Sub-Goal Trees – a Framework for Goal-Based Reinforcement Learning

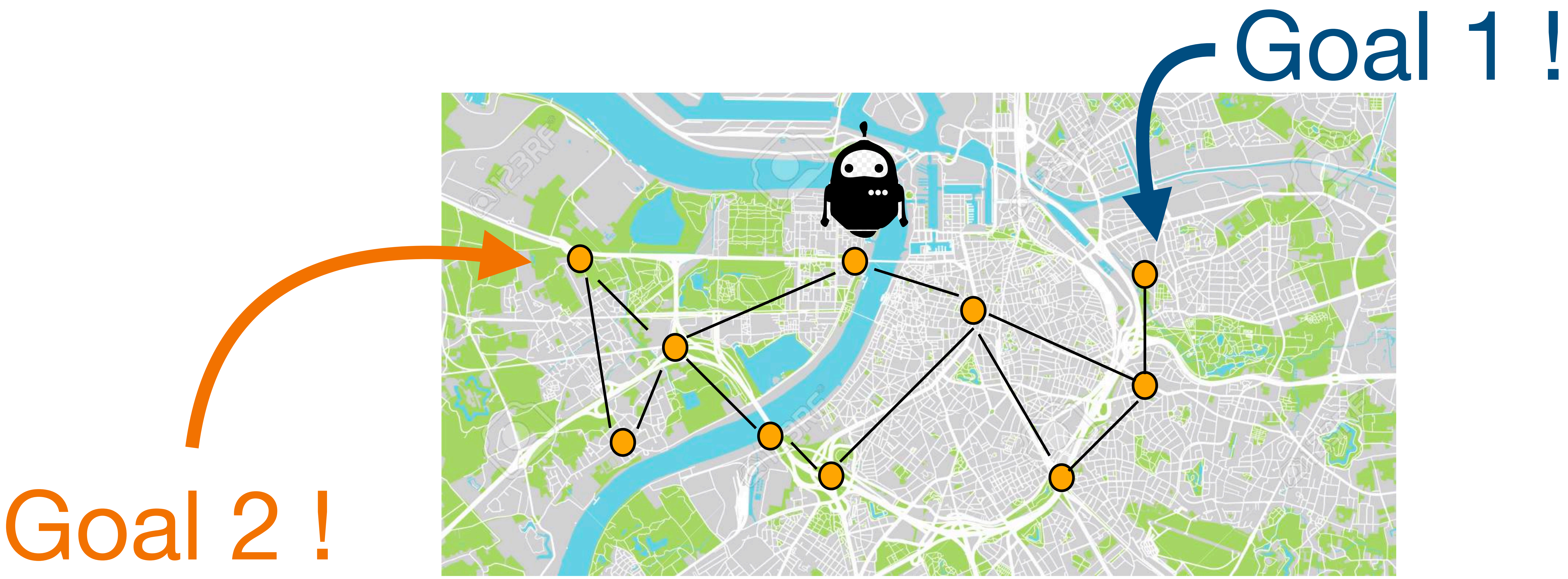
Tom Jurgenson, Or Avner, Edward Groshev, Aviv Tamar

(RL)²
Reinforcement Learning
Research Labs

הטכניון
מכון טכנולוגי
לישראל
TECHNION
Israel Institute
of Technology

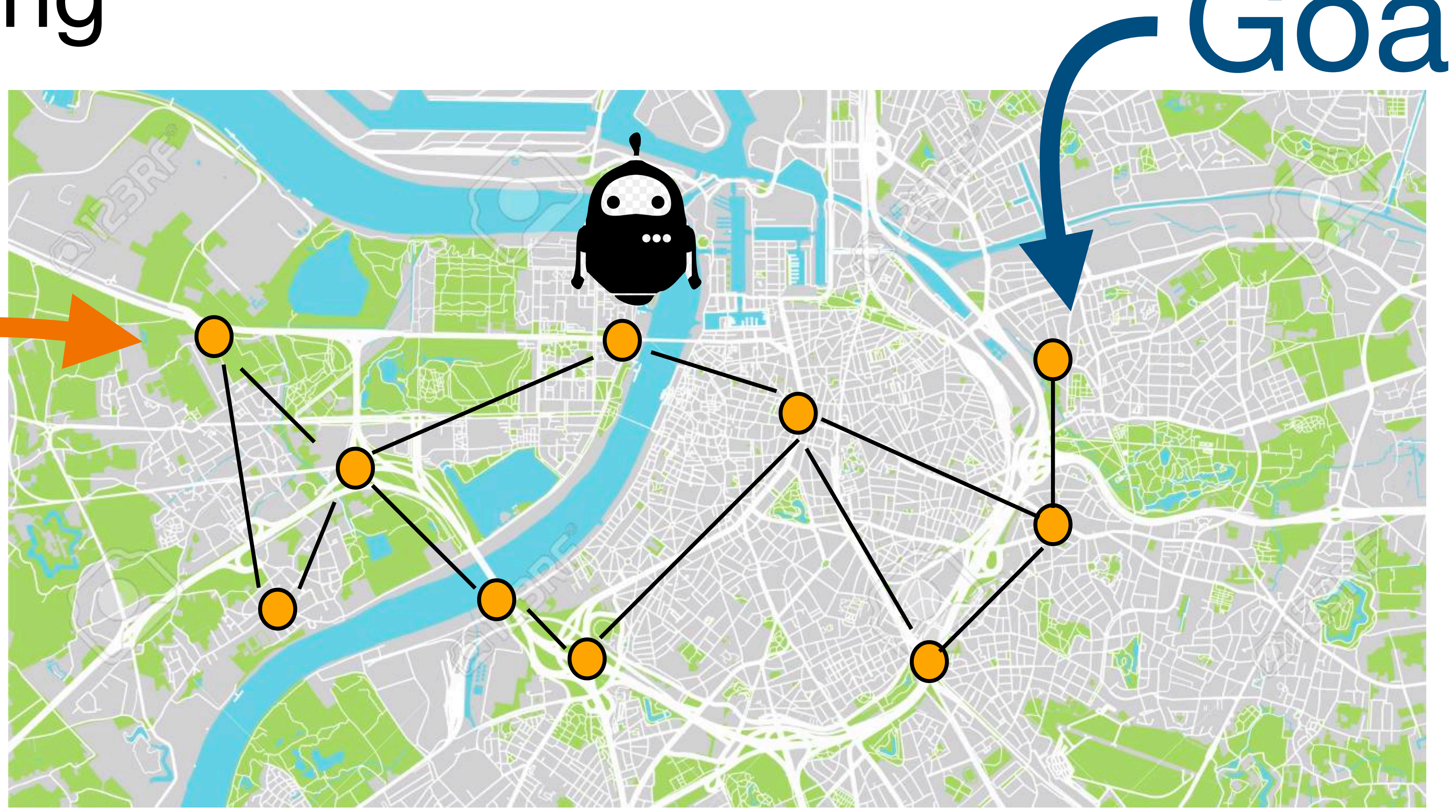
Motivation: multi-goal reinforcement learning

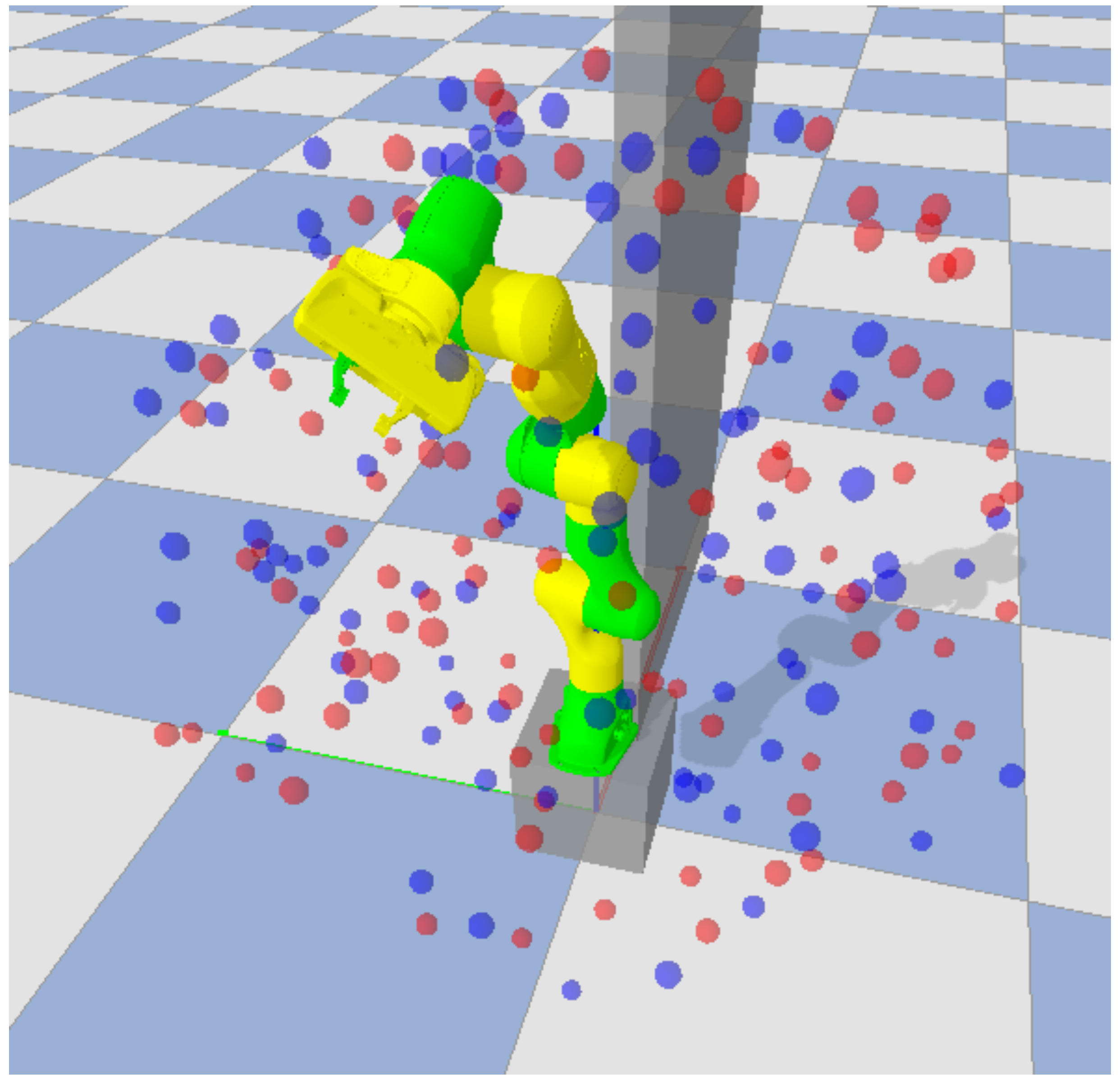
- **Multi-goal task**: agent needs to reach **different goals**



Motivation: multi-goal reinforcement learning

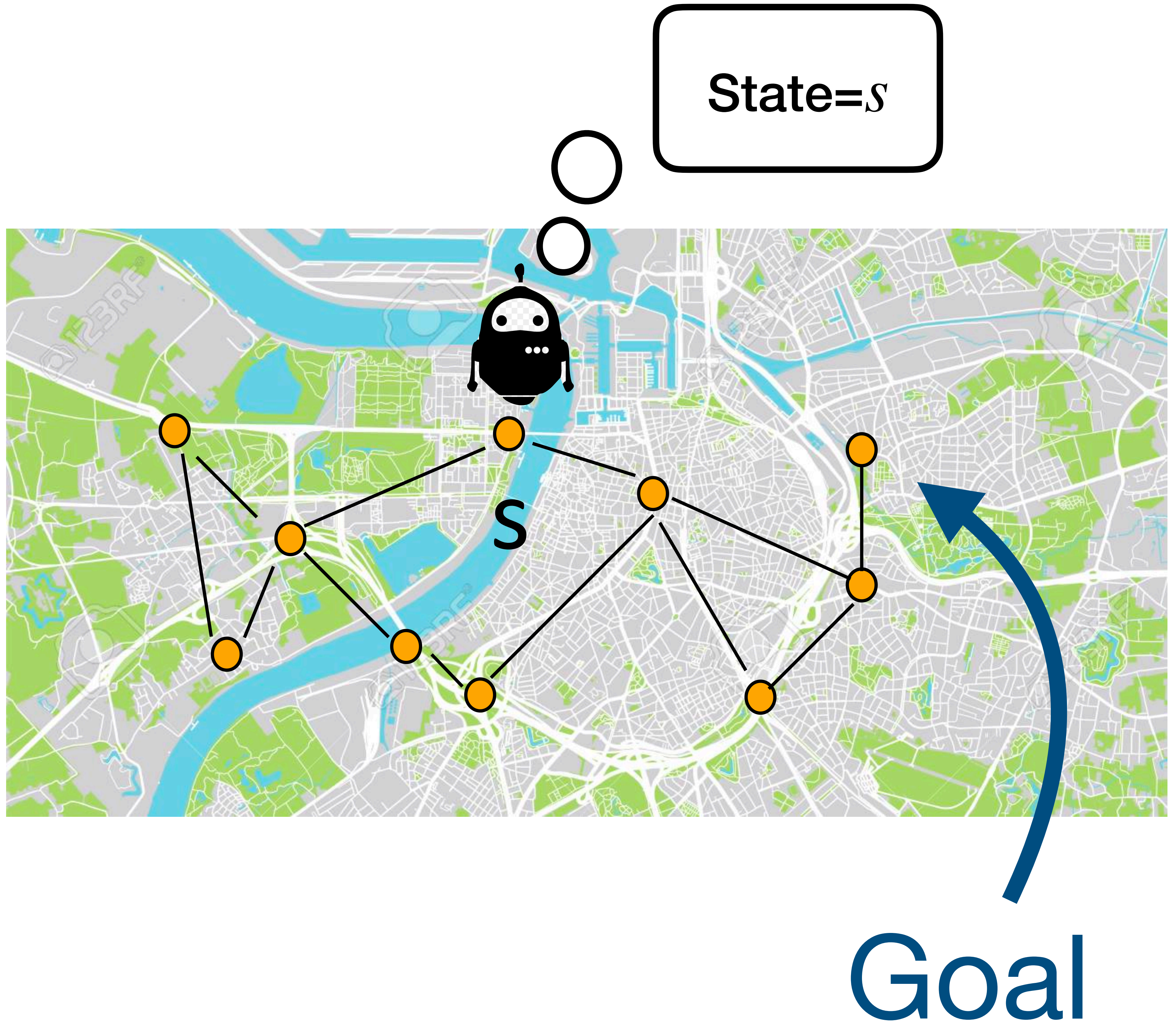
- **Multi-goal task**: agent needs to reach **different goals**
- Frequently encountered in robotics / game-playing etc. :
  - Motion planning
  - Skill learning





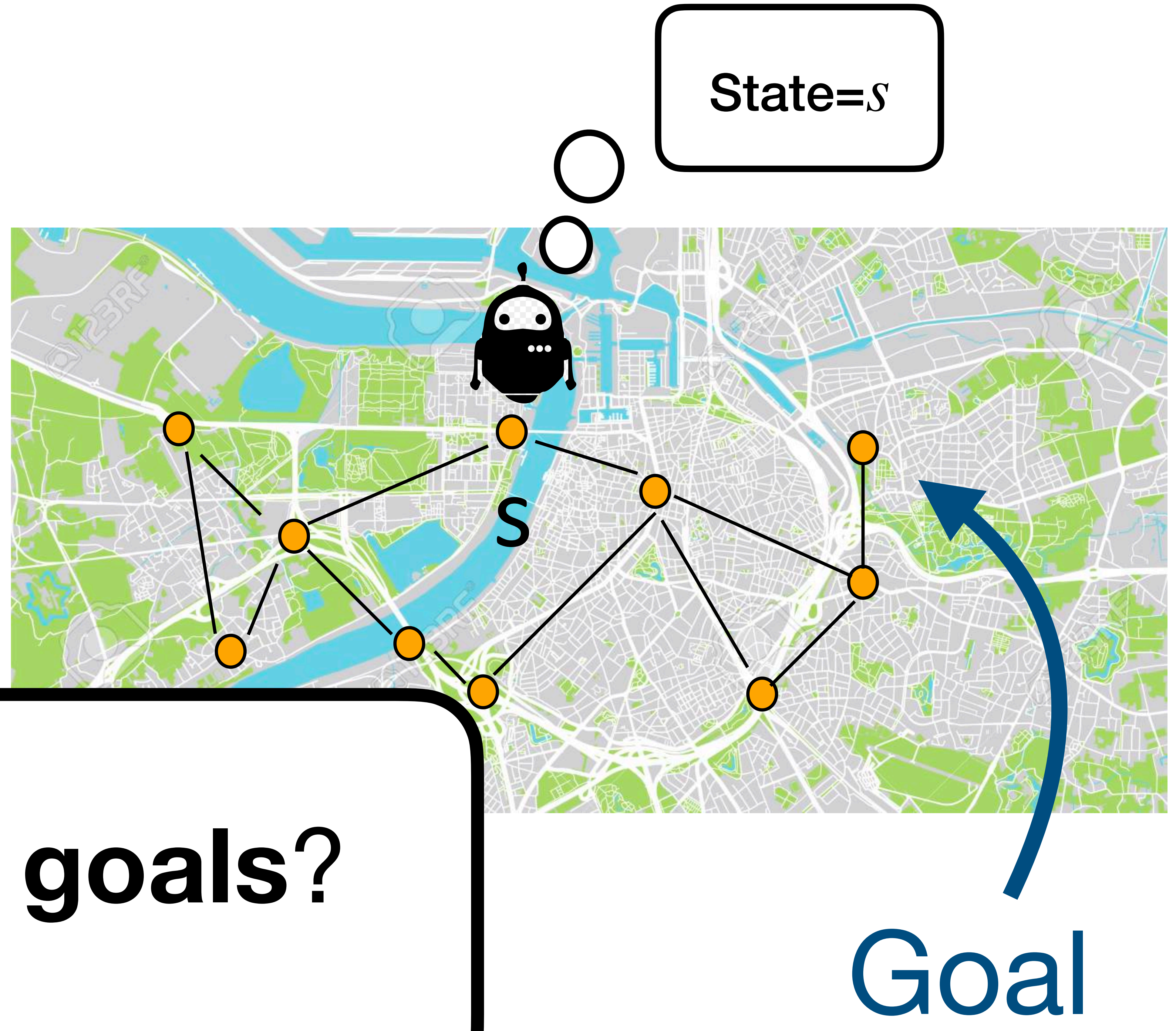
Motivation: single —> multi goal-RL

RL is **single-goal** (cost function)



Motivation: single —> multi goal-RL

RL is **single-goal** (cost function)



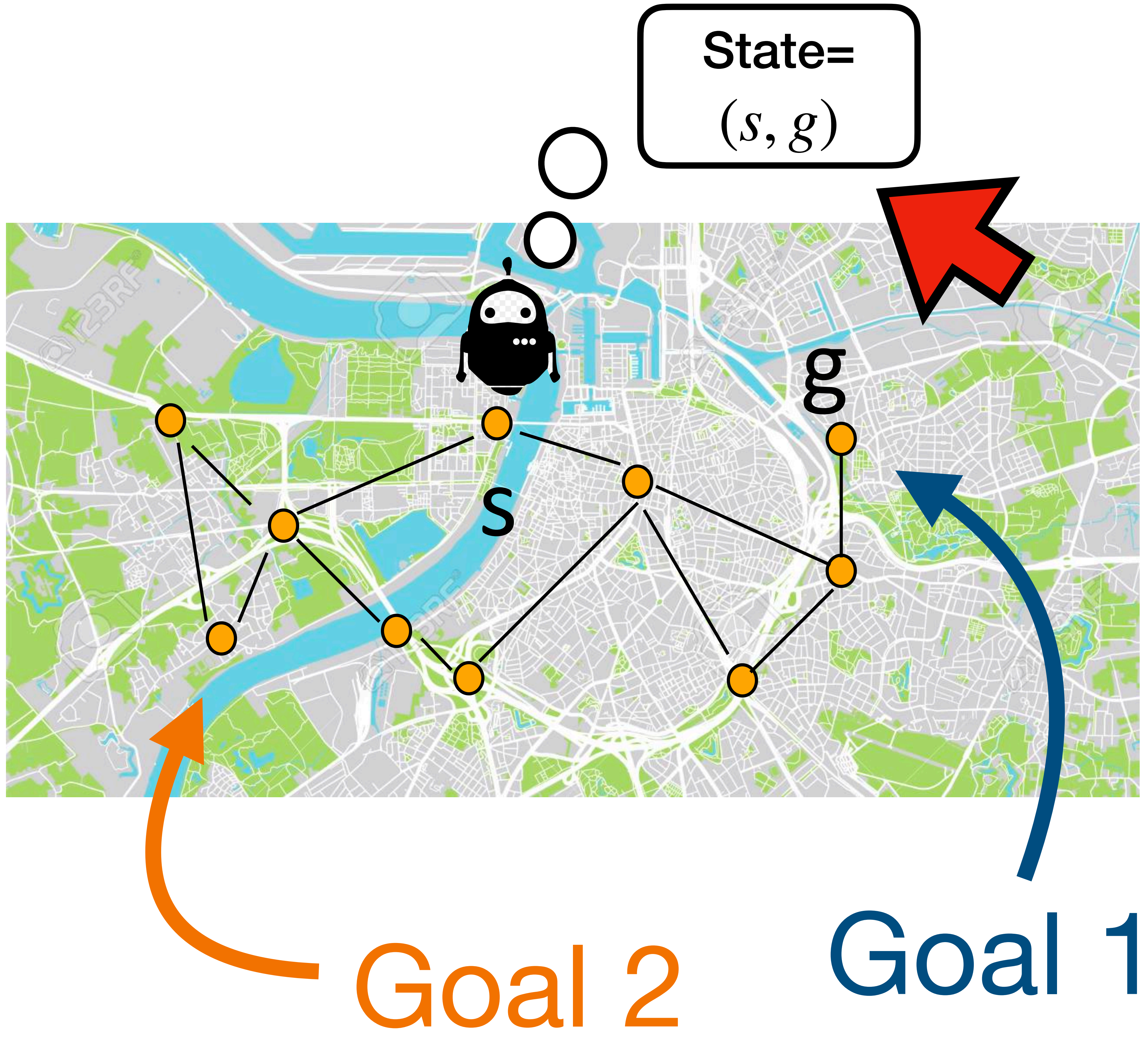
How to extend to **multiple goals**?

Motivation: single —> multi goal-RL

RL is **single-goal** (cost function)

So far: UVFA[1], HER[2], many others

- Add goal → state observation
- Train using standard RL



[1] Schaul et al, Universal Value Function Approximators, 2015
[2] Andrychowicz et al. Hindsight Experience Replay, 2017

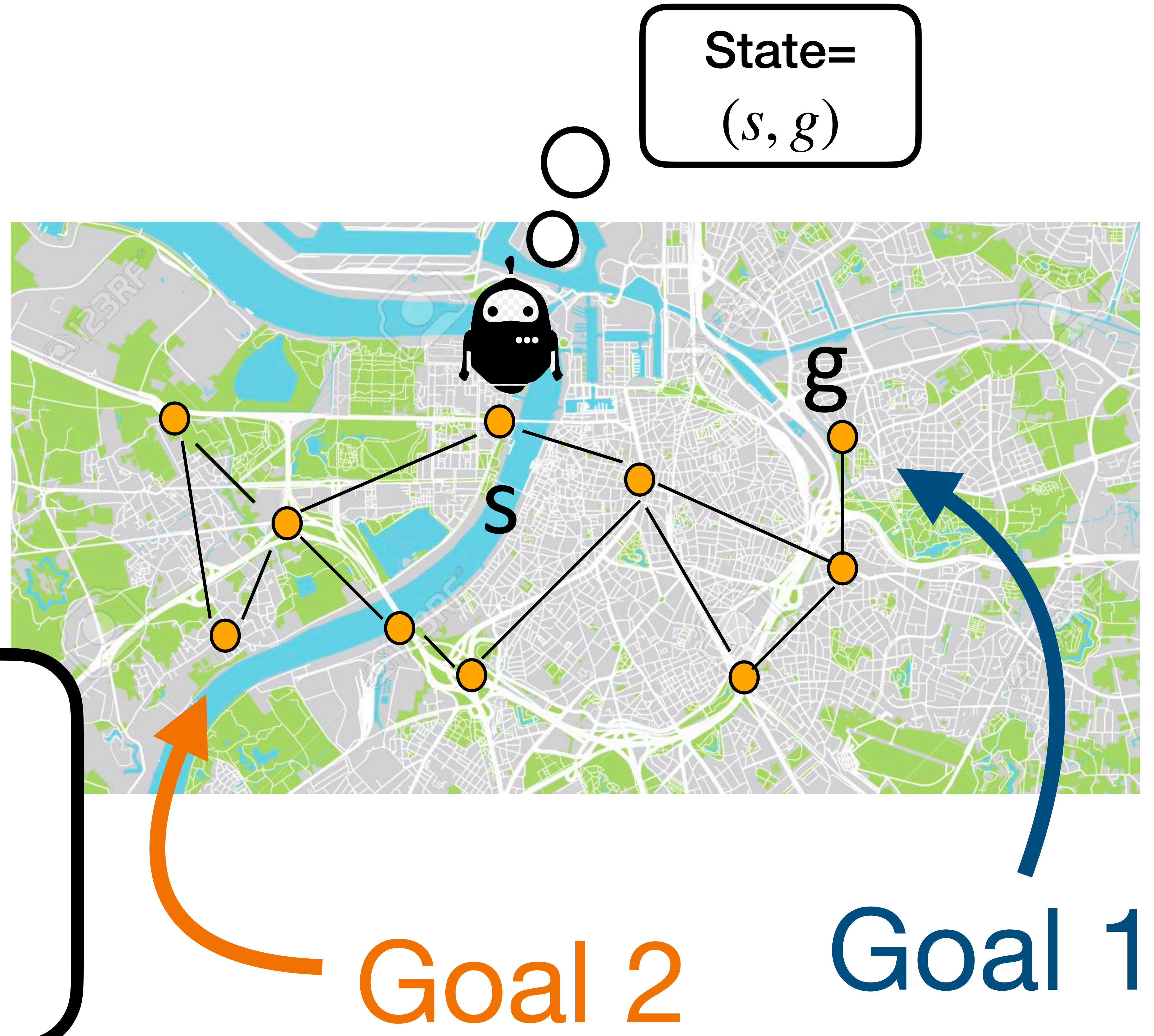
Motivation: single —> multi goal-RL

RL is **single-goal** (cost function)

So far: UVFA[1], HE

- Add goal →
- Train using

“Bellman RL”



[1] Schaul et al, Universal Value Function Approximators, 2015

[2] Andrychowicz et al. Hindsight Experience Replay, 2017

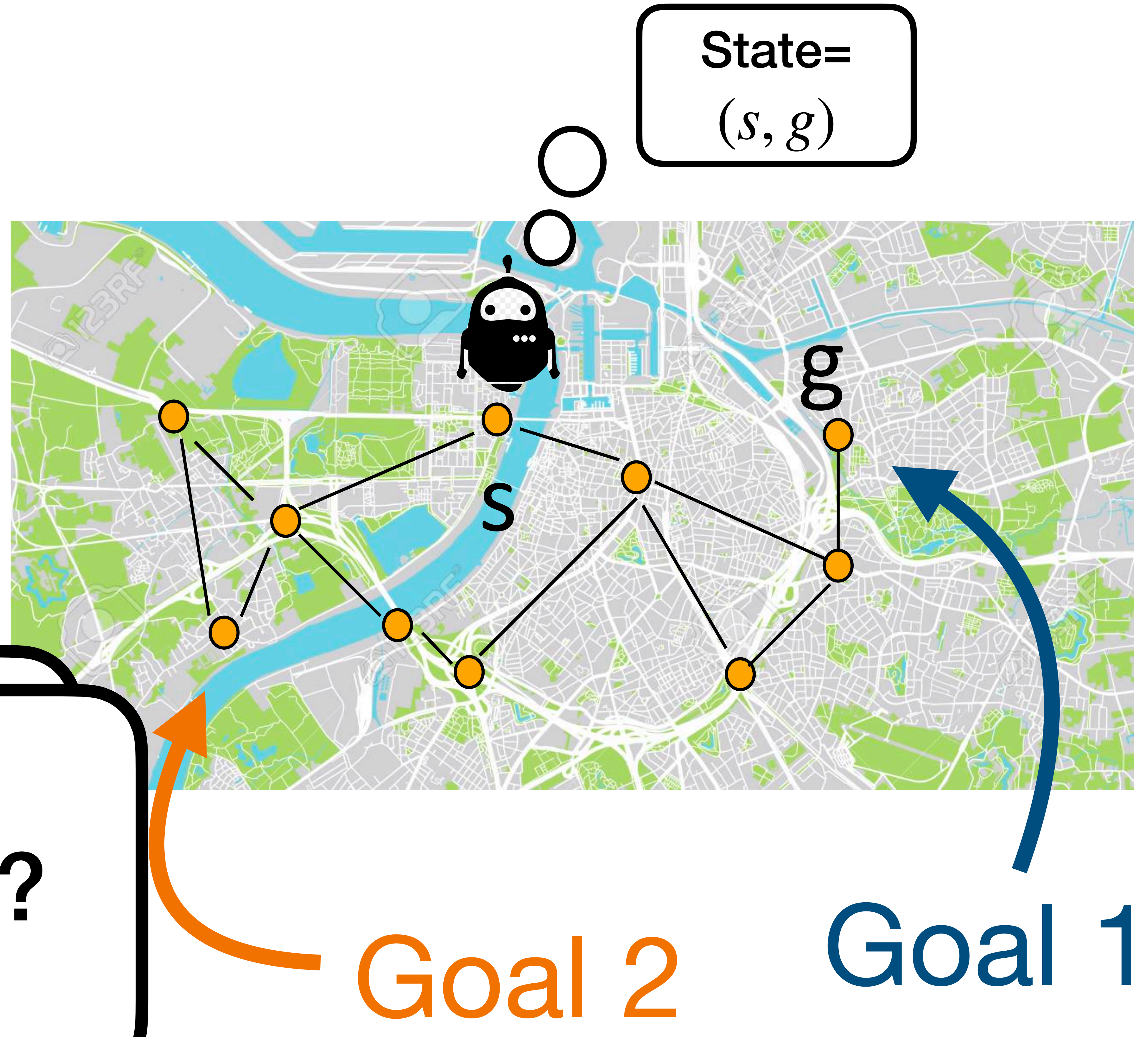
Motivation: single —> multi goal-RL

RL is **single-goal** (cost function)

So far: UVFA[1], HE

- Add goal →
- Train using s

Is that optimal?



[1] Schaul et al, Universal Value Function Approximators, 2015
[2] Andrychowicz et al. Hindsight Experience Replay, 2017

Intuition: utilizing problem structure?

Consider: **All-Pairs-Shortest-Path** (APSP) problem

- Bellman RL approach = Bellman-Ford on all starts

Intuition: utilizing problem structure?

Consider: **All-Pairs-Shortest-Path** (APSP) problem

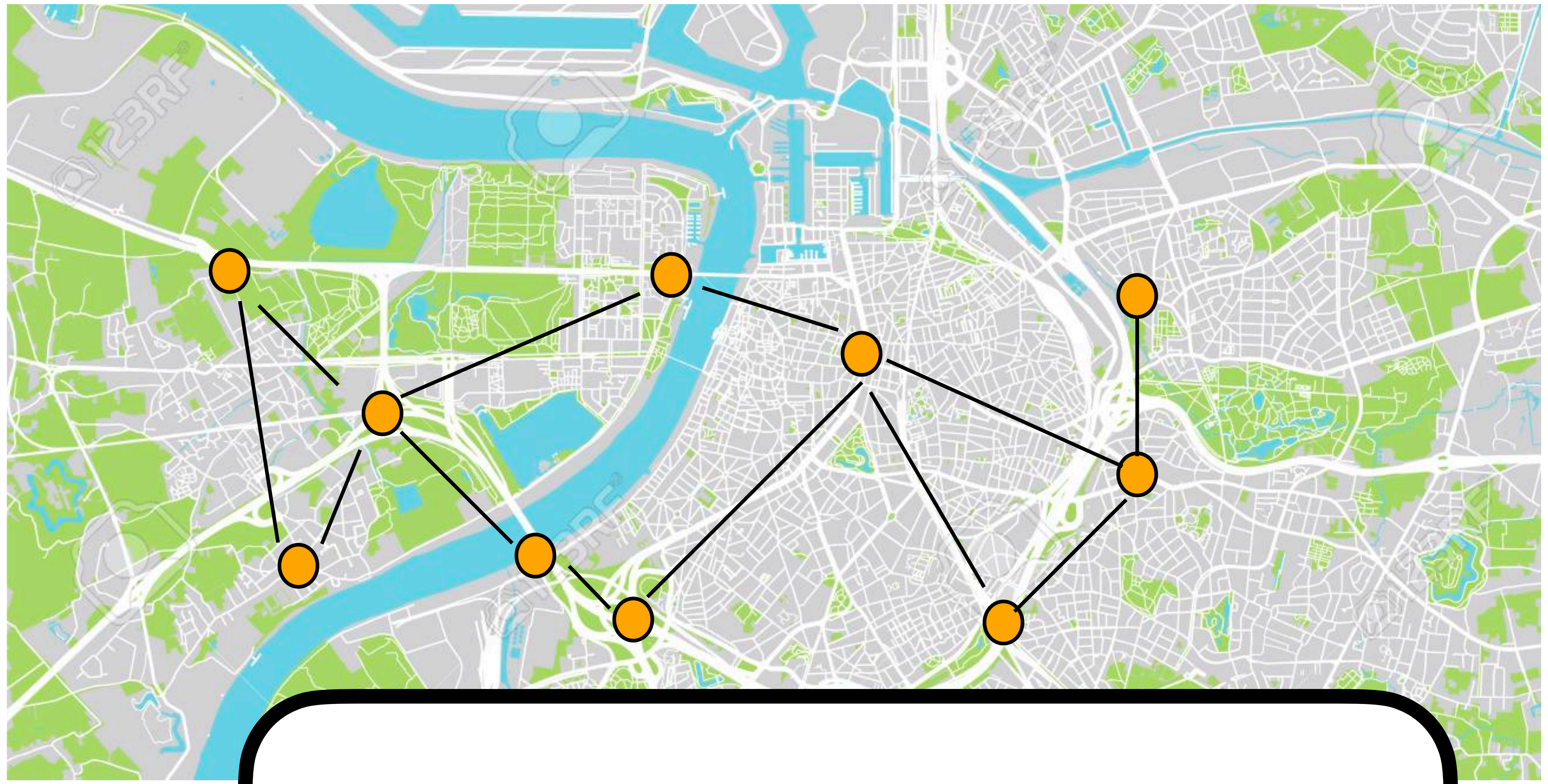
- Bellman RL approach = Bellman-Ford on all starts $\rightarrow O\left(|V|^4\right)$

Intuition: utilizing problem structure?

Consider: **All-Pairs-Shortest-Path** (APSP) problem

- Bellman RL approach = Bellman-Ford on all starts $\rightarrow O\left(|V|^4\right)$
- Floyd-Warshal is faster! $\rightarrow O\left(|V|^3\right)$

Intuition: utilizing problem structure?



Consider: **All-Pairs-Sh**

Structure to the problem we can exploit!

- Bellman RL approa $\rightarrow O\left(|V|^4\right)$
- Floyd-Warshal is faster! $\rightarrow O\left(|V|^3\right)$

Our approach: Sub-goal Tree (SGT)

New APSP formulation for multi-goal RL w/o the Bellman-Eq.!

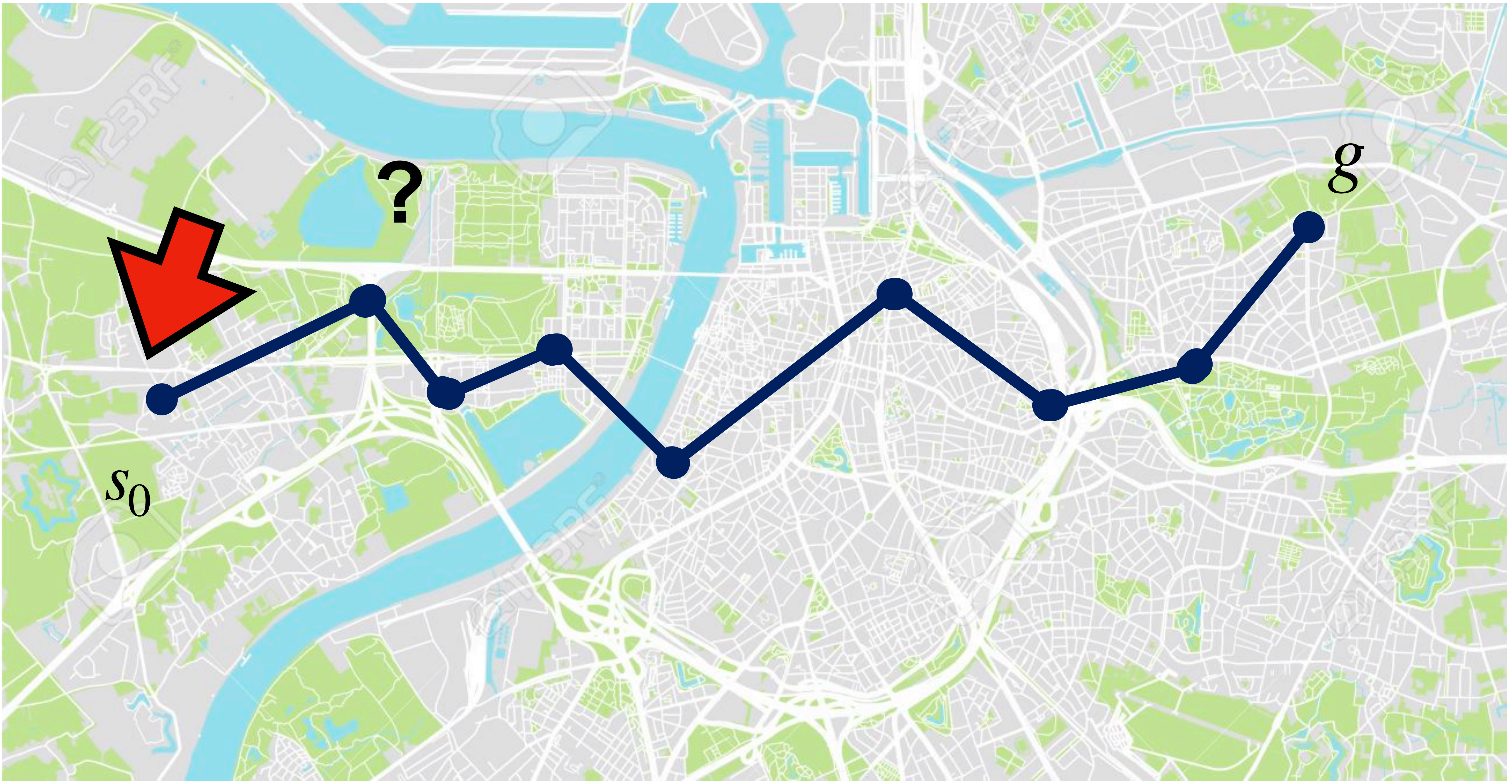
Our approach: Sub-goal Tree (SGT)

New APSP formulation for multi-goal RL w/o the Bellman-Eq.!

"Bellman" RL:

What is the **next** min cost state?

s_0, g



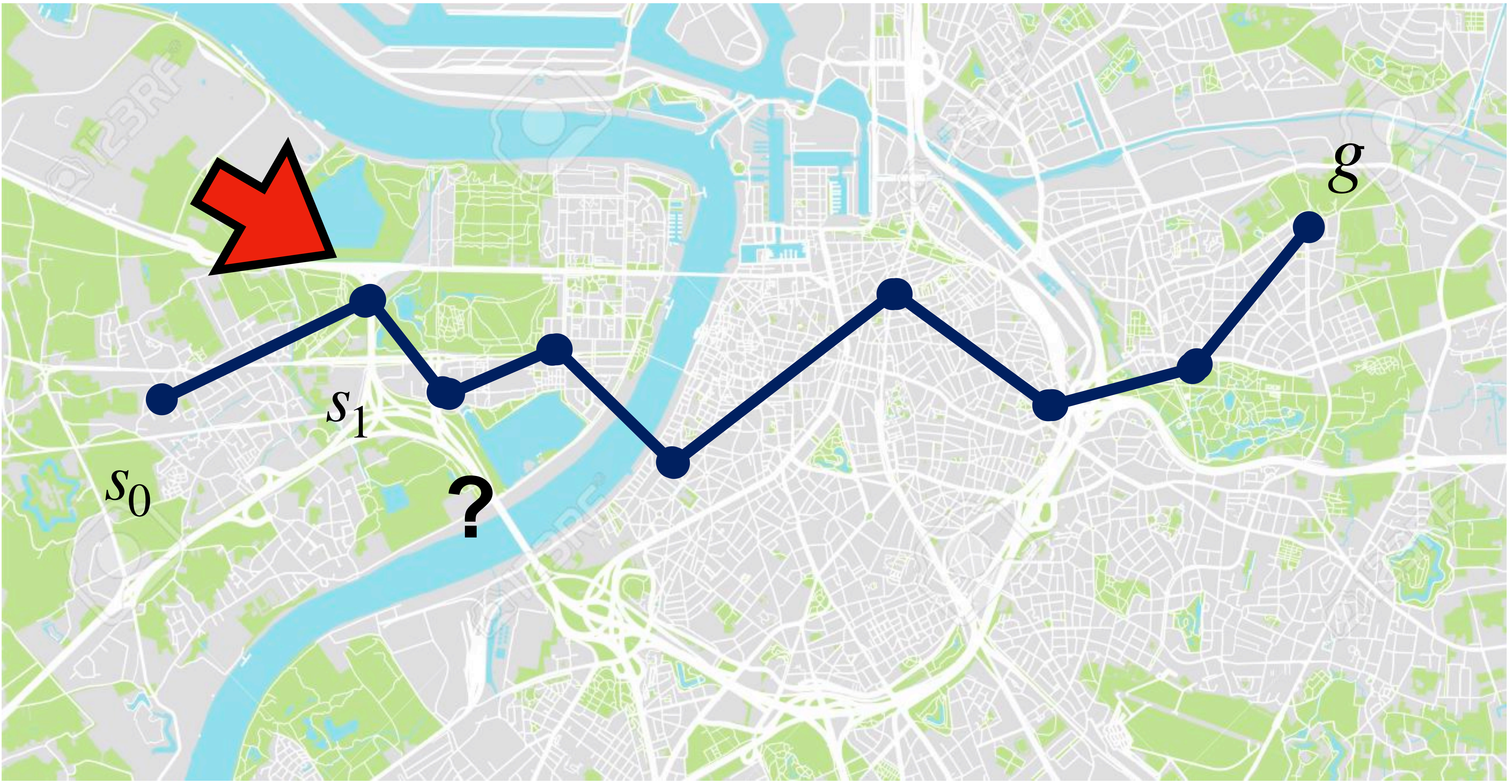
Our approach: Sub-goal Tree (SGT)

New APSP formulation for multi-goal RL w/o the Bellman-Eq.!

“Bellman” RL:

What is the **next** min cost state?

$s_0, g \rightarrow s_1$



Our approach: Sub-goal Tree (SGT)

New APSP formulation for multi-goal RL w/o the Bellman-Eq.!

“Bellman” RL:

What is the **next** min cost state?

$s_0, g \rightarrow$

$s_1, g \rightarrow$

s_2

Our approach: Sub-goal Tree (SGT)

New APSP formulation for multi-goal RL w/o the Bellman-Eq.!

“Bellman” RL:

What is the **next** min cost state?

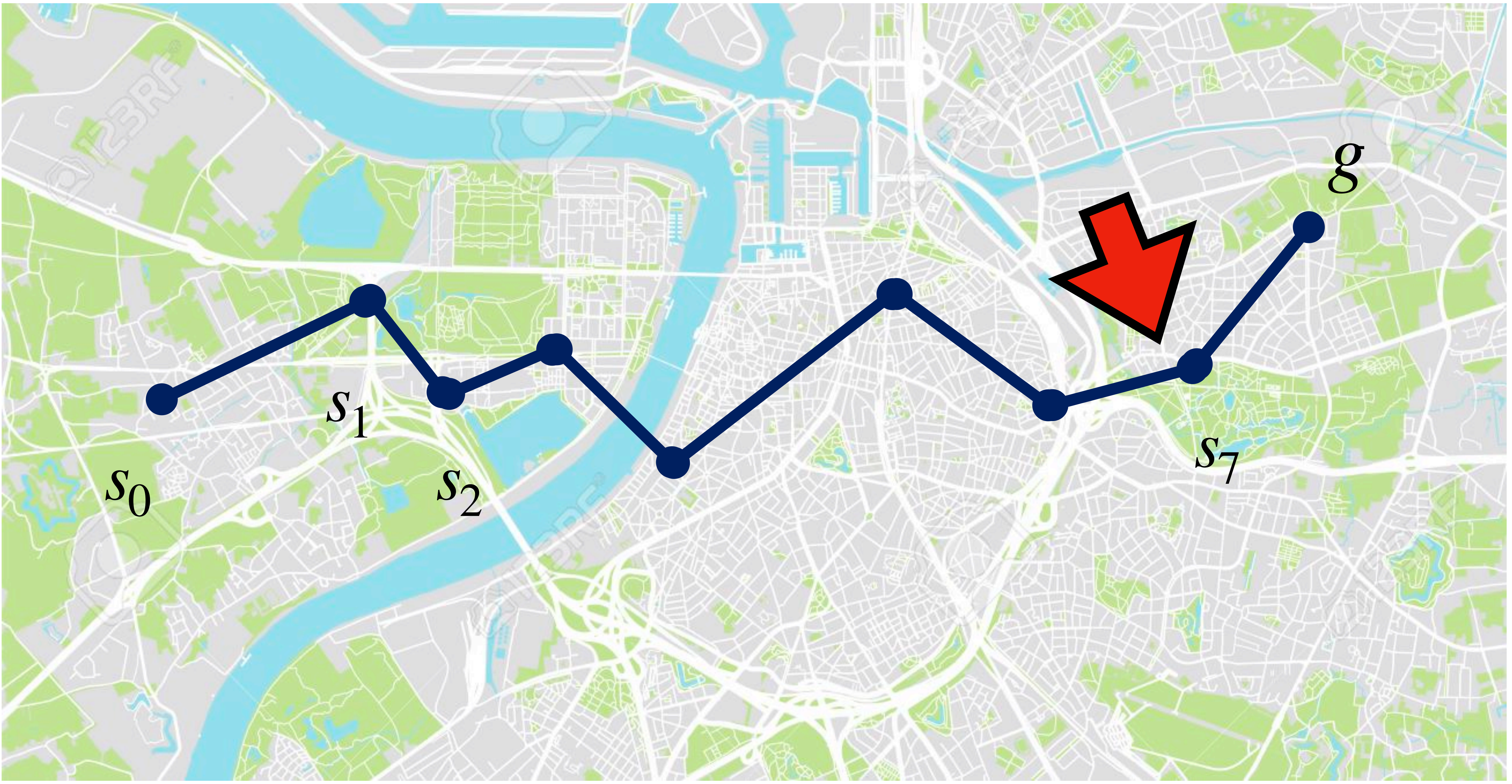
s_0, g ⤵

s_1, g ⤵

s_2, g ⤵

⤵

s_7



Our approach: Sub-goal Tree (SGT)

New APSP formulation for multi-goal RL w/o the Bellman-Eq.!

"SGT" RL:

What is the **middle** min cost state?

(middle state = **subgoal**)

s_0, g

Our approach: Sub-goal Tree (SGT)

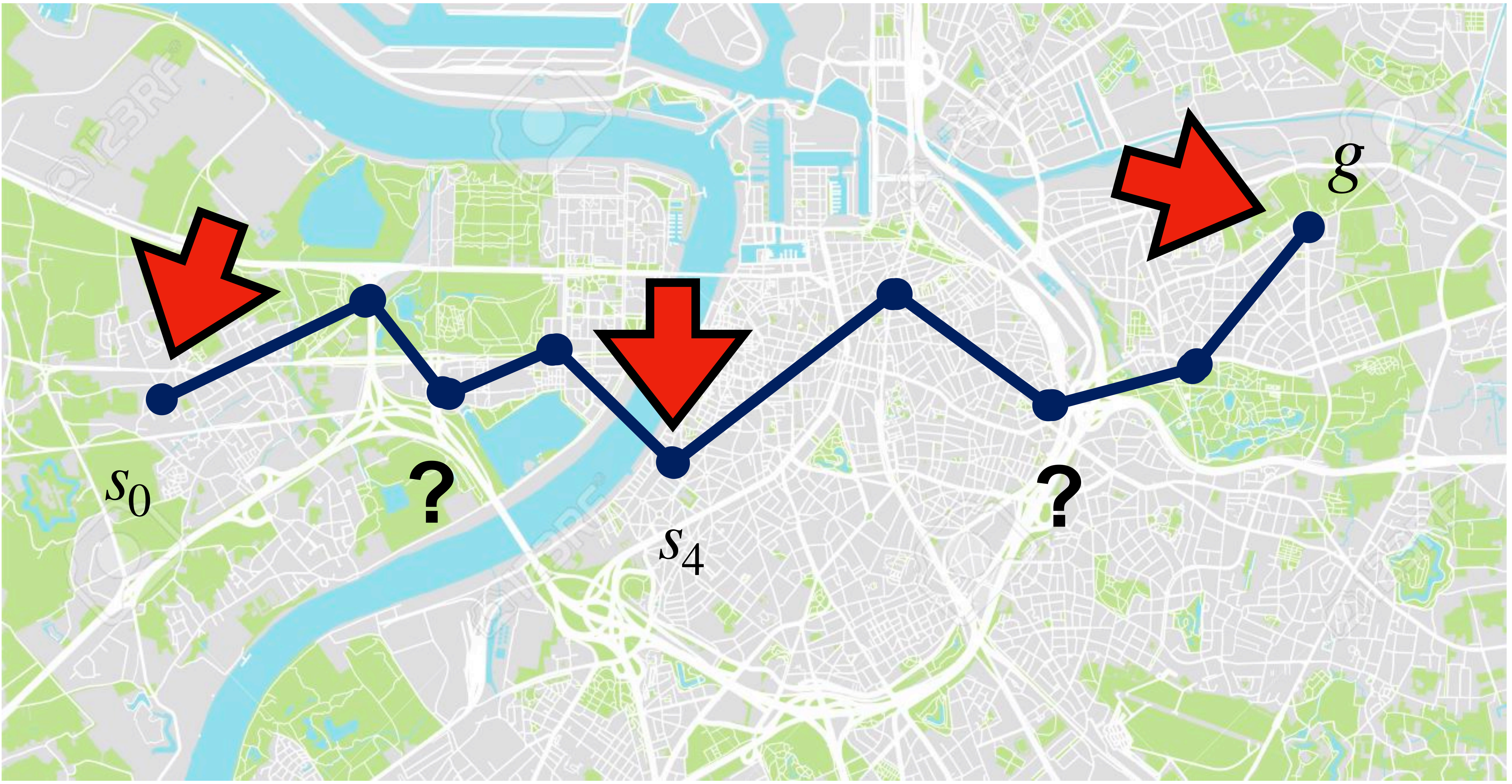
New APSP formulation for multi-goal RL w/o the Bellman-Eq.!

"SGT" RL:

What is the **middle** min cost state?

(middle state = **subgoal**)

$$s_0, g \downarrow s_4$$



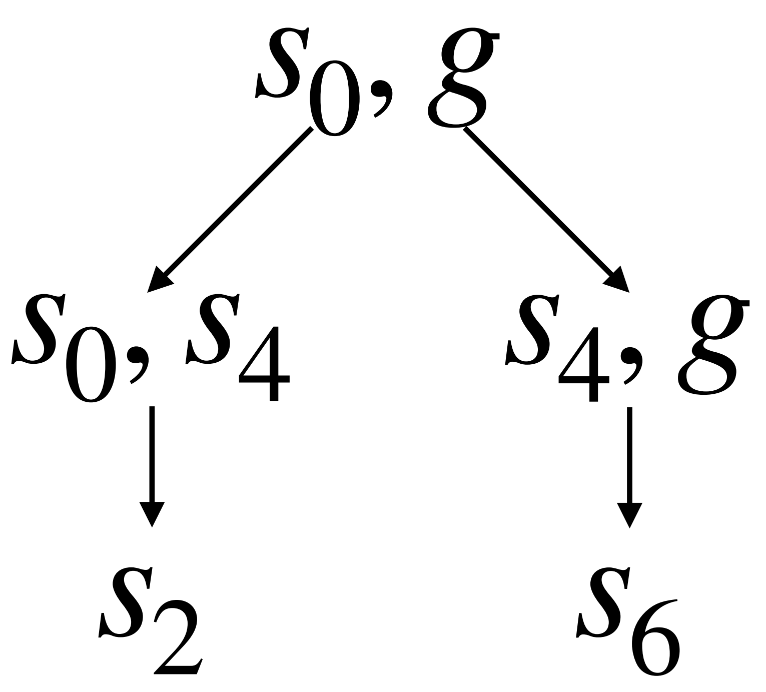
Our approach: Sub-goal Tree (SGT)

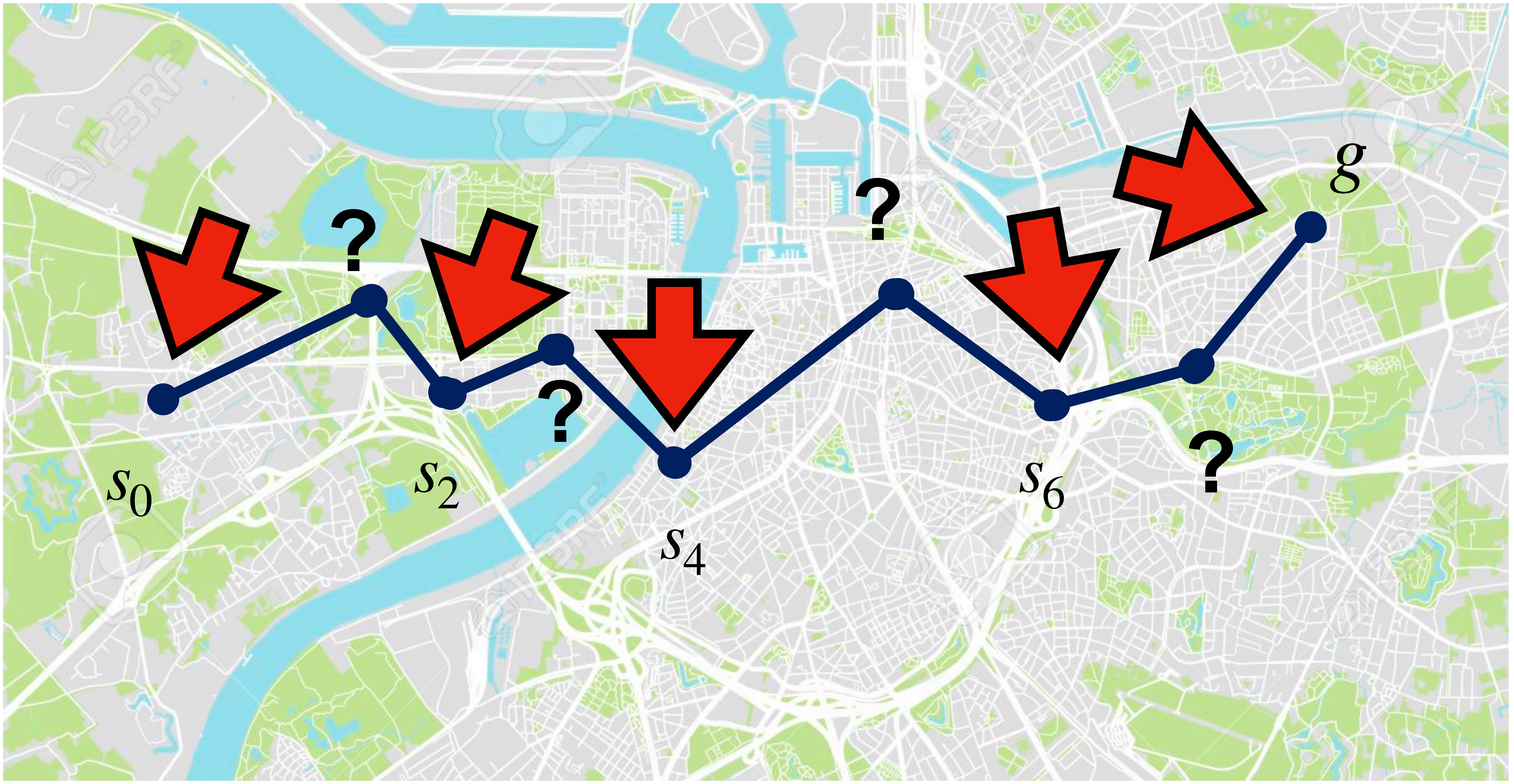
New APSP formulation for multi-goal RL w/o the Bellman-Eq.!

"SGT" RL:

What is the **middle** min cost state?

(middle state = **subgoal**)





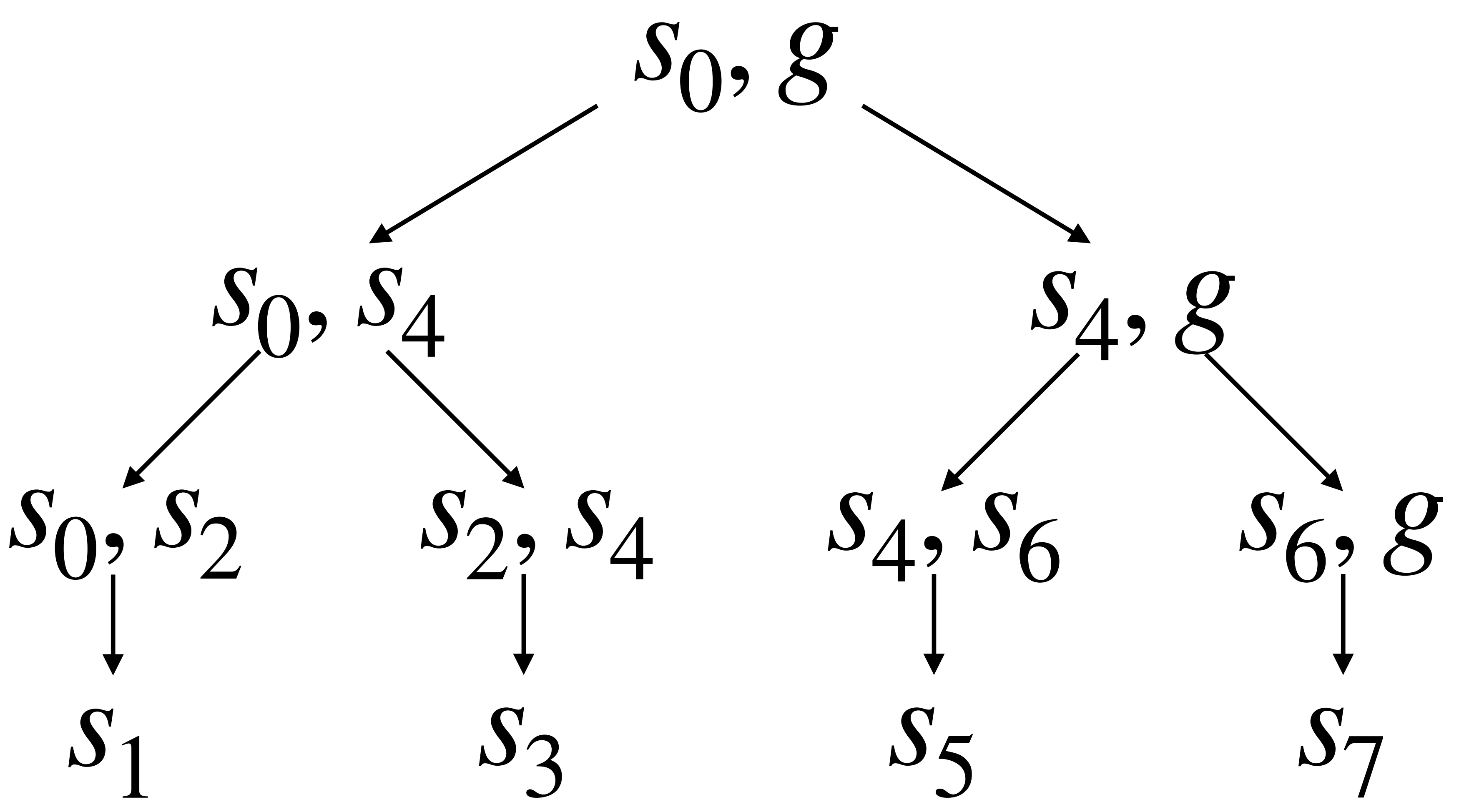
Our approach: Sub-goal Tree (SGT)

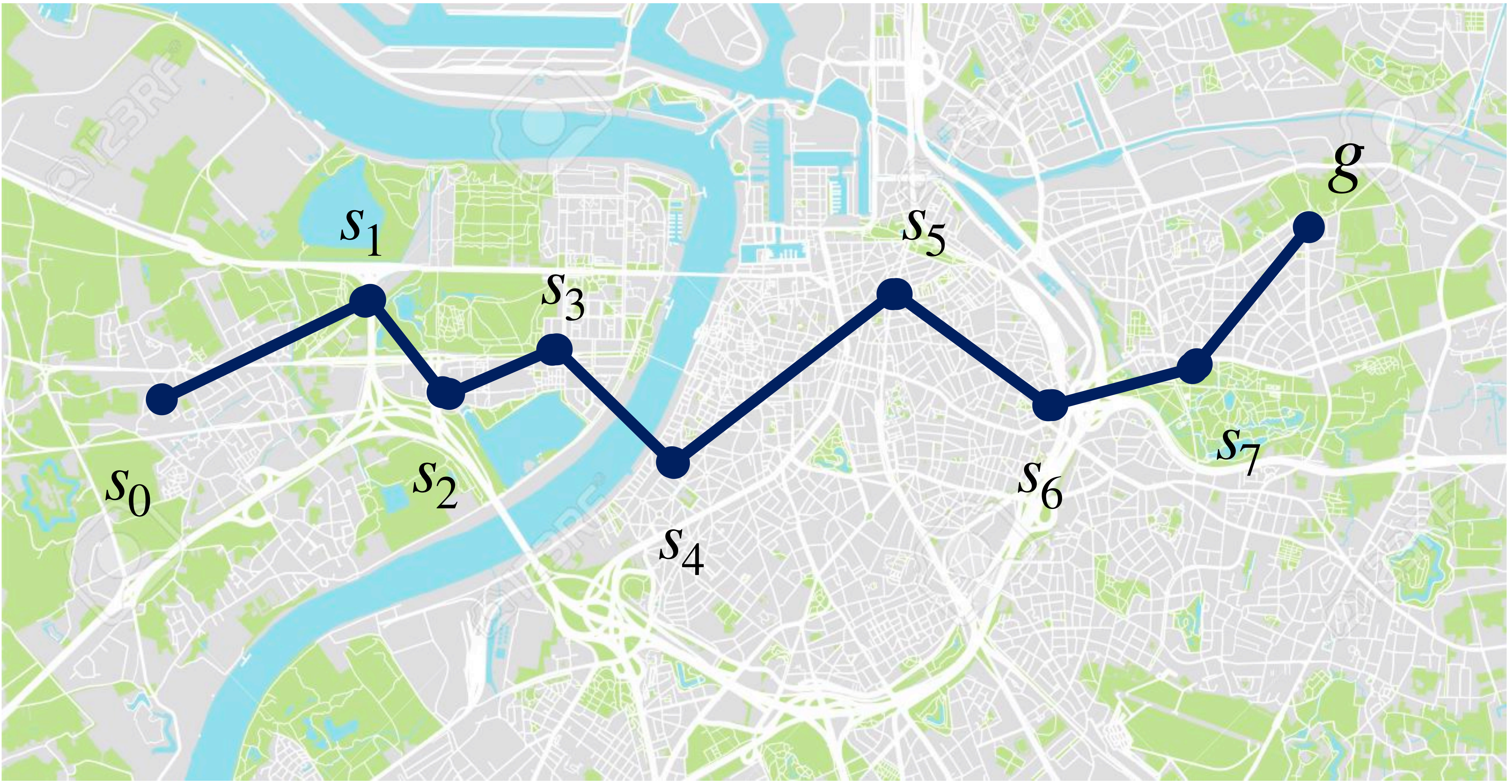
New APSP formulation for multi-goal RL w/o the Bellman-Eq.!

"SGT" RL:

What is the **middle** min cost state?

(middle state = **subgoal**)



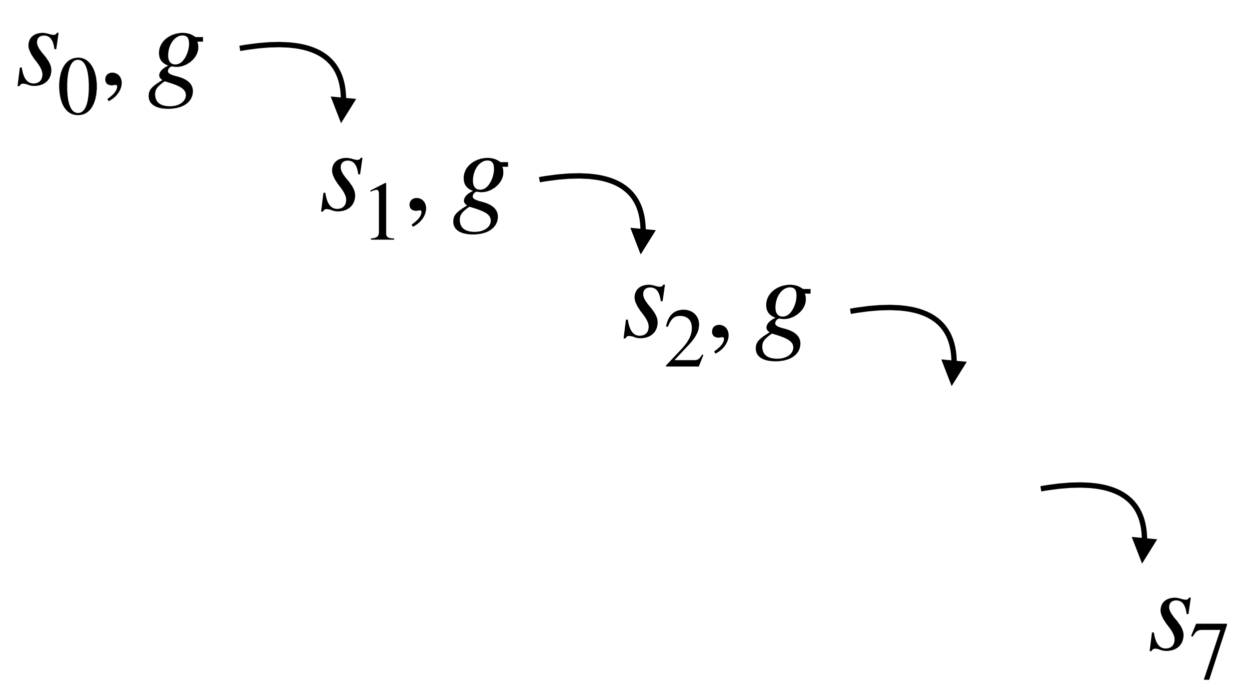


Our approach: Sub-goal Tree (SGT)

New APSP formulation for multi-goal RL w/o the Bellman-Eq.!

"Bellman" RL:

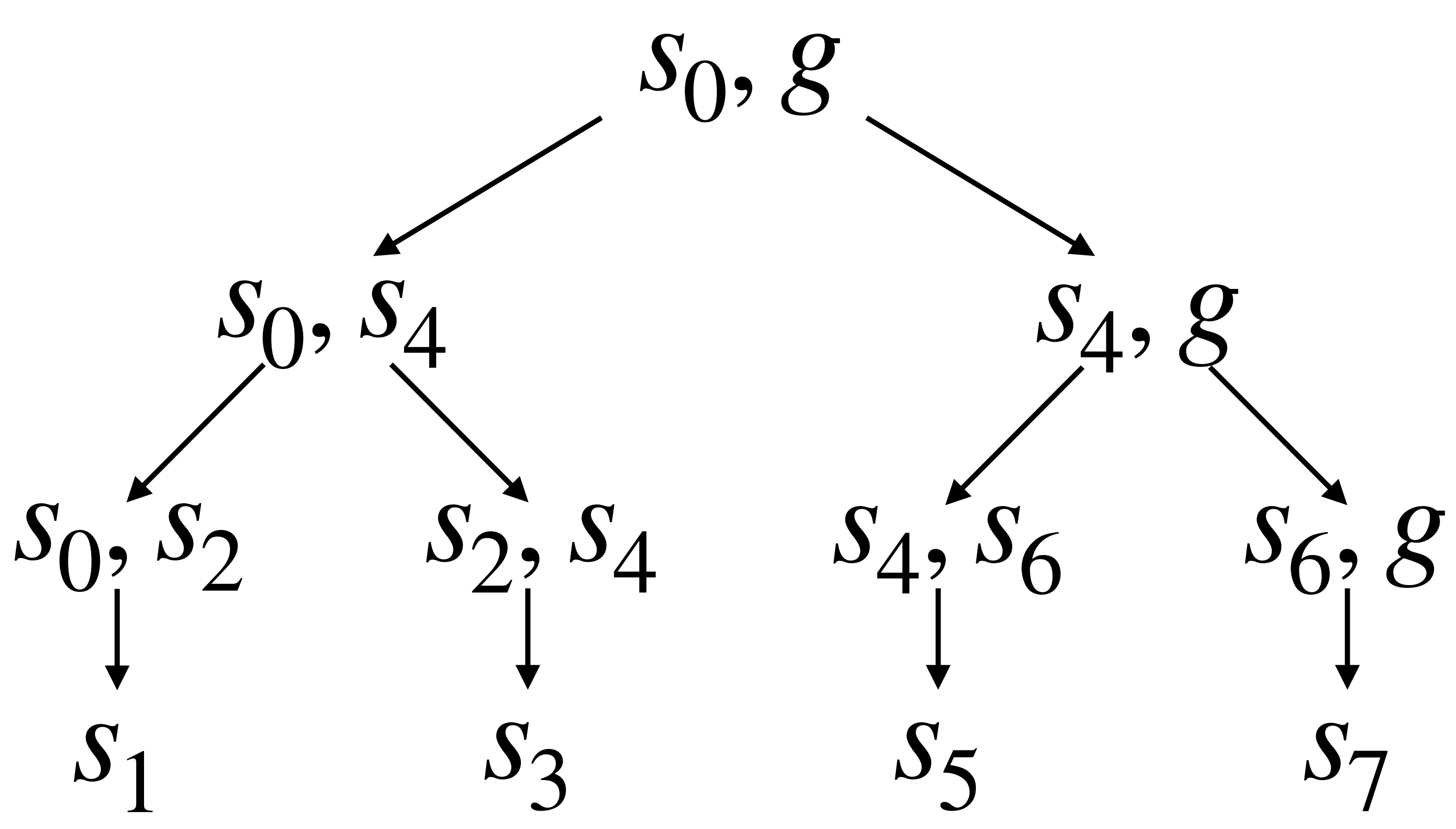
What is the **next** min cost state?



"SGT" RL:

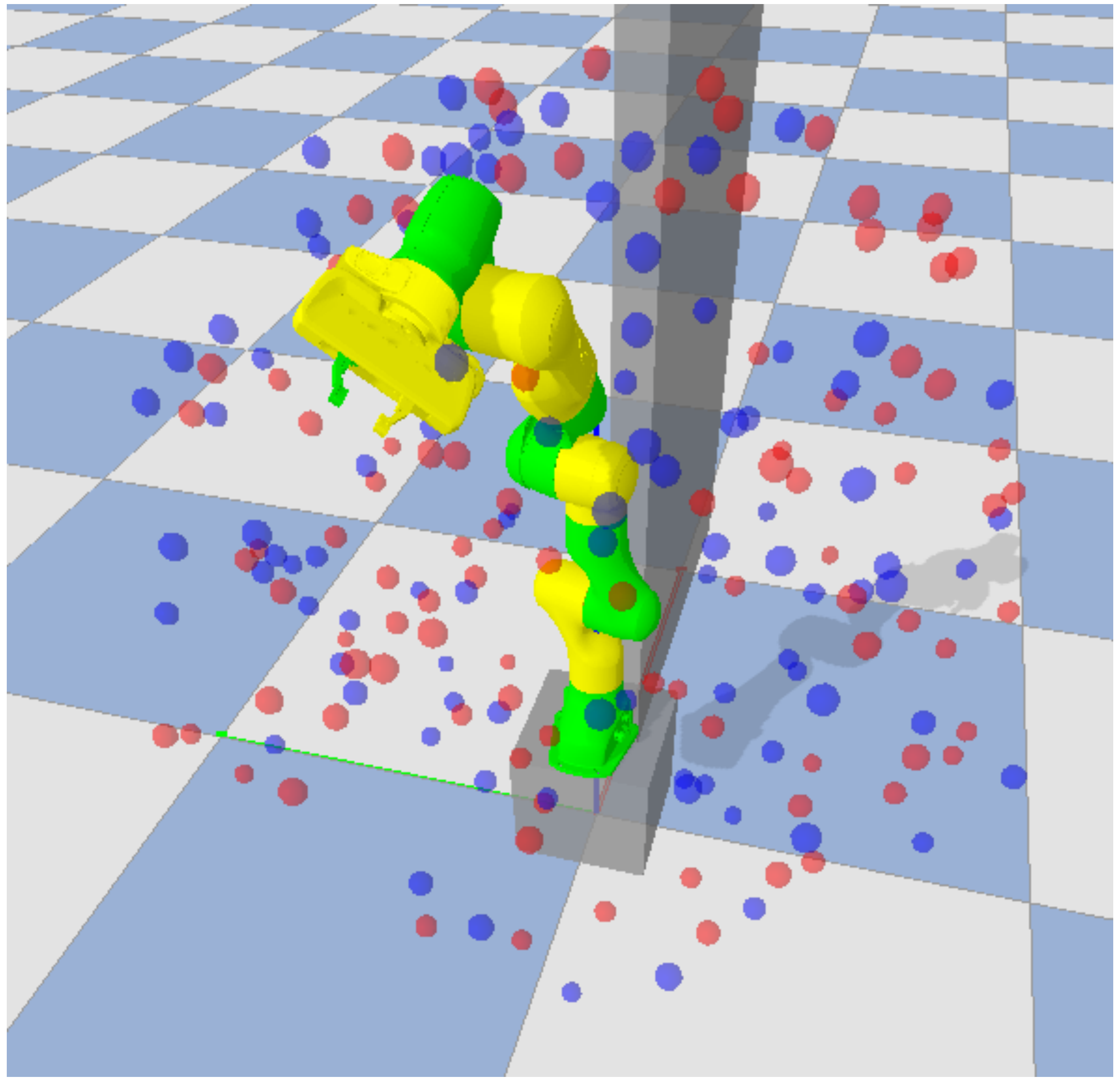
What is the **middle** min cost state?

(middle state = **subgoal**)



Sub-goal Trees - What’s next?

1. **DP principle for APSP RL**
2. SGT is provably more efficient
3. New RL algorithms based on SGT
    1. Value based - *Fitted SGT-DP*
    2. High dim. Problems with *SGT-PG*



Background: Bellman's Equation

<u>Bellman's principle of optimality</u>

- Value function

$$V_t^*(s) = \min_{\pi} \mathbb{E}^{\pi}\left(\sum_{t'=t}^{T} c\left(s_{t'}\right) \middle| s_t = s \right)$$

Background: Bellman's Equation

Bellman's principle of optimality

- Value function

$$V_t^*(s) = \min_\pi \mathbb{E}^\pi \left(\sum_{t'=t}^{T} c\left(s_{t'}\right) \middle| s_t = s \right)$$

- Bellman's equation

$$V_t^*(s) = \min_a \left\{ c(s) + \mathbb{E}\left(V_{t+1}^*(s') \middle| s, a \right) \right\}$$

Immediate cost

Proceed optimally

Background: Bellman's Equation

<u>Bellman's principle of optimality</u>

- Value function

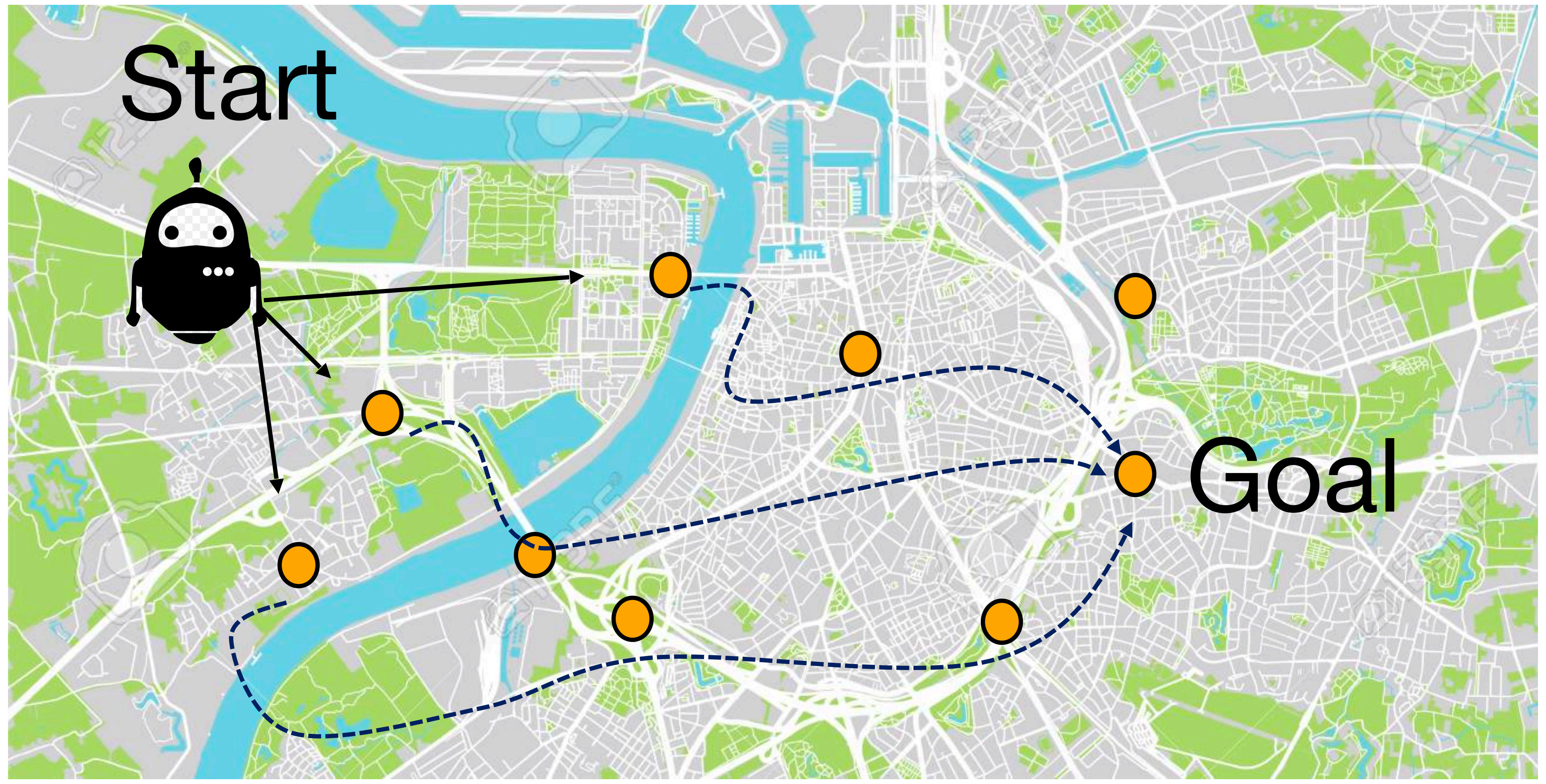
$$V_t^*(s) = \min_\pi \mathbb{E}^\pi\left(\sum_{t'=t}^{T} c\left(s_{t'}\right) \middle| s_t = s \right)$$

- Bellman's equation

$$V_t^*(s) = \min_a \left\{ c(s) + \mathbb{E}\left(V_{t+1}^*(s') \middle| s, a \right) \right\}$$

Immediate cost

Proceed optimally



Background: Bellman's Equation

<u>Bellman's principle of optimality</u>

- Value function

$$V_t^*(s) = \min_\pi \mathbb{E}^\pi \Bigg(\sum_{t'=}^{T} \cdots$$

- Bellman's equation

$$V_t^*(s) = \min_a \Big\{ c(s) + \mathbb{E}\big(V_{t+1}(s') \,\big|\, s, a \big) \Big\}$$

Immediate cost

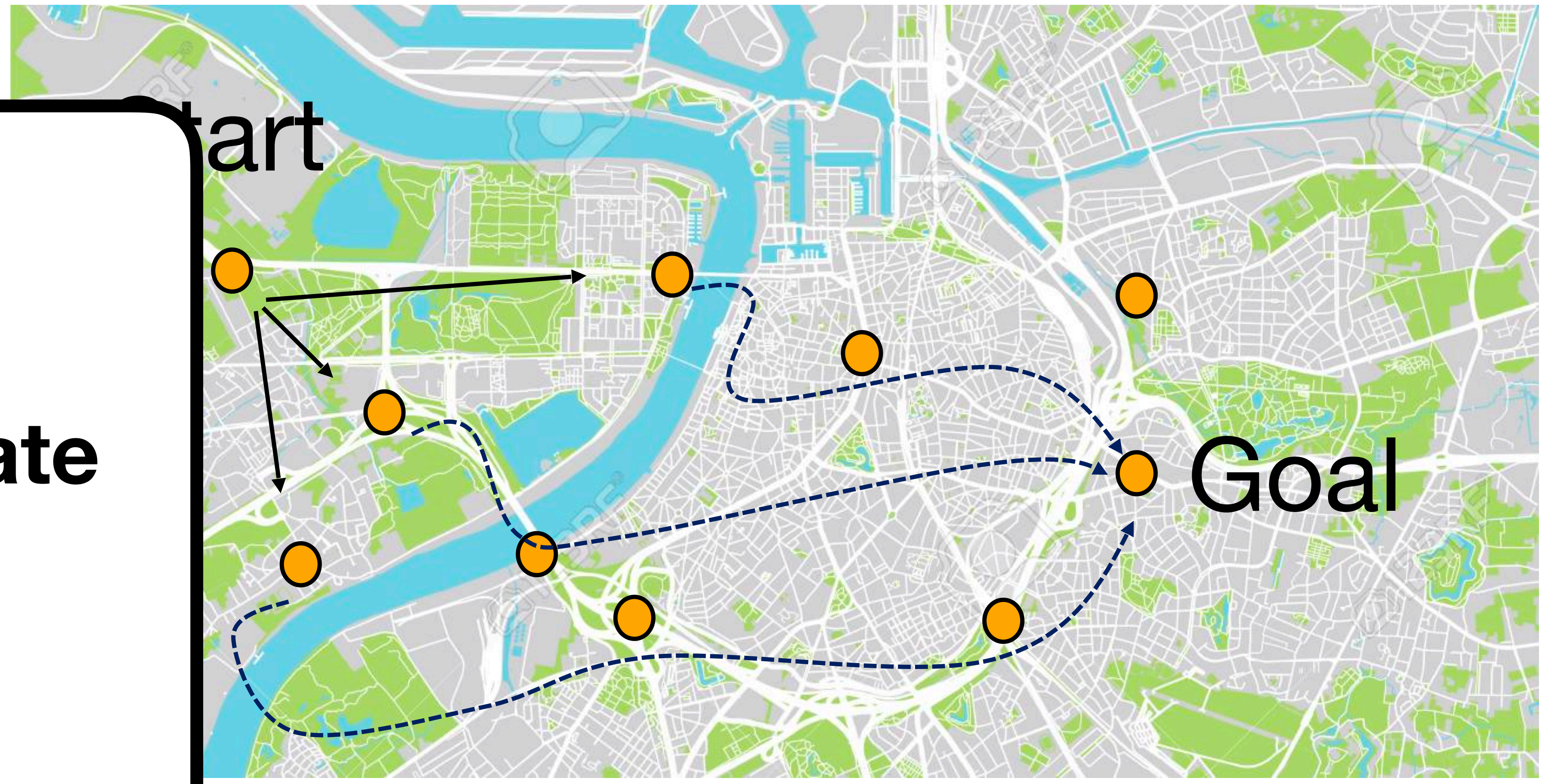
Proceed optimally

Bellman RL:

Decompose by **next state**

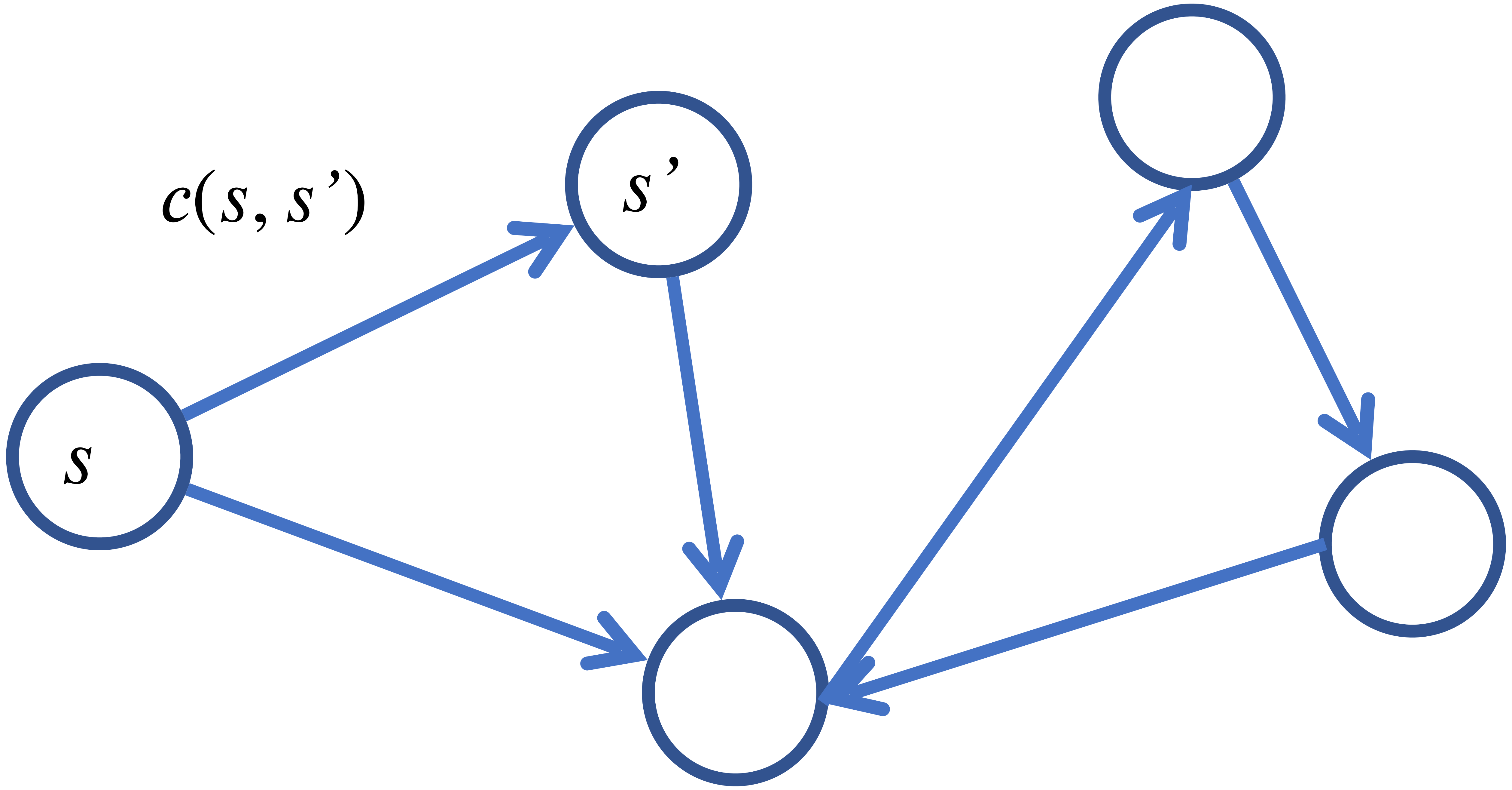
In SGT RL:

Decompose by **subgoal**



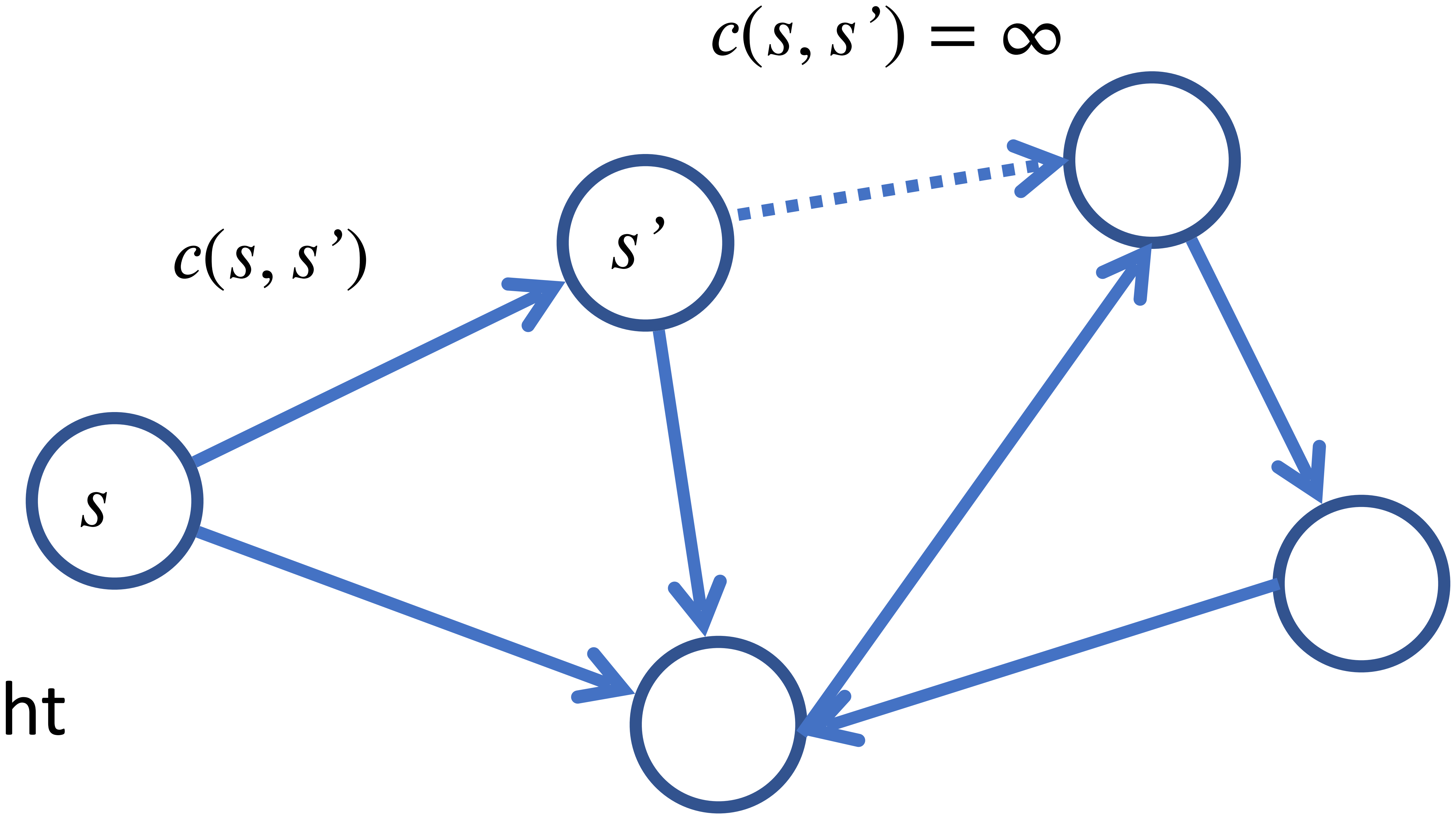
All-pairs shortest-path (APSP)

- Directed, weighted graph
- Describes a **deterministic MDP**
- N nodes, weights $c(s, s') \geq 0$
- $c(s, s) = 0$



All-pairs shortest-path (APSP)

- Directed, weighted graph
- Describes a **deterministic MDP**
- N nodes, weights $c(s, s') \geq 0$
- $c(s, s) = 0$
- Unconnected edge → infinite weight

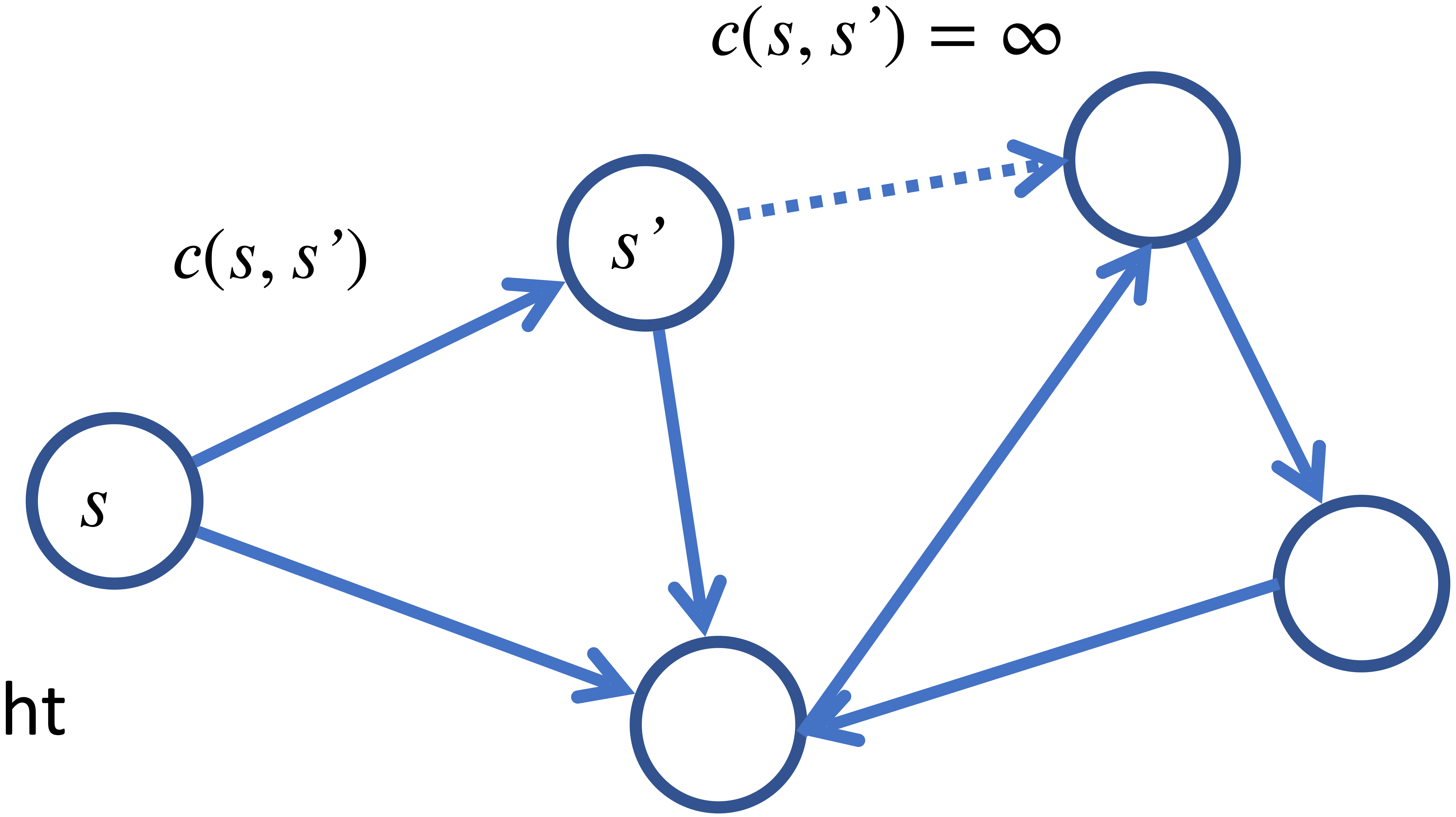


All-pairs shortest-path (APSP)

- Directed, weighted graph
- Describes a **deterministic MDP**
- N nodes, weights $c(s, s') \geq 0$
- $c(s, s) = 0$
- Unconnected edge → infinite weight
- **Objective:**

For any s, g:

$$\min_{T, s_0 = s, s_1, \ldots, s_{T-1}, s_T = g} \sum_{t=0}^{T-1} c(s_t, s_{t+1})$$



Sub-Goal Tree **Dynamic Programming**

- $V_k(s, s')$: length of shortest path $s \to s'$ in 2^k steps or less
- Obeys dynamic programming equations:

$$V_0(s, s') = c(s, s') \qquad \forall s, s'$$

$$V_k(s, s) = 0 \qquad \forall s$$

$$V_k(s, s') = \min_{s_m} \left\{ V_{k-1}\left(s, s_m\right) + V_{k-1}(s_m, s') \right\} \qquad \forall s, s' \colon s \neq s'$$

Sub-Goal Tree **Dynamic Programming**

- $V_k(s, s')$: length of shortest path $s \to s'$ in 2^k steps or less
- Obeys dynamic programming equations:

$$V_0(s, s') = c(s, s') \qquad \forall s, s'$$

$$V_k(s, s) = 0 \qquad \forall s$$

$$V_k(s, s') = \min_{s_m} \left\{ V_{k-1}\left(s, s_m\right) + V_{k-1}(s_m, s') \right\} \qquad \forall s, s' : s \neq s'$$

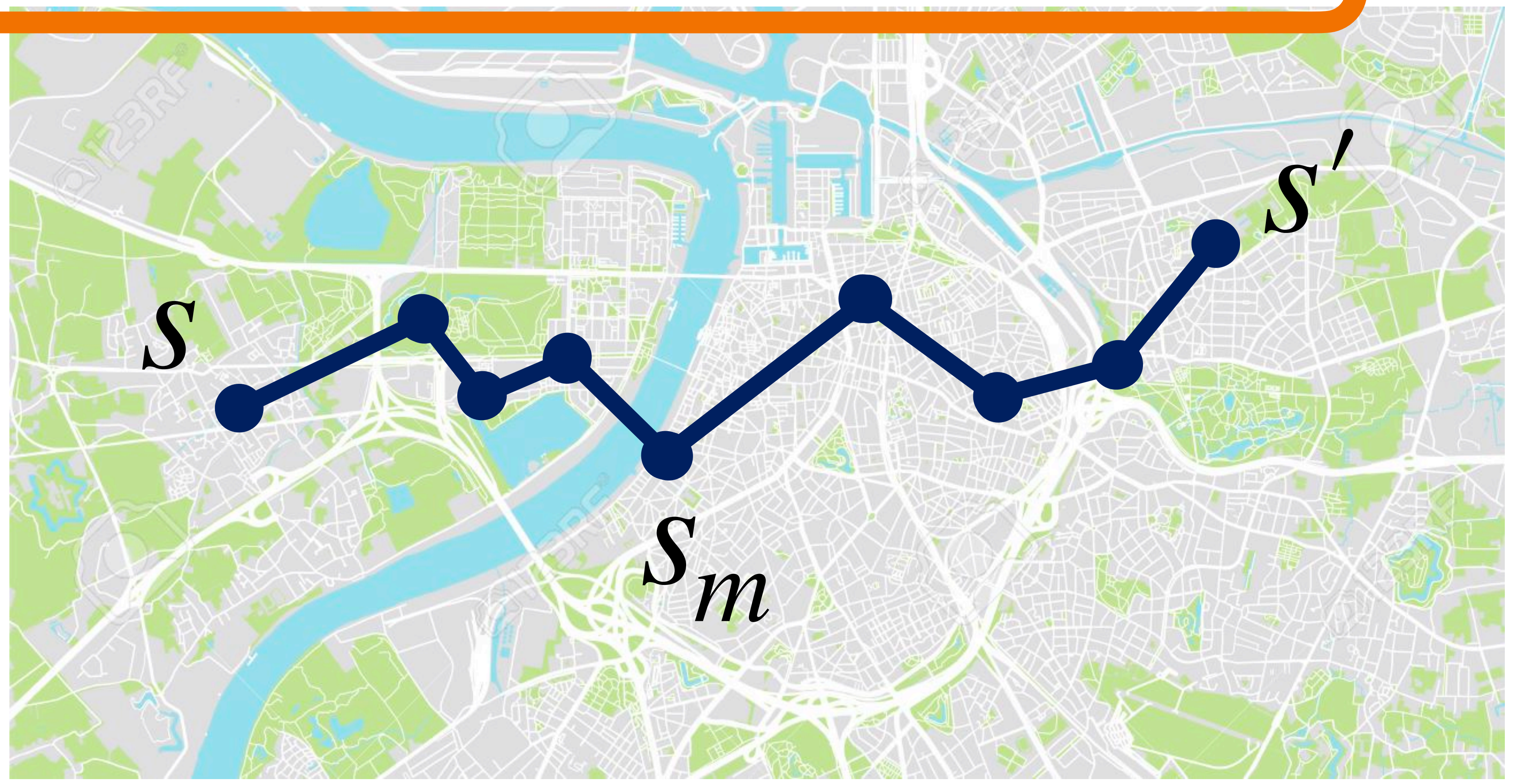
Sub-Goal Tree **Dynamic Programming**

- $V_k(s, s')$: length of shortest path $s \to s'$ in 2^k steps or less
- Obeys dynamic programming equations:

$$V_0(s, s') = c(s, s') \qquad \forall s, s'$$

$$V_k(s, s) = 0 \qquad \forall s$$

$$V_k(s, s') = \min_{s_m} \left\{ V_{k-1}(s, s_m) + V_{k-1}(s_m, s') \right\} \qquad \forall s, s' : s \neq s'$$



Sub-Goal Tree **Dynamic Programming**

- $V_k(s, s')$: length of shortest path $s \to s'$ in 2^k steps or less
- Obeys dynamic programming equations:

$$V_0(s, s') = c(s, s') \qquad \forall s, s'$$

$$V_k(s, s) = 0 \qquad \forall s$$

$$V_k(s, s') = \min_{s_m} \left\{ V_{k-1}(s, s_m) + V_{k-1}(s_m, s') \right\} \qquad \forall s, s' : s \neq s'$$

Sub-Goal Tree **Dynamic Programming**

- $V_k(s, s')$: length of shortest path $s \to s'$ in 2^k steps or less
- Obeys dynamic programming equations:

$$V_0(s, s') = c(s, s') \qquad \forall s, s'$$

$$V_k(s, s) = 0 \qquad \forall s$$

$$V_k(s, s') = \min_{s_m} \left\{ V_{k-1}\left(s, s_m\right) + V_{k-1}(s_m, s') \right\} \qquad \forall s, s': s \neq s'$$

- Dependence on k is important —> similar to finite horizon DP!

Sub-Goal Tree **Dynamic Programming**

- $V_k(s, s')$: length of shortest path $s \to s'$ in 2^k steps or less
- Obeys dynamic programming equations:

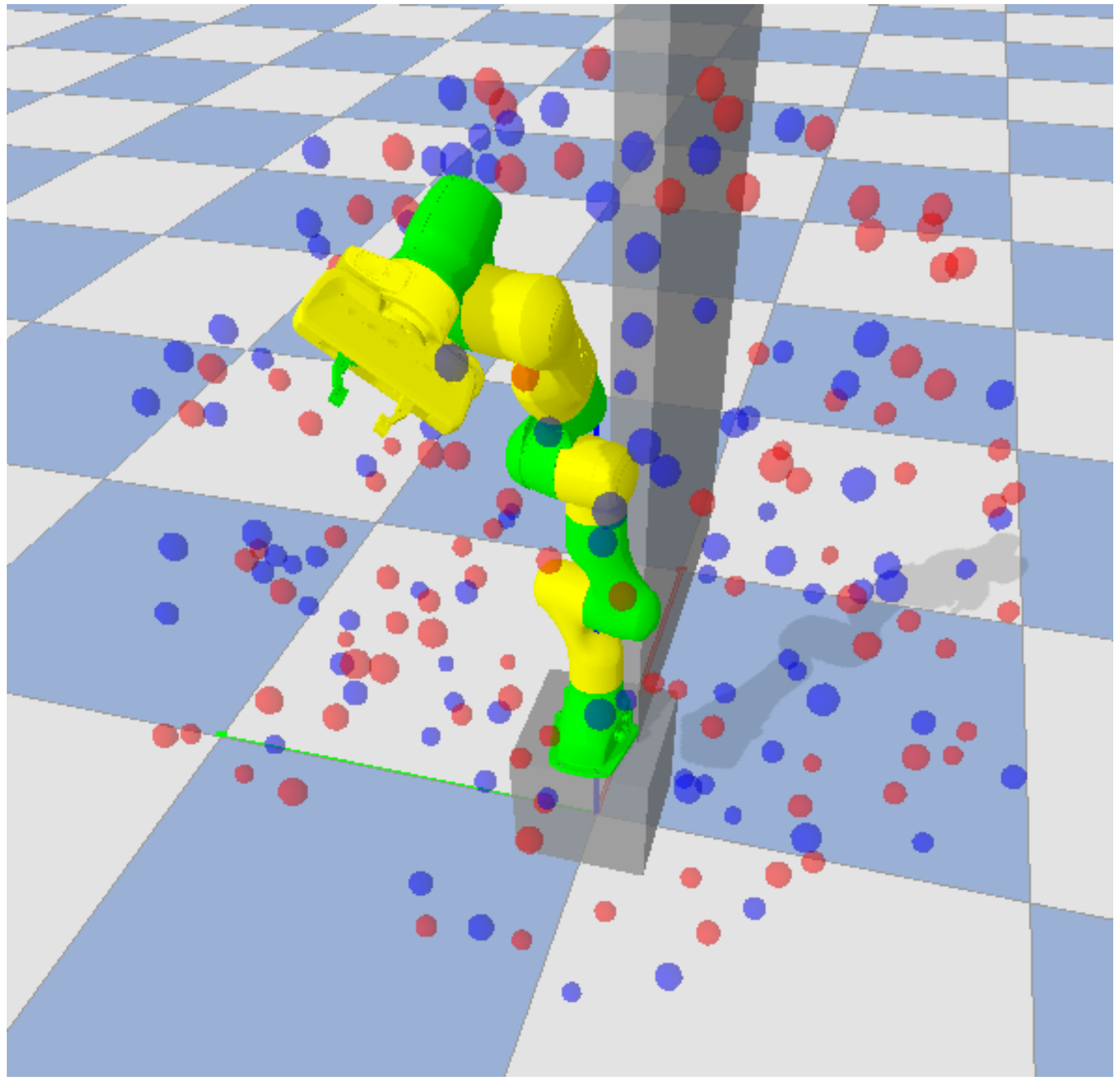
$$
\begin{aligned}
&V_0(s, s') = c(s, s') && \forall s, s' \\
&V_k(s, s) = 0 && \forall s \\
&V_k(s, s') = \min_{s_m} \left\{ V_{k-1}(s, s_m) + V_{k-1}(s_m, s') \right\} && \forall s, s' : s \neq s'
\end{aligned}
$$

Theorem: For $k \geq \log_2 N$, we have:
$V_k(s, s')$ is the length of shortest path from s to s', for all s, s'.

Worst case running time $O\left(N^3 \log N\right)$

Sub-goal Trees - What's next?

1. DP principle for APSP RL
2. **SGT is provably more efficient**
3. New RL algorithms based on SGT
   1. Value based - *Fitted SGT-DP*
   2. High dim. Problems with *SGT-PG*



SGT **Approximate** Dynamic Programming

- Key in RL – function approximation (large state space)
- How does SGT handle approximations?

SGT **Approximate** Dynamic Programming

- Key in RL – function approximation (large state space)
- How does SGT handle approximations?
- Define the SGT operator T:

$$(TV)(\mathrm{s},\mathrm{s}') = \min_{s_m}\left\{ V\left(s, s_m\right) + V\left(s_m, s'\right) \right\}$$

- Approximate SGT iterations:

$$\left\| \hat{V}_{k+1} - T\hat{V}_k \right\|_\infty \leq \epsilon$$

SGT **Approximate** Dynamic Programming

- Key in RL – function approximation (large state space)
- How does SGT handle approximations?
- Define the SGT operator T:

$$(TV)(\mathrm{s}, \mathrm{s}') = \min_{s_m} \left\{ V\left(s, s_m\right) + V(s_m, s') \right\}$$

- Approximate SGT iterations:

$$\left\| \hat{V}_{k+1} - T\hat{V}_k \right\|_\infty \leq \epsilon$$

- Error propagation?

SGT **Approximate** Dynamic Programming

- Error propagation:
    - Value error $O(N\epsilon)$ for Bellman and SGT

SGT **Approximate** Dynamic Programming

- Error propagation:
  - Value error $O(N\epsilon)$ for Bellman and SGT
- But what about the resulting trajectory?
  - Greedy w.r.t. value function

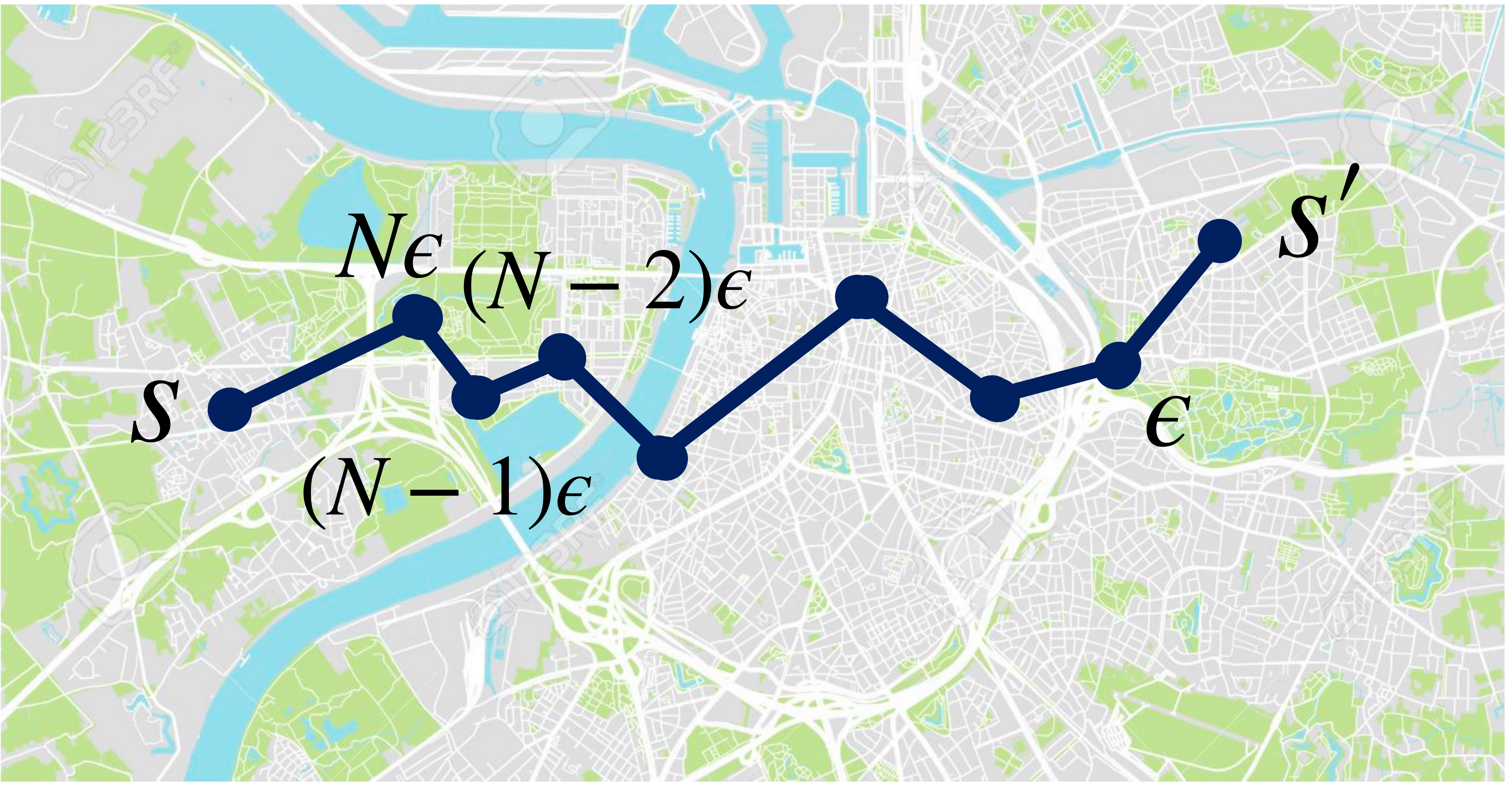
SGT **Approximate** Dynamic Programming

- Error propagation:
    - Value error $O(N\epsilon)$ for Bellman and SGT
- But what about the resulting trajectory?
    - Greedy w.r.t. value function
    - Bellman RL error accumulation: $O(N^2\epsilon)$

SGT **Approximate** Dynamic Programming

- Error propagation:
  - Value error $O(N\epsilon)$ for Bellman and SGT
- But what about the resulting trajectory?
  - Greedy w.r.t. value function
  - Bellman RL error accumulation: $O(N^2\epsilon)$

Grows linearly with distance from goal!



SGT **Approximate** Dynamic Programming

- Error propagation:
  - Value error $O(N\epsilon)$ for Bellman and SGT
- But what about the resulting trajectory?
  - Greedy w.r.t. value function
  - Bellman RL error accumulation: $O(N^2\epsilon)$
  - SGT error accumulation: $O(N \log_2 N\epsilon)$!

SGT **Approximate** Dynamic Programming

- Error propagation:
  - Value error $O(N\epsilon)$ for Bellman and SGT
- But what about the resulting trajectory?
  - Greedy w.r.t. value function
  - Bellman RL error accumulation: $O(N^2\epsilon)$
  - SGT error accumulation: $O(N \log_2 N\epsilon)$!

Error decreases exponentially!

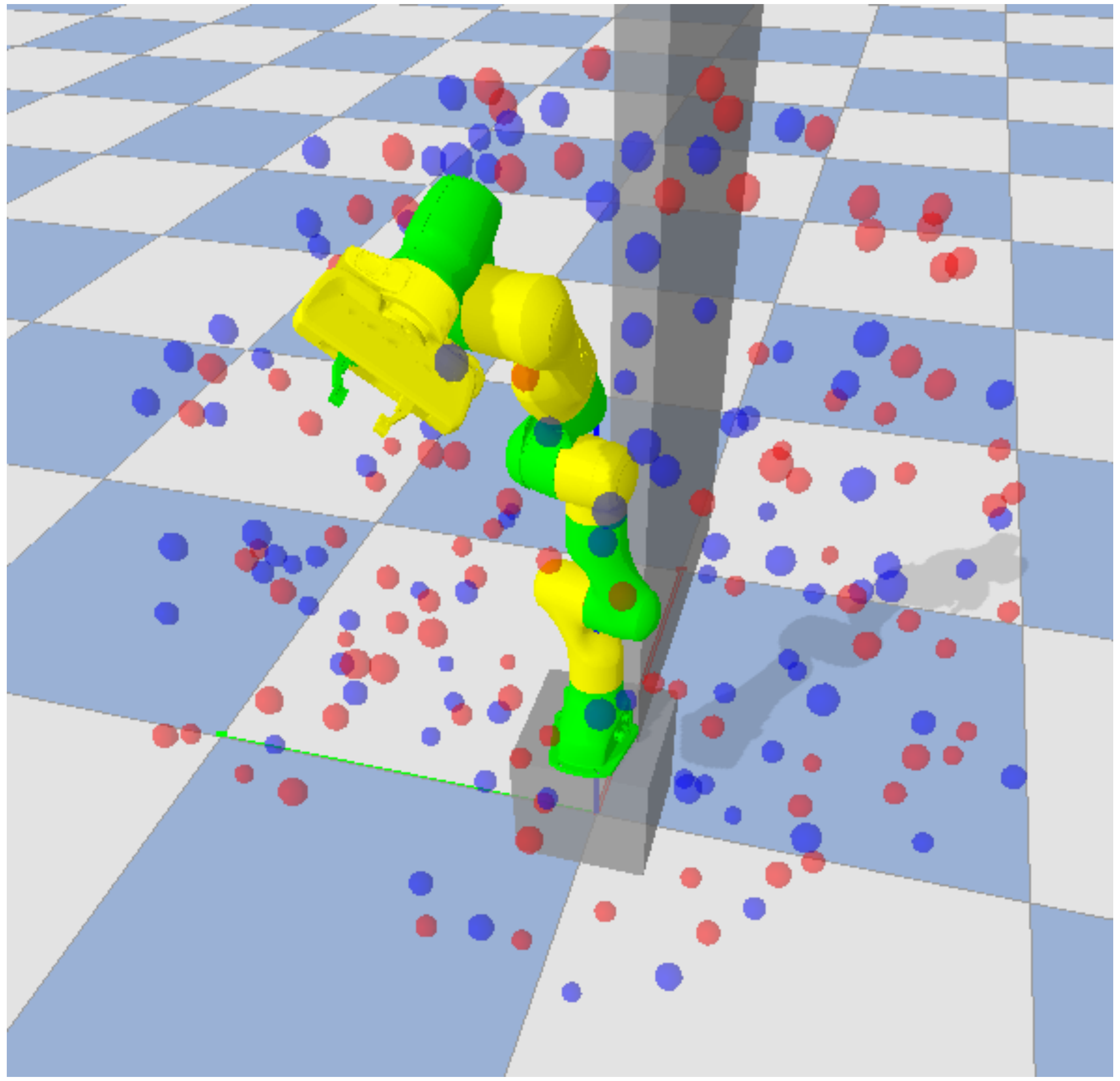
SGT **Approximate** Dynamic Programming

- Error propagation:
  - Value error $O(N\epsilon)$ for Bellman and SGT
- But what about the resulti
  - Greedy w.r.t. value fur
  - Bellman RL error accu
  - SGT error accumulation: $O(N \log_2 N\epsilon)$!

Less drift with SGT trajectories!

Sub-goal Trees - What's next?

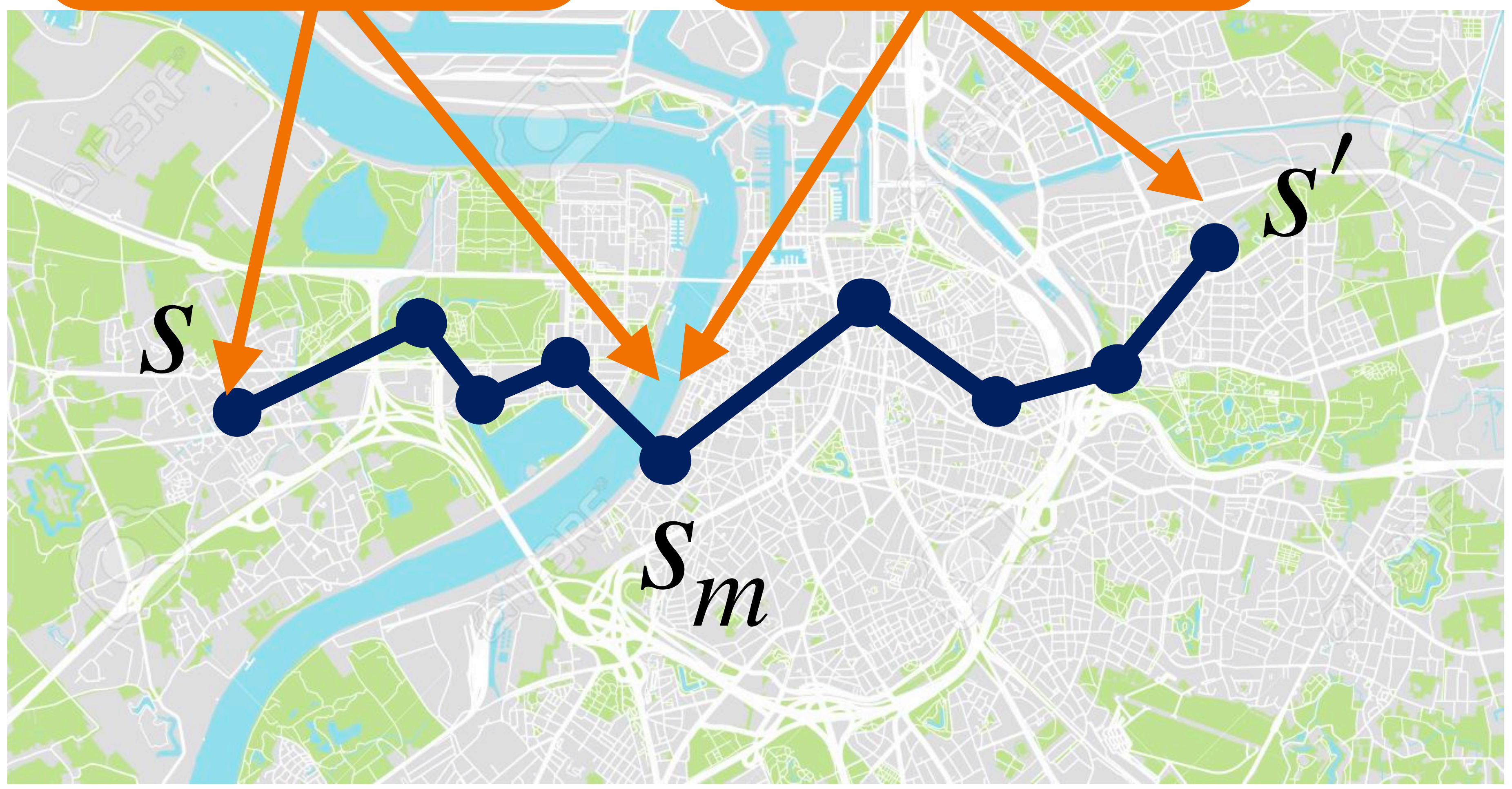
1. DP principle for APSP RL
2. SGT is provably more efficient
3. **New RL algorithms based on SGT**
    1. Value based - *Fitted SGT-DP*
    2. High dim. Problems with *SGT-PG*



Recap

- So far - SGT = a new approximate DP framework

$$V_k(s, s') = \min_{s_m} \left\{ V_{k-1}(s, s_m) + V_{k-1}(s_m, s') \right\}$$



Recap

- So far - SGT = a new approximate DP framework
- Develop new RL algorithms!

Fitted SGT-DP - value-based algorithm

Off-policy batch RL data: (s, s', c) tuples

At iteration k:

1. Sample states, goals from data $\{s, g\}$
2. Generate regression targets: $V_{target} = \min_{s_m} \left\{ \hat{V}_{k-1}\left(s, s_m\right) + \hat{V}_{k-1}(s_m, g) \right\}$
3. Fit new value function: $\hat{V}_k = Fit(V_{target})$

Fitted SGT-DP - value-based algorithm

Off-policy batch RL data: (s, s', c) tuples

At iteration k:

1. Sample states, goals from data $\{s, g\}$
2. Generate regression targets: $V_{target} = \min_{s_m} \left\{ \hat{V}_{k-1}\left(s, s_m\right) + \hat{V}_{k-1}(s_m, g) \right\}$
3. Fit new value function: $\hat{V}_k = Fit(V_{target})$

Fitted SGT-DP - value-based algorithm

Off-policy batch RL data: (s, s', c) tuples

At iteration k:

1. Sample states, goals from data $\left\{s, g\right\}$
2. Generate regression targets: $V_{target} = \min_{s_m} \left\{ \hat{V}_{k-1}\left(s, s_m\right) + \hat{V}_{k-1}(s_m, g) \right\}$
3. Fit new value function: $\hat{V}_k = Fit(V_{target})$

Grid search over s_m

Fitted SGT-DP - value-based algorithm

Off-policy batch RL da

At iteration k:

1. Sample states, g

2. Generate regres ... $_m) + \hat{V}_{k-1}(s_m, g)\}$

3. Fit new value fun

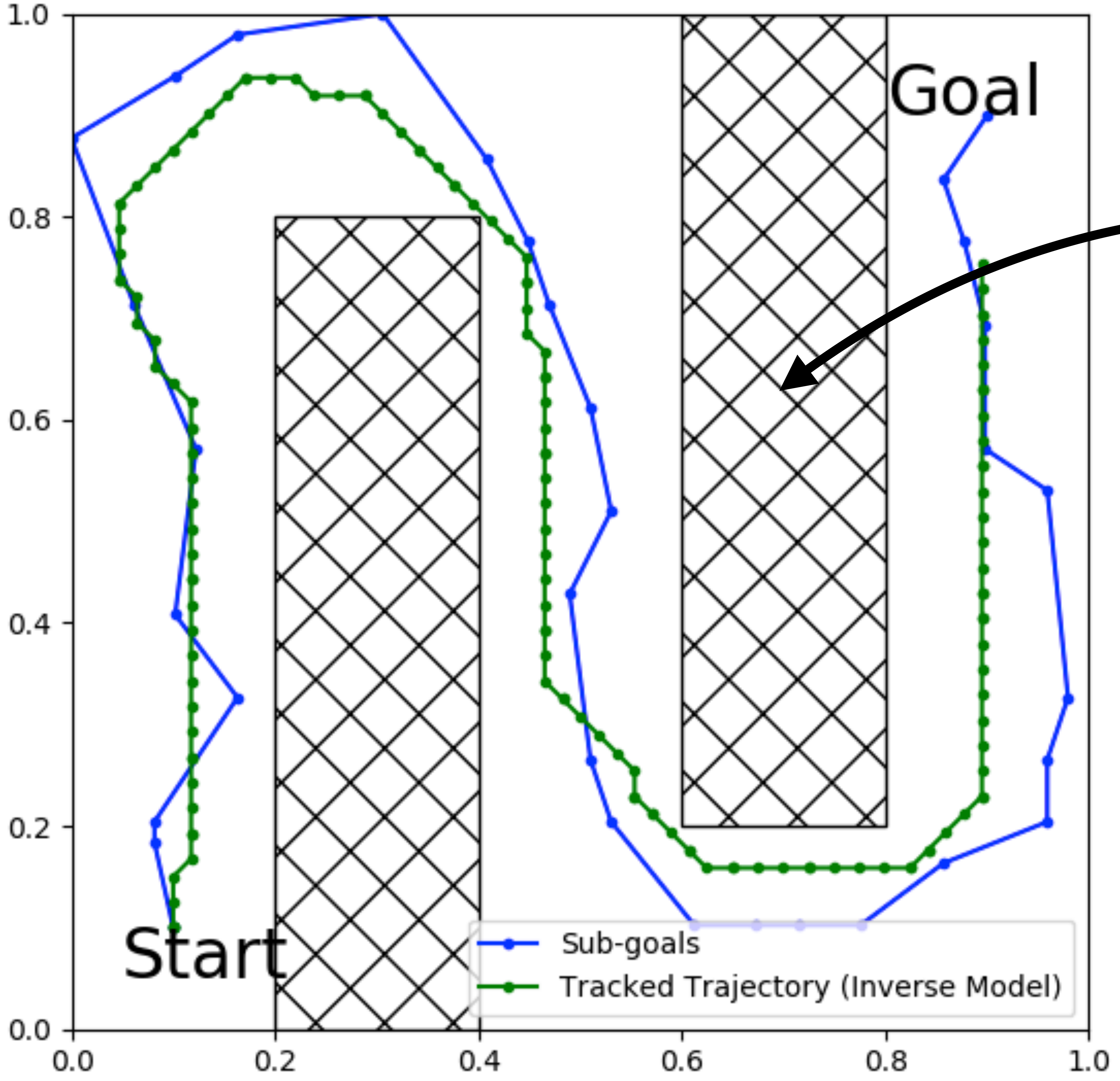
What about actions?

- Learn **inverse model** $(s, s') \rightarrow a$

  Easy if data = (s, a, s', c)

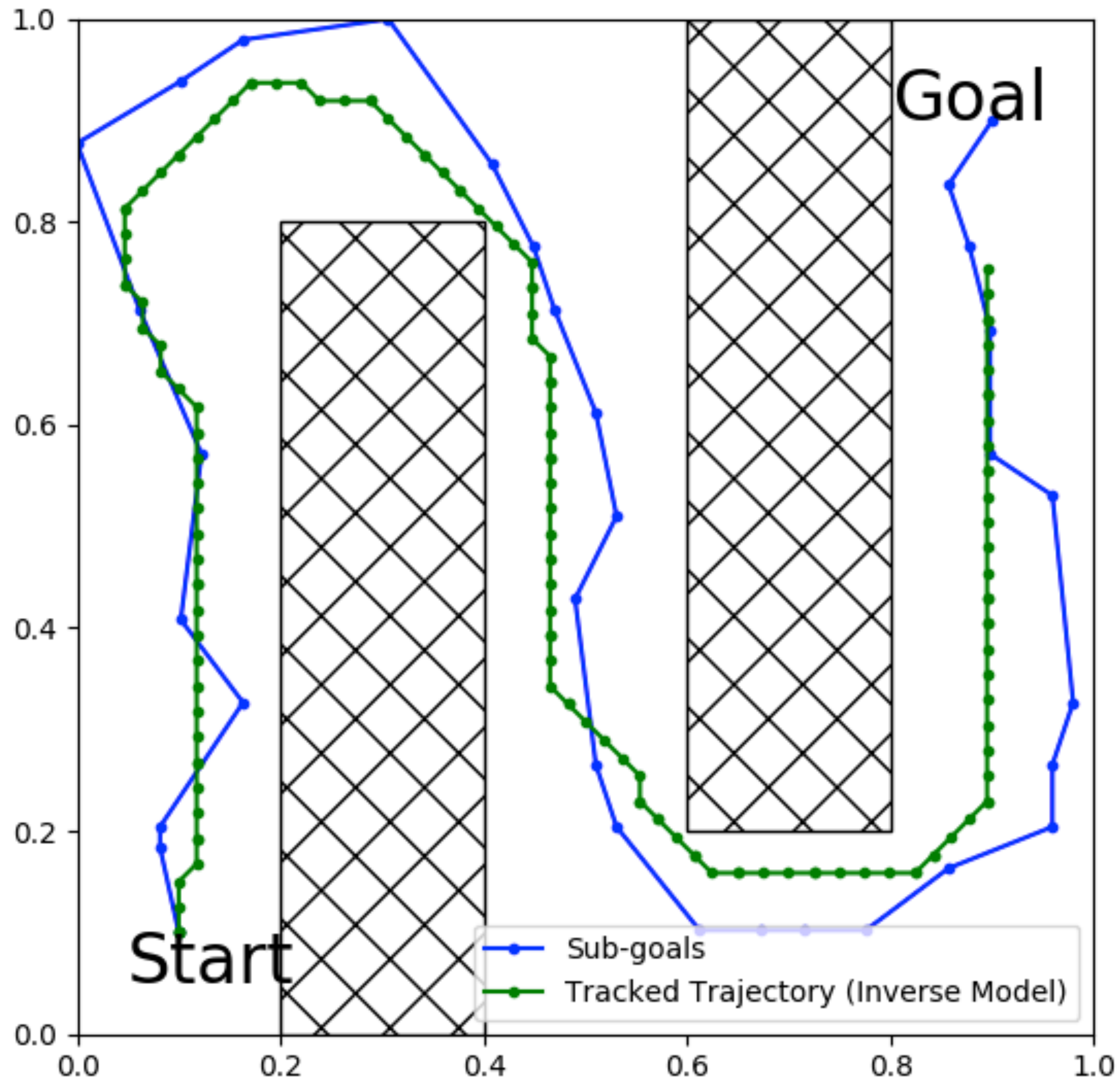
- Use standard goal-based RL for low level actions

Fitted SGT-DP: 2D Point Robot Results



High cost inside obstacles!

Fitted SGT-DP: 2D Point Robot Results



Model	Distance to goal
Fitted Q iteration	0.58
Fitted SGT-DP (ours)	**0.13**

Fitted SGT-DP - high dim states?

Off-policy batch RL data: (s, s', c) tuples

At iteration k:

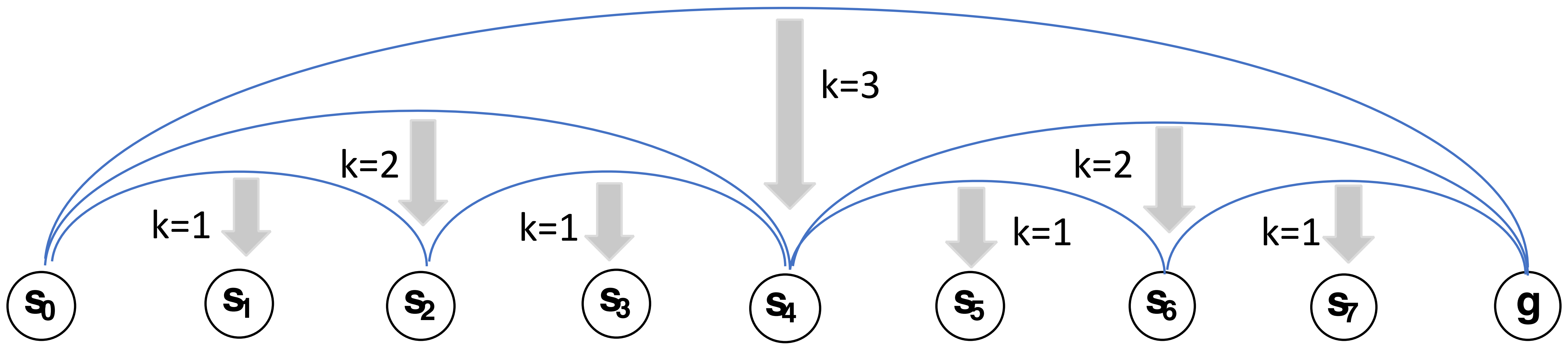
1. Sample states, goals from da

How to handle high-dim states?

2. Generate regression targets: $V_{target} = \min_{s_m} \left\{ \hat{V}_{k-1}\left(s, s_m\right) + \hat{V}_{k-1}(s_m, g) \right\}$
3. Fit new value function: $\hat{V}_k = Fit(V_{target})$

SGT **Policy Gradient** (SGT-PG)

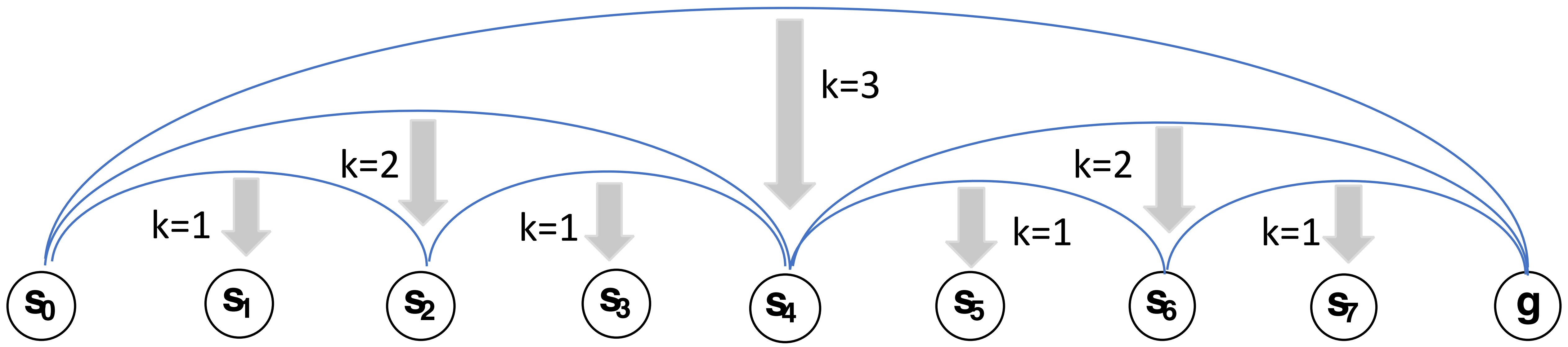
- Stochastic policy for next sub-goal $\pi_\theta(s_m \mid s, s')$ with parameters θ



$$\Pr_\pi[\tau|s,g] = \Pr_\pi[s_0,\ldots,s_T|s,g]$$

SGT **Policy Gradient** (SGT-PG)

- Stochastic policy for next sub-goal $\pi_\theta(s_m \,|\, s, s')$ with parameters θ



$$\Pr_\pi[\tau|s,g] = \Pr_\pi[s_0,\ldots,s_T|s,g]$$

- Find θ that minimizes the trajectory cost:

$$J(\theta) = J^{\pi_\theta} = \mathbb{E}_{\tau\sim\rho(\pi_\theta)}\left[c_\tau\right]$$

SGT **Policy Gradient** (SGT-PG)

- Policy gradient theorem for SGT:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\rho(\pi_\theta)}\left[c_\tau \cdot \nabla_\theta \log \Pr_{\rho(\pi_\theta)}[\tau]\right]$$
$$= \mathbb{E}_{\rho(\pi_\theta)}\left[\sum_{d=1}^{D}\sum_{i=1}^{2^{D-d}} C_\tau^{i,d} \cdot \nabla_\theta \log \pi_\theta\left(s_m^{i,d} \middle| s^{i,d}, g^{i,d}\right)\right]$$

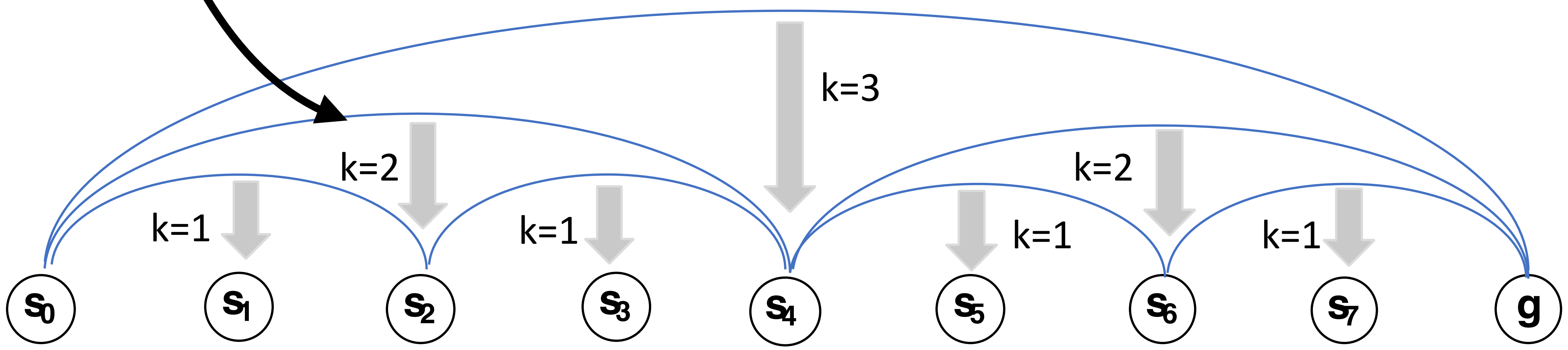
SGT **Policy Gradient** (SGT-PG)

- Policy gradient theorem for SGT:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\rho(\pi_\theta)}\left[c_\tau \cdot \nabla_\theta \log \Pr_{\rho(\pi_\theta)}[\tau]\right]$$

$$= \mathbb{E}_{\rho(\pi_\theta)}\left[\sum_{d=1}^{D}\sum_{i=1}^{2^{D-d}} C_\tau^{i,d} \cdot \nabla_\theta \log \pi_\theta \left(s_m^{i,d} \middle| s^{i,d}, g^{i,d}\right)\right]$$

Sum of costs in segment



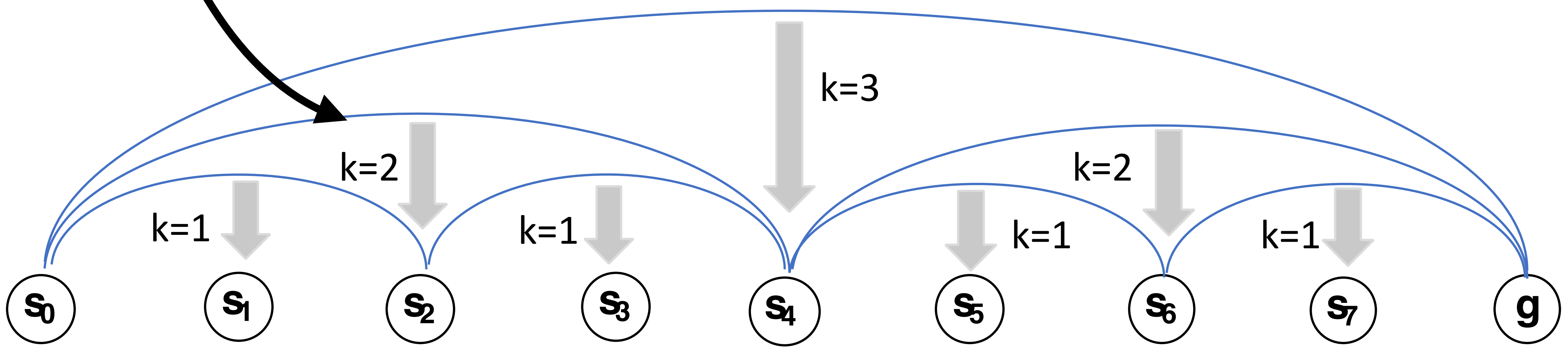
SGT **Policy Gradient** (SGT-PG)

- Policy gradient theorem for SGT:

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\rho(\pi_\theta)}\left[c_\tau \cdot \nabla_\theta \log \Pr_{\rho(\pi_\theta)}[\tau]\right] \\
&= \mathbb{E}_{\rho(\pi_\theta)}\left[\sum_{d=1}^{D} \sum_{i=1}^{2^{D-d}} C_\tau^{i,d} \cdot \nabla_\theta \log \pi_\theta \left(s_m^{i,d} \middle| s^{i,d}, g^{i,d}\right)\right]
\end{aligned}
$$

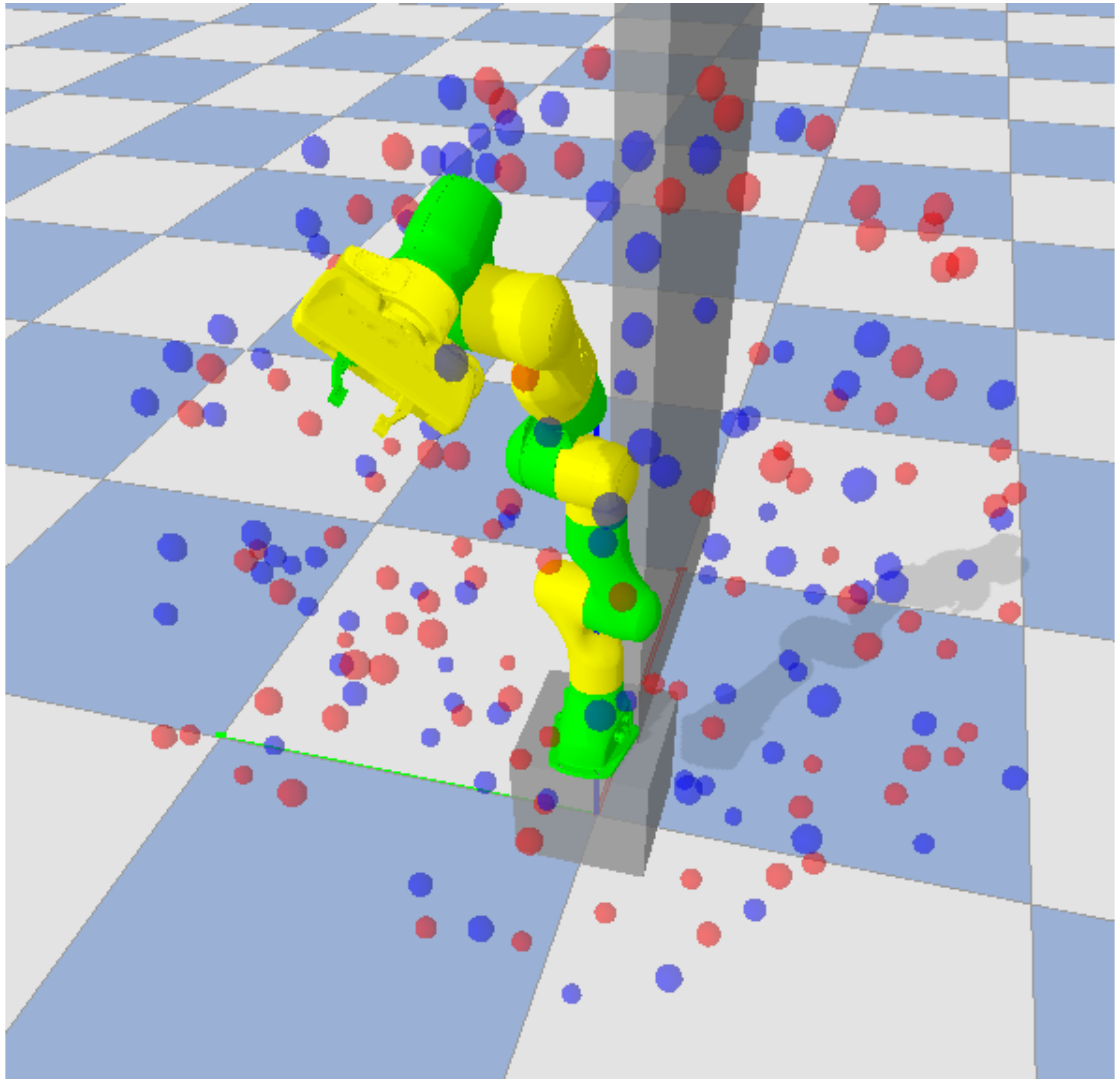
Sum of costs in segment

Can also add standard tricks: baseline, trust region, etc.



SGT-PG - Experiments

- Continuous motion planning
- 7 DoF robot arm
- Obstacles, self-collisions
- Reach from any state to any goal
- NN predicts sub-goals
- Linear motion between sub-goals



SGT-PG - Trajectory Example

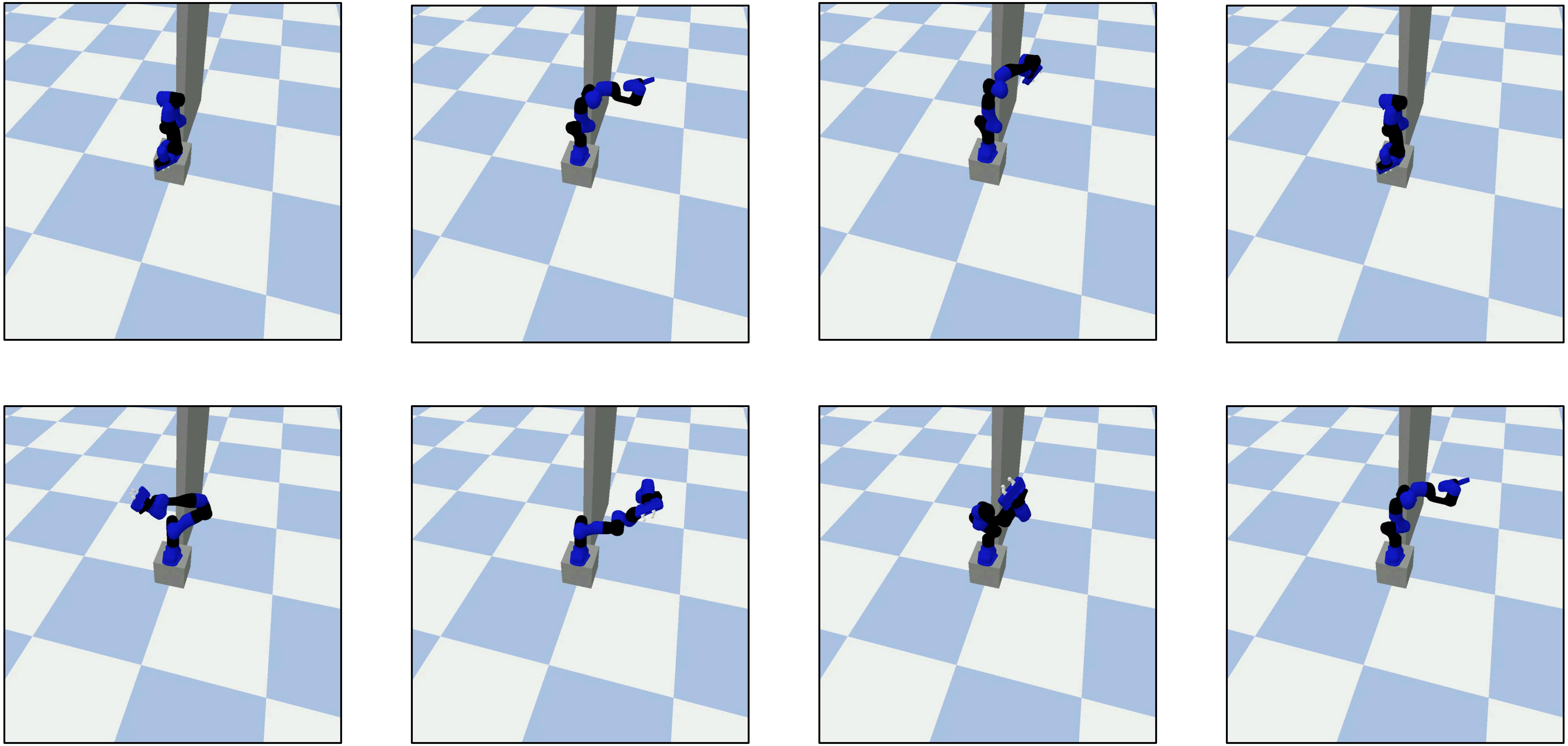
SGT-PG - Trajectory Example

Start
Sub-goal
Sub-goal
Sub-goal
Goal

SGT-PG - Trajectory Example

Model	# Sub-goals	*self-collision*	*wall*
SGT-PG	1	**1.±0.**	0.896±0.016
	3	**1.±0.**	**0.973±0.007**
	7	0.996±0.007	**0.973±0.007**
SeqSG	1	**1.±0.**	0.676±0.034
	3	0.983±0.007	0.593±0.047
	7	0.88±0.03	0.487±0.037

SGT-PG - Coverage of State Space!



Conclusion

- SGT - New multi-goal RL framework
  - First principle – all-pairs shortest path
  - **Provably** more efficient in multi-goal setting
  - Basis for many new algorithms

Conclusion

- SGT - New multi-goal RL framework
  - First principle – all-pairs shortest path
  - **Provably** more efficient in multi-goal setting
  - Basis for many new algorithms
- Future work
  - Stochastic systems (e.g., update plan MPC fashion)
  - Exploration
  - High-dim observations (images)

Conclusion

- SGT - New multi-goal RL framework
  - First pr
  - **Proval**
  - Basis f
- Future w
  - Stocha
  - Explora
  - High-dim observations (images)

Come find us in the virtual poster session!

or reach out:

tomj@campus.technion.ac.il