

Advanced Topics in Reinforcement Learning

Lecture 7: Models and Planning I

Josiah Hanna

University of Wisconsin — Madison

Announcements

- Project proposal due Thursday at 11:59 PM.
- Homework due October 8 at 11:59 PM
- Read chapter 9 and 11 for next week. Function approximation!
- Upcoming dates:
 - Literature survey due: October 22
 - Midterm exam: October 29

Learning Outcomes

After this week, you will be able to:

1. Explain the difference between background and decision-time planning in RL.
2. Implement Dyna, MCTS, and other model-based RL algorithms.
3. Understand how models can be integrated into RL agents.

But first, finish TD learning

Review

- Dynamic Programming Methods
 - Model is given.
- Monte Carlo methods.
 - Model-free but no bootstrapping.
- Temporal Difference Learning.
 - Model-free and bootstrapping.

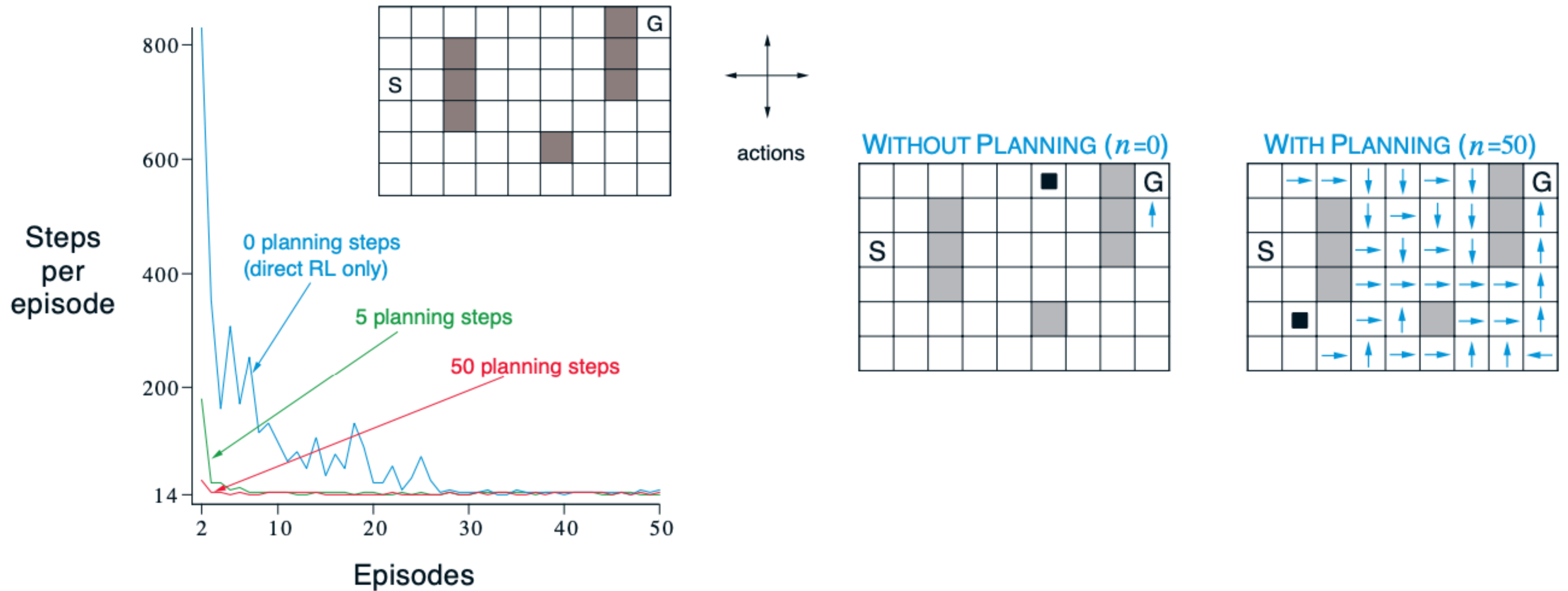
Plan-Space Search

- The RL methods we will learn about in this class all follow the generalized policy iteration scheme.
 - $\pi_k \rightarrow q_k \rightarrow \pi_{k+1} \rightarrow q_{k+1} \rightarrow \dots$
- Alternatively: search directly for a good policy without computing a value function.
 - Genetic algorithms, evolutionary strategies, random search, optimization, etc.
 - Define $f : \pi \rightarrow \mathbb{R}$ and then find π with maximal $f(\pi)$.
- (+) Robust to violations of MDP formalism.
- (+) Can be applied to almost any type of policy.
- (-) Size of policy space is exponential in the number of states and actions.
- (-) If interaction time is long and γ large, then learning methods may be preferred to static policies.

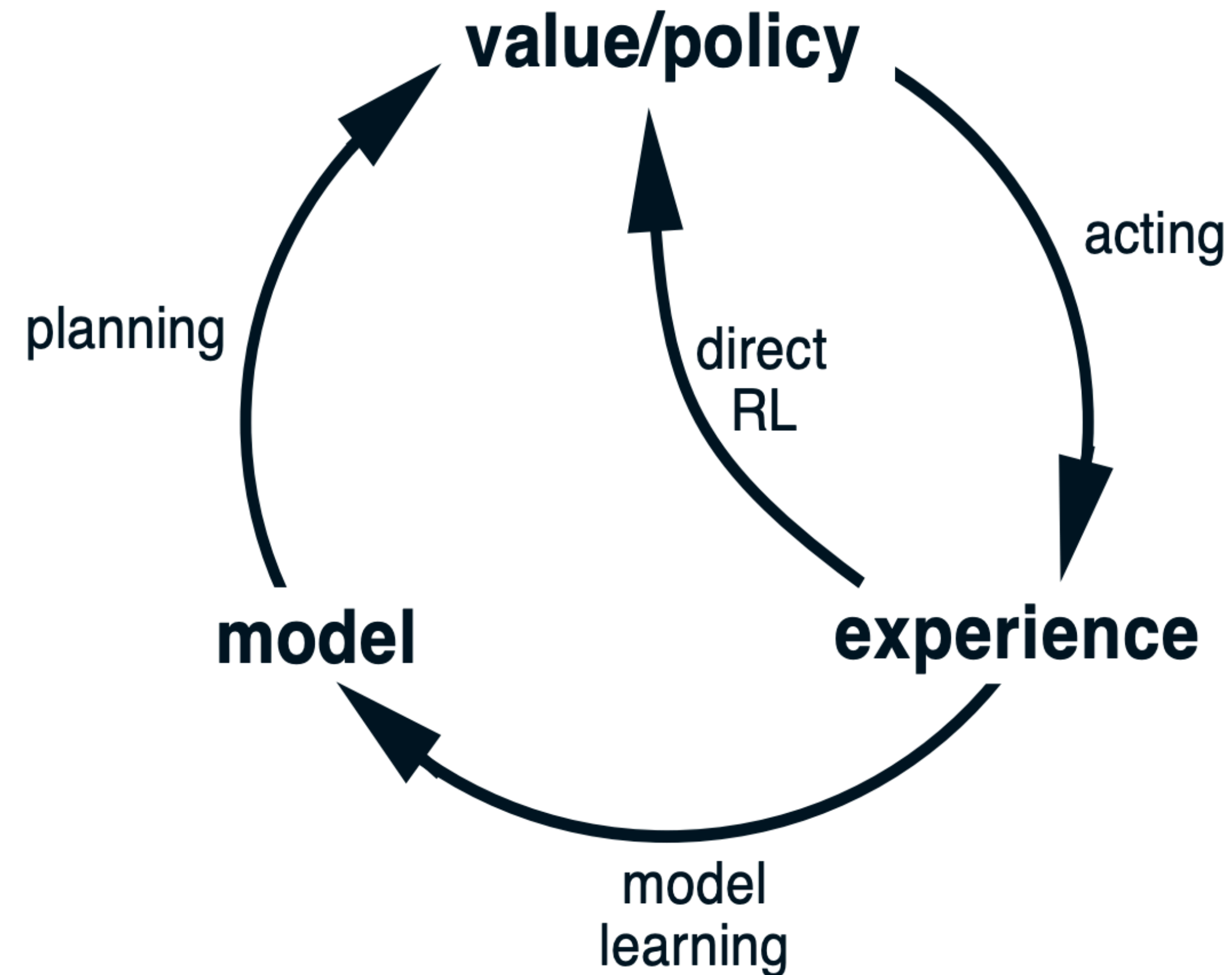
RL with a Learned Model

- Model-based learning and planning.
 - Use experience to model p . Then use planning methods to back-up values.
 - How to model p is a challenging question in practice!
 - Often more data-efficient (better policy with less interaction). Why?
 - Less computationally-efficient per time-step. Why?
- Access to a model provides much flexibility in how we back-up values.

Model-based Data Efficiency



Dyna-style Model-based Learning



Dyna Agents

- In state S , take action A , observe R , and S' .
- Update model: $\text{Model}(S, A) \leftarrow R, S'$
- Repeat n times:
 - Sample random state-action pair, S, A .
 - $R, S' \leftarrow \text{Model}(S, A)$.
 - Apply q-learning update with S, A, R, S' .

Real Experience

Synthetic / Simulated
Experience

Abhishek's Presentation

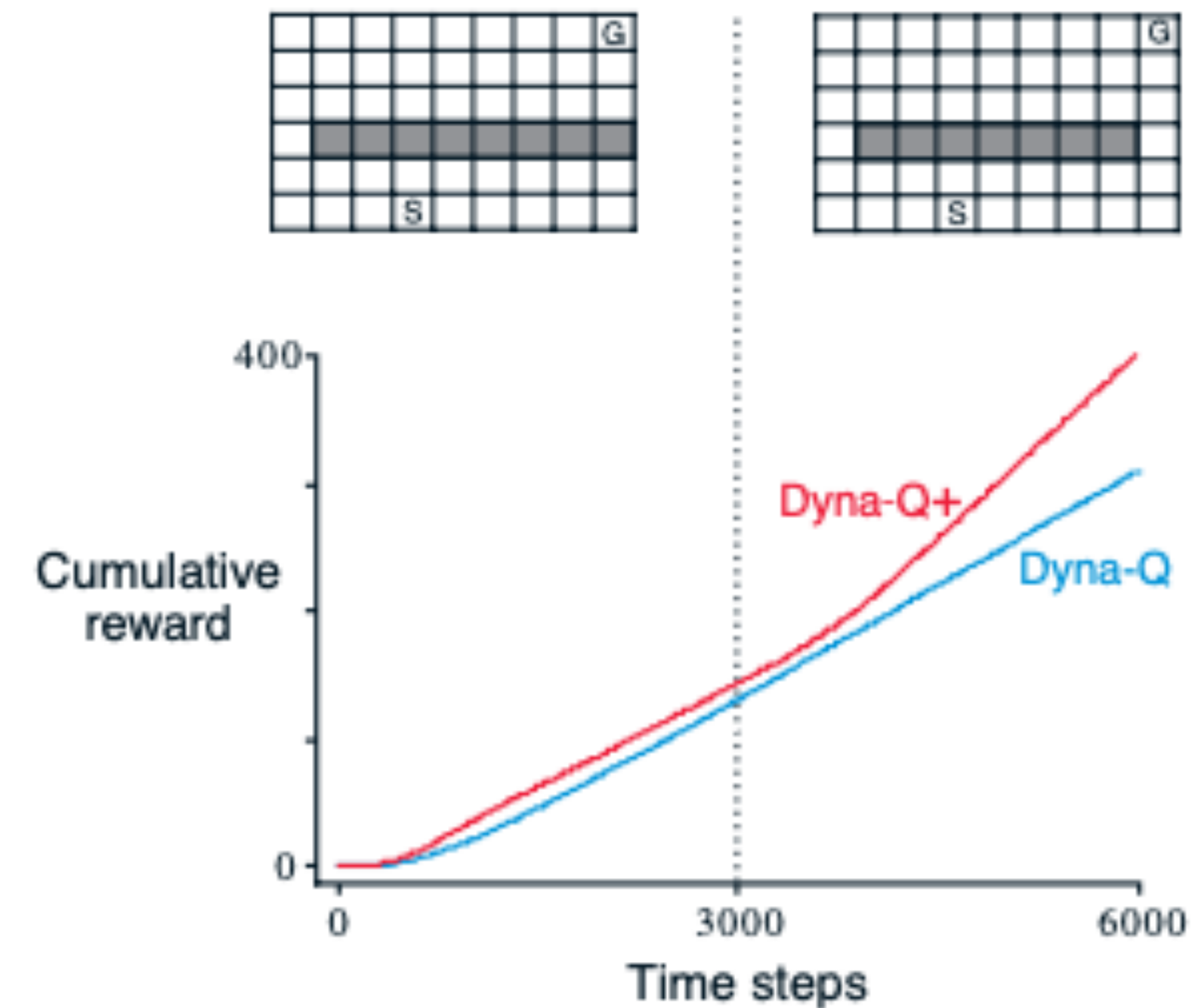
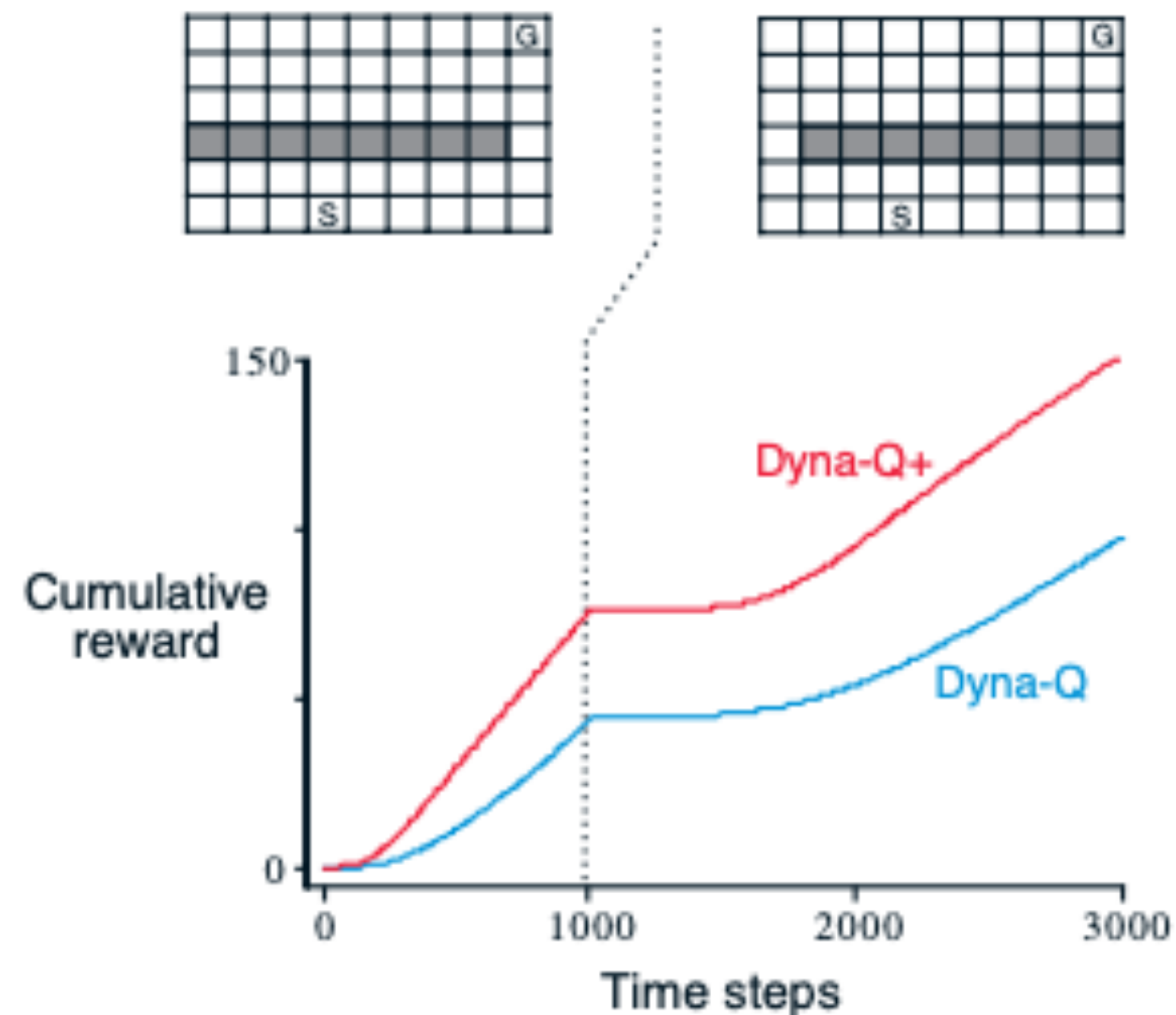
Mastering Diverse Domains through World Models

Hafner et al. 2023

Slides

When the Model is Wrong

- “All models are wrong but some are useful.” - George Box.
- In practice, models can be inaccurate for many reasons.
 - Partial observability, non-stationarity, function approximation, missing data, stochastic environment, etc.



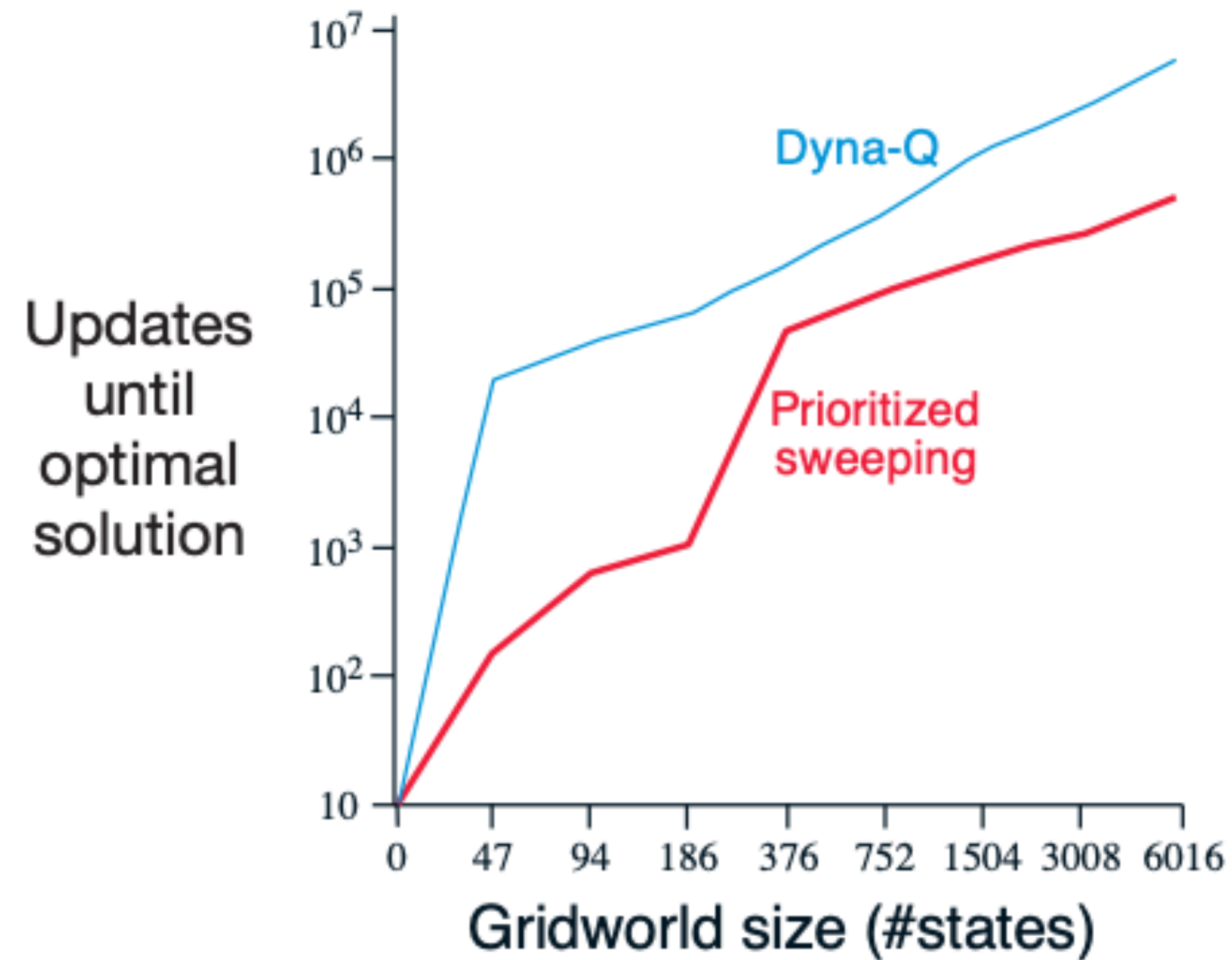
Dyna-Q+ replaces reward following s, a with $r + \kappa\sqrt{\tau}$ where τ is time since s, a tried last

Prioritizing Updates

- Dyna-Q updates a random sub-set of states.
- What is one case where this is inefficient?
 - When many states won't have a value change.
- Prioritized sweeping: keep a priority queue of (S,A) pairs that are likely to have a large value change.
- Can base this off of expected change to predecessor state, $\bar{S}$, i.e., priority is:

$$| \bar{R} + \gamma \max_a Q(S, a) - Q(\bar{S}, \bar{A}) |$$

Prioritizing Updates



To Sample or Not?

- Dynamic programming typically uses *expected updates*.

$$Q(s, a) \leftarrow \sum_{s', r} \hat{p}(s', r | s, a) [r + \gamma \max_{a'} Q(s', a')]$$

- TD-method methods use *sample updates*.

$$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

- Can use either with simulated experience. How and which to choose?

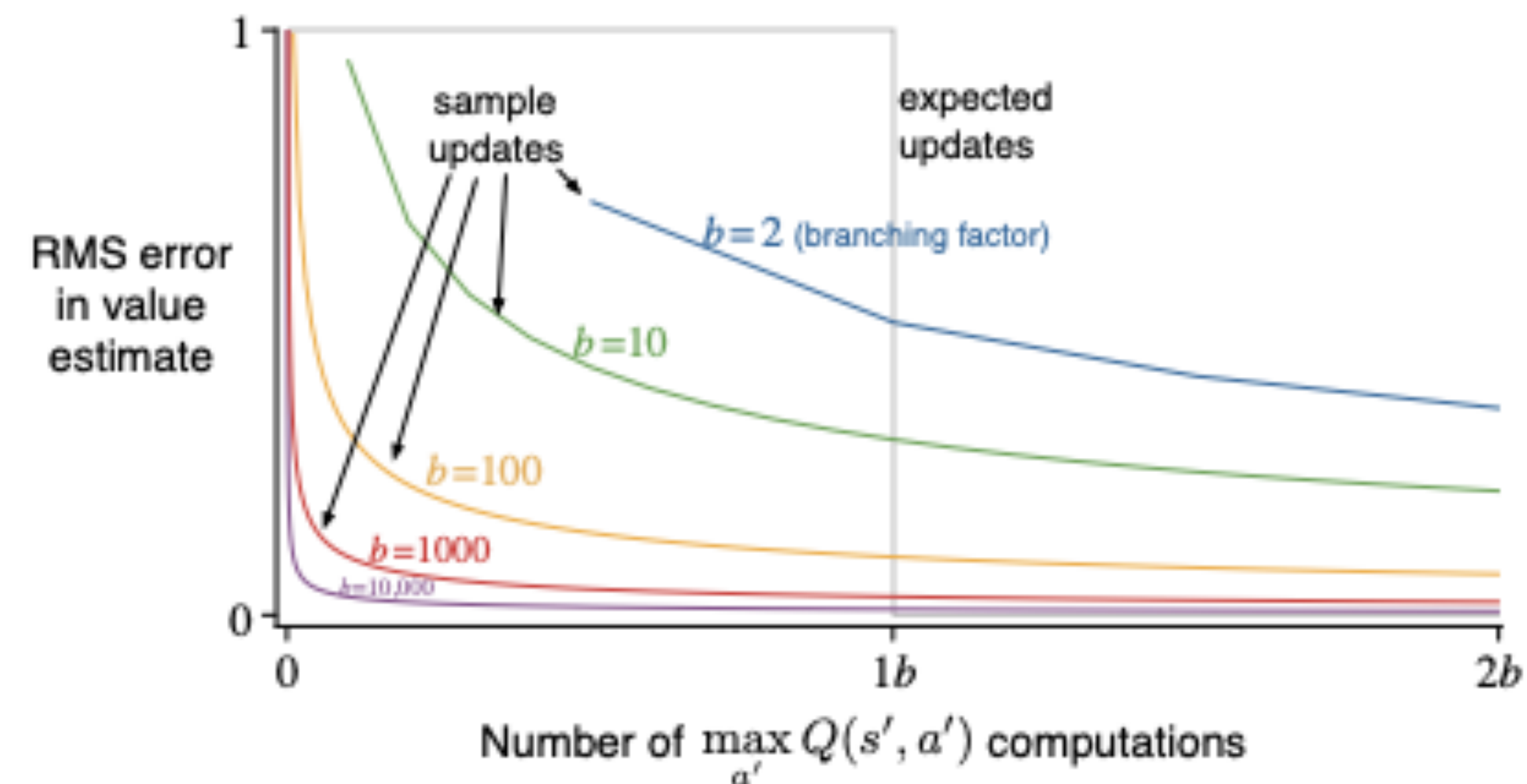


Figure 8.7: Comparison of efficiency of expected and sample updates.

Sample vs Expected Updates

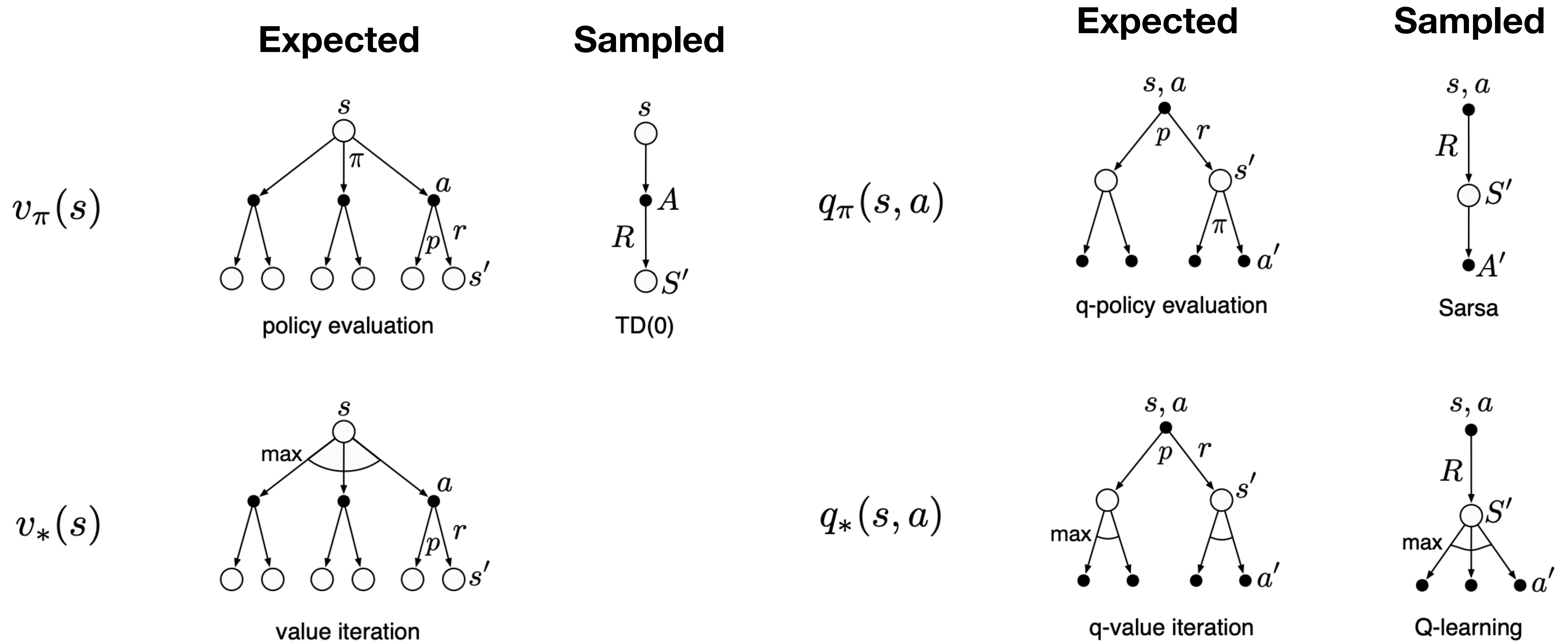


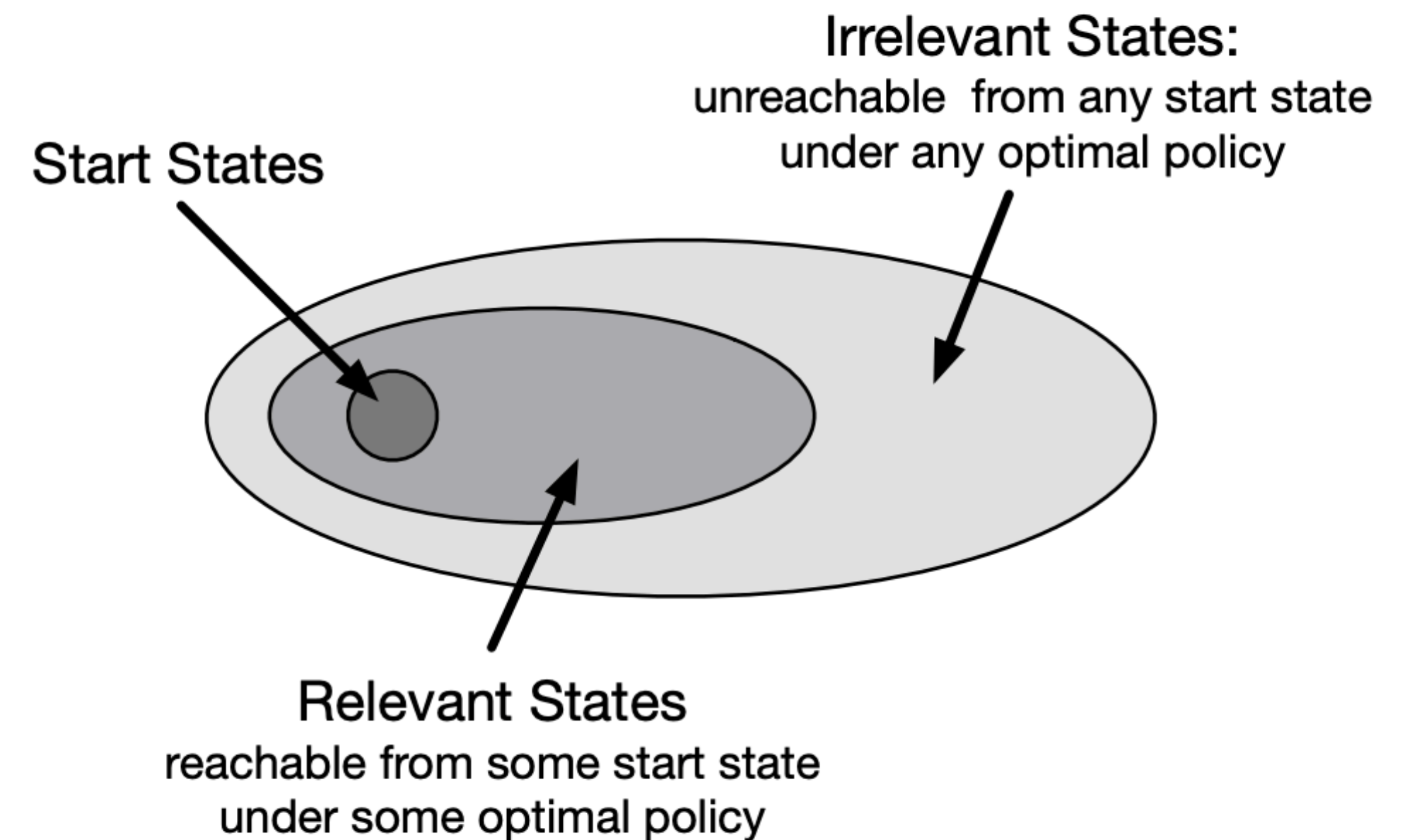
Figure 8.6 from textbook

Trajectory Sampling

- Uniform sampling of states can be inefficient.
- It may be more effective to focus value back-ups on states that the agent will visit often.
- How to know what states the agent will visit?
 - Simulate entire trajectories within the model. Back-up these states.
 - Initialize the agent in a start state and follow the current policy from there.

Real-time Dynamic Programming

- Key Idea: perform a value-iteration update on each state as it is visited.
- For n episodes of real experience:
 - Start in initial state, S_0 .
 - Repeat $A_t \sim \pi(A = a | S_t)$,
 $S', R \sim \text{Model}(S_t, A_t)$ where π is ϵ -greedy.
 - At each step, t, apply the value iteration update to $Q(S_t, A_t)$.



Summary

- Building a model from experience can improve the efficiency of RL.
- Models can be used for:
 - Planning, i.e., dynamic programming updates.
 - Learning with synthetic experience.
- When coupling planning and interaction, we need to make efficient use of limited computational resources.

Action Items

- Complete project proposal.
- Complete homework.
- Begin literature review.
- Begin reading Chapter 9 and 11.