15-887
Planning, Execution and Learning

Learning in Planning: Experience Graphs

Maxim Likhachev

Robotics Institute

Carnegie Mellon University

Robots Often Perform Repetitive Tasks

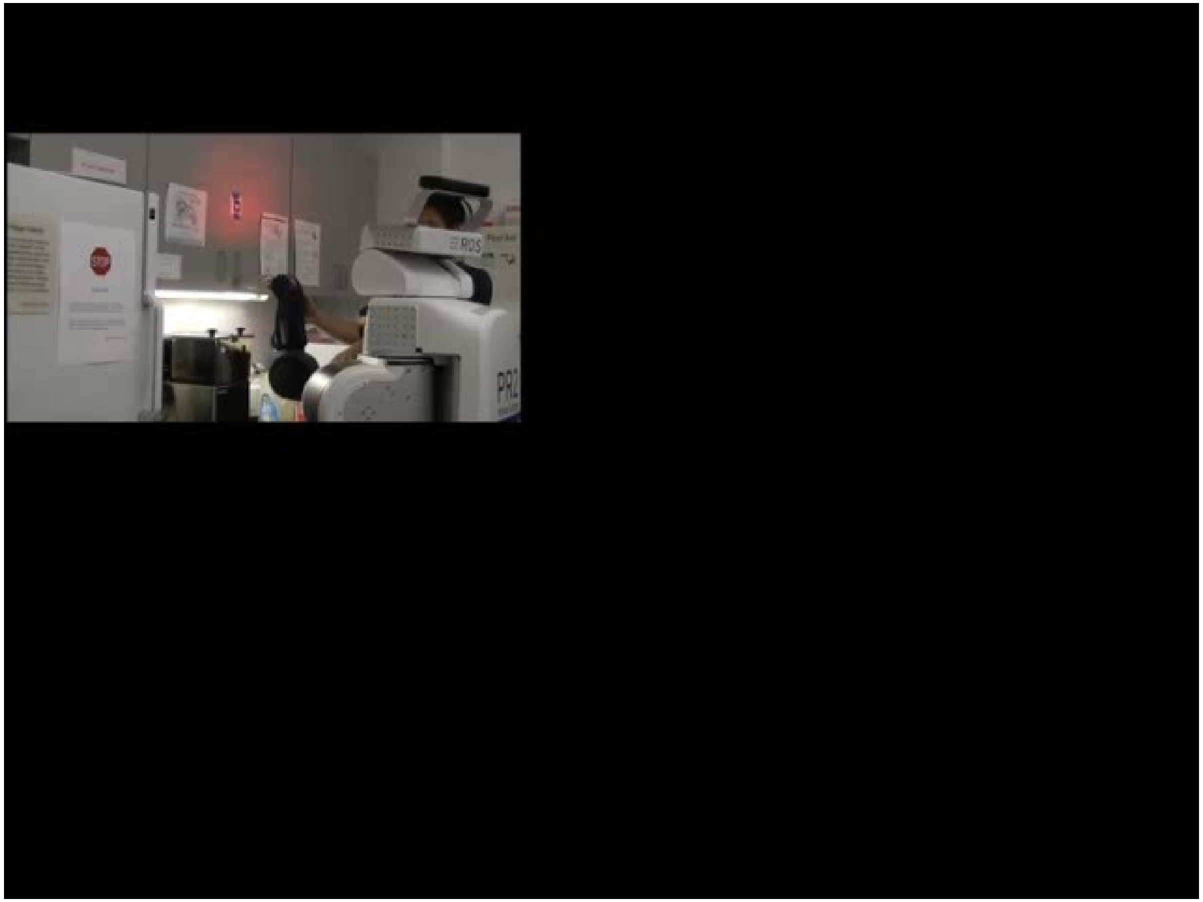
[Cowley et al., '13]

Robots Often Perform Repetitive Tasks

Can we re-use prior **experiences**
to accelerate heuristic search?

[Cowley et al., '13]

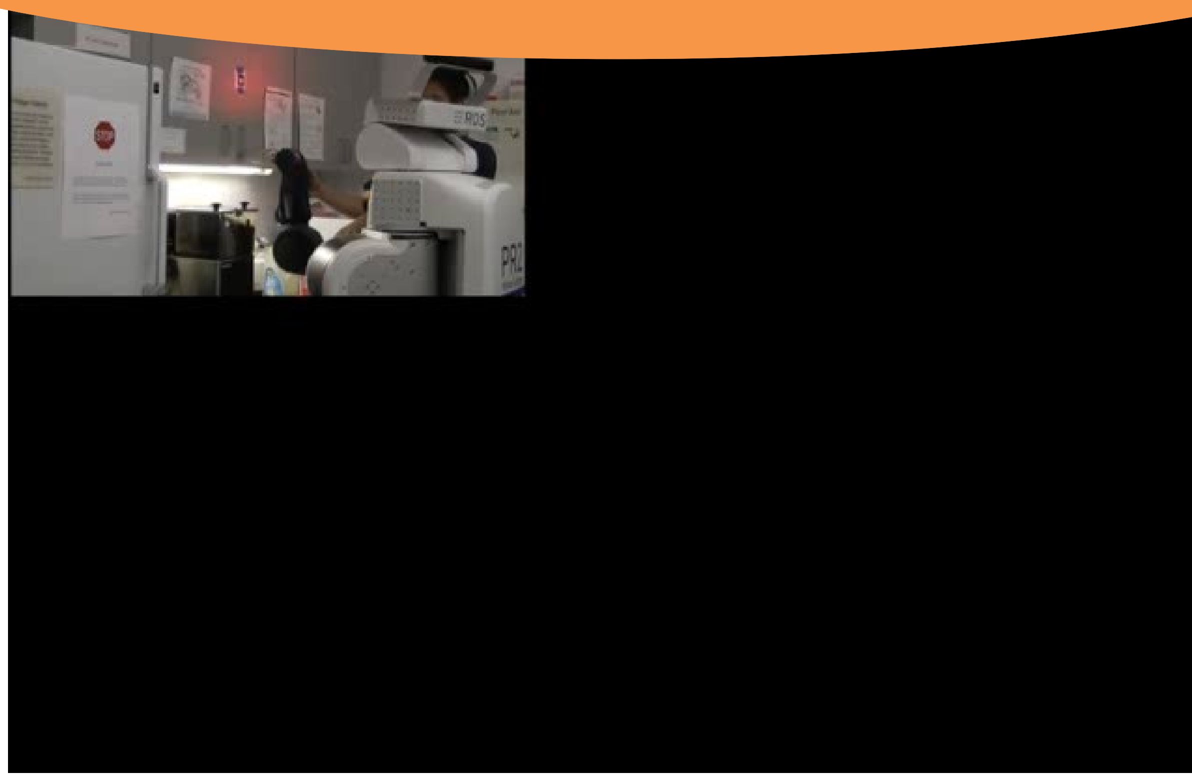
Learning from Demonstrations



[Phillips et al., ‘13]

Learning from Demonstrations

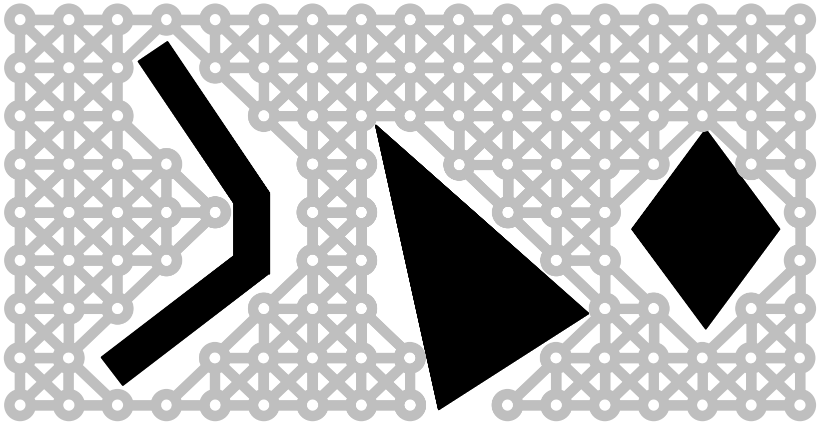
Can we re-use prior **demonstrations**
to accelerate heuristic search?



[Phillips et al., '13]

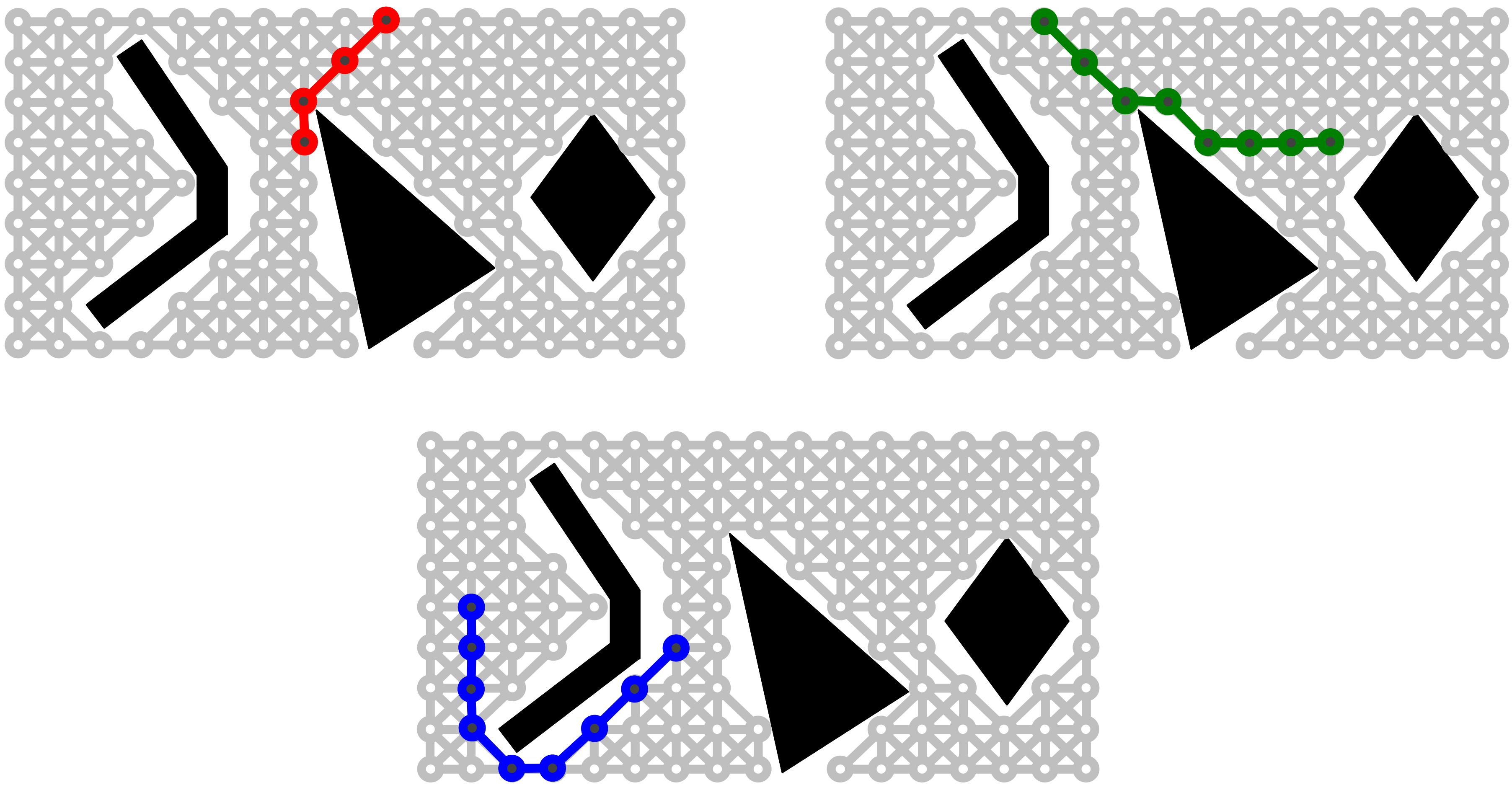
Experience Graphs [Phillips et al., RSS'12]

Consider original graph $G = \{V,E\}$



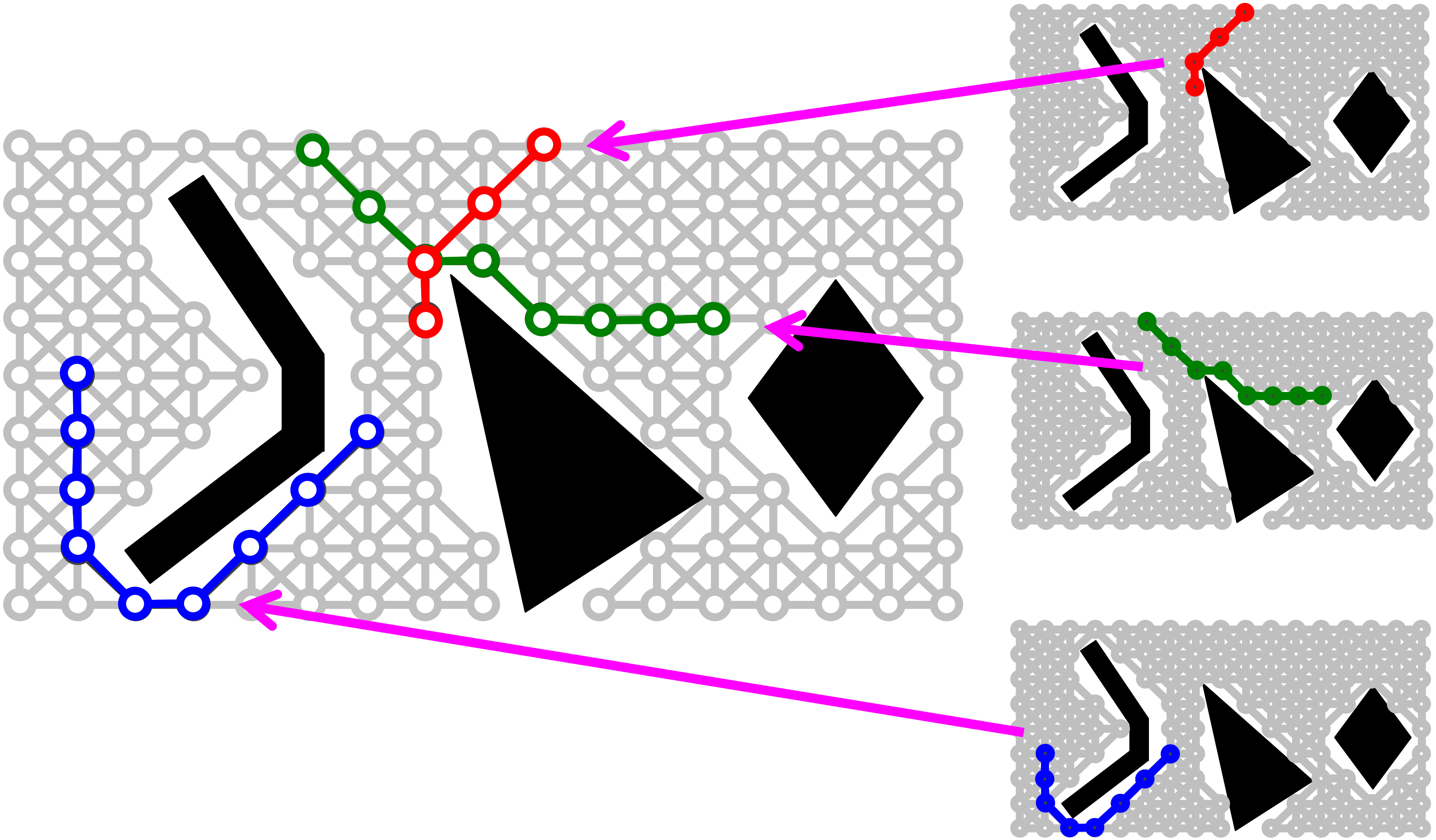
Experience Graphs [Phillips et al., RSS'12]

Given a set of previous paths (experiences or demonstrations)…



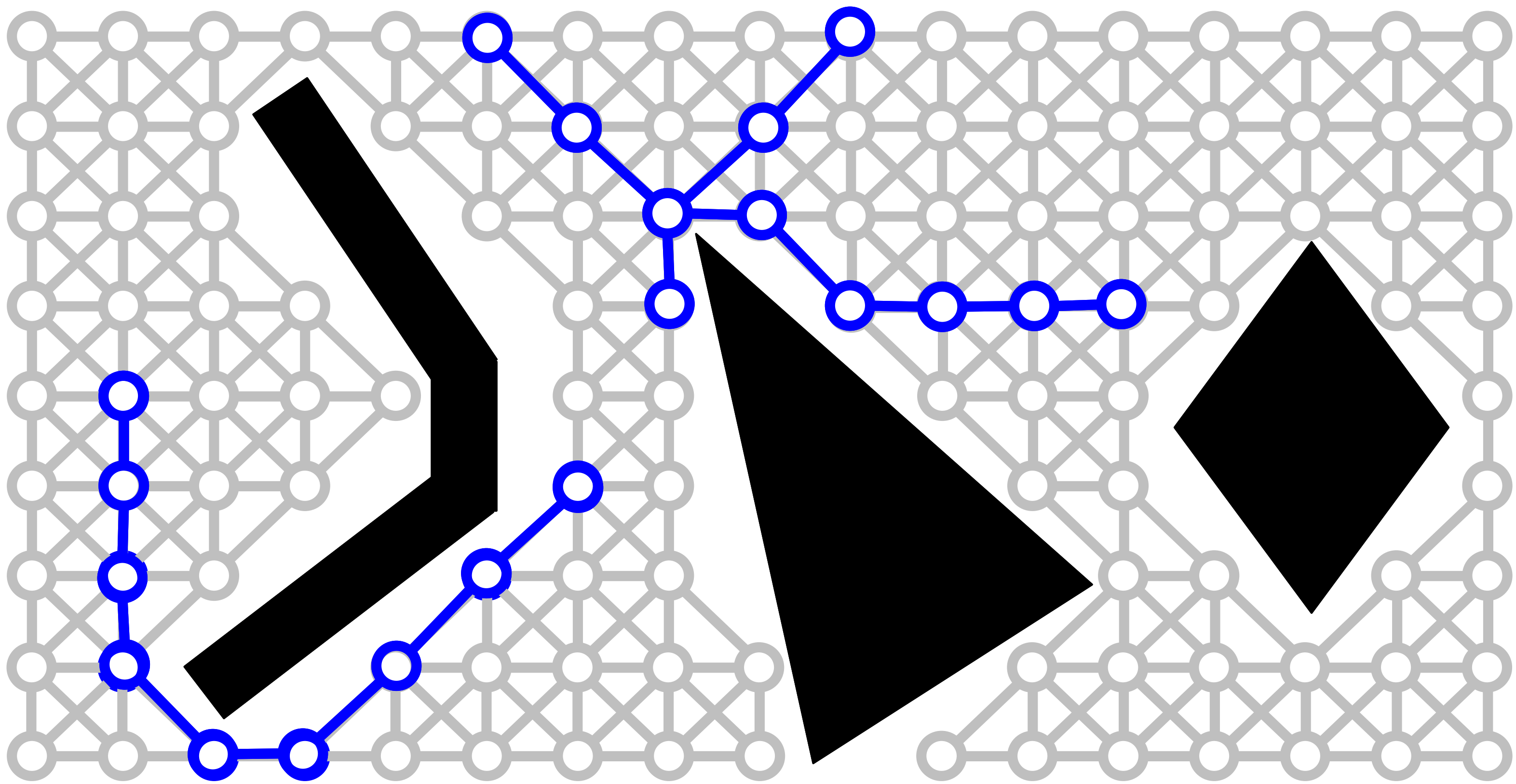
Experience Graphs [Phillips et al., RSS'12]

Put them together into an *E*-graph (Experience graph), $G^E = \{V^E, E^E\}$



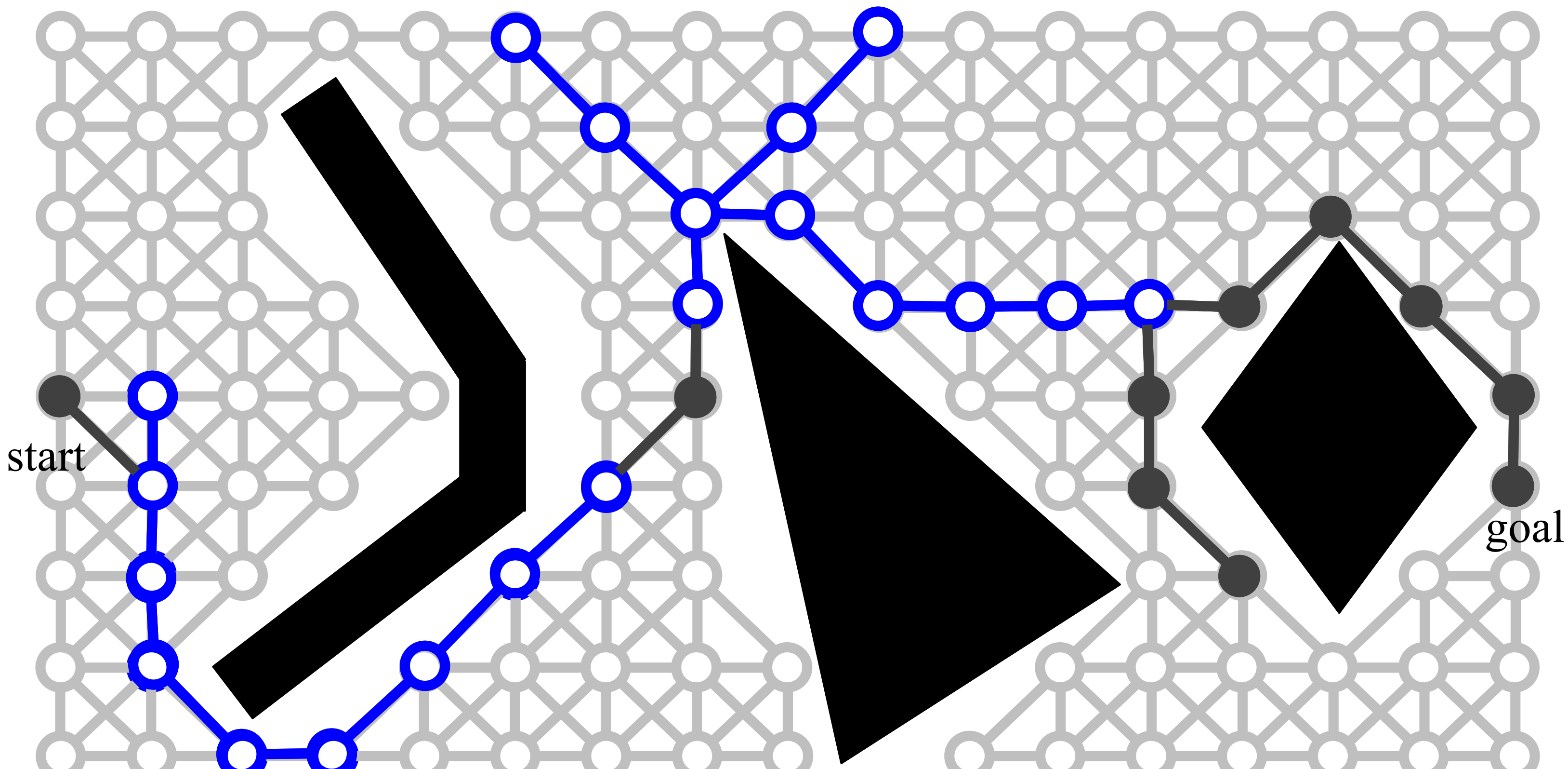
Experience Graphs [Phillips et al., RSS'12]

Given a new planning query…



Experience Graphs [Phillips et al., RSS'12]

…re-use E-graph G^E. For repetitive tasks, planning becomes much faster

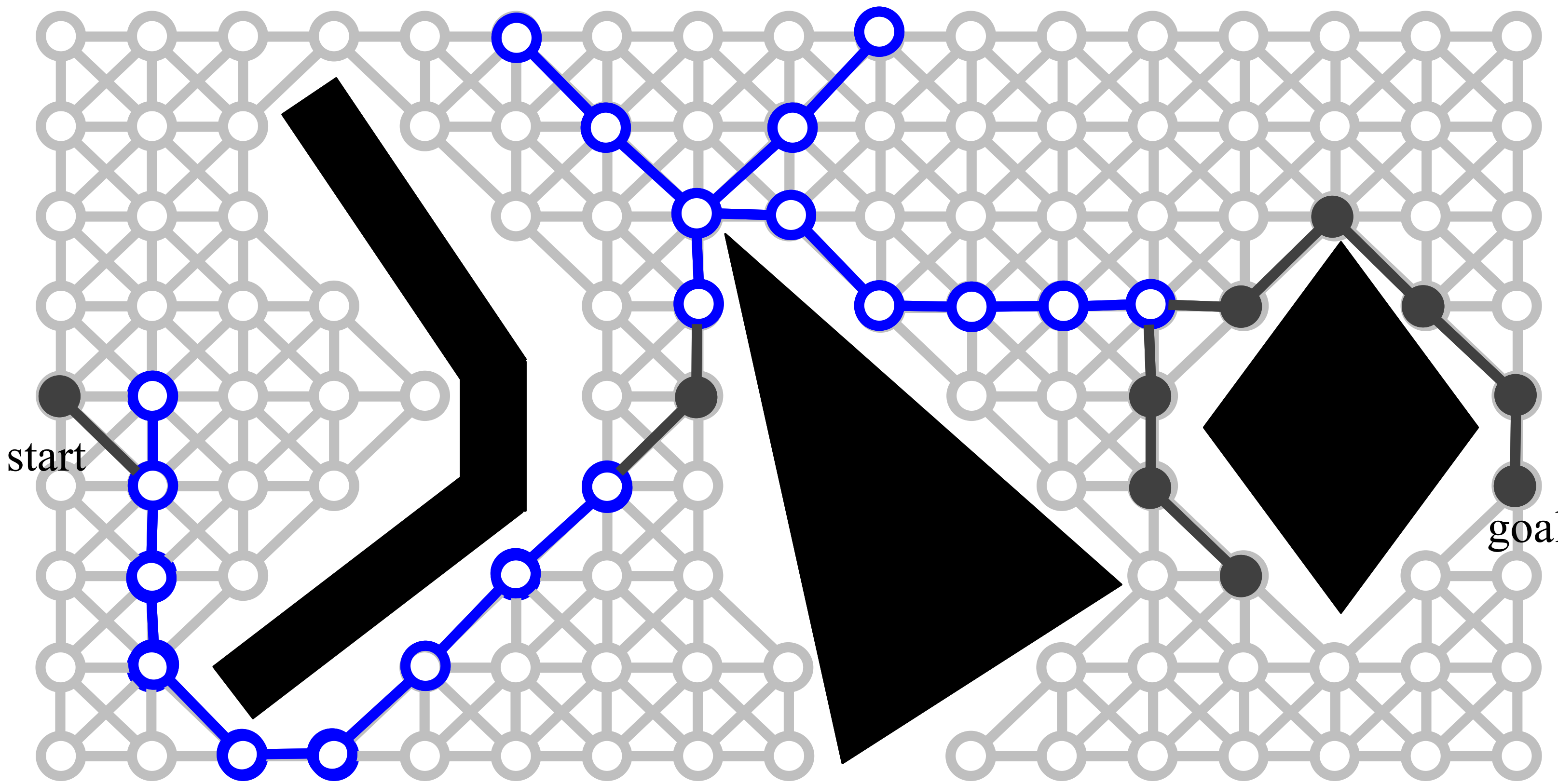


states expanded shown in grey

Experience Graphs [Phillips et al., RSS'12]

…re-use E-graph G^E. For repetitive tasks, planning becomes much faster

for which domains it would be useful and for which useless?

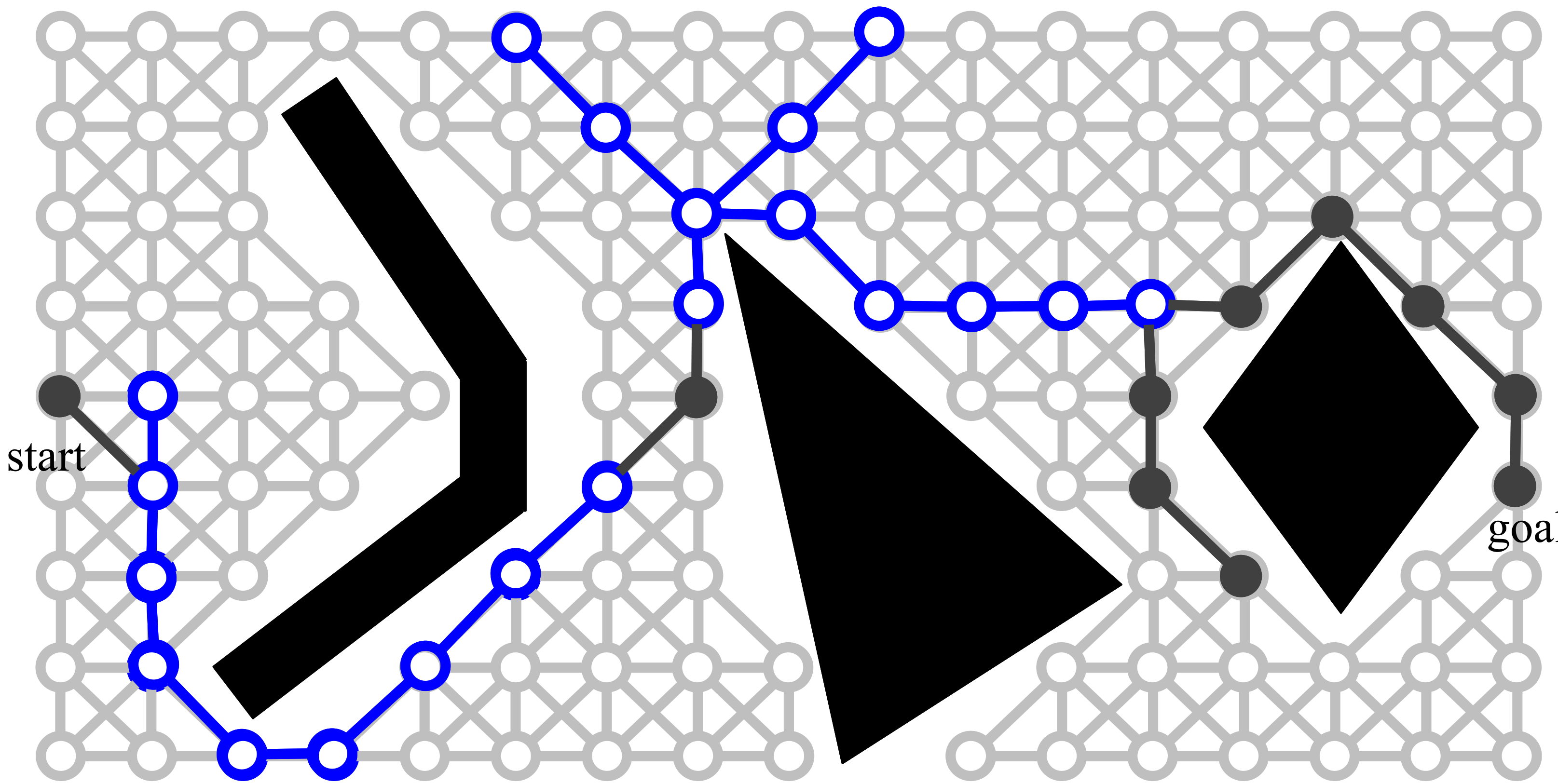


states expanded shown in grey

Experience Graphs [Phillips et al., RSS'12]

…re-use E-graph G^E. For repetitive tasks, planning becomes much faster

General idea:
Instead of using traditional heuristic to bias the search towards the goal, ***compute a new heuristics $h^E(s)$ that biases the search towards a set of paths in G^E.***



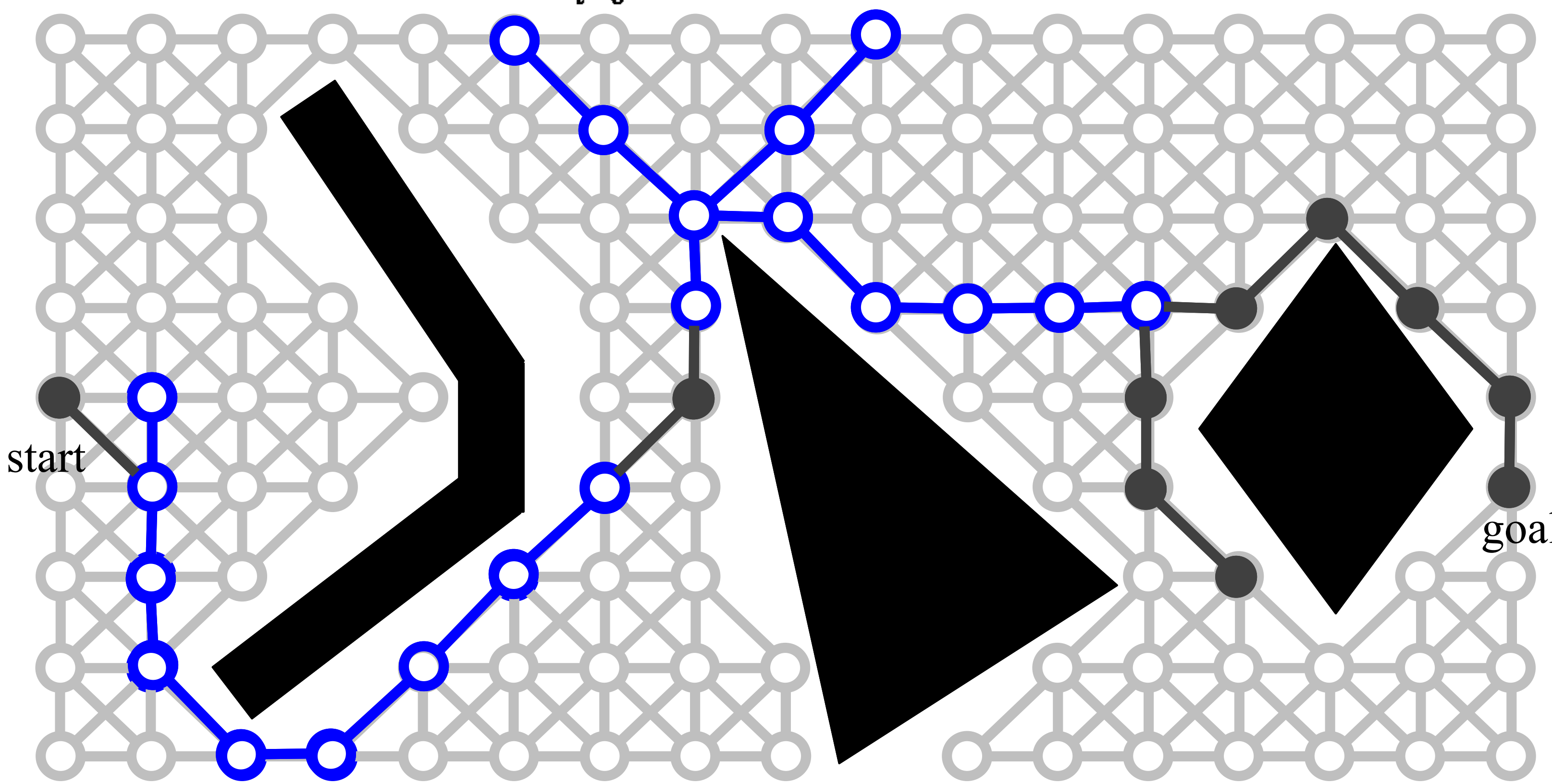
states expanded shown in grey

Experience Graphs [Phillips et al., RSS'12]

…re-use E-graph G^E. For repetitive tasks, planning becomes much faster

For any state s_0 in G, the new heuristic $h^E()$ function is:

$$h^E(s_0) = \min_{\pi(s_0, s_{goal})} \sum_{i=0}^{length(\pi)-1} \min\left(\varepsilon^E h^G(s_i, s_{i+1}), c^E(s_i, s_{i+1})\right)$$

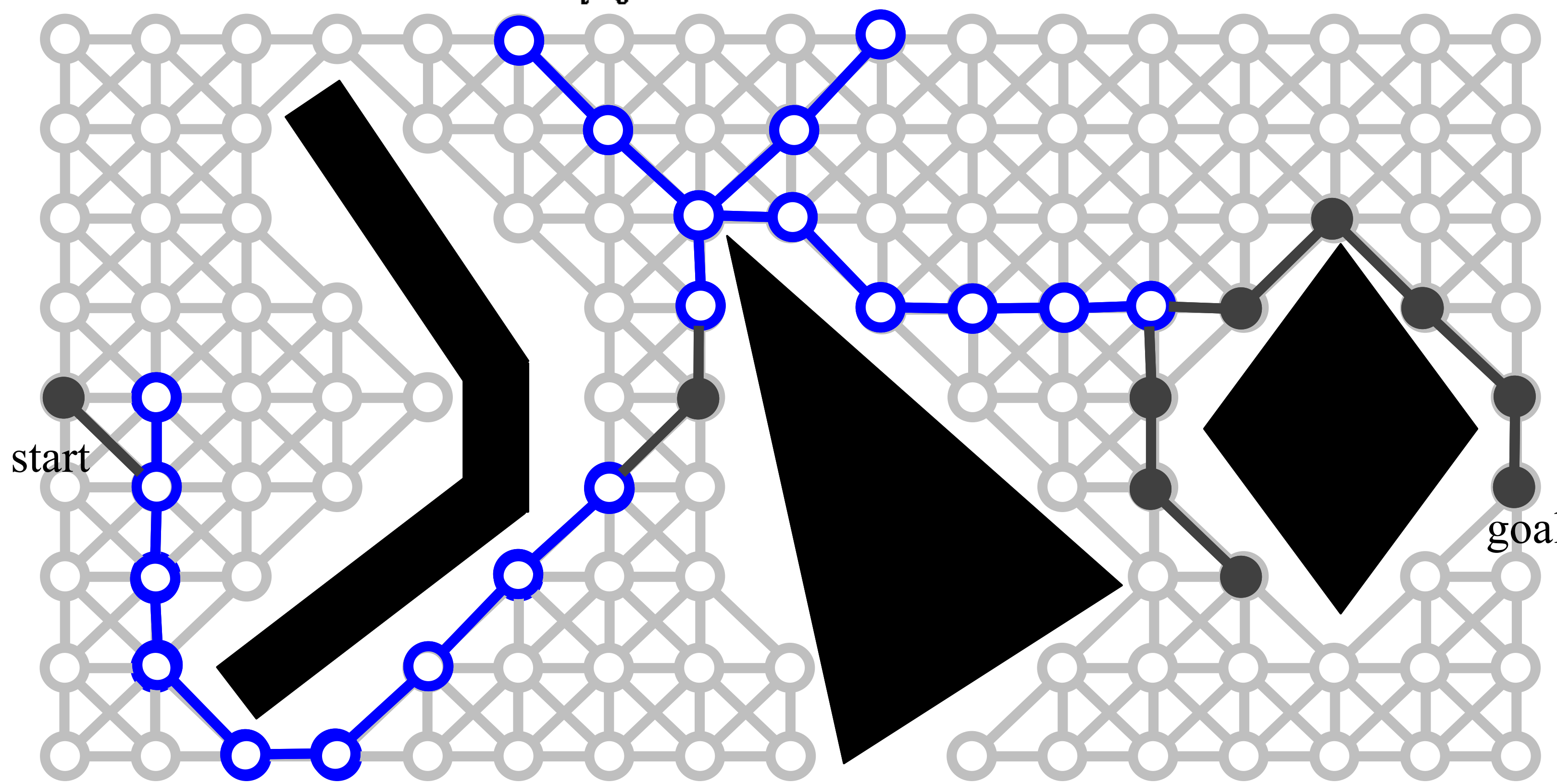


states expanded shown in grey

Experience Graphs [Phillips et al., RSS'12]

heuristics $h^E(s)$ biases the search towards those paths that can be used to get to the goal with the least amount of traversal outside of G^E.

$$h^E(s_0) = \min_{\pi(s_0, s_{goal})} \sum_{i=0}^{length(\pi)-1} \min\left(\varepsilon^E h^G(s_i, s_{i+1}), c^E(s_i, s_{i+1})\right)$$



states expanded shown in grey

Planning with Experience Graphs Pseudocode

Given a planning graph $G = \{V, E\}$

Initialize E-graph $G^E = \{V^E=0, E^E=0\}$

Every time a new planning query (s_{start}, s_{goal}) comes in

re-compute heuristic h^E

run weighted A search with heuristics h^E inflated by weight w;*

execute the found path π

$G^E = G^E \cup \pi$ //add the found path to the E-graph

Planning with Experience Graphs Pseudocode

Given a planning graph $G = \{V, E\}$

Initialize E-graph $G^E = \{V^E=0, E^E=0\}$

Every time a new planning query (s_{start}, s_{goal}) comes in

re-compute heuristic h^E

run weighted A search with heuristics h^E inflated by weight w;*

execute the found path π

$G^E = G^E \cup \pi$ //add the found path to the E-graph

$$h^E(s_0) = \min_{\pi(s_0, s_{goal})} \sum_{i=0}^{length(\pi)-1} \min\left(\varepsilon^E h^G(s_i, s_{i+1}), c^E(s_i, s_{i+1})\right)$$

For any state s, $h^E(s)$ is the cost of a least cost path from s to s_{goal} in a graph G' ={ V, E'}, where E' consists of:

a) All edges (u,v) in E^E with cost $c^E(u,v)$ (i.e., original cost c(u,v))

b) All pairs (u,v) in V with cost $\varepsilon^E h^G(u,v)$

Planning with Experience Graphs Pseudocode

How do you efficiently compute $h^E()$?

Given a plannin...

Initialize E-graph $G^E = \{V^E=0, E^E=0\}$

Every time a new planning query (s_{start}, s_{goal}) comes in

re-compute...

run weighted A... weight w;

What is $h^E()$ with no prior experiences ($G^E = 0$)?

execute the fou...

$G^E = G^E \cup \pi$...

Time for the whiteboard example

$$h^E(s_0) = \min_{\pi(s_0, s_{goal})} \sum_{i=0}^{length(\pi)-1} \min\left(\varepsilon^E h^G(s_i, s_{i+1}), c^E(s_i, s_{i+1})\right)$$

For any state s, $h^E(s)$ is the cost of a least cost path from s to s_{goal} in a graph G' = { V, E'}, where E' consists of:

a) All edges (u,v) in E^E with cost $c^E(u,v)$ (i.e., original cost c(u,v))

b) All pairs (u,v) in V with cost $\varepsilon^E h^G(u,v)$

Efficient Computation of h^E

re-compute heuristic h^E *for all states s' in* $\{V^E \cup s_{goal}\}$:

run a <u>*single*</u> *Dijkstra's on graph* $G^f = \{V^f, E^f\}$, *where*

a) $V^f = \{V^E \cup s_{goal}\}$ *and* E^f = *all pairs of states in* V^f

b) $cost(u,v) = c(u,v)$ *if* (u,v) *in* E^E

$cost(u,v) = \varepsilon^E h^G(u,v)$ *otherwise*

during the weighted A search itself, for any state s:*

$$h^E(s) = \min_{s' \text{ in } \{V^E \cup s_{goal}\}} (\varepsilon^E h^G(s,s') + h^E(s'))$$

Planning with Experience Graphs Pseudocode

Given a planning graph $G = \{V, E\}$

Initialize E-graph $G^E = \{V^E=0, E^E=0\}$

Every time a new planning query (s_{start}, s_{goal}) comes in

re-compute heuristic h^E

heuristics $h^\varepsilon(s)$ is guaranteed to be ε-consistent

$G^E = G^E \cup \pi$

$$h^E(s_0) = \min_{\pi(s_0, s_{goal})} \sum_{i=0}^{length(\pi)-1} \min\left(\varepsilon^E h^G(s_i, s_{i+1}), c^E(s_i, s_{i+1})\right)$$

Plannin… Pseudocode

Theorem 1: *Planning with E-graphs is complete with respect to the original graph*

Theorem 2: *When running weighted A* with inflation w:*

cost(solution) ≤ w·ε^E·cost(optimal solution)

re-compu…

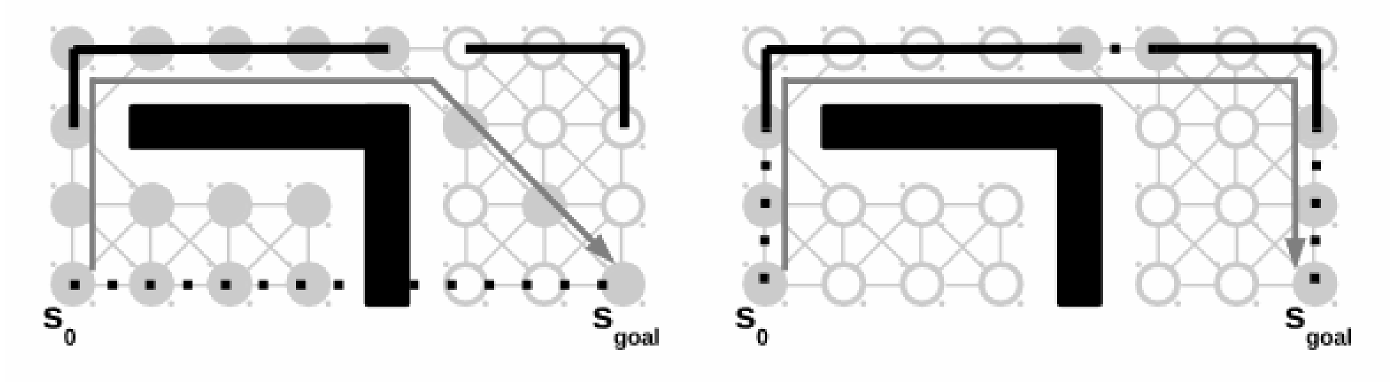
heuristics $h^\varepsilon(s)$ is guaranteed to be ε-consistent

$G^E = G^E \cup \pi$ … graph

$$h^E(s_0) = \min_{\pi(s_0, s_{goal})} \sum_{i=0}^{length(\pi)-1} \min\left(\varepsilon^E h^G(s_i, s_{i+1}), c^E(s_i, s_{i+1})\right)$$

Effect of $\mathcal{E}^E$

- $\mathcal{E}^E$ controls how much the search can rely on experiences/demonstrations in E-graph



(a) $\varepsilon^E = 1$ (b) $\varepsilon^E \to \infty$

Summary

- Planning with Experience Graphs – a method for biasing search towards the reuse of previously planned paths and demonstrations
  - Based on the idea of re-computing heuristics to guide the search towards a set of paths rather than towards a goal
  - Not all domains may benefit from reusing prior paths
  - Useful in domains where a robot has a similar workspace across planning instances (e.g., manipulation for manufacturing)