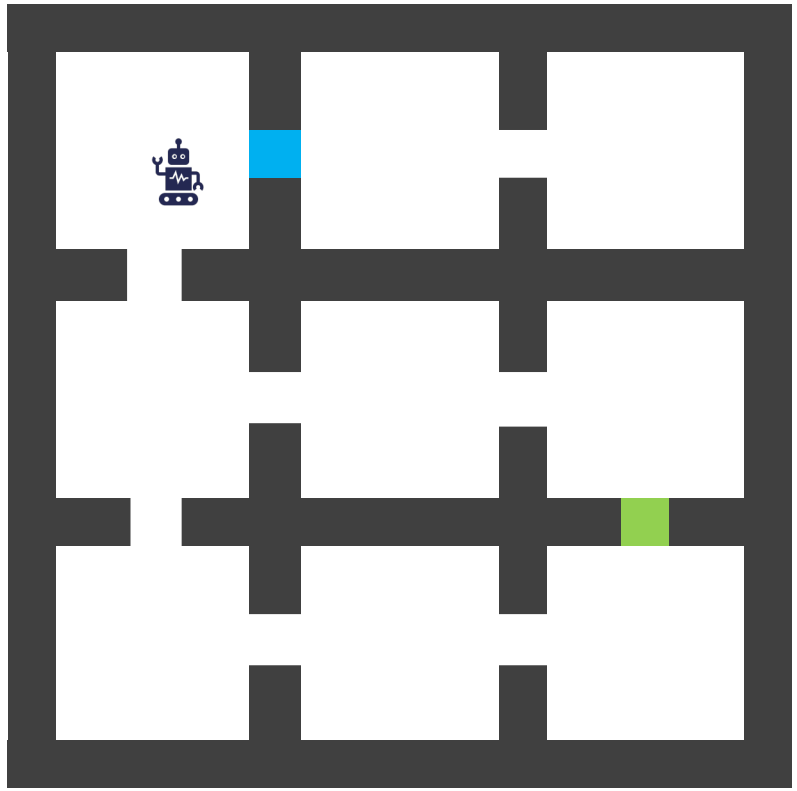
Abstract Value Iteration for Hierarchical RL

Kishor Jothimurugan, Osbert Bastani, Rajeev Alur

Hierarchical RL



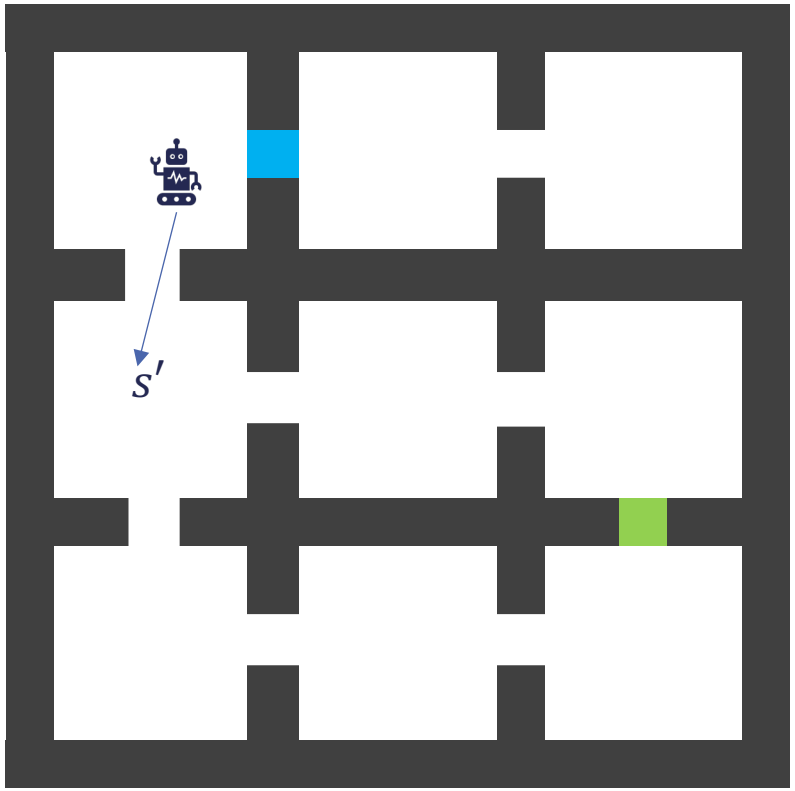
Continuous state and action spaces

State $s = (x, y) \in \mathbb{R}^2$

Action $a = (v_x, v_y) \in \mathbb{R}^2$

- Learn a policy to go from ■ to ■ ?
- Challenging task for RL even if robot dynamics are simple.
  - Requires a lot of exploration.
  - Can get stuck in a local optimum.
- Hierarchical RL is a promising approach for such problems.

Hierarchical RL



Continuous state and action spaces

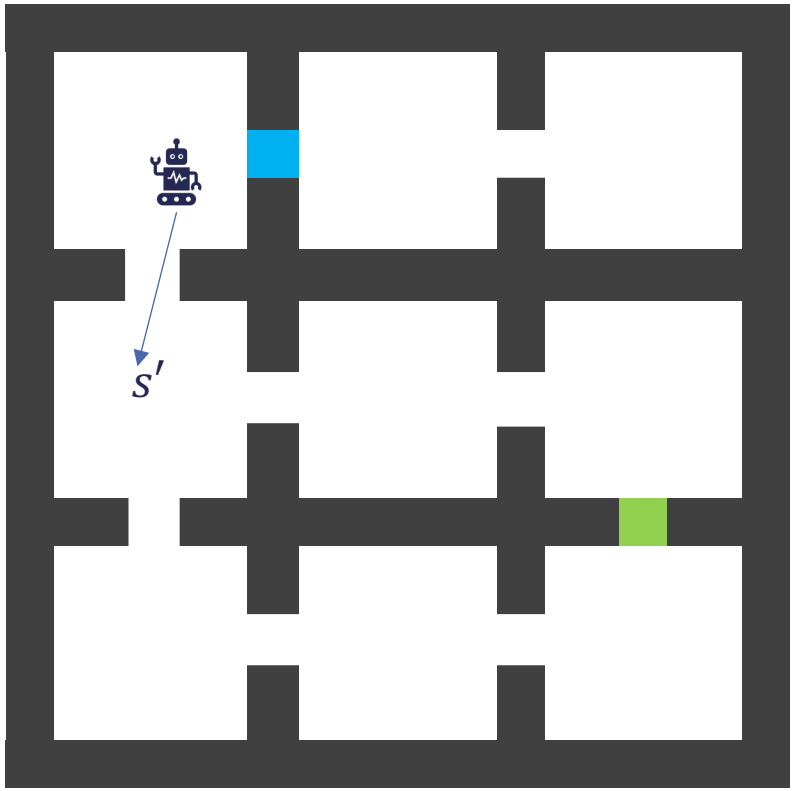
State $s = (x, y) \in \mathbb{R}^2$

Action $a = (v_x, v_y) \in \mathbb{R}^2$

Policy is decomposed into two policies[1].

- A high-level policy π_h:
  - Maps a state to next intermediate goal, $\pi_h(s) = s' \in S$.
- A low-level policy π_l:
  - Maps a state and intermediate goal to an action, $\pi_l(s, s') = a \in A$.

Hierarchical RL



Continuous state and action spaces

State $s = (x, y) \in \mathbb{R}^2$

Action $a = (v_x, v_y) \in \mathbb{R}^2$

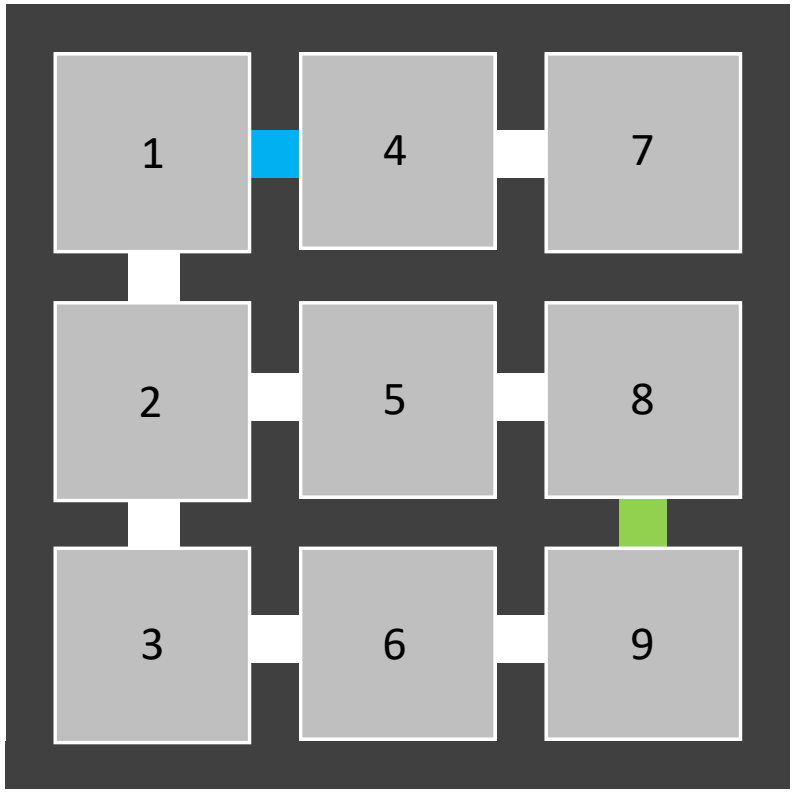
ADVANTAGES

- Decomposes the problem into two simpler RL problems.
- High level exploration is better.

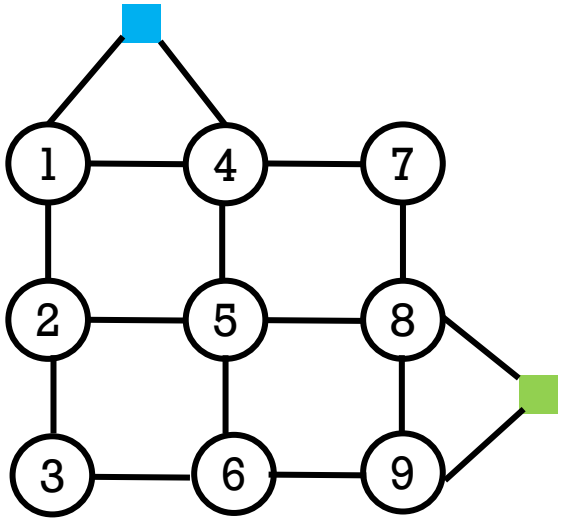
SHORTCOMINGS

- Can get stuck in local optimum.
- Does not exploit structure of the problem to explore systematically.

Abstract States



Abstract states in grey



Abstract graph

- Abstract states are disjoint from each other.
- Abstract graph indicates which abstract states are nearby.
- Domain expert provides abstract states and abstract graphs.

Our Hierarchical RL Framework

Given abstract states and abstract graph,

1. Learn policies to go from every abstract state to every adjacent abstract state.
2. Use high level planning to find a path from start to goal.

QUESTIONS

1. How to plan using learned policies?
2. Under what conditions on the abstract states, is the learned policy provably good?

Value Iteration using Options

An option is a tuple $o = (\pi, I, \beta)$ where

- $\pi : S \to A$ is a policy
- $I \subseteq S$ is a set of states at which the option is available
- $\beta : S \to [0,1]$ and $\beta(s)$ is the probability that the option terminates at s.

For options we can define rewards and discounted transition probabilities.

$$R_{\text{opt}}(s, o) = \mathbb{E}_{s_0, a_0, \ldots, s_t \sim o} \left[\sum_{i=0}^{t-1} \gamma^i R(s_i, a_i) \mid s_0 = s \right],$$

$$T_{\text{opt}}(s, o, s') = \sum_{t=1}^{\infty} \gamma^t p(s' \mid t, s, o) P(t \mid s, o)$$

Value Iteration using Options

An option is a tuple $o = (\pi, I, \beta)$ where

- $\pi : S \to A$ is a policy
- $I \subseteq S$ is a set of states at which the option is available
- $\beta : S \to [0,1]$ and $\beta(s)$ is the probability that the option terminates at s.

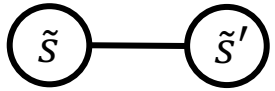
Value iteration using options corresponds to solving the Bellman equations,

$$V^*_{\mathcal{O}}(s) = \max_{o \in \mathcal{O}} Q^*_{\mathcal{O}}(s, o),$$

$$\rho^*(s) = \arg\max_{o \in \mathcal{O}} Q^*_{\mathcal{O}}(s, o)$$

$$Q^*_{\mathcal{O}}(s, o) = R_{\text{opt}}(s, o) + \int_S T_{\text{opt}}(s, o, s') V^*_{\mathcal{O}}(s') ds'.$$

Robust Abstract Value Iteration (R-AVI)

For each edge in abstract graph  we train an option $o = (\pi, I, \beta)$ where

- π is trained to take the system from states in $\tilde{s}$ to states in $\tilde{s}'$
- $I = \tilde{s}$
- β is given by $\beta(s) = 1$ if $s \in \cup \{c : c \neq \tilde{s}\}$ and 0 otherwise.

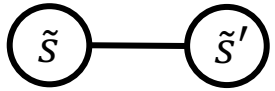
We perform value iteration using these options.

For abstract states, define

$$\tilde{T}_{\inf}(\tilde{s}, o, \tilde{s}') = \inf_{s \in \tilde{s}} \tilde{T}(s, o, \tilde{s}') \qquad \tilde{R}_{\inf}(\tilde{s}, o) = \inf_{s \in \tilde{s}} R_{\text{opt}}(s, o)$$

$$\tilde{T}_{\sup}(\tilde{s}, o, \tilde{s}') = \sup_{s \in \tilde{s}} \tilde{T}(s, o, \tilde{s}') \qquad \tilde{R}_{\sup}(\tilde{s}, o) = \sup_{s \in \tilde{s}} R_{\text{opt}}(s, o).$$

Robust Abstract Value Iteration (R-AVI)

For each edge in abstract graph  we train an option $o = (\pi, I, \beta)$ where

- π is trained to take the system from states in $\tilde{s}$ to states in $\tilde{s}'$
- $I = \tilde{s}$
- β is given by $\beta(s) = 1$ if $s \in \cup \{c : c \neq \tilde{s}\}$ and 0 otherwise.

We perform value iteration using these options.

For abstract states, define

$$\tilde{V}_z^*(\tilde{s}) = \max_{o \in \mathcal{O}} \tilde{Q}_z^*(\tilde{s}, o)$$

$$\tilde{Q}_z^*(\tilde{s}, o) = \tilde{R}_z(\tilde{s}, o) + \sum_{\tilde{s}' \in \tilde{\mathcal{S}}} \tilde{T}_z(\tilde{s}, o, \tilde{s}') \cdot \tilde{V}_z^*(\tilde{s}')$$

Theoretical Guarantees

Define worst-case error in T and R,

$$\varepsilon_T = \max_{\tilde{s},\tilde{s}' \in \tilde{\mathcal{S}},\ o \in \mathcal{O}} \tilde{T}_{\text{sup}}(\tilde{s}, o, \tilde{s}') - \tilde{T}_{\text{inf}}(\tilde{s}, o, \tilde{s}'), \qquad \varepsilon_R = \max_{\tilde{s} \in \tilde{\mathcal{S}},\ o \in \mathcal{O}} \tilde{R}_{\text{sup}}(\tilde{s}, o) - \tilde{R}_{\text{inf}}(\tilde{s}, o).$$

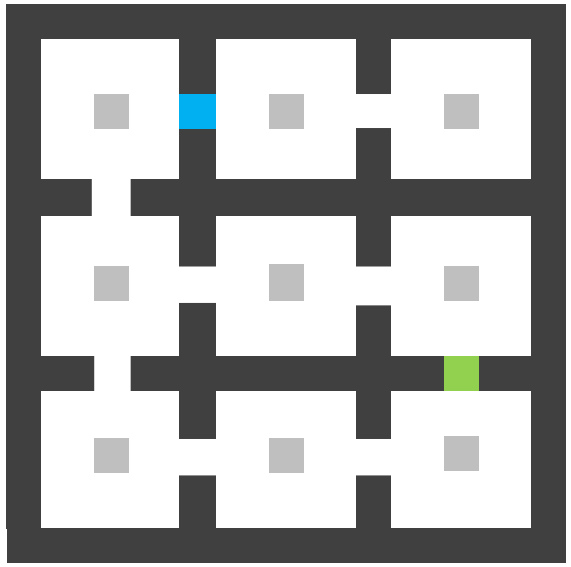
We can show the following about R-AVI:

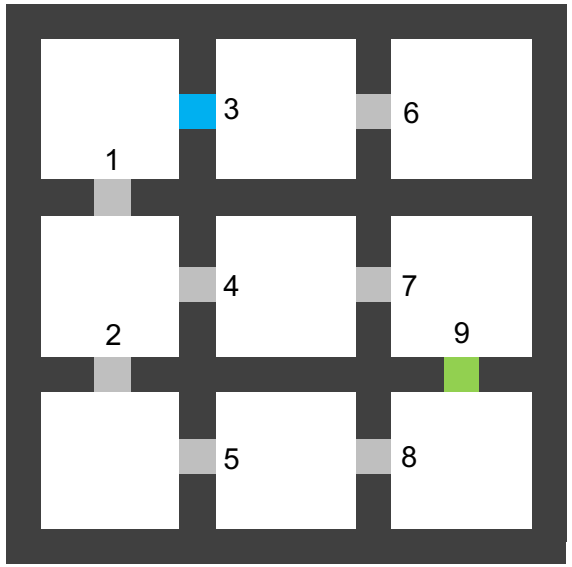
- R-AVI converges if $|\tilde{\mathcal{S}}|\varepsilon_T < 1 - \gamma$.
- We have $\tilde{V}^*_{inf}(\tilde{s}) \leq V^*_{\mathcal{O}}(s) \leq \tilde{V}^*_{sup}(\tilde{s})$.
- If $\tilde{\rho}$ is defined using $\tilde{\rho}(\tilde{s}) = argmax_o\ Q^*_{inf}(\tilde{s}, o)$, then $J(\pi_{\tilde{\rho}}) \geq J(\pi_{\rho^*}) - \frac{(1-\gamma)\varepsilon_R + |\tilde{\mathcal{S}}|\varepsilon_T}{(1-\gamma)(1-(\gamma+|\tilde{\mathcal{S}}|\varepsilon_T))}$

Better Abstract States

Theory suggests that all states in an abstract states should be similar.

Since they don't have to cover the state space, call them **subgoal regions.**





The subgoal regions in the second picture are **bottlenecks!**

Comparison to Best Policy

If the following hold,

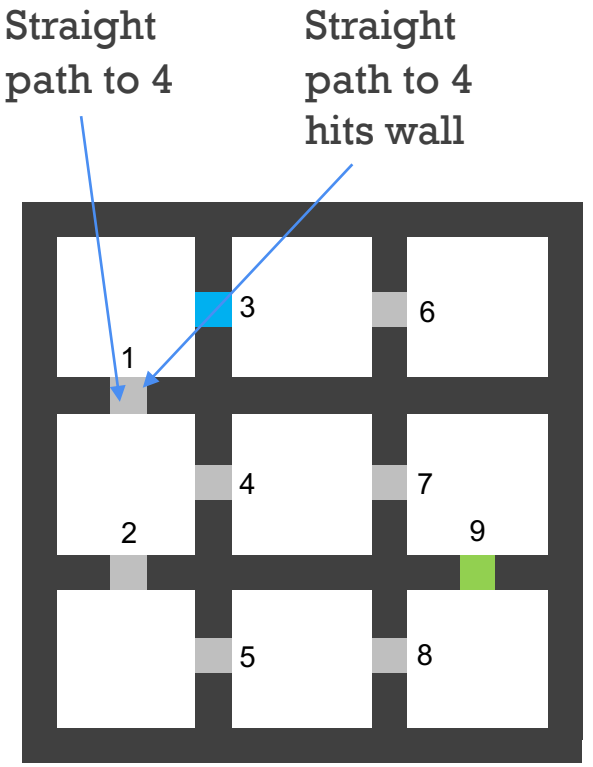
- $|\tilde{\mathcal{S}}|\varepsilon_T < 1 - \gamma.$
- The system is deterministic and has sparse rewards,
- The abstract states are bottlenecks.

Then we can show,

$$J(\pi_{\tilde{\rho}}) \geq J(\pi^*) - \frac{(1-\gamma)\varepsilon_R + |\tilde{\mathcal{S}}|\varepsilon_T}{(1-\gamma)(1-(\gamma+|\tilde{\mathcal{S}}|\varepsilon_T))}.$$

A Problem with R-AVI

- R-AVI assumes worst case behavior of the options.
- Harder to train options to work well in worst case.
- We only have to train to work on states visited during execution!



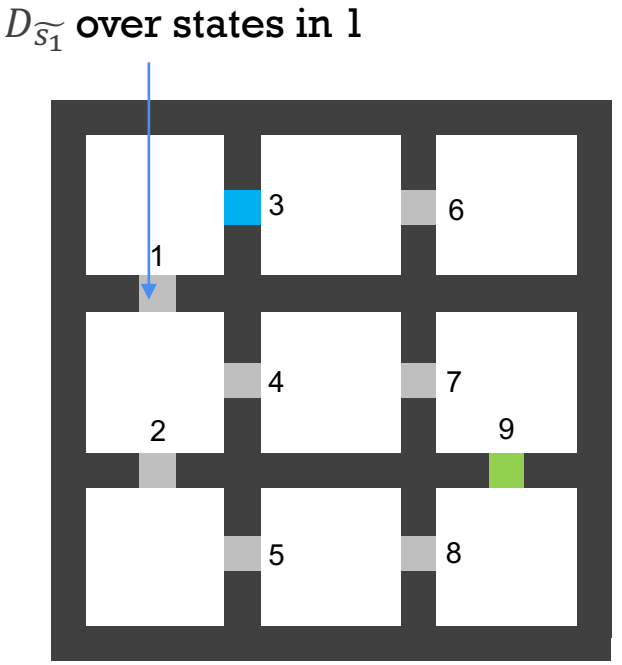
Alternating Abstract Value Iteration (A-AVI)

- Suppose we have distributions $D_{\tilde{s}}$ for every abstract state.
- We can train options to work on these distributions.
- Also define expected values of transition probabilities and rewards.

$$\tilde{T}_{\mathcal{D}}(\tilde{s}, o, \tilde{s}') = \mathbb{E}_{s \sim \mathcal{D}_{\tilde{s}}}[\tilde{T}(s, o, \tilde{s}')]$$
$$\tilde{R}_{\mathcal{D}}(\tilde{s}, o) = \mathbb{E}_{s \sim \mathcal{D}_{\tilde{s}}}[\tilde{R}(s, o)],$$

- Use these values for value iteration.

$$\tilde{V}^*_{\mathcal{D}}(\tilde{s}) = \max_{o \in \mathcal{O}_{\mathcal{D}}} \tilde{Q}^*_{\mathcal{D}}(\tilde{s}, o)$$
$$\tilde{Q}^*_{\mathcal{D}}(\tilde{s}, o) = \tilde{R}_{\mathcal{D}}(\tilde{s}, o) + \sum_{\tilde{s}' \in \tilde{\mathcal{S}}} \tilde{T}_{\mathcal{D}}(\tilde{s}, o, \tilde{s}') \cdot \tilde{V}^*_{\mathcal{D}}(\tilde{s}')$$

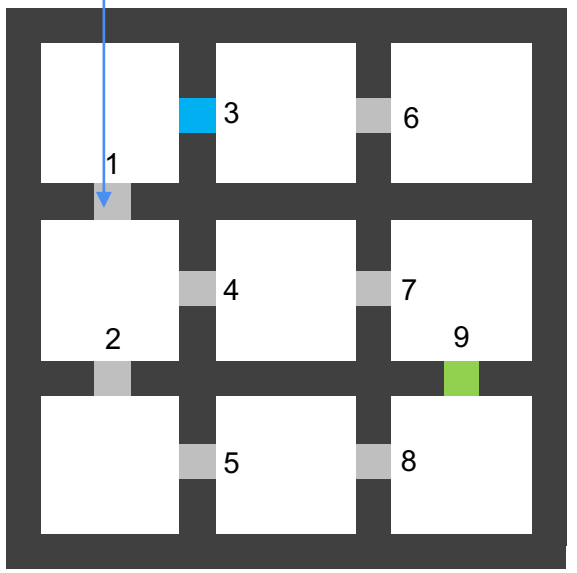


Alternating Abstract Value Iteration (A-AVI)

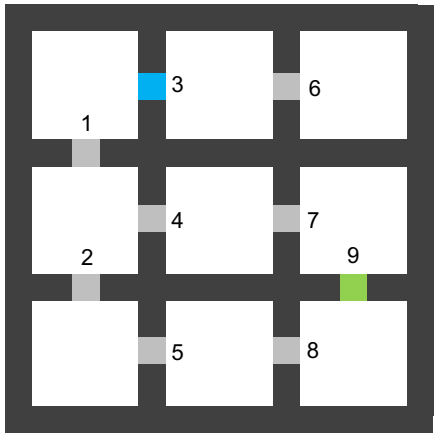
Algorithm 1 (A-AVI) Iterative algorithm for constructing hierarchical policy.

function LearnPolicy($\mathcal{M}, \tilde{\mathcal{S}}, E, N$)
  Initialize $\mathcal{D}$
  for $i \in \{1, ..., N\}$ **do**
    Learn the ideal subgoal transitions $\mathcal{O}_{\mathcal{D}}$
    Estimate $\tilde{T}_{\mathcal{D}}$ and $\tilde{R}_{\mathcal{D}}$ for $\mathcal{O}_{\mathcal{D}}$
    Compute $\tilde{\rho}_{\mathcal{D}}$ using abstract value iteration
    for $\tilde{s} \in \tilde{\mathcal{S}}$ **do**
      $\bar{\mathcal{D}} \leftarrow$ Distribution over $\tilde{s}$ induced by $\pi_{\tilde{\rho}_{\mathcal{D}}}$
      Update $\mathcal{D} \leftarrow (1 - \alpha_i)\mathcal{D} + \alpha_i \bar{\mathcal{D}}$
    end for
  end for
  return $\mathcal{O}_{\mathcal{D}}$, $\pi_{\rho_{\mathcal{D}}}$
end function

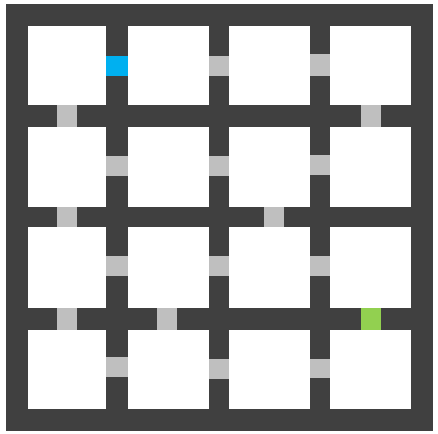
$D_{\widetilde{s_1}}$ over states in 1



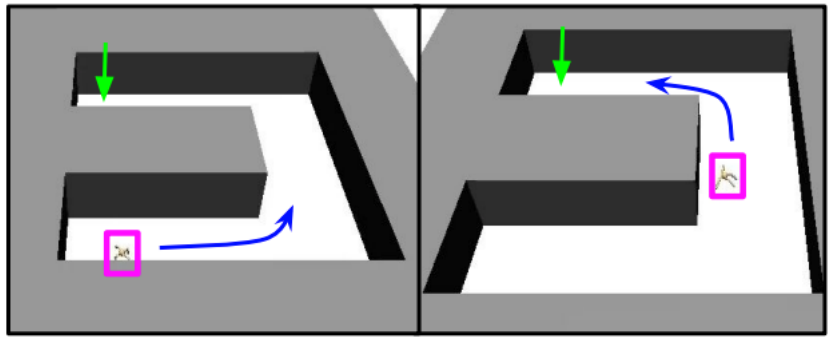
Experiments



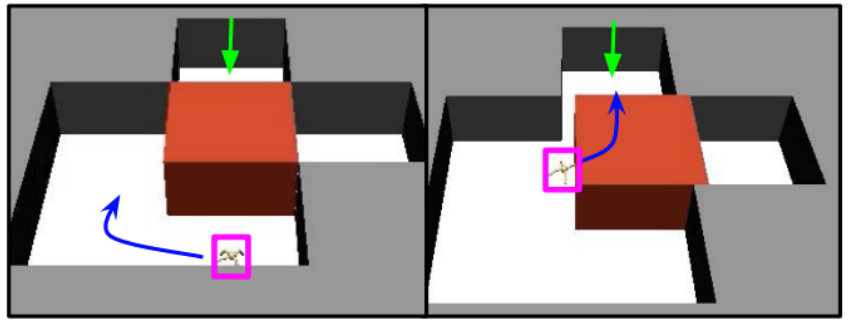
9 Rooms



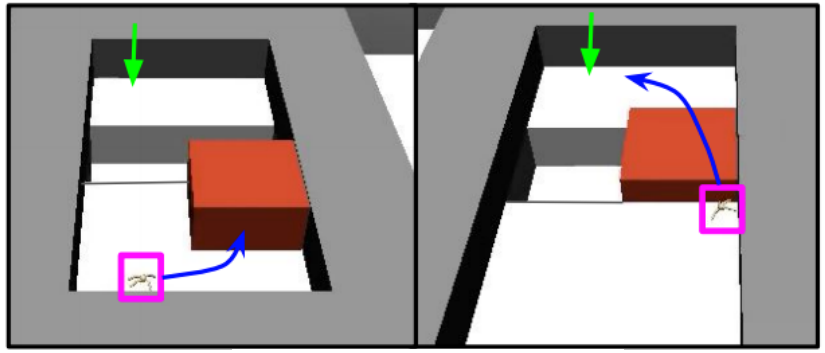
16 Rooms



Ant Maze

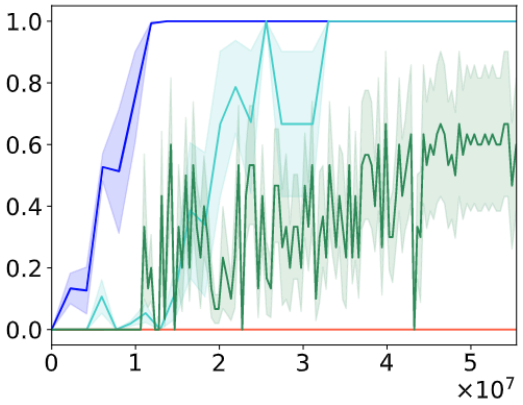


Ant Push

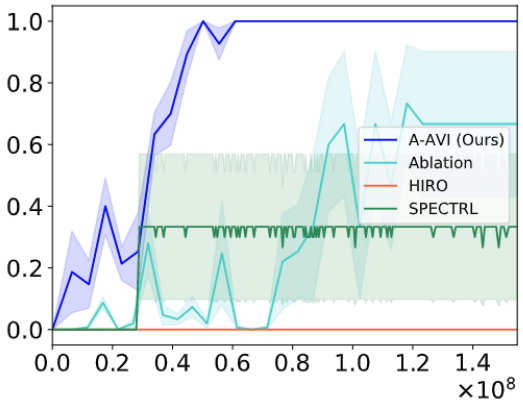


Ant Fall

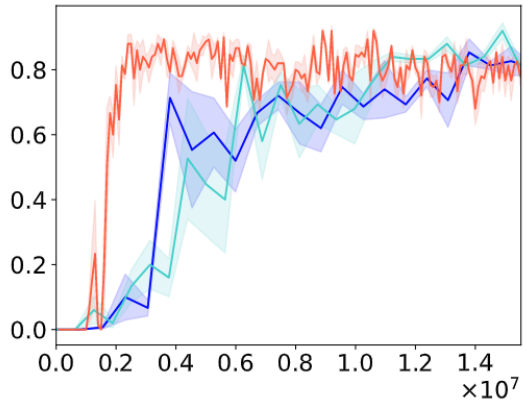
Experiments



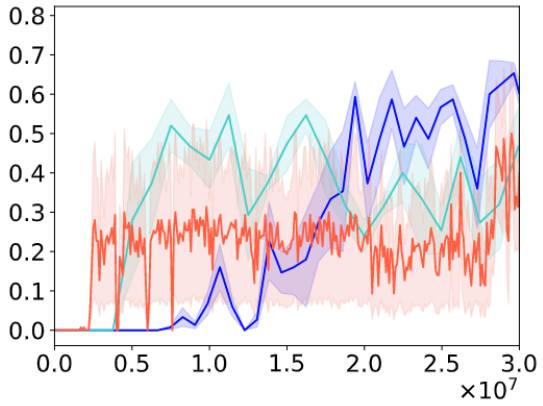
(a) 9-Rooms



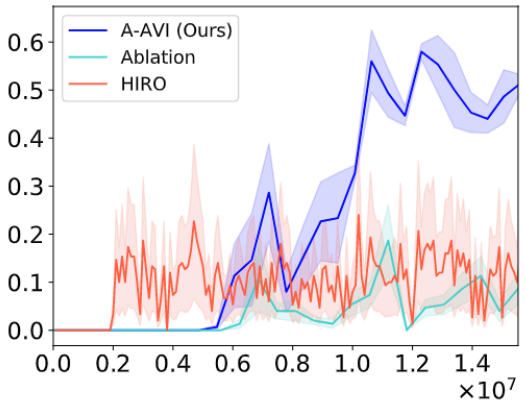
(b) 16-Rooms



(c) AntMaze

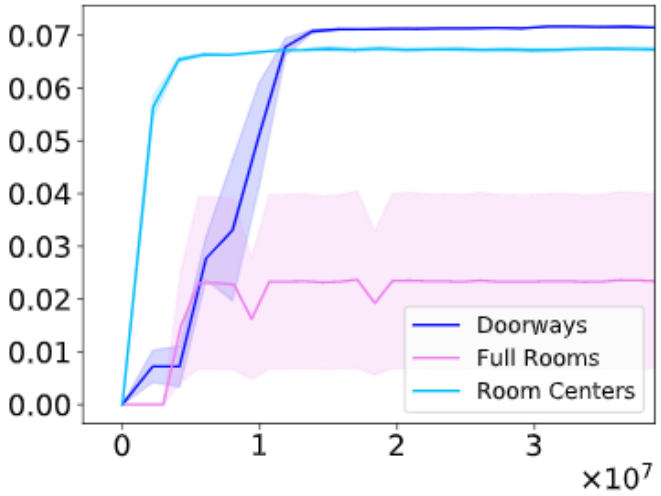


(d) AntPush

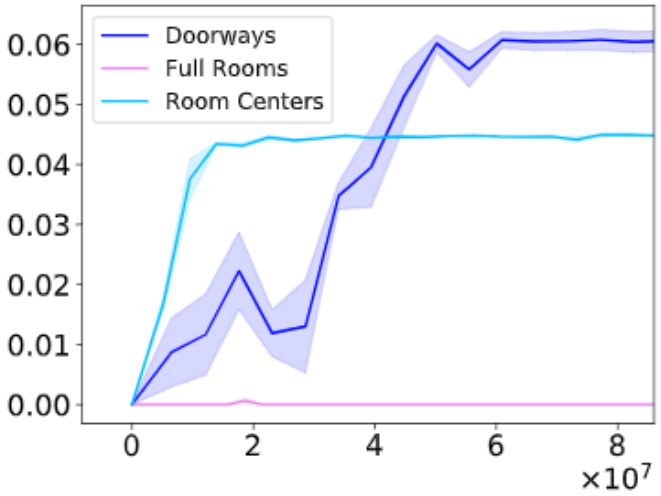


(e) AntFall

Comparison of Abstract States

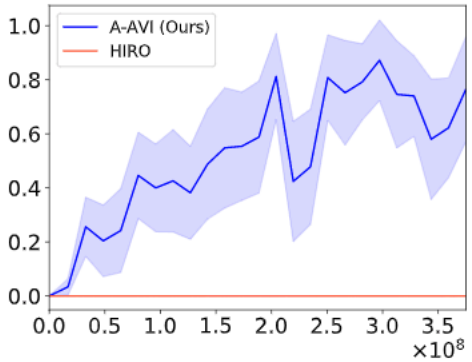


(a) 9-Rooms

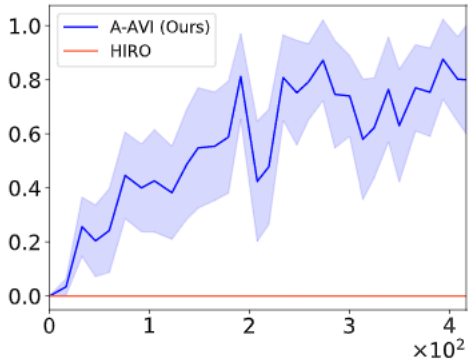


(b) 16-Rooms

Random Abstract States

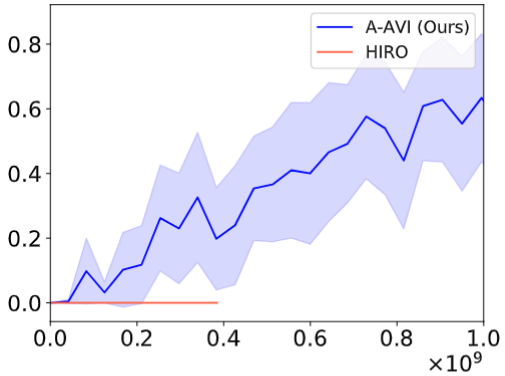


(a) Sample Complexity

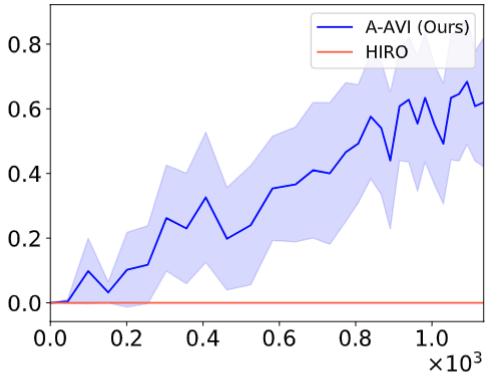


(b) Learning Time

9 Rooms



(b) Sample Complexity



(c) Learning Time

16 Rooms

Thank You!