# Cloud RRT* :
# Sampling Cloud based RRT*

Donghyuk Kim, Junghwan Lee, Sung-Eui Yoon
**KAIST**
**(Korea Advanced Institute of Science Technology)**

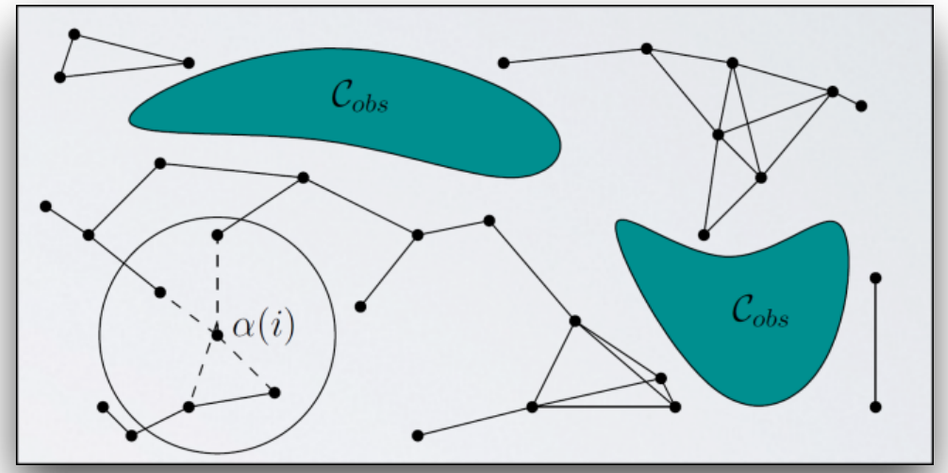
**ICRA2014, HongKong**

*Tuesday, June 3, 2014*

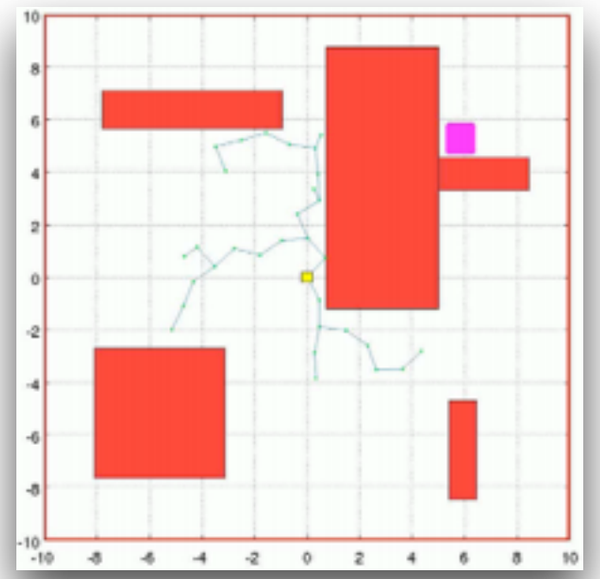
KAIST 한국과학기술원
Korea Advanced Institute of Science and Technology

# Sampling-based Motion Planning

- Probabilistically complete
- Scalable to high dimensions
- RRT(Rapidly-exploring Random Tree), [LaValle & Kufner, IJRR 2001]
- PRM(Probabilistic Roadmap Method) [Kavraki et al., IEEE T. Robotics and Automation 1996]



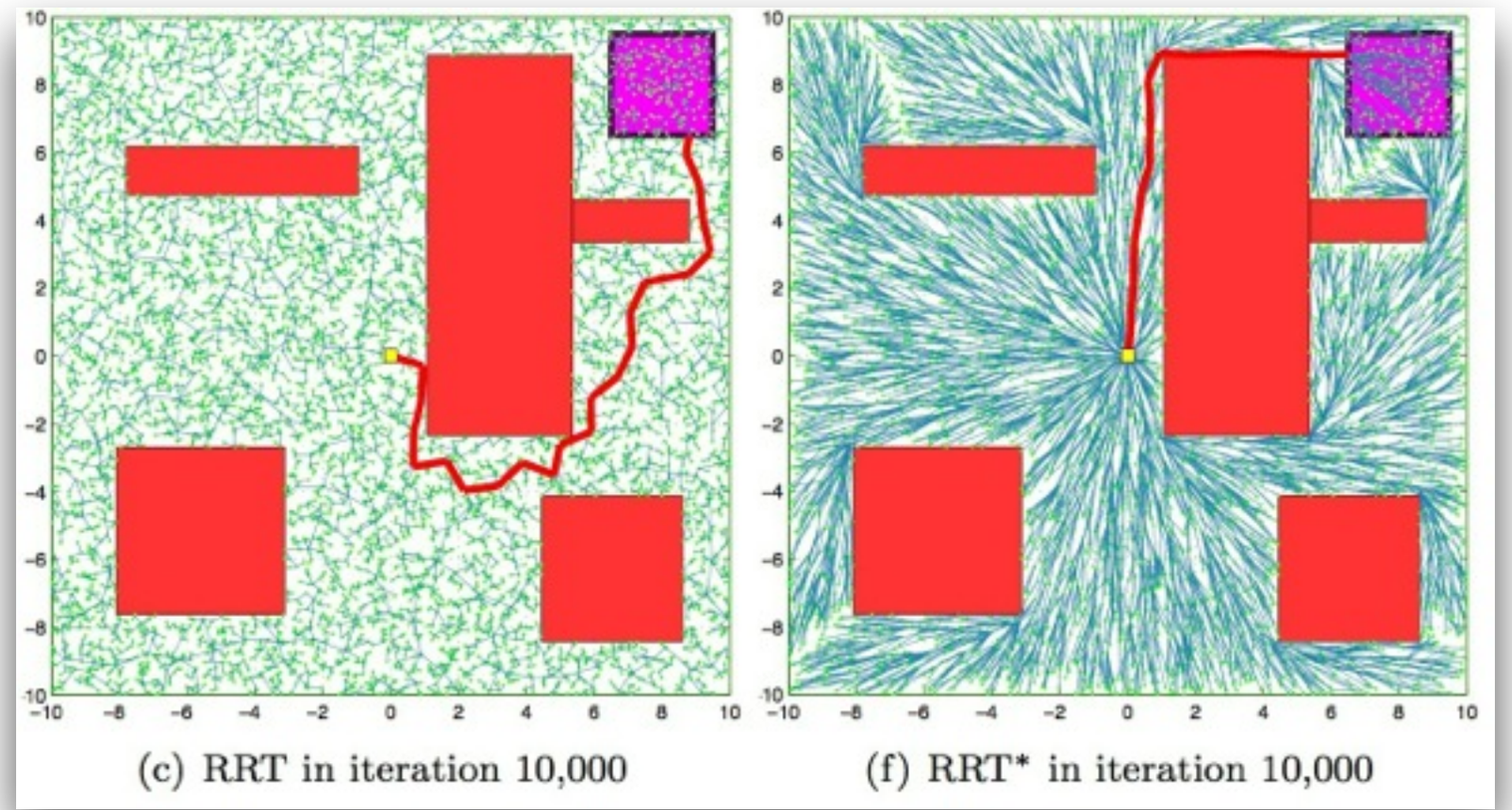

*LaValle, Planning Algorithms 2006*



*Karaman et al.,*
*Anytime Motion Planning using RRT*, 2011*

# Optimal Motion Planning

- RRT* : Asymptotically optimal version of RRT
- No substantial, computing overhead compared to RRT

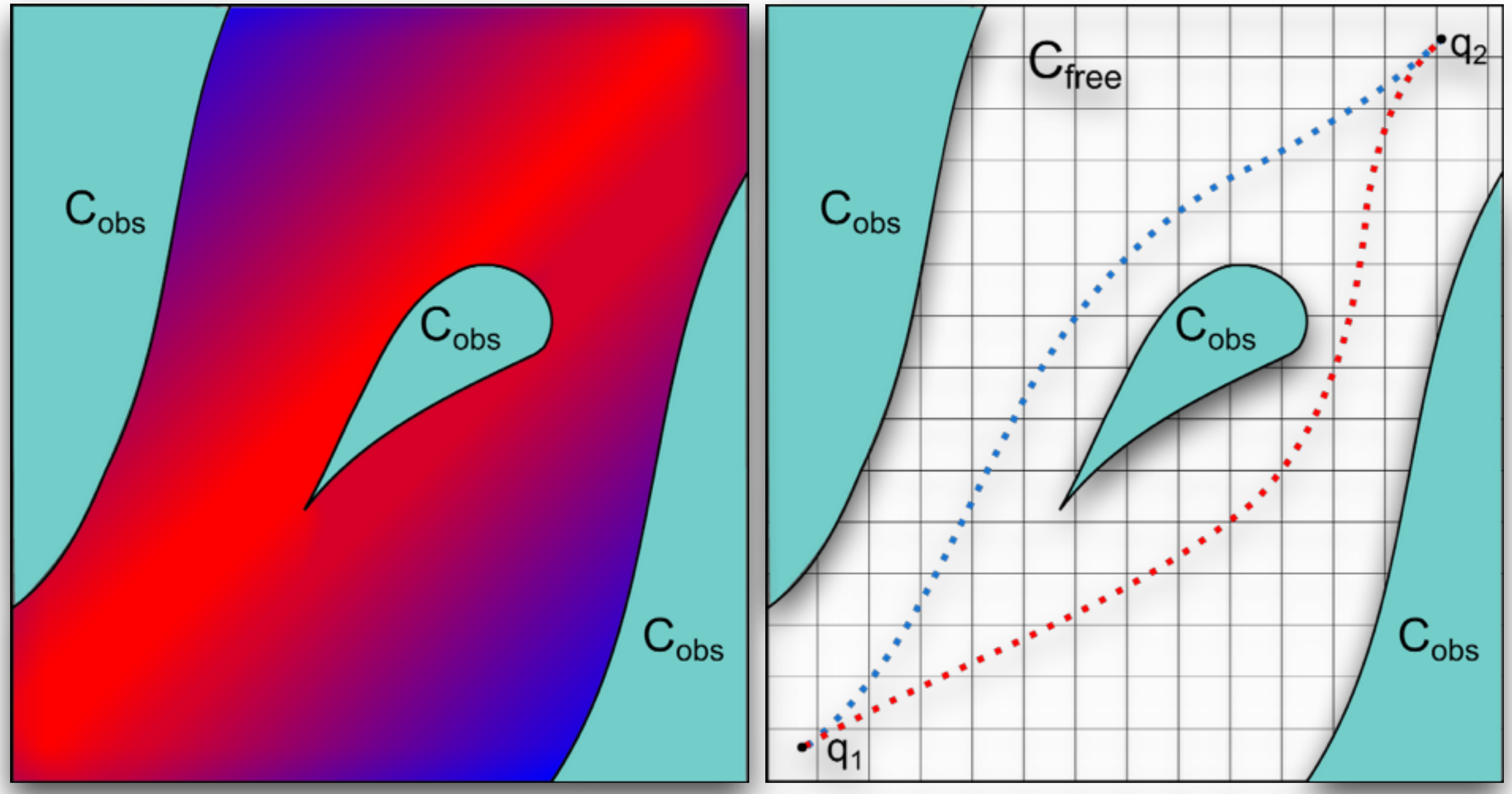



*(Karaman and Frazzoli, "Incremental Sampling-based Algorithms for Optimal Motion Planning", IJRR 2011)*

# Exploration vs Exploitation

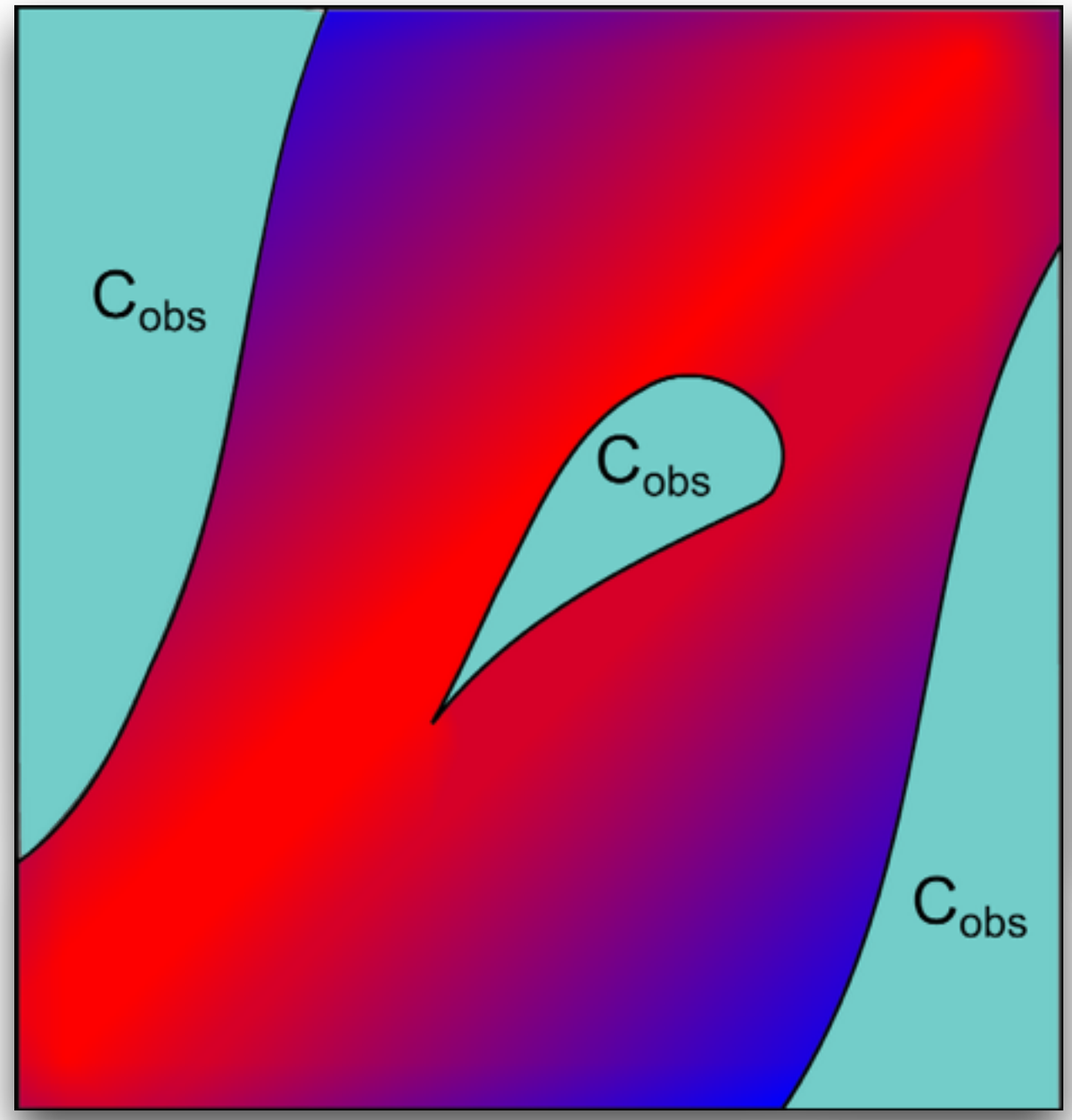
- **Global Search** vs **Local Search**
  - Refining local solutions (**Exploitation**)
  - Finding another solution (**Exploration**)
- Trade-off between two strategies

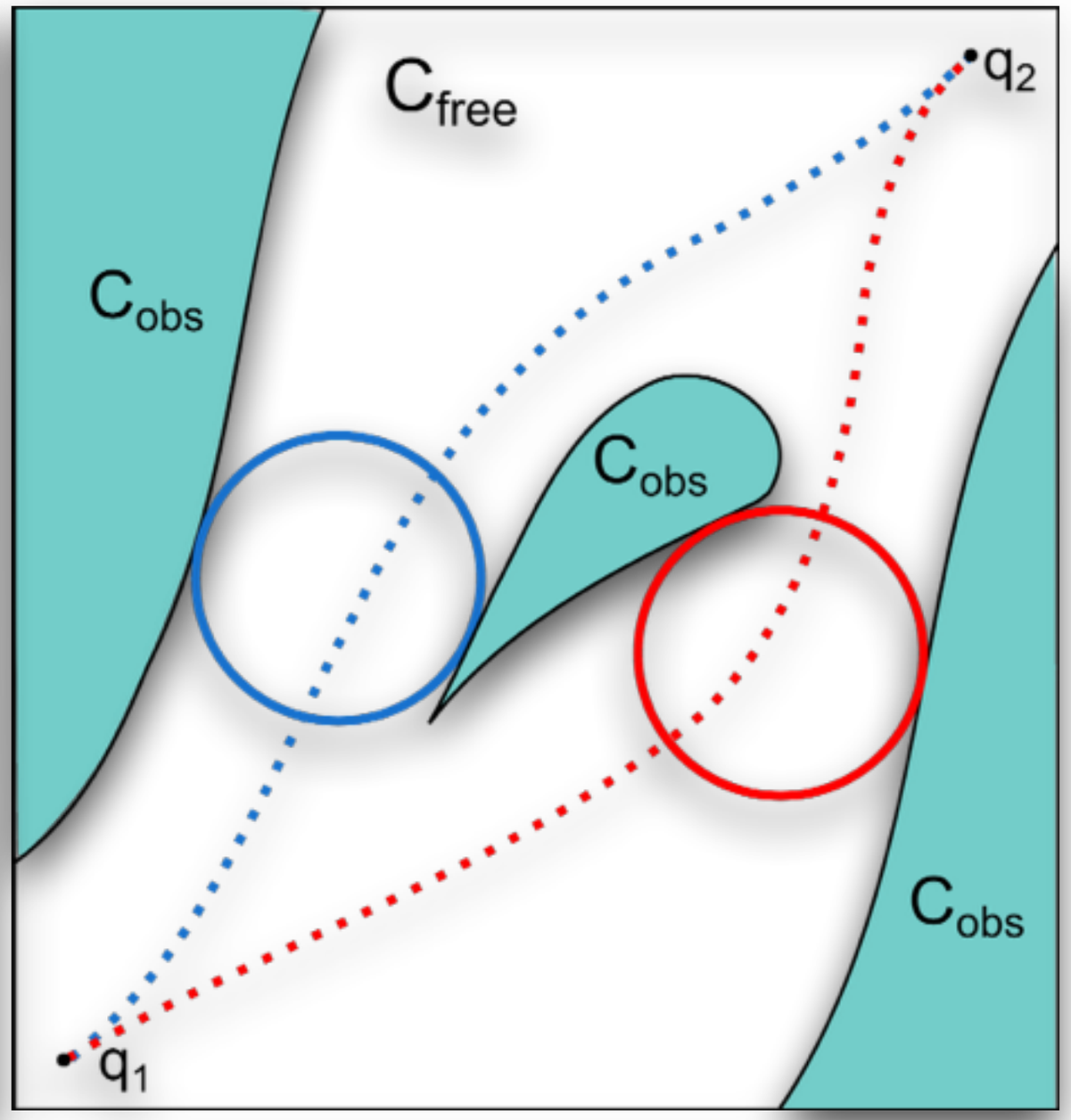
# Exploration vs Exploitation




- **Difficult choice for finding the global optimum**

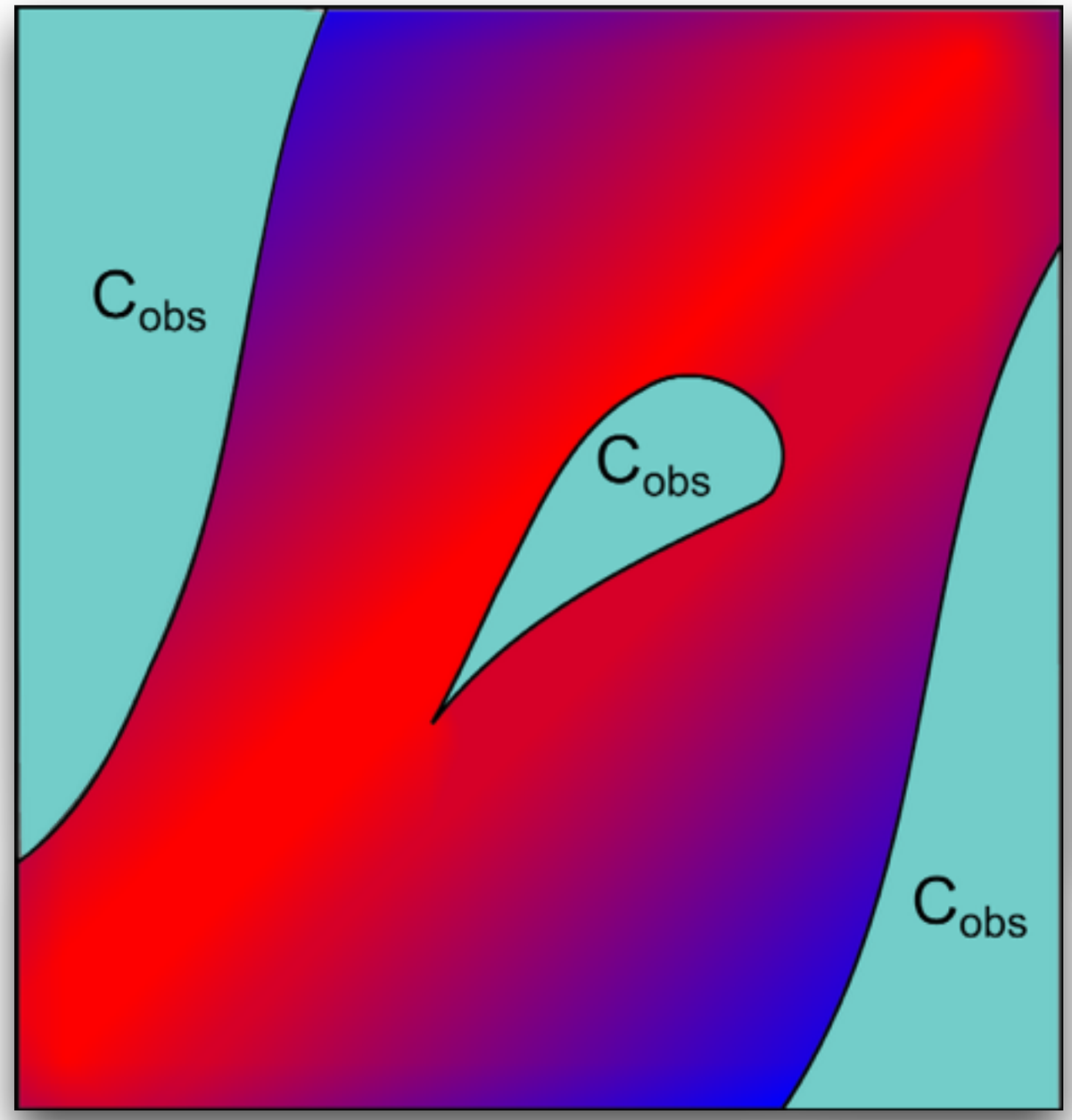
# Exploration vs Exploitation

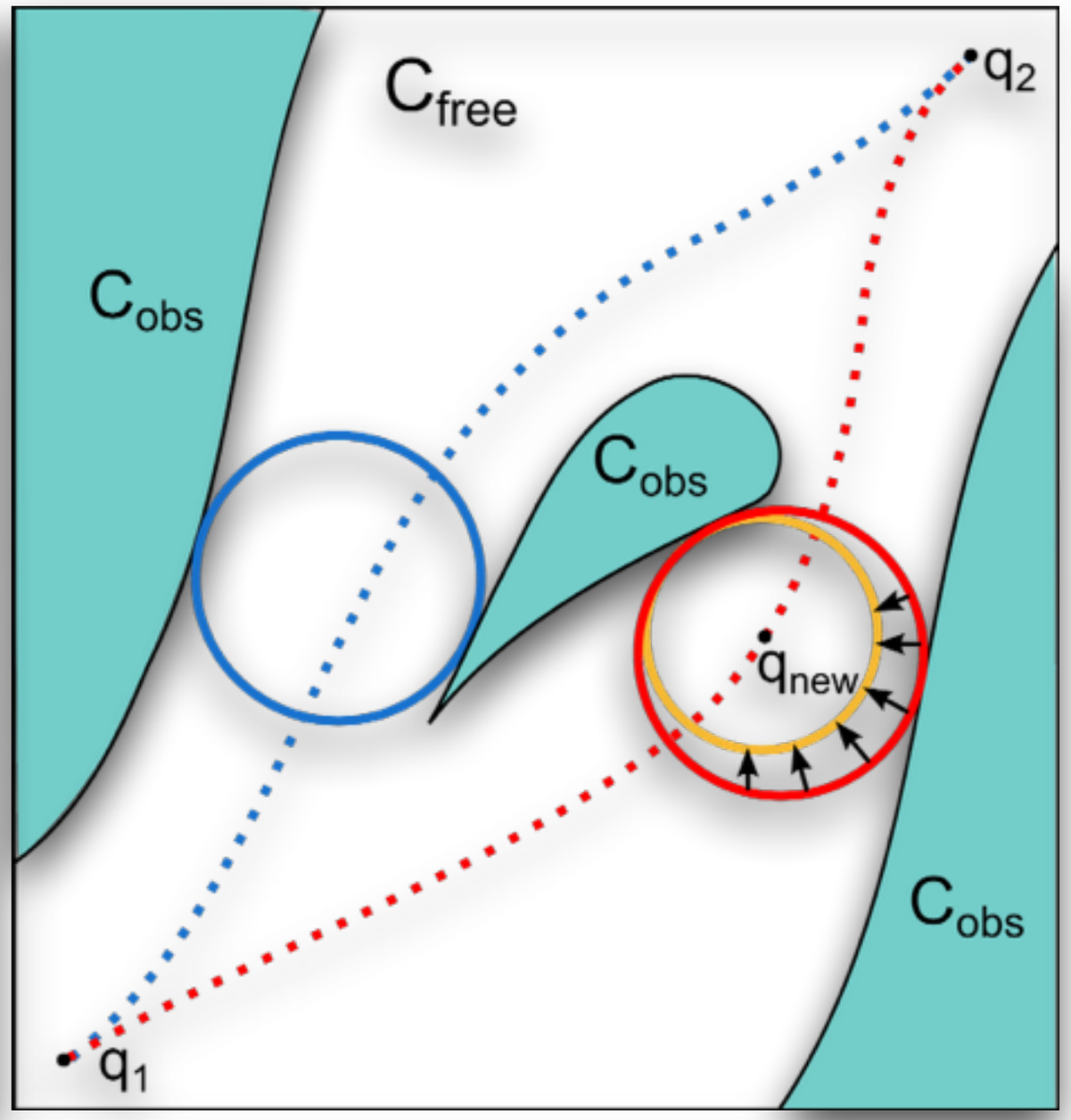



- **Identify all the homotopy classes**

# Exploration vs Exploitation




- **Exploit locally successful sampling regions, while exploring different homotopy classes**

# Our Goal

- Achieve a rapid convergence speed toward the optimal solution

- Balance exploration and exploitation

- Exploit locally successful sampling regions, while exploring different homotopy classes

# Relevant Works

- **RRT(Rapidly-exploring Random Tree) (S. M. LaValle, 1998)**
- **RRT* (S. Karaman, IJRR'2011)**

- **Balancing exploration & exploitation (M. Rickert, ICRA2008)**
- **Ball-tree : Free-space approximation method (A Shkolnik, 2011)**
- **Decomposition-based motion planning : Wavefront Expansion (O. Brock, ICRA2001)**

- **Sampling Heuristics for rapid convergence toward optimal**
  - **Local Biasing & Node Rejection (B. Akgun, IROS2011)**
  - **RRT*-Smart (F. Islam, ICMA2012)**
  - Refine the current best solution
  - Mainly consider the exploitation rather than exploration

# Relevant Works

- **RRT(Rapidly-exploring Random Tree) (S. M. LaValle, 1998)**
- **RRT* (S. Karaman, IJRR'2011)**

- **Balancing exploration & exploitation (M. Rickert, ICRA2008)**
- **Ball-tree : Free-space approximation method (A Shkolnik, 2011)**
- **Decomposition-based motion planning : Wavefront Expansion (O. Brock, ICRA2001)**

- **Sampling Heuristics for rapid convergence toward optimal**
  - **Local Biasing & Node Rejection (B. Akgun, IROS2011)**
  - **RRT*-Smart (F. Islam, ICMA2012)**
  - Refine the current best solution
  - Mainly consider the exploitation rather than exploration

# Relevant Works

- **RRT(Rapidly-exploring Random Tree) (S. M. LaValle, 1998)**
- **RRT* (S. Karaman, IJRR'2011)**
- **Balancing exploration & exploitation (M. Rickert, ICRA2008)**
- **Ball-tree : Free-space approximation method (A Shkolnik, 2011)**
- **Decomposition-based motion planning : Wavefront Expansion (O. Brock, ICRA2001)**

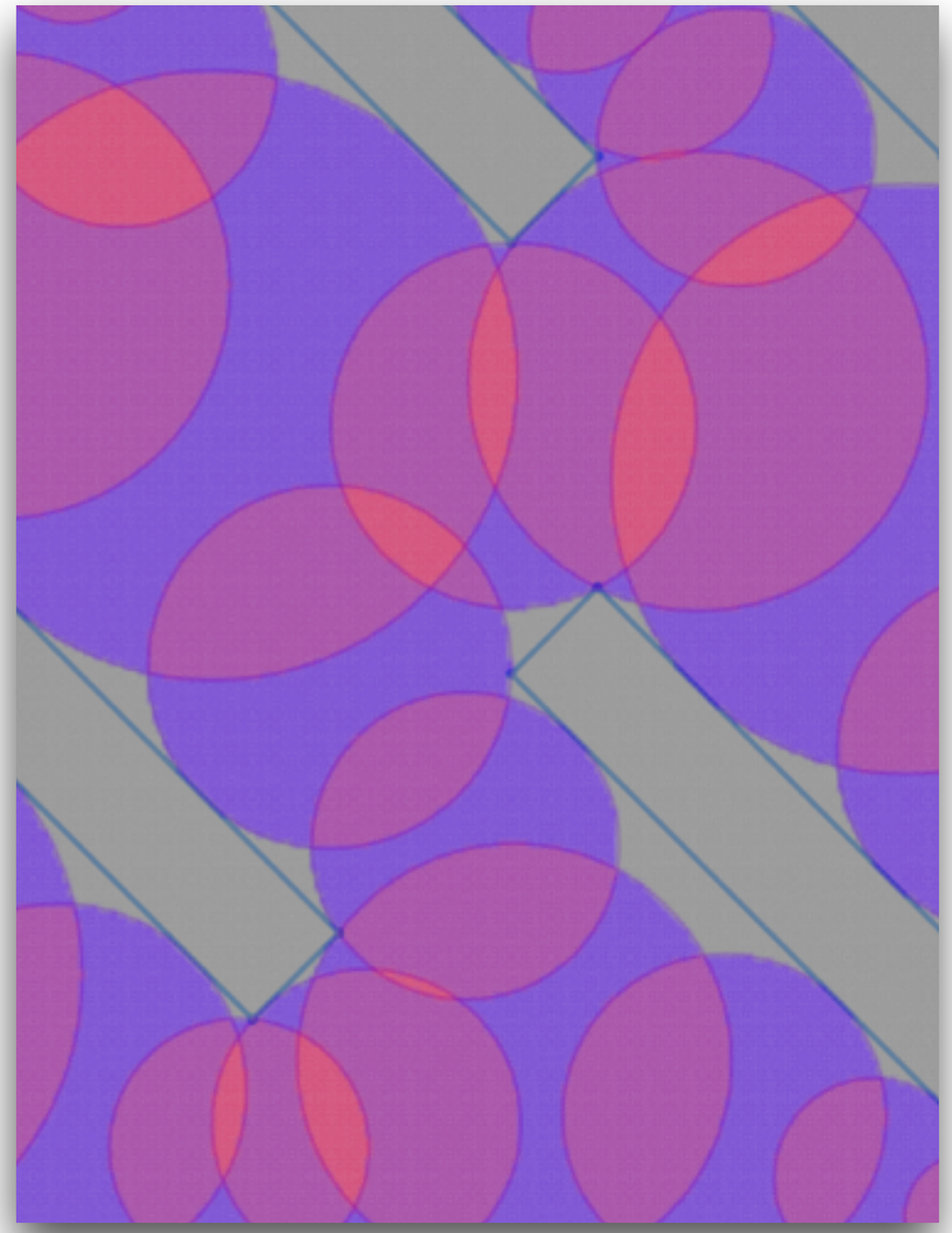
- **Sampling Heuristics for rapid convergence toward optimal**
  - **Local Biasing & Node Rejection (B. Akgun, IROS2011)**
  - **RRT*-Smart (F. Islam, ICMA2012)**
  - Refine the current best solution
  - Mainly consider the exploitation rather than exploration

# Algorithm Overview

- Sampling cloud: sampling space decomposition
- Sampling & update rules for sampling cloud
- Cloud RRT*: sampling cloud integrated with the original RRT*

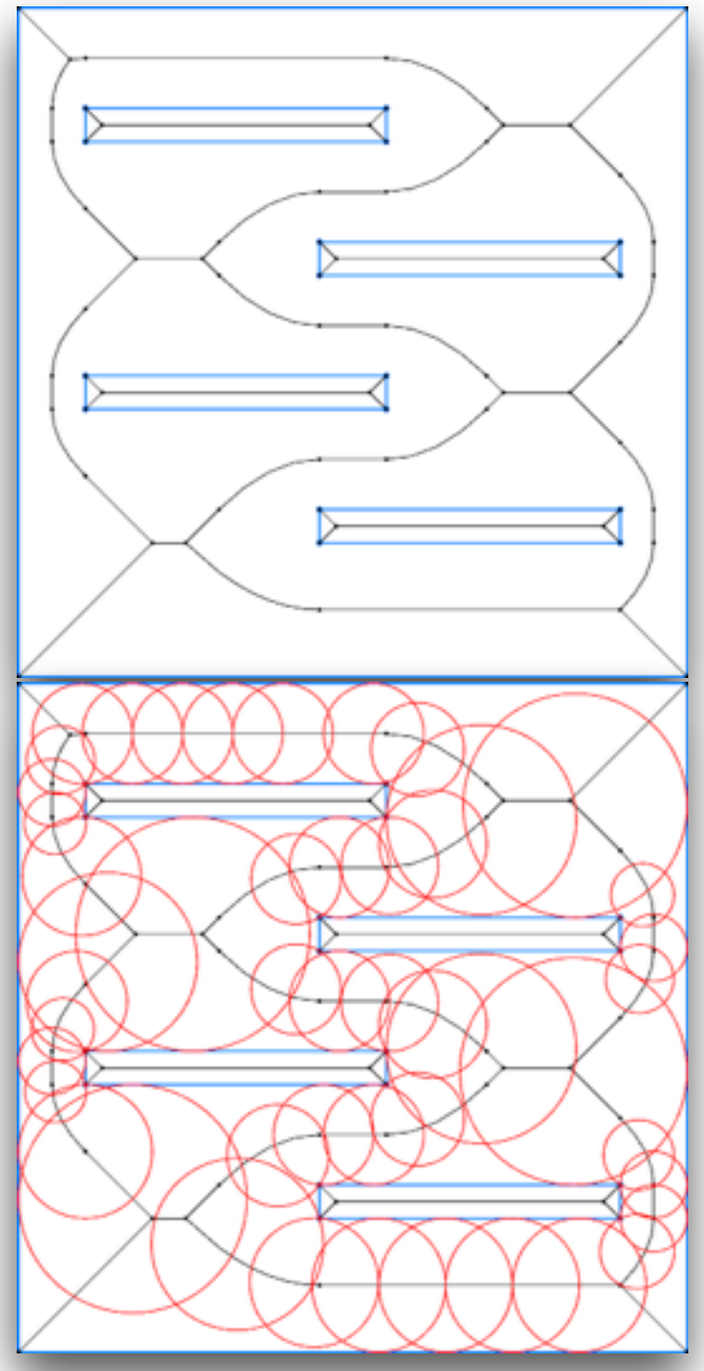
# Sampling Cloud

- Sampling cloud is a set of spheres, which represents a subset of C-space
- A sampling sphere in sampling cloud has:
  - **Center position**
  - **Radius**
  - **Orientation range** $[\phi - \theta, \phi + \theta]$
  - **Importance value**
    - A probability to be sampled. Overlapped region or one with high importance value is likely to be more frequently sampled



# GVG-guided initialization

- Based on GVG (Generalized Voronoi Graph)
  - Can cover all of the possible homotopy classes in 2D
    - Each component has the maximum clearance to the nearest obstacle



- Initialize the sampling cloud by sphere expansion (O. Brock, ICRA2001)

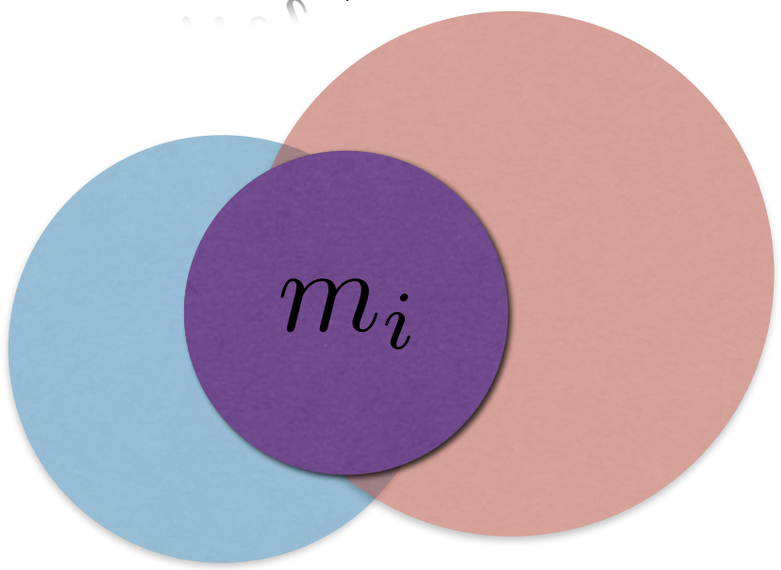
# Sampling Cloud Update

- Milestone
  - A set of configurations from the current best solution over all the prior paths
- For each configuration of the milestone $m_i$ , find all sampling spheres containing $m_i$


*Two sampling spheres(blue, red) & a configuration in milestone $m_i$*

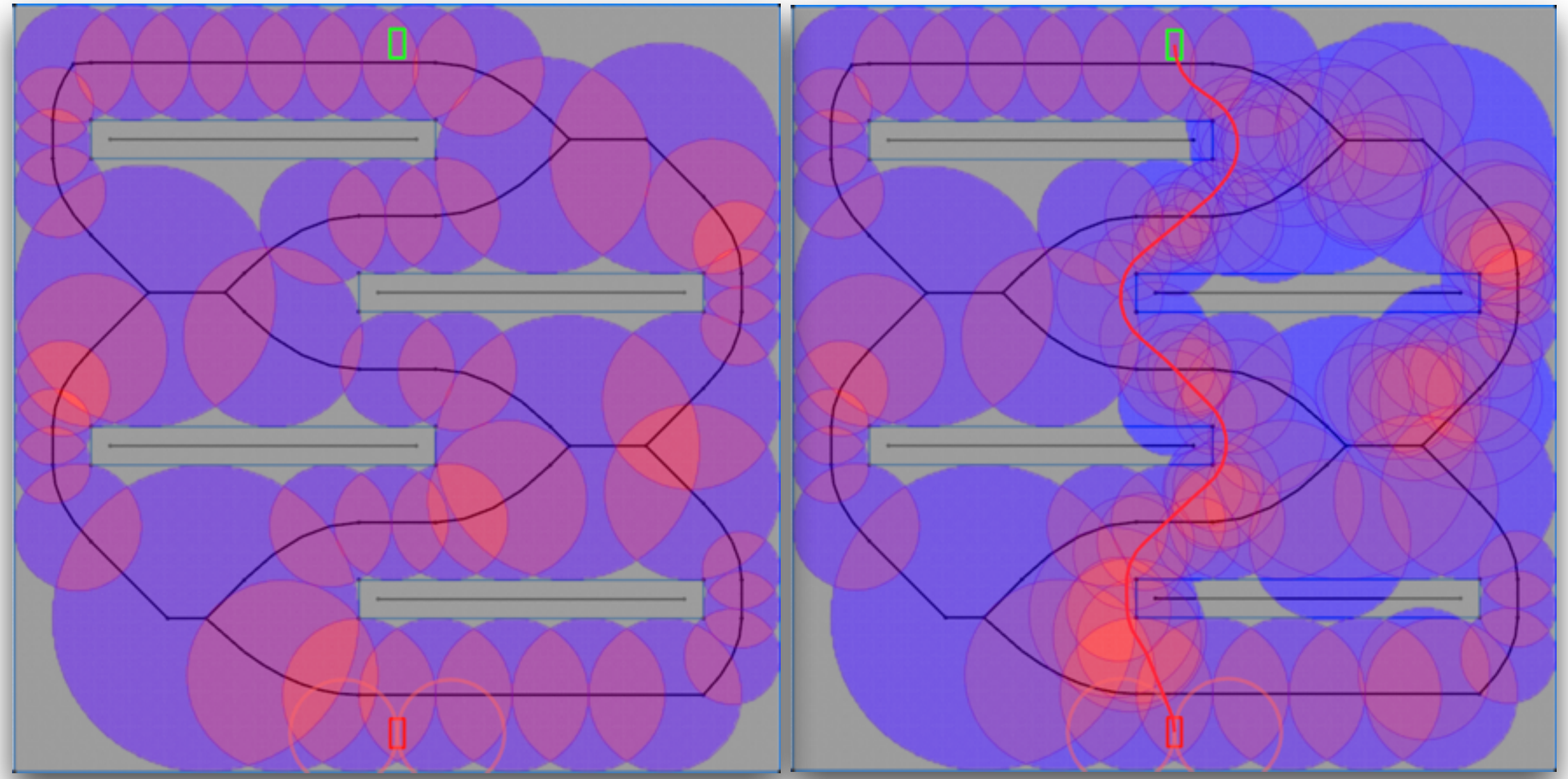
# Sampling Cloud Update

- Milestone
  - A set of configurations from the current best solution over all the prior paths
- For each configuration of the milestone $m_i$ , find all sampling spheres containing $m_i$
- Generate a new sampling sphere centered at $m_i$




*Newly generated sampling sphere(purple) centered at $m_i$*

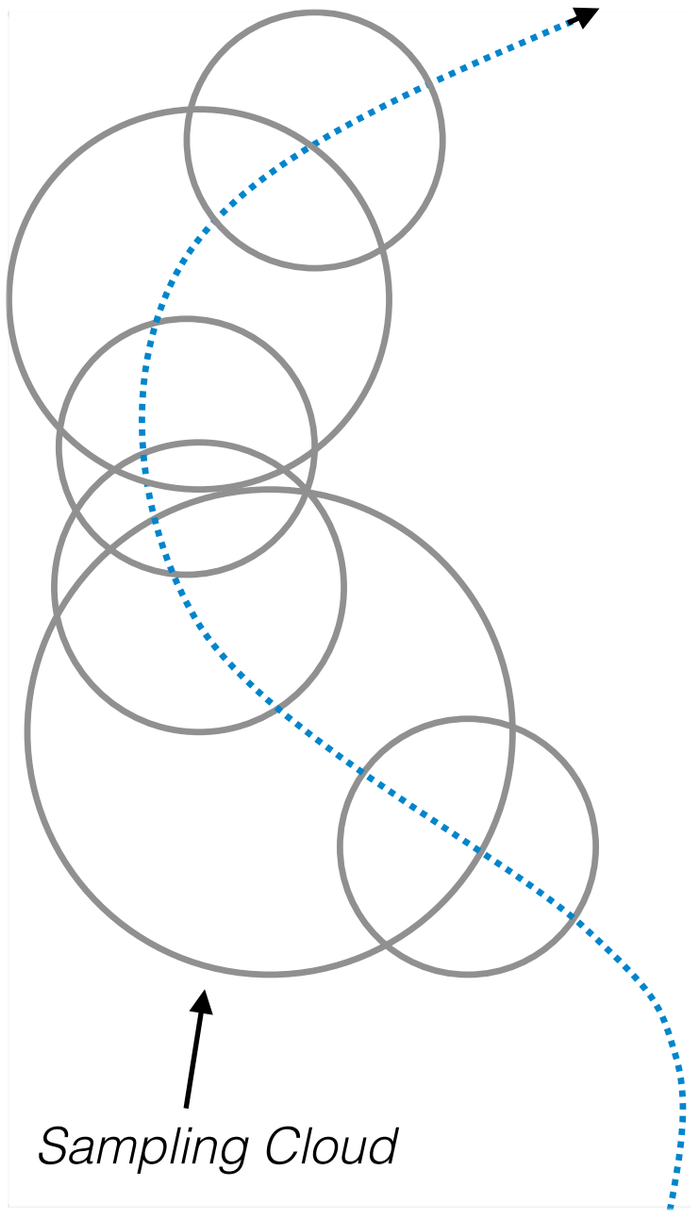
# Examples of Sampling Cloud



Initial state of sampling cloud

After updated several times

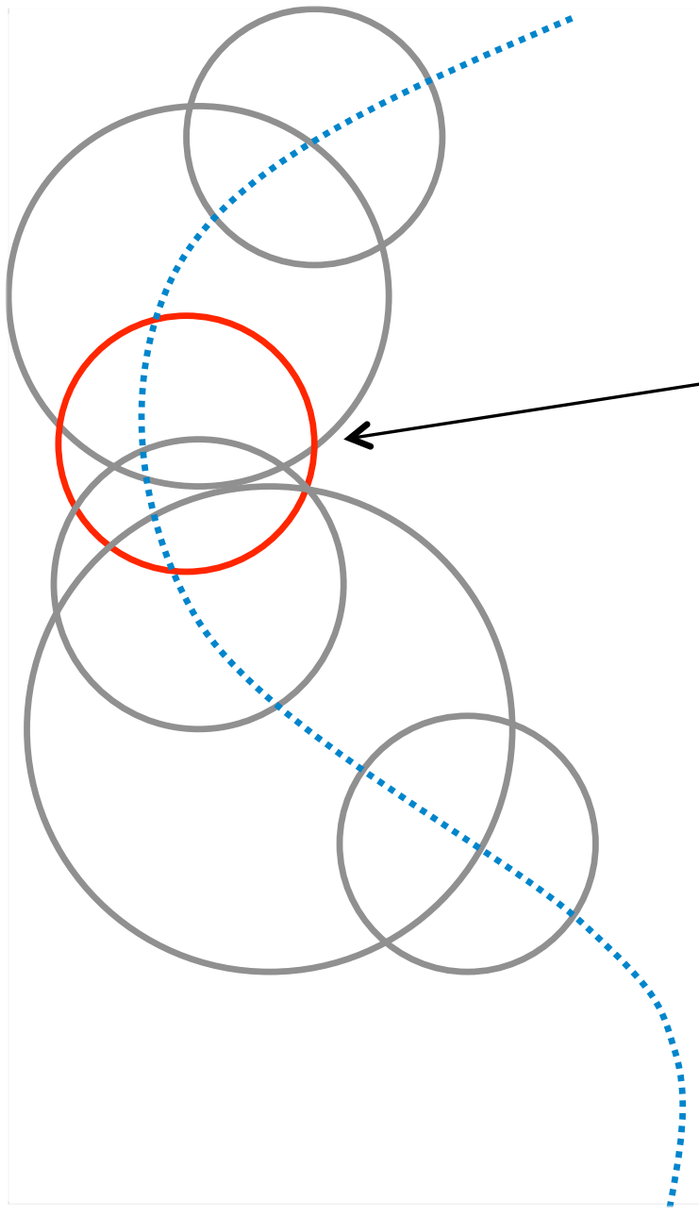
# Cloud RRT*




**Input**: A sampling cloud $\mathscr{S}$, an init. configuration, $q_{init}$, and a goal region, $Q_{goal}$
**Output**: A random tree, $\mathscr{T}$, and a solution path, $Q_{sol}$

$\mathscr{T} \leftarrow \{q_{init}\}$ and $Q_{sol} \leftarrow \emptyset$
**while** ***not*** *termination conditions are satisfied* **do**
  $s \leftarrow SampleSphere(\mathscr{S})$
  $q_s \leftarrow Sample(s)$
  $q_n \leftarrow Nearest(q_s)$
  $(q_{new}, u_{new}) \leftarrow Steer(q_n, q_s)$
  **if** $IsCollisionFree(q_{new}, u_{new}, q_n)$ **then**
    $Q_{near} \leftarrow Near(q_{new})$
    $q_{min} \leftarrow ChooseParent(Q_{near}, q_n, q_{new})$
    $\mathscr{T} \leftarrow InsertNode(q_{min}, q_{new}, \mathscr{T}, s)$
    $\mathscr{T} \leftarrow Rewire(\mathscr{T}, Q_{near}, q_{min}, q_{new})$
    **if** *A better solution is found* **then**
      $Q_{sol} \leftarrow UpdateSolution(\mathscr{T})$
      $M \leftarrow UpdateMilestone(Q_{sol})$
      $\mathscr{S} \leftarrow UpdateSamplingCloud(\mathscr{S}, M)$
**return** $\mathscr{T}$

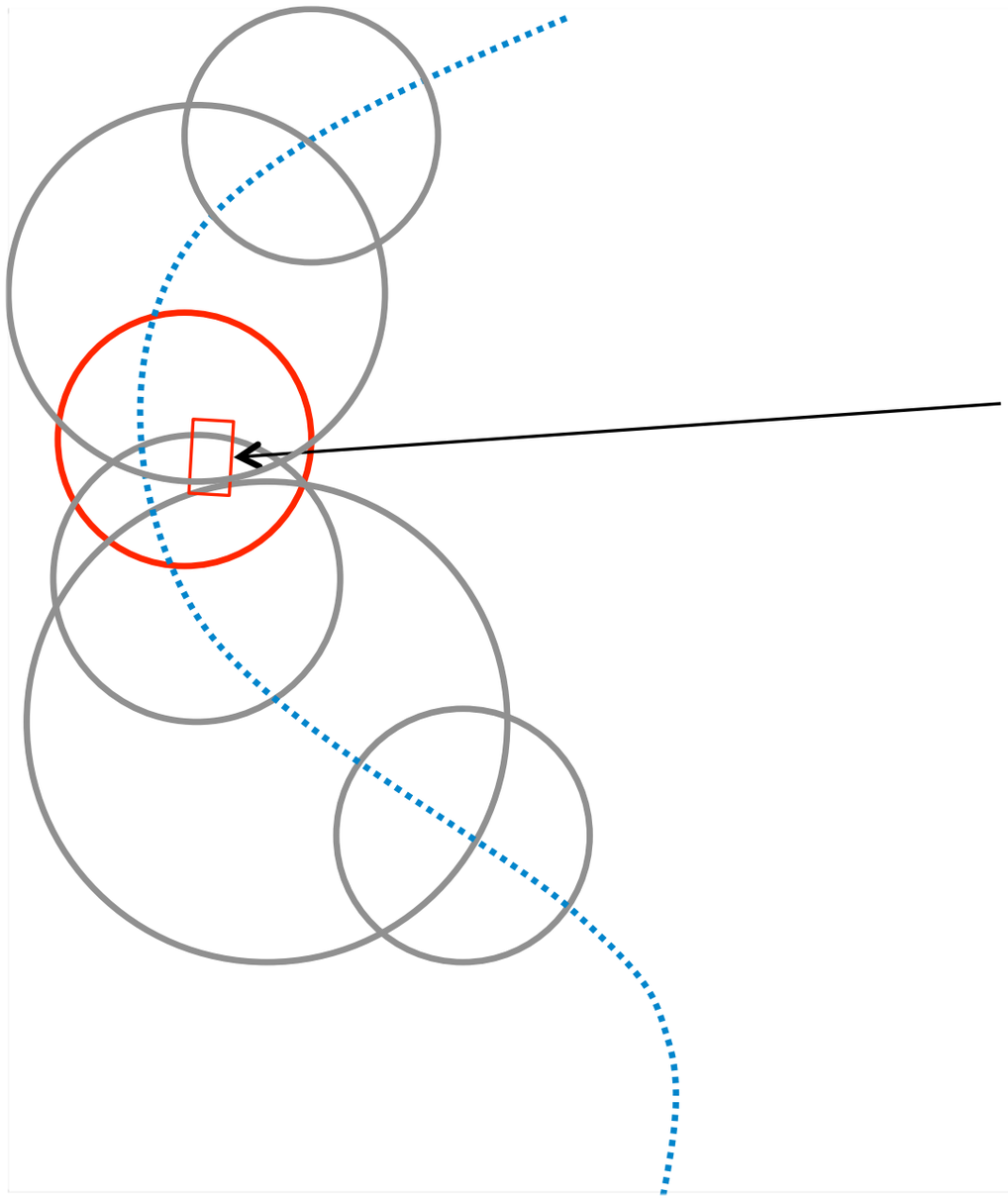
# Cloud RRT*



**Input**: A sampling cloud $\mathscr{S}$, an init. configuration, $q_{init}$, and a goal region, $Q_{goal}$
**Output**: A random tree, $\mathscr{T}$, and a solution path, $Q_{sol}$

$\mathscr{T} \leftarrow \{q_{init}\}$ and $Q_{sol} \leftarrow \emptyset$
**while** ***not*** *termination conditions are satisfied* **do**
  s$\leftarrow SampleSphere(\mathscr{S})$
  $q_s \leftarrow Sample(\mathrm{s})$
  $q_n \leftarrow Nearest(q_s)$
  $(q_{new}, u_{new}) \leftarrow Steer(q_n, q_s)$
  **if** $IsCollisionFree(q_{new}, u_{new}, q_n)$ **then**
    $Q_{near} \leftarrow Near(q_{new})$
    $q_{min} \leftarrow ChooseParent(Q_{near}, q_n, q_{new})$
    $\mathscr{T} \leftarrow InsertNode(q_{min}, q_{new}, \mathscr{T}, s)$
    $\mathscr{T} \leftarrow Rewire(\mathscr{T}, Q_{near}, q_{min}, q_{new})$
    **if** *A better solution is found* **then**
      $Q_{sol} \leftarrow UpdateSolution(\mathscr{T})$
      $M \leftarrow UpdateMilestone(Q_{sol})$
      $\mathscr{S} \leftarrow UpdateSamplingCloud(\mathscr{S}, M)$
**return** $\mathscr{T}$

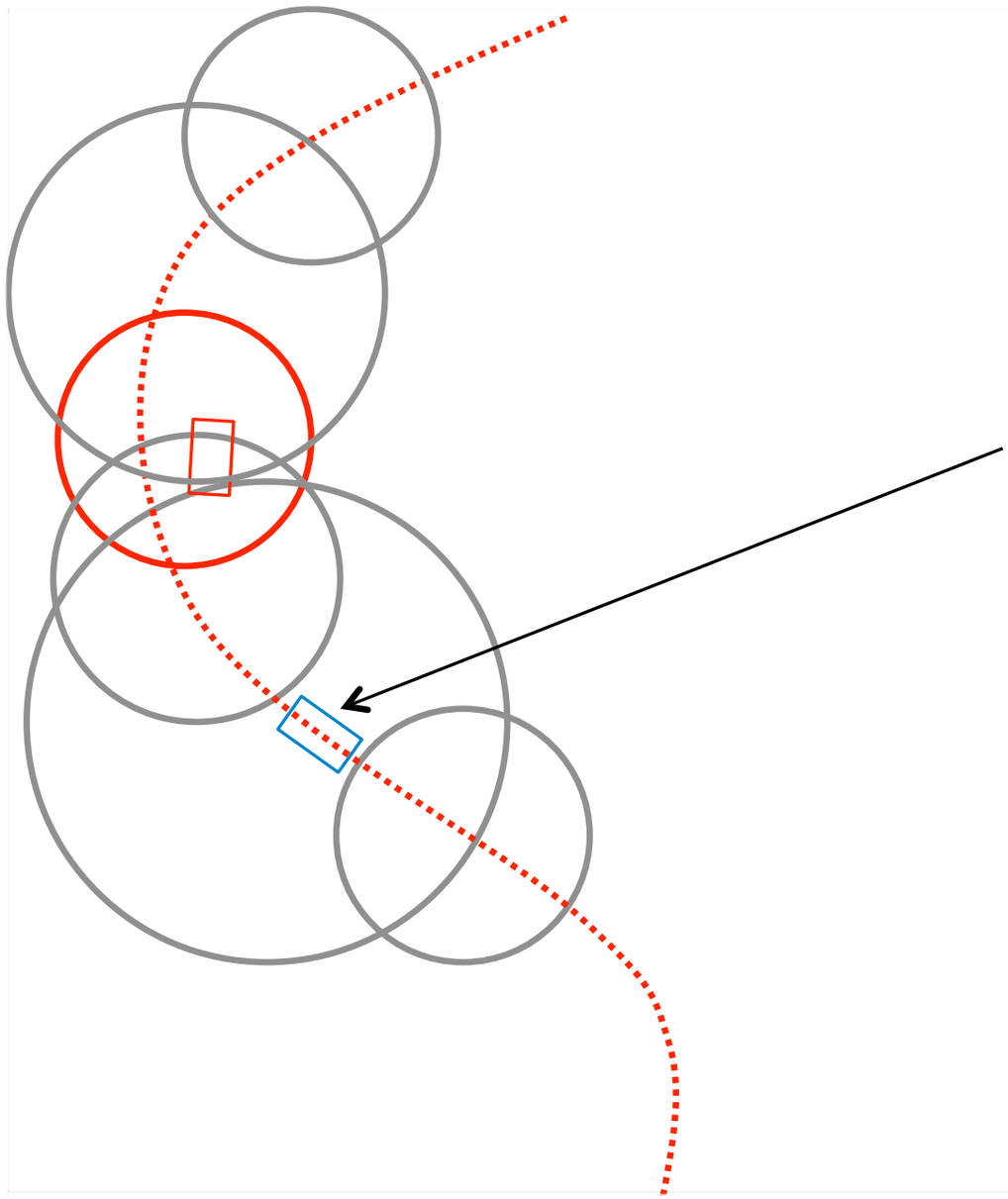
# Cloud RRT*



**Input**: A sampling cloud $\mathscr{S}$, an init. configuration, $q_{init}$, and a goal region, $Q_{goal}$
**Output**: A random tree, $\mathscr{T}$, and a solution path, $Q_{sol}$

$\mathscr{T} \leftarrow \{q_{init}\}$ and $Q_{sol} \leftarrow \emptyset$
**while** ***not*** *termination conditions are satisfied* **do**
  s$\leftarrow SampleSphere(\mathscr{S})$
  $q_s \leftarrow Sample(\mathrm{s})$
  $q_n \leftarrow Nearest(q_s)$
  $(q_{new}, u_{new}) \leftarrow Steer(q_n, q_s)$
  **if** $IsCollisionFree(q_{new}, u_{new}, q_n)$ **then**
    $Q_{near} \leftarrow Near(q_{new})$
    $q_{min} \leftarrow ChooseParent(Q_{near}, q_n, q_{new})$
    $\mathscr{T} \leftarrow InsertNode(q_{min}, q_{new}, \mathscr{T}, s)$
    $\mathscr{T} \leftarrow Rewire(\mathscr{T}, Q_{near}, q_{min}, q_{new})$
    **if** *A better solution is found* **then**
      $Q_{sol} \leftarrow UpdateSolution(\mathscr{T})$
      $M \leftarrow UpdateMilestone(Q_{sol})$
      $\mathscr{S} \leftarrow UpdateSamplingCloud(\mathscr{S}, M)$
**return** $\mathscr{T}$

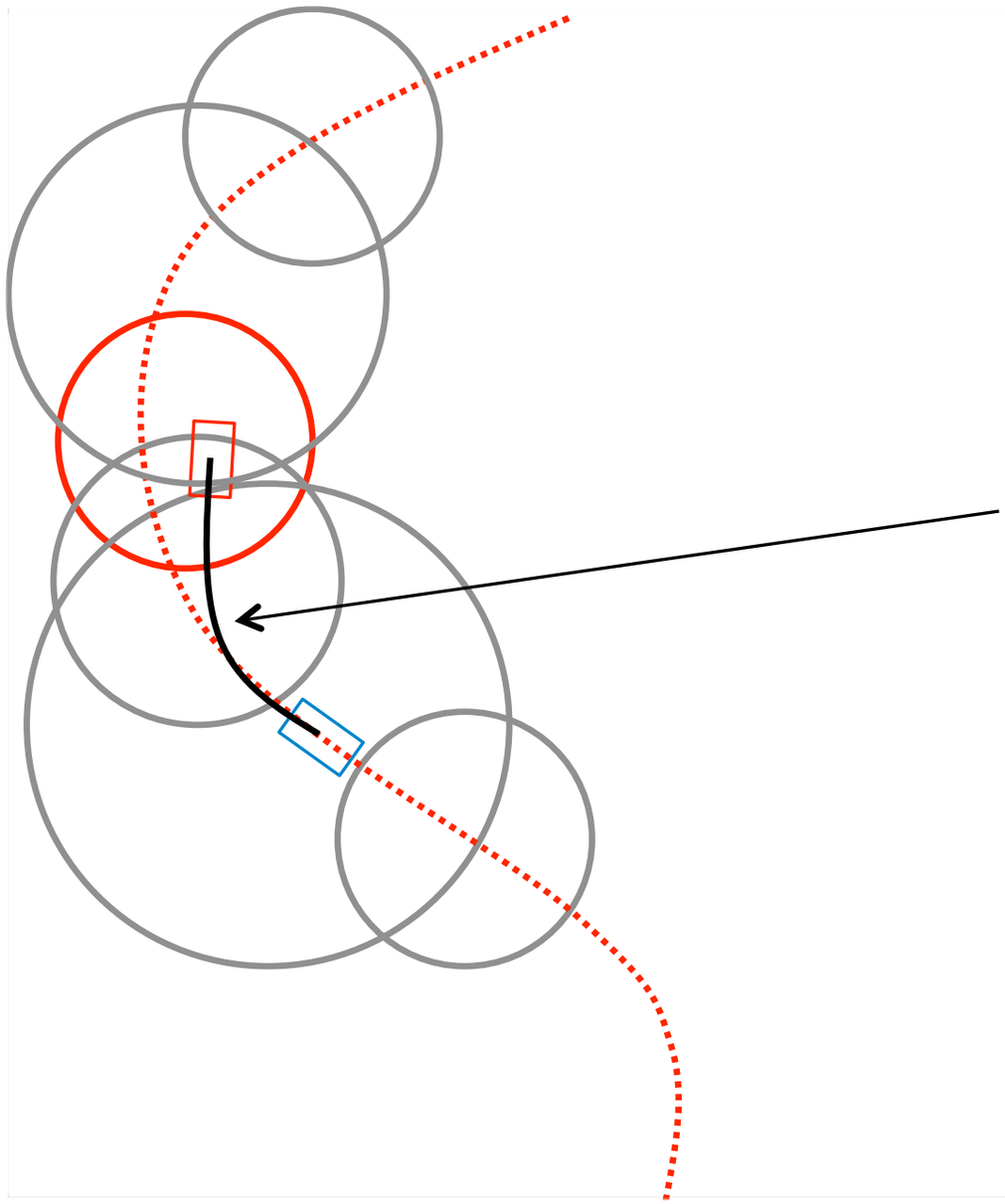
# Cloud RRT*



**Input**: A sampling cloud $\mathscr{S}$, an init. configuration, $q_{init}$, and a goal region, $Q_{goal}$
**Output**: A random tree, $\mathscr{T}$, and a solution path, $Q_{sol}$

$\mathscr{T} \leftarrow \{q_{init}\}$ and $Q_{sol} \leftarrow \emptyset$
**while** ***not*** *termination conditions are satisfied* **do**
  $s \leftarrow SampleSphere(\mathscr{S})$
  $q_s \leftarrow Sample(s)$
  $q_n \leftarrow Nearest(q_s)$
  $(q_{new}, u_{new}) \leftarrow Steer(q_n, q_s)$
  **if** $IsCollisionFree(q_{new}, u_{new}, q_n)$ **then**
    $Q_{near} \leftarrow Near(q_{new})$
    $q_{min} \leftarrow ChooseParent(Q_{near}, q_n, q_{new})$
    $\mathscr{T} \leftarrow InsertNode(q_{min}, q_{new}, \mathscr{T}, s)$
    $\mathscr{T} \leftarrow Rewire(\mathscr{T}, Q_{near}, q_{min}, q_{new})$
    **if** *A better solution is found* **then**
      $Q_{sol} \leftarrow UpdateSolution(\mathscr{T})$
      $M \leftarrow UpdateMilestone(Q_{sol})$
      $\mathscr{S} \leftarrow UpdateSamplingCloud(\mathscr{S}, M)$
**return** $\mathscr{T}$

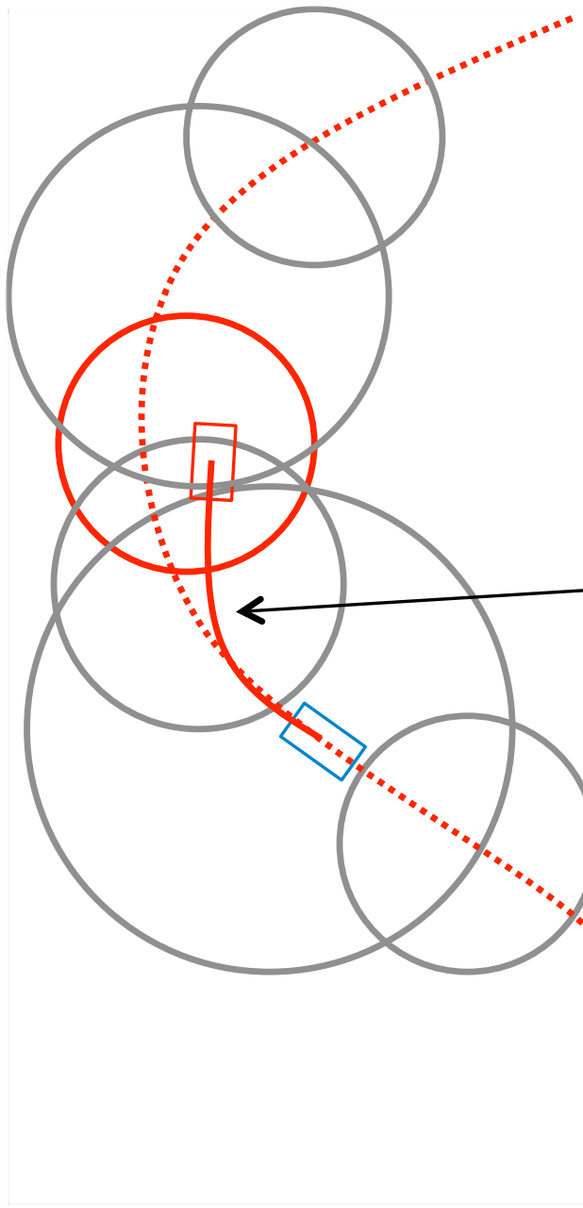
# Cloud RRT*



**Input**: A sampling cloud $\mathscr{S}$, an init. configuration, $q_{init}$, and a goal region, $Q_{goal}$
**Output**: A random tree, $\mathscr{T}$, and a solution path, $Q_{sol}$

$\mathscr{T} \leftarrow \{q_{init}\}$ and $Q_{sol} \leftarrow \emptyset$
**while** ***not*** *termination conditions are satisfied* **do**
  $s \leftarrow SampleSphere(\mathscr{S})$
  $q_s \leftarrow Sample(s)$
  $q_n \leftarrow Nearest(q_s)$
  $(q_{new}, u_{new}) \leftarrow Steer(q_n, q_s)$
  **if** $IsCollisionFree(q_{new}, u_{new}, q_n)$ **then**
    $Q_{near} \leftarrow Near(q_{new})$
    $q_{min} \leftarrow ChooseParent(Q_{near}, q_n, q_{new})$
    $\mathscr{T} \leftarrow InsertNode(q_{min}, q_{new}, \mathscr{T}, s)$
    $\mathscr{T} \leftarrow Rewire(\mathscr{T}, Q_{near}, q_{min}, q_{new})$
    **if** *A better solution is found* **then**
      $Q_{sol} \leftarrow UpdateSolution(\mathscr{T})$
      $M \leftarrow UpdateMilestone(Q_{sol})$
      $\mathscr{S} \leftarrow UpdateSamplingCloud(\mathscr{S}, M)$
**return** $\mathscr{T}$

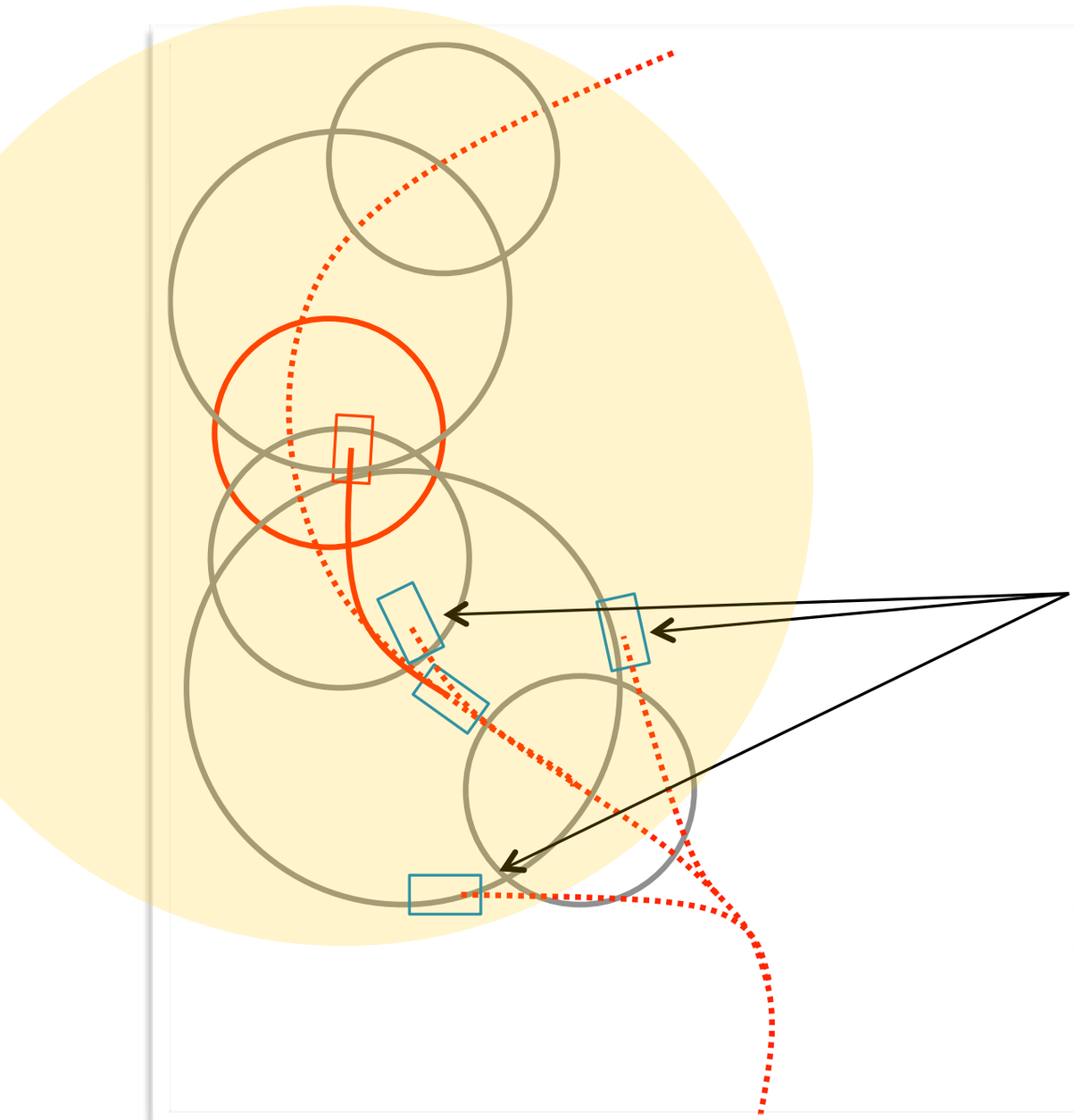
# Cloud RRT*



**Input**: A sampling cloud $\mathscr{S}$, an init. configuration, $q_{init}$, and a goal region, $Q_{goal}$
**Output**: A random tree, $\mathscr{T}$, and a solution path, $Q_{sol}$

$\mathscr{T} \leftarrow \{q_{init}\}$ and $Q_{sol} \leftarrow \emptyset$
**while** ***not*** *termination conditions are satisfied* **do**
  $s \leftarrow SampleSphere(\mathscr{S})$
  $q_s \leftarrow Sample(s)$
  $q_n \leftarrow Nearest(q_s)$
  $(q_{new}, u_{new}) \leftarrow Steer(q_n, q_s)$
  **if** $IsCollisionFree(q_{new}, u_{new}, q_n)$ **then**
    $Q_{near} \leftarrow Near(q_{new})$
    $q_{min} \leftarrow ChooseParent(Q_{near}, q_n, q_{new})$
    $\mathscr{T} \leftarrow InsertNode(q_{min}, q_{new}, \mathscr{T}, s)$
    $\mathscr{T} \leftarrow Rewire(\mathscr{T}, Q_{near}, q_{min}, q_{new})$
    **if** *A better solution is found* **then**
      $Q_{sol} \leftarrow UpdateSolution(\mathscr{T})$
      $M \leftarrow UpdateMilestone(Q_{sol})$
      $\mathscr{S} \leftarrow UpdateSamplingCloud(\mathscr{S}, M)$
**return** $\mathscr{T}$

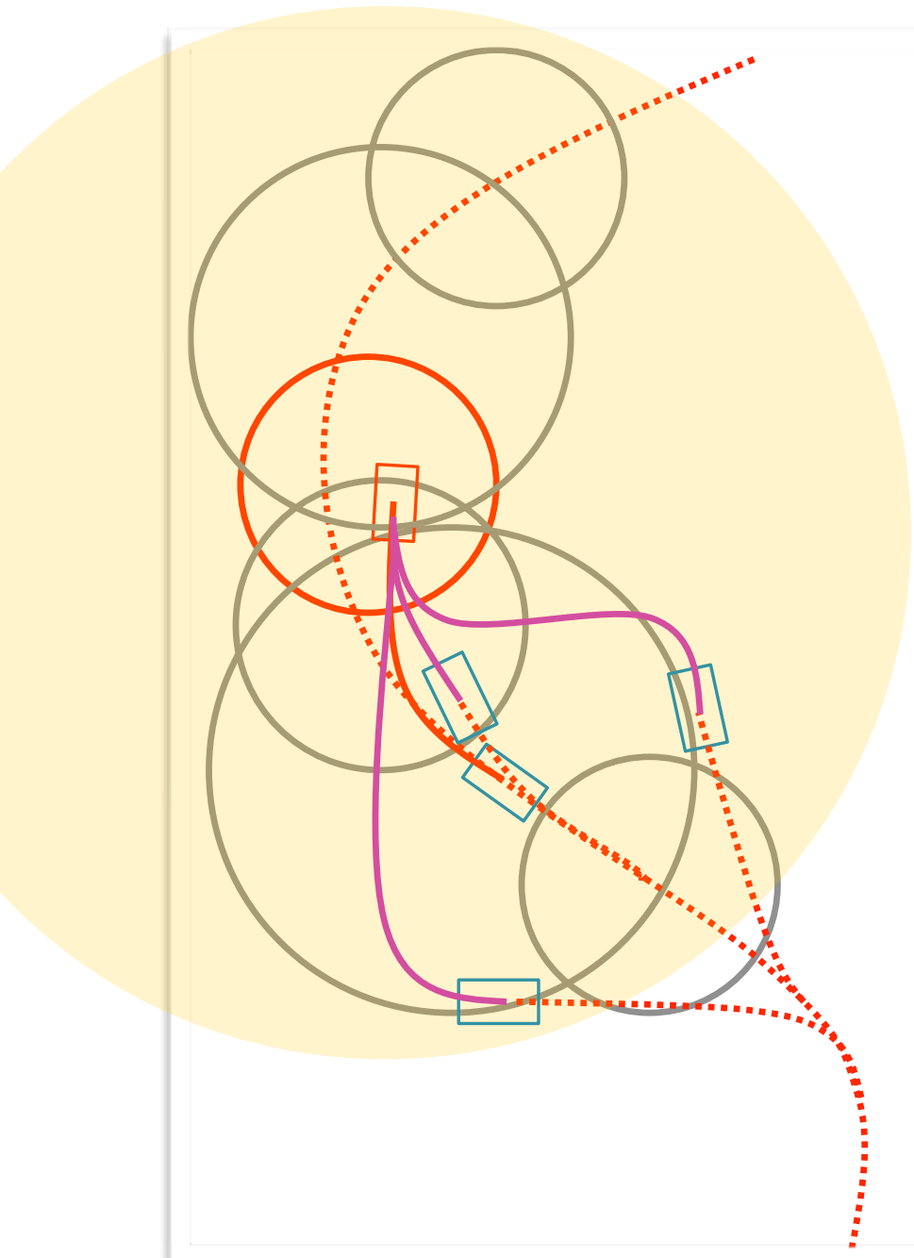
# Cloud RRT*



**Input**: A sampling cloud $\mathscr{S}$, an init. configuration, $q_{init}$, and a goal region, $Q_{goal}$
**Output**: A random tree, $\mathscr{T}$, and a solution path, $Q_{sol}$

$\mathscr{T} \leftarrow \{q_{init}\}$ and $Q_{sol} \leftarrow \emptyset$
**while** ***not*** *termination conditions are satisfied* **do**
  s$\leftarrow SampleSphere(\mathscr{S})$
  $q_s \leftarrow Sample(\mathrm{s})$
  $q_n \leftarrow Nearest(q_s)$
  $(q_{new}, u_{new}) \leftarrow Steer(q_n, q_s)$
  **if** $IsCollisionFree(q_{new}, u_{new}, q_n)$ **then**
    $Q_{near} \leftarrow Near(q_{new})$
    $q_{min} \leftarrow ChooseParent(Q_{near}, q_n, q_{new})$
    $\mathscr{T} \leftarrow InsertNode(q_{min}, q_{new}, \mathscr{T}, s)$
    $\mathscr{T} \leftarrow Rewire(\mathscr{T}, Q_{near}, q_{min}, q_{new})$
    **if** *A better solution is found* **then**
      $Q_{sol} \leftarrow UpdateSolution(\mathscr{T})$
      $M \leftarrow UpdateMilestone(Q_{sol})$
      $\mathscr{S} \leftarrow UpdateSamplingCloud(\mathscr{S}, M)$
**return** $\mathscr{T}$

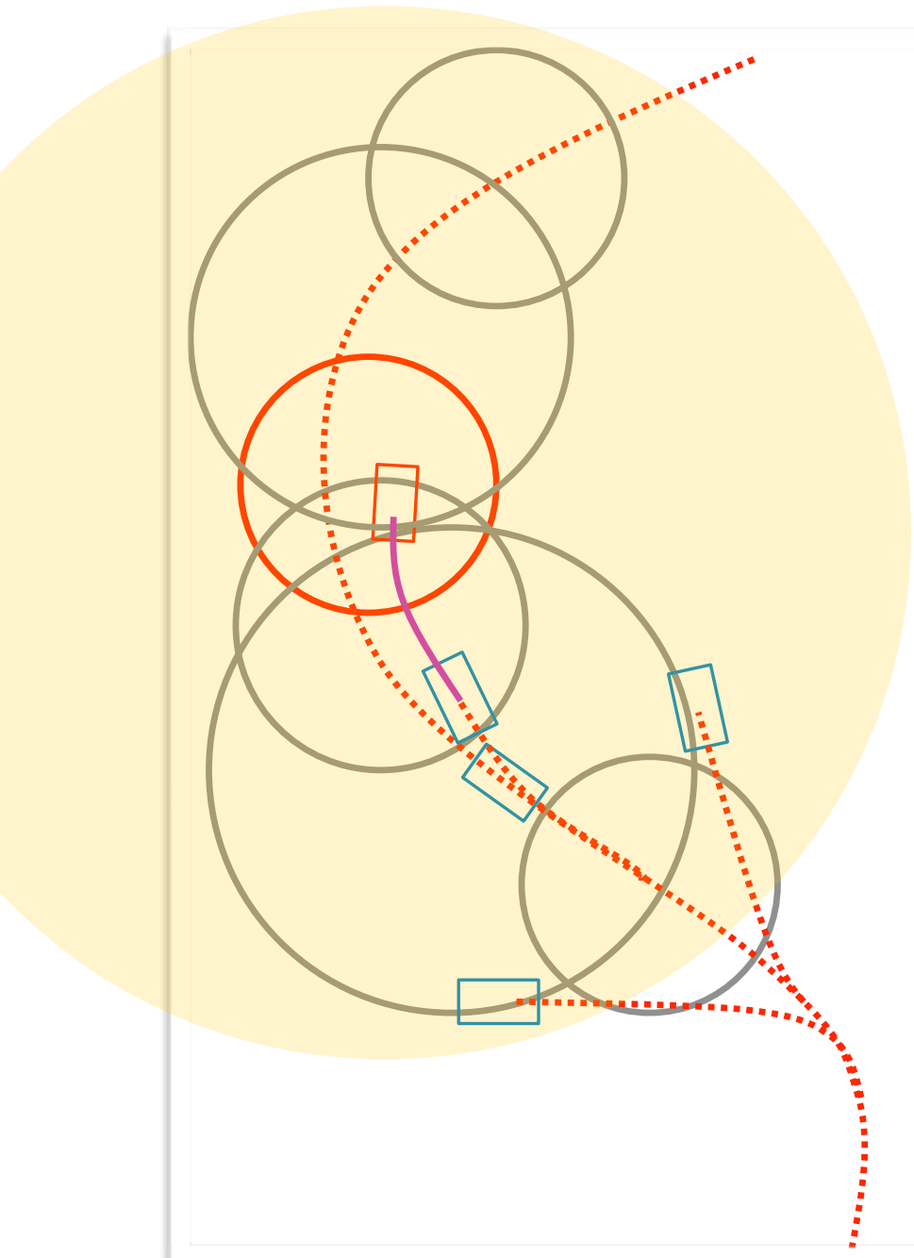
# Cloud RRT*



**Input**: A sampling cloud $\mathscr{S}$, an init. configuration, $q_{init}$, and a goal region, $Q_{goal}$
**Output**: A random tree, $\mathscr{T}$, and a solution path, $Q_{sol}$

$\mathscr{T} \leftarrow \{q_{init}\}$ and $Q_{sol} \leftarrow \emptyset$
**while** ***not*** *termination conditions are satisfied* **do**
  $s \leftarrow SampleSphere(\mathscr{S})$
  $q_s \leftarrow Sample(s)$
  $q_n \leftarrow Nearest(q_s)$
  $(q_{new}, u_{new}) \leftarrow Steer(q_n, q_s)$
  **if** $IsCollisionFree(q_{new}, u_{new}, q_n)$ **then**
    $Q_{near} \leftarrow Near(q_{new})$
    $q_{min} \leftarrow ChooseParent(Q_{near}, q_n, q_{new})$
    $\mathscr{T} \leftarrow InsertNode(q_{min}, q_{new}, \mathscr{T}, s)$
    $\mathscr{T} \leftarrow Rewire(\mathscr{T}, Q_{near}, q_{min}, q_{new})$
    **if** *A better solution is found* **then**
      $Q_{sol} \leftarrow UpdateSolution(\mathscr{T})$
      $M \leftarrow UpdateMilestone(Q_{sol})$
      $\mathscr{S} \leftarrow UpdateSamplingCloud(\mathscr{S}, M)$
**return** $\mathscr{T}$

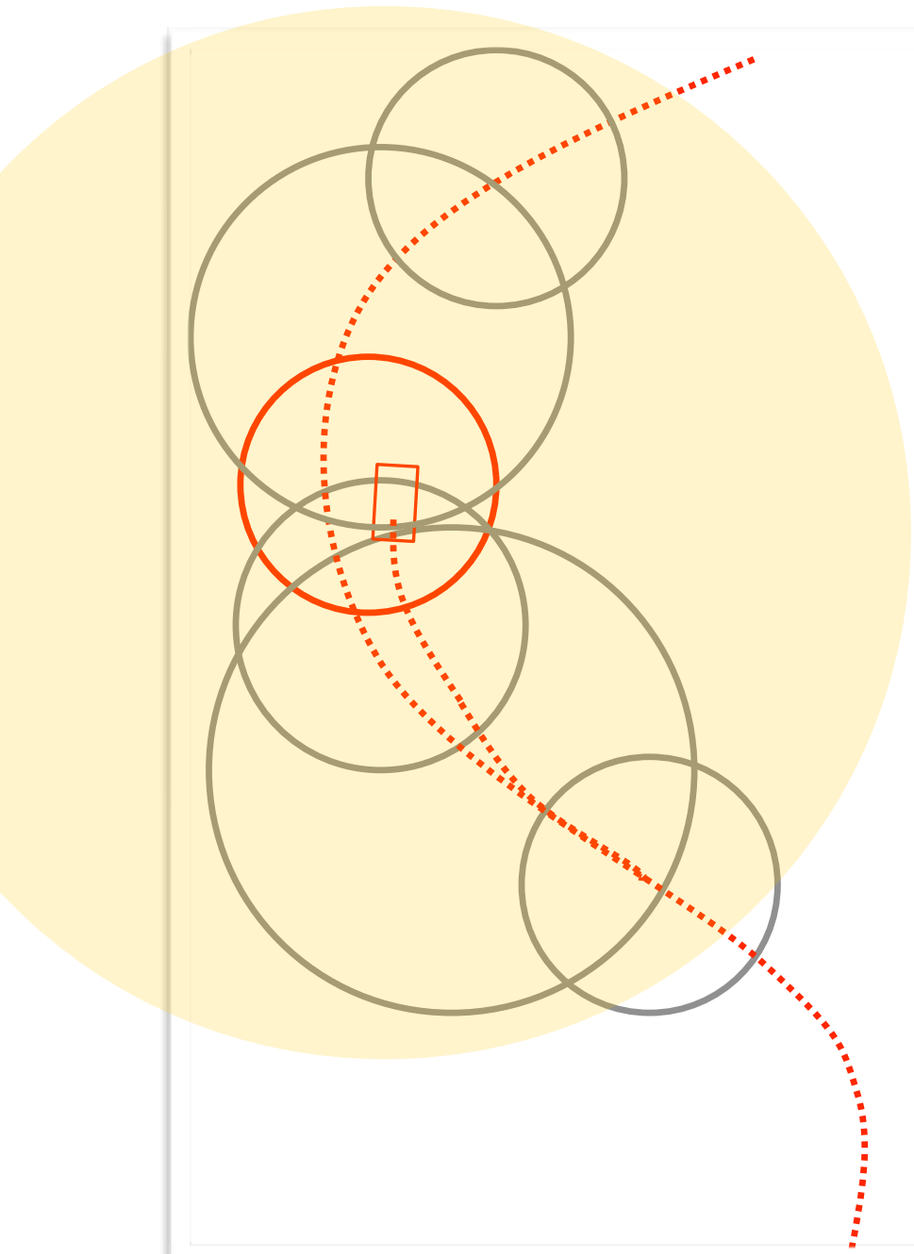
# Cloud RRT*



**Input**: A sampling cloud $\mathscr{S}$, an init. configuration, $q_{init}$, and a goal region, $Q_{goal}$
**Output**: A random tree, $\mathscr{T}$, and a solution path, $Q_{sol}$

$\mathscr{T} \leftarrow \{q_{init}\}$ and $Q_{sol} \leftarrow \emptyset$
**while** ***not*** *termination conditions are satisfied* **do**
  s$\leftarrow SampleSphere(\mathscr{S})$
  $q_s \leftarrow Sample(\mathrm{s})$
  $q_n \leftarrow Nearest(q_s)$
  $(q_{new}, u_{new}) \leftarrow Steer(q_n, q_s)$
  **if** $IsCollisionFree(q_{new}, u_{new}, q_n)$ **then**
    $Q_{near} \leftarrow Near(q_{new})$
    $q_{min} \leftarrow ChooseParent(Q_{near}, q_n, q_{new})$
    $\mathscr{T} \leftarrow InsertNode(q_{min}, q_{new}, \mathscr{T}, s)$
    $\mathscr{T} \leftarrow Rewire(\mathscr{T}, Q_{near}, q_{min}, q_{new})$
    **if** *A better solution is found* **then**
      $Q_{sol} \leftarrow UpdateSolution(\mathscr{T})$
      $M \leftarrow UpdateMilestone(Q_{sol})$
      $\mathscr{S} \leftarrow UpdateSamplingCloud(\mathscr{S}, M)$
**return** $\mathscr{T}$

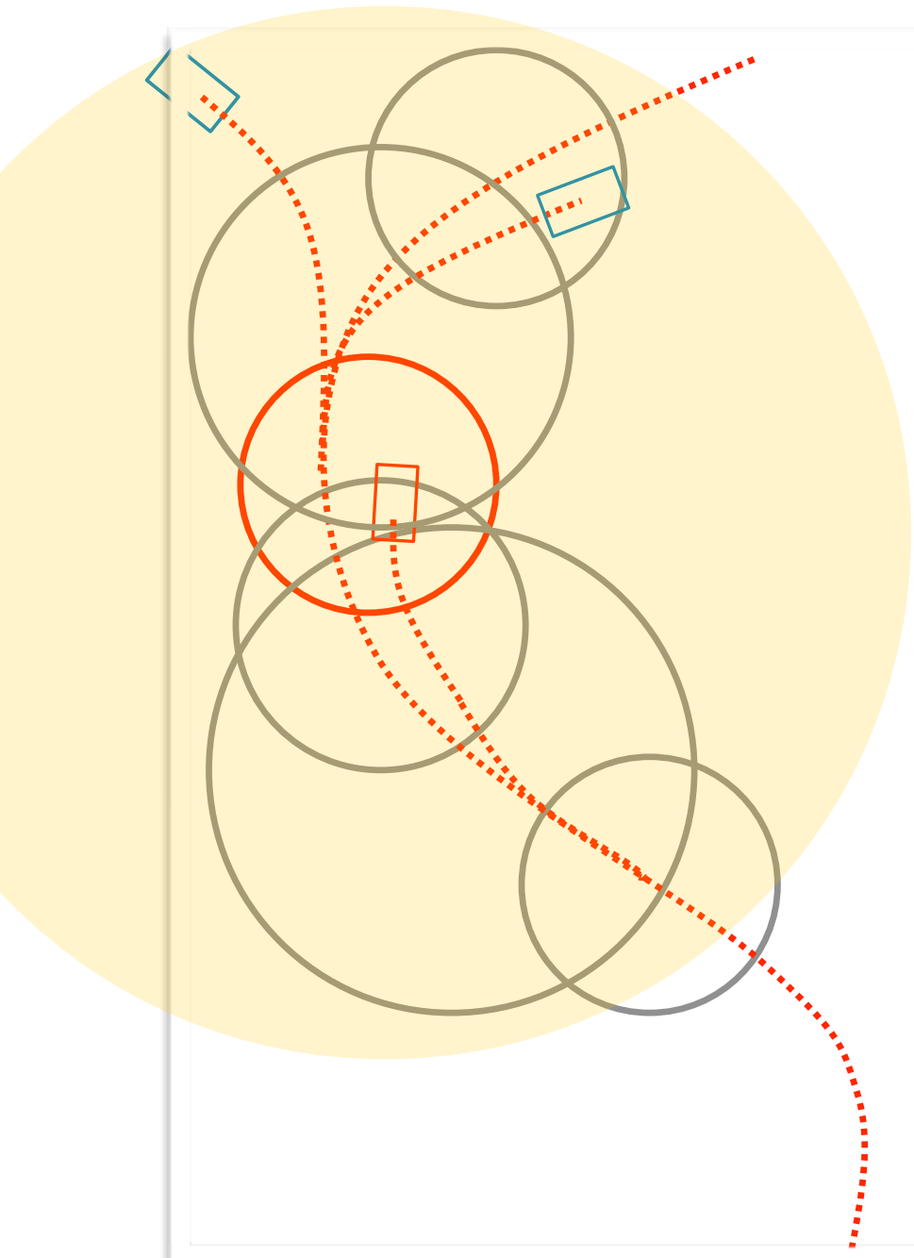
# Cloud RRT*



**Input**: A sampling cloud $\mathscr{S}$, an init. configuration, $q_{init}$, and a goal region, $Q_{goal}$
**Output**: A random tree, $\mathscr{T}$, and a solution path, $Q_{sol}$

$\mathscr{T} \leftarrow \{q_{init}\}$ and $Q_{sol} \leftarrow \emptyset$
**while** ***not*** *termination conditions are satisfied* **do**
  $s \leftarrow SampleSphere(\mathscr{S})$
  $q_s \leftarrow Sample(s)$
  $q_n \leftarrow Nearest(q_s)$
  $(q_{new}, u_{new}) \leftarrow Steer(q_n, q_s)$
  **if** $IsCollisionFree(q_{new}, u_{new}, q_n)$ **then**
    $Q_{near} \leftarrow Near(q_{new})$
    $q_{min} \leftarrow ChooseParent(Q_{near}, q_n, q_{new})$
    $\mathscr{T} \leftarrow InsertNode(q_{min}, q_{new}, \mathscr{T}, s)$
    $\mathscr{T} \leftarrow Rewire(\mathscr{T}, Q_{near}, q_{min}, q_{new})$
    **if** *A better solution is found* **then**
      $Q_{sol} \leftarrow UpdateSolution(\mathscr{T})$
      $M \leftarrow UpdateMilestone(Q_{sol})$
      $\mathscr{S} \leftarrow UpdateSamplingCloud(\mathscr{S}, M)$
**return** $\mathscr{T}$

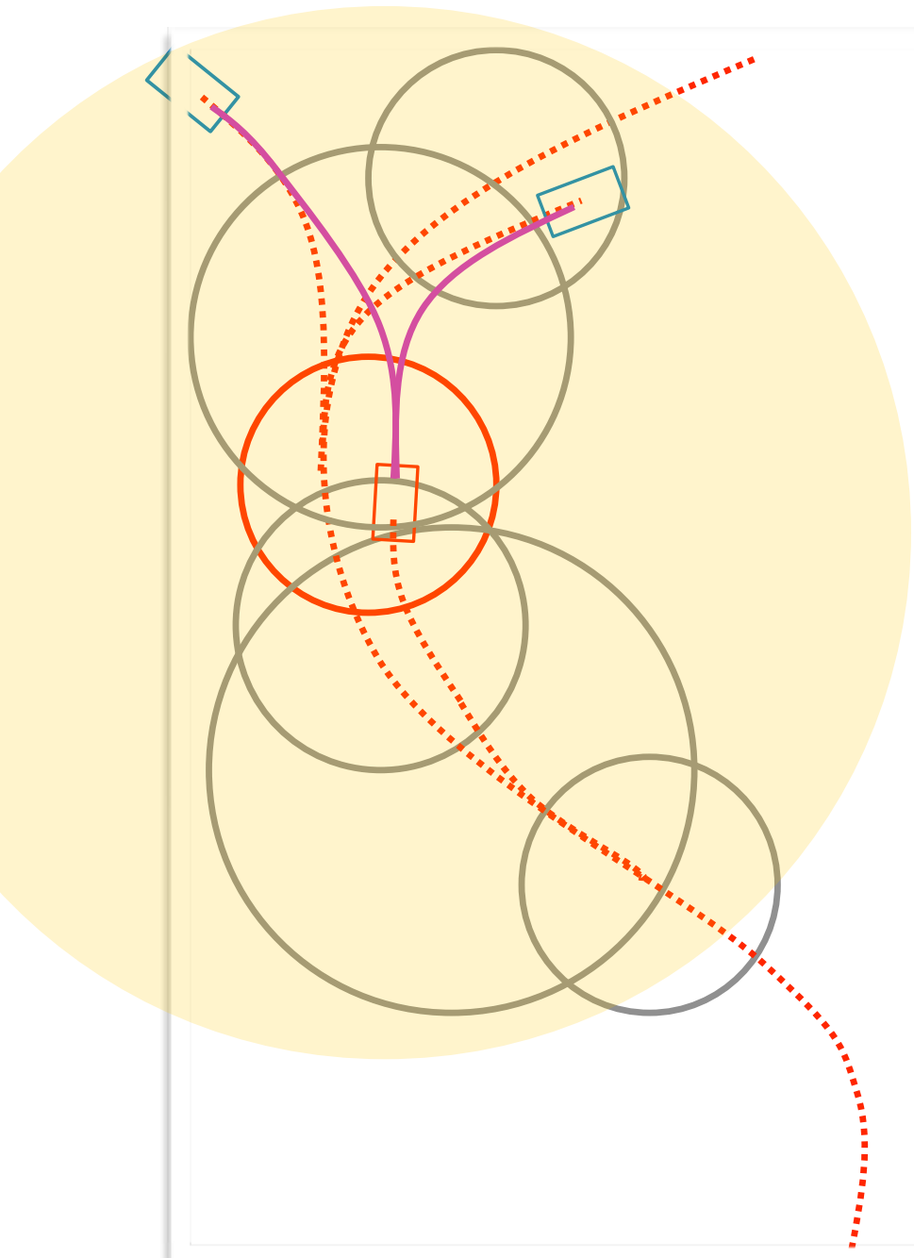
# Cloud RRT*



**Input**: A sampling cloud $\mathscr{S}$, an init. configuration, $q_{init}$, and a goal region, $Q_{goal}$
**Output**: A random tree, $\mathscr{T}$, and a solution path, $Q_{sol}$

$\mathscr{T} \leftarrow \{q_{init}\}$ and $Q_{sol} \leftarrow \emptyset$
**while** ***not*** *termination conditions are satisfied* **do**
  $s \leftarrow SampleSphere(\mathscr{S})$
  $q_s \leftarrow Sample(s)$
  $q_n \leftarrow Nearest(q_s)$
  $(q_{new}, u_{new}) \leftarrow Steer(q_n, q_s)$
  **if** $IsCollisionFree(q_{new}, u_{new}, q_n)$ **then**
    $Q_{near} \leftarrow Near(q_{new})$
    $q_{min} \leftarrow ChooseParent(Q_{near}, q_n, q_{new})$
    $\mathscr{T} \leftarrow InsertNode(q_{min}, q_{new}, \mathscr{T}, s)$
    $\mathscr{T} \leftarrow Rewire(\mathscr{T}, Q_{near}, q_{min}, q_{new})$
    **if** *A better solution is found* **then**
      $Q_{sol} \leftarrow UpdateSolution(\mathscr{T})$
      $M \leftarrow UpdateMilestone(Q_{sol})$
      $\mathscr{S} \leftarrow UpdateSamplingCloud(\mathscr{S}, M)$
**return** $\mathscr{T}$

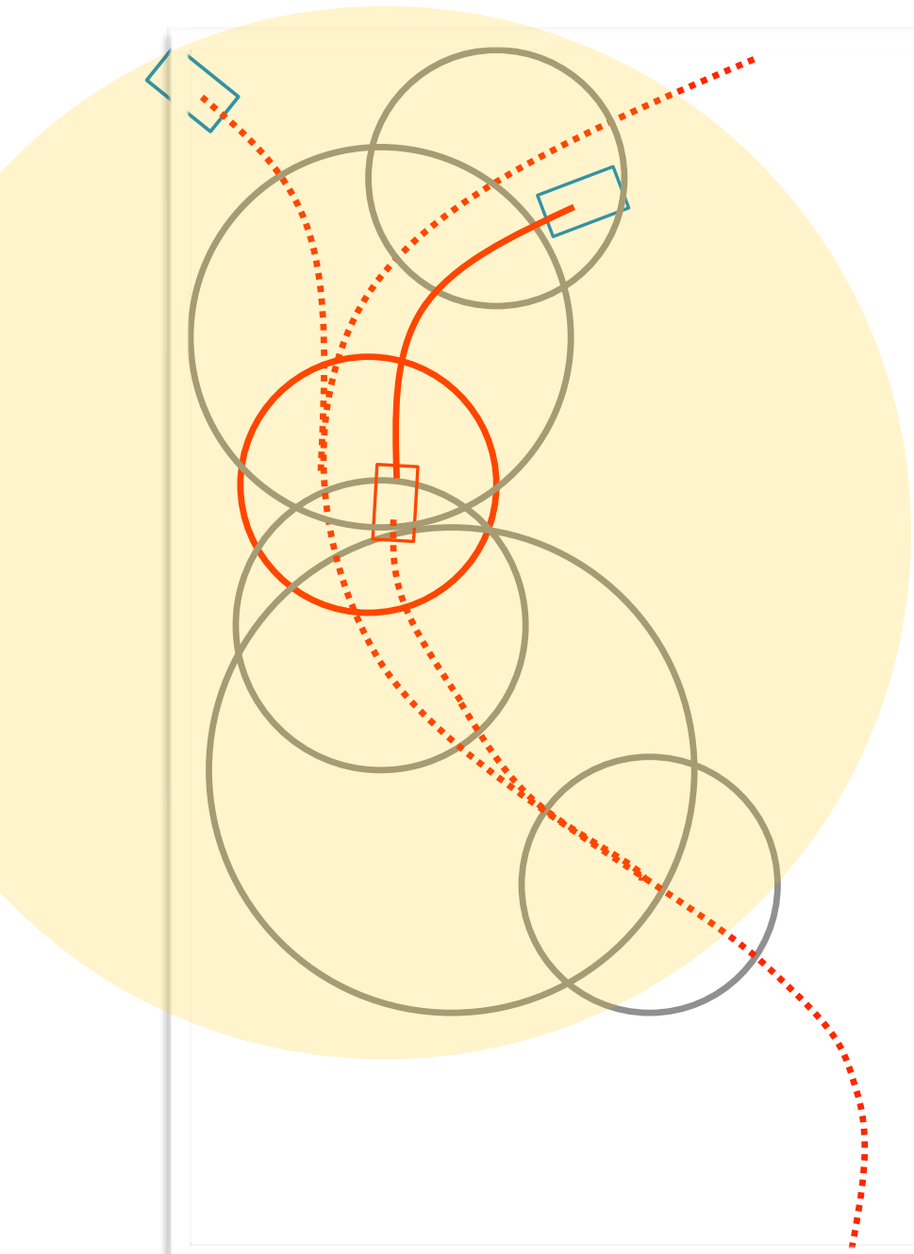
# Cloud RRT*



**Input**: A sampling cloud $\mathscr{S}$, an init. configuration, $q_{init}$, and a goal region, $Q_{goal}$
**Output**: A random tree, $\mathscr{T}$, and a solution path, $Q_{sol}$

$\mathscr{T} \leftarrow \{q_{init}\}$ and $Q_{sol} \leftarrow \emptyset$
**while** ***not*** *termination conditions are satisfied* **do**
  $s \leftarrow SampleSphere(\mathscr{S})$
  $q_s \leftarrow Sample(s)$
  $q_n \leftarrow Nearest(q_s)$
  $(q_{new}, u_{new}) \leftarrow Steer(q_n, q_s)$
  **if** $IsCollisionFree(q_{new}, u_{new}, q_n)$ **then**
    $Q_{near} \leftarrow Near(q_{new})$
    $q_{min} \leftarrow ChooseParent(Q_{near}, q_n, q_{new})$
    $\mathscr{T} \leftarrow InsertNode(q_{min}, q_{new}, \mathscr{T}, s)$
    $\mathscr{T} \leftarrow Rewire(\mathscr{T}, Q_{near}, q_{min}, q_{new})$
    **if** *A better solution is found* **then**
      $Q_{sol} \leftarrow UpdateSolution(\mathscr{T})$
      $M \leftarrow UpdateMilestone(Q_{sol})$
      $\mathscr{S} \leftarrow UpdateSamplingCloud(\mathscr{S}, M)$
**return** $\mathscr{T}$

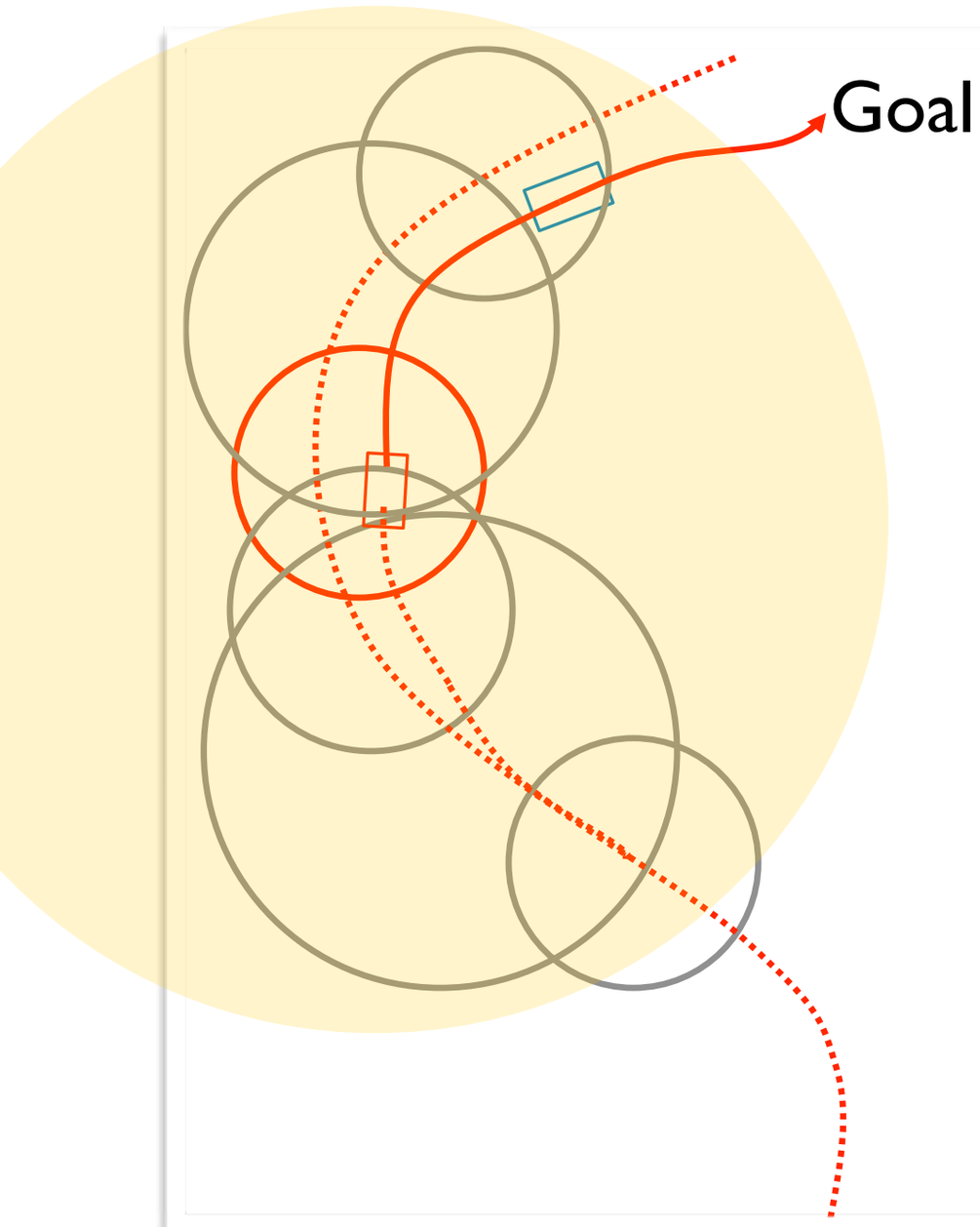
# Cloud RRT*



**Input**: A sampling cloud $\mathscr{S}$, an init. configuration, $q_{init}$, and a goal region, $Q_{goal}$
**Output**: A random tree, $\mathscr{T}$, and a solution path, $Q_{sol}$

$\mathscr{T} \leftarrow \{q_{init}\}$ and $Q_{sol} \leftarrow \emptyset$
**while** ***not*** *termination conditions are satisfied* **do**
  $s \leftarrow SampleSphere(\mathscr{S})$
  $q_s \leftarrow Sample(s)$
  $q_n \leftarrow Nearest(q_s)$
  $(q_{new}, u_{new}) \leftarrow Steer(q_n, q_s)$
  **if** $IsCollisionFree(q_{new}, u_{new}, q_n)$ **then**
    $Q_{near} \leftarrow Near(q_{new})$
    $q_{min} \leftarrow ChooseParent(Q_{near}, q_n, q_{new})$
    $\mathscr{T} \leftarrow InsertNode(q_{min}, q_{new}, \mathscr{T}, s)$
    $\mathscr{T} \leftarrow Rewire(\mathscr{T}, Q_{near}, q_{min}, q_{new})$
    **if** *A better solution is found* **then**
      $Q_{sol} \leftarrow UpdateSolution(\mathscr{T})$
      $M \leftarrow UpdateMilestone(Q_{sol})$
      $\mathscr{S} \leftarrow UpdateSamplingCloud(\mathscr{S}, M)$
**return** $\mathscr{T}$

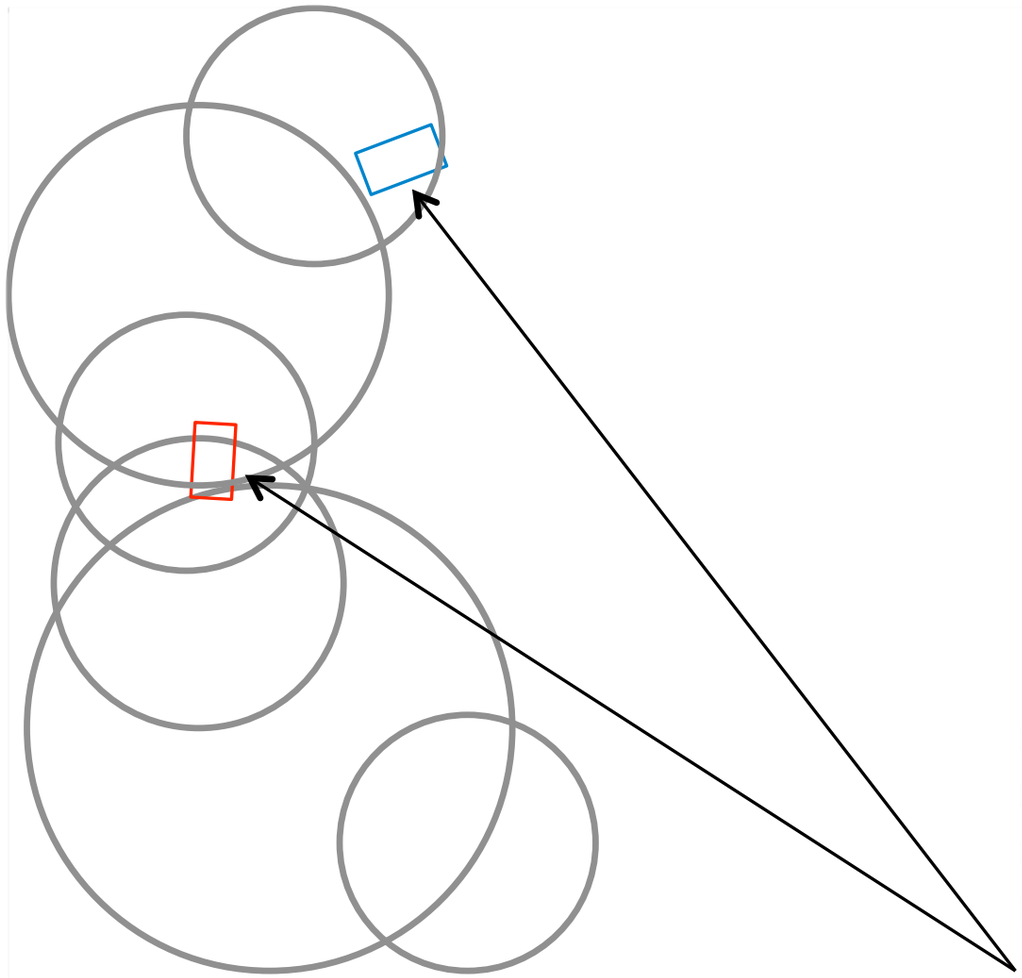
# Cloud RRT*




**Input**: A sampling cloud $\mathscr{S}$, an init. configuration, $q_{init}$, and a goal region, $Q_{goal}$
**Output**: A random tree, $\mathscr{T}$, and a solution path, $Q_{sol}$

$\mathscr{T} \leftarrow \{q_{init}\}$ and $Q_{sol} \leftarrow \emptyset$
**while** ***not*** *termination conditions are satisfied* **do**
  $s \leftarrow SampleSphere(\mathscr{S})$
  $q_s \leftarrow Sample(s)$
  $q_n \leftarrow Nearest(q_s)$
  $(q_{new}, u_{new}) \leftarrow Steer(q_n, q_s)$
  **if** $IsCollisionFree(q_{new}, u_{new}, q_n)$ **then**
    $Q_{near} \leftarrow Near(q_{new})$
    $q_{min} \leftarrow ChooseParent(Q_{near}, q_n, q_{new})$
    $\mathscr{T} \leftarrow InsertNode(q_{min}, q_{new}, \mathscr{T}, s)$
    $\mathscr{T} \leftarrow Rewire(\mathscr{T}, Q_{near}, q_{min}, q_{new})$
    **if** *A better solution is found* **then**
      $Q_{sol} \leftarrow UpdateSolution(\mathscr{T})$
      $M \leftarrow UpdateMilestone(Q_{sol})$
      $\mathscr{S} \leftarrow UpdateSamplingCloud(\mathscr{S}, M)$
**return** $\mathscr{T}$

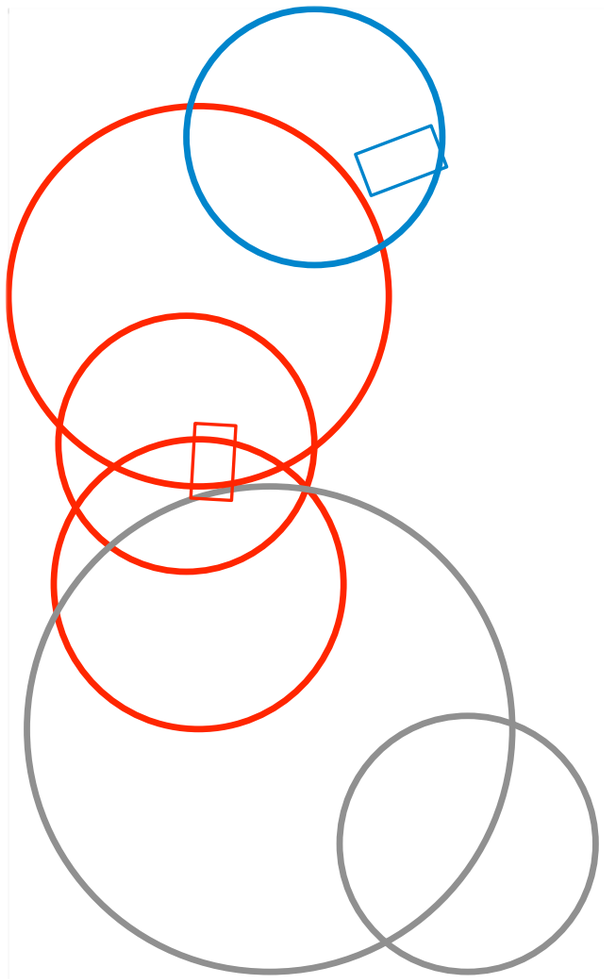
# Cloud RRT* : Update



**Input**: A sampling cloud $\mathscr{S}$, an init. configuration, $q_{init}$, and a goal region, $Q_{goal}$
**Output**: A random tree, $\mathscr{T}$, and a solution path, $Q_{sol}$

$\mathscr{T} \leftarrow \{q_{init}\}$ and $Q_{sol} \leftarrow \emptyset$
**while** ***not*** *termination conditions are satisfied* **do**
  $s \leftarrow SampleSphere(\mathscr{S})$
  $q_s \leftarrow Sample(s)$
  $q_n \leftarrow Nearest(q_s)$
  $(q_{new}, u_{new}) \leftarrow Steer(q_n, q_s)$
  **if** $IsCollisionFree(q_{new}, u_{new}, q_n)$ **then**
    $Q_{near} \leftarrow Near(q_{new})$
    $q_{min} \leftarrow ChooseParent(Q_{near}, q_n, q_{new})$
    $\mathscr{T} \leftarrow InsertNode(q_{min}, q_{new}, \mathscr{T}, s)$
    $\mathscr{T} \leftarrow Rewire(\mathscr{T}, Q_{near}, q_{min}, q_{new})$
    **if** *A better solution is found* **then**
      $Q_{sol} \leftarrow UpdateSolution(\mathscr{T})$
      $M \leftarrow UpdateMilestone(Q_{sol})$
      $\mathscr{S} \leftarrow UpdateSamplingCloud(\mathscr{S}, M)$
**return** $\mathscr{T}$

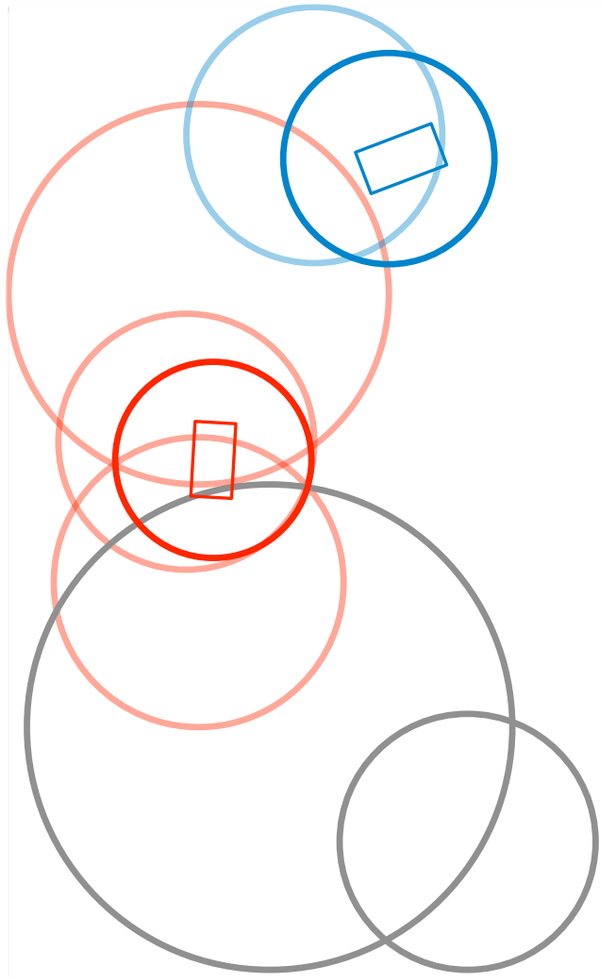
# Cloud RRT* : Update



**Input**: A sampling cloud $\mathscr{S}$, an init. configuration, $q_{init}$, and a goal region, $Q_{goal}$
**Output**: A random tree, $\mathscr{T}$, and a solution path, $Q_{sol}$

$\mathscr{T} \leftarrow \{q_{init}\}$ and $Q_{sol} \leftarrow \emptyset$
**while** ***not*** *termination conditions are satisfied* **do**
  $s \leftarrow SampleSphere(\mathscr{S})$
  $q_s \leftarrow Sample(s)$
  $q_n \leftarrow Nearest(q_s)$
  $(q_{new}, u_{new}) \leftarrow Steer(q_n, q_s)$
  **if** $IsCollisionFree(q_{new}, u_{new}, q_n)$ **then**
    $Q_{near} \leftarrow Near(q_{new})$
    $q_{min} \leftarrow ChooseParent(Q_{near}, q_n, q_{new})$
    $\mathscr{T} \leftarrow InsertNode(q_{min}, q_{new}, \mathscr{T}, s)$
    $\mathscr{T} \leftarrow Rewire(\mathscr{T}, Q_{near}, q_{min}, q_{new})$
    **if** *A better solution is found* **then**
      $Q_{sol} \leftarrow UpdateSolution(\mathscr{T})$
      $M \leftarrow UpdateMilestone(Q_{sol})$
      $\mathscr{S} \leftarrow UpdateSamplingCloud(\mathscr{S}, M)$
**return** $\mathscr{T}$

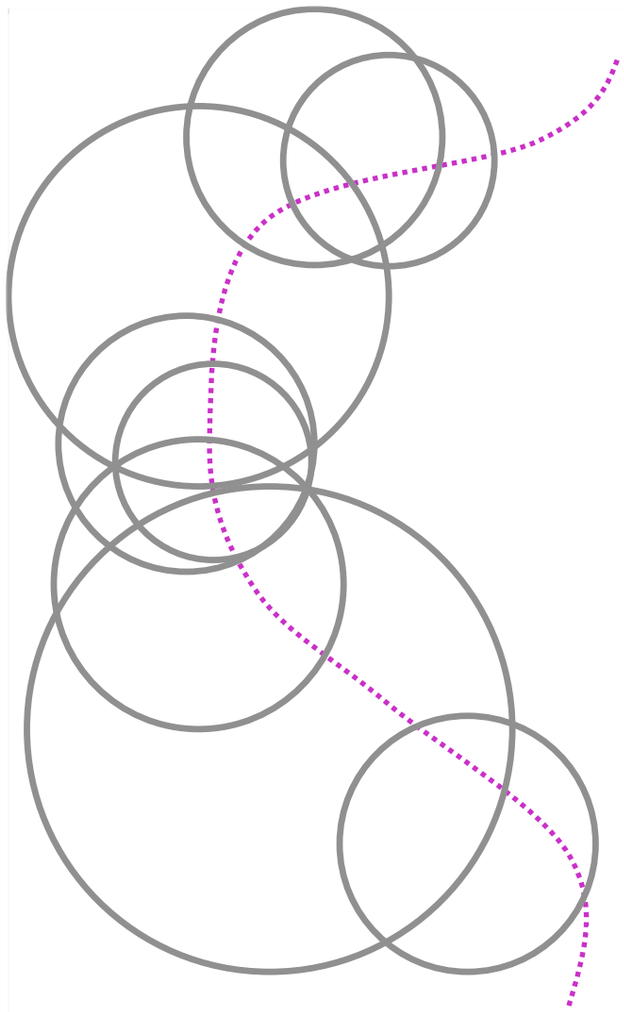
# Cloud RRT* : Update



**Input**: A sampling cloud $\mathscr{S}$, an init. configuration, $q_{init}$, and a goal region, $Q_{goal}$
**Output**: A random tree, $\mathscr{T}$, and a solution path, $Q_{sol}$

$\mathscr{T} \leftarrow \{q_{init}\}$ and $Q_{sol} \leftarrow \emptyset$
**while** ***not*** *termination conditions are satisfied* **do**
  s$\leftarrow SampleSphere(\mathscr{S})$
  $q_s \leftarrow Sample(\mathrm{s})$
  $q_n \leftarrow Nearest(q_s)$
  $(q_{new}, u_{new}) \leftarrow Steer(q_n, q_s)$
  **if** $IsCollisionFree(q_{new}, u_{new}, q_n)$ **then**
    $Q_{near} \leftarrow Near(q_{new})$
    $q_{min} \leftarrow ChooseParent(Q_{near}, q_n, q_{new})$
    $\mathscr{T} \leftarrow InsertNode(q_{min}, q_{new}, \mathscr{T}, s)$
    $\mathscr{T} \leftarrow Rewire(\mathscr{T}, Q_{near}, q_{min}, q_{new})$
    **if** *A better solution is found* **then**
      $Q_{sol} \leftarrow UpdateSolution(\mathscr{T})$
      $M \leftarrow UpdateMilestone(Q_{sol})$
      $\mathscr{S} \leftarrow UpdateSamplingCloud(\mathscr{S}, M)$
**return** $\mathscr{T}$

# Cloud RRT*

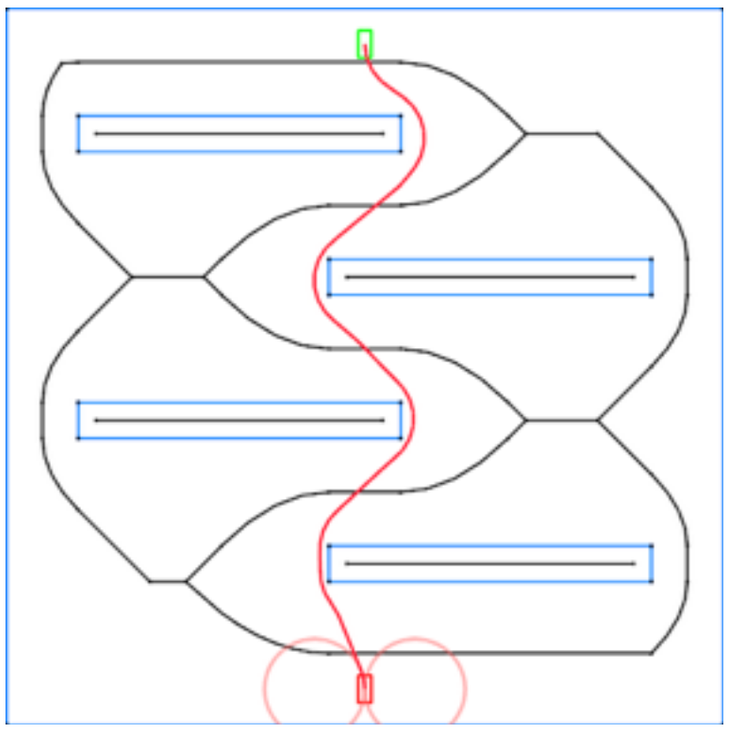


**Input**: A sampling cloud $\mathscr{S}$, an init. configuration, $q_{init}$, and a goal region, $Q_{goal}$
**Output**: A random tree, $\mathscr{T}$, and a solution path, $Q_{sol}$

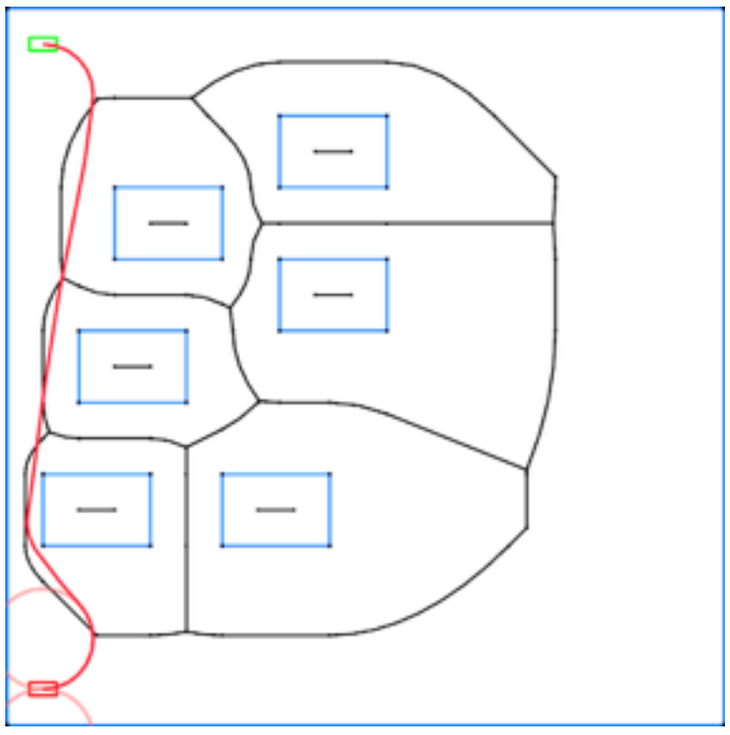
$\mathscr{T} \leftarrow \{q_{init}\}$ and $Q_{sol} \leftarrow \emptyset$
**while** ***not*** *termination conditions are satisfied* **do**
  $\mathrm{s} \leftarrow SampleSphere(\mathscr{S})$
  $q_s \leftarrow Sample(\mathrm{s})$
  $q_n \leftarrow Nearest(q_s)$
  $(q_{new}, u_{new}) \leftarrow Steer(q_n, q_s)$
  **if** $IsCollisionFree(q_{new}, u_{new}, q_n)$ **then**
    $Q_{near} \leftarrow Near(q_{new})$
    $q_{min} \leftarrow ChooseParent(Q_{near}, q_n, q_{new})$
    $\mathscr{T} \leftarrow InsertNode(q_{min}, q_{new}, \mathscr{T}, s)$
    $\mathscr{T} \leftarrow Rewire(\mathscr{T}, Q_{near}, q_{min}, q_{new})$
    **if** *A better solution is found* **then**
      $Q_{sol} \leftarrow UpdateSolution(\mathscr{T})$
      $M \leftarrow UpdateMilestone(Q_{sol})$
      $\mathscr{S} \leftarrow UpdateSamplingCloud(\mathscr{S}, M)$
**return** $\mathscr{T}$

# Experimental Results:

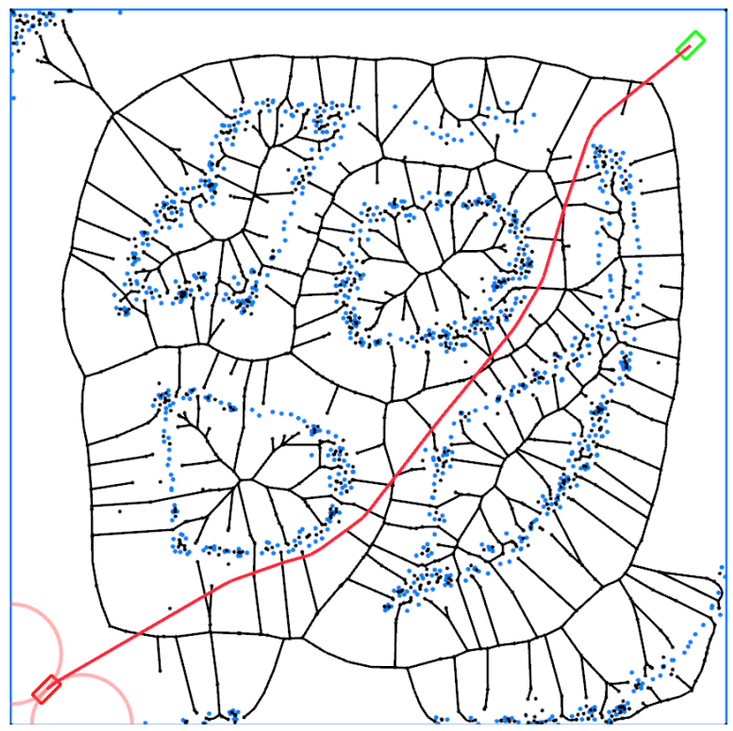
- Tested our algorithm on 3 different 2D environments
- Dubins vehicle model (Dubin,1957) with kinematic constraints



*4squares*



*6squares*

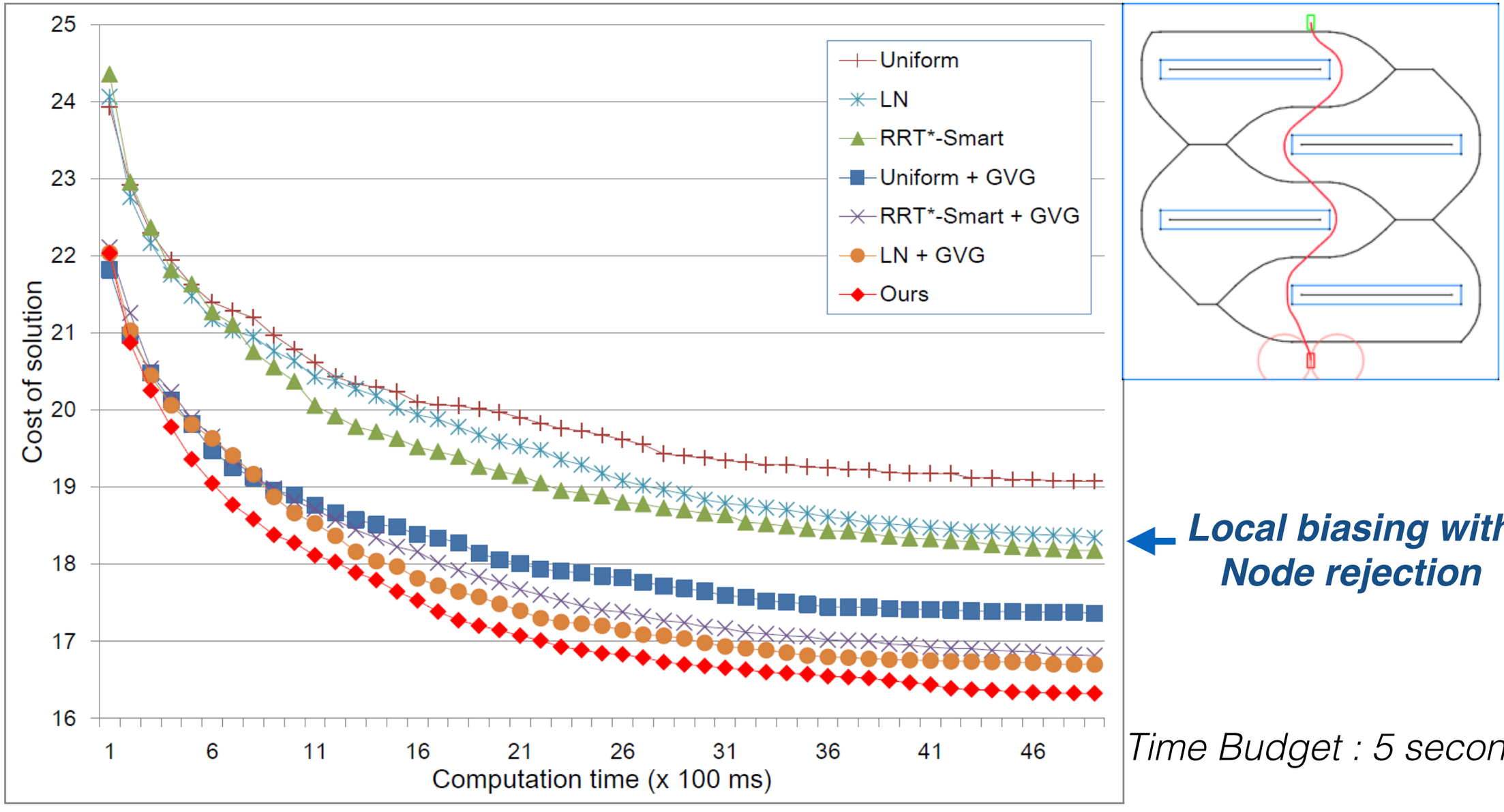


*Many points*

# Results:

- Tested our algorithm on 3 different 2D environment
- Dubins vehicle model with kinematic constraints
- Compared with:
  - ***Original RRT* (Uniform sampling)***
  - ***Akgun & Stilman, Sampling heuristic for optimal motion planning in high dimensions, IROS2011***
  - ***Islam et al., RRT*-Smart : Rapid convergence implementation of RRT* toward optimal solution, ICMA2012***

# Results: 4 squares

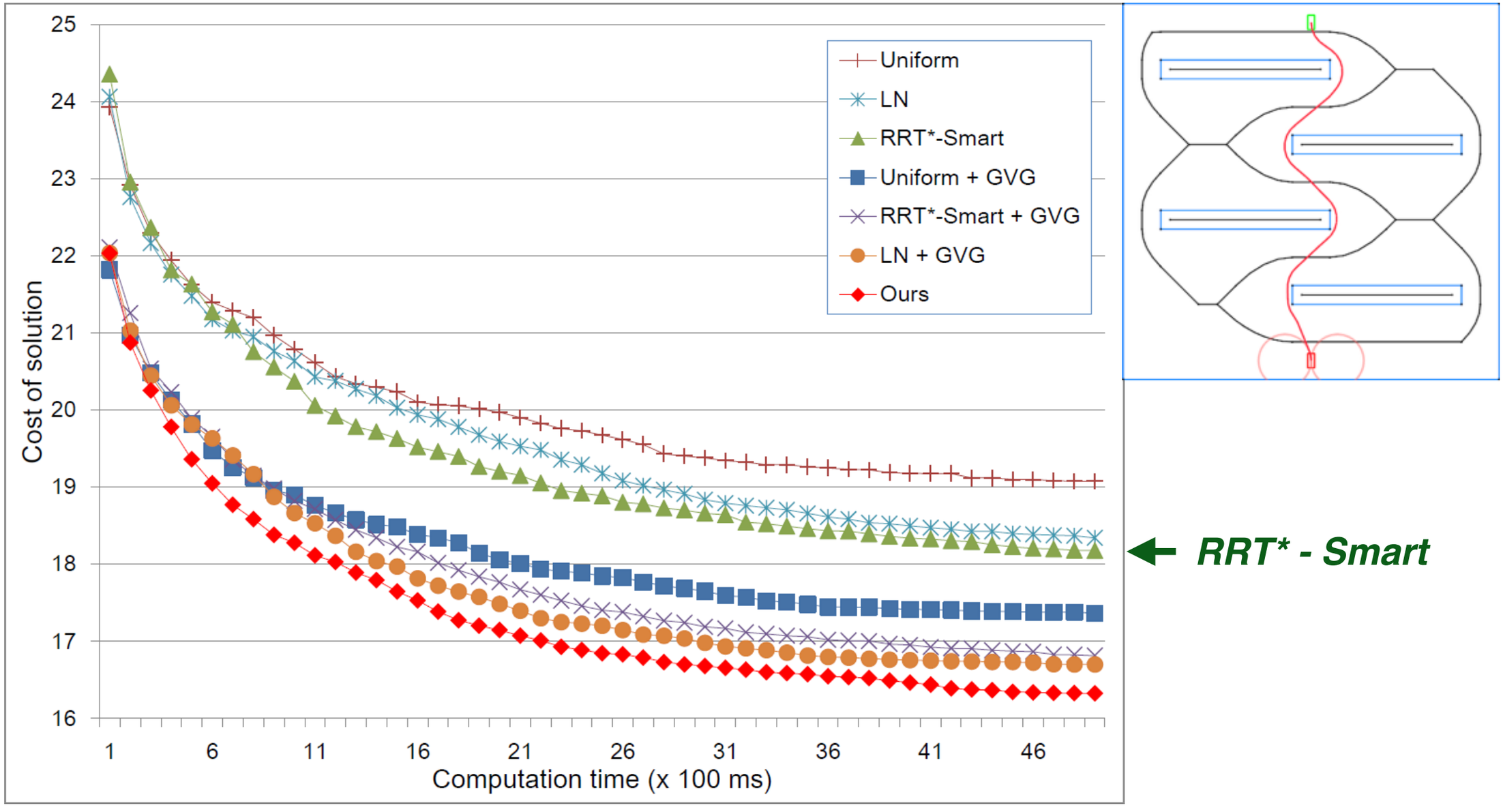





***Local biasing with Node rejection***

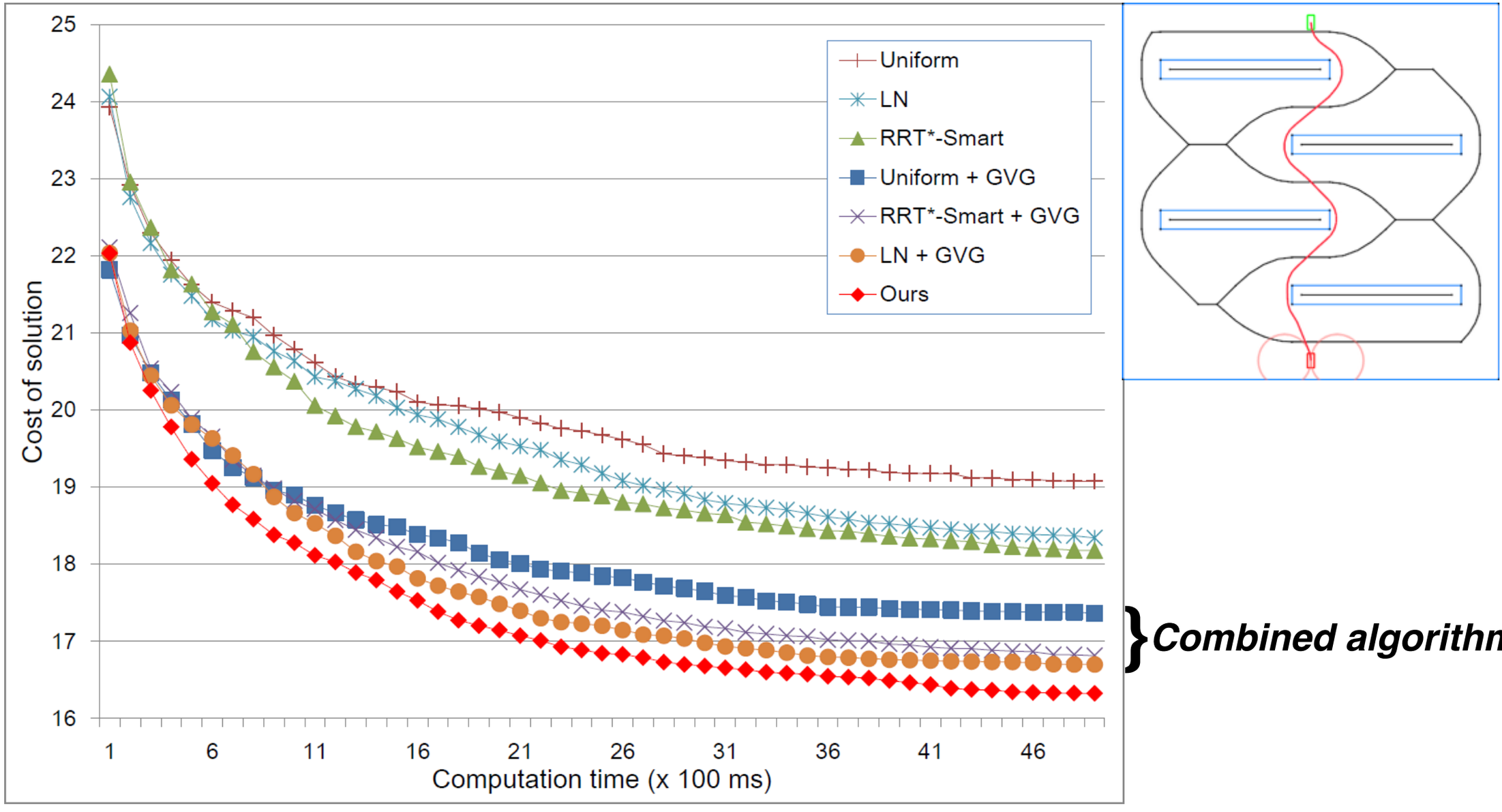
*Time Budget : 5 seconds*

# Results: 4 squares




← ***RRT* - Smart***

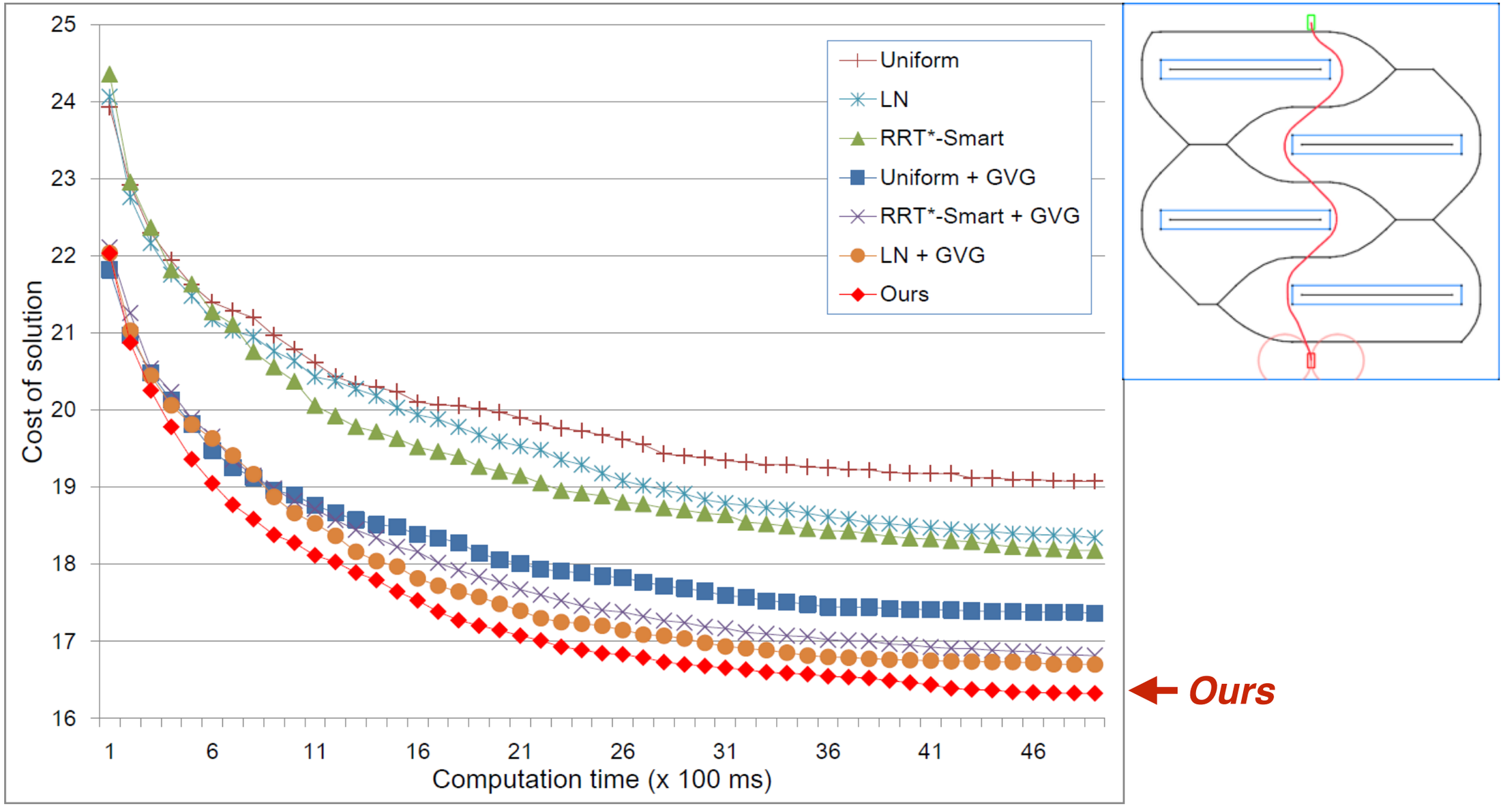
# Results: 4 squares



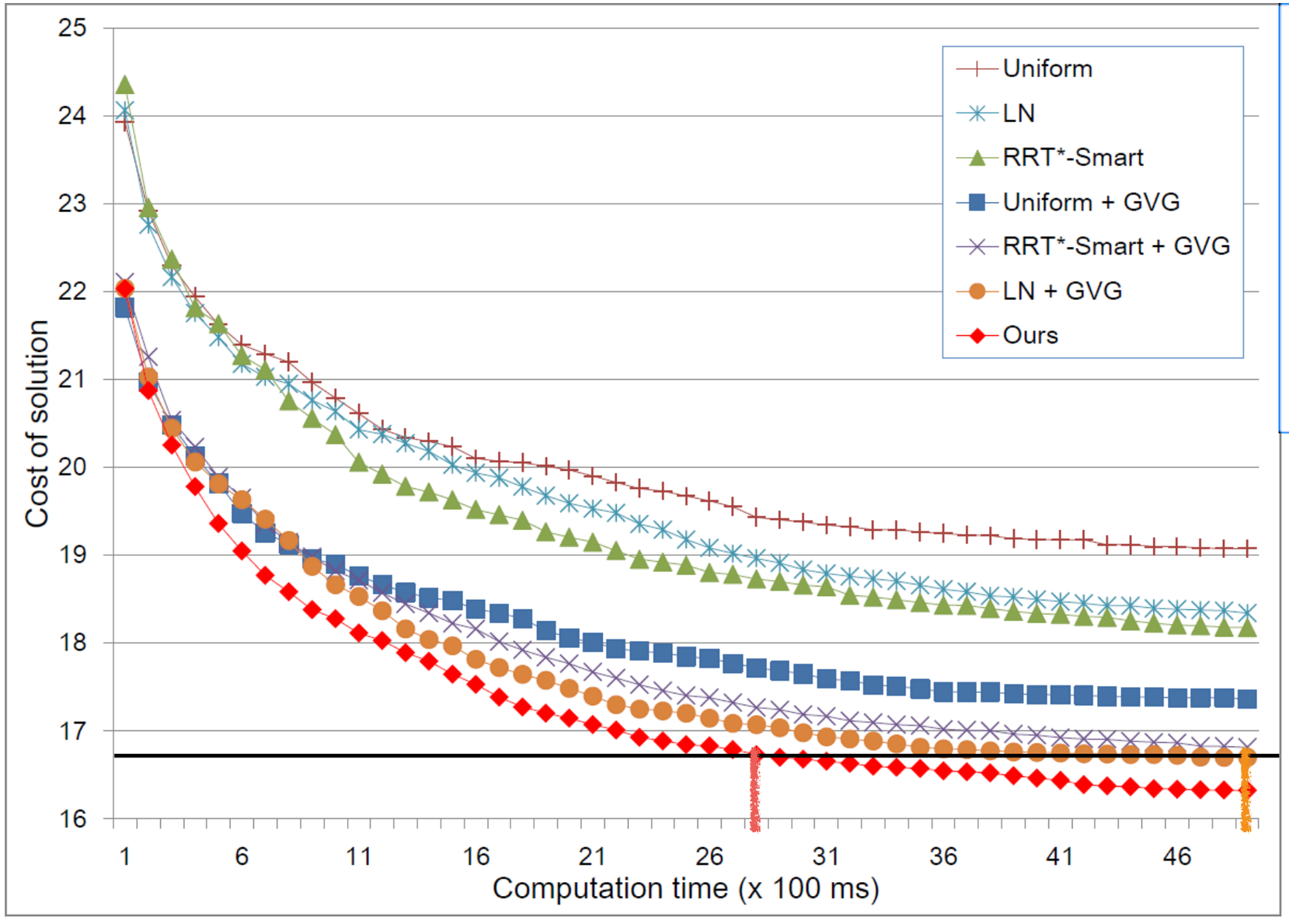


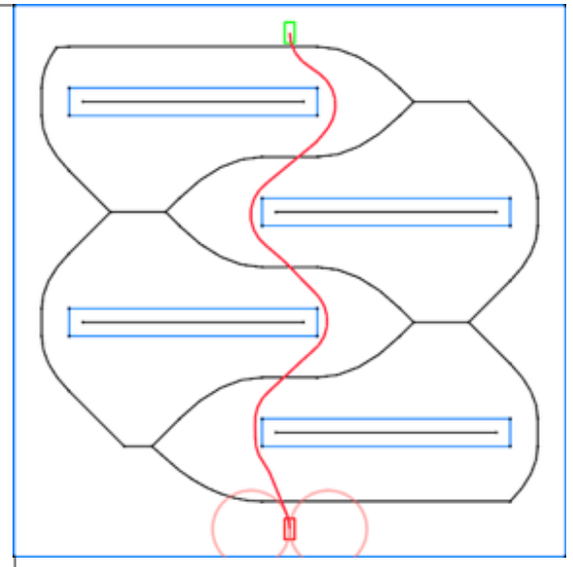

} ***Combined algorithms***

# Results: 4 squares



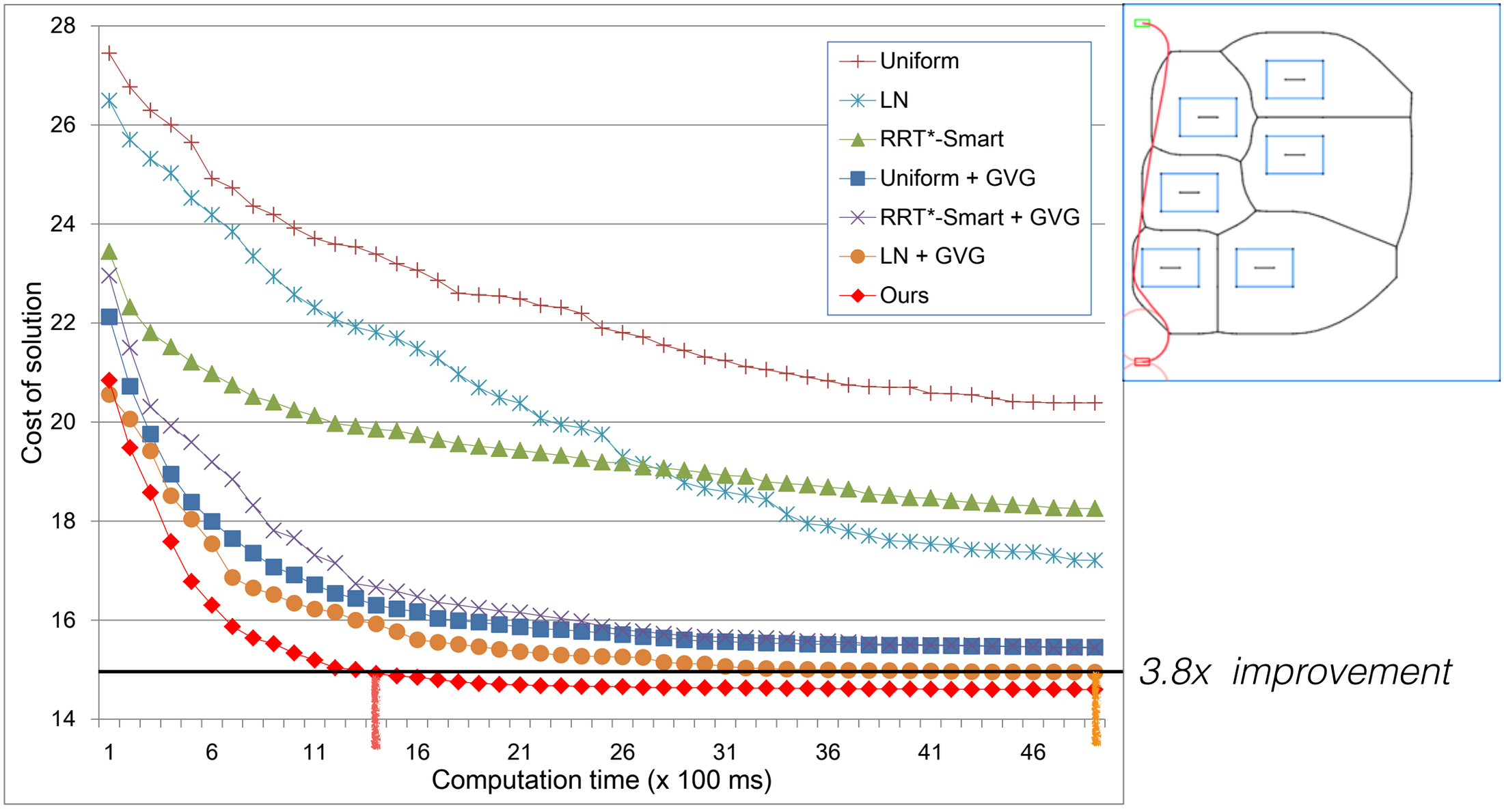
# Results: 4 squares



*1.8x improvement*

# Results: 6 squares




*3.8x improvement*

# Conclusions

- Presented RRT* based sampling heuristic algorithm that:
  - Locally exploits solutions in different homotopy classes to achieve a better convergence speed toward optimal
- Future work
  - 3D workspace problem

# Thanks

- For more details:

**donghyuk.kim@kaist.ac.kr**

*http://sglab.kaist.ac.kr/CloudRRT/*