# Lecture 15: Planning II

Professor Katie Driggs-Campbell

March 21, 2024

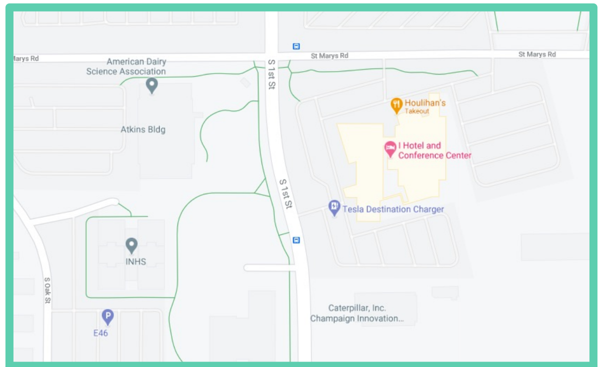
ECE484: Principles of Safe Autonomy

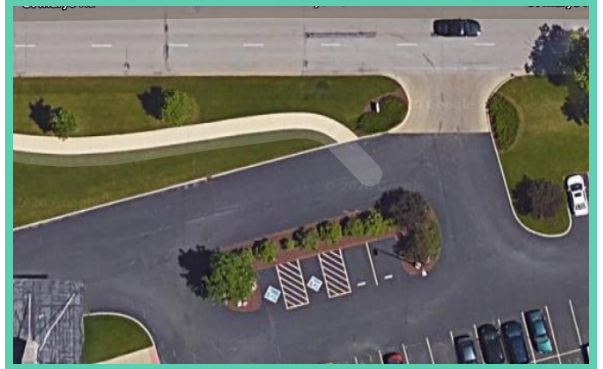
## Administrivia

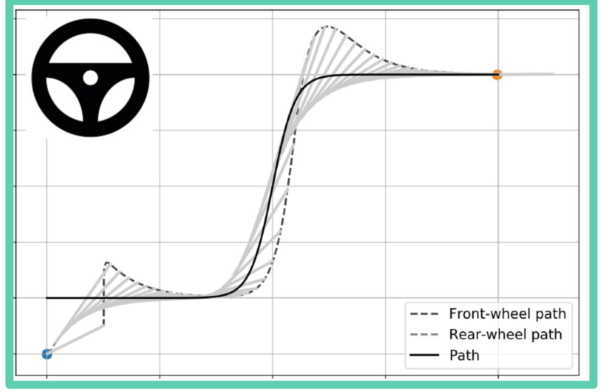
- Upcoming due dates:
  - HW3 and MP3 due Friday 3/22
  - **Mid-point check in on 3/29 or 4/05**
  - Final Presentations in class on 4/23 and 4/25
  - Final Video due 5/3
- Guest Lectures next week (3/26 and 3/28)
  - Attendance will be taken as that week's pop quiz: Attending both will give 100% for that week's pop quiz, attending one will give 50%
  - Tuesday will start at 10am – I'll have office hours at 9:30am
- Exam on 4/18 at 7pm
  - Email me about conflict exams
  - Will use testing center for DRES accommodations

# Typical planning and control modules

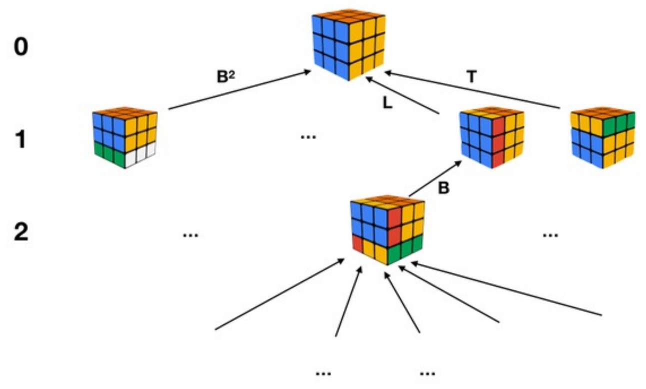
- Global navigation and planner
  - Find paths from source to destination with static obstacles
  - Algorithms: Graph search, Dijkstra, Sampling-based planning
  - Time scale: Minutes
  - Look ahead: Destination
  - Output: reference center line, semantic commands
- Local planner
  - Dynamically feasible trajectory generation
  - Dynamic planning w.r.t. obstacles
  - Time scales: 10 Hz
  - Look ahead: Seconds
  - Output: Waypoints, high-level actions, directions / velocities
- Controller
  - Waypoint follower using steering, throttle
  - Algorithms: PID control, MPC, Lyapunov-based controller
  - Lateral/longitudinal control
  - Time scale: 100 Hz
  - Look ahead: current state
  - Output: low-level control actions


American Dairy Science Association
Atkins Bldg
St Marys Rd
S 1st St
Houlihan's
Hotel and Conference Center
Tesla Destination Charger
INHS
E46
Caterpillar, Inc. Champaign Innovation...





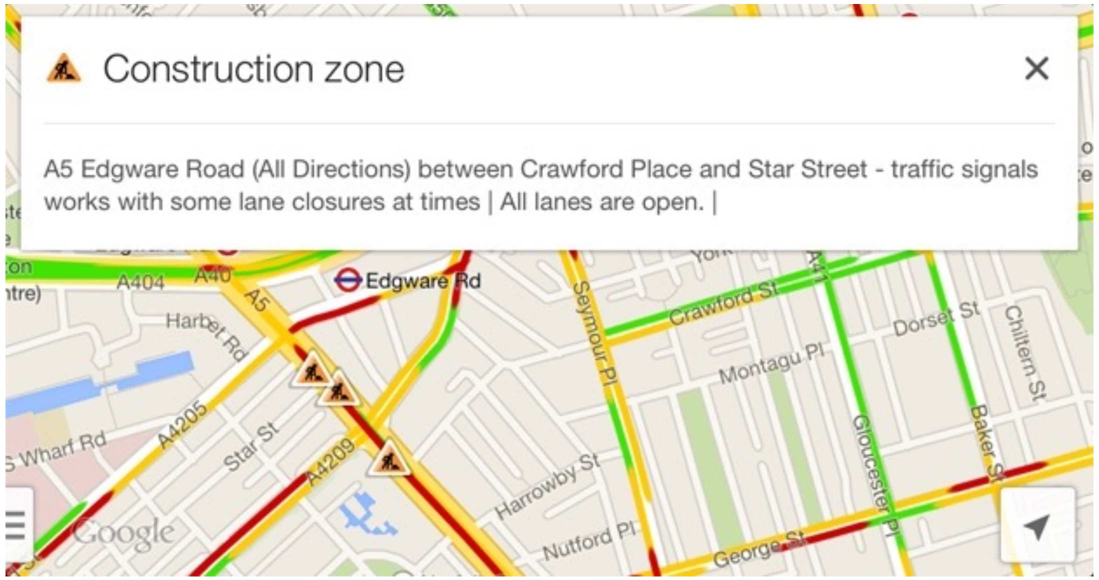
# Examples



The number of states or vertices can be large!

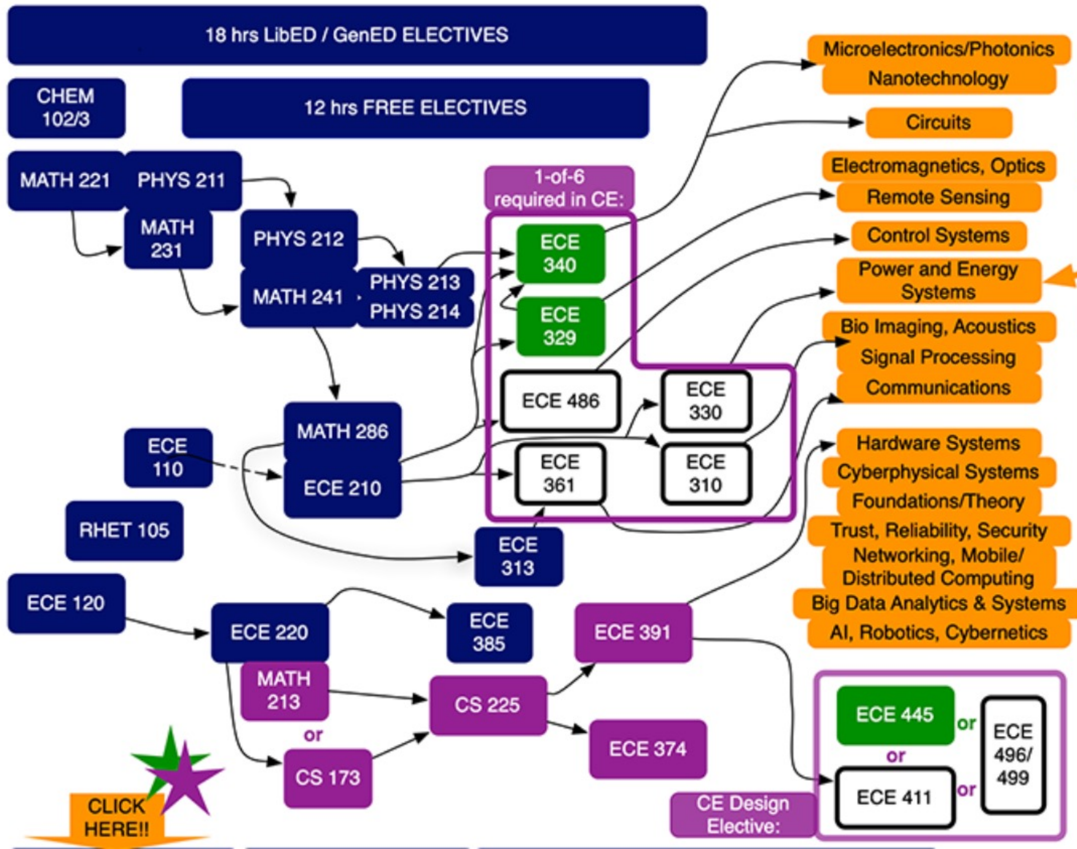
Rubik's cube num states: 43,252,003,274,489,856,000

Many paths and weights are not often known upfront!

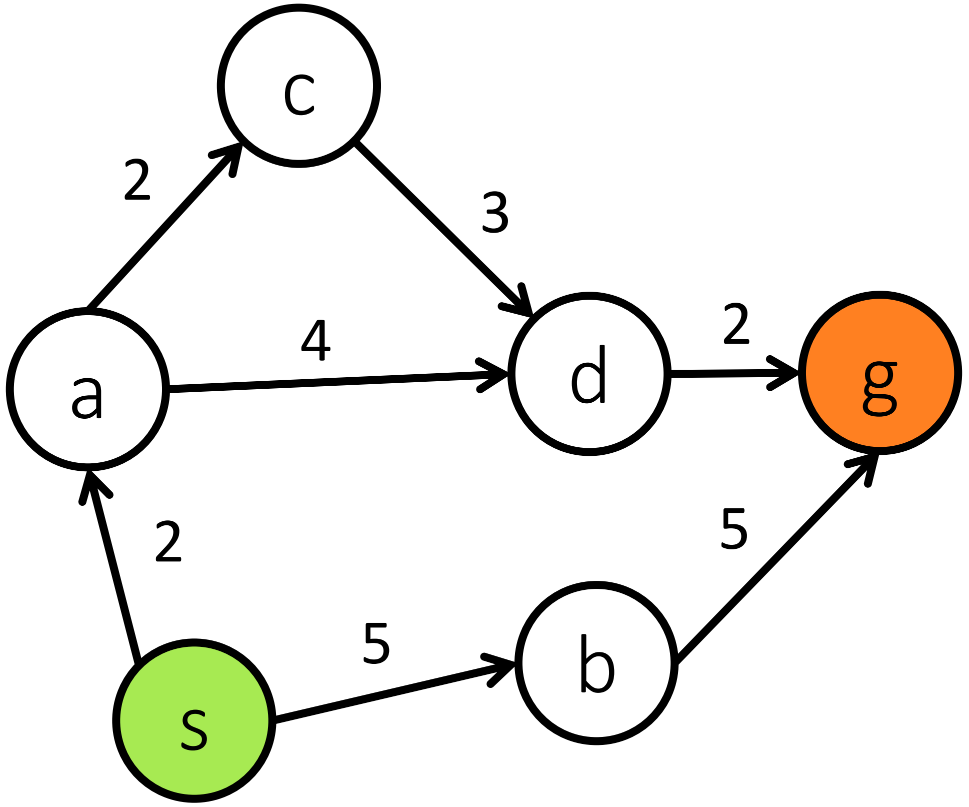


Construction zone

A5 Edgware Road (All Directions) between Crawford Place and Star Street - traffic signals works with some lane closures at times | All lanes are open. |



## Example: Find the minimal path from s to g



# Today's Plan

- Finish up graph search methods:
  - A and A* Search
- Sampling-based motion planning
  - Probabilistic Roadmaps
  - RRTs

# Informed Search: A Search

- UCS is optimal, but may wander around a lot before finding the goal
- Greedy is not optimal, but can be efficient, as it is heavily biased towards moving towards the goal
- A new idea:
  - Keep track of both the cost of the partial path to get to a vertex $g(v)$ and the heuristic function estimating the cost to reach the goal from a vertex $h(v)$
  - Choose a "ranking" function to be the sum of the two costs:
$$f(v) = g(v) + h(v)$$
  - $g(v)$: cost-to-arrive (from the start to $v$)
  - $h(v)$: cost-to-go estimate (from $v$ to the goal)
  - $f(v)$: estimated cost of the path (from the start to $v$ and then to the goal)

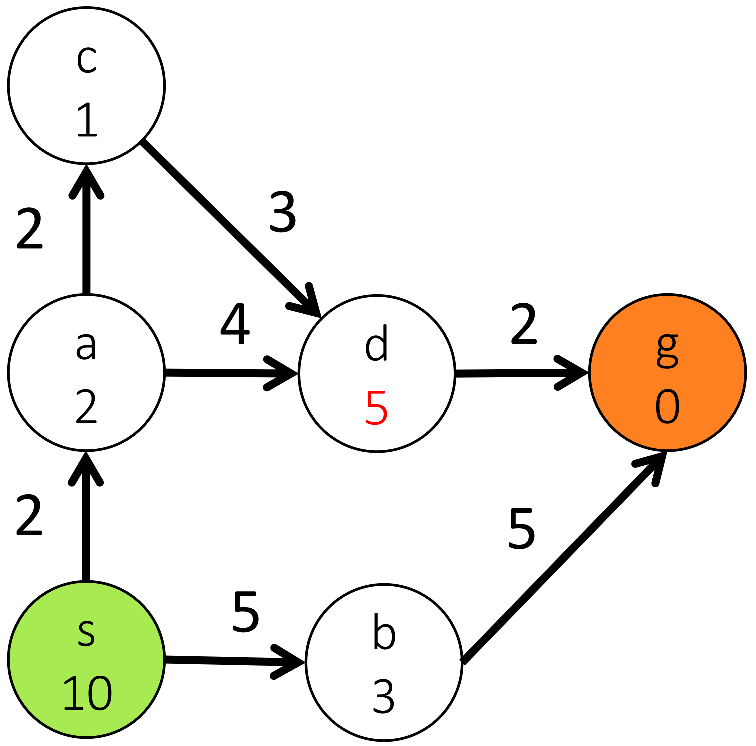
# A Search

```
$\boldsymbol{Q} \leftarrow \langle \boldsymbol{start} \rangle$                                   // initialize queue with start
while $\boldsymbol{Q} \neq \emptyset$:
    pick (and remove) the path $\boldsymbol{P}$ with the lowest estimated cost ($\boldsymbol{f}(\boldsymbol{P}) = \boldsymbol{g}(\boldsymbol{P}) + \boldsymbol{h}(\boldsymbol{head}(\boldsymbol{P}))$) from Q
    if $\boldsymbol{head}(\boldsymbol{P}) = \boldsymbol{x_{goal}}$ then return $\boldsymbol{P}$        // Reached the goal
    for each vertex $\boldsymbol{v}$ such that $(\boldsymbol{head}(\boldsymbol{P}), \boldsymbol{v}) \in \boldsymbol{E}$, do   // for all neighbors
        add $\langle v, P \rangle$ to $Q$                             // Add expanded paths
Return FAILURE                                         // nothing left to consider
```

# Example of A Search

Q:

| Path | g | h | f | |
|---|---|---|---|---|
| $\langle s\rangle$ | 0 | 10 | 10 | |
| $\langle a,s\rangle$ | 2 | 2 | 4 | ✓ |
| $\langle b,s\rangle$ | 5 | 3 | 8 | |
| $\langle c,a,s\rangle$ | 4 | 1 | 5 | ✓ |
| $\langle d,a,s\rangle$ | 6 | 5 | 11 | |
| $\langle d,c,a,s\rangle$ | 7 | 5 | 12 | |
| $\langle g,b,s\rangle$ | 10 | 0 | 10 | ✓ |



# Remarks on A search

- A search is similar to UCS, with a bias induced by the heuristic $h$
  - If *h = 0*, A is equivalent to UCS
- A search is complete, but is *not optimal*
- Let's try to make it optimal with A* Search
- Choose an *admissible heuristic*: $h(v) \leq h^*(v)$
  - $h^*(v)$ is the "optimal" heuristic (i.e., perfect cost to go)
  - An admissible $h(v)$ should be at most $h^*(v)$
  - Guaranteed to find optimal path

# Proof of optimality of A*

- Let $w^*$ be the cost of the optimal path
- Suppose for the sake of contradiction, that A* returns $P$ with $w(P) > w^*$
- Find the first *unexpanded node* $n$ on the optimal path $P^*$
  - $f(n) > w(P)$, otherwise $n$ would have been expanded
- $f(n) = g(n) + h(n)$

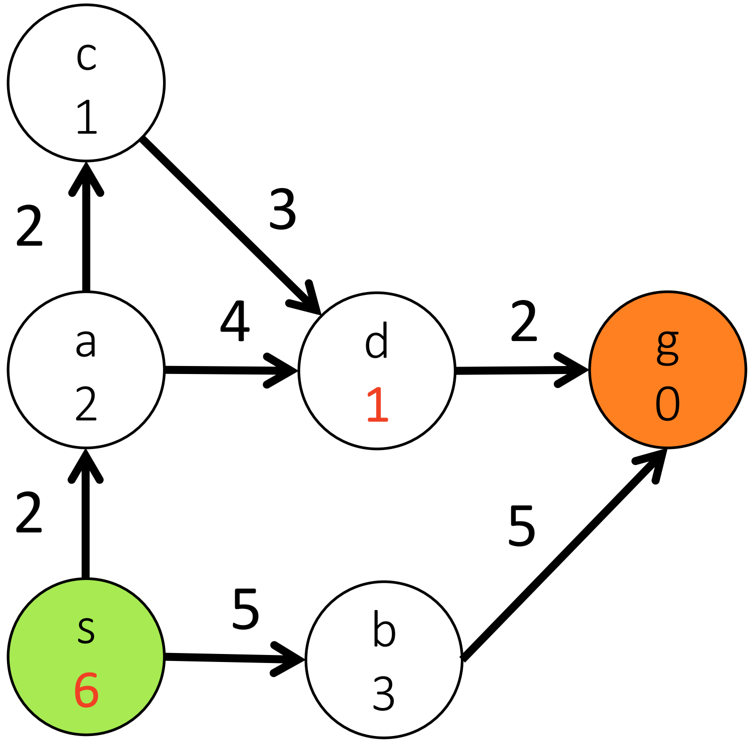
$$\begin{aligned}
&= g^*(n) + h(n) && \text{[since } n \text{ is on the optimal path]} \\
&\le g^*(n) + h^*(n) && \text{[since } h \text{ is admissible]} \\
&= f^*(n) = w^* && \text{[by def. of } f\text{, and since } w^* \text{ is the cost of the optimal path]}
\end{aligned}$$

- Hence, $w^* \geq f(n) = w(P)$, which is a contradiction

# Example of A$^*$ Search

Q:

| Path | g | h | f | |
|---|---|---|---|---|
| $\langle s \rangle$ | 0 | 6 | 6 | |
| $\langle a, s \rangle$ | 2 | 2 | 4 | |
| $\langle c, a, s \rangle$ | 4 | 1 | 5 | |
| $\langle d, a, s \rangle$ | 6 | 1 | 7 | |
| $\langle d, c, a, s \rangle$ | 7 | 1 | 8 | |
| $\langle g, d, a, s \rangle$ | 8 | 1 | 9 | ✓ |



## Admissible heuristics

- How to find an admissible heuristic?
  - i.e., a heuristic that never overestimates the cost-to-go
- Examples of admissible heuristics
  - $h(v) = 0$: this always works! However, it is not very useful,  A* = UCS
  - $h(v) = d(v, g)$: when the vertices of the graphs are physical locations
  - $h(v) = ||v - g||_p$: when the vertices of the graph are points in a normed vector space
- Generally: Choose $h$ as the optimal cost-to-go function for a *relaxed problem*, that is easy to compute
  - A relaxed problem ignores some of the constraints in the original problem

# Admissible heuristics for the 8-puzzle

Initial:

| | | |
|---|---|---|
| 1 | | 5 |
| 2 | 6 | 3 |
| 7 | 4 | 8 |

Goal:

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | |

Which of the following are admissible heuristics?

| | |
|---|---|
| • h = 0 | YES, always good |
| • h = 1 | Not valid in goal state |
| • h = number of tiles in the wrong position | YES, “teleport” each tile to the goal in one move |
| • h = sum of (Manhattan) distance between tiles and their goal position | YES, move each tile to the goal ignoring other tiles |

# A *partial* order of admissible heuristics

- Some admissible heuristics are better than others
  - $h = 0$ is an admissible heuristic, but is not very useful
  - $h = h^*$ is also an admissible heuristic (because it is the "best" possible one)
- Partial order
  - We say that $h_1$ dominates $h_2$ if $h_1(v) \geq h_2(v)$ for all vertices $v$
  - $h^*$ dominates all admissible heuristics, and $0$ is dominated by all admissible heuristics
- Choosing the right heuristic
  - We want a heuristic that is as close to $h^*$ as possible
  - However, such a heuristic may be too complicated to compute

  → There is a tradeoff between complexity of computing $h$ and the complexity of the search

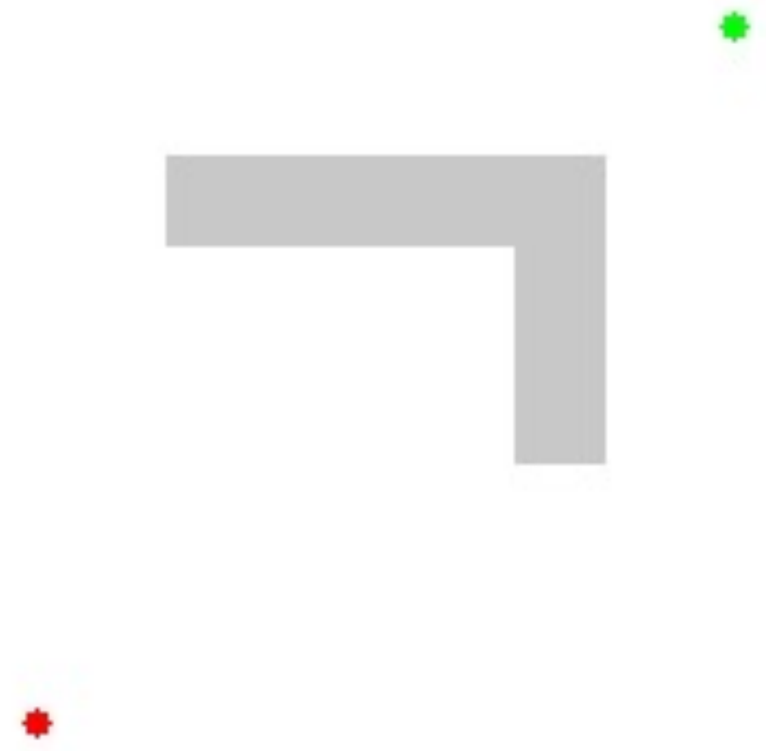
# Consistent Heuristics

- An additional useful property for A* heuristics is called consistency
  - A heuristic $h: X \to \mathbb{R}_{\geq 0}$ is said consistent if $\forall e_{uv} \in E$
    $$h(u) \leq w(e) + h(v)$$

  → a consistent heuristics satisfies a triangle inequality
- If $h$ is a consistent heuristic, then $f = g + h$ is non-decreasing along paths: $f(v) = g(v) + h(v) = g(u) + w(u, v) + h(v) \geq f(u)$
- Hence, the values of $f$ on the sequence of nodes expanded by A* is non-decreasing: the first path found to a node is also the optimal path $\Rightarrow$ no need to compare costs!

# A* recap

- A* algorithm is an informed search technique that combines cost-to-arrive $g(v)$ and a heuristic function $h(v)$ for cost-to-go to find shortest path
- Heuristic function must be *admissible:* $h(v) \leq h^*(v)$
  - Never over-estimates actual cost to go
  - Are all $h(v)$ values needed?
  - What if $h$ is not admissible?
  - How to find heuristic functions?



# A* variants are widely used in practice!

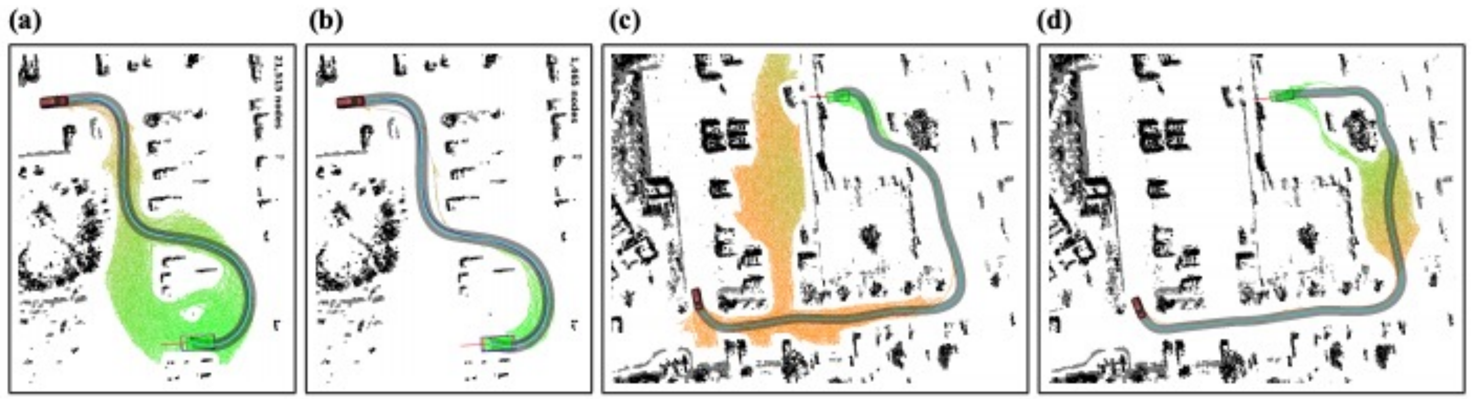
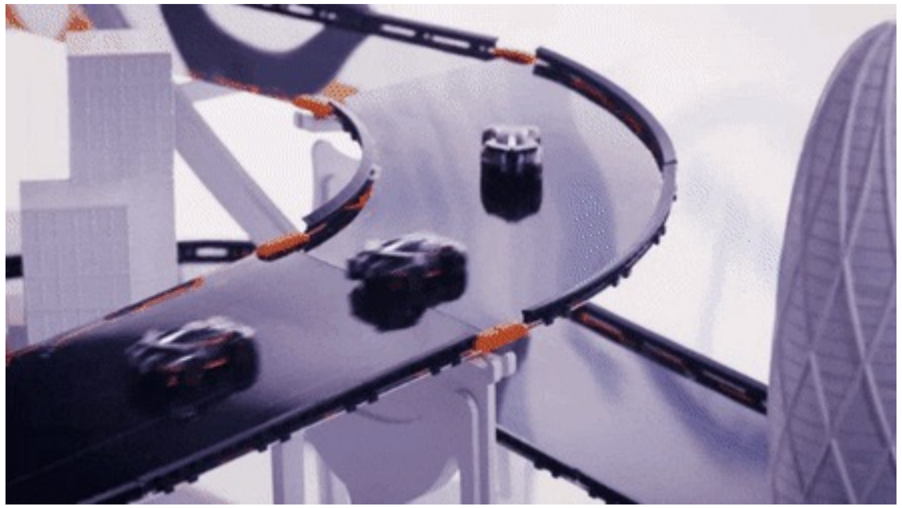


Figure 16: Hybrid-state A* heuristics. (a) Euclidean distance in 2-D expands 21, 515 nodes. (b) The non-holonomic-without-obstacles heuristic is a significant improvement, as it expands 1, 465 nodes, but as shown in (c), it can lead to wasteful exploration of dead-ends in more complex settings (68, 730 nodes). (d) This is rectified by using the latter in conjunction with the holonomic-with-obstacles heuristic (10, 588 nodes).



http://robots.stanford.edu/papers/junior08.pdf

## Today's Plan

- Finish up graph search methods:
  - A and A* Search
- Sampling-based motion planning
  - Probabilistic Roadmaps
  - RRTs

# Probabilistic RoadMaps (PRM)

Kavraki and Latombe, 1994

- **Idea:** build (offline) a graph (i.e., the roadmap) representing the "connectivity" of the environment
- Offline phase:
  - Sample $N$ points from $X_{free} = [0,1]^d \setminus X_{obs}$
  - Try to connect these points using a fast "local planner"
  - If connection is successful, add an edge between the points.
- At run time:
  - Connect the start and end goal to the closest nodes in the roadmap
  - Find a path on the roadmap
- First planner ever to demonstrate the ability to solve general planning problems in > 4-5 dimensions!

# Probabilistic completeness

**Definition.** A motion planning problem $P = (X_{free}, x_{init}, X_{goal})$ is ***robustly feasible*** if there exists some small δ>0 such that a solution remains a solution if obstacles are "dilated" by δ.

**Definition.** An algorithm ALG is ***probabilistically complete*** if

$$\lim_{N \to \infty} \Pr(ALG \text{ returns a solution to } P) = 1$$

for any *robustly feasible* motion planning problem defined by $P = \left(X_{free}, x_{init}, X_{goal}\right)$.

→ Applicable to motion planning problems with a robust solution

## Asymptotic optimality

Suppose we have a cost function $c$ that associates to each path $\sigma$ a non-negative cost $c(\sigma)$, e.g., $c(\sigma) = \int_\sigma \chi(s)\, ds$.

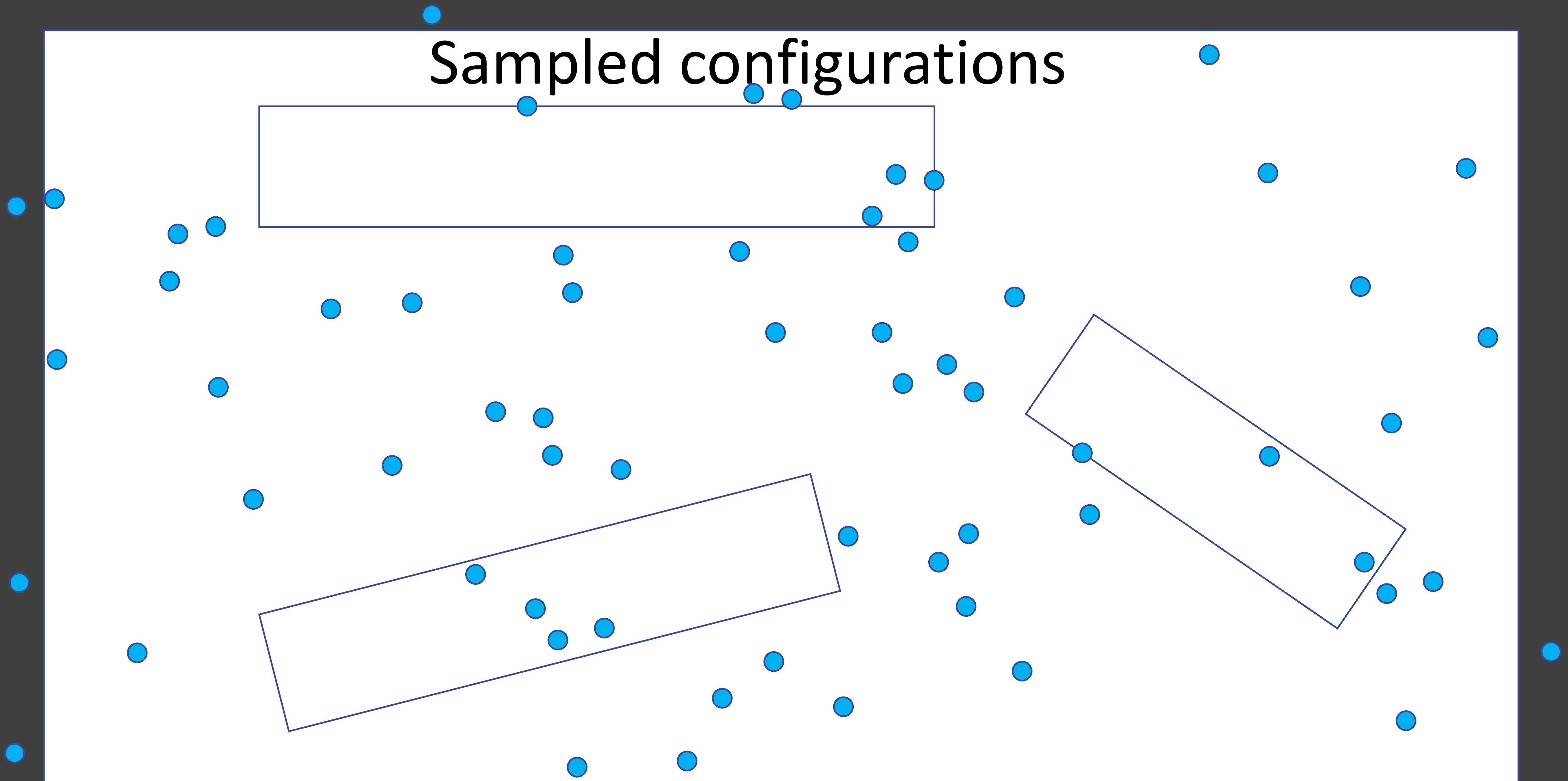
**Definition.** An algorithm ALG is *asymptotically optimal* if, for any motion planning problem $P = (X_{free}, x_{init}, X_{goal})$ and cost function $c$ that admit a robust optimal solution with finite cost $c^*$,

$$\boldsymbol{P}\left(\left\{\lim_{i\to\infty} Y_i^{ALG} = c^*\right\}\right) = 1$$

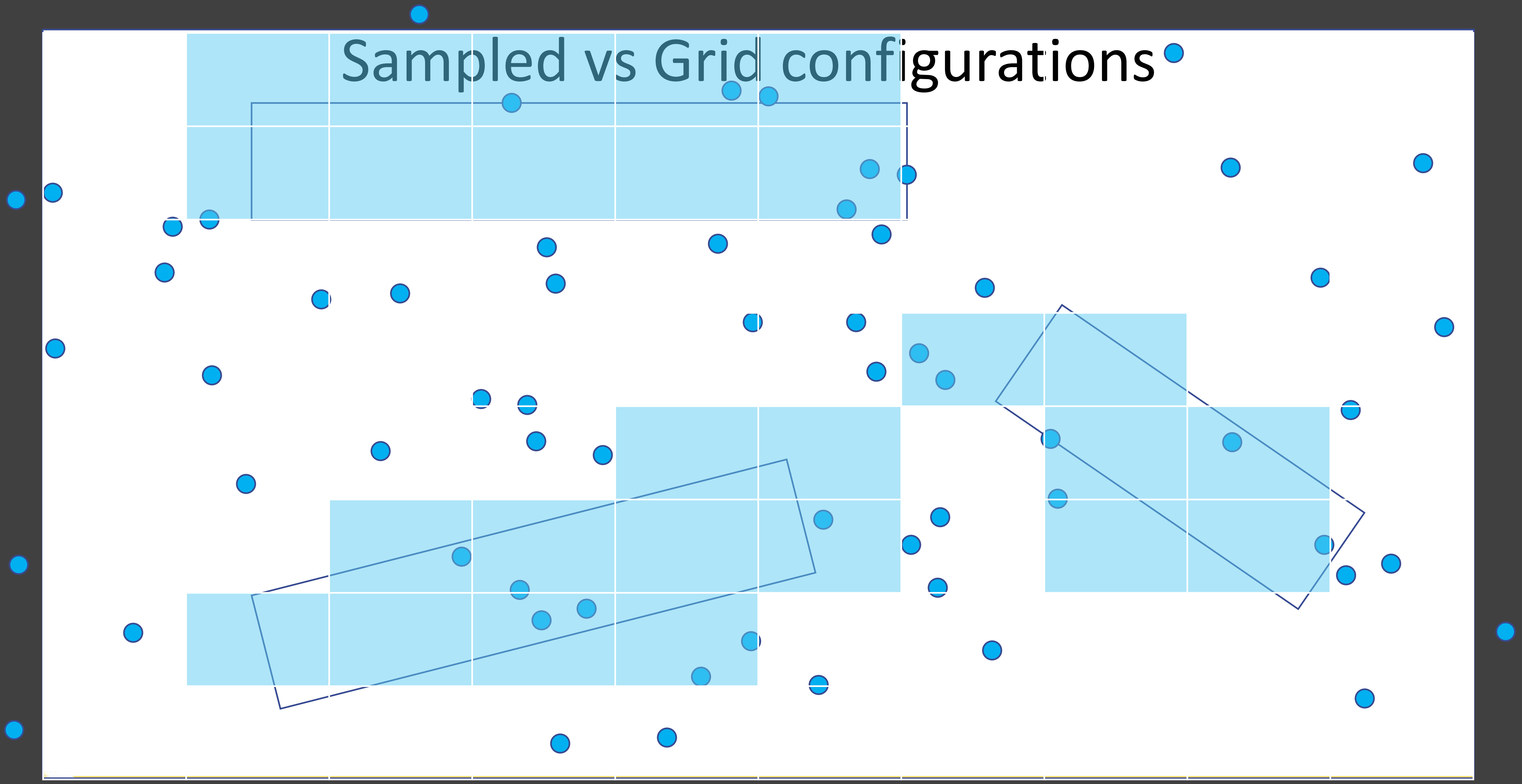
# Simple PRM construction: Step 1

```
PRM(N,k)
V ← $\{x_{init}\} \cup \{\text{SampleFree}i\}_{i=1,\ldots,N-1}$
E ← $\emptyset$
foreach $v \in V$ do
   U ← Near(V, v, k) // selects the k nearest neighbors of v in V
      foreach $u \in U$ do
            if CollisionFree(v, u) then
               E ← $E \cup \{(v, u), (u, v)\}$
return G = (V, E)
```

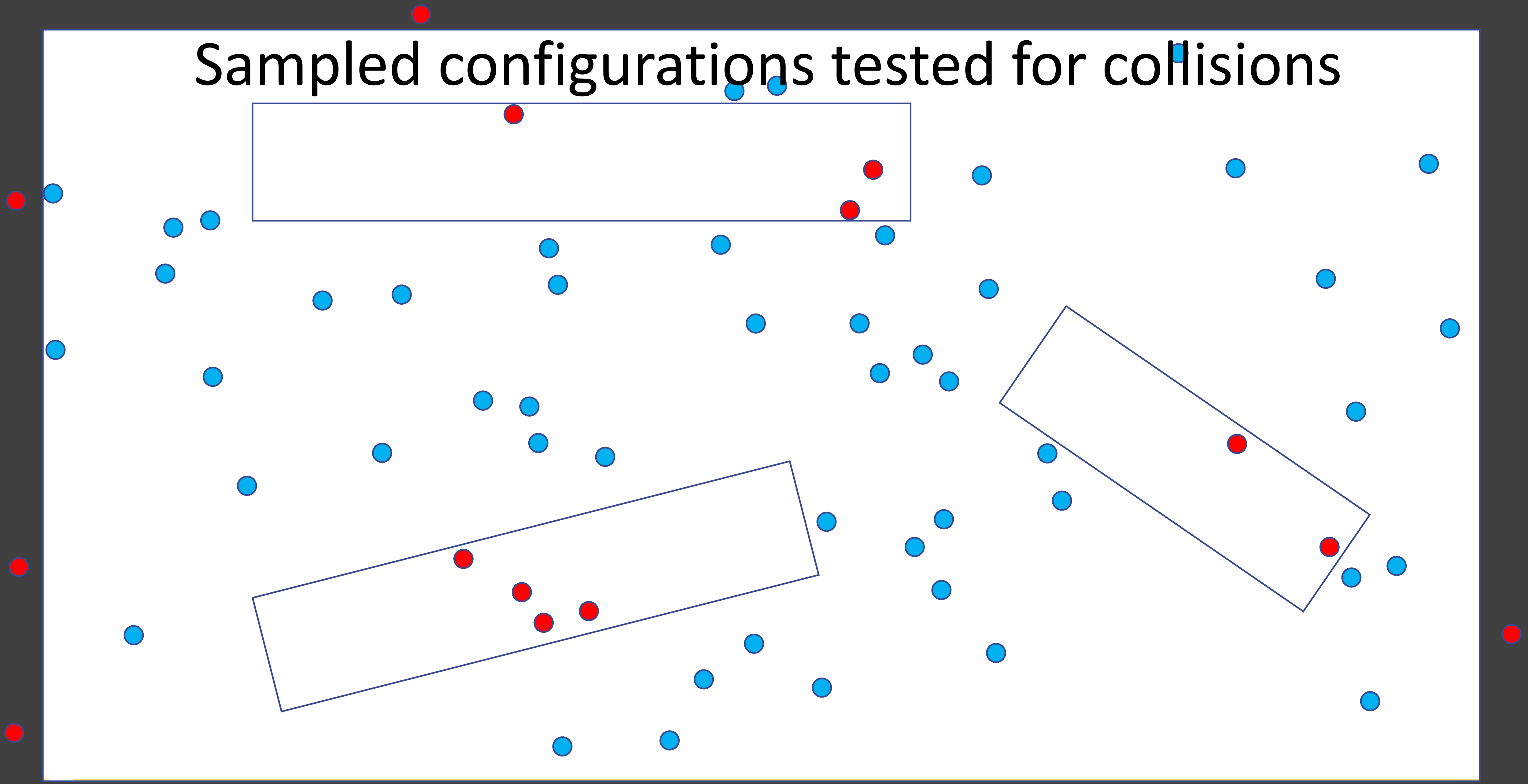
## Sampled configurations



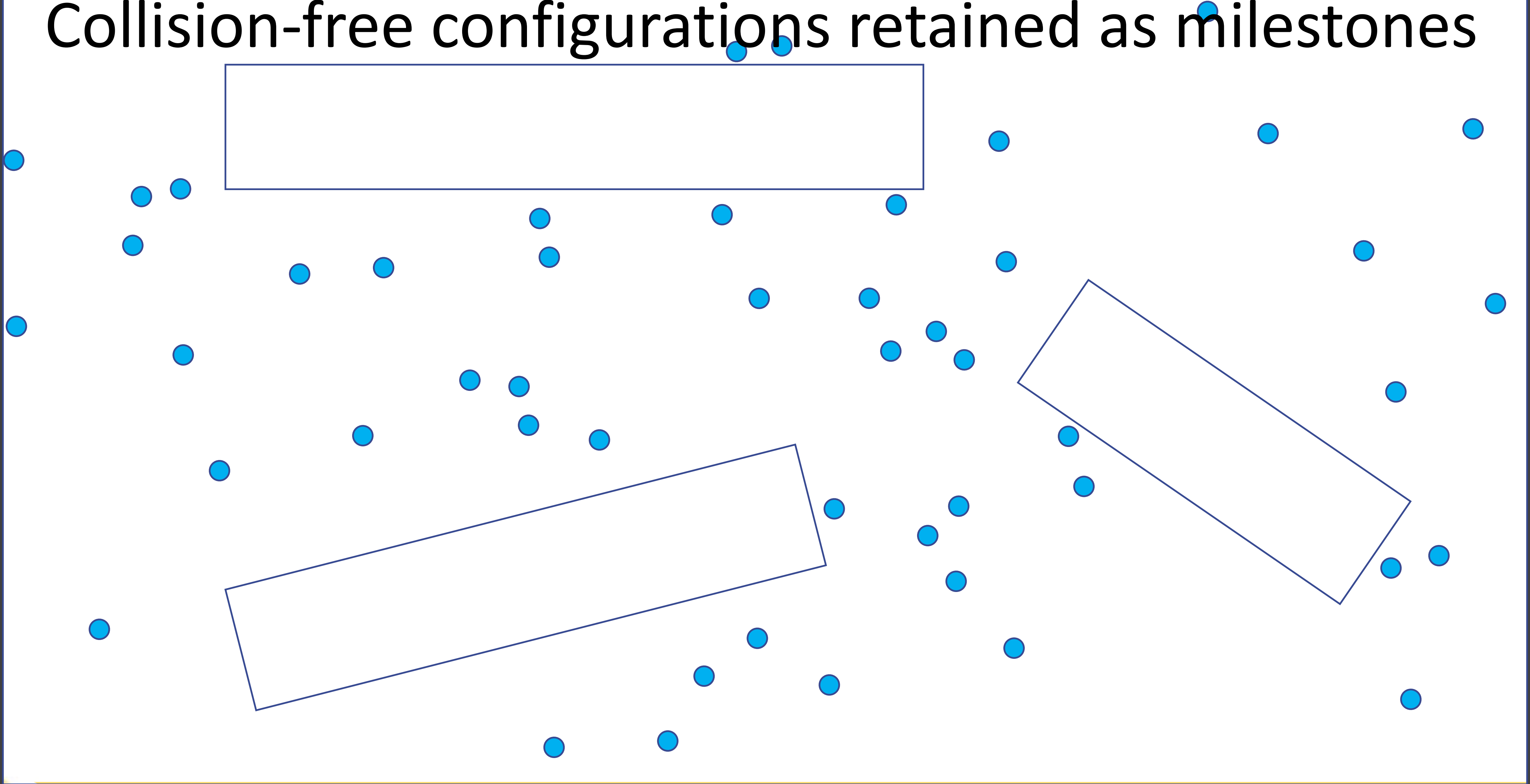
# Sampled vs Grid configurations



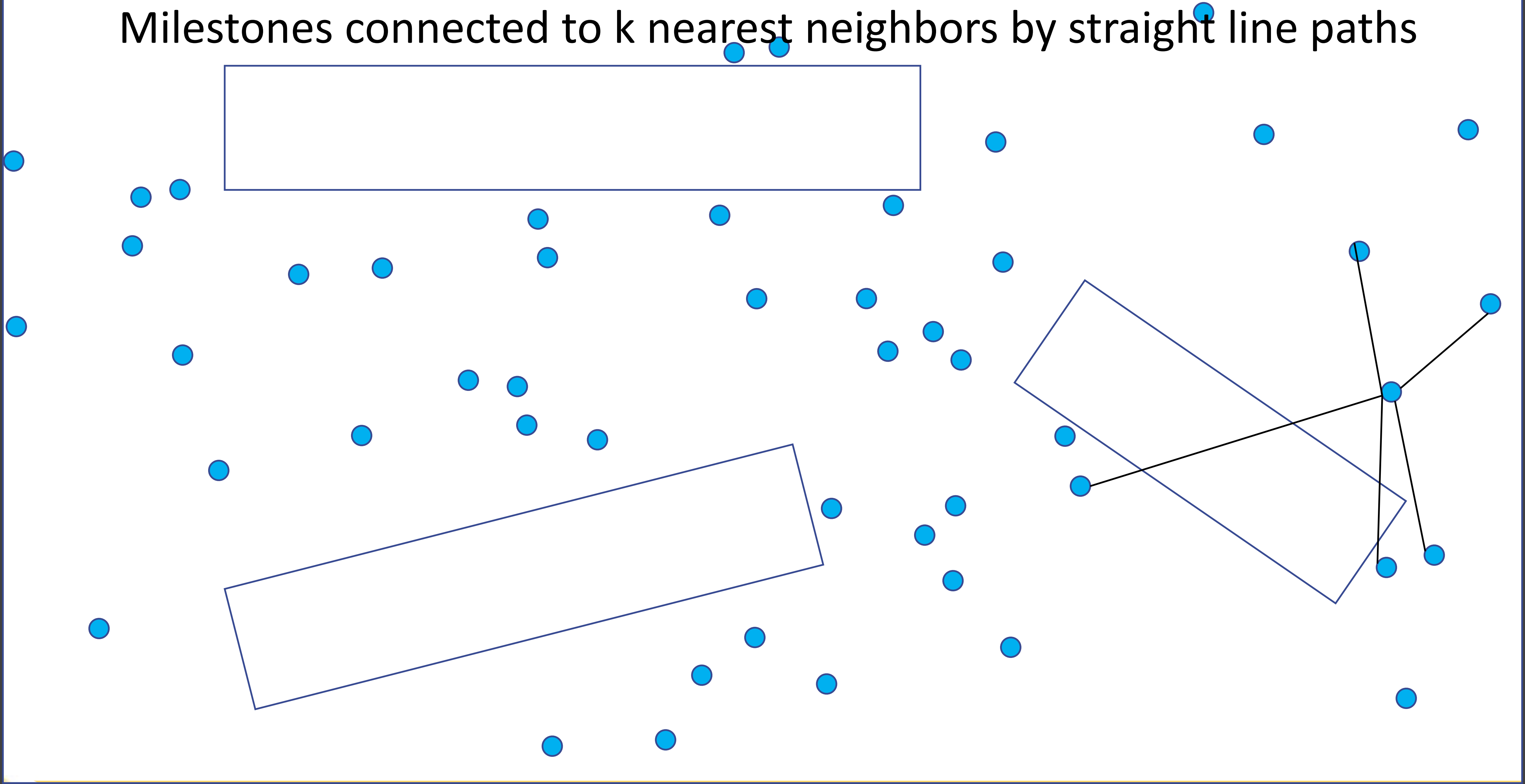
# Sampled configurations tested for collisions



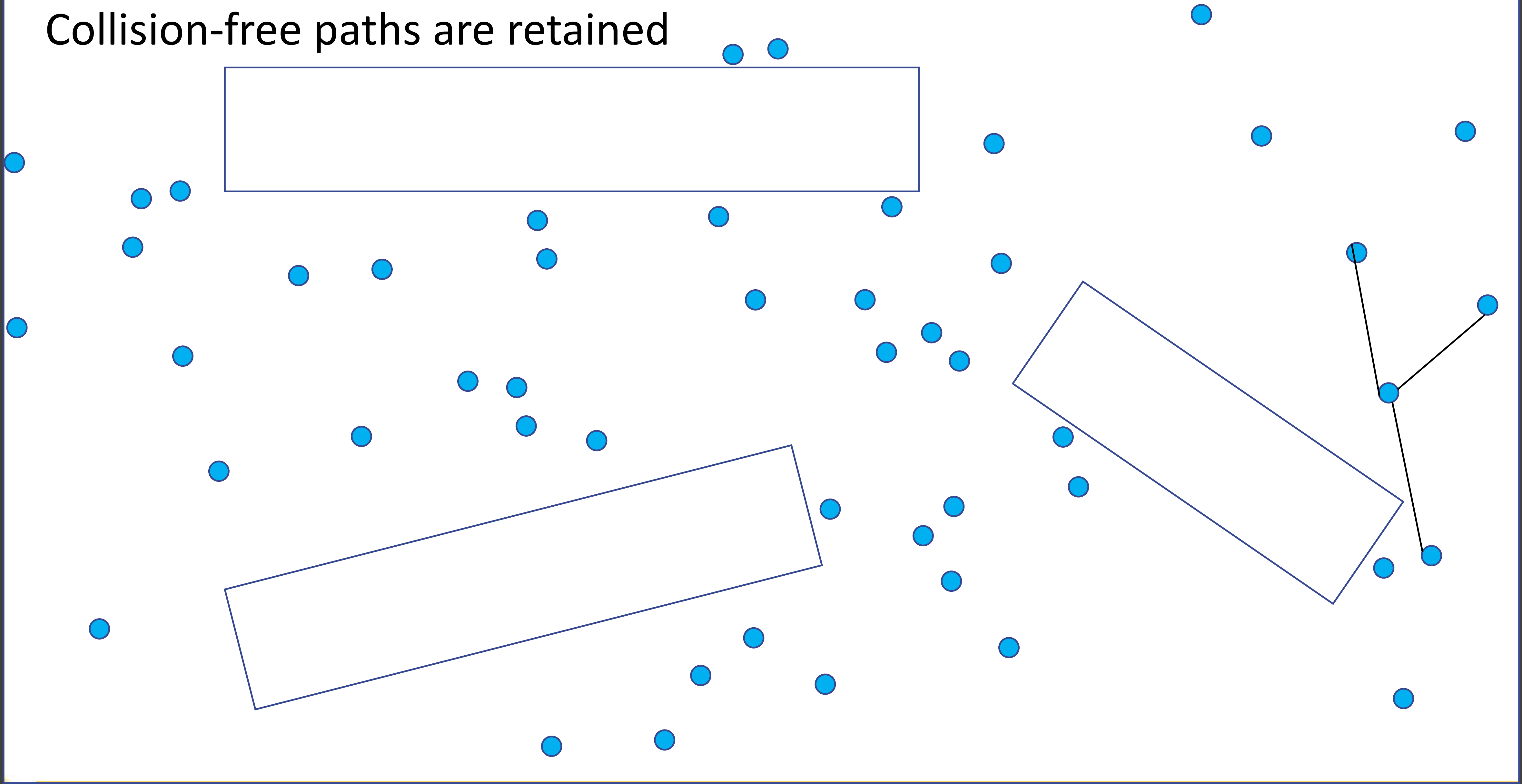
# Collision-free configurations retained as milestones



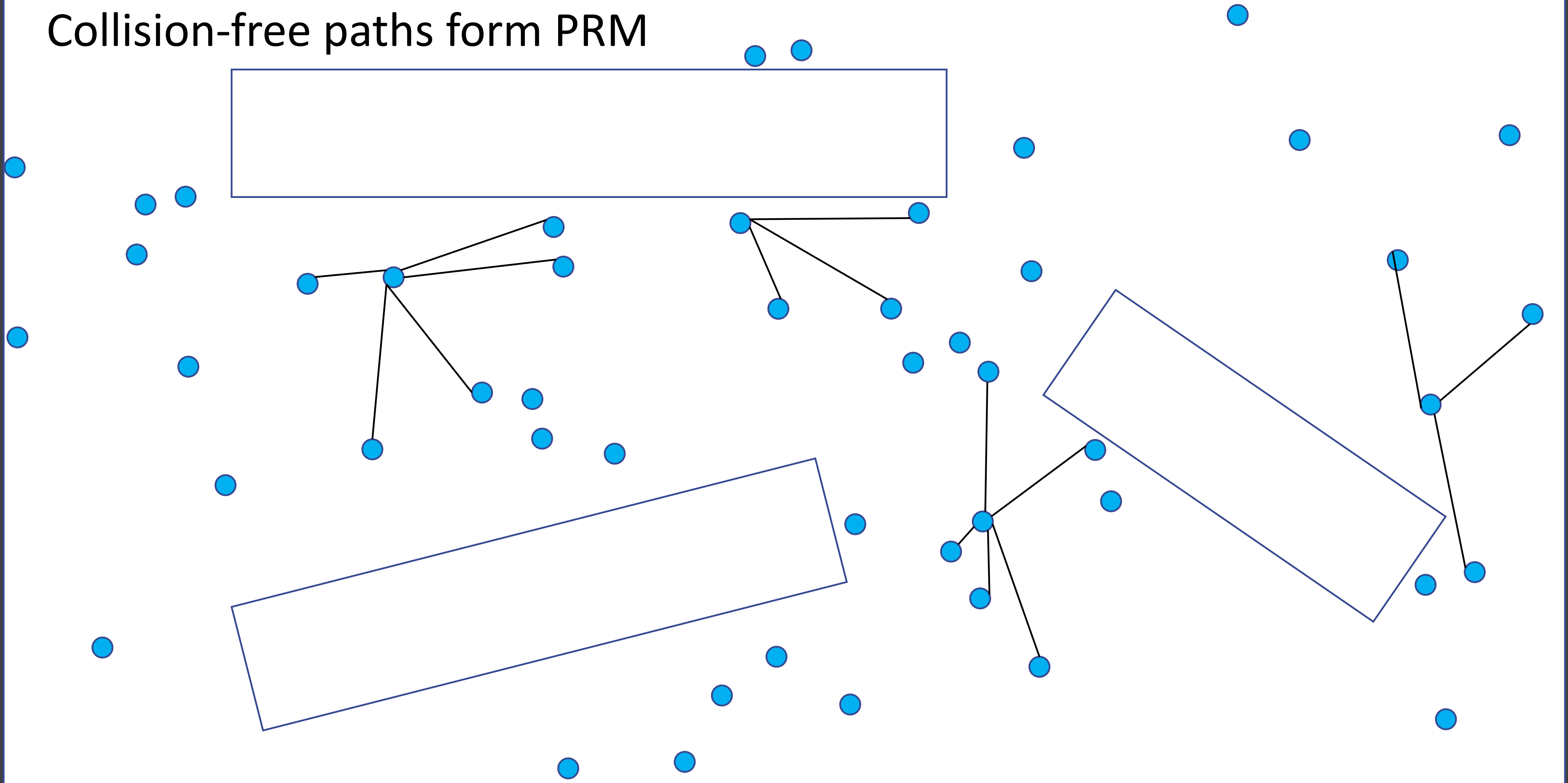
# Milestones connected to k nearest neighbors by straight line paths



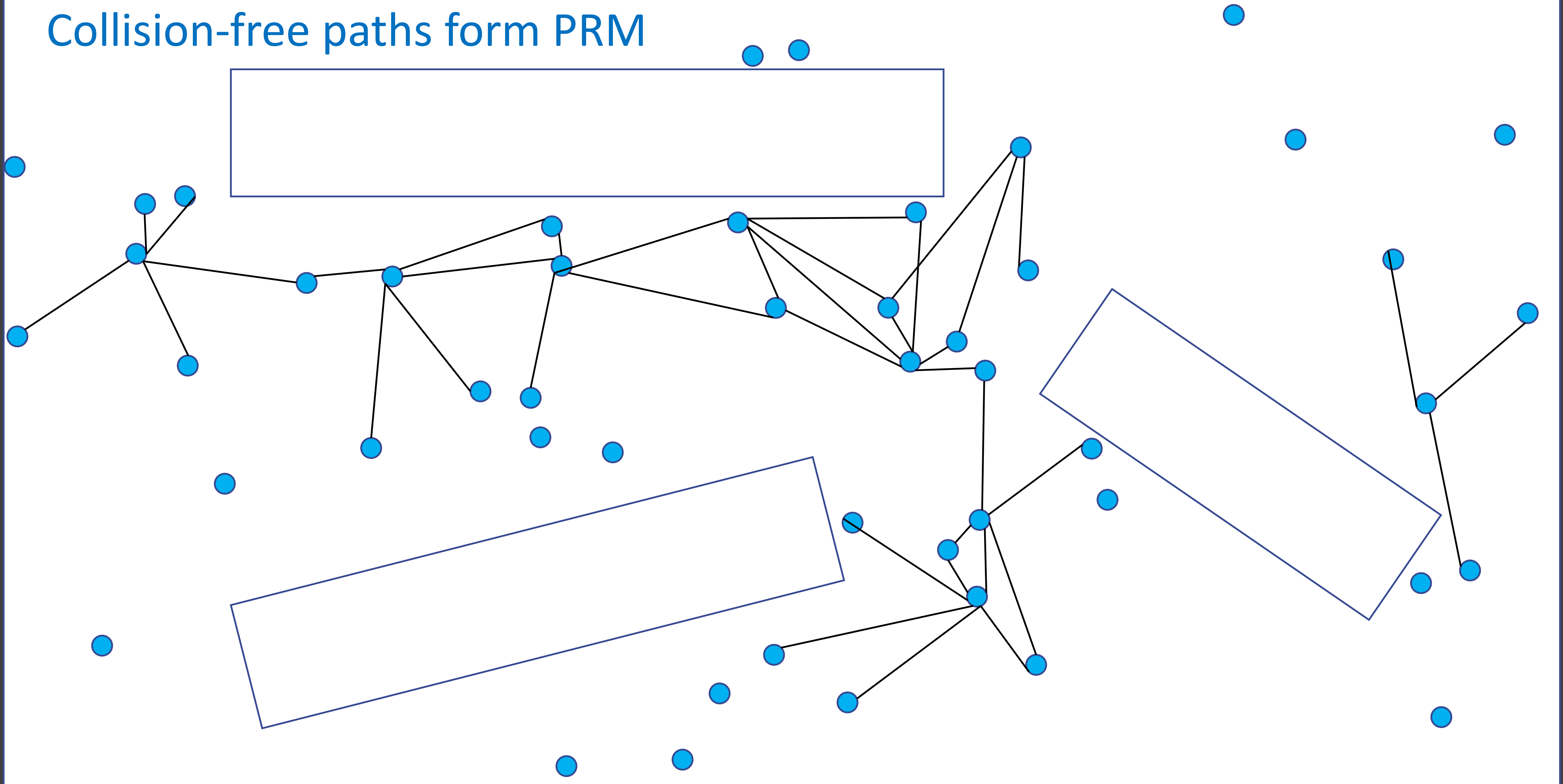
## Collision-free paths are retained



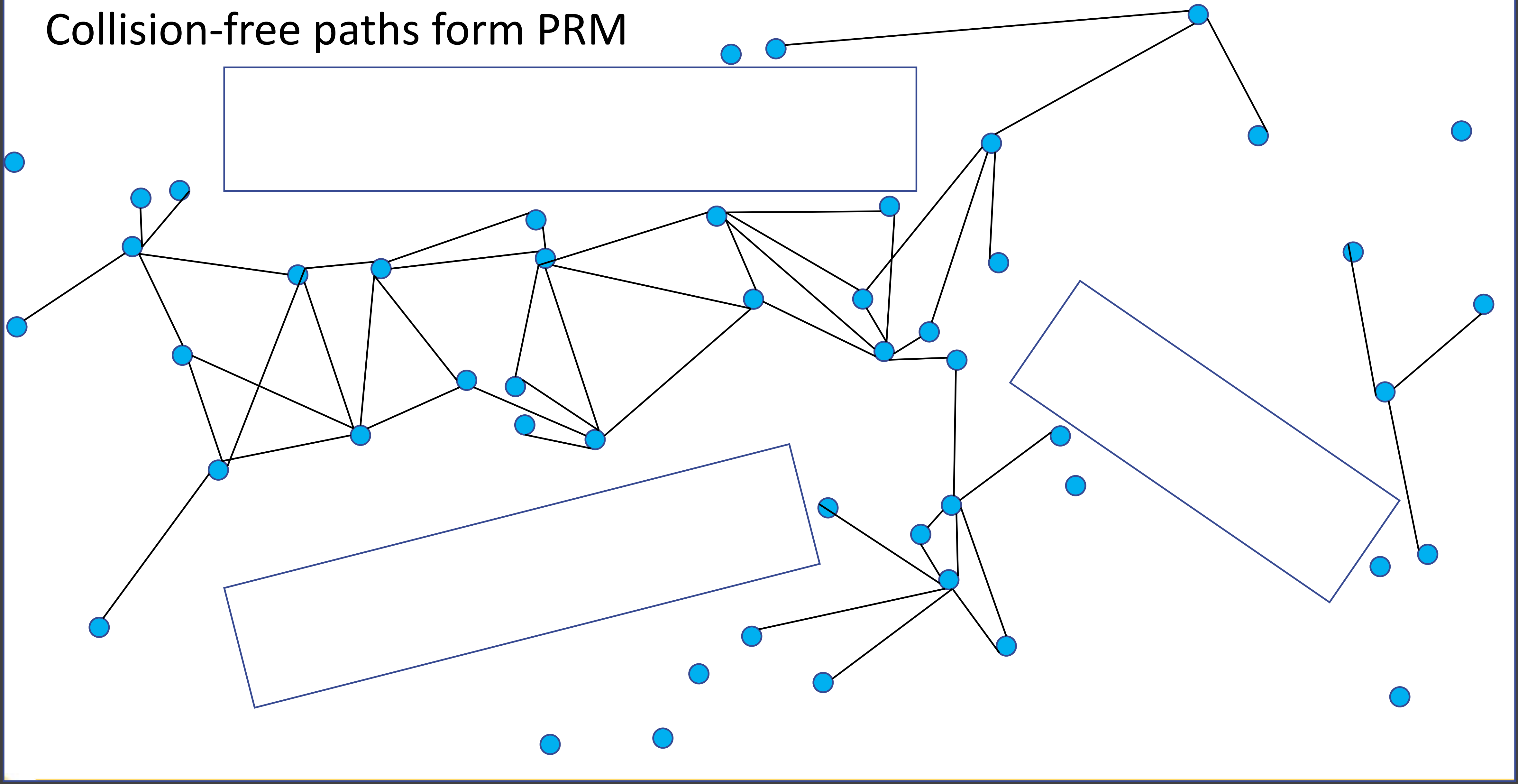
## Collision-free paths form PRM



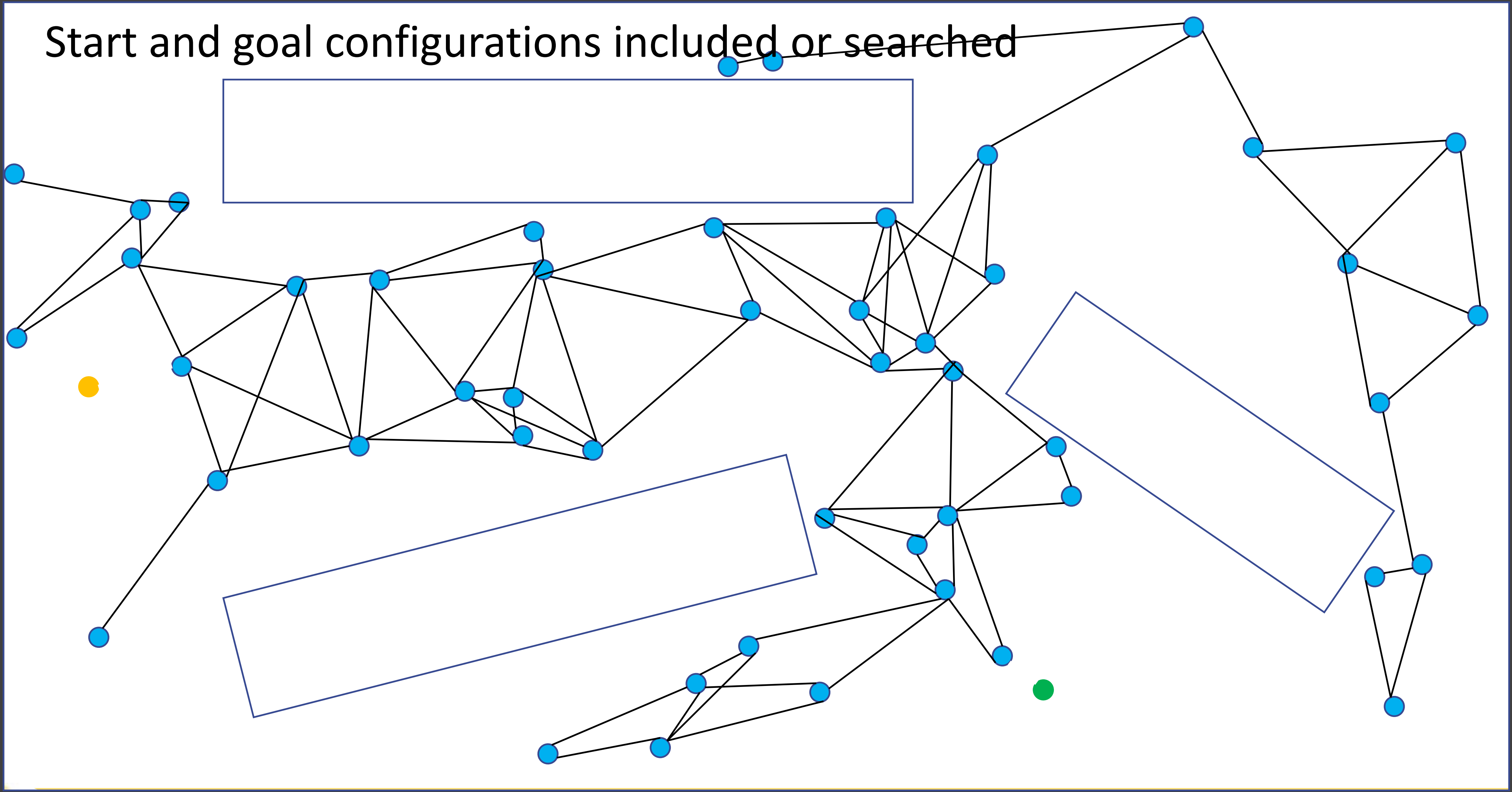
## Collision-free paths form PRM



## Collision-free paths form PRM



Start and goal configurations included or searched



## Remarks on PRM

- PRM has been shown to be ***probabilistically complete***
  - Moreover, the probability of success goes to 1 exponentially fast, if the environment satisfies certain “good visibility” conditions.
- However, there are some limitations:
  - NOT asymptotically optimal
  - Builds graph without particular focus on generating path
  - Required to solve 2-point collision detection problem

# Complexity of Sampling-based Algorithms

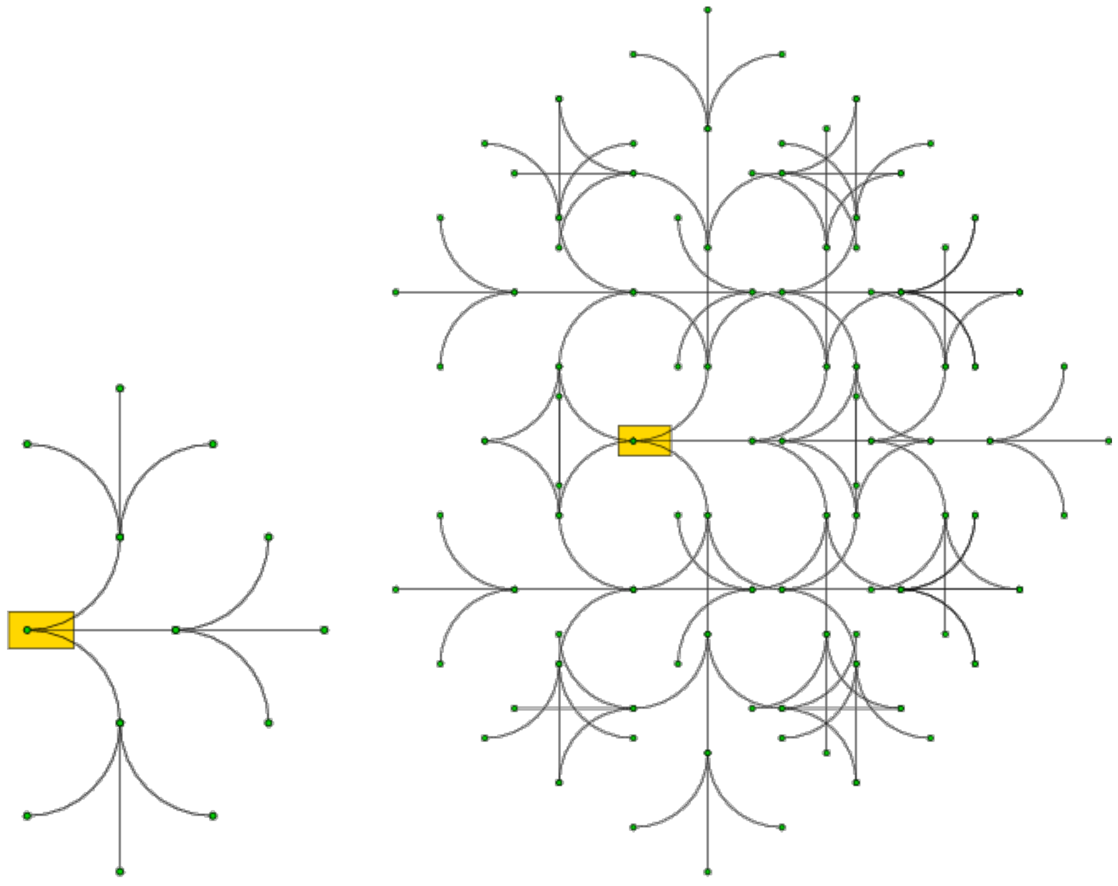
- How can we measure complexity for an algorithm that does not necessarily terminate?
  - Treat the number of samples as “the size of the input.” (Everything else stays the same)
  - Complexity per sample: how much work (time/memory) is needed to process one sample.
  - Useful for comparison of sampling-based algorithms. Not for deterministic, complete algorithms.
- Complexity of PRM for N samples $\Theta(N^2)$
- Practical complexity reduction tricks
  - **k-nearest neighbors:** connect to the k nearest neighbors. Complexity Θ(N log N).
  - **Bounded degree:** connect at most k neighbors among those within radius r.
  - **Variable radius:** change the connection radius r as a function of N. How?

# Rapidly Exploring Random Trees

LaValle and Kuffner, 1998

- **Idea:** build (online) a tree, exploring the region of the state space that can be reached from the initial condition
- At each step:
  - sample one point from $X_{free}$
  - Try to connect it to the closest vertex in the tree
- Very effective in practice

# Reachability Tree for Dubin's Car

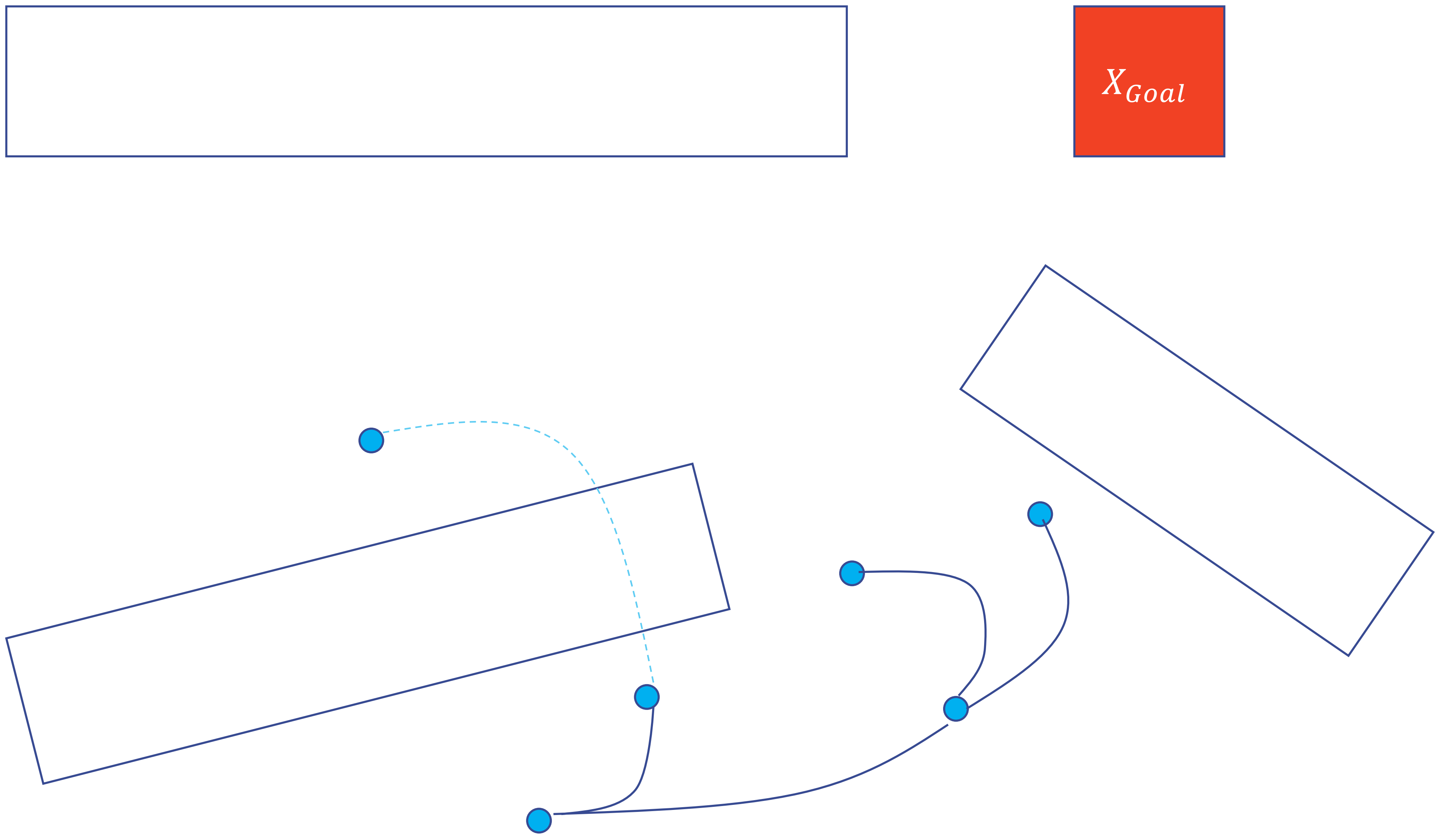


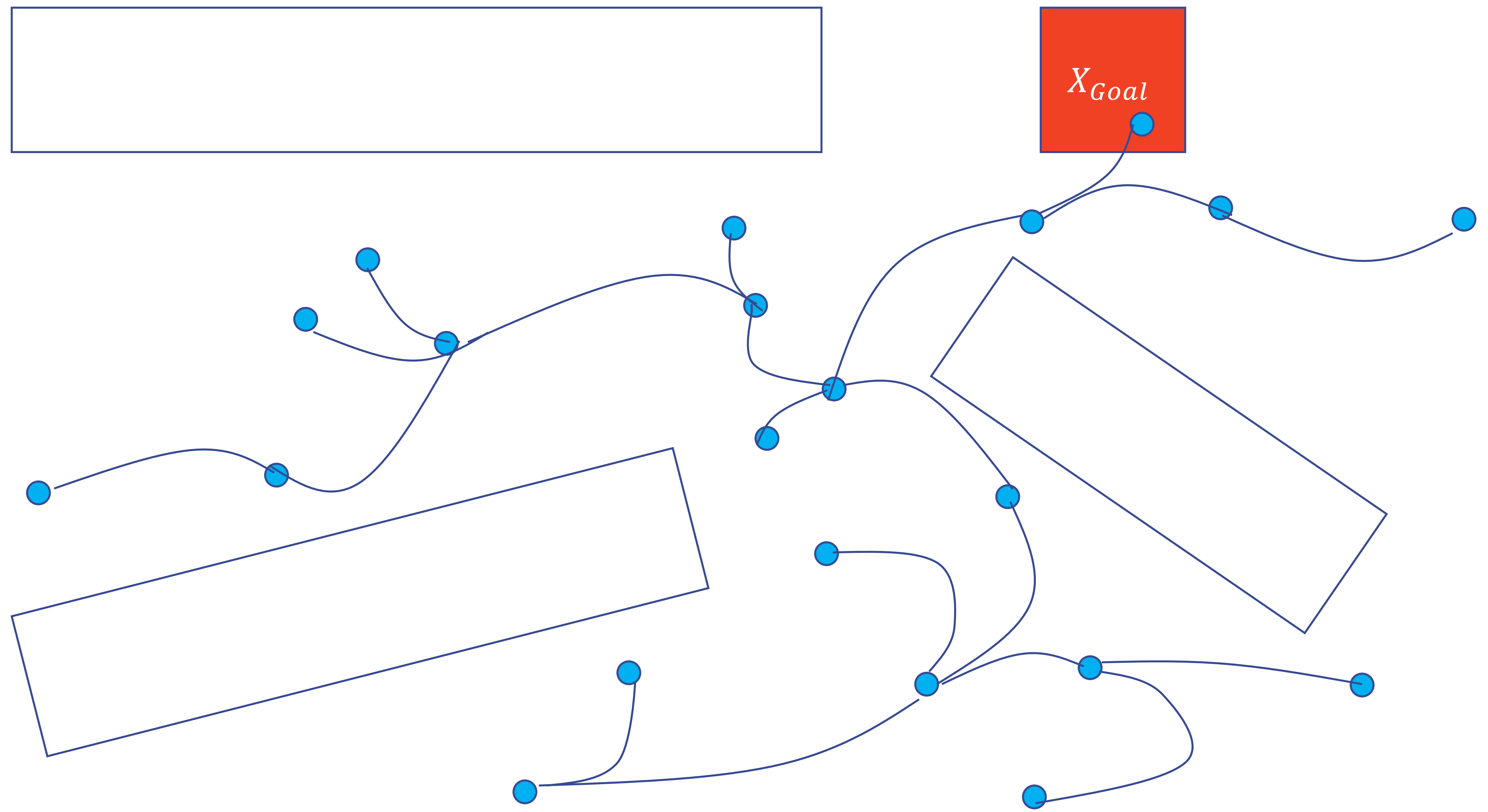
Credit: Steven LaValle, Planning Algorithms

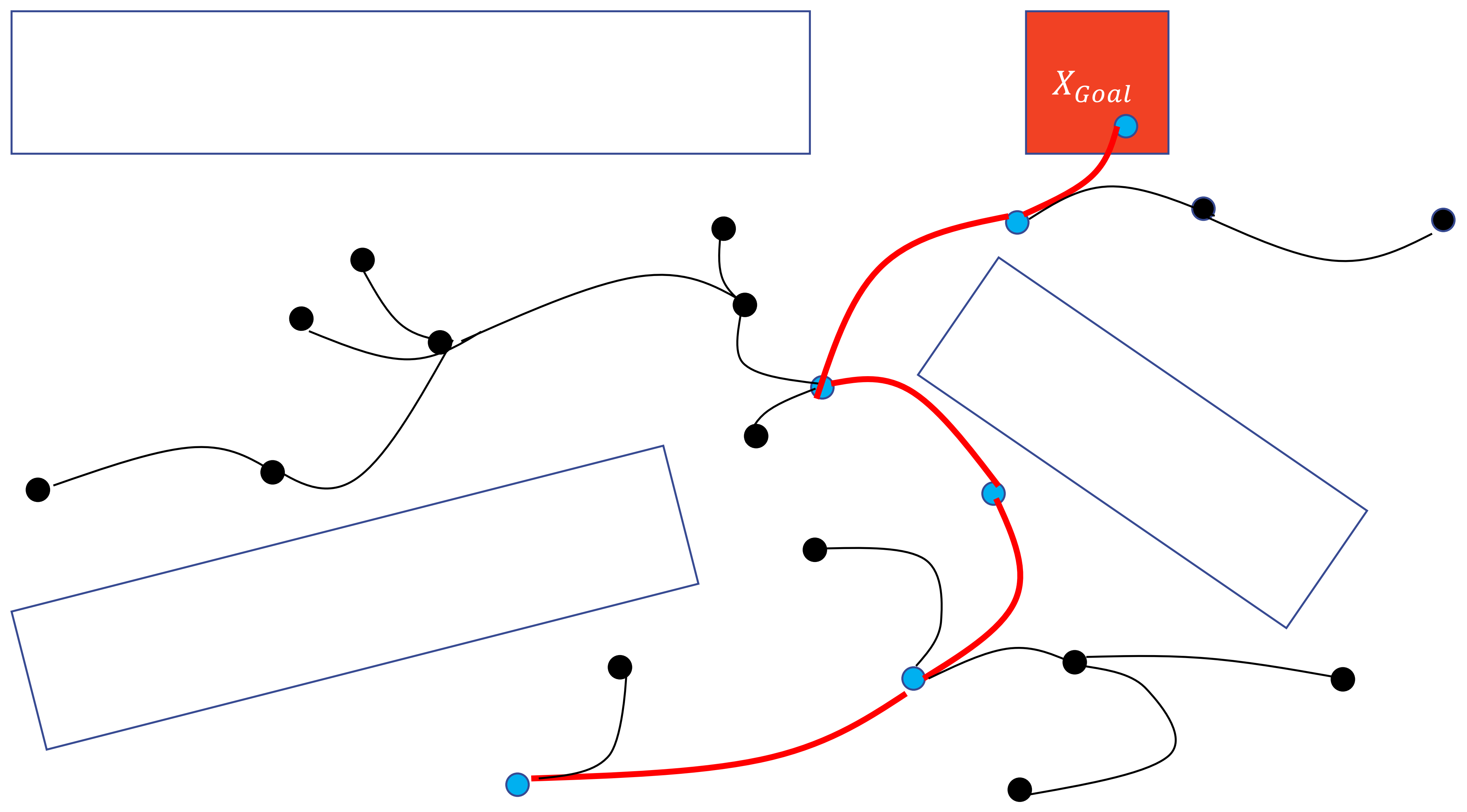
## RRT

```
$V \leftarrow \{x_{init}\}; E \leftarrow \emptyset$
**for** i = 1, . . . , N **do**
  $x_{rand} \leftarrow \text{SampleFree}_i$
  $x_{nearest} \leftarrow \text{Nearest}(G = (V, E), x_{rand})$ // Find node in G that is closest to $x_{rand}$
  $x_{new} \leftarrow \text{Steer}(x_{nearest}, x_{rand})$       // Use local controller to steer $x_{nearest}$ to $x_{rand}$

  if $\text{ObtacleFree}(x_{nearest}, x_{new})$ **then**
    $V \leftarrow V \cup \{x_{new}\}$
    $E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$
**return** $G = (V, E)$
```


$X_{Goal}$


$X_{Goal}$


$X_{Goal}$

# RRTs and Asymptotic Optimality*

- RRTs have many nice properties:
  - RRTs are ***probabilistically complete***
  - RRTs are great at finding feasible trajectories quickly and work well in practice
- However, the generated trajectories are often not "good". Why?
  - Let $Y^{RRT}_n$ be the cost of the best path in the RRT at the end of iteration n.
  - It is shown that $Y^{RRT}_n$ converges (to a random variable), $\lim_{n\to\infty} Y_n^{RRT} = Y_\infty^{RRT}$.
  - The random variable $Y_\infty^{RRT}$ is sampled from a distribution with zero mass at the optimum
  - **Theorem [Karaman & Frizzoli`10]** (Almost sure suboptimality of RRTs) If the set of sampled optimal paths has measure zero, the sampling distribution is absolutely continuous with positive density in $X_{free}$, and d ≥ 2, then the best path in the RRT converges to a sub-optimal solution almost surely, i.e., $\Pr[Y_\infty^{RRT} > c^*] = 1$.

## Summary

- Introduced the graph search algorithm A*, which uses admissible heuristic functions to efficiently find the shortest path
  - Hybrid A* will be covered in MP4!
- Introduced probabilistic or sampling-based motion planning methods, which can help address complexity and feasibility concerns
  - State-of-the-art algorithms such as RRT converge to a NON-optimal solution almost-surely
  - *Not discussed:* There are many new algorithms (e.g., RRG, RRT*), which almost-surely converge to optimal solutions with minimal cost
- *Not discussed:* Local trajectory planners (e.g., MPC) or smoothers are often needed to improve graph-based methods
- *Next time:* High-level decision making, which is often an input to control and planning algorithms

## Extra Slides

## Dynamic programming/Dijkstra

- The optimality principle
  - Let P = (s, . . . , v, . . . g) be an optimal path (from s to g).
  - Then, for any v ∈ P, the sub-path S = (v, . . . , g) is itself an optimal path (from v to g)
- Using the optimality principle
  - Essentially, optimal paths are made of optimal paths. Hence, we can construct long complex optimal paths by putting together short optimal paths, which can be easily computed. Fundamental formula in dynamic programming: $h*(u) = \min_{(u,v) \in E} [w((u, v)) + h*(v)]$ . Typically, it is convenient to build optimal paths working backwards from the goal.

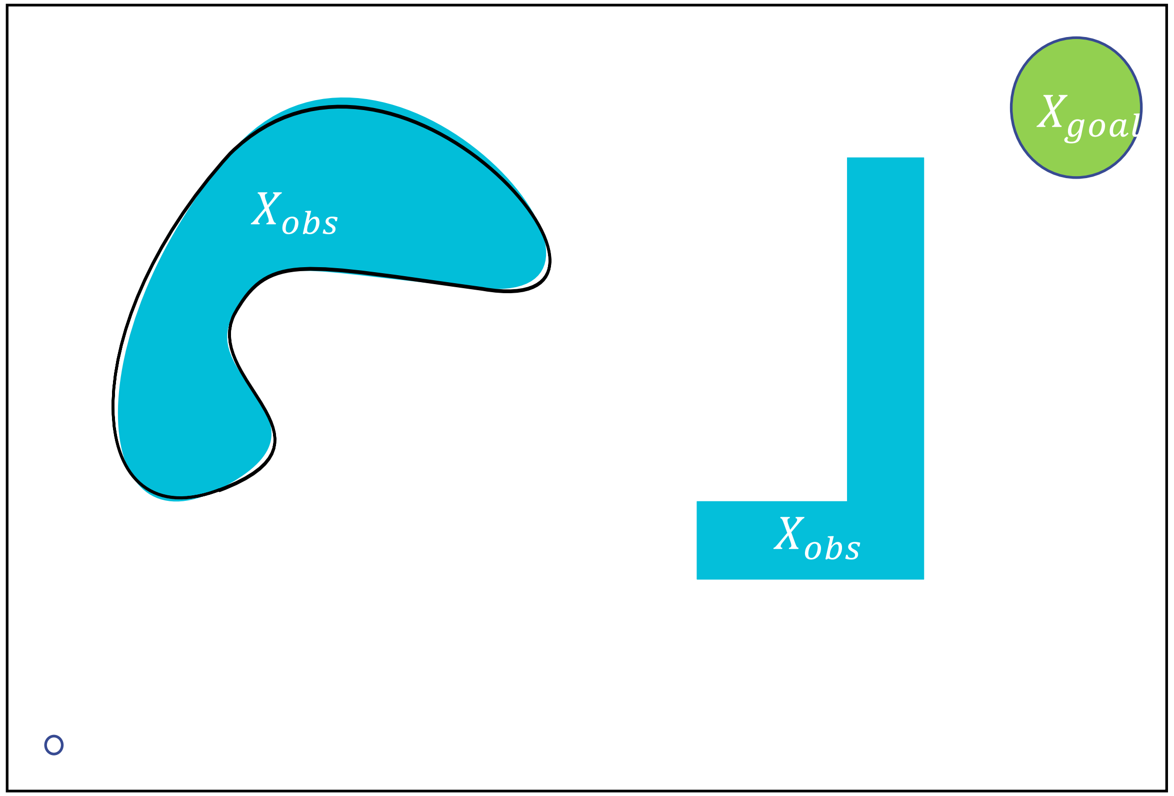
## A* to Continuous State Spaces

# How to apply A* to continuous state spaces? Hybrid A*

- Represent vehicle state in a *uniform* discrete grid
  - 4D grid: $x, y, \theta\ (heading), dir$ (fwd,rev)
- A path (a) over this discrete grid is a start for a plan
  - But the discrete path (a) may not be executable by the vehicle dynamics
- *Hybrid A** solves this problem by shifting the points that represent the discrete cells

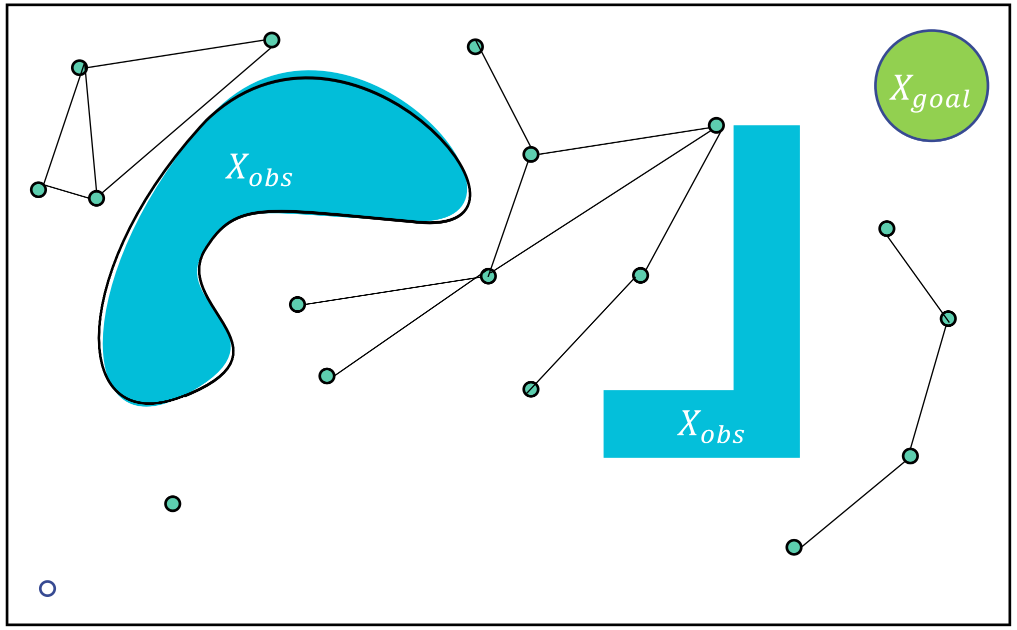
- Represent vehicle state in a *uniform* discrete grid
  - 4D grid: $x, y, \theta\ (heading), dir$ (fwd,rev)
- A path (a) over this discrete grid is a start for a plan
  - But the discrete path (a) may not be executable by the vehicle dynamics
- *Hybrid A** solves this problem by shifting the points that represent the discrete cells

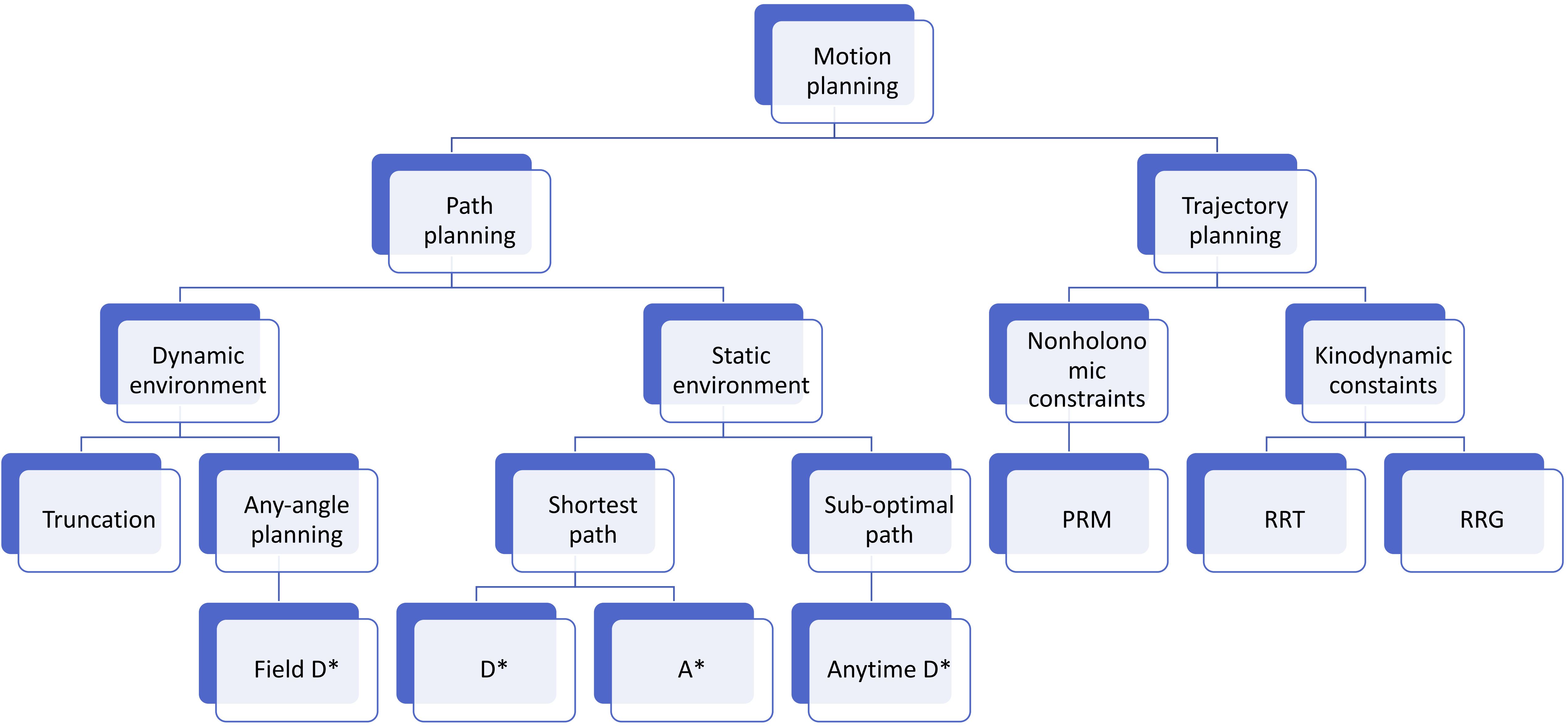
| Algorithm | Prob. Completeness | Asymptotic Optimality | Complexity |
|---|---|---|---|
| sPRM | Yes | Yes | O(N) |
| k-nearest sPRM | No | No | O(log N) |
| RRT | Yes | No | O(log N) |
| PRM* | Yes | Yes | O(log N) |
| k-nearest PRM* | Yes | Yes | O(log N) |
| RRG | Yes | Yes | O(log N) |
| k-nearest RRG | Yes | Yes | O(log N) |
| RRT* | Yes | Yes | O(log N) |
| k-nearest RRT* | Yes | Yes | O(log N) |



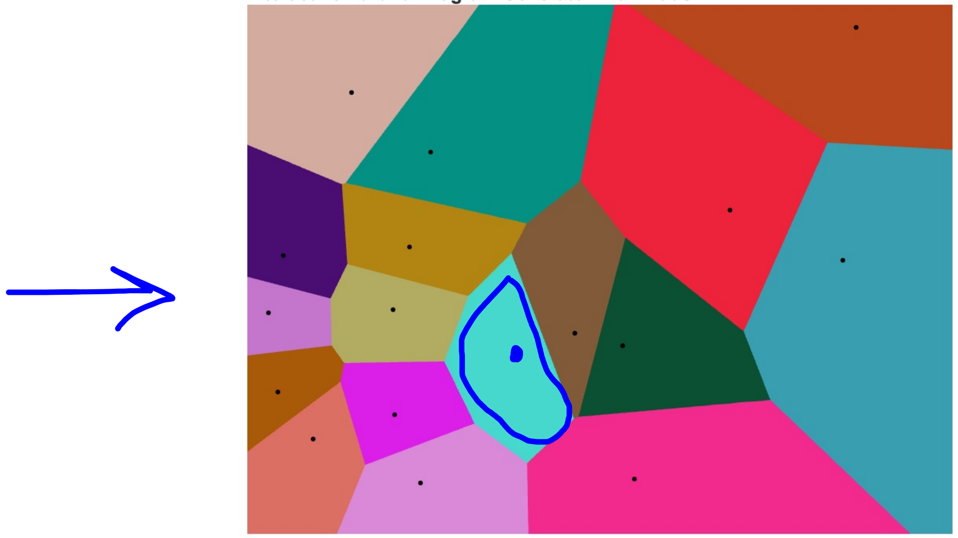
# Probabilistic RoadMap

- Connect points within a radius r, starting from "closest" ones
- Do not attempt to connect points already on the same connected component of PRM
- What properties does this algorithm have?
  - Will it find a solution if one exists?
  - Is this an optimal solution?
  - What is the complexity?




Motion planning
Path planning
Trajectory planning
Dynamic environment
Static environment
Nonholonomic constraints
Kinodynamic constaints
Truncation
Any-angle planning
Shortest path
Sub-optimal path
PRM
RRT
RRG
Field D*
D*
A*
Anytime D*

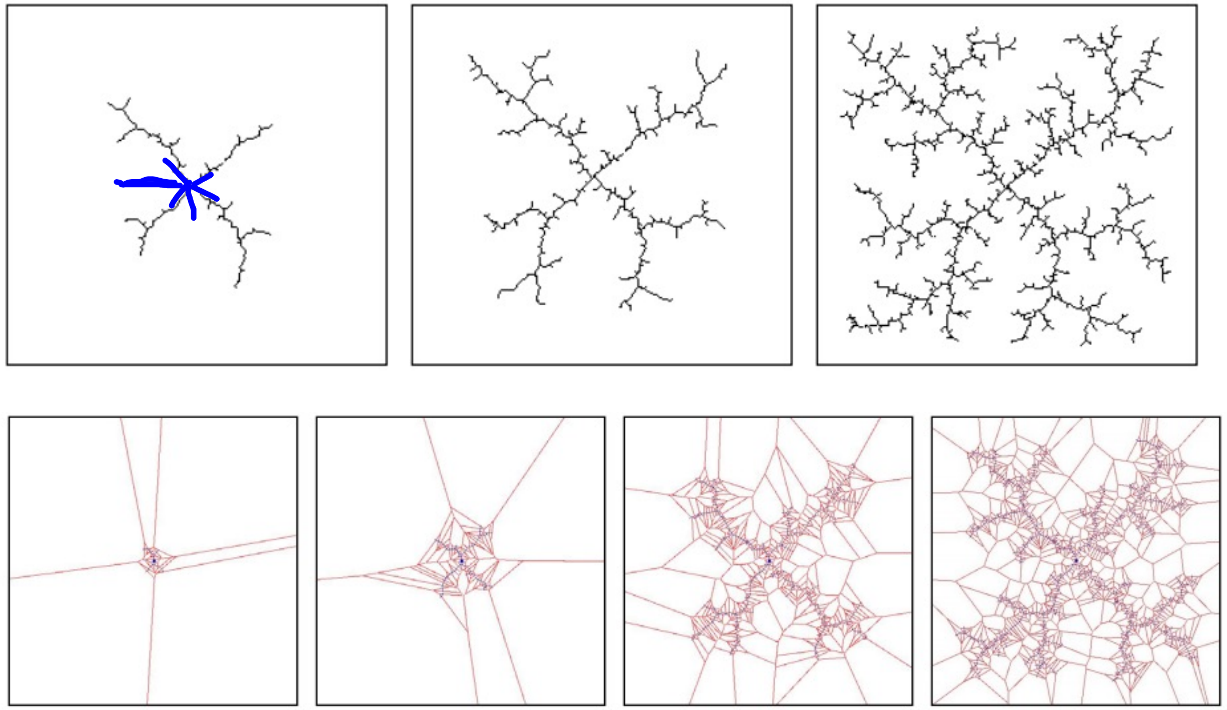
# Voronoi bias



- Given n points in d dimensions, the ***Voronoi diagram*** of the sites is a partition of $R^d$ into regions, one region per point, such that all points in the interior of each region lie closer to that regions site than to any other site.

- Try it: [http://alexbeutel.com/webgl/voronoi.html](http://alexbeutel.com/webgl/voronoi.html)

- ***Voronoi bias.*** Vertices of the RRT that are more "isolated" (e.g., in unexplored areas, or at the boundary of the explored area) have larger Voronoi regions—and are more likely to be selected for extension.
- [http://lavalle.pl/rrt/](http://lavalle.pl/rrt/)



# Why is RRT not asymptotically optimal?

- Root node has infinitely many subtrees that extend at least a distance $\epsilon$ away from $x_{init}$.
- The RRT algorithm "traps" itself by disallowing new better paths to emerge. Why?
- Heuristics such as running the RRT multiple times, running multiple trees concurrently etc., work better than the standard RRT, but also result in almost-sure sub-optimality.
- A careful rethinking of the RRT algorithm is required for (asymptotic) optimality.

# RRT in action [Frazzoli]

- Talos, the MIT entry to the 2007 DARPA Urban Challenge, relied on an“RRT-like” algorithm for real-time motion planning and control.
- The devil is in the details: provisions needed for, e.g.,
  - Real-time, on-line planning for a safety-critical vehicle with substantial momentum.
  - Uncertain, dynamic environment with limited/faulty sensors.
- Main innovations [Kuwata, et al. ’09]
  - Closed-loop planning: plan reference trajectories for a closed-loop model of the vehicle under a stabilizing feedback
  - Safety invariance: Always maintain the ability to stop safely within the sensing region.
  - Lazy evaluation: the actual trajectory may deviate from the planned one,need to efficiently re-check the tree for feasibility.
- The RRT-based P+C system performed flawlessly throughout the race.
- https://journals.sagepub.com/doi/abs/10.1177/0278364911406761

## Limitations

The MIT DARPA Urban Challenge code, as well as other incremental sampling methods, suffer from the following limitations:

- No characterization of the quality (e.g., “cost”) of the trajectories returned by the algorithm.
- Keep running the RRT even after the first solution has been obtained, for as long as possible (given the real-time constraints), hoping to find a better path than that already available.
- No systematic method for imposing temporal/logical constraints, such as, e.g., the rules of the road, complicated mission objectives, ethical/deontic code.
- In the DARPA Urban Challenge, all logics for, e.g., intersection handling, had to be hand-coded, at a huge cost in terms of debugging effort/reliability of the code.

## Rapidly Exploring Random Graphs (possibly cyclic)

```
$V \leftarrow \{x_{init}\}$; $E \leftarrow \emptyset$;
for i = 1, . . . , N do
  $x_{rand} \leftarrow$ SampleFreei;
  $x_{nearest} \leftarrow$ Nearest($G = (V, E), x_{rand}$);
  $x_{new} \leftarrow$ Steer($x_{nearest}, x_{rand}$) ;
  if ObtacleFree($x_{nearest}, x_{new}$) then
    $X_{near} \leftarrow$ Near($G = (V, E), x_{new}, \min\{\gamma_{RRG}(\log(\text{card } V)/ \text{card } V)^{1/d}, \eta\}$) ;
    $V \leftarrow V \cup \{x_{new}\}$; $E \leftarrow E \cup \{(x_{nearest}, x_{new}),(x_{new}, x_{nearest})\}$ ;
    foreach $x_{near} \in X_{near}$ do
      if CollisionFree($x_{near}, x_{new}$) then $E \leftarrow E \cup \{(x_{near}, x_{new}),(x_{new}, x_{near})\}$
return $G = (V, E)$;
```

At each iteration, the RRG tries to connect the new sample to all vertices in a ball of radius $r_n$ centered at it. (Or just default to the nearest one if such b all is empty.)

# Theorems [Proofs not required for exam]

- **Probabilistic completeness.** Since $V_n^{RRG} = V_n^{RRT}$, for all n RRG has the same completeness properties as RRT, i.e.,

$$Pr\left[V_n^{RRG} \cap X_{goal} = \emptyset\right] = O\left(e^{-bn}\right).$$

- **Asymptotic optimality.** If the **Near** procedure returns all nodes in V within a ball of volume $Vol = \frac{\gamma \log n}{n}$, $\gamma > 2^d\left(1 + \frac{1}{d}\right)$, under some additional technical assumptions (e.g., on the sampling distribution, on the $\epsilon$ clearance of the optimal path, and on the continuity of the cost function), the best path in the RRG converges to an optimal solution almost surely, i.e.,

$$\Pr[Y_\infty^{RRG} = c^*] = 1.$$

# Final thoughts on RRG*

- What is the additional computational load?
  - O(log n) extra calls to ObstacleFree compared to RRT
- Key idea in RRG/RRT*:
  - Combine optimality and computational efficiency, it is necessary to attempt connection to Θ(log N) nodes at each iteration.
  - Reduce volume of the "connection ball" as log(N)/N;
  - Increase the number of connections as log(N).
- These principles can be used to obtain "optimal" versions of PRM, etc.