CSC2621 Topics in Robotics
Reinforcement Learning in Robotics

Week 11: Hierarchical Reinforcement Learning

Animesh Garg

Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning

Richard S. Sutton , Doina Precup , Satinder Singh

Topic: Hierarchical RL

Presenter: Panteha Naderian

Motivation: Temporal abstraction

- Consider an activity such as cooking
  - High-level: Choose a recipe, make grocery List
  - Medium-level: get a pot, put ingredients in the Pot, stir until smooth
  - Low-level: wrist and arm movement, muscle Contraction
- All have to be seamlessly integrated.

Contributions

- Temporal abstraction within the framework of RL by introducing options.
- Applying results from theory of SMDPs for planning and Learning in the context of options.
- Changing and learning option’s internal structure.
    - Interrupting options
    - Sub goals
    - Intra-option learning

Background: MDP

MDP consists of:

- A set of actions
- A set of states
- Transition dynamics: $p_{ss'}^{a} = \Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$
- Expected reward: $r_s^a = E\{r_{t+1} | s_t = s, a_t = a\}$

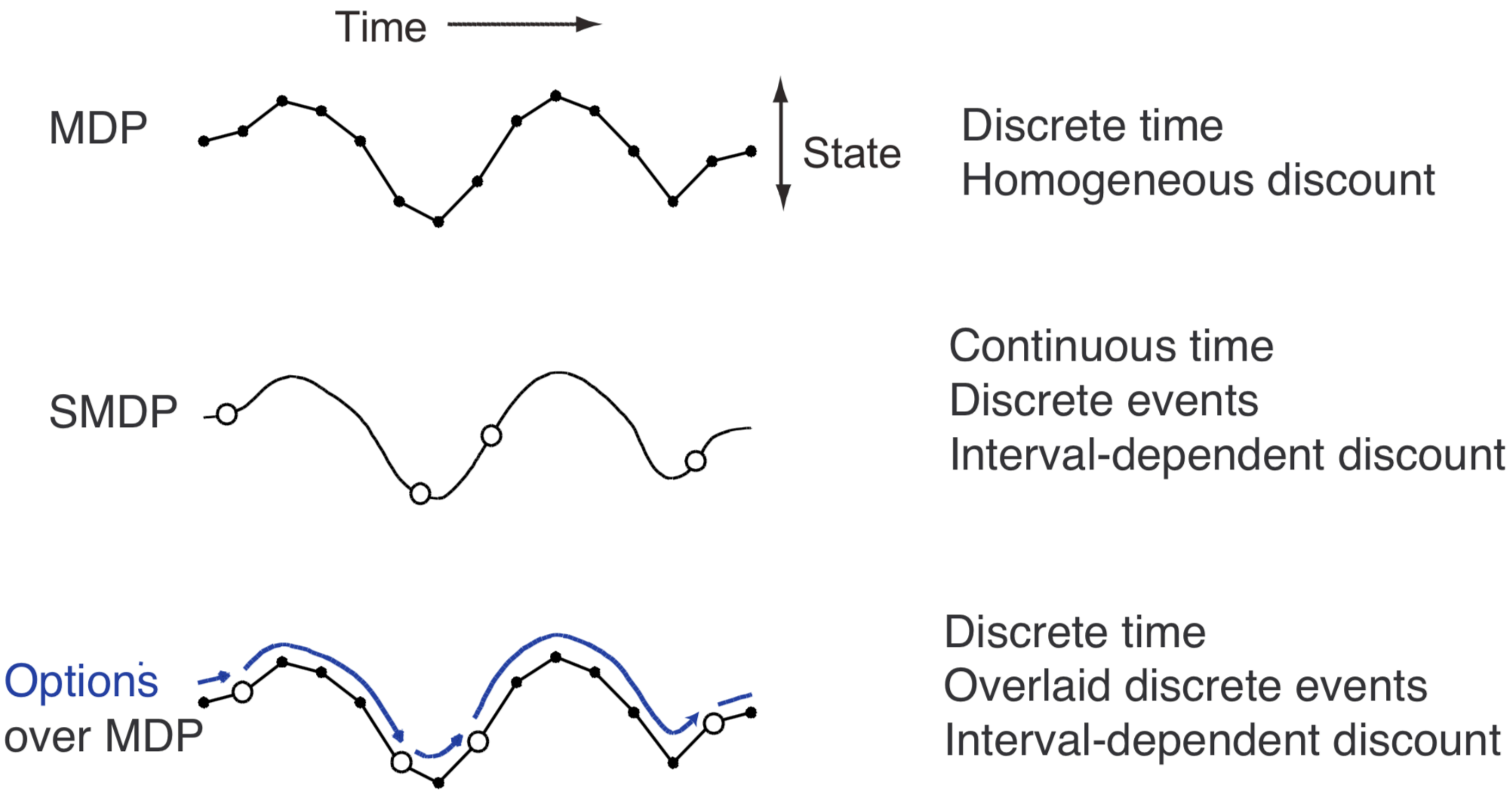
Background: MDP

- Policy: $\pi: S \times \mathcal{A} \to [0,1]$

- $V^{\pi}(s) = E\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s_t = s, \pi\}$
  $= \sum_{a \in \mathcal{A}_s} \pi(s,a)[r_s^a + \gamma \sum_{s'} p_{ss'}^a V^{\pi}(s')]$

- $V^*(s) = \max_{\pi} V^{\pi}(s) = \max_{a \in \mathcal{A}_s} [r_s^a + \gamma \sum_{s'} p_{ss'}^a V^*(s')]$

Background: Semi-MDP



Options

- Generalize actions to include temporally extended courses of actions.
- An option (I, π, β) has three components:
  - An initiation set $I \subseteq S$
  - A terminations condition $\beta: S \to [0,1]$
  - A policy π: $S \times \mathcal{A} \to [0,1]$
- If the option (I, π, β) is taken at $s \in I$, then actions are selected according to π until the option terminates stochastically according to β.

Options: Example

- Open-the-door
  - I: all states in which a closed door is within reach
  - π: pre-defined controller for reaching, grasping, and turning the door knob
  - β: terminate when the door is open

Option: more definitions and details

- Viewing simple actions as single-step options
- Composing options
- Policies over options: $\mu: S \times O \rightarrow [0,1]$
- *Theorem 1. (MDP+ options=SMDP). For any MDP, and any set of options defined on that MDP, the decision process that only selects among those options, executing each to the termination, is an SMDP.*

Option models

- Rewards:

$$R_s^o = E\{r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{k-1} r_{k+t} \big| \, O \text{ is initiated in state } s \text{ at time } t \text{ and last } k \text{ steps} \}$$

- Dynamics:

$$P_{ss'}^{O} = \sum_{k=1}^{\infty} \gamma^{\tau} p(s', k)$$

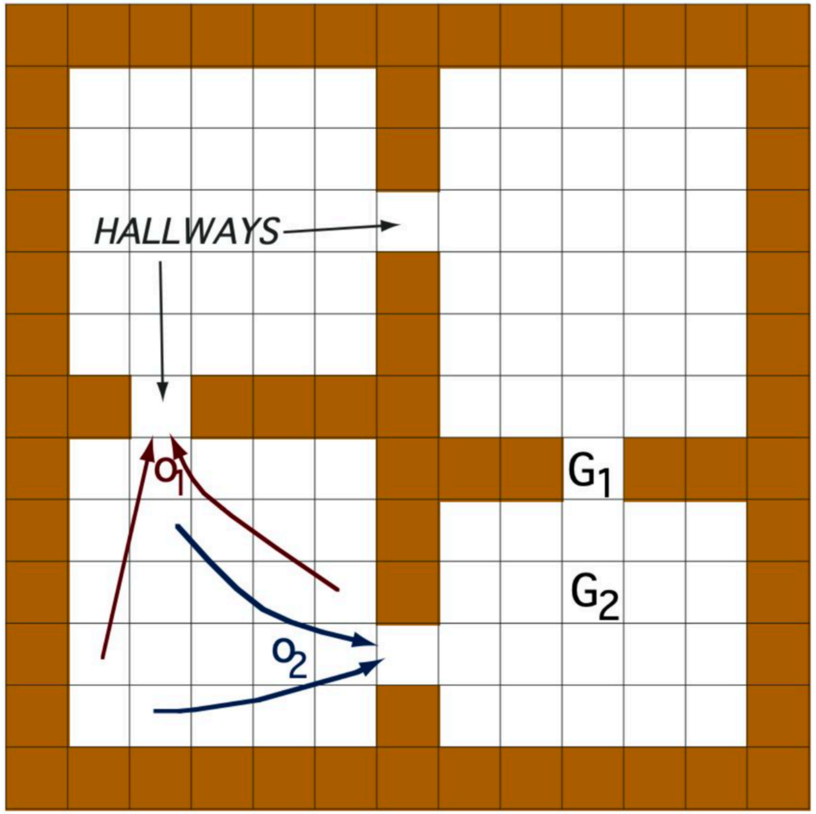
Rewriting Bellman Equations with Options

$$V^{\mu}(s) = E\{r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{k-1} r_{k+t} + \gamma^{k} V^{\mu}(s_{t+k}) \big| \varepsilon(\mu, s, t)\}$$

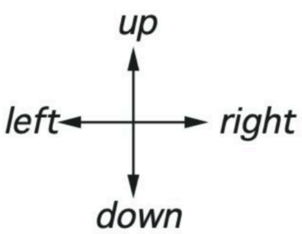
(k is the duration of the first option selected by μ)

$$= \sum_{o \in O_s} \mu(s, o)[r_s^o + \sum_{s'} p_{ss'}^{O} V^{\mu}(s')]$$

$$V^*(s) = \max_{o \in O_s} [r_s^o + \sum_{s'} p_{ss'}^{O} V^*(s')]$$



4 stochastic primitive actions

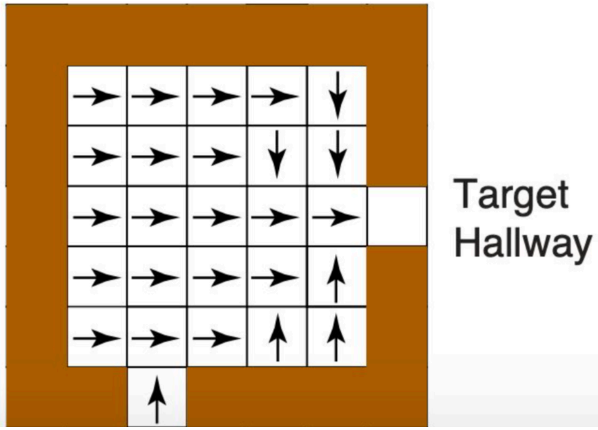


Fail 33% of the time

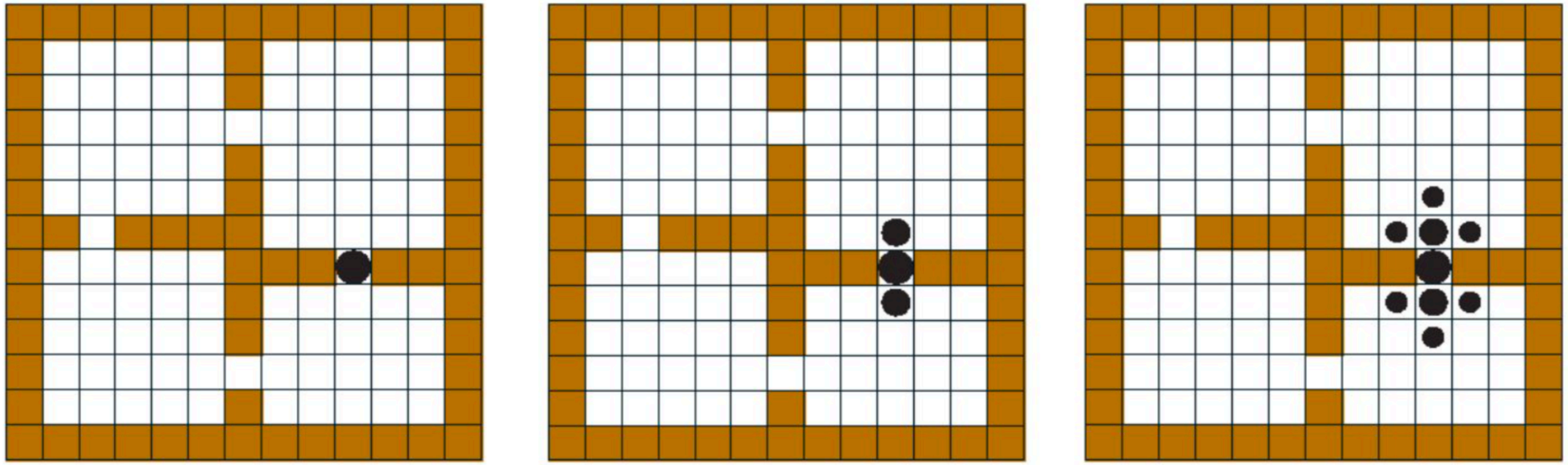
8 multi-step options

(to each room's 2 hallways)

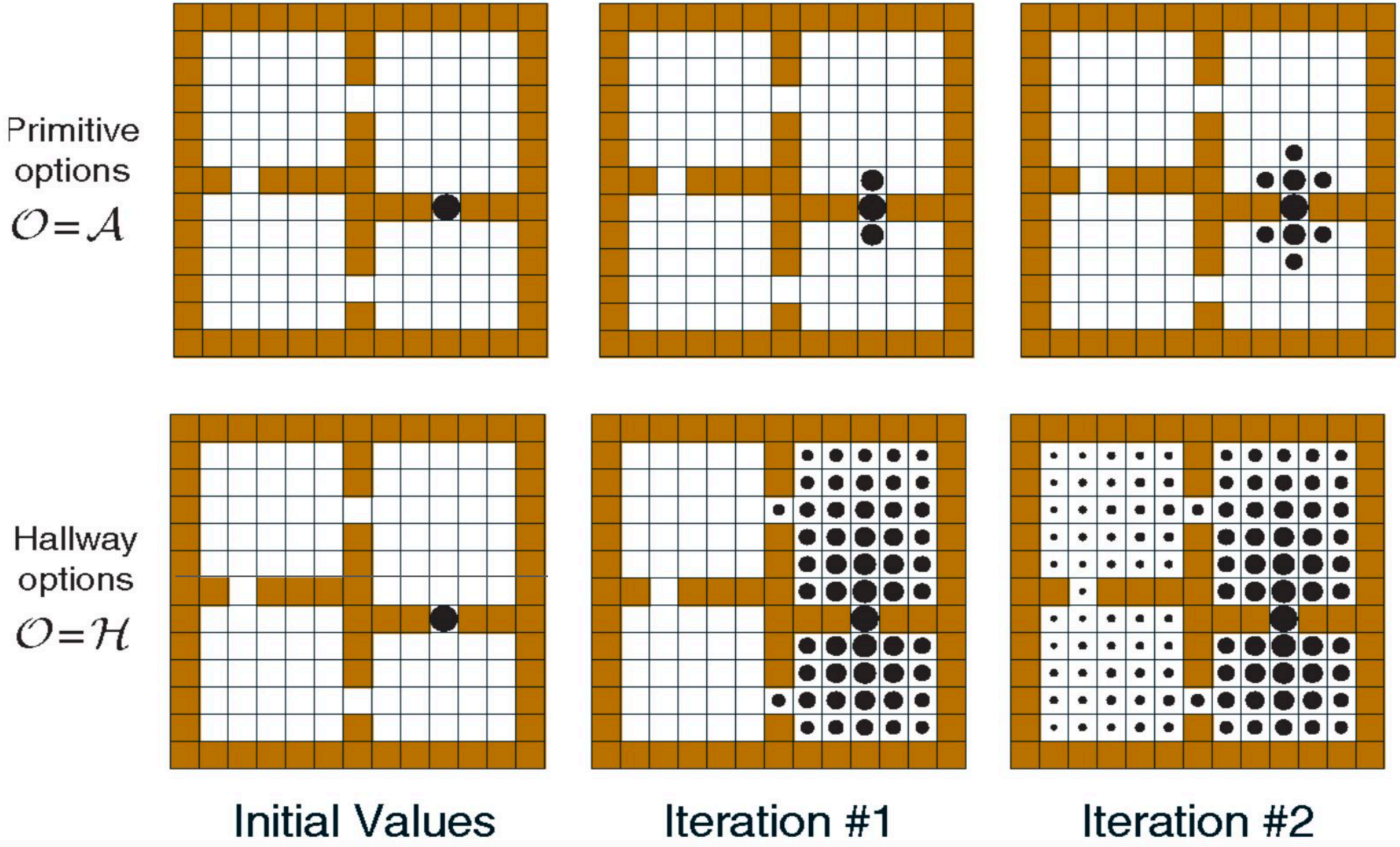
Example of one option's policy:



Primitive options
$\mathcal{O} = \mathcal{A}$

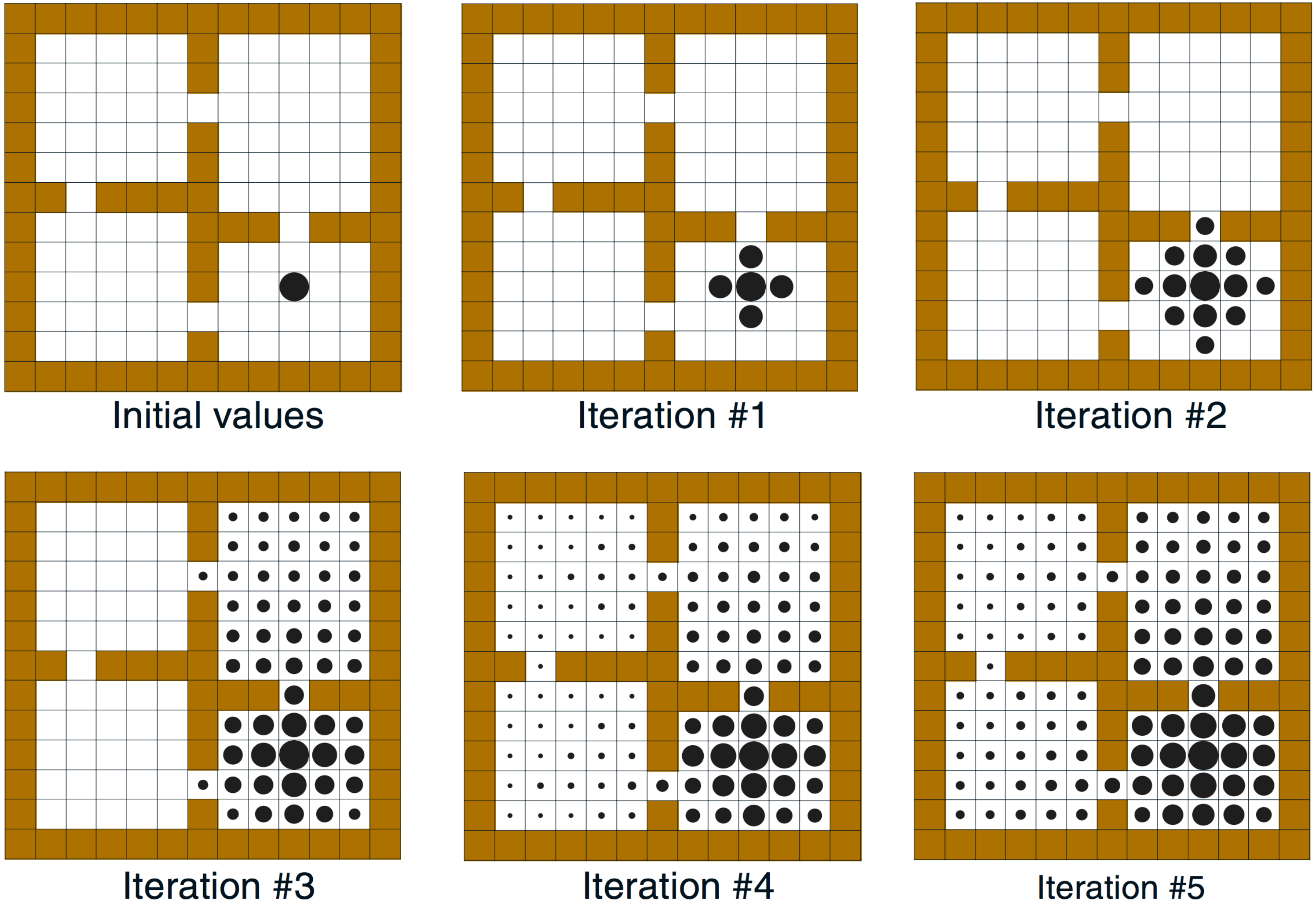


Initial Values Iteration #1 Iteration #2


Primitive options
$\mathcal{O}=\mathcal{A}$
Hallway options
$\mathcal{O}=\mathcal{H}$
Initial Values
Iteration #1
Iteration #2

Primitive and hallway options

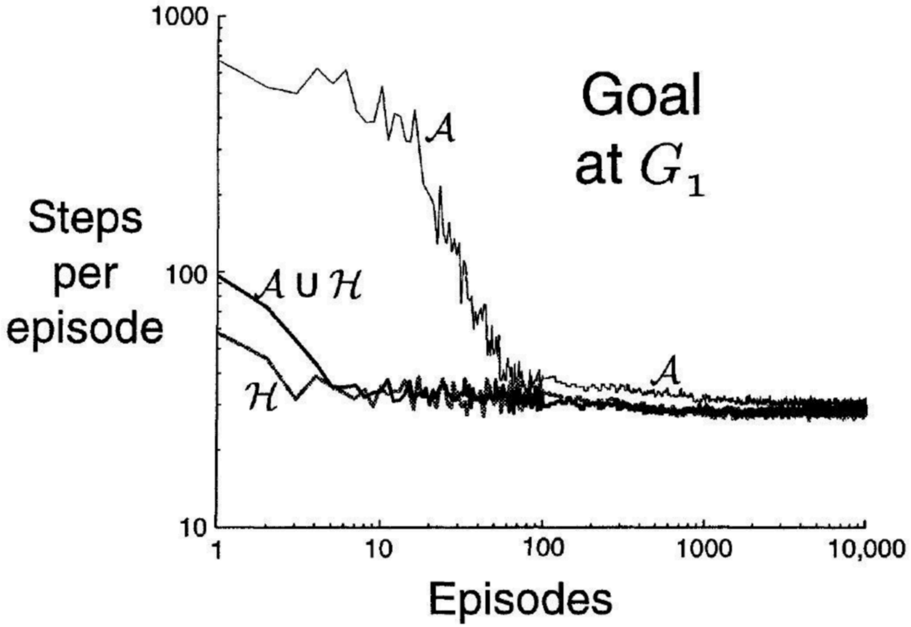
$\mathcal{O}=\mathcal{A}\cup\mathcal{H}$



Options value learning

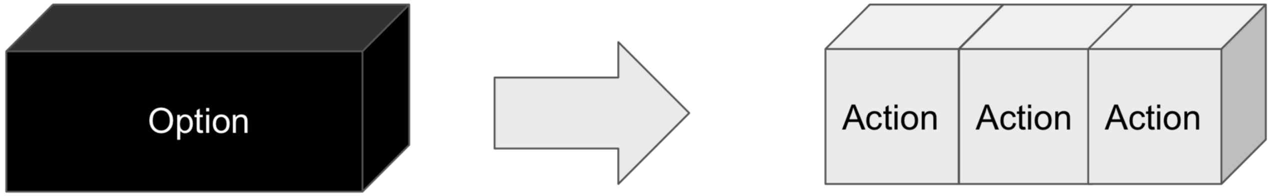
- State s, initiate option o, execute until termination
- Observe termination state s', number of steps k, discounted reward r

$$Q(s,o) = Q(s,o) + \alpha\left(r + \gamma^k \max_{o' \in O_{s'}} Q(s',o') - Q(s,o)\right)$$



Between MDPs and semi-MDPs

Open up the black-box when
Option is Markov!

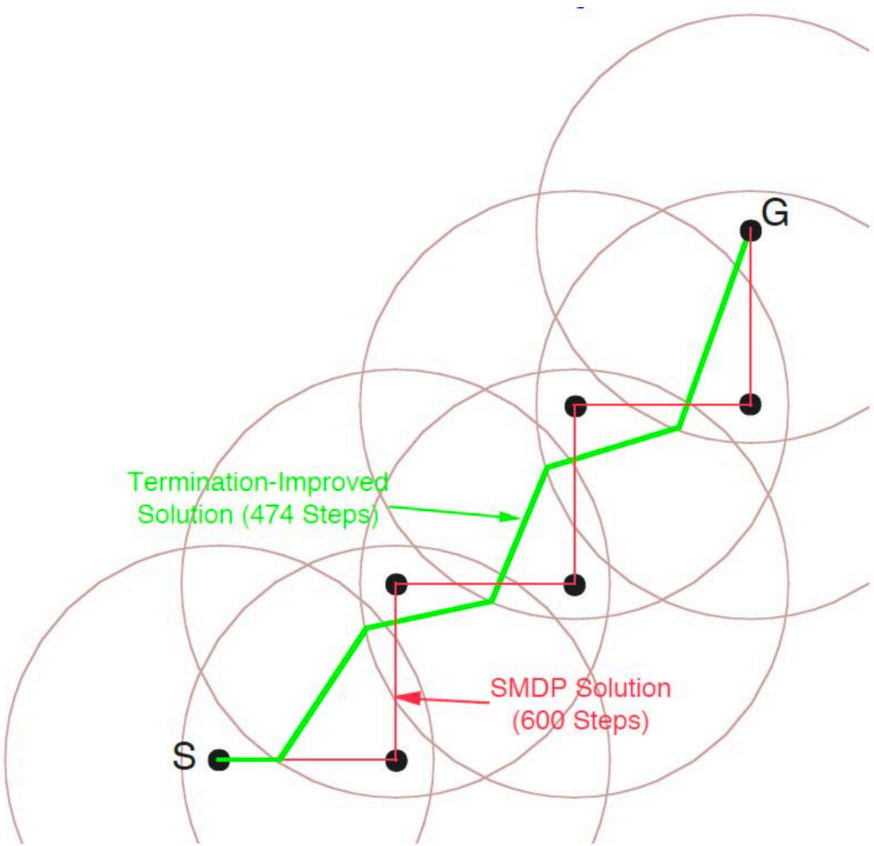


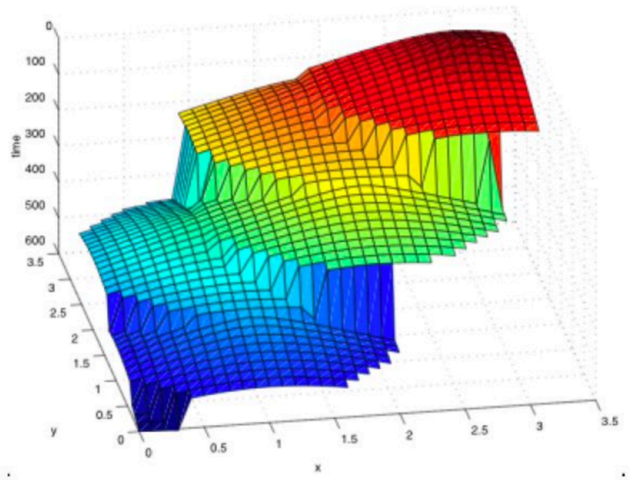
1. Interrupting options
2. Intra-option model/ value learning
3. Sub goals

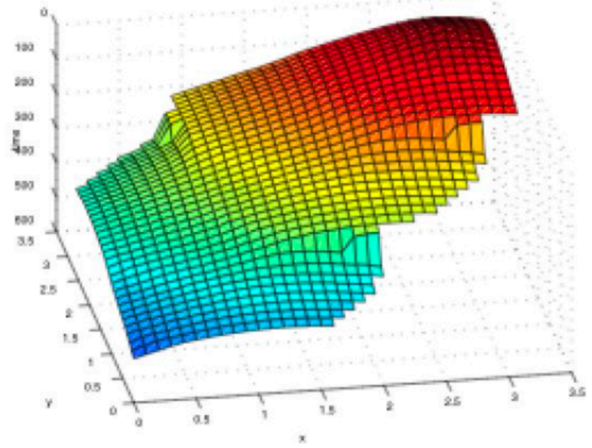
1.Interrupting options

- We don’t have to follow options until termination, we can re-evaluate our commitment at each step.
- If the value of continuing option o, $Q(s, o)$is less than the value of selecting a new option $V^{\mu}(s) = \sum_q \mu(s, q)Q^{\mu}(s, q)$, then switch.
- Theorem 2. policy μ' is the interrupted policy of μ. Then:
  I. For all s ∈ S: $V^{\mu'}(s) \geq V^{\mu}(s)$
  II. If from state $s \in S$, there is a non zero probability of encountering an interrupted history, then $V^{\mu'}(s) > V^{\mu}(s)$

Interrupting options: Example







2.Intra-option algorithms

- Learning about one option at a time is very inefficient.
- Instead, learn all options consistent with the behavior.
- Update every Markov option o whose policy could have selected a_t according to the same distribution $\pi(s_t, .)$.

$$Q(s_t, o) \leftarrow Q(s_t, o) + \alpha\left[\left(r_{t+1} + \gamma U(s_{t+1}, o)\right) - Q(s_t, o)\right]$$

- Where

$$U(s, o) = \left(1 - \beta(s)\right)Q(s, o) + \beta(s) \max_{o' \in O} Q(s, o')$$

Is an estimate of the value of state-option pair (s,o) upon arrival in state s.

2.Intra-option algorithms

- **Theorem 3** (Convergence of intra-option Q-learning). *For any set of Markov options*, O, *with deterministic policies, one-step intra-option Q-learning converges with probability* 1 *to the optimal Q-values, for every option regardless of what options are executed during learning, provided that every action gets executed in every state infinitely often.*

- Proof.

$$Q(s,o) \leftarrow Q(s,o) + \alpha\big[\big(r' + \gamma U(s',o)\big) - Q(s,o)\big]$$

We prove that the operator $E[r' + \gamma U(s',o)]$ is a contraction.

$$|E[r' + \gamma U(s',o)] - Q^*(s,o)| = |r_s^a + \gamma \sum_{s'} p_{ss'}^a\, U\,(s',o) - Q^*(s,o)|$$

$$= \left| r_s^a + \gamma \sum_{s'} p_{ss'}^a\, U\,(s',o) - r_s^a + \gamma \sum_{s'} p_{ss'}^a\, U^*(s',o) \right| \leq$$

$$|\sum_{s'} p_{ss'}^a [\big(1 - \beta(s')\big)\big(Q(s',o) - Q^*(s',o)\big) + \beta(s')(\max_{o'} Q(s',o') - \max_{o'} Q^*(s',o'))]\,| \leq$$

$$\sum_{s'} p_{ss'}^a \max_{s'',o''} |Q(s'',o'') - Q^*(s'',o'')| =$$

$$\gamma \max_{s'',o''} |Q(s'',o'') - Q^*(s'',o'')|$$

Theorem 1 *A random iterative process* $\Delta_{n+1}(x) = (1-\alpha_n(x))\Delta_n(x)+\beta_n(x)F_n(x)$ *converges to zero w.p.1 under the following assumptions:*

1) *The state space is finite.*

2) $\sum_n \alpha_n(x) = \infty$, $\sum_n \alpha_n^2(x) < \infty$, $\sum_n \beta_n(x) = \infty$, $\sum_n \beta_n^2(x) < \infty$, *and* $\mathrm{E}\{\beta_n(x)|P_n\} \leq \mathrm{E}\{\alpha_n(x)|P_n\}$ *uniformly w.p.1.*

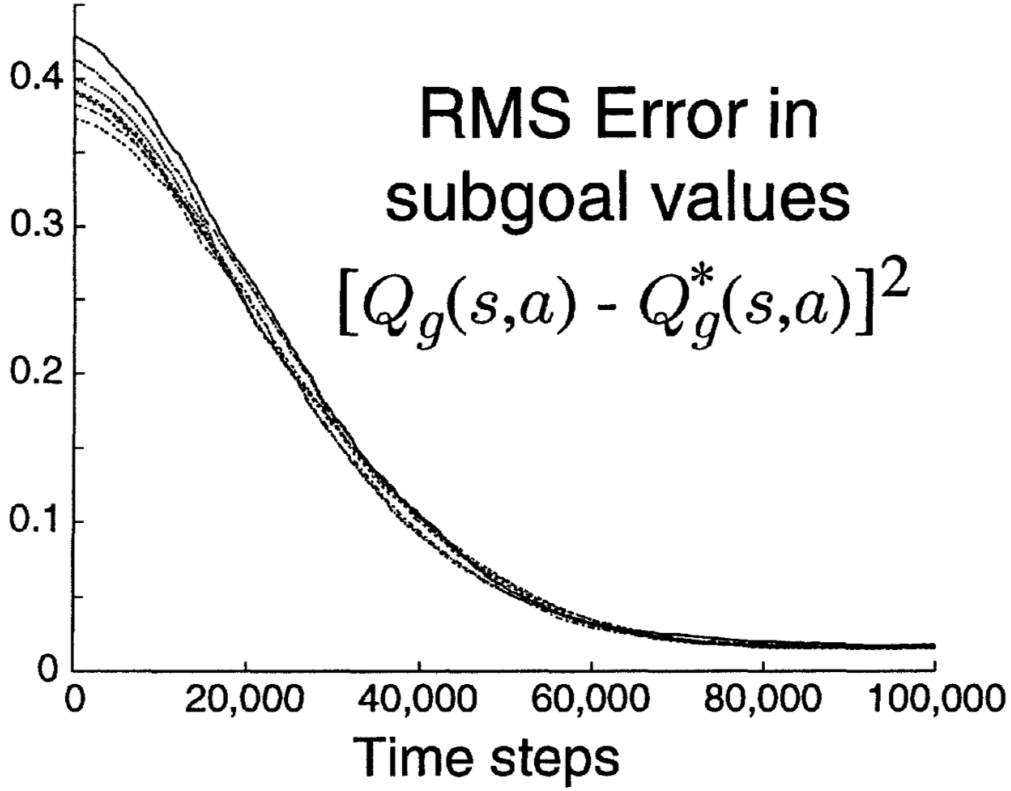
3) $\| \mathrm{E}\{F_n(x)|P_n\} \|_W \leq \gamma \| \Delta_n \|_W$, *where* $\gamma \in (0,1)$.

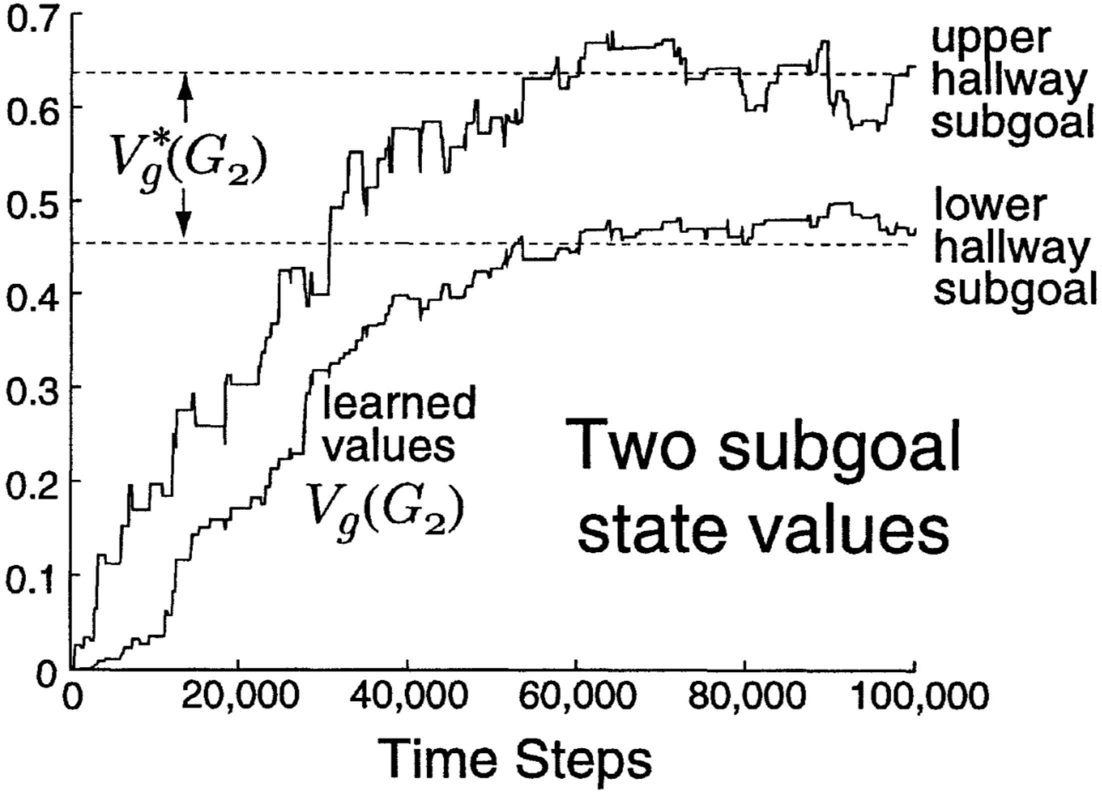
4) $\mathrm{Var}\{F_n(x)|P_n\} \leq C(1+\| \Delta_n \|_W)^2$, *where* C *is some constant.*

3.Subgoals for learning options

- It is natural to think of options as achieving subgoals of some kind, and to adapt each option’s policy to better achieve its subgoal.
- A simple way to formulate a subgoal for an option is to assign a terminal subgoal value, g(s), to each state.
- For example, to learn a hallway option in the rooms task, the target hallway might be assigned a subgoal value of +1, while other get the subgoal value of zero.
- Learn policies using subgoals independently using an off-policy learning method such as Q-learning .

3.Subgoals for learning options





Contributions (Recap)

- Problem: enable temporally abstract knowledge and action to be included in the reinforcement learning
- Introduced options, temporally extended courses of actions.
- Extended theory of SMDPs to the context of options.
- Introduction of intra-option learning algorithm that are able to learn about options from fragment of execution.
- Propose notion of subgoals that can be used to improve option themselves.

Limitations

- Require to formalize subgoals/options.
- Might necessitate a small state-action space.
- The integration with state abstraction remain incompletely understood.

Questions

1. Why should we use off-policy learning methods for learning the option policies using subgoals?
2. What cases can you think of which intra value learning improve upon the original option value learning?
3. Is planning over options always going to speed up the planning?