Motion planning: sampling-based planners III

Vojtěch Vonásek

Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague

Known issues of sampling-based planning

- One may consider sampling-based planning as a “magic” tool

  . . . but that’s not true at all!

Sampling-based planners have many issues

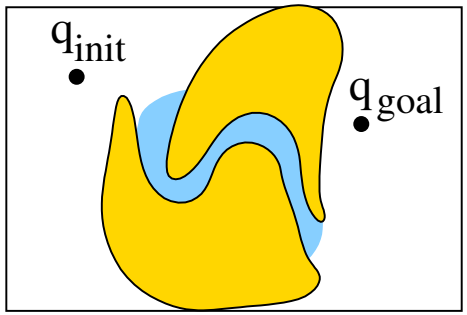
- Narrow passage problem
  - Difficulty of sampling small region in $\mathcal{C}_{\text{free}}$ surrounded by $\mathcal{C}_{\text{obs}}$
  - Problematic if (all) solutions have to pass that region
- Sensitivity to metric & parameters
  - How to measure distance in $\mathcal{C}$?
  - Selecting a good metric is as difficult as motion planning!
  - Many methods have “too many” parameters
  - Some parameters are hidden (or not well described)
  - How to tune the parameters?
- Supporting functions
  - Collision detection & nearest-neighbor search
  - Fast and reliable implementation

How do we recognize the issue? $\rightarrow$ performance measurement!

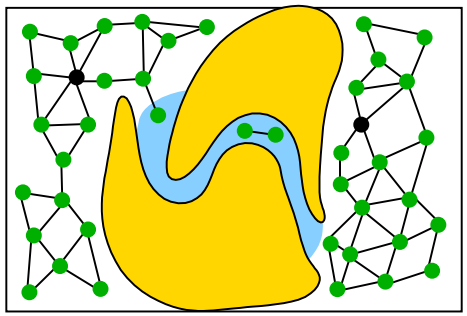
Narrow passage problem

Narrow passage (NP)

- A region $\mathcal{R} \subseteq \mathcal{C}_{\text{free}}$ with a small volume $vol(\mathcal{R}) < vol(\mathcal{C})$
- Probability that a random sample falls to $\mathcal{R}$ is $\sim vol(\mathcal{R})/vol(\mathcal{C})$
- NP are problematic if their removal changes connectivity of $\mathcal{C}_{\text{free}}$
- NP are regions in $\mathcal{C} \rightarrow$ they are given implicitly
- Location/size/volume/shape of NPs is not known!



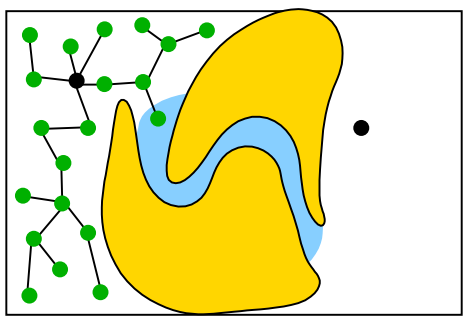
narrow passage (NP)



PRM & NP

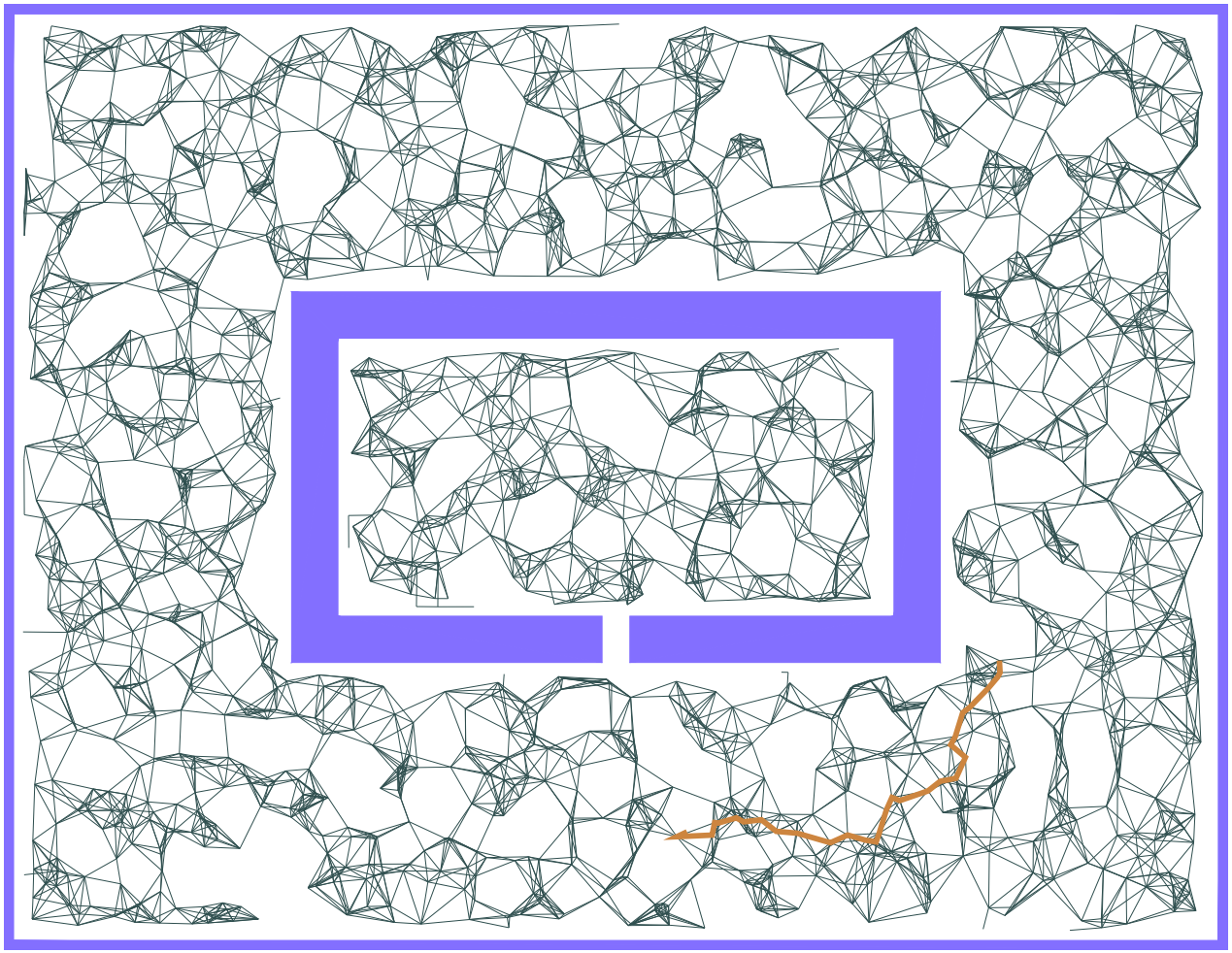
Consequences of having NP

- PRM builds unconnected roadmaps $\rightarrow$ no solution
- RRT/EST cannot enter NP $\rightarrow$ no solution
- Number of samples must be significantly increased
- Runtime is increased

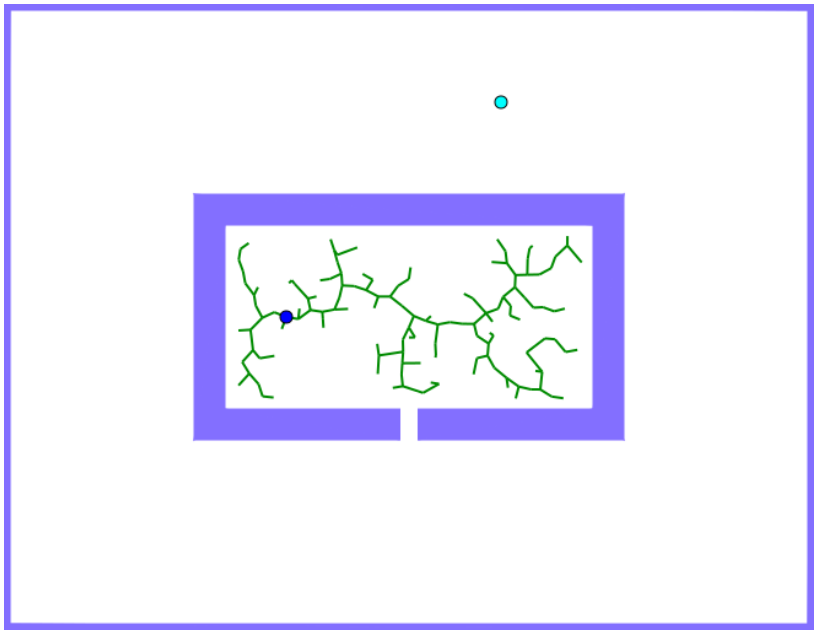


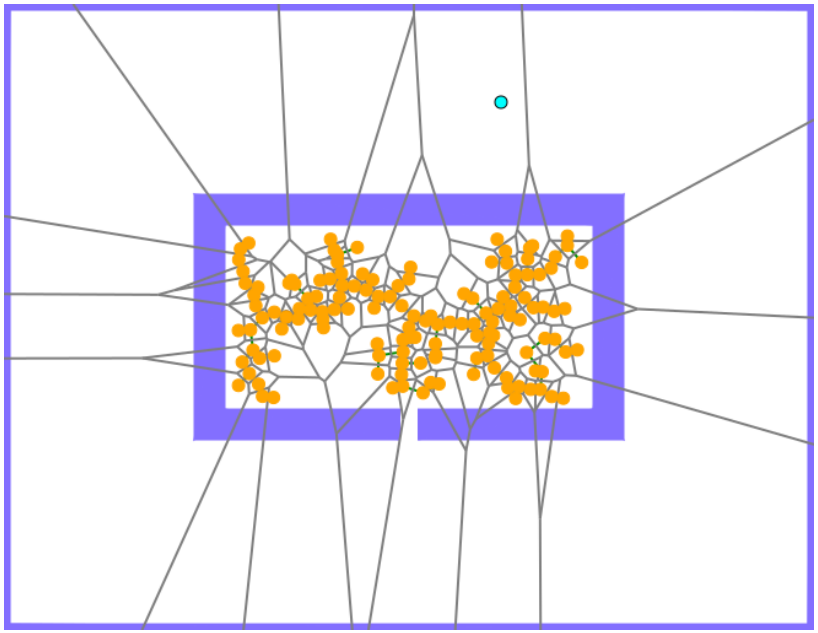
RRT/EST & NP

Narrow passage & PRM



Narrow passage & RRT



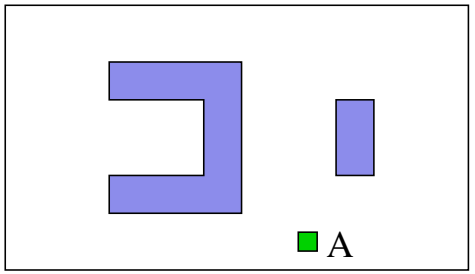


Narrow passage

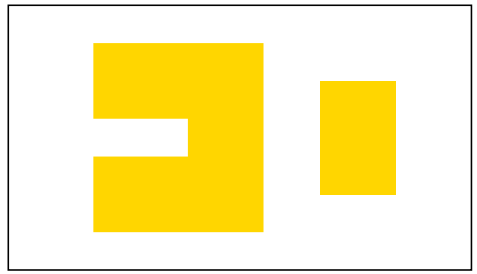
- Narrow passages are in $\mathcal{C}$
- Sometimes, we cannot (easily) see/estimate them from workspace!
- What makes the narrow passage in the Alpha-puzzle benchmark?

How does $\mathcal{C}_{\text{obs}}$ appears?

- Can we guess shape of $\mathcal{C}_{\text{obs}}$ based on workspace?
- $vol(\mathcal{A}) \ll vol(\mathcal{O})$

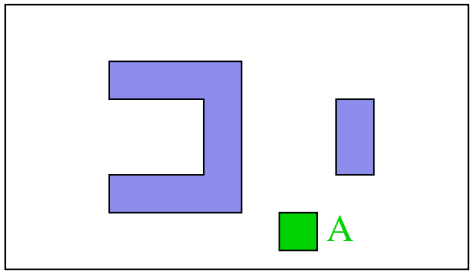


Workspace

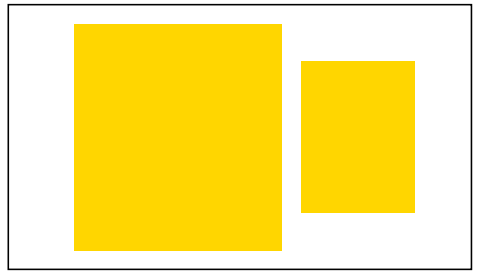


Configuration space

- $vol(\mathcal{A}) < vol(\mathcal{O})$



Workspace



Configuration space

- When obstacles $\mathcal{O}$ dominate, they mostly influence the shape of $\mathcal{C}_{\text{obs}}$

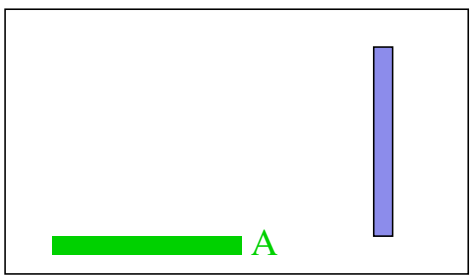
How does $\mathcal{C}_{\text{obs}}$ appear?

- Let $X, Y \subset R^n$, X and Y are nonempty
- Brunn-Minkowski theorem:

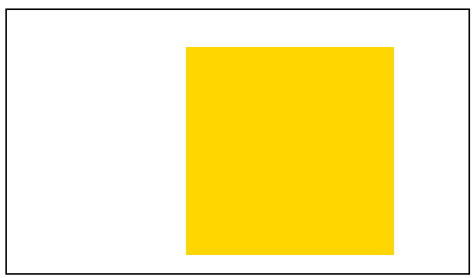
$$vol(X \oplus Y) \geq (vol(X)^{\frac{1}{n}} + vol(Y)^{\frac{1}{n}})^n$$

- $vol(\mathcal{C}_{\text{obs}})$ is larger than $\min(vol(\mathcal{A}), vol(\mathcal{O}))$
- $vol(\mathcal{C}_{\text{obs}})$ can be much larger!

Example: $vol(\mathcal{A}) = vol(\mathcal{O})$



Workspace



Configuration space

Improvements

- To cope with the narrow passage problem, improve path quality, speed-up planning, to enable planning in specific cases

Main tricks

- Change distribution of random samples
- Dedicated metrics
- Improved nearest-neighbor search
- Use suitable local planners
- Improve collision-detection

```
1  initialize tree T with q_init
2  for i = 1, ..., I_max do
3      q_rand = generate randomly in C
4      q_near = find nearest node in T towards
         q_rand
5      q_new = localPlanner from q_near towards
         q_rand
6      if canConnect(q_near, q_new) then
7          T.addNode(q_new)
8          T.addEdge(q_near, q_new)
9          if ϱ(q_new, q_goal) < d_goal then
10             return path from q_init to q_new
```

Modifications

- Many existing modifications of sampling-based planners[1][2][3][4][5]

[1] L. Zhang et al. "Motion Planning for Robotics: A Review for Sampling-Based Planners". In: *Biomimetic Intelligence and Robotics* 5.1 (Mar. 2025), p. 100207. DOI: 10.1016/j.birob.2024.100207.

[2] T. McMahon et al. "A survey on the integration of machine learning with sampling-based motion planning". In: *Foundations and Trends® in Robotics* 9.4 (2022), pp. 266–327.

[3] X. Xiao et al. "Motion planning and control for mobile robot navigation using machine learning: a survey". In: *Autonomous Robots* 46.5 (2022), pp. 569–597. DOI: https://doi.org/10.1007/s10514-022-10039-8.

[4] J. Wang et al. "A survey of learning-based robot motion planning". In: *IET Cyber-Systems and Robotics* 3.4 (2021), pp. 302–314. DOI: https://doi.org/10.1049/csy2.12020.

[5] L. G. D. O. Váras et al. "Systematic Literature Review of Sampling Process in Rapidly-Exploring Random Trees". In: *IEEE Access* 7 (2019), pp. 50933–50953. DOI: 10.1109/ACCESS.2019.2908100.

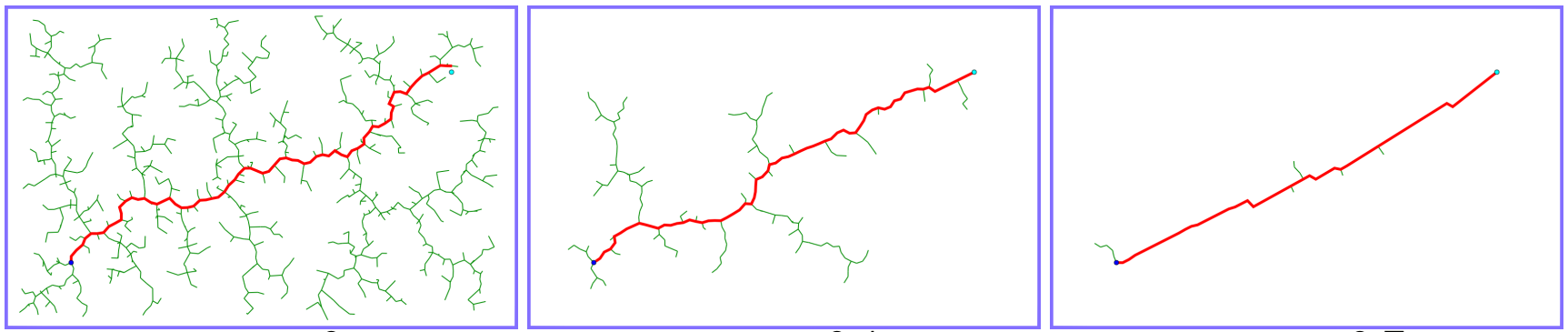
RRT modification: goal bias

Observation

- RRT tree grows towards random samples
- If we samples some region more dense, the tree is “attracted” to grow there

Goal-bias

- Random sample q_{rand} is generated in $\mathcal{C}$ with probability $(1 - p_{goal})$, otherwise it is set to $q_{\text{rand}} = q_{\text{goal}}$
- The rest of RRT algorithm is the same
- Improves the performance if the tree can directly reach the goal
- Decreases the performance if the tree is hindered by obstacles



$p_{goal} = 0$ $p_{goal} = 0.1$ $p_{goal} = 0.7$

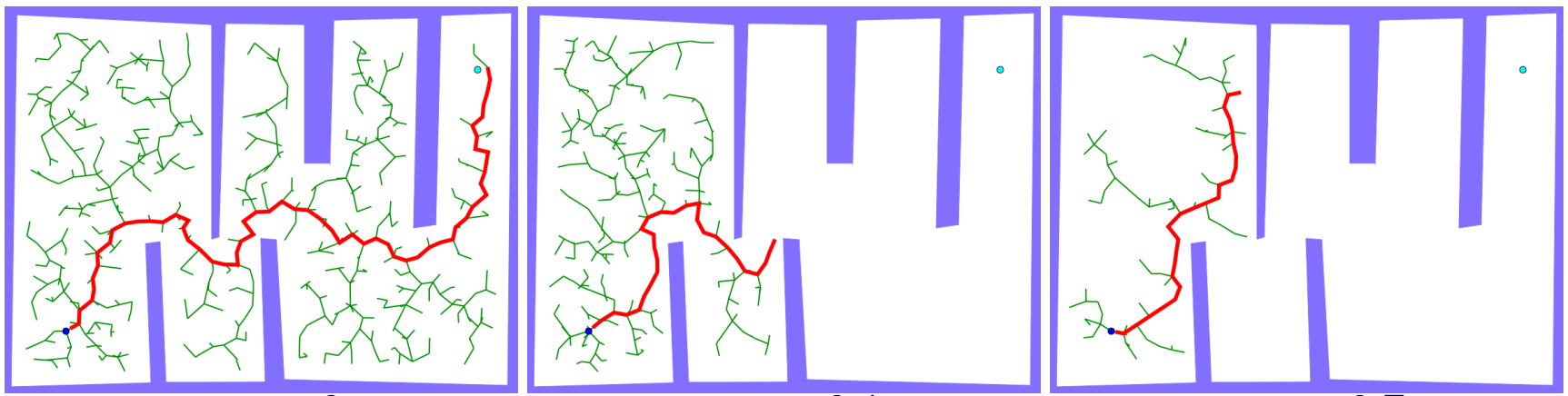
RRT modification: goal bias

Observation

- RRT tree grows towards random samples
- If we samples some region more dense, the tree is “attracted” to grow there

Goal-bias

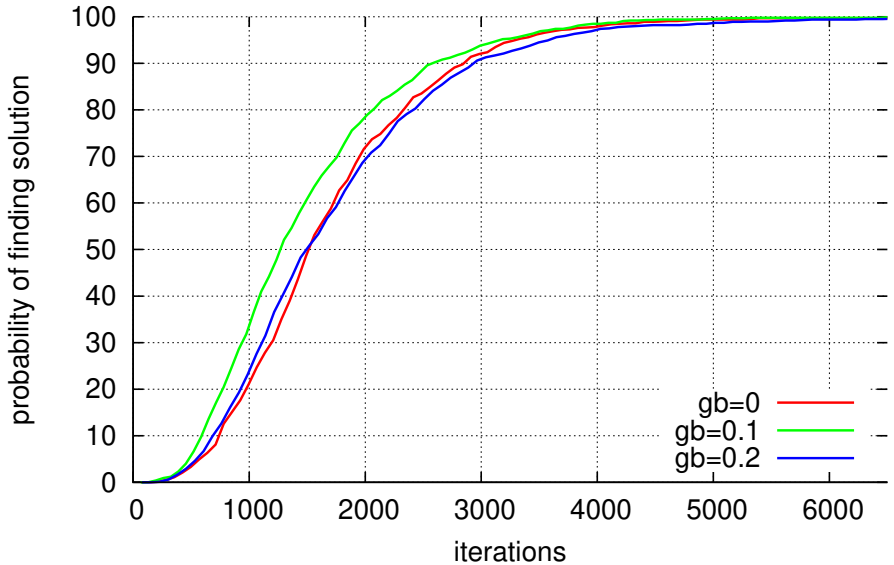
- Random sample q_{rand} is generated in $\mathcal{C}$ with probability $(1 - p_{goal})$, otherwise it is set to $q_{\text{rand}} = q_{\text{goal}}$
- The rest of RRT algorithm is the same
- Improves the performance if the tree can directly reach the goal
- Decreases the performance if the tree is hindered by obstacles

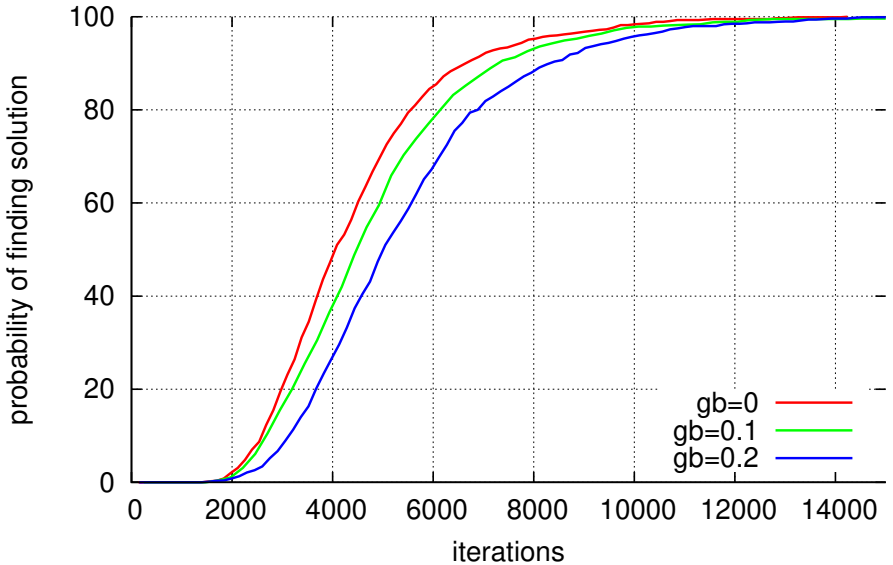


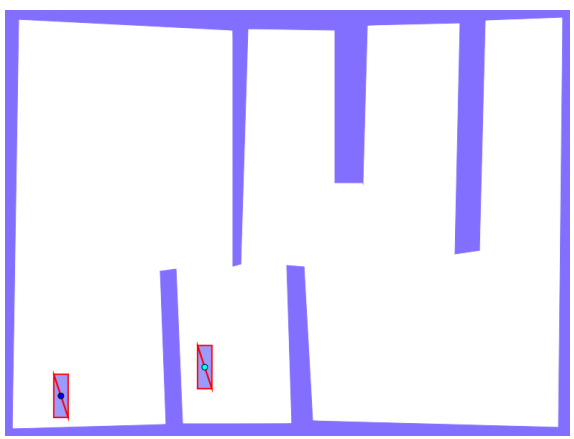
$p_{goal} = 0$ $p_{goal} = 0.1$ $p_{goal} = 0.7$

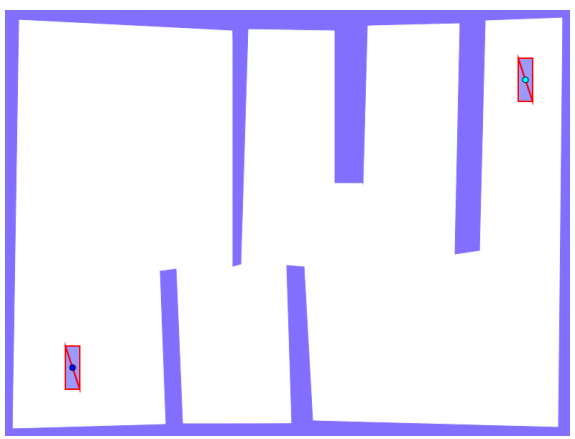
RRT modification: goal bias

- Goal-bias may improve or even worse the performance!





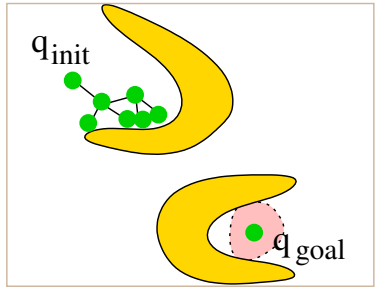




RRT modification: guided sampling

Observation

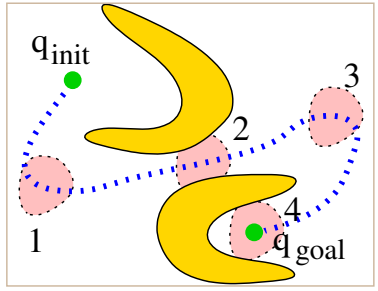
- Goal-bias attracts the tree towards q_{goal}, but the tree may be blocked by obstacles
- Generalization: we can attract the tree toward any region $\mathcal{R} \subseteq \mathcal{C}$ if we sample $\mathcal{R}$ densely



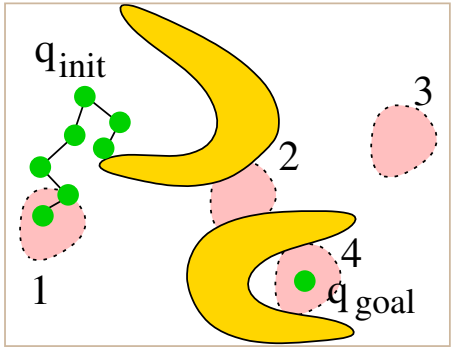
RRT + goal-bias

Guided-based sampling

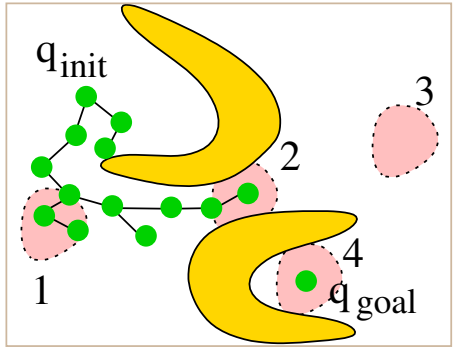
- Estimate a path that can "guide" the tree in the $\mathcal{C}$-space
- Generate q_{rand} around the path-waypoints (starting from first waypoint) until the tree reaches the waypoint
- Then generate q_{rand} around the next waypoint



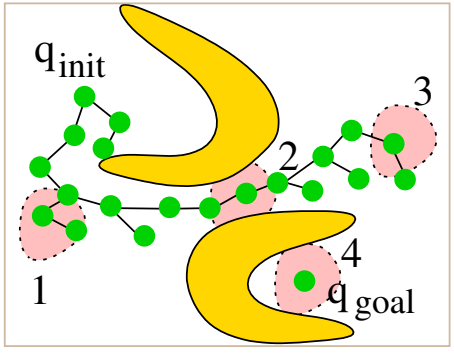
Guiding path



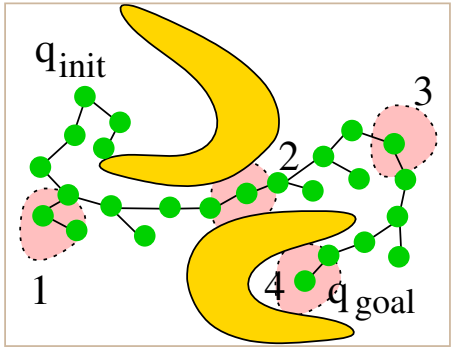
Sampling at 1



Sampling at 2

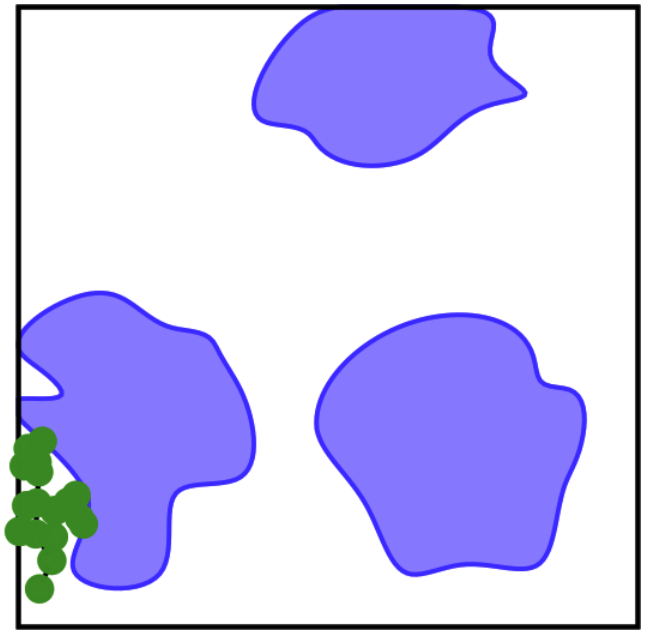


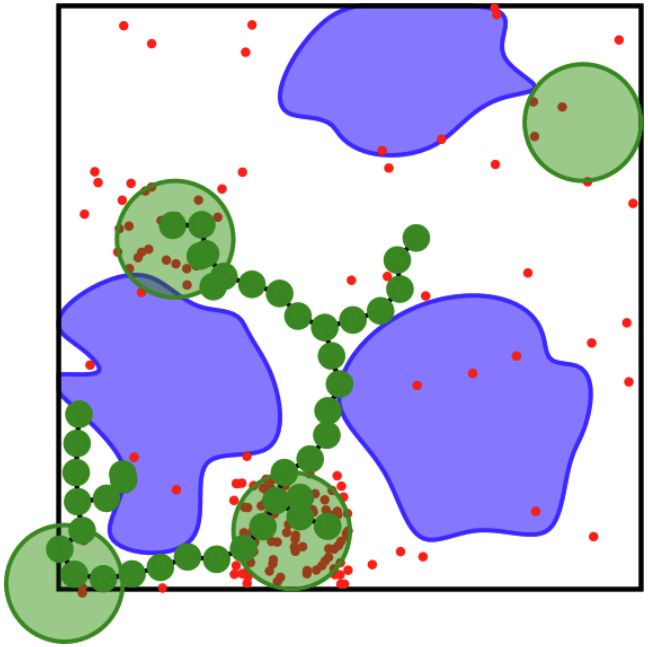
Sampling at 3



Sampling at 4

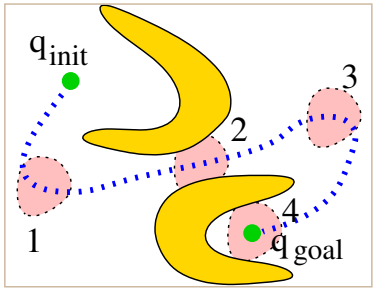
Guided sampling

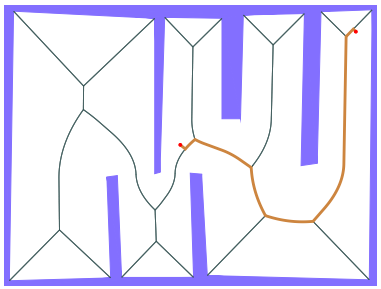




How to compute the guiding path?

- Generally, the guiding path has to be located in $\mathcal{C}$!!
- Finding a good guiding path has the same complexity as the original planning problem!
- (i.e., guiding sampling is 'planning solved by planning')
- Practically, we have two options





Guiding path in $\mathcal{W}$
$q = (x, y, \varphi)$
(x, y) from the path
φ randomly

Guiding path in $\mathcal{W}$

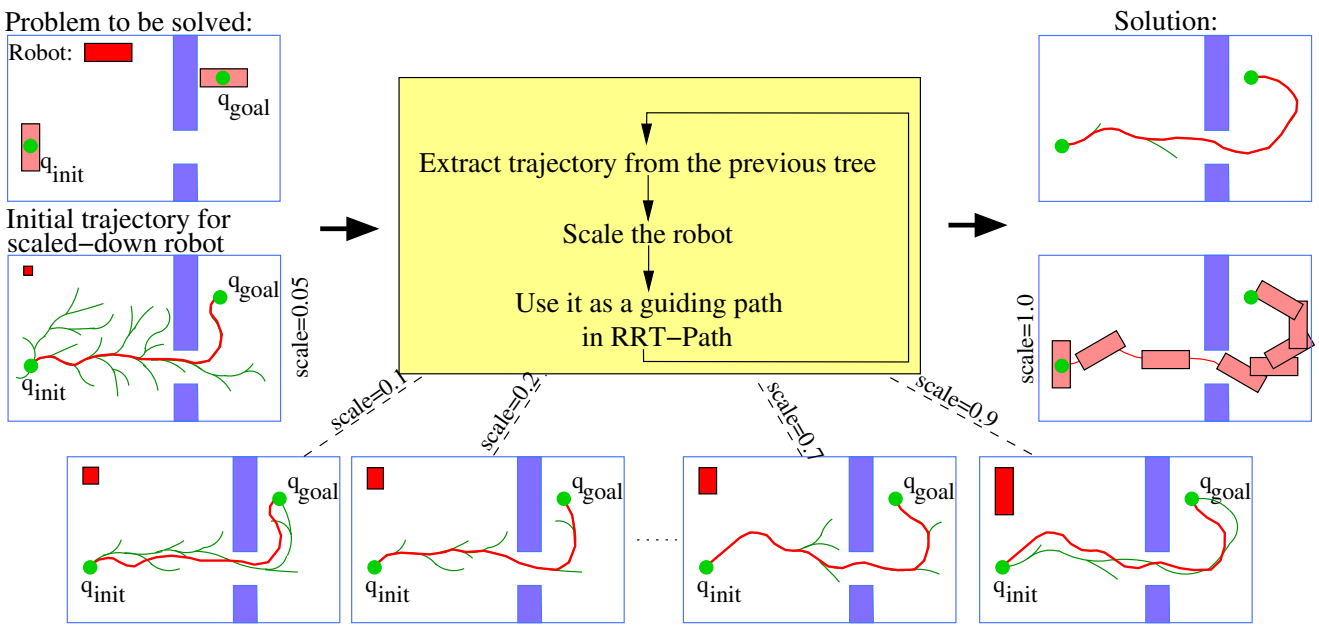
- Path is computed in workspace — geometric planning (Voronoi diagram, Visibility graph, etc.)
- Suitable for low-dimensional problems
- The remaining dimensions are sampled uniformly

Guiding path in $\mathcal{C}$

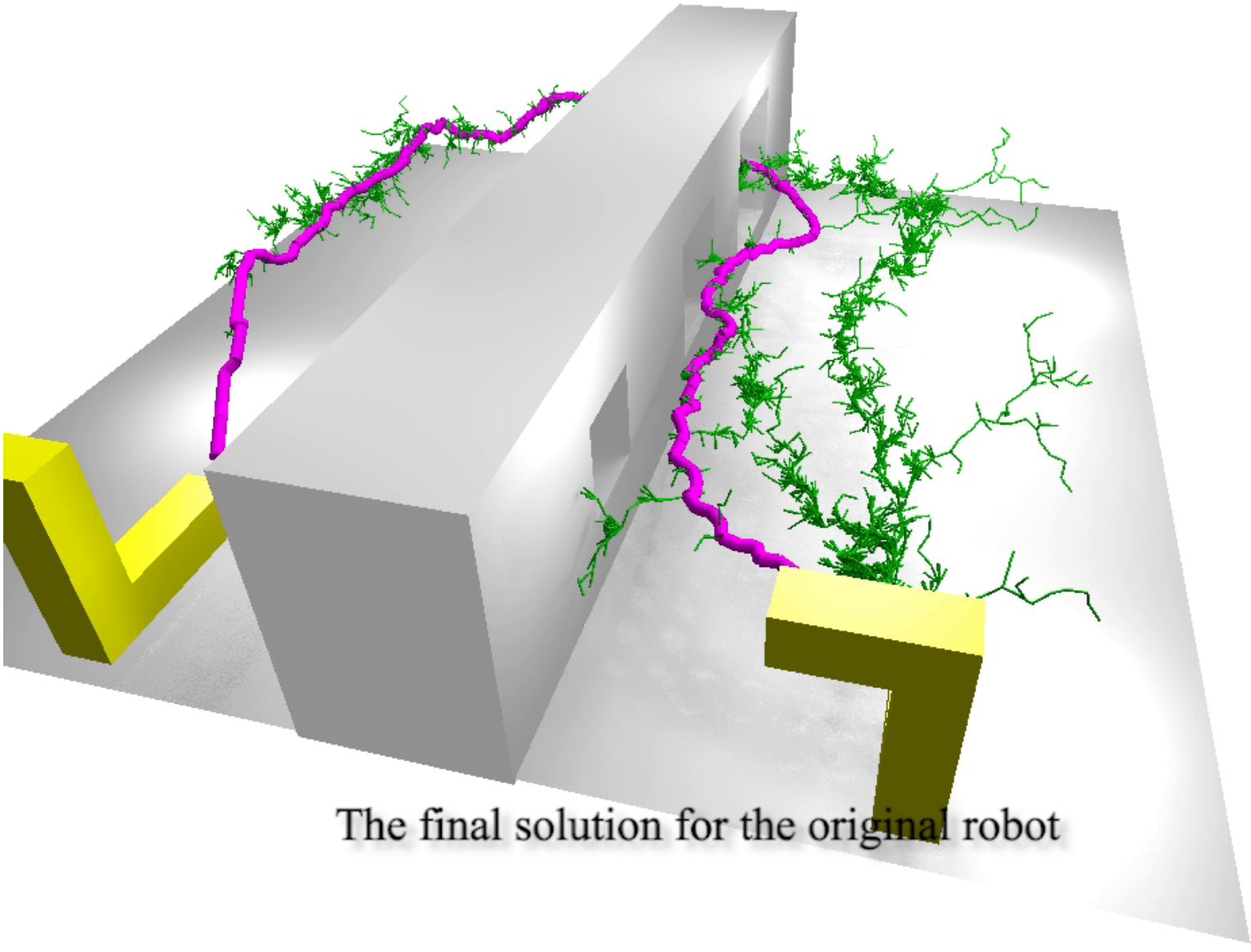
- Path is computed in $\mathcal{C}$ by a simplified search

Computing guiding path in $\mathcal{C}$

- Problem is simplified — relaxation of constraints
- For example, robot is scaled-down
- Solve simplified planning problem
- Use the solution to generate random samples along it
- The process can be iterative

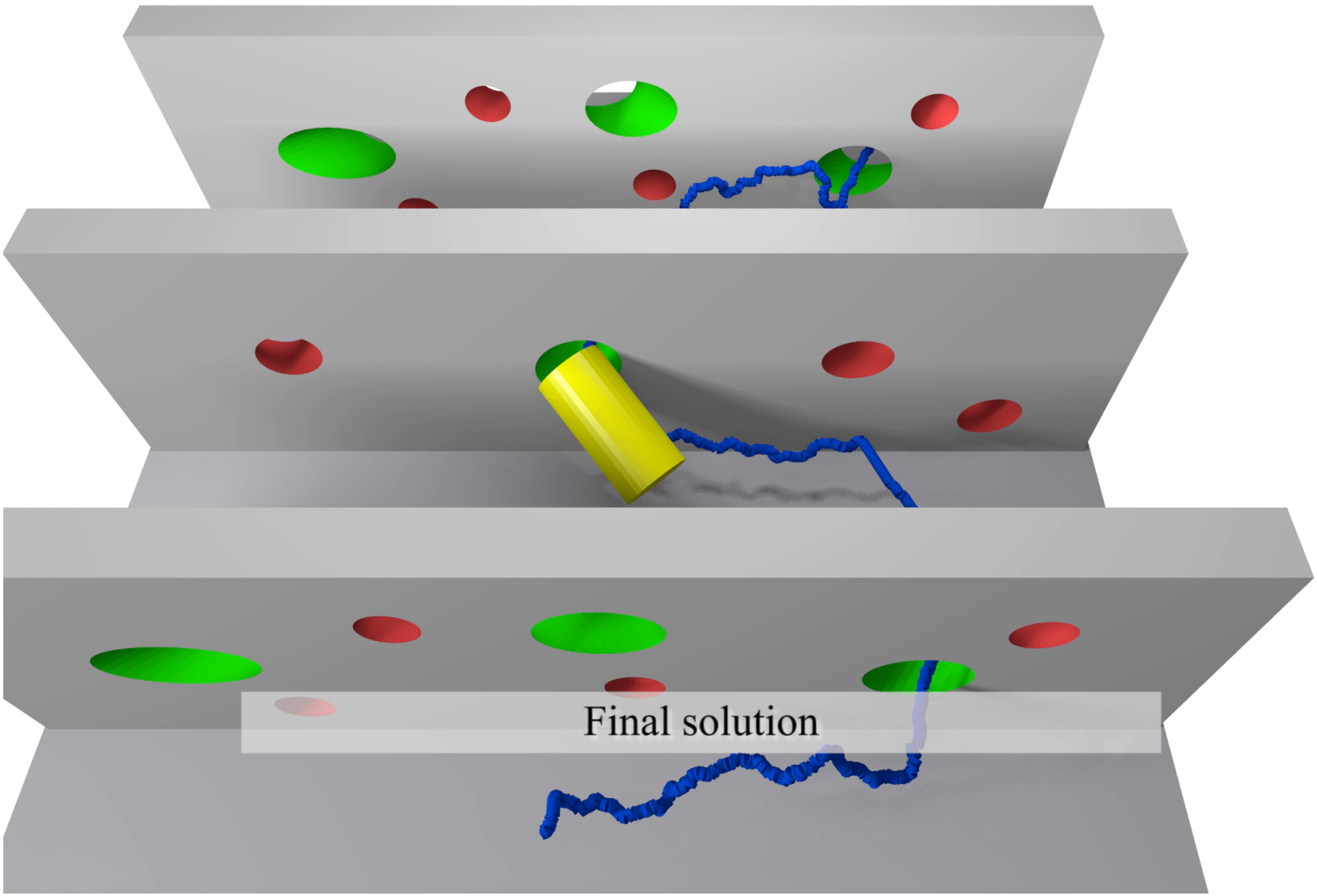


Computing guiding path in $\mathcal{C}$

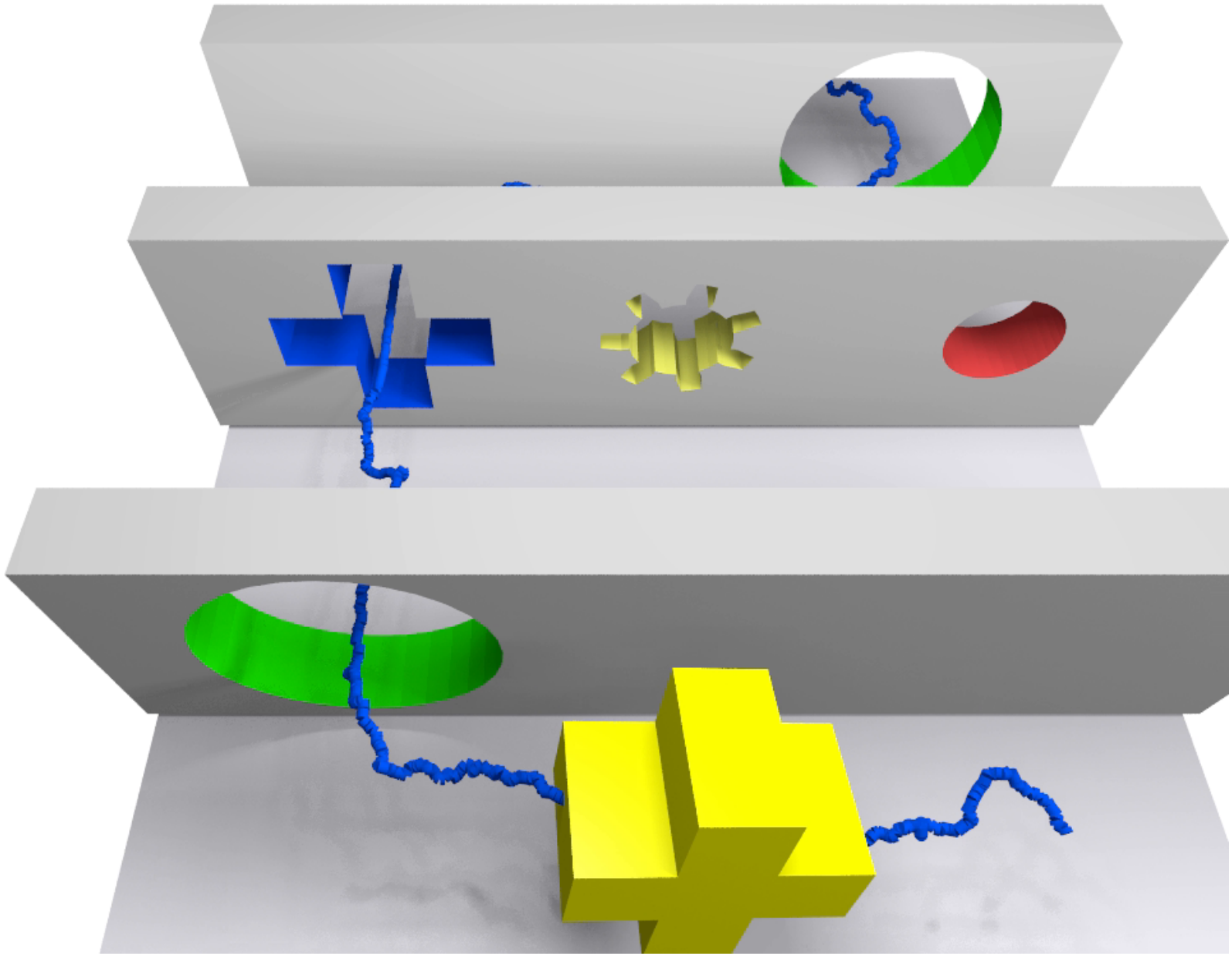


The final solution for the original robot

Computing guiding path in $\mathcal{C}$



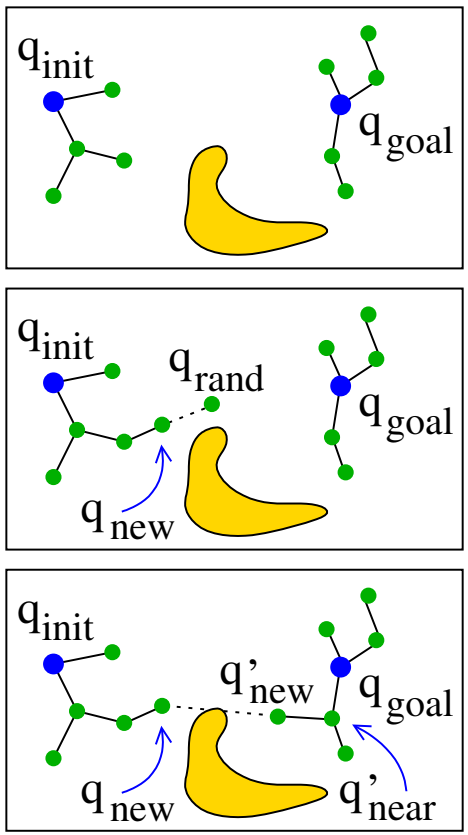
Computing guiding path in $\mathcal{C}$



RRT modification: bidirectional search

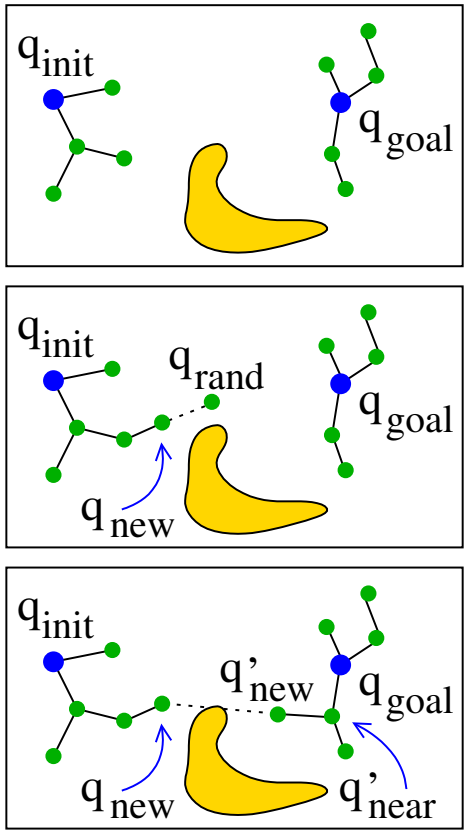
- Use two trees: $\mathcal{T}_i$ rooted at q_{init}, $\mathcal{T}_g$ rooted q_{goal}
- One tree expands towards q_{rand}, second tree expands towards q_{new} of the first tree

```
1  T_i.addNode(q_init)
2  T_g.addNode(q_goal)
3  for i = 1, ..., I_max do
4      q_rand = generate randomly in C
5      q_near = find nearest node in T_i towards q_rand
6      q_new = localPlanner from q_near towards q_rand
7      if canConnect(q_near, q_new) then
8          T_i.addNode(q_new)
9          T_i.addEdge(q_near, q_new)
10         q'_near = find nearest node in T_g towards q_new
11         q'_new = localPlanner from q_near towards q_rand
12         if canConnect(q'_near, q'_new) then
13             T_g.addNode(q_new)
14             T_g.addEdge(q_near, q_new)
15             if canConnect(q'_new, q_new) then
16                 joint trees
17                 return path from q_init to q_goal
18     T_i, T_g = T_g, T_i                          // swap trees
```



RRT modification: bidirectional search

- Use two trees: $\mathcal{T}_i$ rooted at q_{init}, $\mathcal{T}_g$ rooted q_{goal}
- One tree expands towards q_{rand}, second tree expands towards q_{new} of the first tree
- Helps to enter narrow passages (sometimes)
- Connection of two trees
  - Computationally intensive
  - To speed up, performs only if $\varrho(q_{\text{new}}, q'_{\text{new}})$ is small enough
  - Difficult if motion model/constraints have to be considered
- Balanced trees: swap trees if $|\mathcal{T}_i| > |\mathcal{T}_g|$



PRM variants: sampling strategies

Original PRM/sPRM

- Uniform sampling $q \sim U(\mathcal{C})$

Gaussian sampling: two-samples

- Uniform sample $q_1 \sim U(\mathcal{C})$, then another sample $q_2 \sim N(q, \Sigma)$ (around q_1 from Gaussian distribution)
- Ignore if $q_1, q_2 \in \mathcal{C}_{\text{free}}$ or $q_1, q_2 \in \mathcal{C}_{\text{obs}}$, otherwise
- add the collision-free one to the roadmap
- Generates the random samples near $\mathcal{C}_{\text{obs}}$ only!

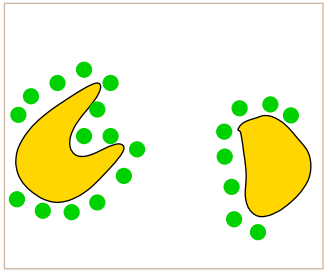
Gaussian + uniform

- Combination of two previous methods
- More dense sampling around $\mathcal{C}_{\text{obs}}$ than basic PRM

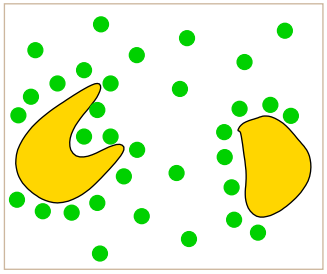
Bridge test

- Generate q_1 and q_2 using the Gaussian method
- Determine the midpoint q' on the line segment $|q_1, q_2|$
- Use q' if $q' \in \mathcal{C}_{\text{free}}$ and $q_1, q_2 \in \mathcal{C}_{\text{obs}}$

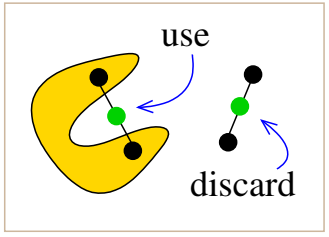
Uniform



Gaussian



Gaussian + Uniform



Bridge-Test

Lazy evaluation

Main idea: postpone time-consuming routines until they are needed

Lazy sampling-based planning[6789]

- Postpone collision detection on edges/configurations
- Approximate collision detection on edges
- Do exact collision detection only when the edge/configuration is part of a solution

[6] A. Mandalika et al. "Generalized Lazy Search for Robot Motion Planning: Interleaving Search and Edge Evaluation via Event-based Toggles". In: *CoRR* abs/1904.02795 (2019). arXiv: 1904.02795.

[7] J. D. Hernández et al. "Lazy evaluation of goal specifications guided by motion planning". In: *2019 International Conference on Robotics and Automation*. 2019, pp. 944–950.

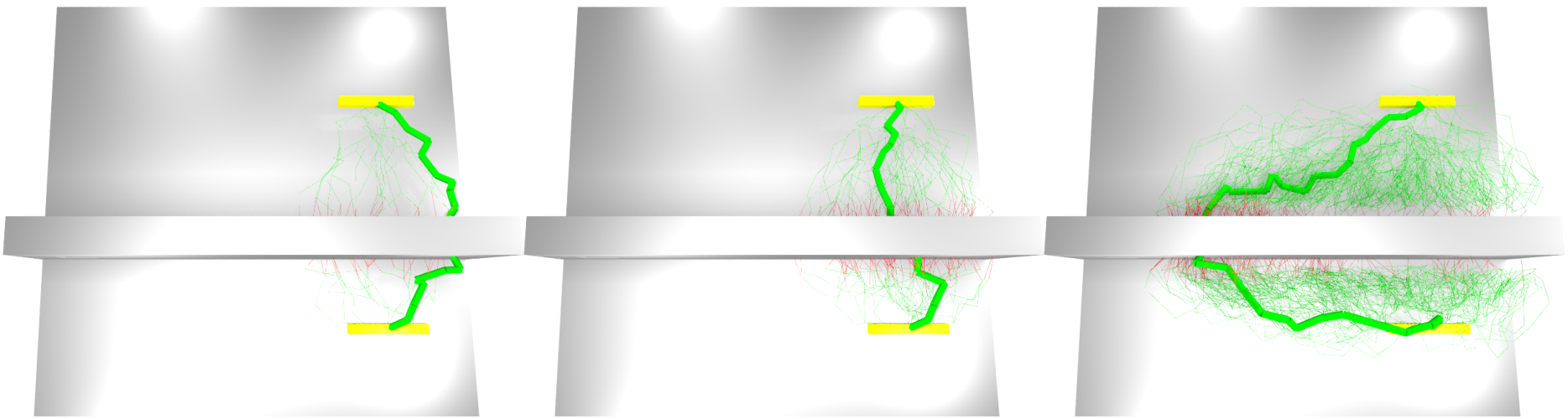
[8] K. Hauser. "Lazy collision checking in asymptotically-optimal motion planning". In: *IEEE International Conference on Robotics and Automation*. 2015, pp. 2951–2957. DOI: 10.1109/ICRA.2015.7139603.

[9] J. Denny et al. "Lazy toggle prm: A single-query approach to motion planning". In: *2013 IEEE International Conference on Robotics and Automation*. IEEE. 2013, pp. 2407–2414.

PRM variants: lazy evaluation

Lazy PRM[10]

- Build PRM roadmap without collision detection of edges
- Find path in it, check collisions only for path edges
- Recalculate path is some edge is not collision free
- If no path is found, extend the roadmap by new samples/edges; repeat
- Otherwise, the path is collision-free



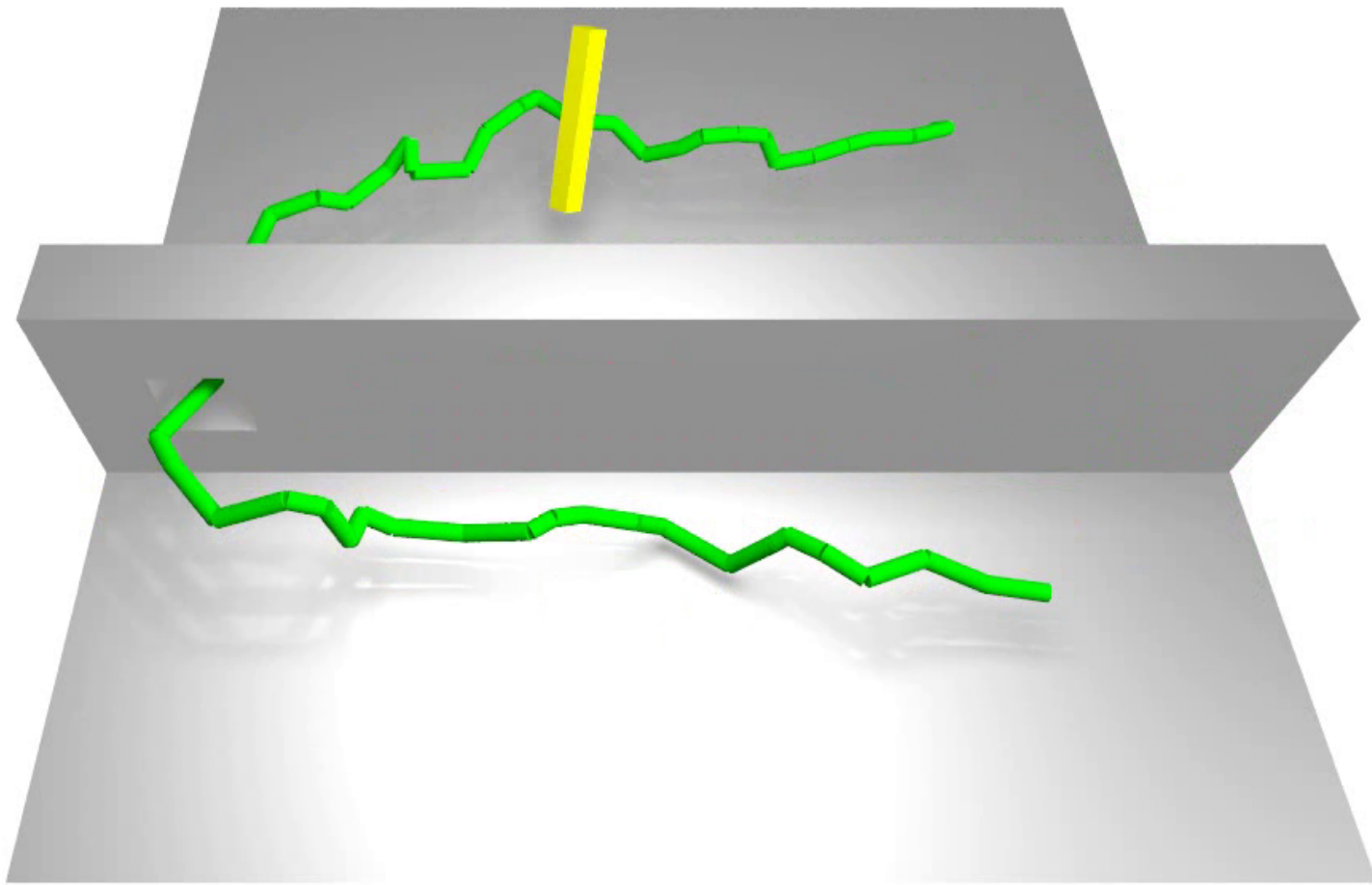
∼200 nodes ∼500 nodes ∼2000 nodes

green: valid edges, red: invalid edges

- Faster planning in certain scenarios, but not always!

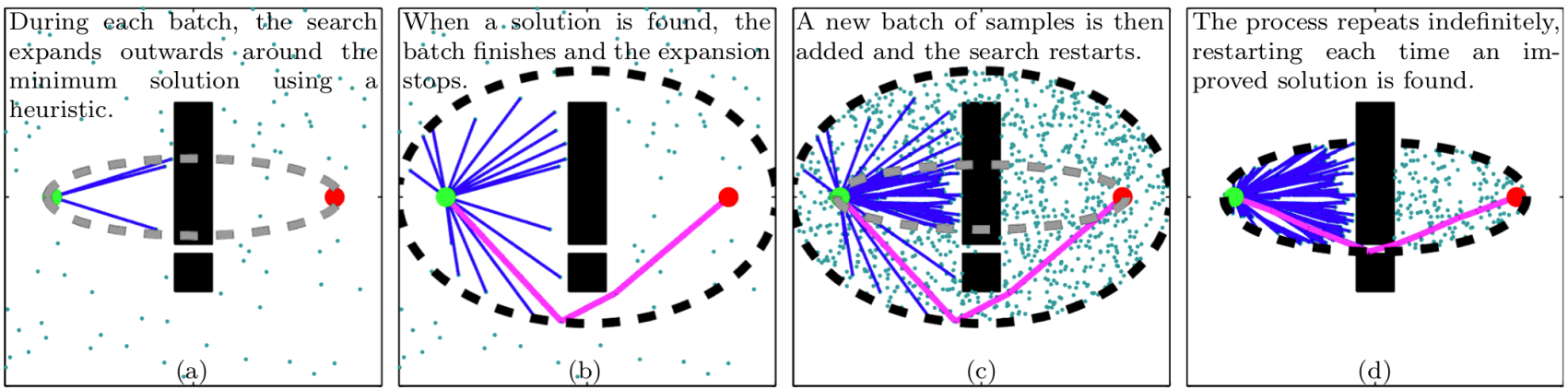
[10] R. Bohlin and L. E. Kavraki. "Path planning using lazy PRM". In: *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*. Vol. 1. IEEE. 2000, pp. 521–528.

PRM variants II: Lazy PRM



Batch informed Tree (BIT*)[11] — most advanced planner

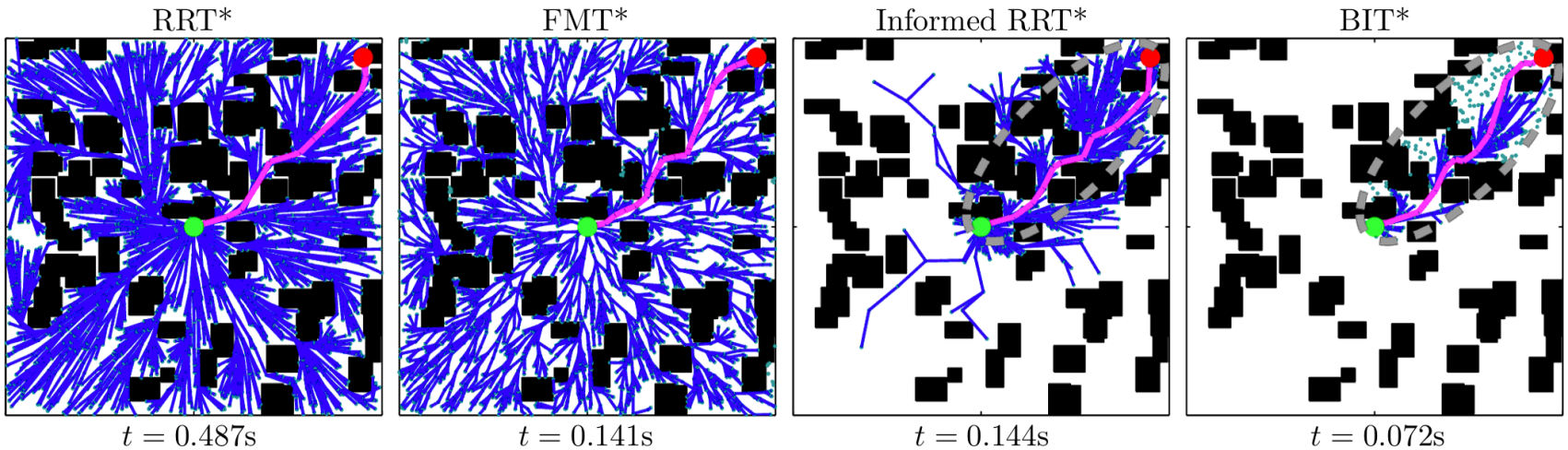
- Samples are added to the tree in batches
- Batch of samples is processed using prioriy queue (using the estimated cost of solution)
- Batches contain multiple samples drawn using Informed-RRT* principle
- Edges are evaluated in lazy manner: and pruned if they do not improve solution
- Only promising edges are checked for collisions



[11] J. D. Gammell et al. "Batch Informed Trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 3067–3074. DOI: 10.1109/ICRA.2015.7139620.

Batch informed Tree (BIT*)[11] — most advanced planner

- Samples are added to the tree in batches
- Batch of samples is processed using prioriy queue (using the estimated cost of solution)
- Batches contain multiple samples drawn using Informed-RRT* principle
- Edges are evaluated in lazy manner: and pruned if they do not improve solution
- Only promising edges are checked for collisions

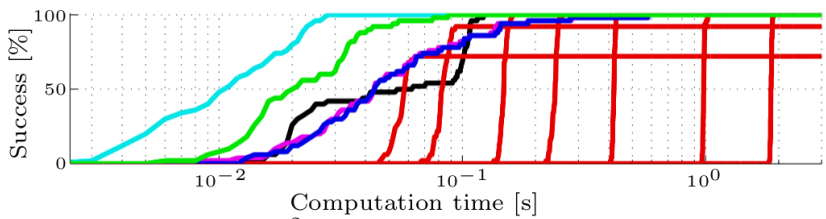


[11] J. D. Gammell et al. "Batch Informed Trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 3067–3074. DOI: 10.1109/ICRA.2015.7139620.

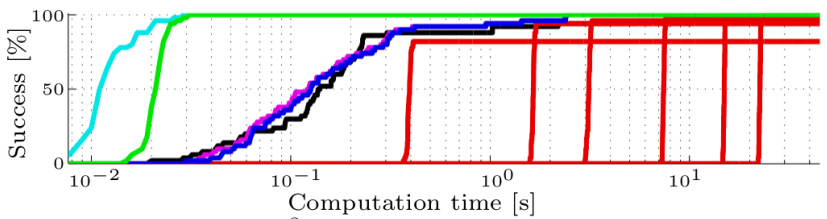
BIT* planner

Batch informed Tree (BIT*)[11] — most advanced planner

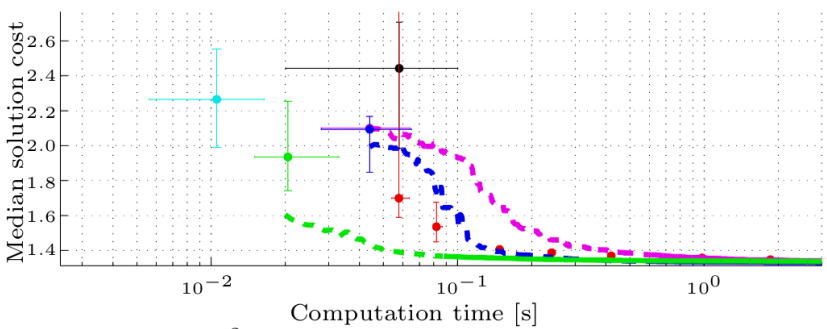
- Samples are added to the tree in batches
- Batch of samples is processed using prioriy queue (using the estimated cost of solution)
- Batches contain multiple samples drawn using Informed-RRT* principle
- Edges are evaluated in lazy manner: and pruned if they do not improve solution
- Only promising edges are checked for collisions



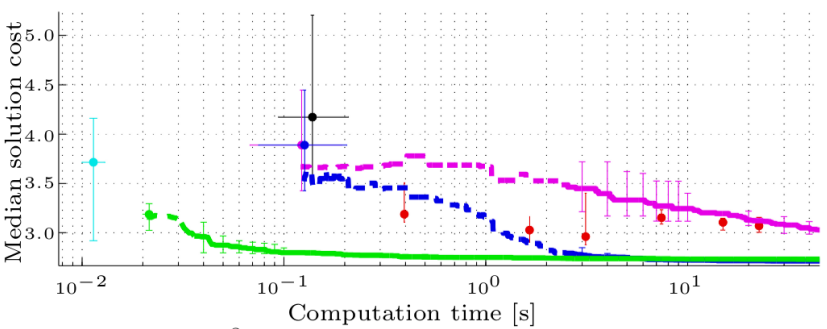
(a) $\mathbb{R}^2$: Runs solved vs. time



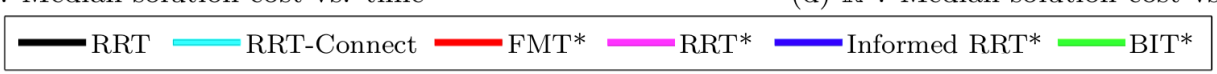
(b) $\mathbb{R}^8$: Runs solved vs. time



(c) $\mathbb{R}^2$: Median solution cost vs. time



(d) $\mathbb{R}^8$: Median solution cost vs. time



BIT* planner