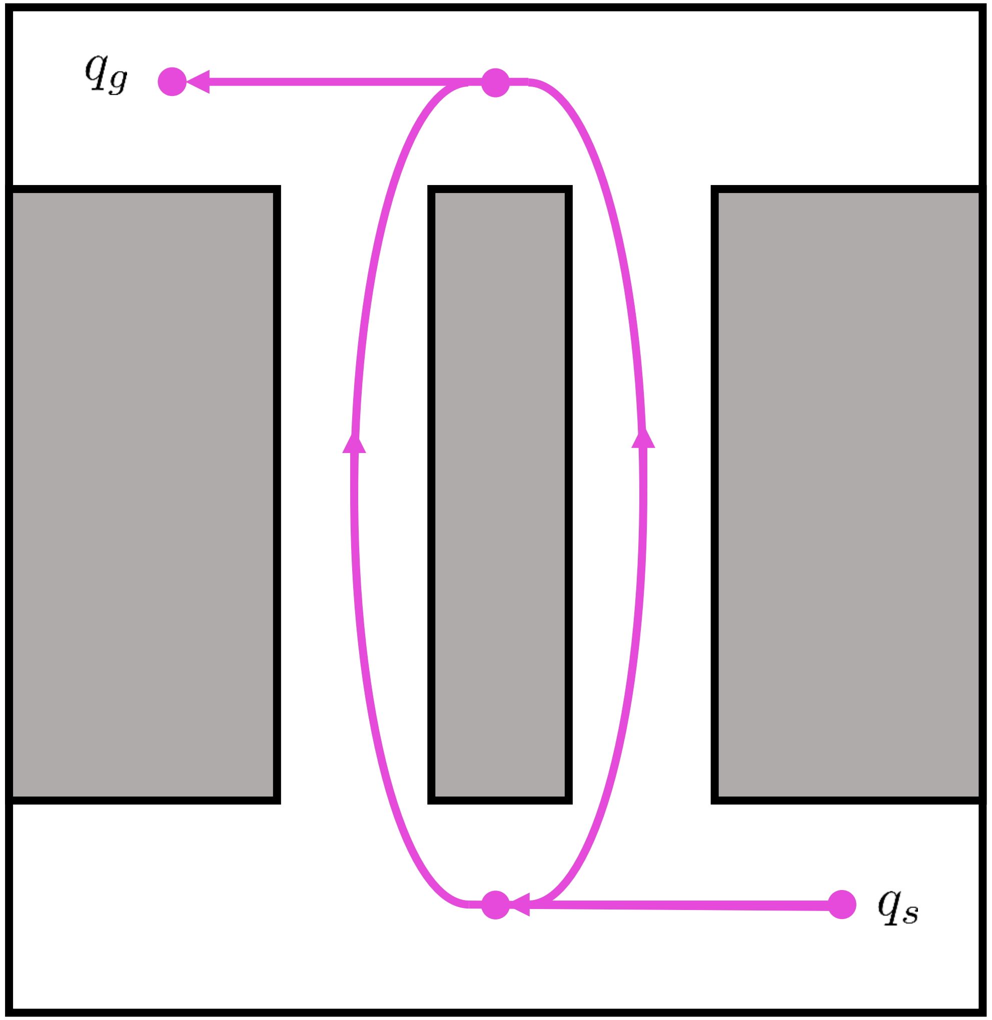
2. Flow Graph

- Computing Flow Graph
  1. Perform BFS from the nearest node q_s
  2. Backtrack from the nearest node to q_g to trim unrelated edges to a solution path (pruning)



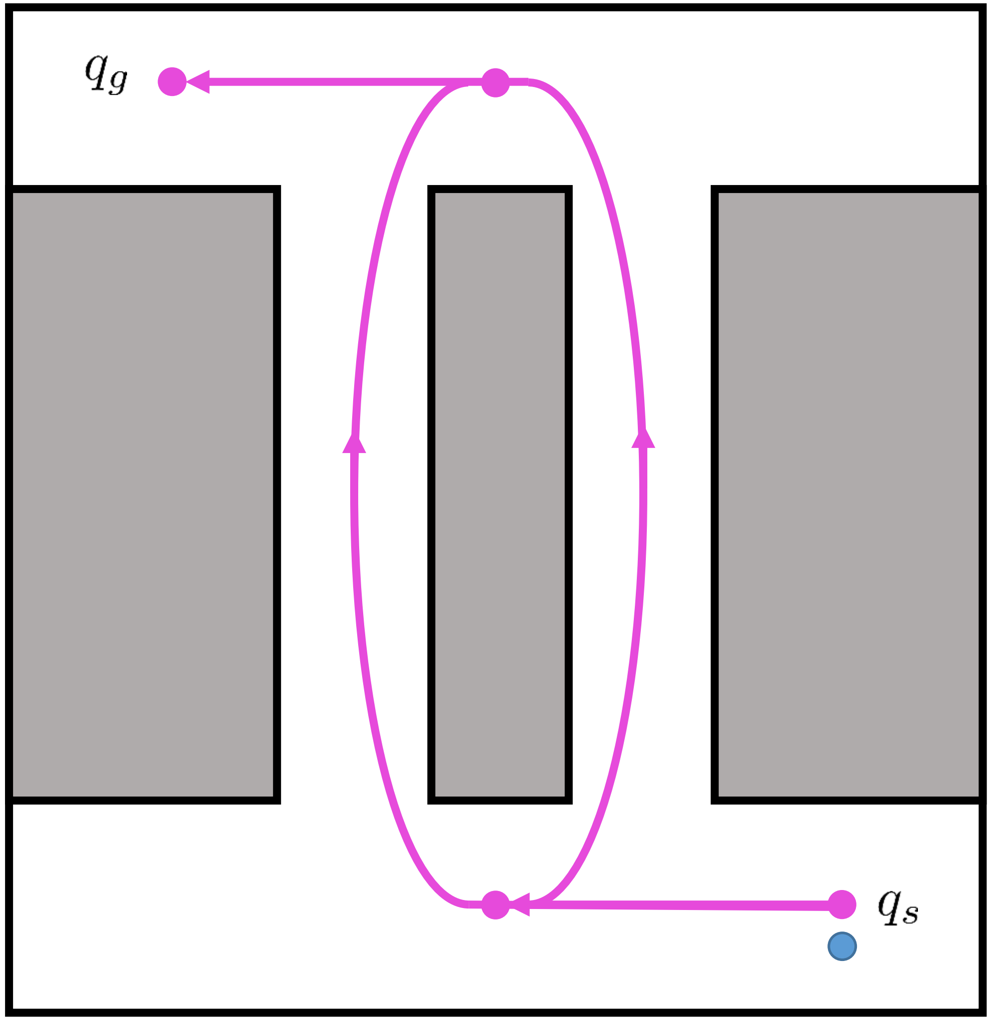
2. Flow Graph

- Computing Flow Graph
  1. Perform BFS from the nearest node q_s
  2. Backtrack from the nearest node to q_g to trim unrelated edges to a solution path (pruning)



3. Region-biased RRT Growth

- Four steps

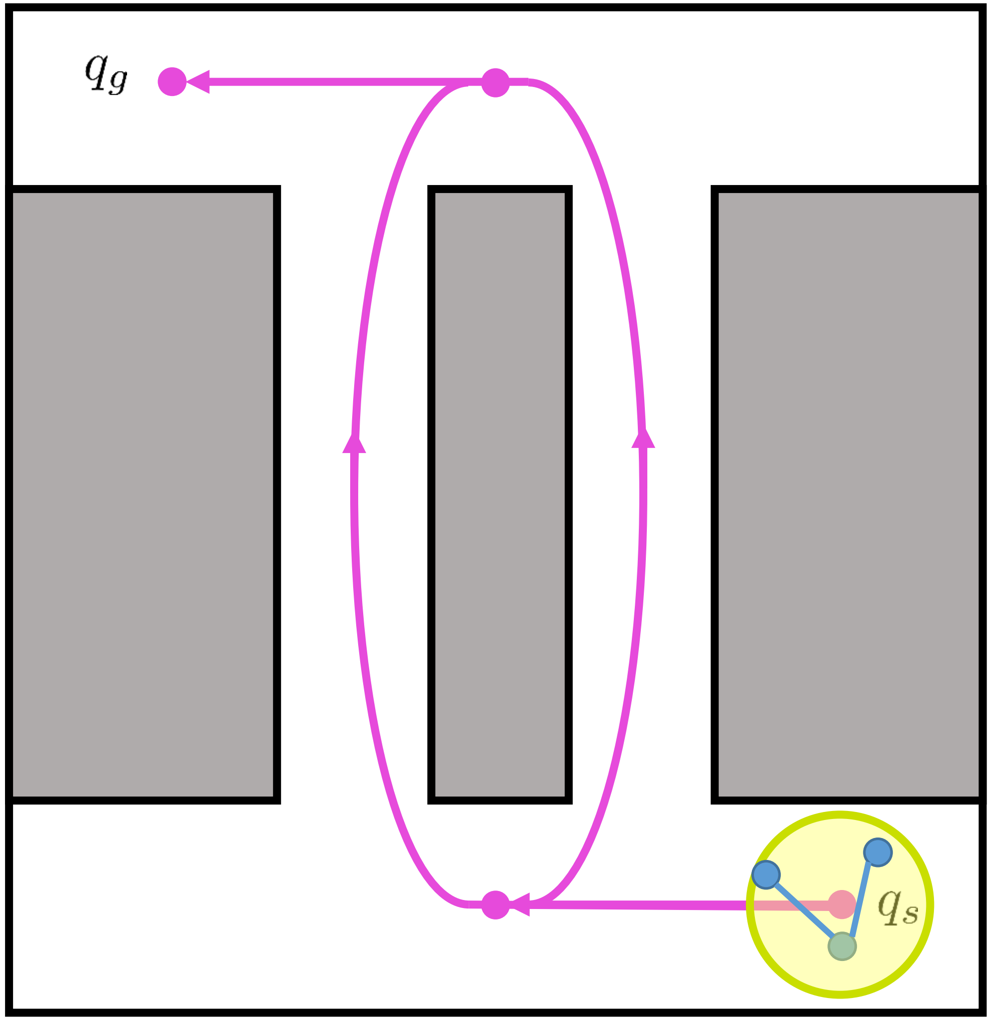


3. Region-biased RRT Growth

- Four steps

1. Region-biased RRT extension

   * Samples the region for a q_{rand} and then performs like any RRT method



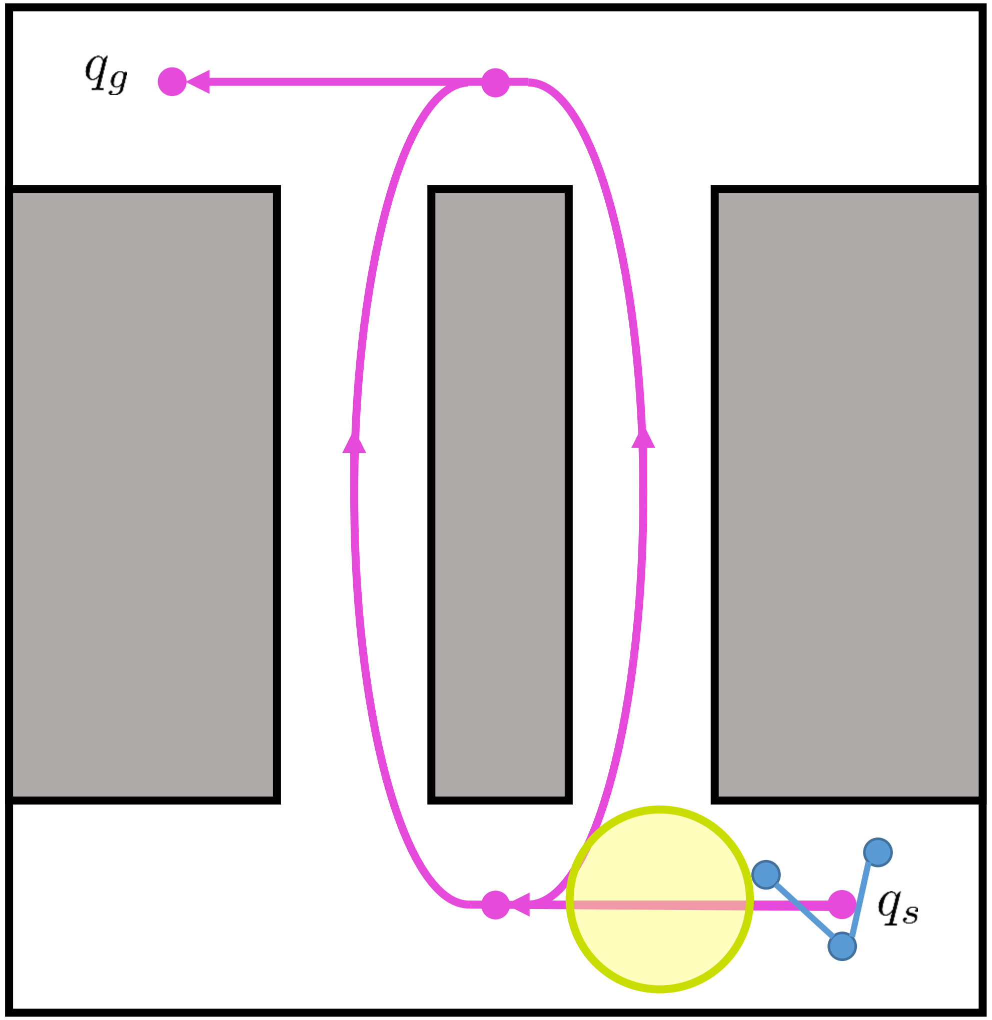
3. Region-biased RRT Growth

- Four steps

1. Region-biased RRT extension

   * Samples the region for a q_{rand} and then performs like any RRT method

2. Advance regions along flow edges



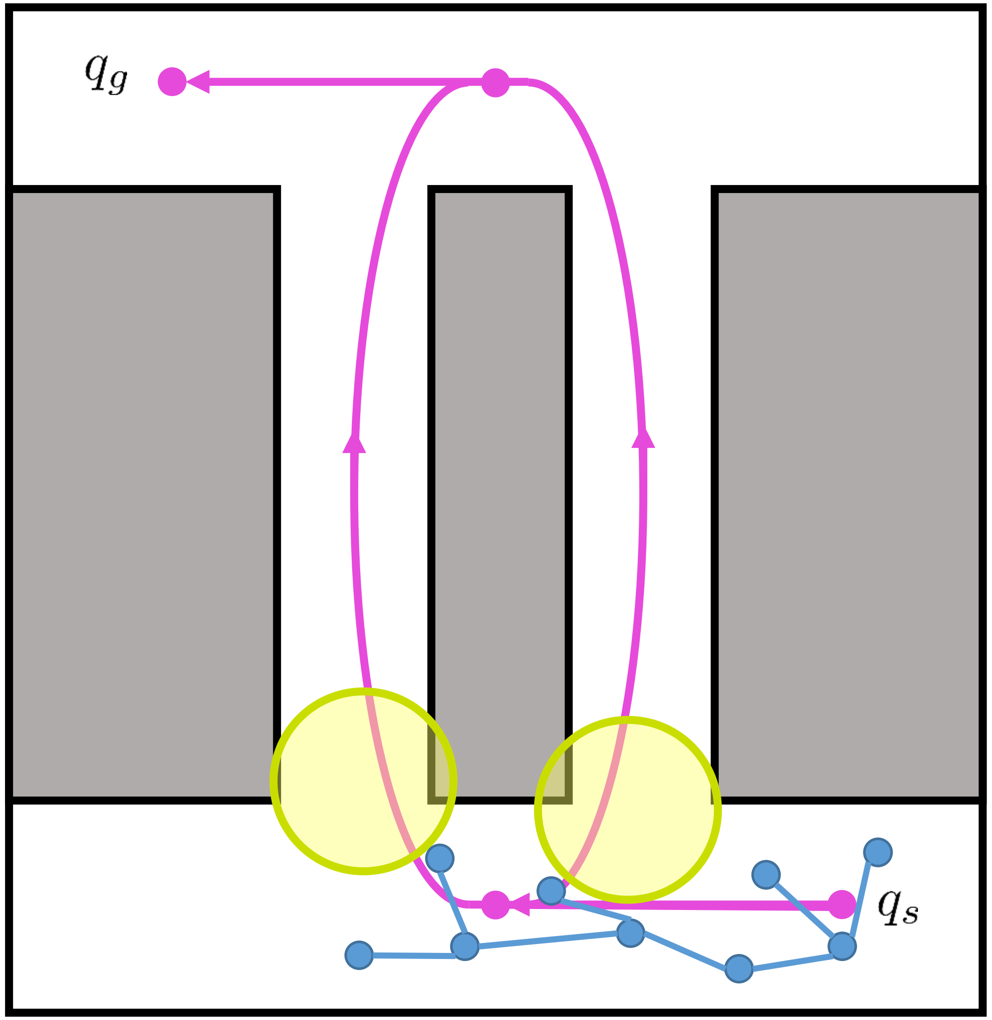
3. Region-biased RRT Growth

- Four steps

1. Region-biased RRT extension

   * Samples the region for a q_{rand} and then performs like any RRT method

2. Advance regions along flow edges
3. Delete useless regions(heuristic)
4. Create new regions



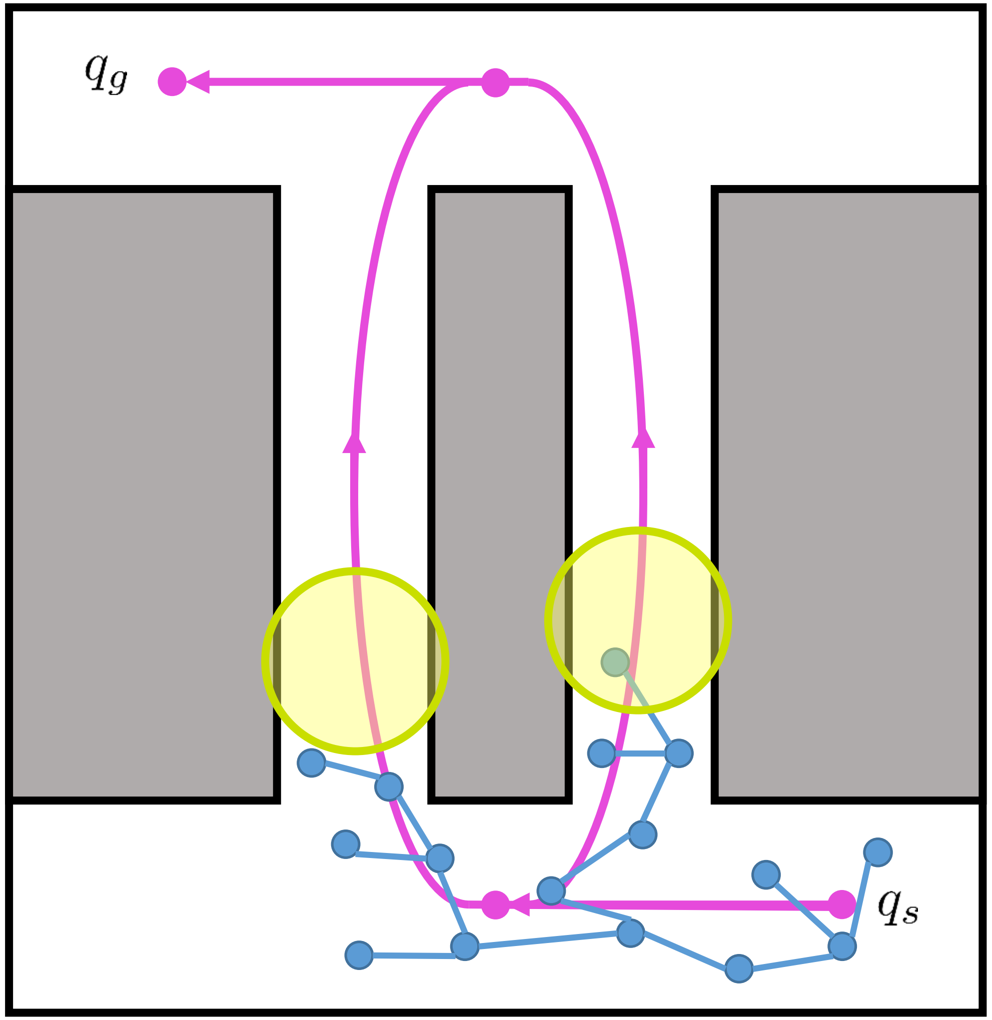
3. Region-biased RRT Growth

- Four steps

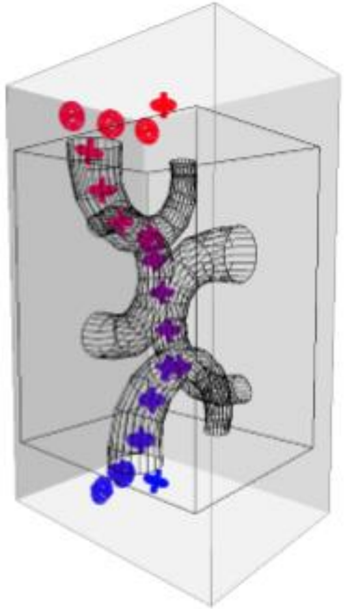
1. Region-biased RRT extension

    * Samples the region for a q_{rand} and then performs like any RRT method

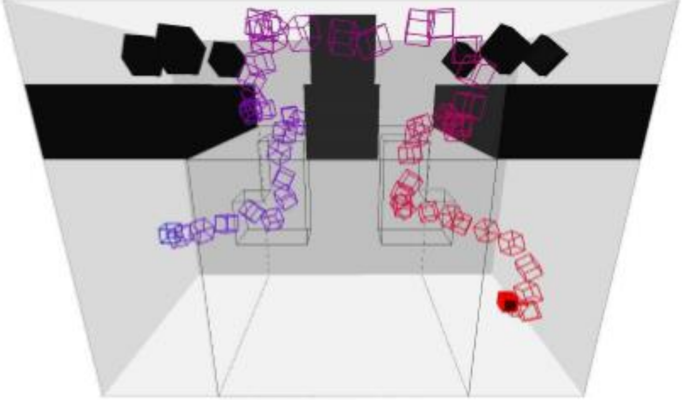
2. Advance regions along flow edges
3. Delete useless regions(heuristic)
4. Create new regions



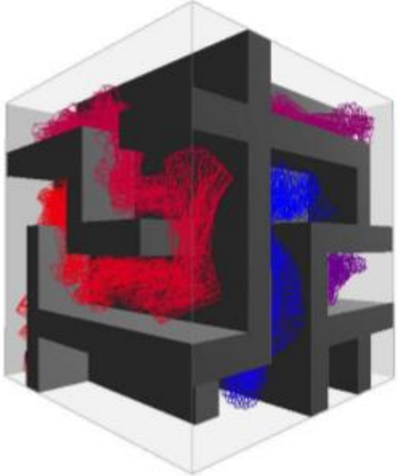
Evaluation



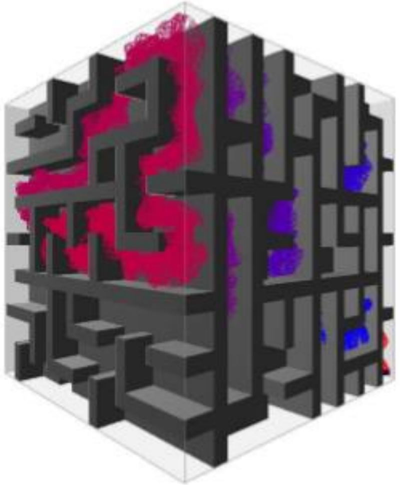
(a) MazeTunnel (toroidal plus)



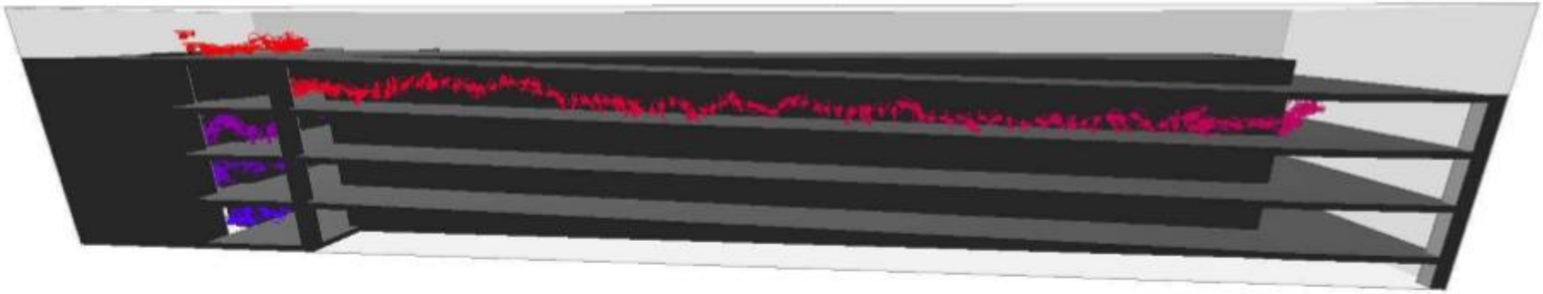
(b) LTunnel (box)



(d) GridMaze4 (stick)

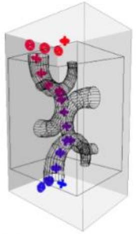


(e) GridMaze8 (stick)

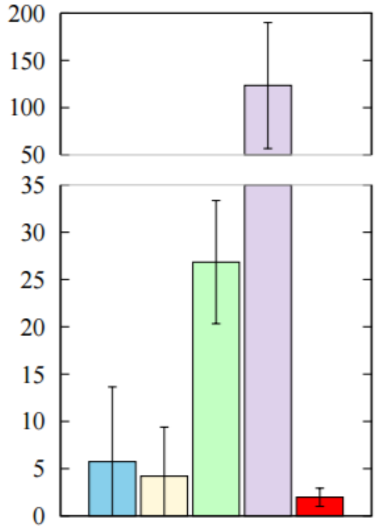


(c) Garage (helicopter)

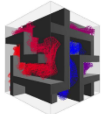
Results on Holonomic



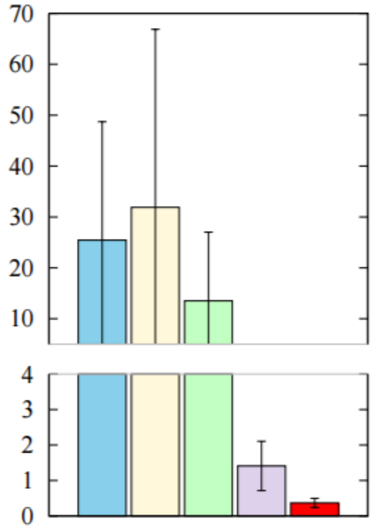
(a) MazeTunnel (toroidal plus)



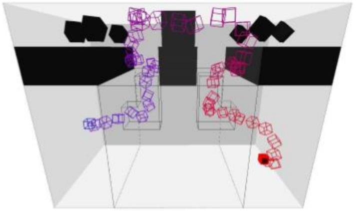
(a) MazeTunnel



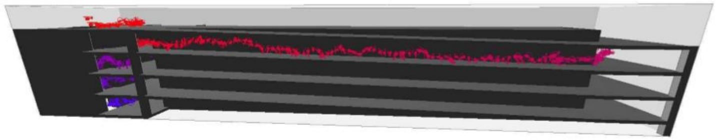
(d) GridMaze4 (stick)



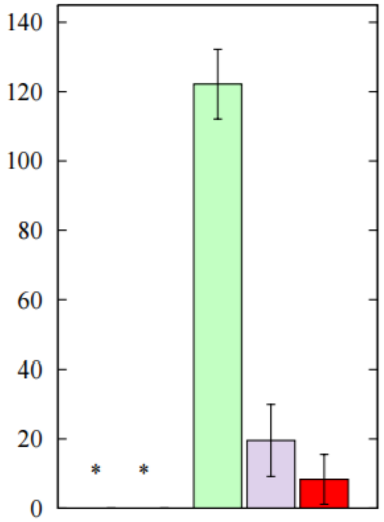
(b) LTunnel



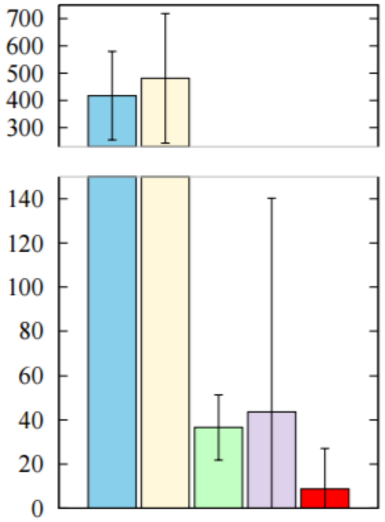
(b) LTunnel (box)



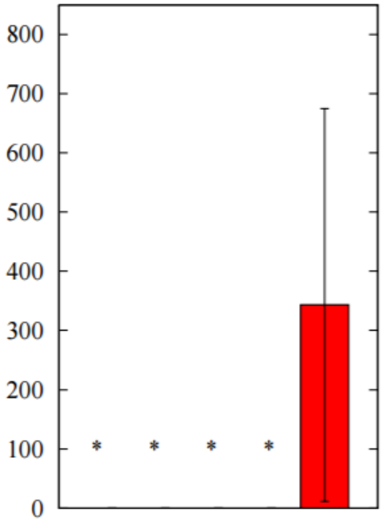
(c) Garage (helicopter)



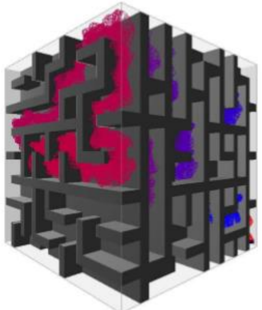
(c) Garage



(d) GridMaze4



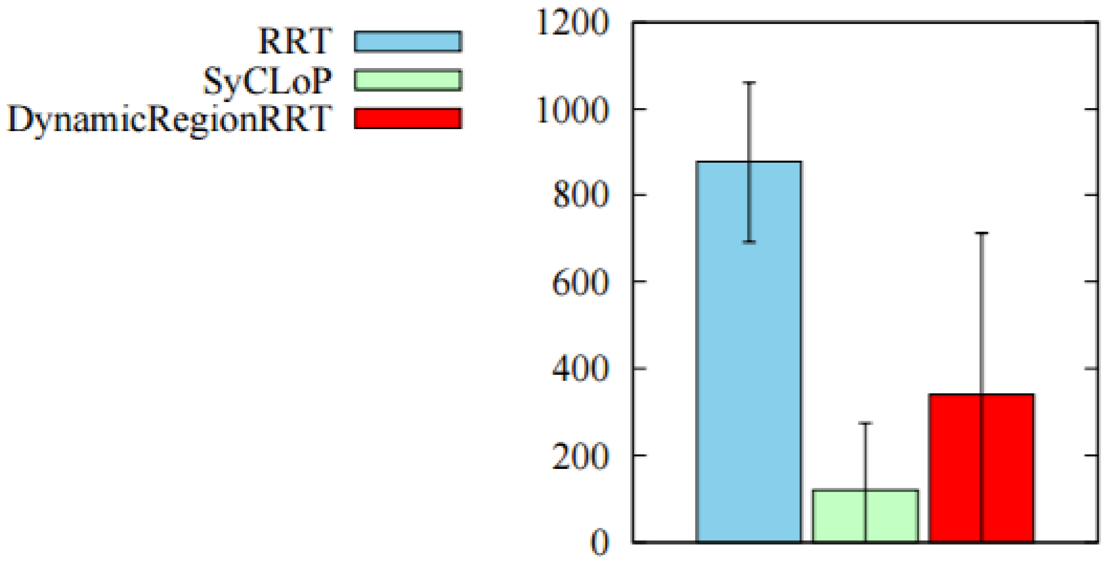
(e) GridMaze8

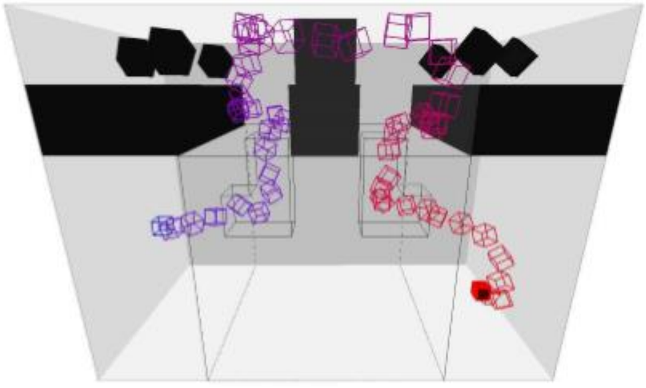


(e) GridMaze8 (stick)

Results on non-holonomic

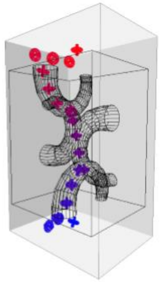
+ Dynamic biased RRT works on non-holonomic problems
- SyClop performs better
    * SyClop has faster neighbor selection routine



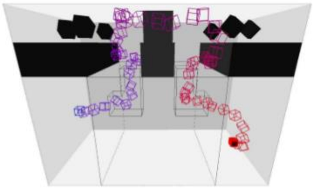


(b) LTunnel (box)

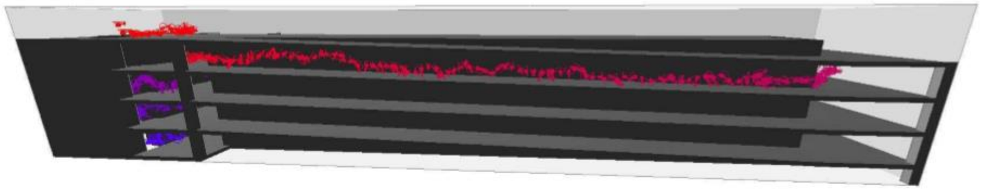
Results



(a) MazeTunnel (toroidal plus)



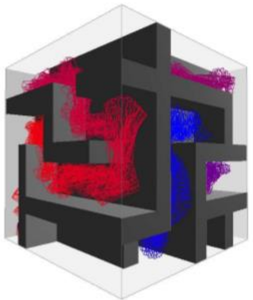
(b) LTunnel (box)



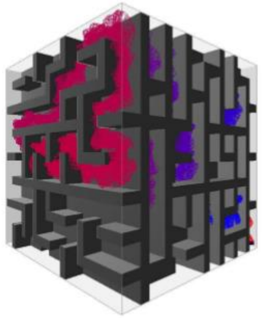
(c) Garage (helicopter)

Table 1. Success rates in each experiment.

Planner	DRRRT	RRT	Dynamic-Domain RRT	SyCLoP-RRT	WIS
MazeTunnel	100%	100%	100%	100%	100%
LTunnel	100%	100%	100%	100%	100%
Garage	100%	0%	0%	100%	100%
GridMaze4	100%	100%	100%	100%	100%
GridMaze8	76%	0%	0%	0%	0%
LTunnel (nonholonomic)	90%	24%	-	97%	-



(d) GridMaze4 (stick)



(e) GridMaze8 (stick)

Q&A