

# Advanced Topics in Reinforcement Learning

## Lecture 8: Models and Planning II

Josiah Hanna

University of Wisconsin — Madison

# Announcements

- Project proposals due October 2 at 11:59pm
- Homework released. Due: October 8 at 11:59PM
- Read chapter 9 and 11 for next week. Function approximation!
- Upcoming dates:
  - Literature survey due: October 22
  - 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.

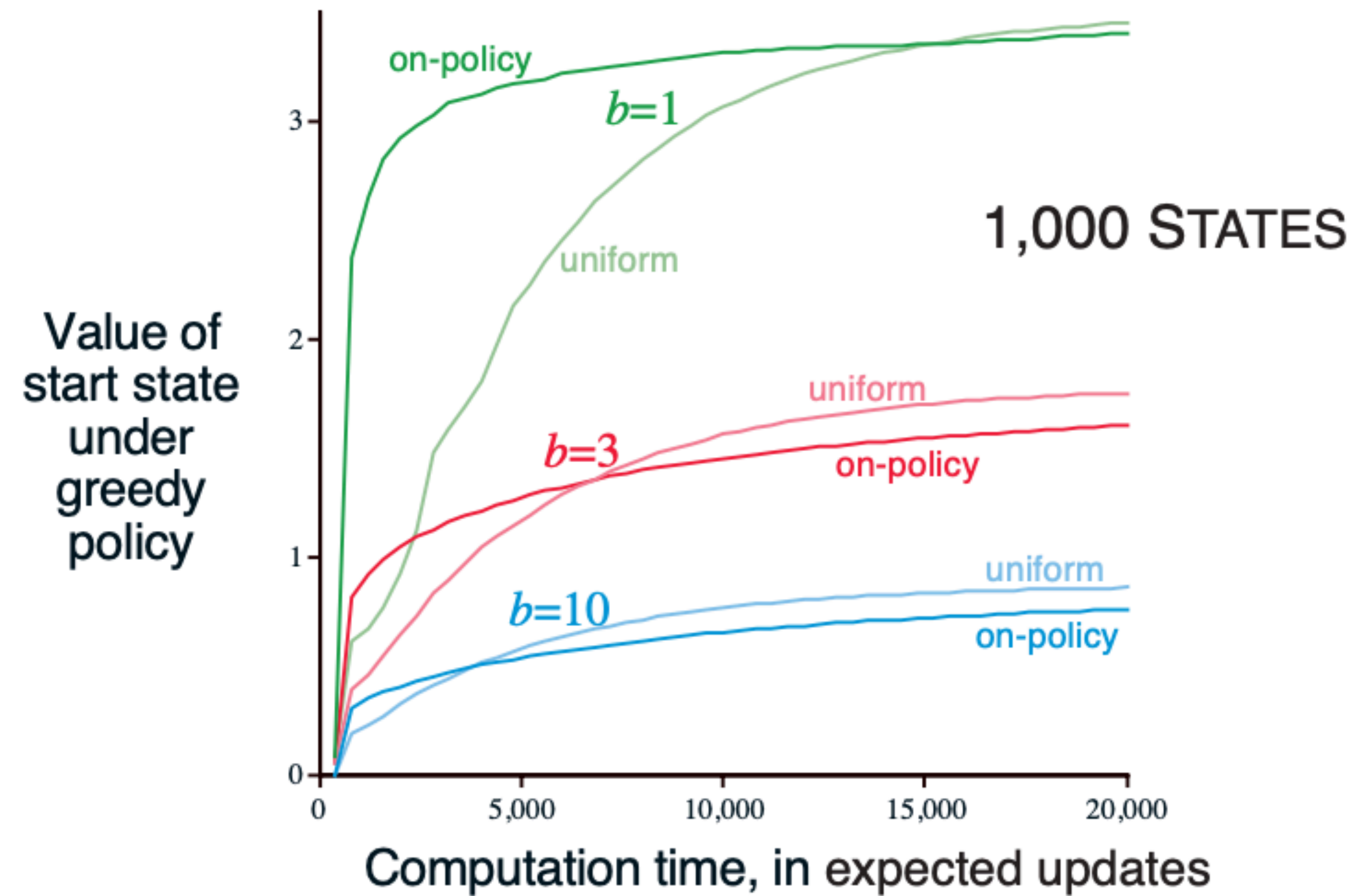
# Today

- Finish RTDP
- Planning at decision-time:
  - Heuristic Search
  - Roll-out Algorithms
  - MCTS

# 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?
  - Initialize the agent in a start state and follow the current policy from there.
  - Simulate entire trajectories within the model or real world. Back-up the values for these states.

# Trajectory Sampling

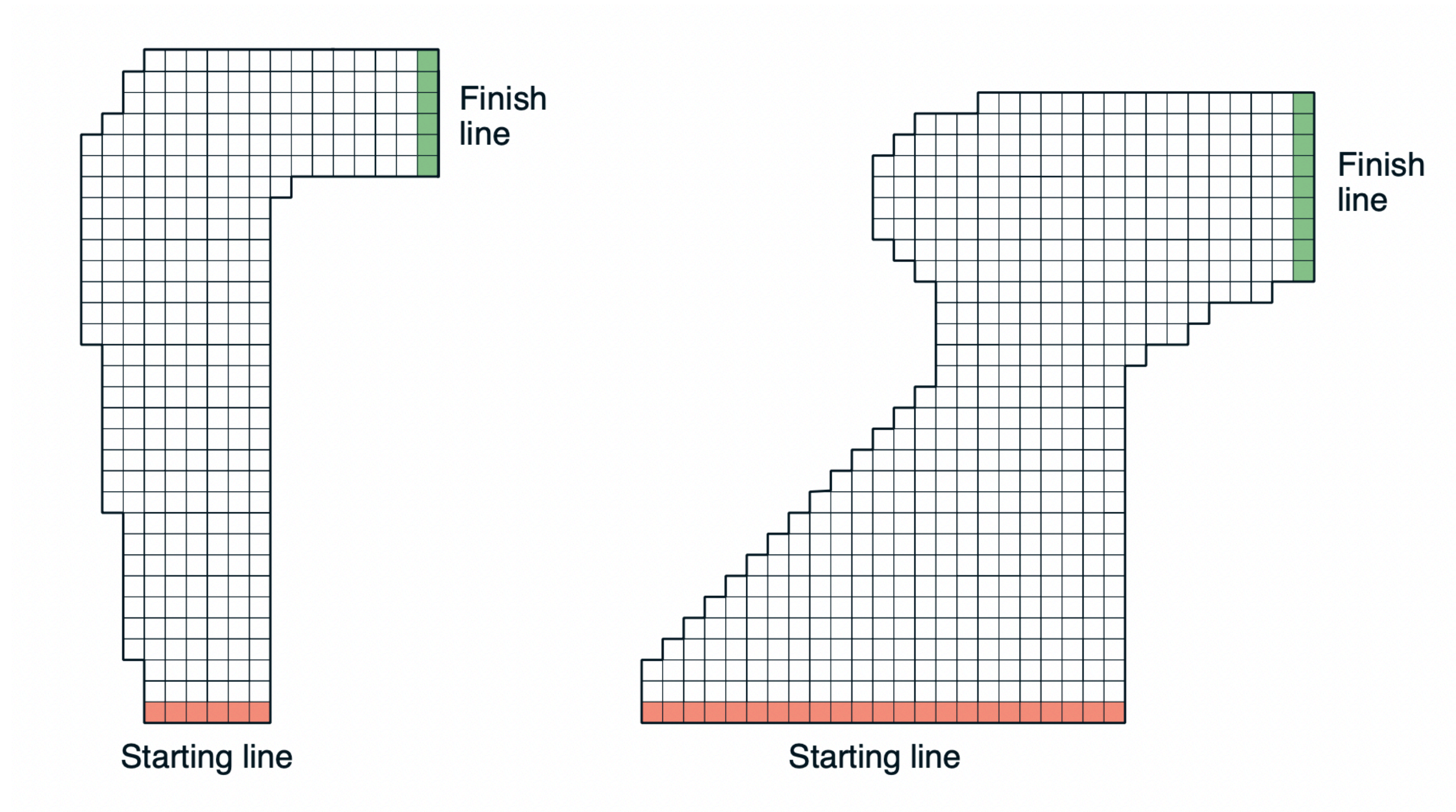


# Real-time Dynamic Programming

- Key Idea: perform a value-iteration update on each state as it is visited.
- For  $n$  real episodes:
  - Start in initial state,  $S_0$ . Follow greedy policy until termination.
  - For each real step, run  $k$  simulations:
    - Repeat  $A_t \sim \pi(A = a \mid S_t)$ ,  $S', R \sim \text{Model}(S_t, A_t)$  where  $\pi$  is  $\epsilon$ -greedy.
    - At each simulated step,  $t$ , apply the value iteration update to  $Q(S_t, A_t)$ :

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

# RTDP Example



# Planning at Decision Time

- So far we considered using planning to improve the value-function and speed-up policy iteration.
- Now we consider using planning to immediately compute an action for a given state.

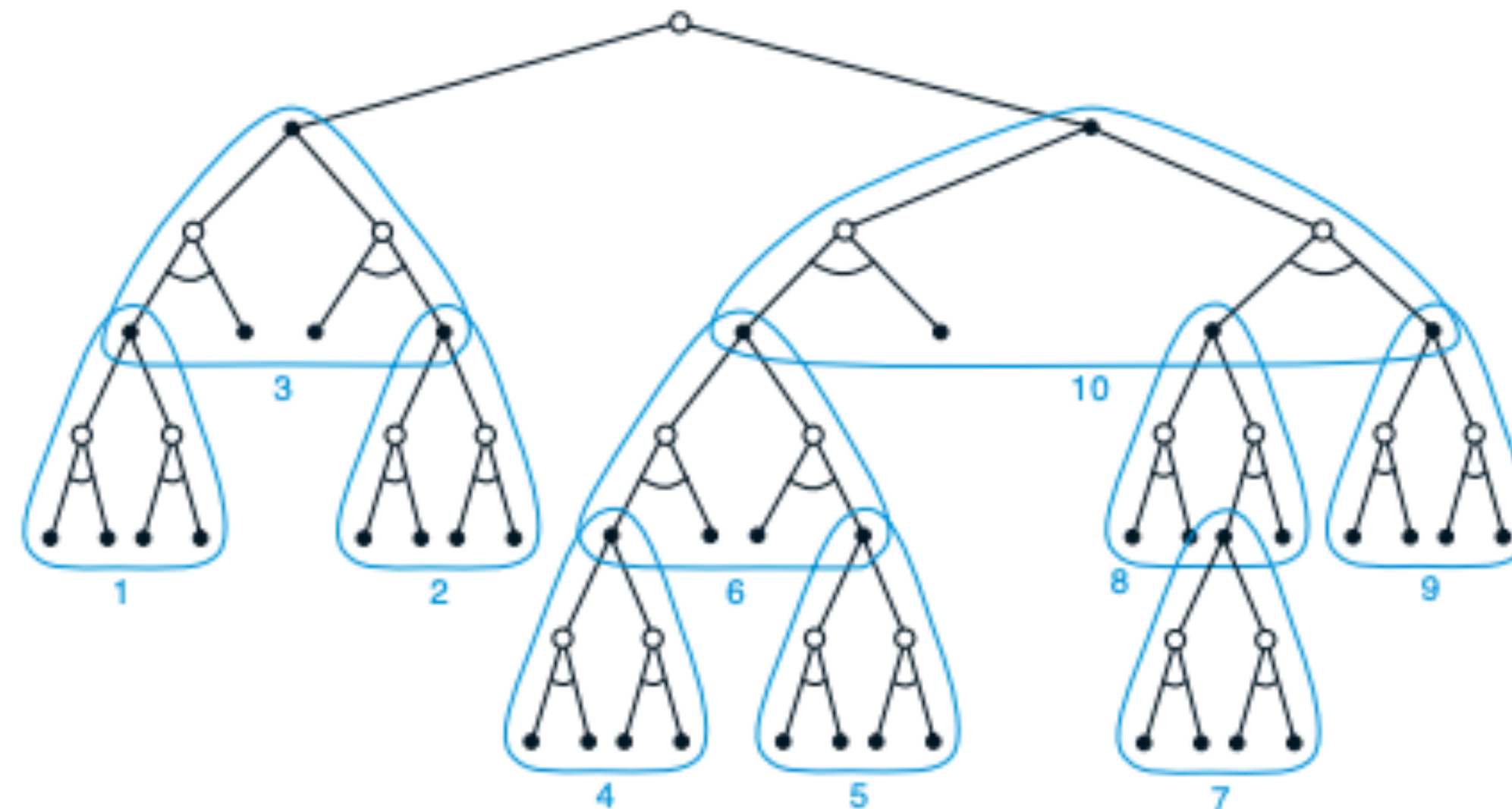
- $$\pi(s) \leftarrow \arg \max_a \sum_{s', r} p(s', r | s, a) [r + \gamma v_\pi(s')]$$

- $$\pi(s) \leftarrow \arg \max_a \sum_{s', r} p(s', r | s, a) [r + \gamma \max_{a'} \sum_{s'', r'} p(s'', r' | s'', a') [r' + \gamma v_\pi(s'')]]$$

- Slow deliberation before making a decision (System 2).
- Contrasts with immediate decision-making of model-free methods (System 1).

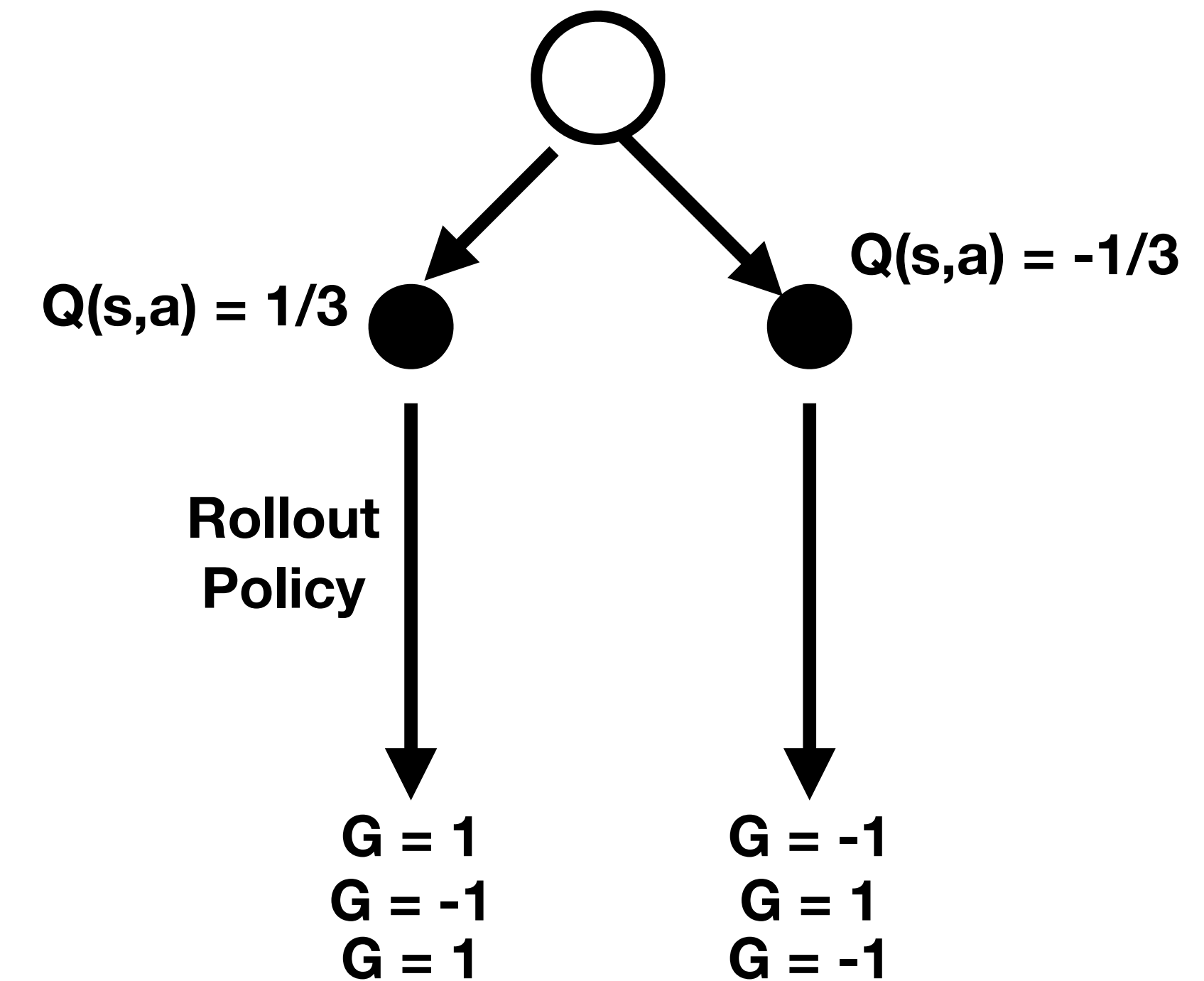
# Heuristic Search

- Motivation: model is perfect and action-value function is imperfect.
- Focus memory and computation on immediate relevant state and next decision.
- Deeper search generally leads to a better action choice at expense of more computation.

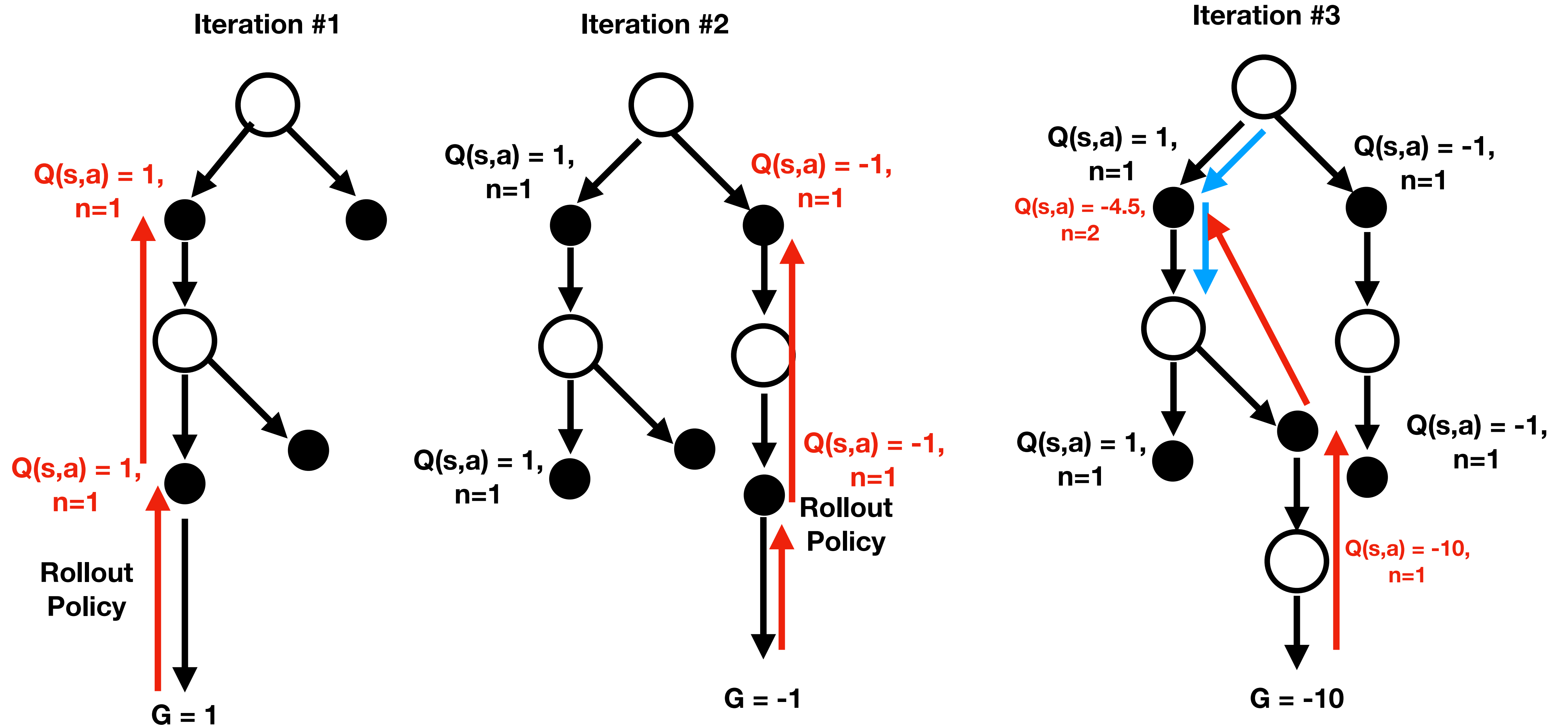


# Roll-out Algorithms

- **Rollout:** following a policy until termination, i.e, rolling out the policy.
- Monte Carlo learning at decision-time; improve upon the roll-out policy.
- Rolling out the policy only requires a sample model.
- Computation time is a limiting factor. Roll-out algorithms computation affected by:
  - Speed to sample from model.
  - Speed to execute rollout policy.



# Monte Carlo Tree Search



Exact implementation of back-up can vary. This slide shows using Monte Carlo return.

# Akhil's Presentation

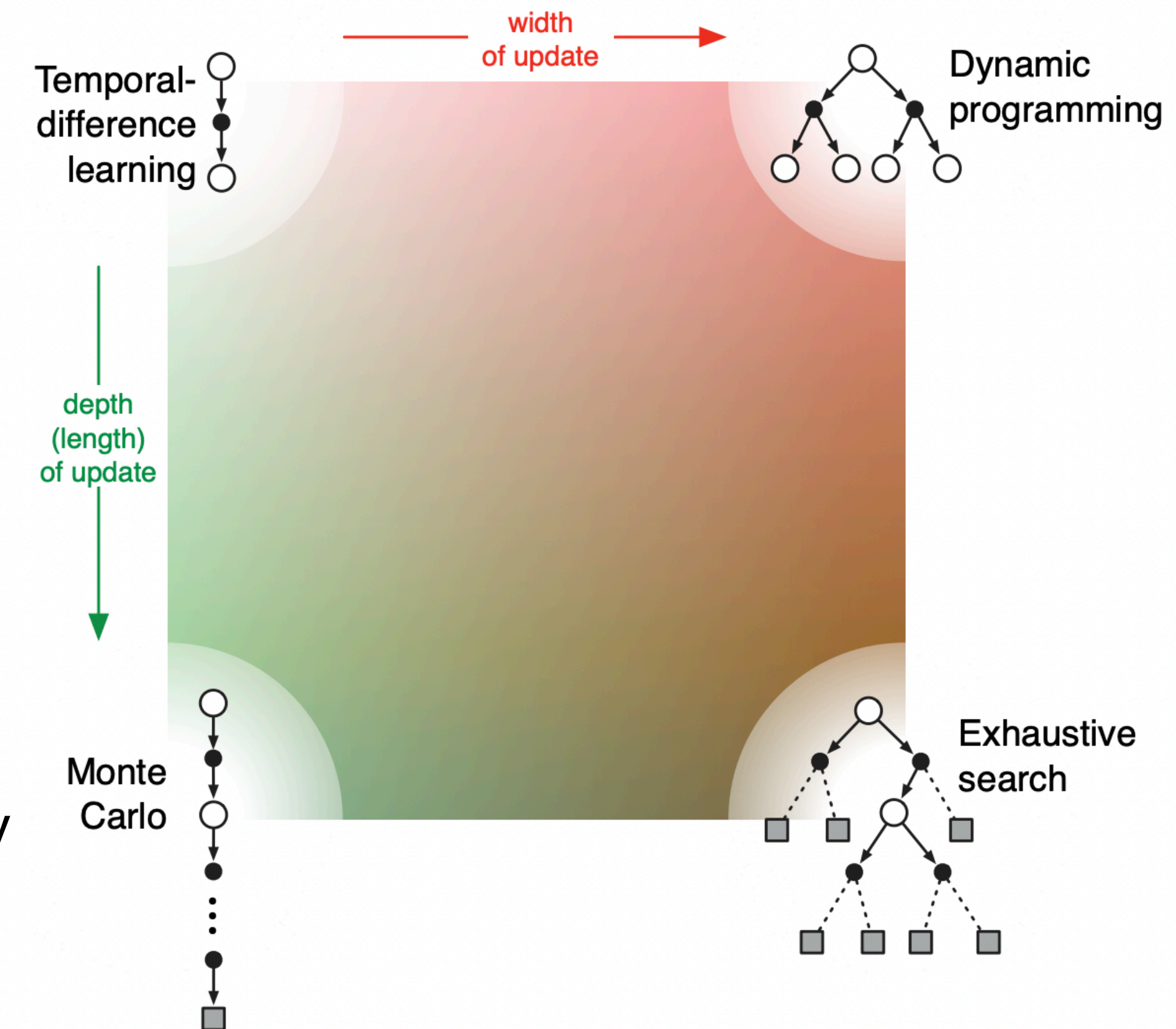
**Mastering the game of Go with deep neural networks and tree search.**

Silver et al. 2016

Slides

# Part I Summary

- Functions (policies, value functions, and models) have been represented as look-up tables.
- We have seen 4 types of algorithms:
  - Dynamic programming methods.
  - Model-free Monte Carlo methods.
  - Model-free temporal difference learning methods.
  - Model-based learning and planning methods.
- All algorithms we have seen are instances of generalized policy iteration:
  - $\pi_0 \rightarrow q_0 \rightarrow \dots \rightarrow \pi_k \rightarrow q_k \rightarrow \pi_{k+1} \rightarrow \dots \rightarrow q_\star \rightarrow \pi_\star$



# Part I Summary

- Much intuition and understanding carries forward as we move into Part II.
  - Returns and values defined similarly.
  - On-policy and off-policy methods.
  - Exploration vs. Exploitation trade-off.
- Looking ahead:
  - The learning agent has limited capacity to model  $v_{\pi}(s)$  for all  $s$ .
  - The learning agent may never visit the same state twice.

# Summary

- Models can be used to produce simulated experience to learn from.
  - Makes better use of finite data.
  - Distribution models permit *expected updates* such as those made by RTDP.
  - Sample models are generally easier to acquire and can be used with *sample* updates.
- Models can be used to compute a better decision than acting greedily w.r.t. an inaccurate action-value function.
  - Requires planning at decision time.
  - Computation is the bottleneck for making better decisions.

# Action Items

- Complete project proposals
- Complete homework.
- Begin reading Chapter 9 and 11.