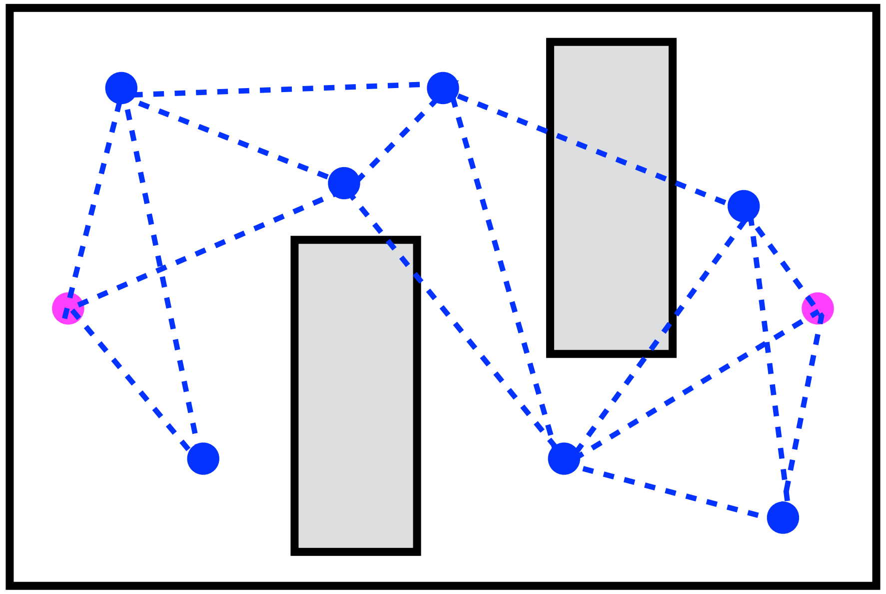
Incremental Planning

Sanjiban Choudhury

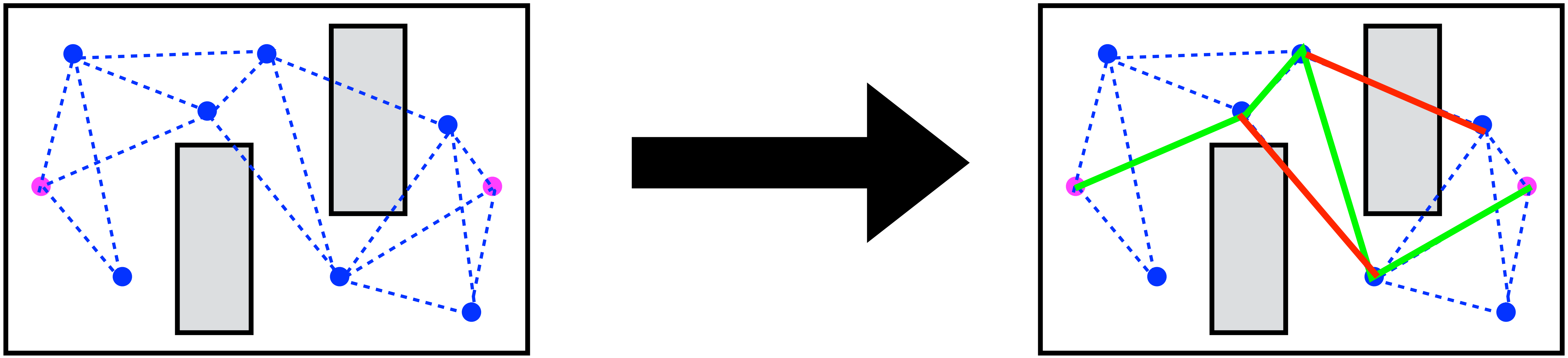
TAs: Matthew Rockett, Gilwoo Lee, Matt Schmittle

General framework for motion planning



Create a graph

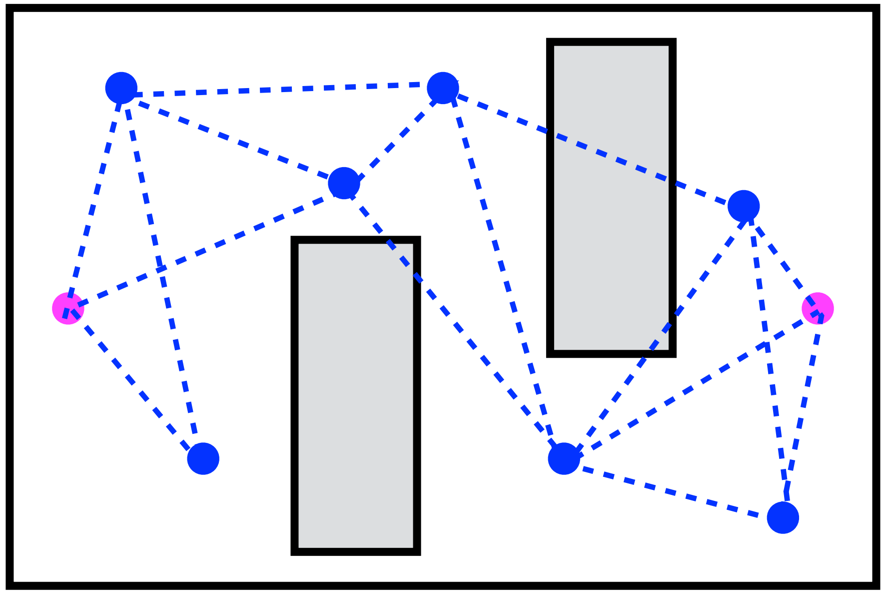
General framework for motion planning



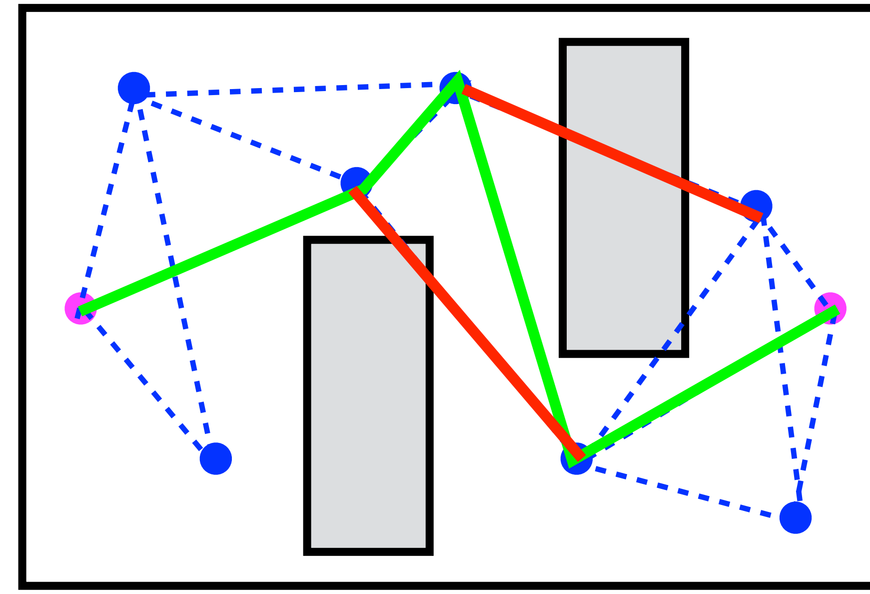
Create a graph

Search the graph

General framework for motion planning



Create a graph



Search the graph

Interleave

General framework for motion planning

Any planning algorithm

Create graph

Search graph

Interleave

General framework for motion planning

Any planning algorithm

Create graph

Search graph

Interleave

General framework for motion planning

Any planning algorithm

=

Create graph

Search graph

Interleave

General framework for motion planning

Any planning algorithm

$=$

Create graph

e.g. fancy random sampler

$\times$

Search graph

e.g. fancy heuristic

$\times$

Interleave

e.g. fancy way of densifying

General framework for motion planning

Any planning algorithm		Create graph		Search graph		Interleave
	=	e.g. fancy random sampler	×	e.g. fancy heuristic	×	e.g. fancy way of densifying
		Whats the best we can do?		Whats the best we can do?		Whats the best we can do?

Today's discussion

1. Why would we want to interleave?

2. How do we search when we interleave?

(repairing search)

3. How do we improve graphs when we interleave?

(incremental sampling)

4. Putting it all together

Today's discussion

1. Why would we want to interleave?

2. How do we search when we interleave?

(repairing search)

3. How do we improve graphs when we interleave?

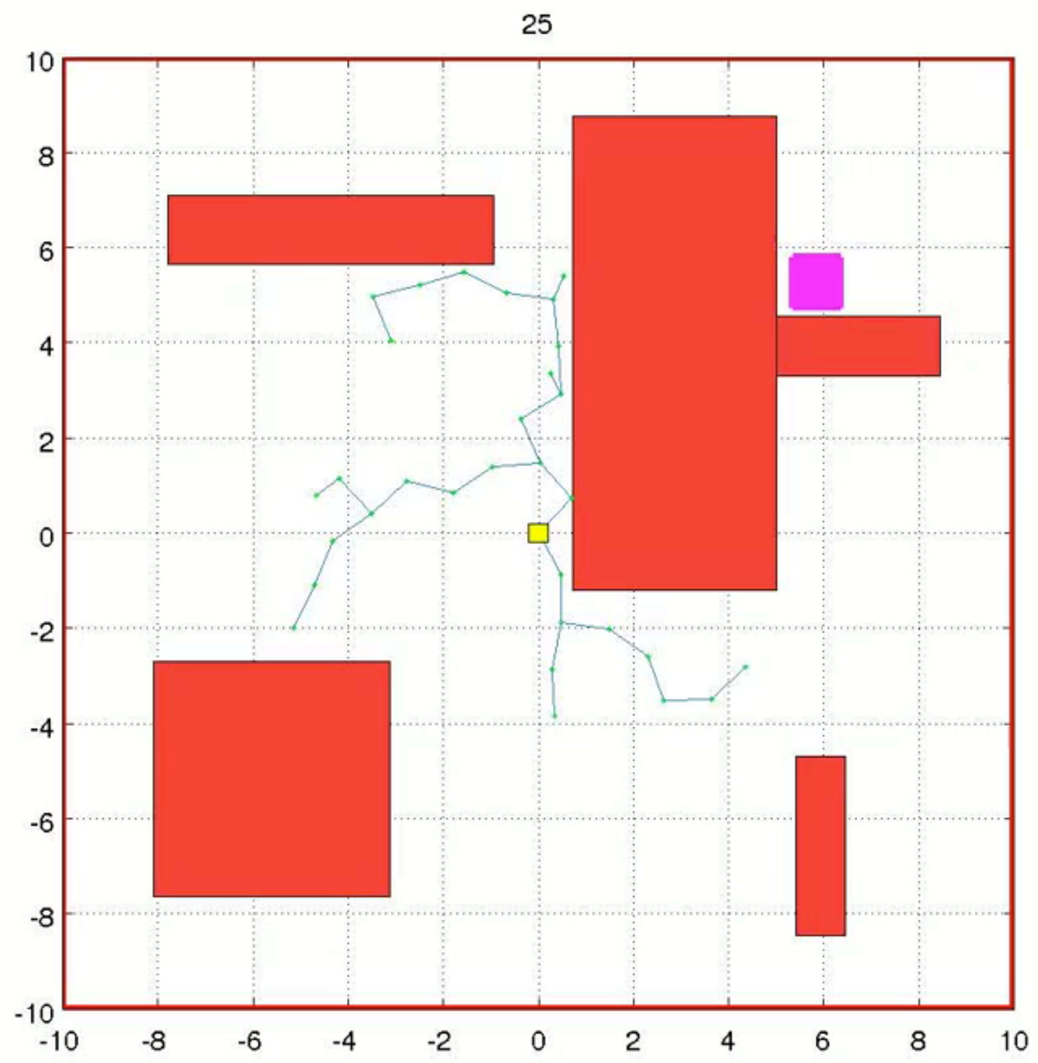
(incremental sampling)

4. Putting it all together

Anytime Planning

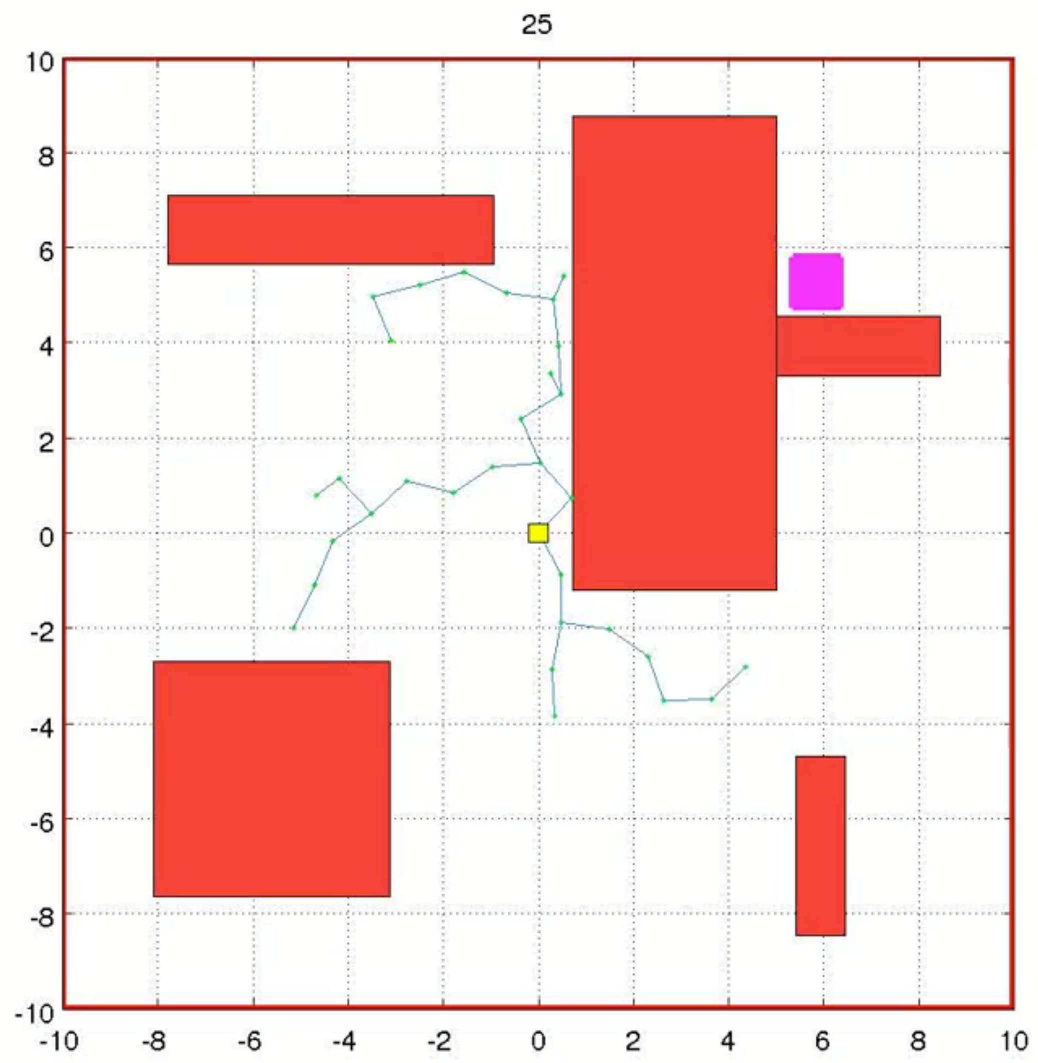
Anytime Planning

Quickly get a feasible path. Improve if you have more time.



Anytime Planning

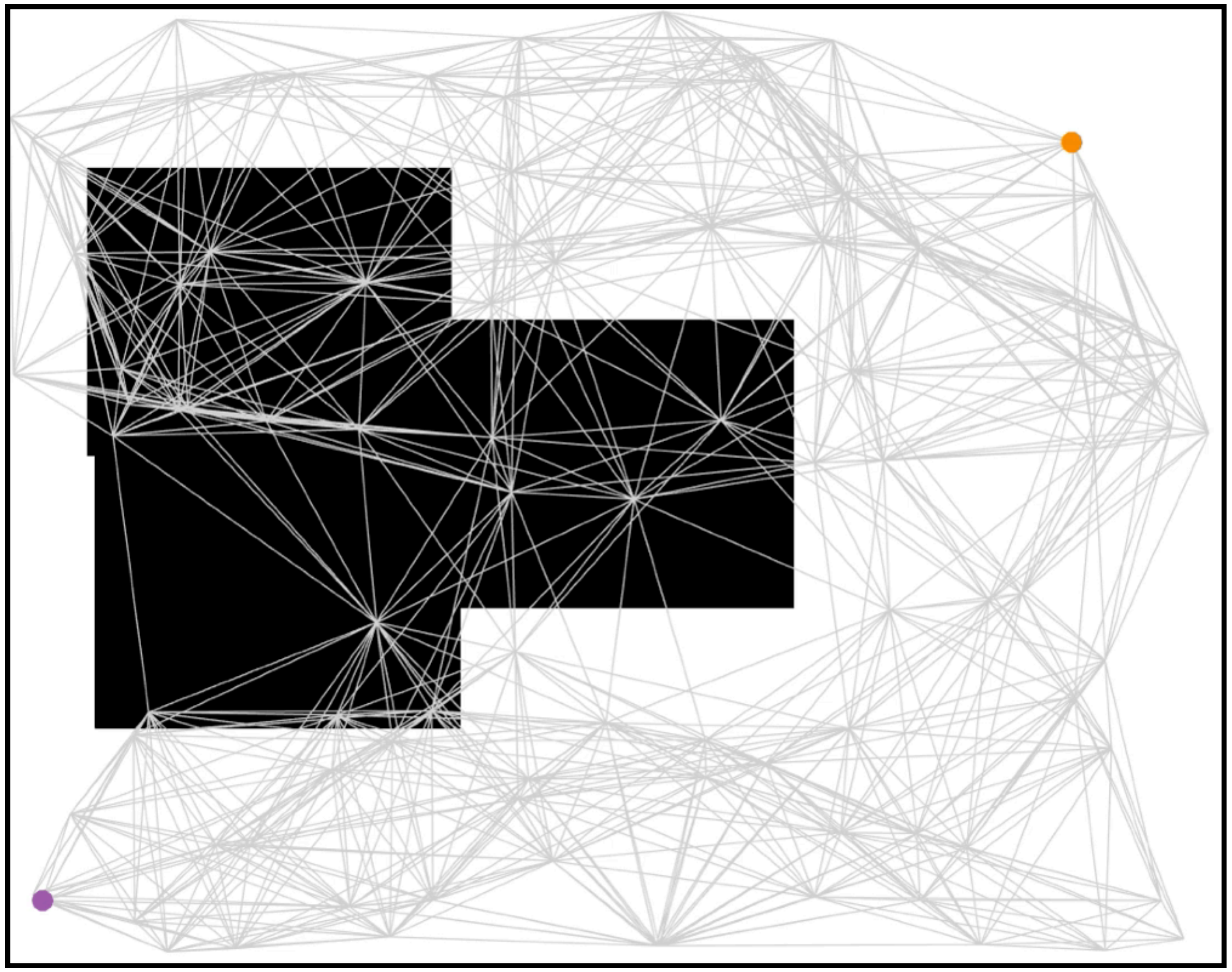
Quickly get a feasible path. Improve if you have more time.



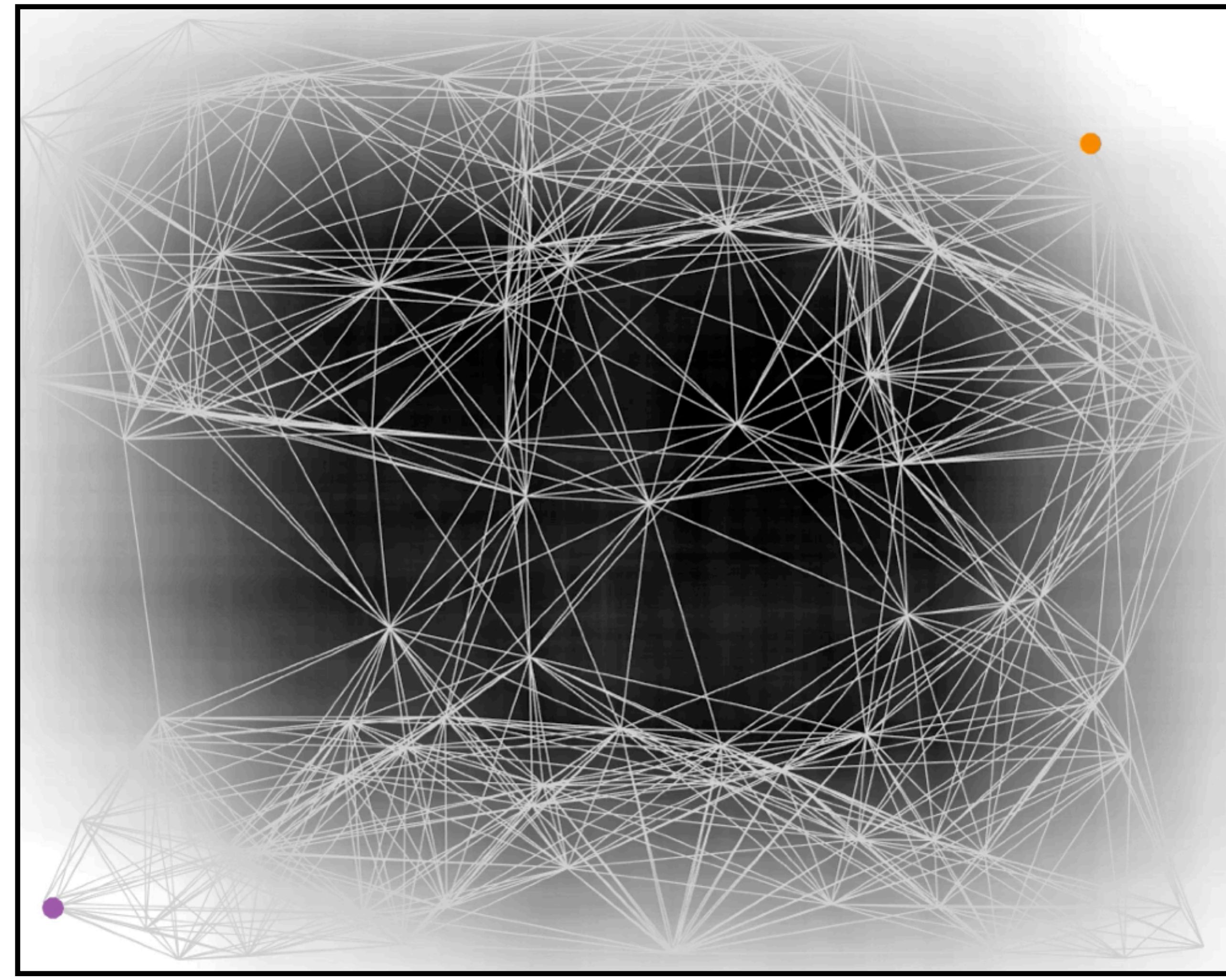
Planning as Inference

A Bayesian Approach to Edge Evaluation

Ground truth



Agent's belief

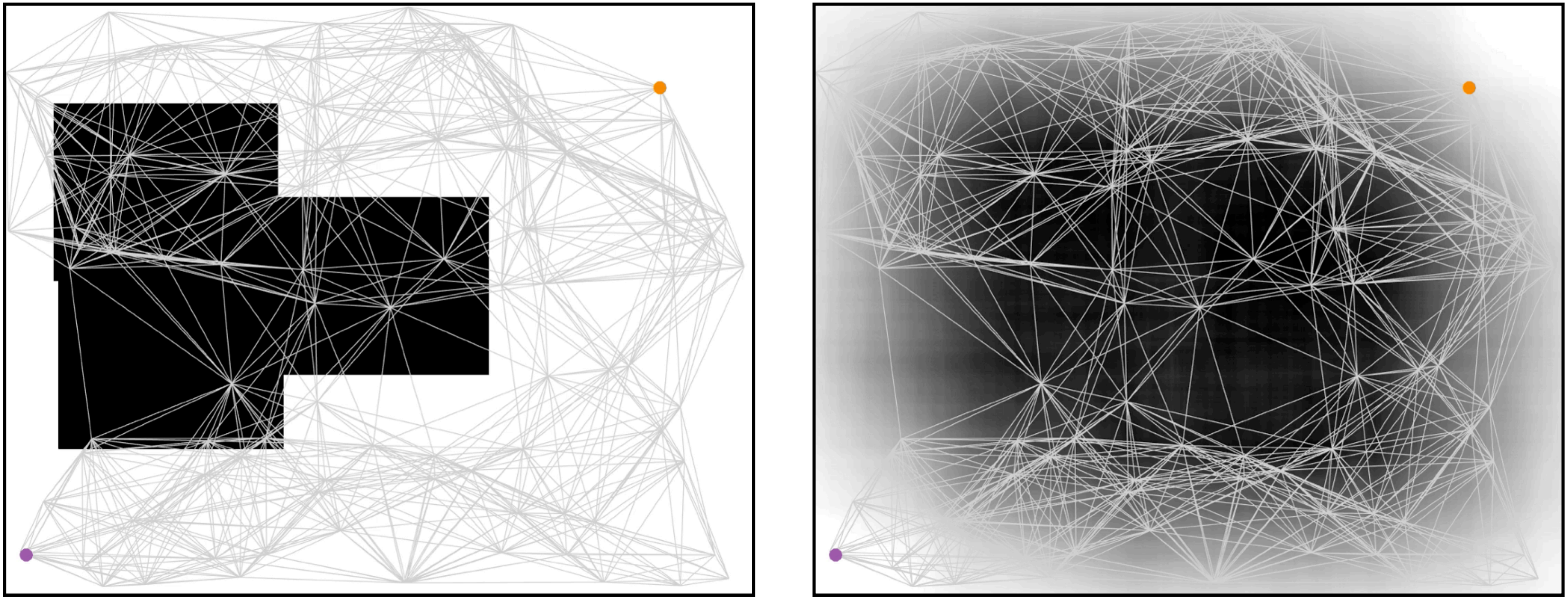


First Set of Provably Near Bayes-Optimal Planning Algorithms

[NIPS'17, ISRR'17, IJCAI'18]

A Bayesian Approach to Edge Evaluation

Ground truth　　　　Agent's belief



First Set of Provably Near Bayes-Optimal Planning Algorithms

[NIPS'17, ISRR'17, IJCAI'18]

Today's discussion

1. Why would we want to interleave?

2. How do we search when we interleave?

(repairing search)

3. How do we improve graphs when we interleave?

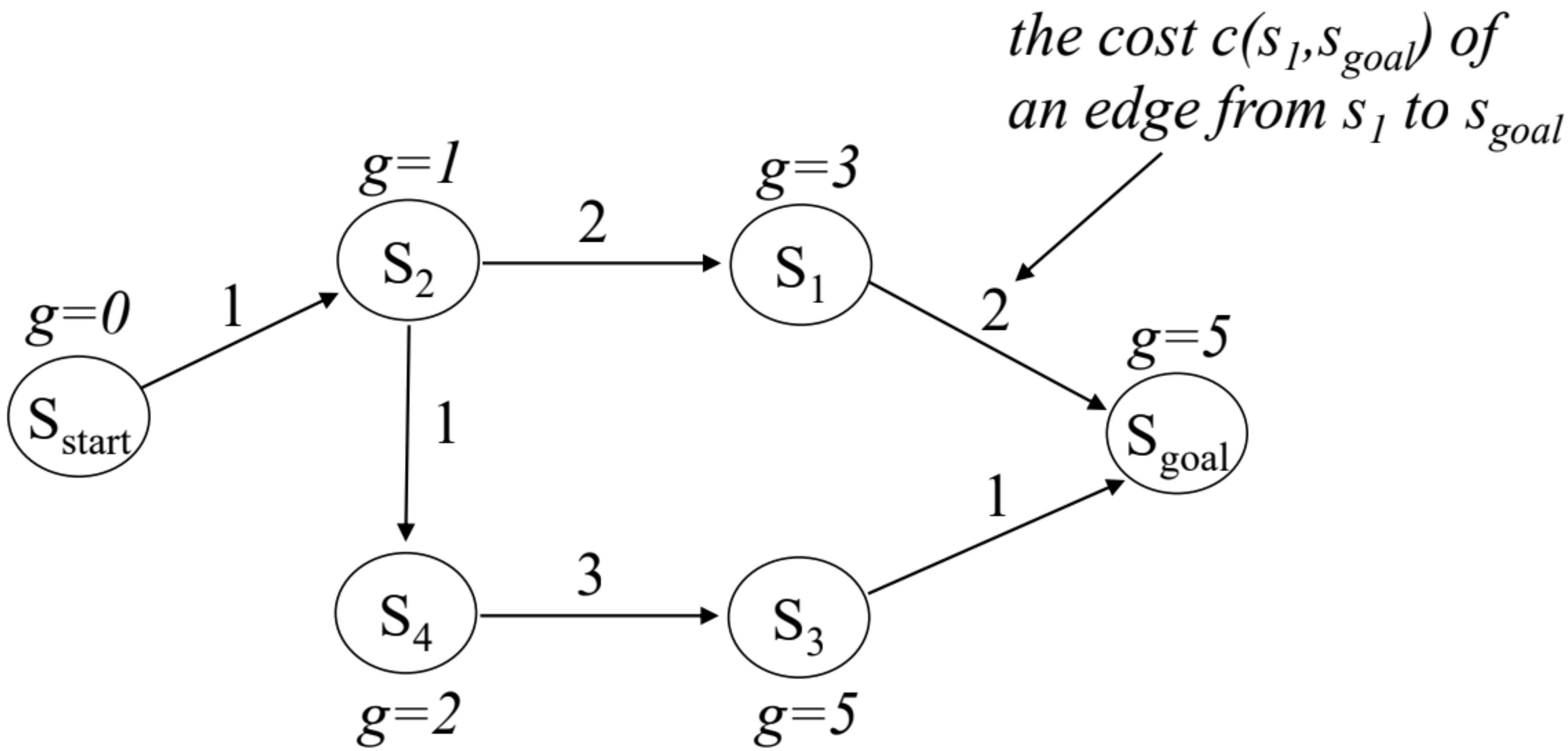
(incremental sampling)

4. Putting it all together

Interleaving implies
new vertices / edges appear

Do we always have to replan
whenever the graph changes?

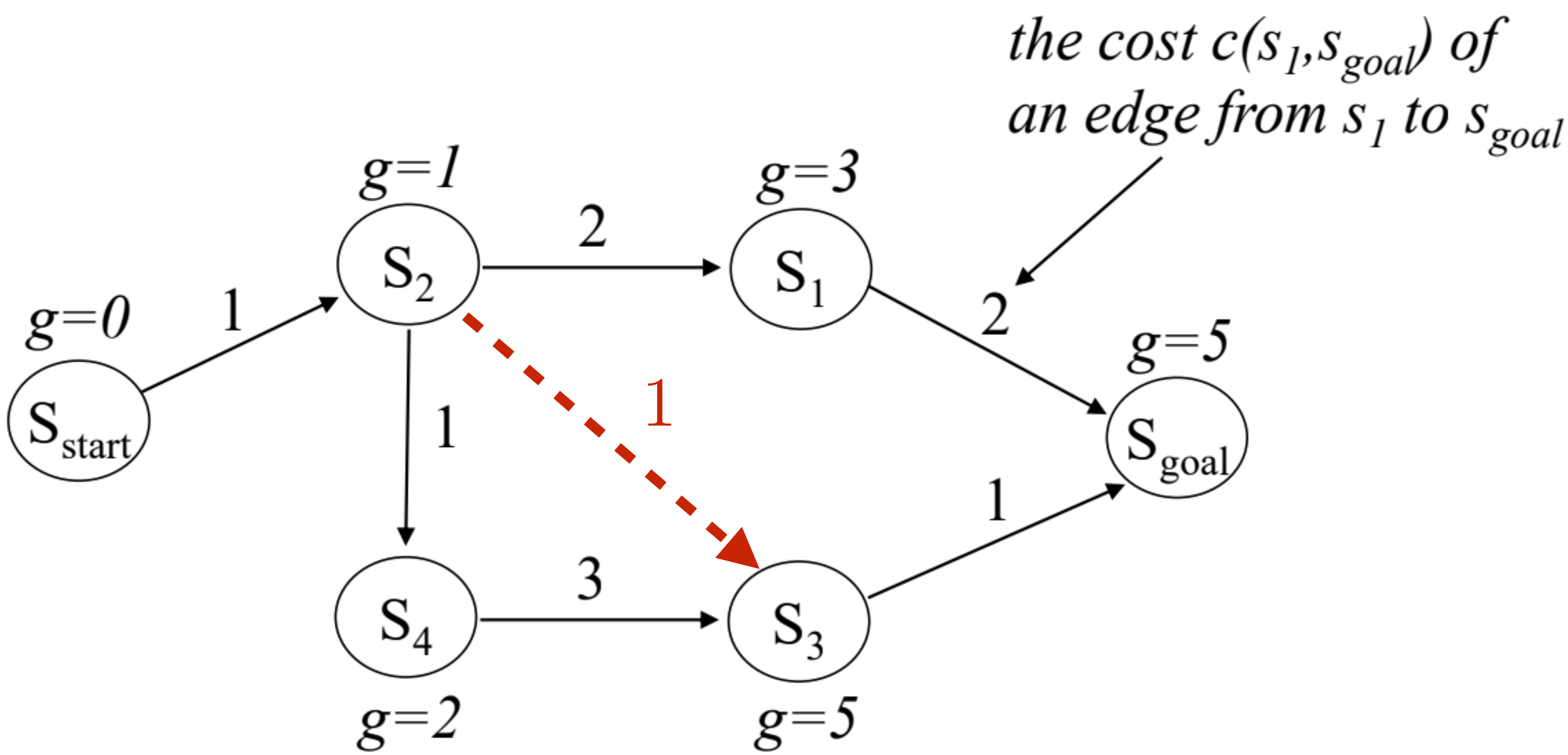
What's true about g(s) values after search?



Vertices are locally consistent

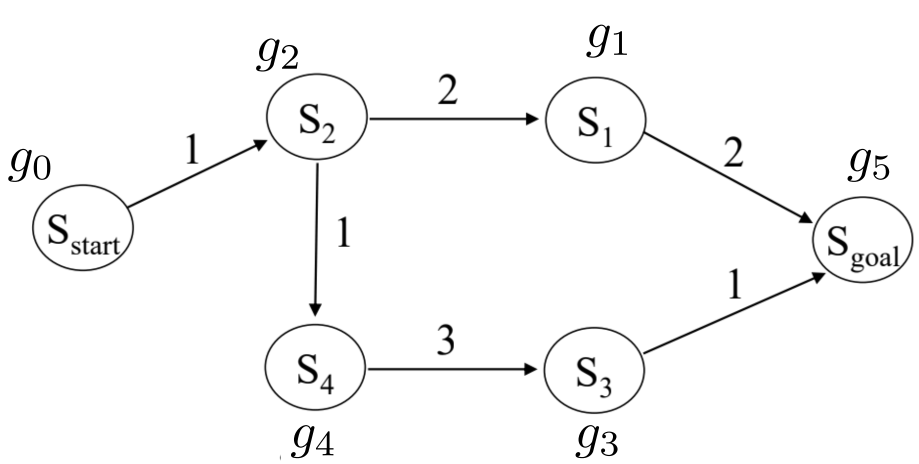
$$g(s) = \min_{s' \in \mathrm{pred}(s)} \left(g(s') + c(s', s) \right)$$

What happens when we introduce a new edge?



Why
Reinforcement Learning
played a big role in
developing planning ...
(obv. the reverse is true)

Value iteration on graphs



$$\begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \end{bmatrix}$$

$$V(s)$$

Value iteration step

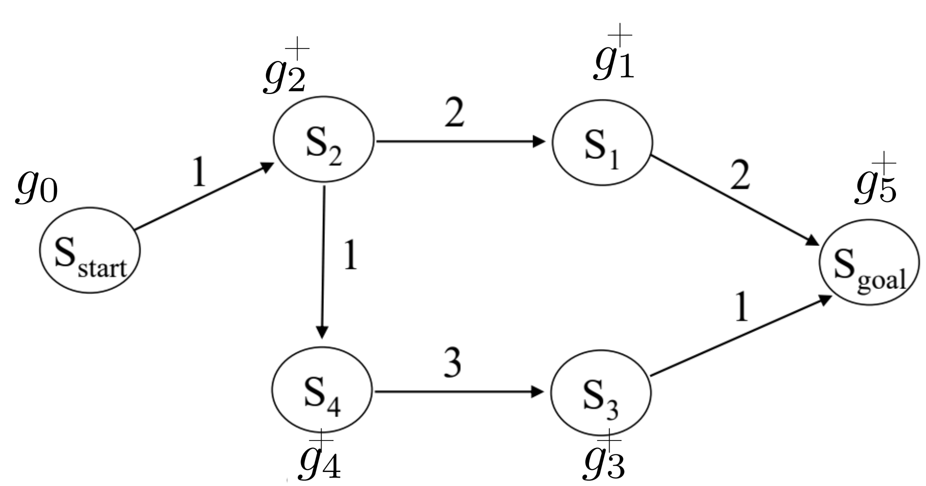
$$g^{+}(s) = \min_{s' \in \mathrm{pred}(s)} \left(g(s') + c(s', s) \right)$$

Do this for all states!

Value iteration on graphs

$$\begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \end{bmatrix} \xrightarrow{\displaystyle g^+(s) = \min_{s' \in \text{pred}(s)} (g(s') + c(s', s))} \begin{bmatrix} g_1^+ \\ g_2^+ \\ g_3^+ \\ g_4^+ \\ g_5^+ \end{bmatrix}$$

$$V(s) \qquad\qquad V^+(s)$$



Does this converge?

$$V^1(s) \longrightarrow V^2(s) \quad \cdots \quad \longrightarrow V^n(s)$$

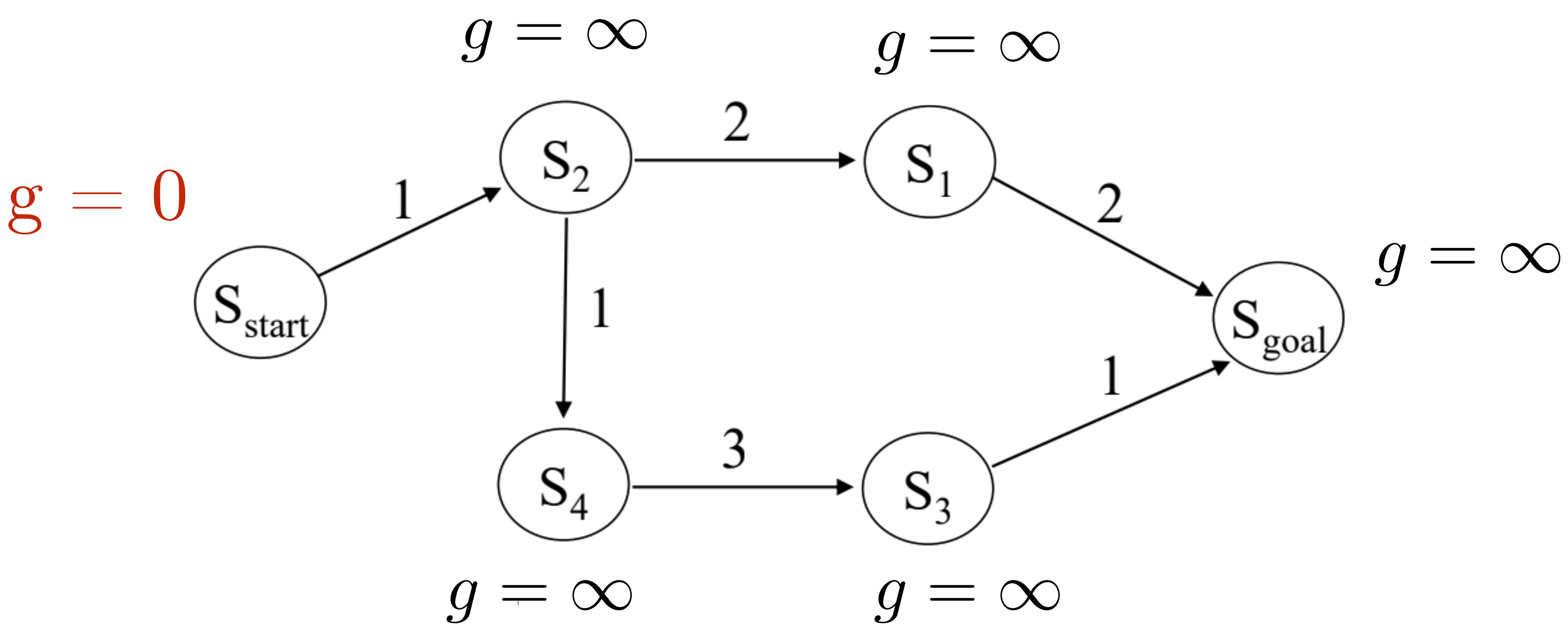
Does this converge?

$$V^1(s) \longrightarrow V^2(s) \quad \cdots \quad \longrightarrow V^n(s)$$

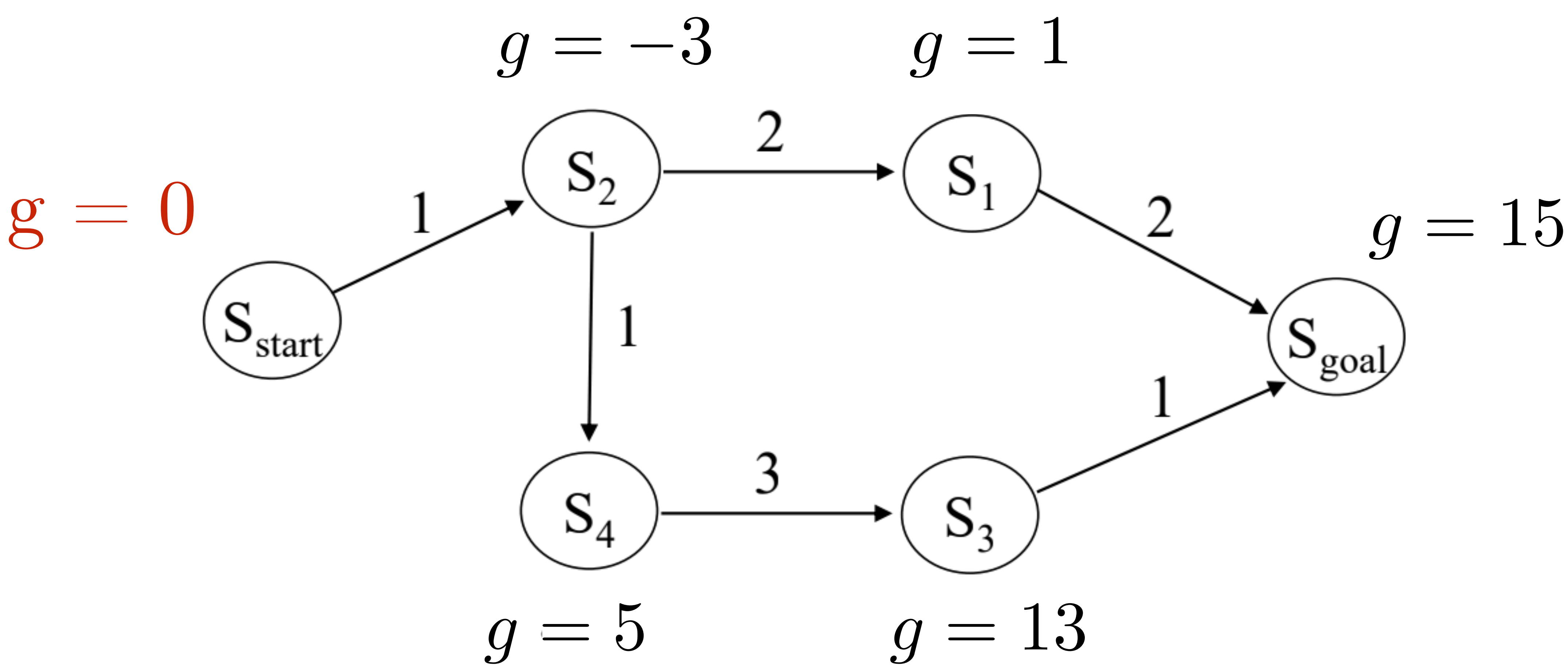
Yes!

Value iteration is a contraction

Does this converge?



Does this converge?



Asynchronous value iteration

What if we didn't update for ALL the states?

$$\underset{V(s)}{\begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \end{bmatrix}} \xrightarrow{g^+(s) = \min\limits_{s' \in \text{pred}(s)} (g(s') + c(s', s))} \underset{V^+(s)}{\begin{bmatrix} g_1^+ \\ g_2^+ \\ g_3^+ \\ g_4^+ \\ g_5^+ \end{bmatrix}}$$

Asynchronous value iteration

What if we didn't update for ALL the states?

$$g^+(s) = \min_{s' \in \text{pred}(s)} (g(s') + c(s', s))$$

What if we did this for a RANDOM SUBSET of states?

Asynchronous value iteration

What if we didn't update for ALL the states?

$$g^+(s) = \min_{s' \in \text{pred}(s)} \left(g(s') + c(s', s) \right)$$

What if we did this for a RANDOM SUBSET of states?

Does this converge?

Asynchronous value iteration

What if we didn't update for ALL the states?

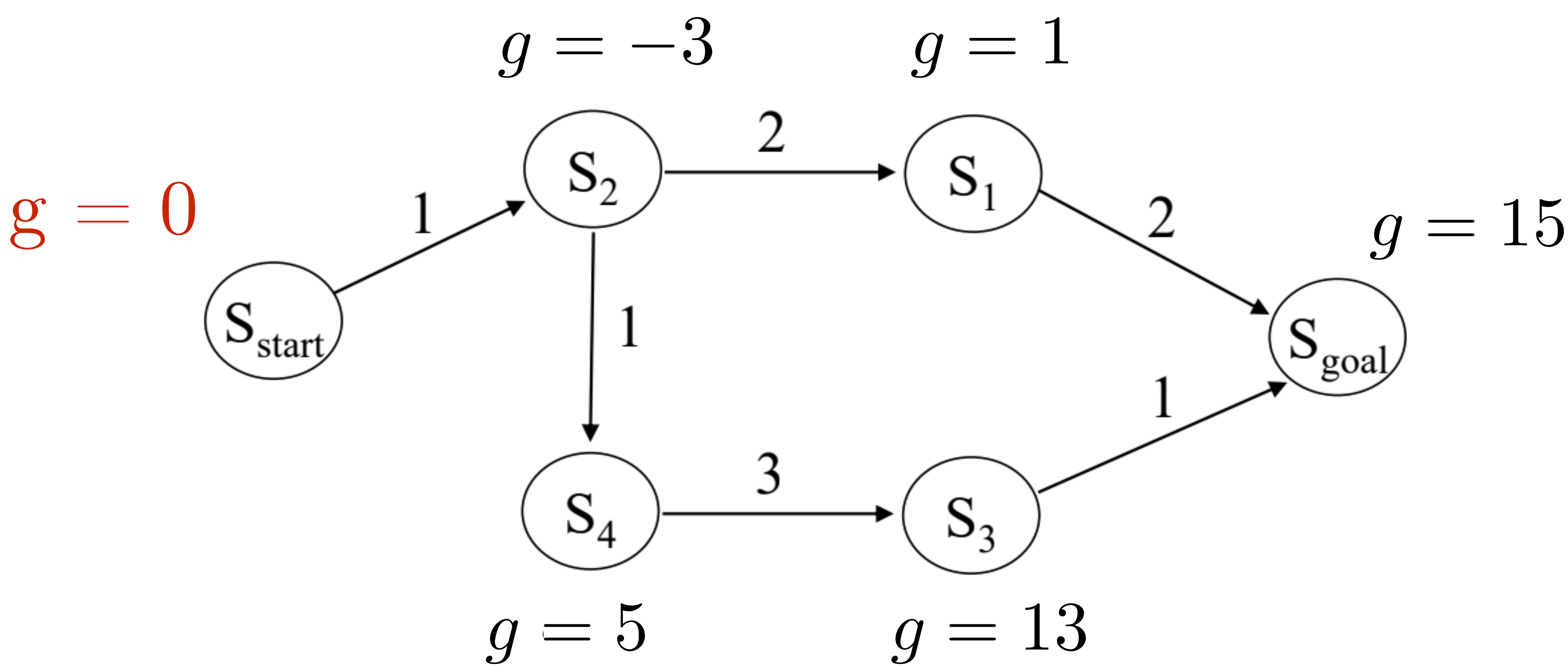
$$g^+(s) = \min_{s' \in \mathrm{pred}(s)} (g(s') + c(s', s))$$

What if we did this for a RANDOM SUBSET of states?

Does this converge?

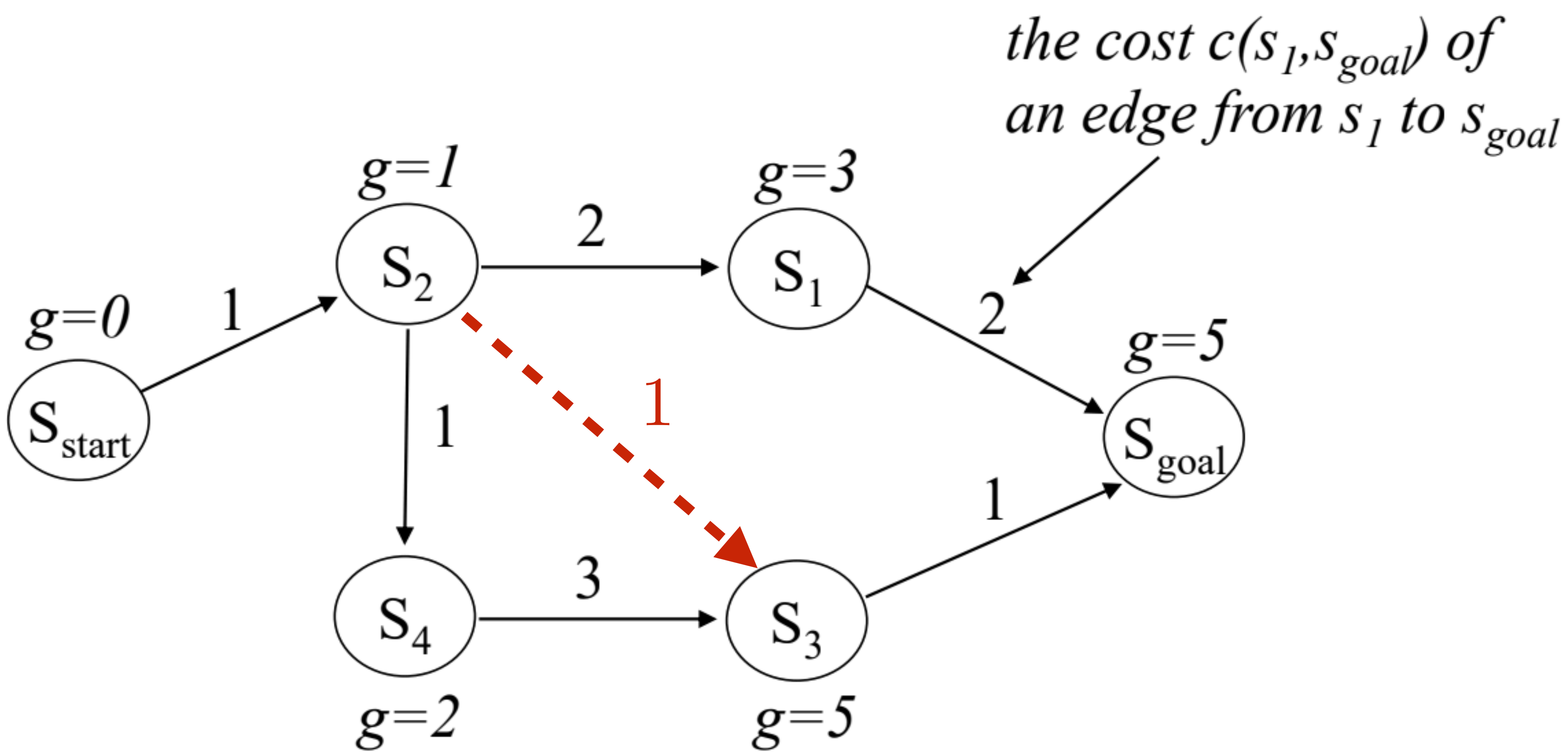
YES

Does this converge?



Back to our problem ...

What happens if you run asynchronous value iteration?



Key Idea

Run asynchronous value iteration in
an
organized way

LPA* (Koenig and Likhachev)

How general is this idea?

How general is this idea?

How many ways can a graph change?

How general is this idea?

How many ways can a graph change?

New edges / vertices appear

How general is this idea?

How many ways can a graph change?

New edges / vertices appear

Cost of edges increase (lazy evaluation)

How general is this idea?

How many ways can a graph change?

New edges / vertices appear

Cost of edges increase (lazy evaluation)

Cost of edges increase/decrease (approximation tech)

How general is this idea?

How many ways can a graph change?

New edges / vertices appear

Cost of edges increase (lazy evaluation)

Cost of edges increase/decrease (approximation tech)

F-value of nodes change (dynamic heuristic)

How general is this idea?

How many ways can a graph change?

New edges / vertices appear

Cost of edges increase (lazy evaluation)

Cost of edges increase/decrease (approximation tech)

F-value of nodes change (dynamic heuristic)

What about planning across iterations?

How general is this idea?

How many ways can a graph change?

New edges / vertices appear

Cost of edges increase (lazy evaluation)

Cost of edges increase/decrease (approximation tech)

F-value of nodes change (dynamic heuristic)

What about planning across iterations?

New obstacles appear/disappear - cost of edges increase/decrease

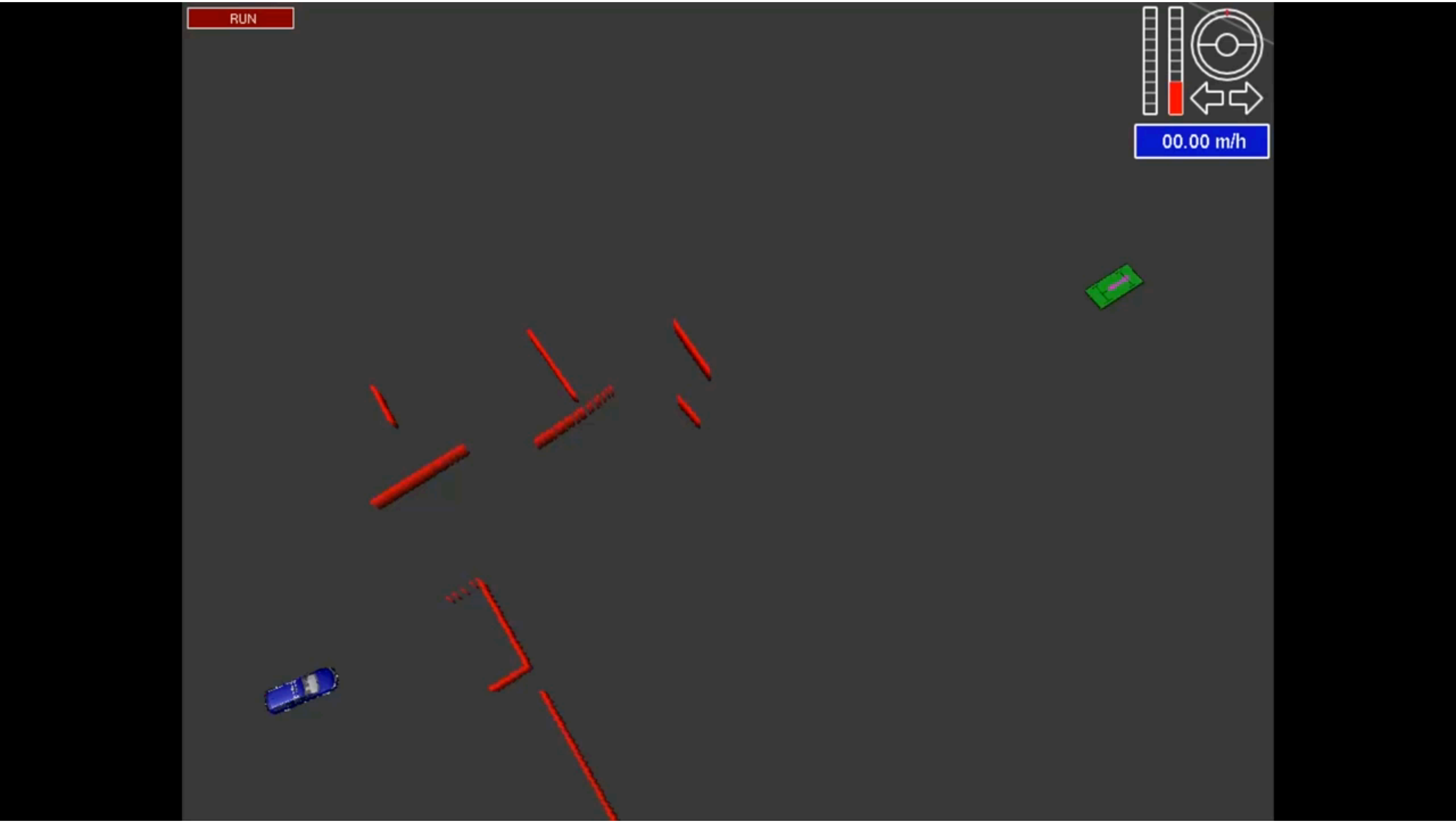
Anytime Repairing A* (ARA*)

https://www.youtube.com/watch?v=rZHtHJlJa2w

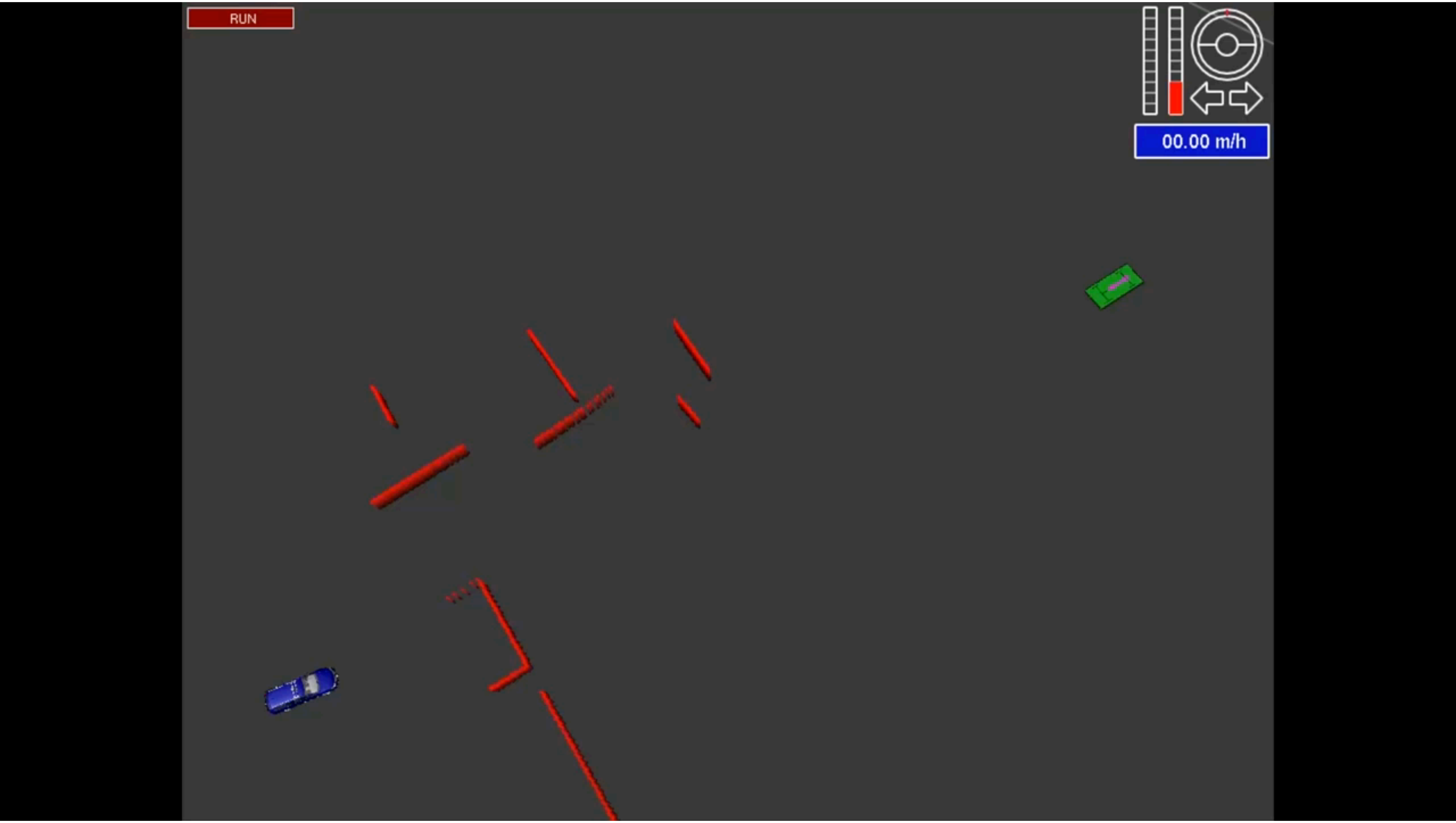
Anytime Repairing A* (ARA*)

https://www.youtube.com/watch?v=rZHtHJlJa2w

D*-Lite



D*-Lite



Today's discussion

1. Why would we want to interleave?

2. How do we search when we interleave?

(repairing search)

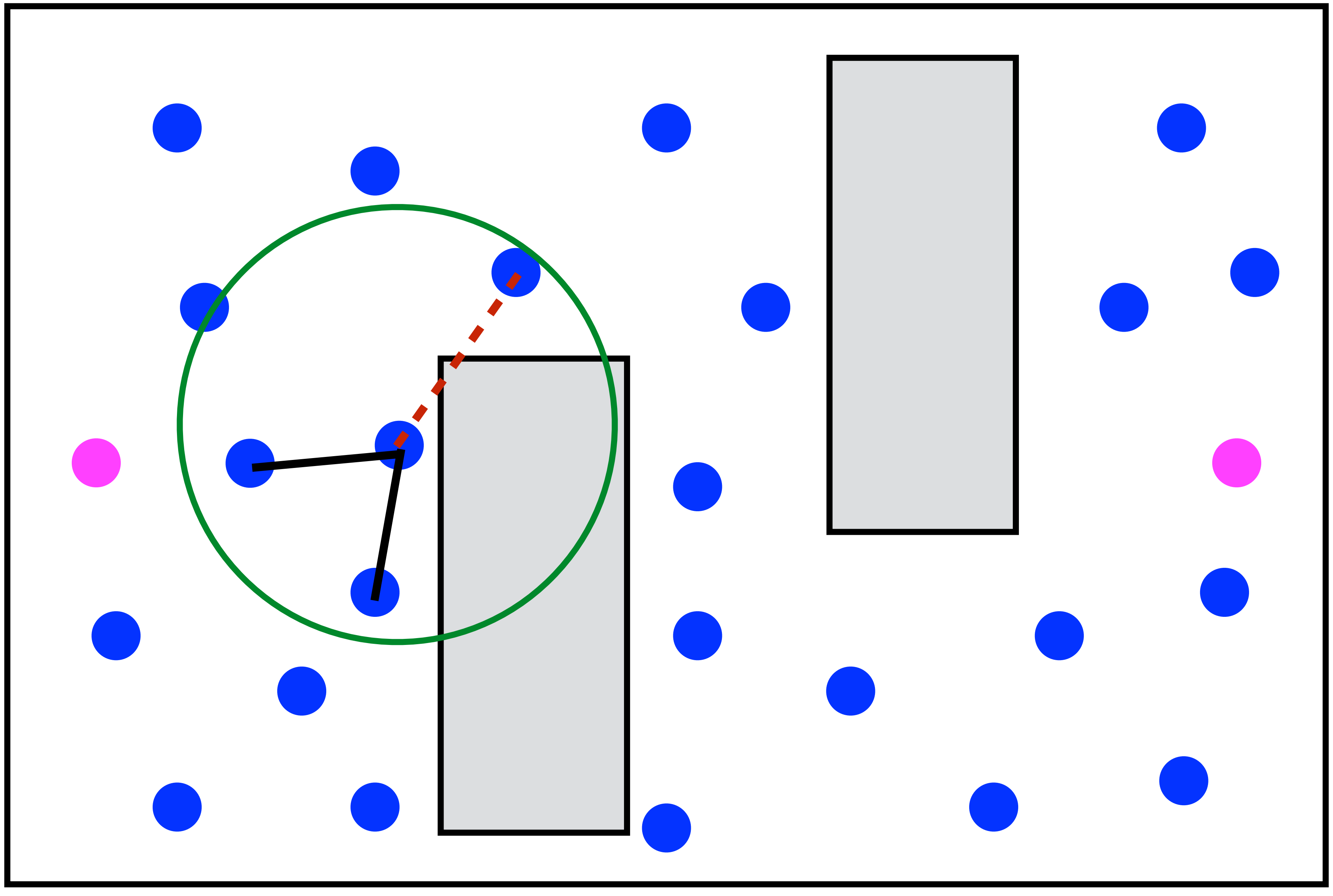
3. How do we improve graphs when we interleave?

(incremental sampling)

4. Putting it all together

What is incremental sampling?

Probabilistic Roadmaps were batch

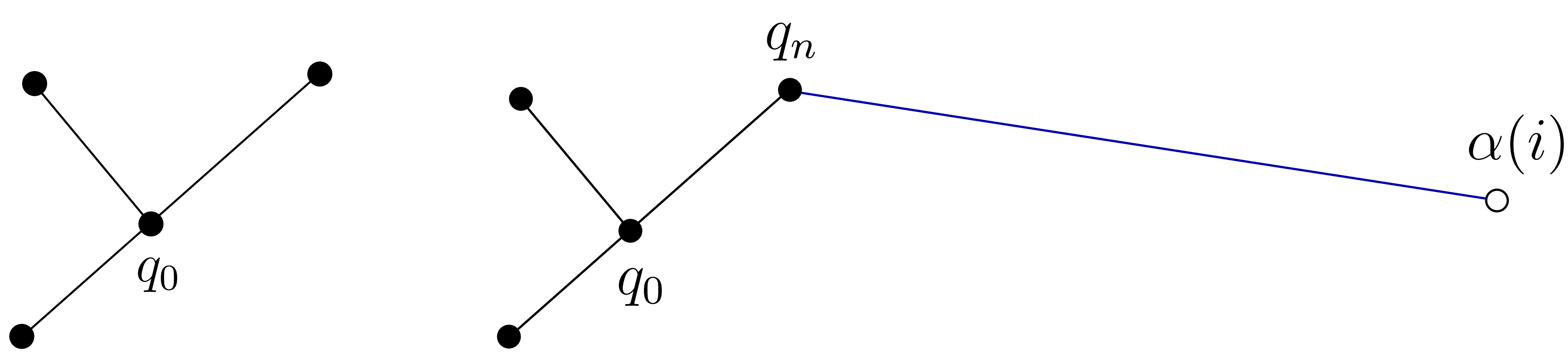


Rapidly Exploring Dense Tree (RDT)

LaVille, 1998

SIMPLE_RDT(q_0)

$\mathcal{G}$.init(q_0);
for $i = 1$ **to** k **do**
    $\mathcal{G}$.add_vertex($\alpha(i)$);
    $q_n \leftarrow$ NEAREST($S(\mathcal{G}), \alpha(i)$);
    $\mathcal{G}$.add_edge($q_n, \alpha(i)$);



(a) (b)

Rapidly Exploring Dense Tree (RDT)

LaValle, 1998

SIMPLE_RDT(q_0)

$\mathcal{G}$.init(q_0);
for $i = 1$ **to** k **do**
    $\mathcal{G}$.add_vertex($\alpha(i)$);
    $q_n \leftarrow$ NEAREST($S(\mathcal{G}), \alpha(i)$);
    $\mathcal{G}$.add_edge($q_n, \alpha(i)$);

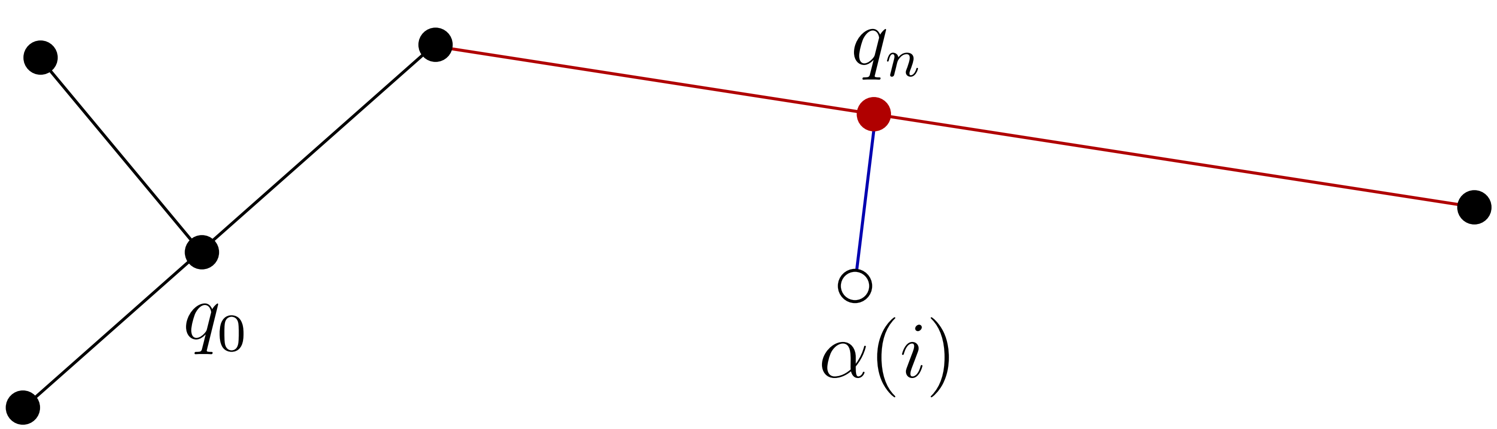


Figure 5.18: If the nearest point in S lies in an edge, then the edge is split into two, and a new vertex is inserted into $\mathcal{G}$.

Rapidly Exploring Dense Tree (RDT)

LaVallé, 1998

SIMPLE_RDT(q_0)

$\mathcal{G}$.init(q_0);
for $i = 1$ **to** k **do**
    $\mathcal{G}$.add_vertex($\alpha(i)$);
    $q_n \leftarrow$ NEAREST($S(\mathcal{G}), \alpha(i)$);
    $\mathcal{G}$.add_edge($q_n, \alpha(i)$);

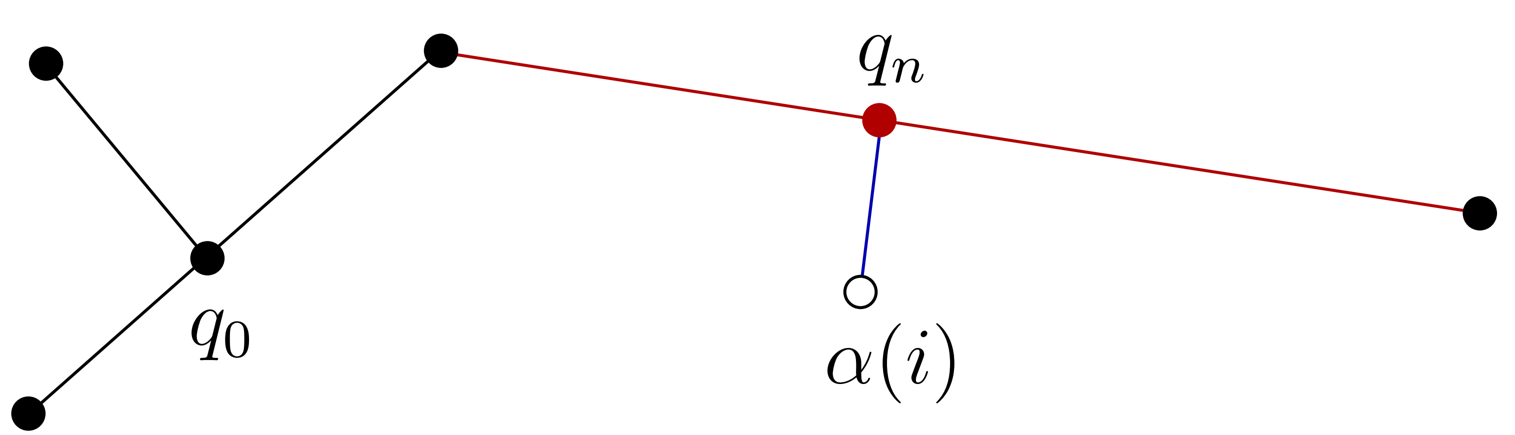
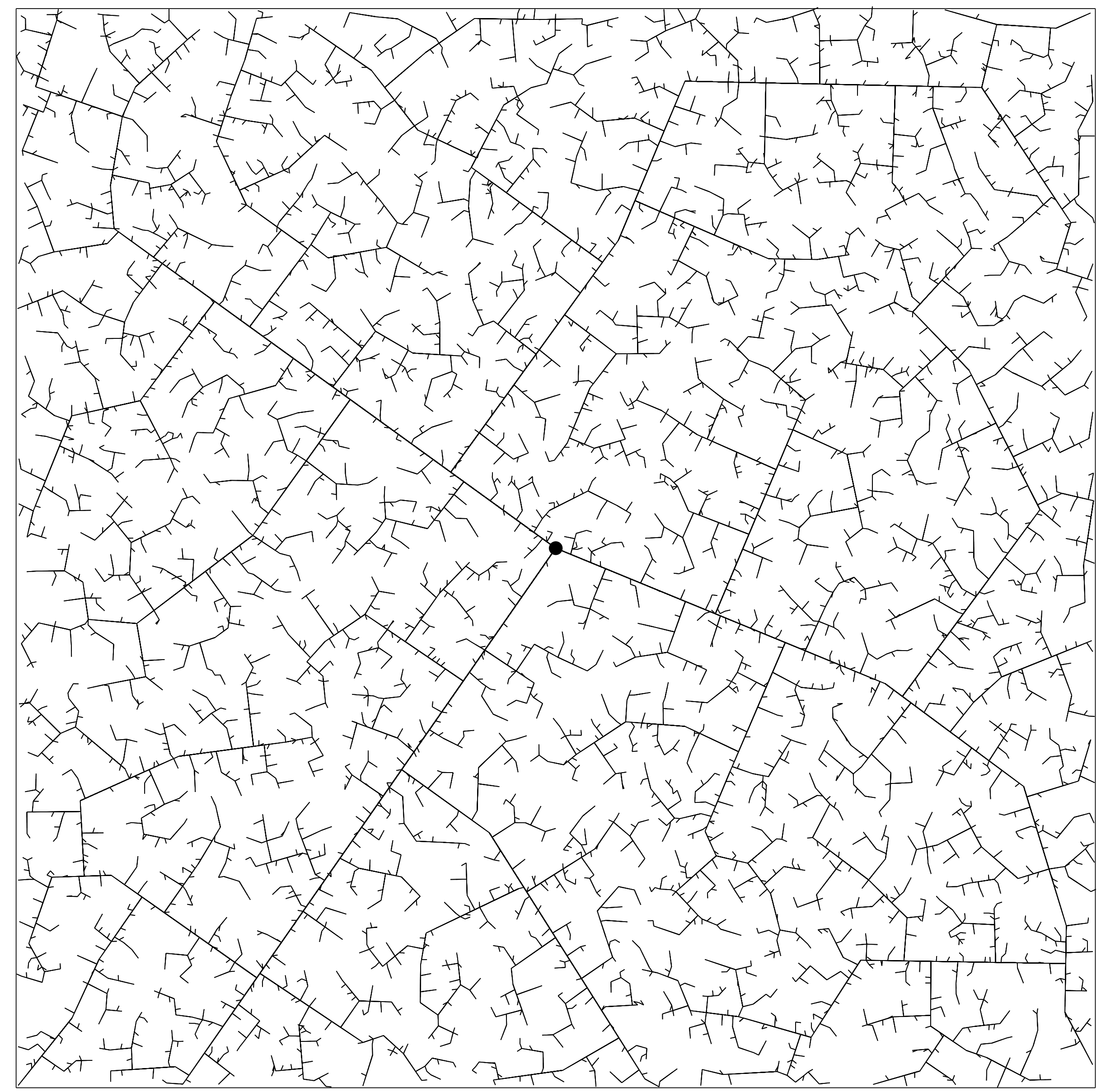


Figure 5.18: If the nearest point in S lies in an edge, then the edge is split into two, and a new vertex is inserted into $\mathcal{G}$.

RDT with iterations

LaValle, 1998

45 iterations

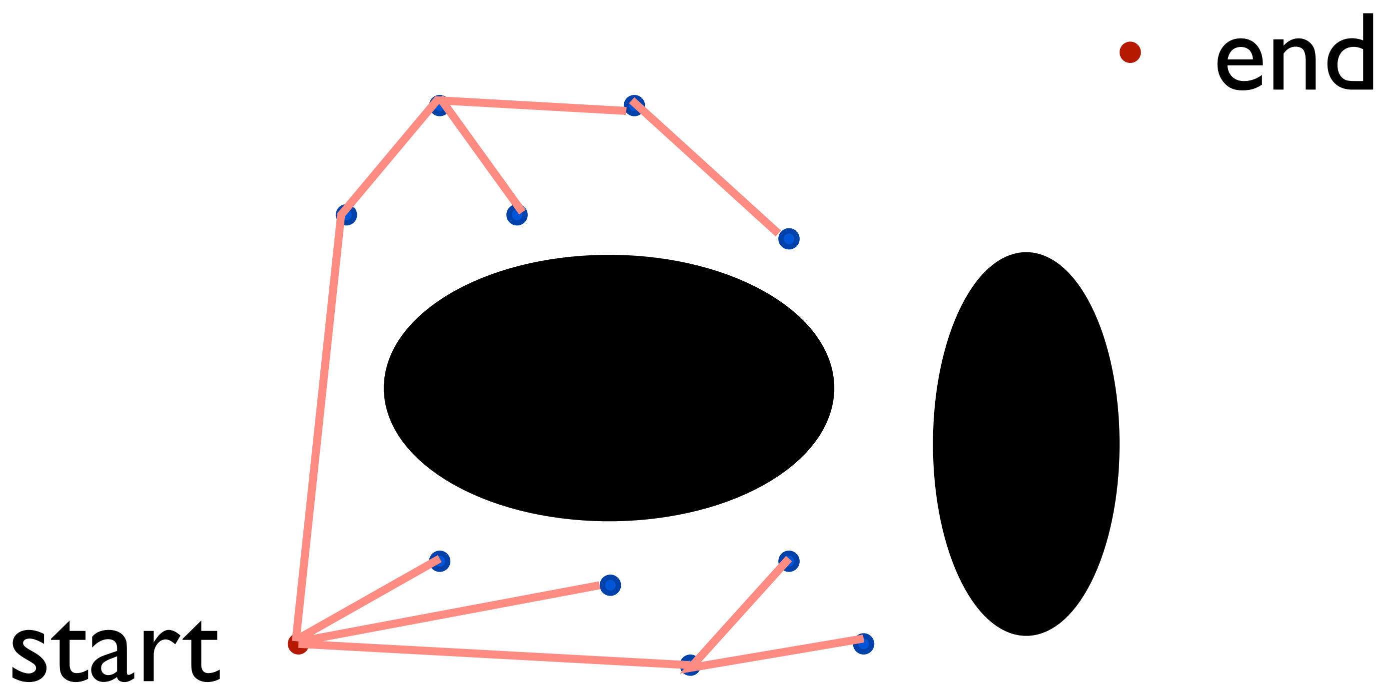


2345 iterations

RRT* (Karaman and Frazolli, 2010)

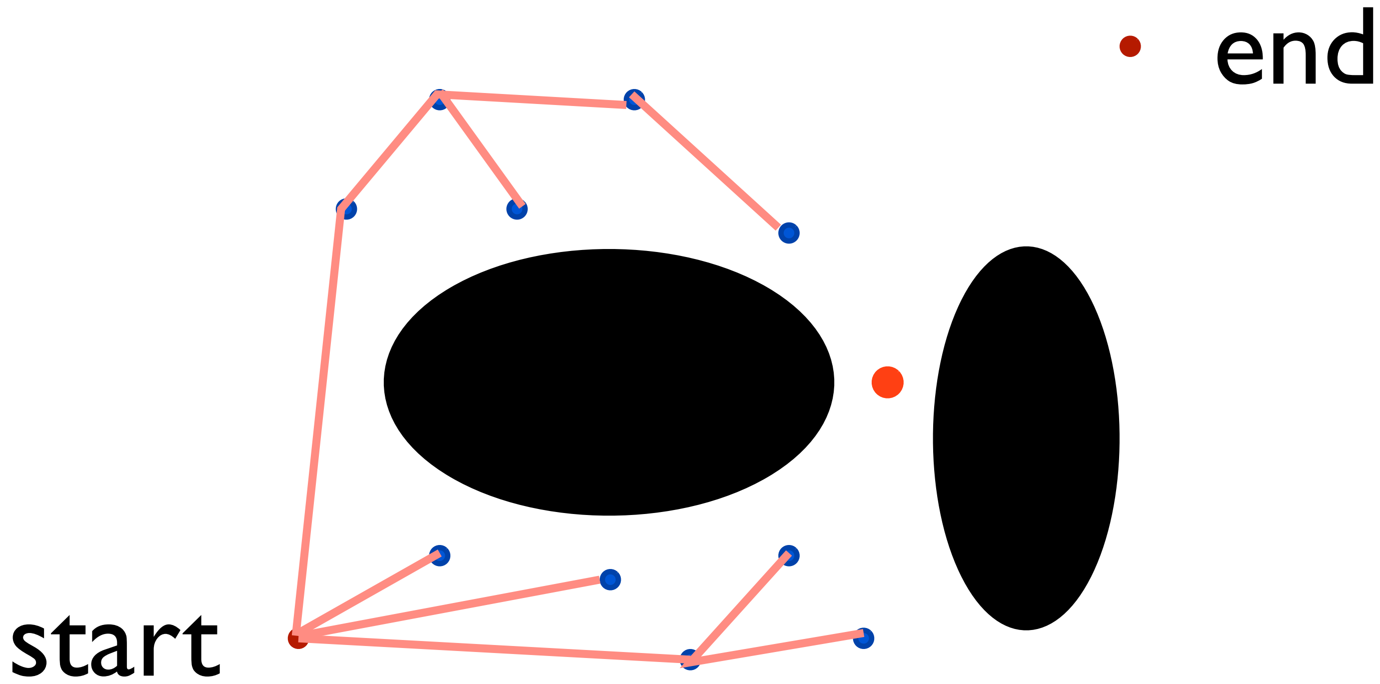
RRT* (Karaman and Frazolli, 2010)

At the i^{th} iteration,



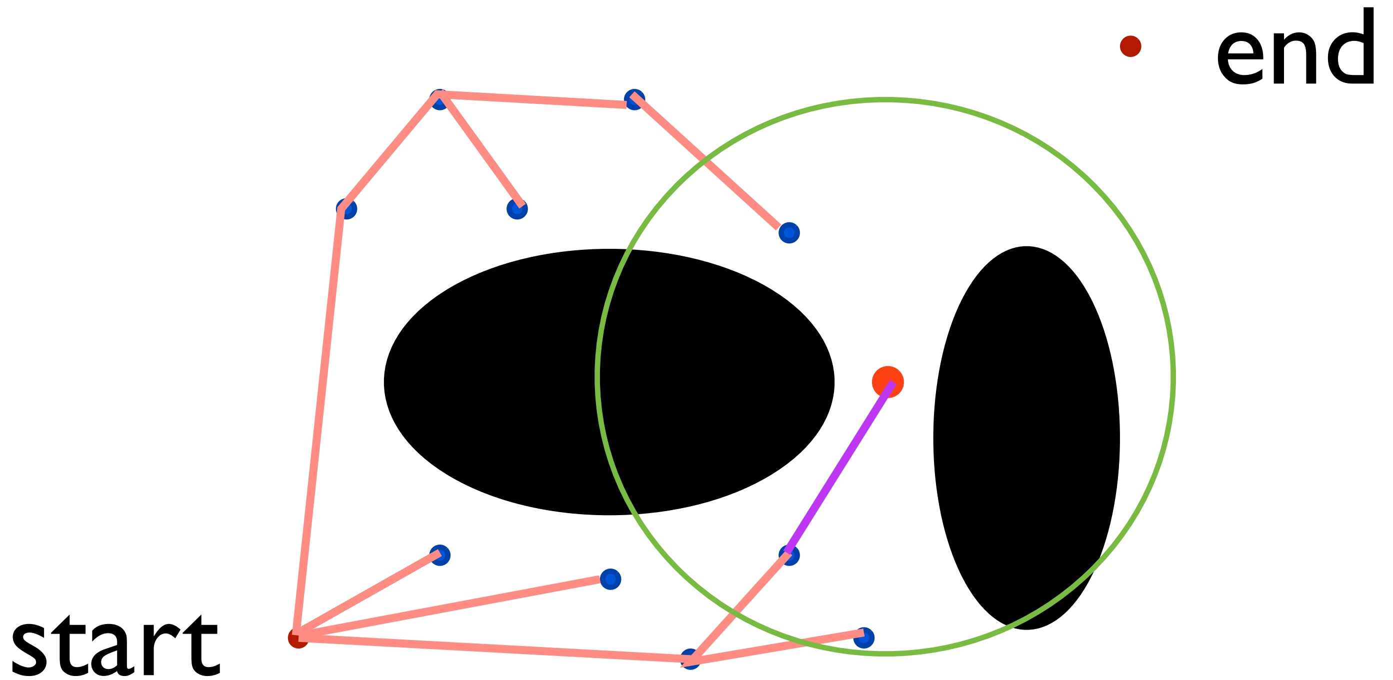
RRT* (Karaman and Frazolli, 2010)

At the i^{th} iteration,



SAMPLE

RRT* (Karaman and Frazolli, 2010)

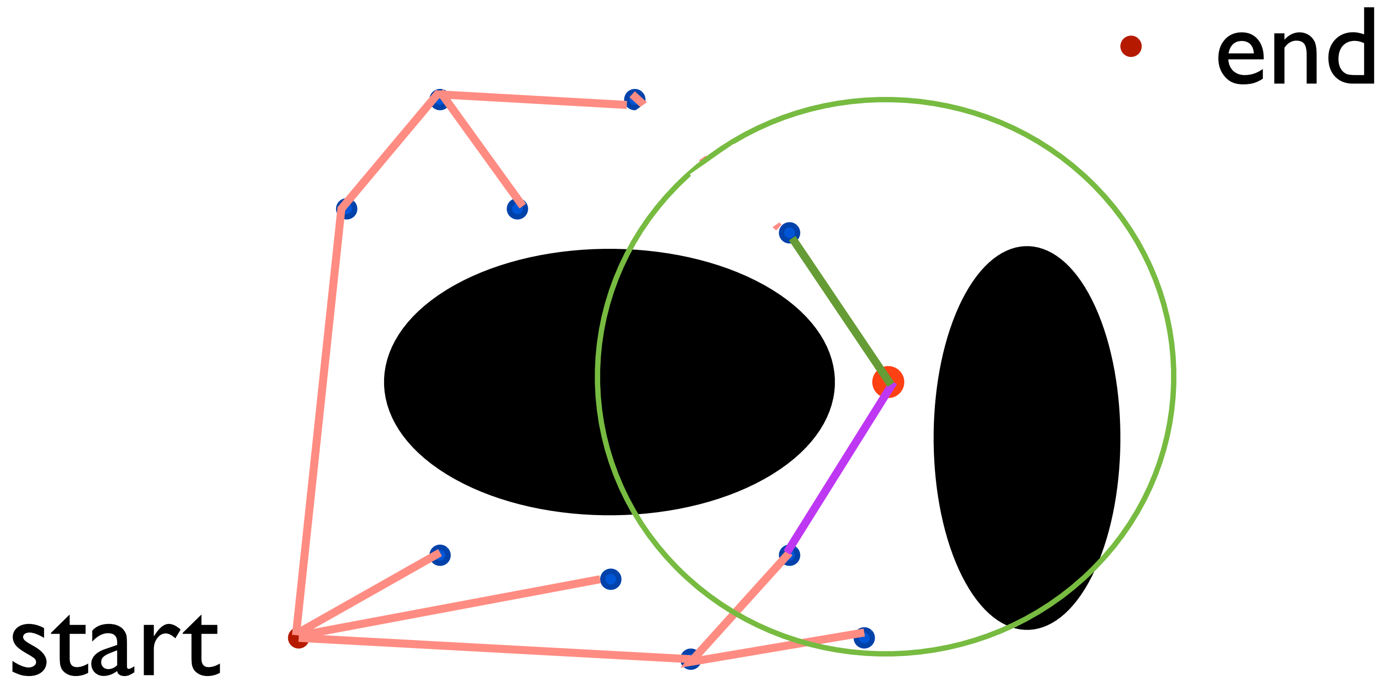


At the i^{th} iteration,

SAMPLE

FIND BEST PARENT

RRT* (Karaman and Frazolli, 2010)



At the i^{th} iteration,

SAMPLE

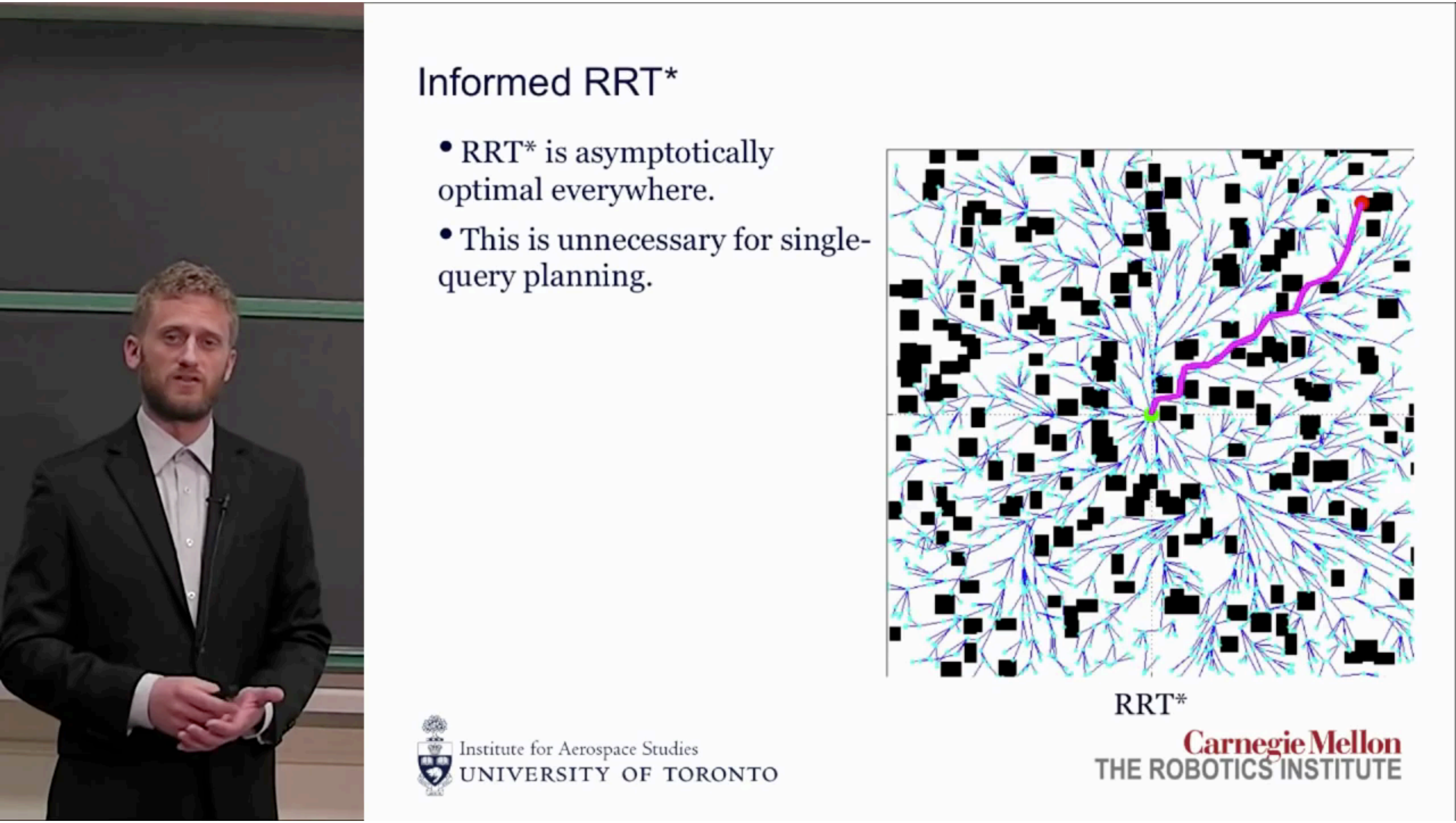
FIND BEST PARENT

REWIRE TO CHILDREN

RRT* is asynchronous Value Iteration!

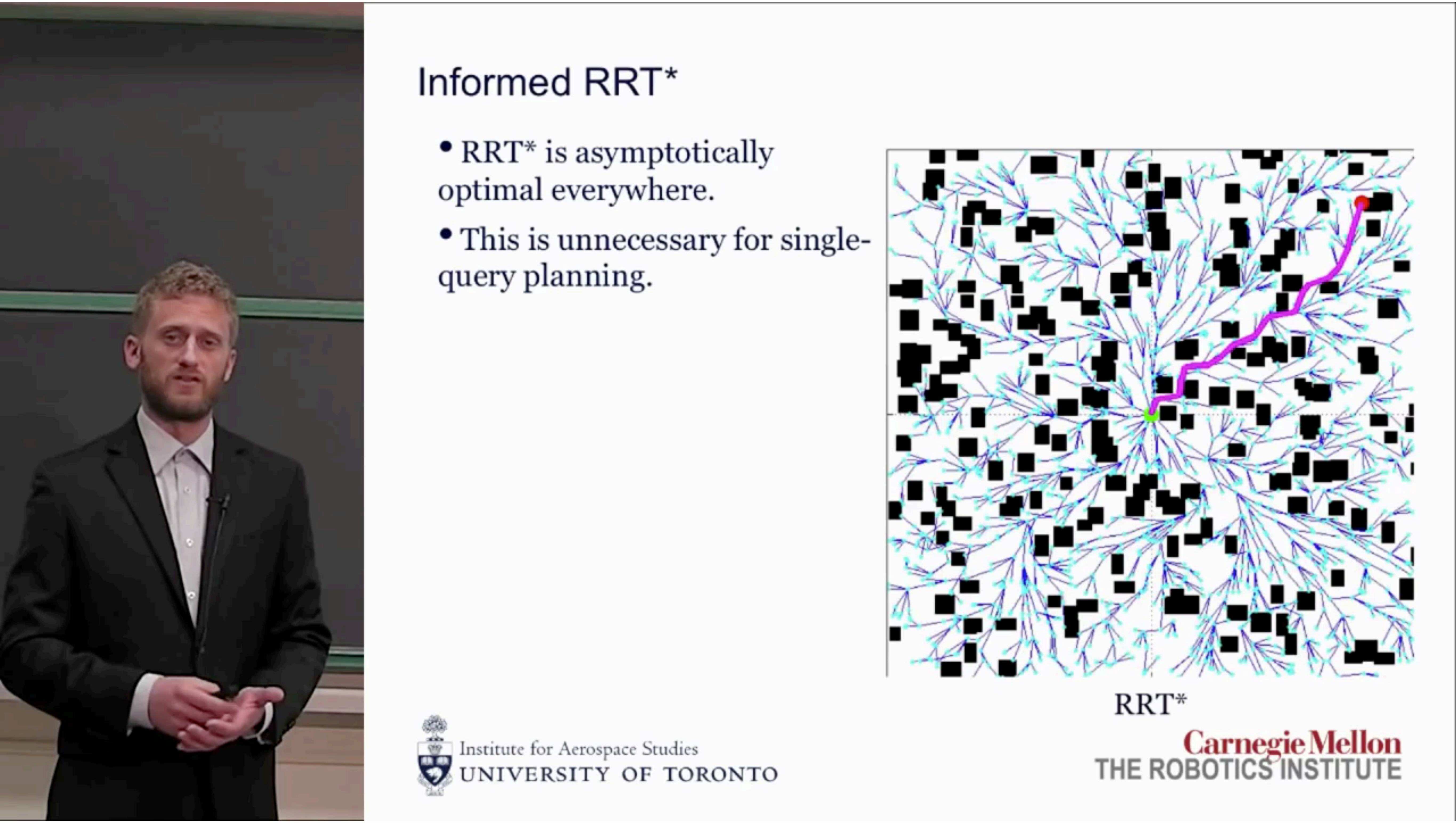
Can we do better?

Informed RRT*



https://www.youtube.com/watch?v=nsl-5MZfwu4&t=48s J. Gammell, S.Srinivasa, T.Barfoot, 2014

Informed RRT*



https://www.youtube.com/watch?v=nsl-5MZfwu4&t=48s J. Gammell, S.Srinivasa, T.Barfoot, 2014

Today's discussion

1. Why would we want to interleave?

2. How do we search when we interleave?

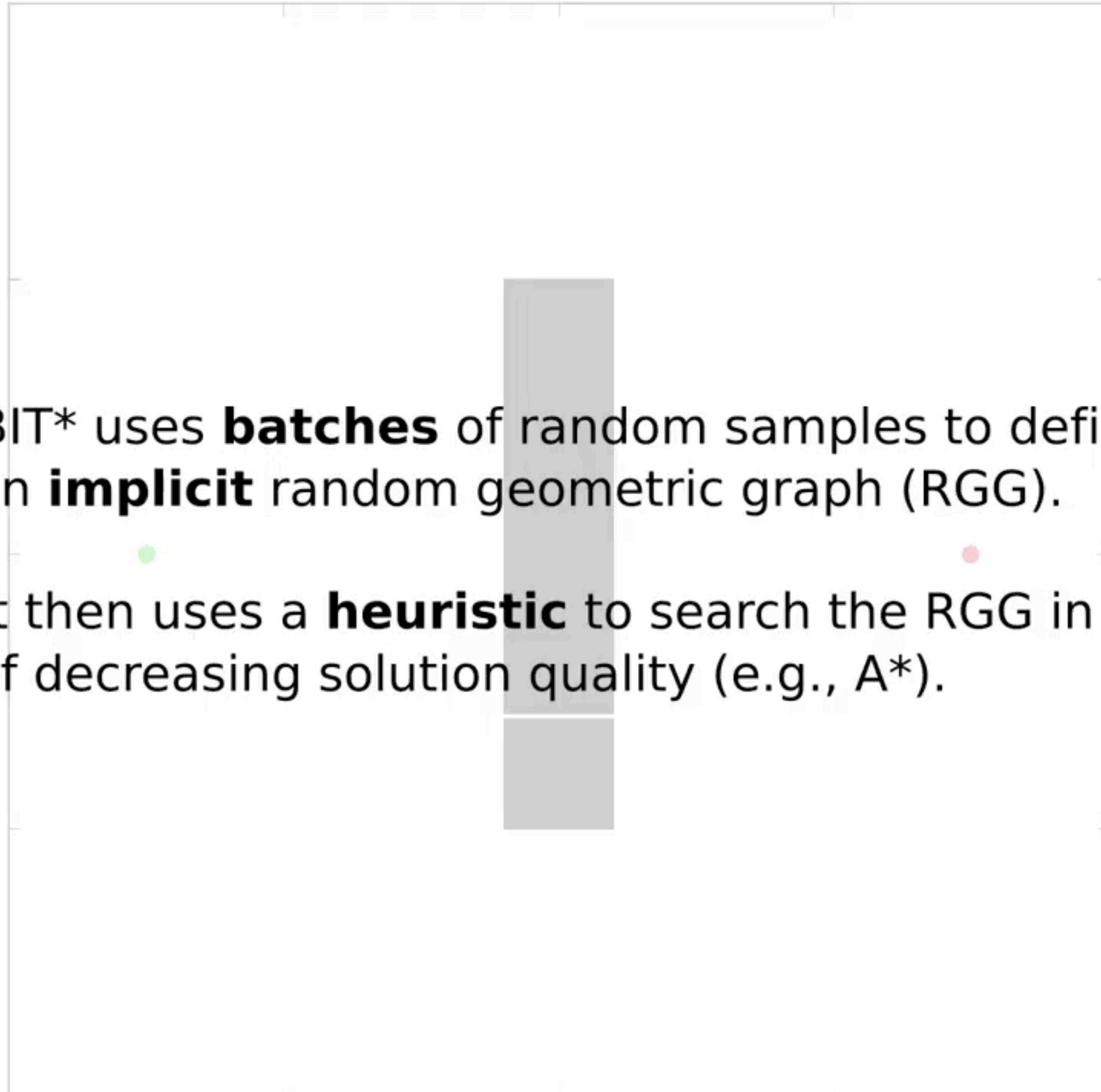
(repairing search)

3. How do we improve graphs when we interleave?

(incremental sampling)

4. Putting it all together

Batch Informed Trees

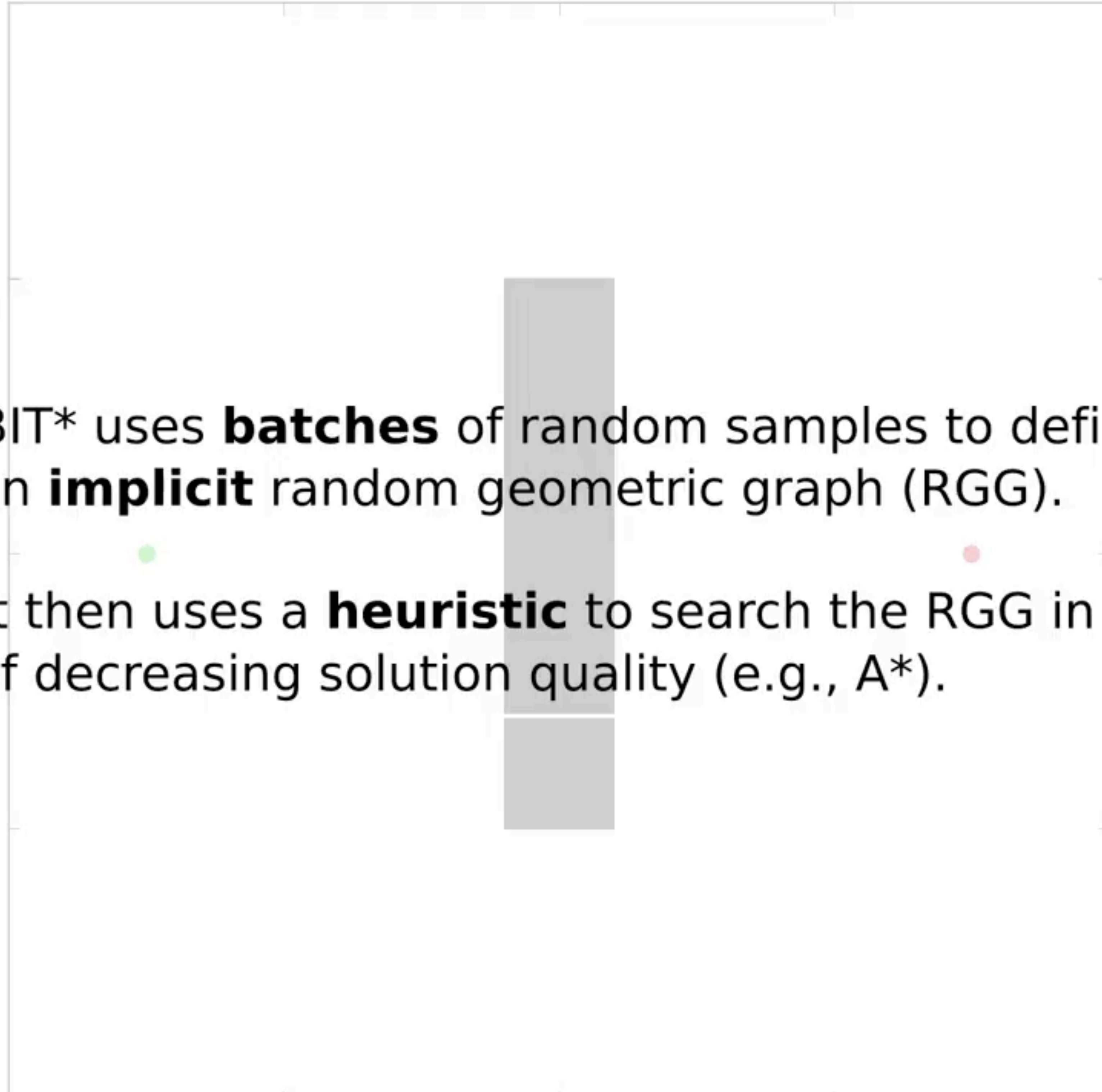


– BIT* uses **batches** of random samples to define an **implicit** random geometric graph (RGG).

– It then uses a **heuristic** to search the RGG in order of decreasing solution quality (e.g., A*).

https://www.youtube.com/watch?v=TQIoCC48gp4 J. Gammell, S.Srinivasa, T.Barfoot, 2015

Batch Informed Trees



https://www.youtube.com/watch?v=TQIoCC48gp4 J. Gammell, S.Srinivasa, T.Barfoot, 2015