Planning and Learning

Robert Platt
Northeastern University

(some slides/material borrowed from Rich Sutton)

Planning

<u>What do you think of when you think about "planning"?</u>

– often, the word "planning" often means a specific class of algorithm

– here, we use "planning" to mean any computational process that uses a model to create or improve a policy

For example: an unusual way to do planning

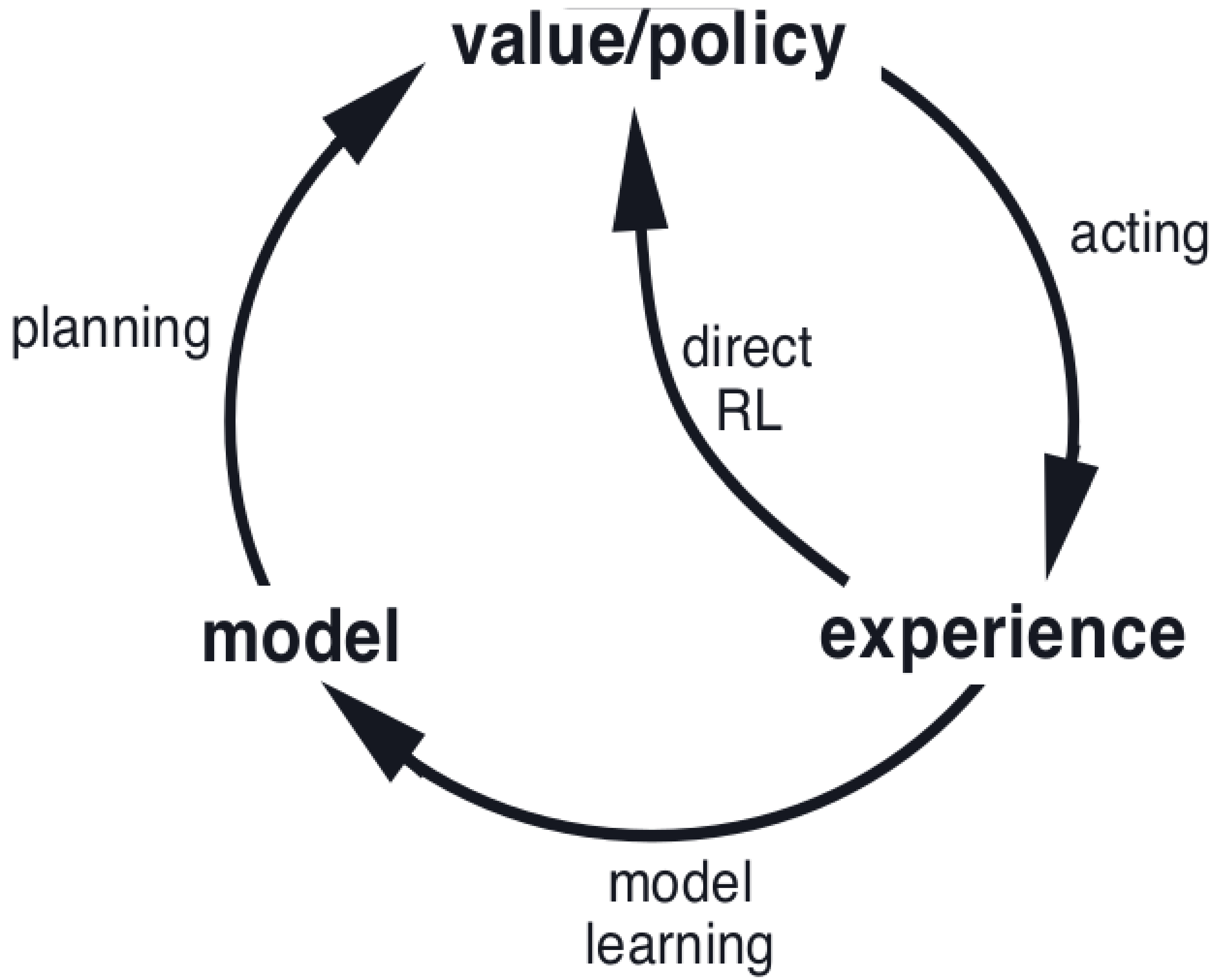
Random-sample one-step tabular Q-planning

Loop forever:

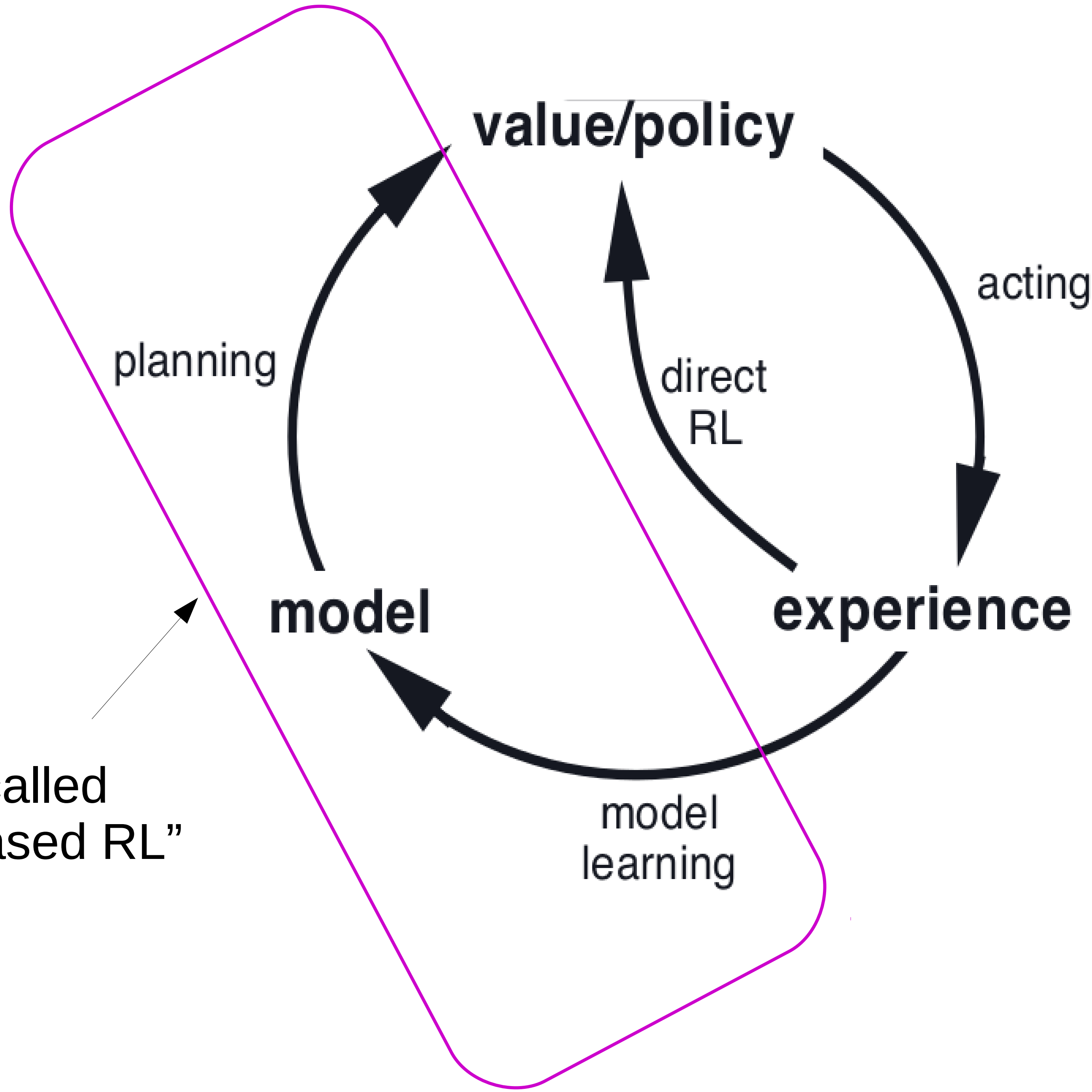
1. Select a state, $S \in \mathcal{S}$, and an action, $A \in \mathcal{A}(S)$, at random
2. Send S, A to a sample model, and obtain
   a sample next reward, R, and a sample next state, S'
3. Apply one-step tabular Q-learning to S, A, R, S':
   $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$

– why does this satisfy our expanded definition?

Planning v Learning



Planning v Learning



Often called
“model-based RL”

Models in RL

Model: anything the agent can use to predict how the environment will respond to its actions

Two types of models:

1. Distribution model: description of all possibilities and their probabilities

2. Sample model: a.k.a. a simulation model
   – given a *s,a* pair, the sample model returns next state & reward
   – a sample model is often much easier to get than the distribution model

Models in RL

Model: anything the a ent will respond to its actions

This is how we defined “model” at the beginning of this course

Two types of models:

1. Distribution model: description of all possibilities and their probabilities

2. Sample model: a.k.a. a simulation model
   – given a *s,a* pair, the sample model returns next state & reward
   – a sample model is often much easier to get than the distribution model

In this section, we’re going to use this type of model a lot

Planning

<u>An unusual way to do planning:</u>

$$\text{model} \longrightarrow \text{simulated experience} \xrightarrow{\text{backups}} \text{values} \longrightarrow \text{policy}$$

Random-sample one-step tabular Q-planning

Loop forever:
1. Select a state, $S \in \mathcal{S}$, and an action, $A \in \mathcal{A}(S)$, at random
2. Send S, A to a sample model, and obtain
   a sample next reward, R, and a sample next state, S'
3. Apply one-step tabular Q-learning to S, A, R, S':
   $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$

Planning

An unusual way to do planning:

$$\text{model} \longrightarrow \text{simulated experience} \xrightarrow{\text{backups}} \text{values} \longrightarrow \text{policy}$$

Random-sample one-step tabular Q-planning

Loop forever:

1. Select a state, $S \in \mathcal{S}$, and an action, $A \in \mathcal{A}(S)$, at random
2. Send S, A to a sample model, and obtain
   a sample next reward, R, and a sample next state, S'
3. Apply one-step tabular Q-learning to S, A, R, S':
   $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$

Here, we're using a sample model,
but we don't learn the model

Dyna-Q

Tabular Dyna-Q

Initialize $Q(s, a)$ and $Model(s, a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Loop forever:
(a) $S \leftarrow$ current (nonterminal) state
(b) $A \leftarrow \varepsilon$-greedy(S, Q)
(c) Take action A; observe resultant reward, R, and state, S'
(d) $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
(e) $Model(S, A) \leftarrow R, S'$ (assuming deterministic environment)
(f) Loop repeat n times:
$S \leftarrow$ random previously observed state
$A \leftarrow$ random action previously taken in S
$R, S' \leftarrow Model(S, A)$
$Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$

<u>Essentially, perform these two steps continuously:</u>

1. learn model
2. plan using current model estimate

Dyna-Q

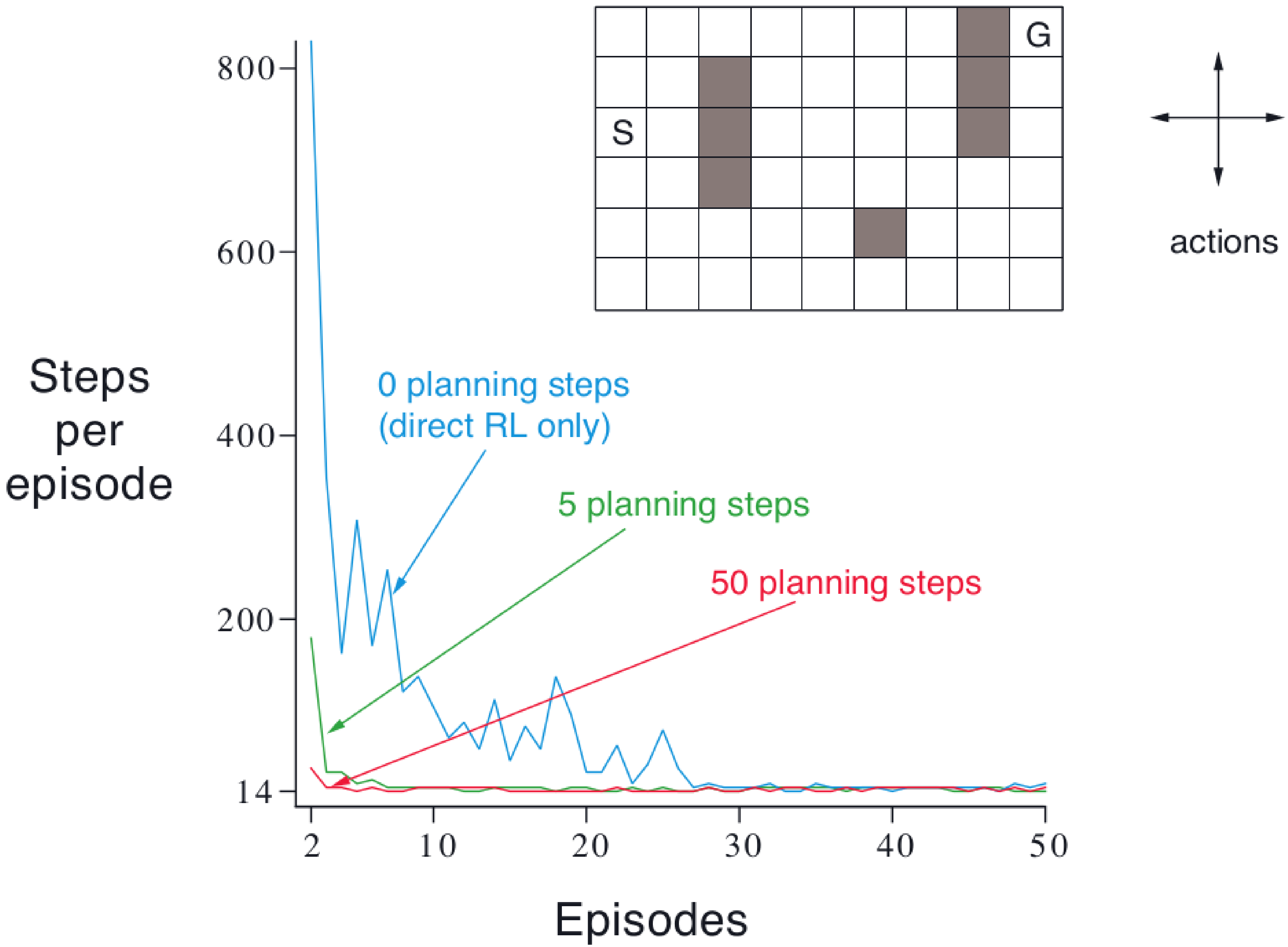
Tabular Dyna-Q

Initialize $Q(s, a)$ and $Model(s, a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Loop forever:
 (a) $S \leftarrow$ current (nonterminal) state
 (b) $A \leftarrow \varepsilon$-greedy(S, Q)
 (c) Take action A; observe resultant reward, R, and state, S'
 (d) $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
 (e) $Model(S, A) \leftarrow R, S'$ (assuming deterministic environment)
 (f) Loop repeat n times:
  $S \leftarrow$ random previously observed state
  $A \leftarrow$ random action previously taken in S
  $R, S' \leftarrow Model(S, A)$
  $Q(S, A) \leftarrow Q(S, A) + \alpha\big[$ $\max_a Q(S', a) - Q(S, A)\big]$

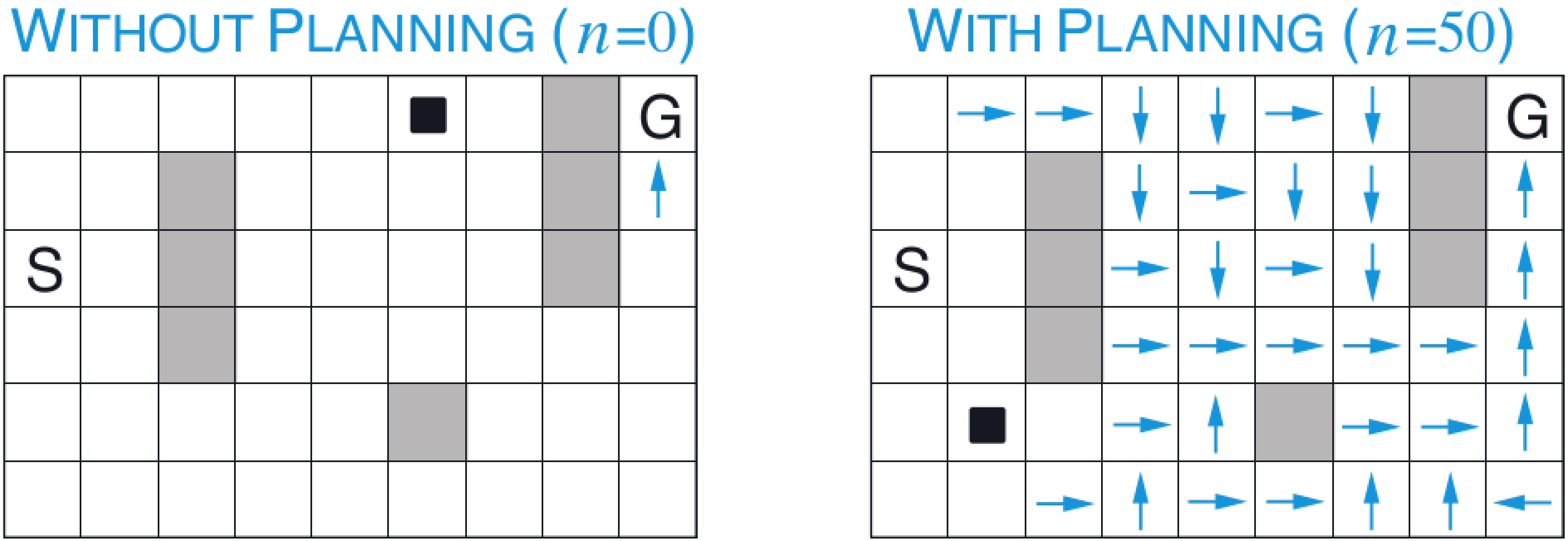
<u>Essentially, perform thes</u>
1. learn model
2. plan using current

<u>This "model" could be very simple</u>
– it could just be a memory of
 previously experienced transitions
– make predictions based on memory
 of most recent previous outcomes in
 this state/action.

Dyna-Q on a Simple Maze



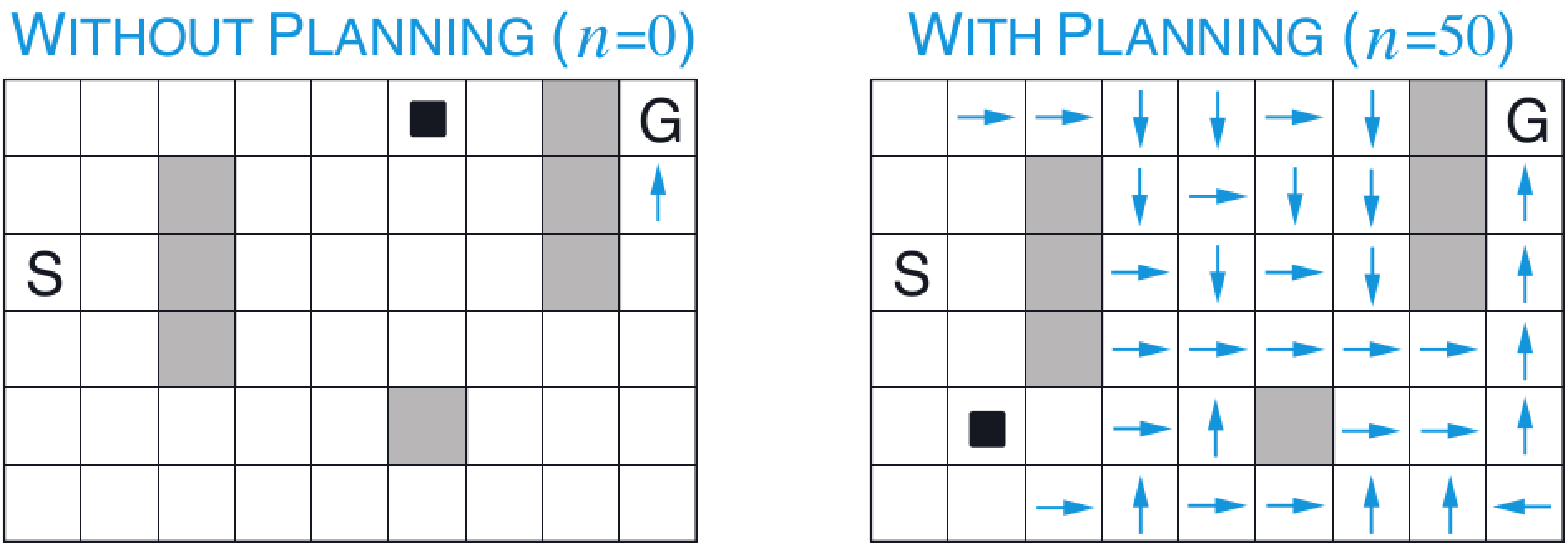
Why does Dyna-Q do so well?



Policies found using q-learning vs dyna-q halway through second episode

- dyna-q w/ n=50
- optimal policy after three episodes!

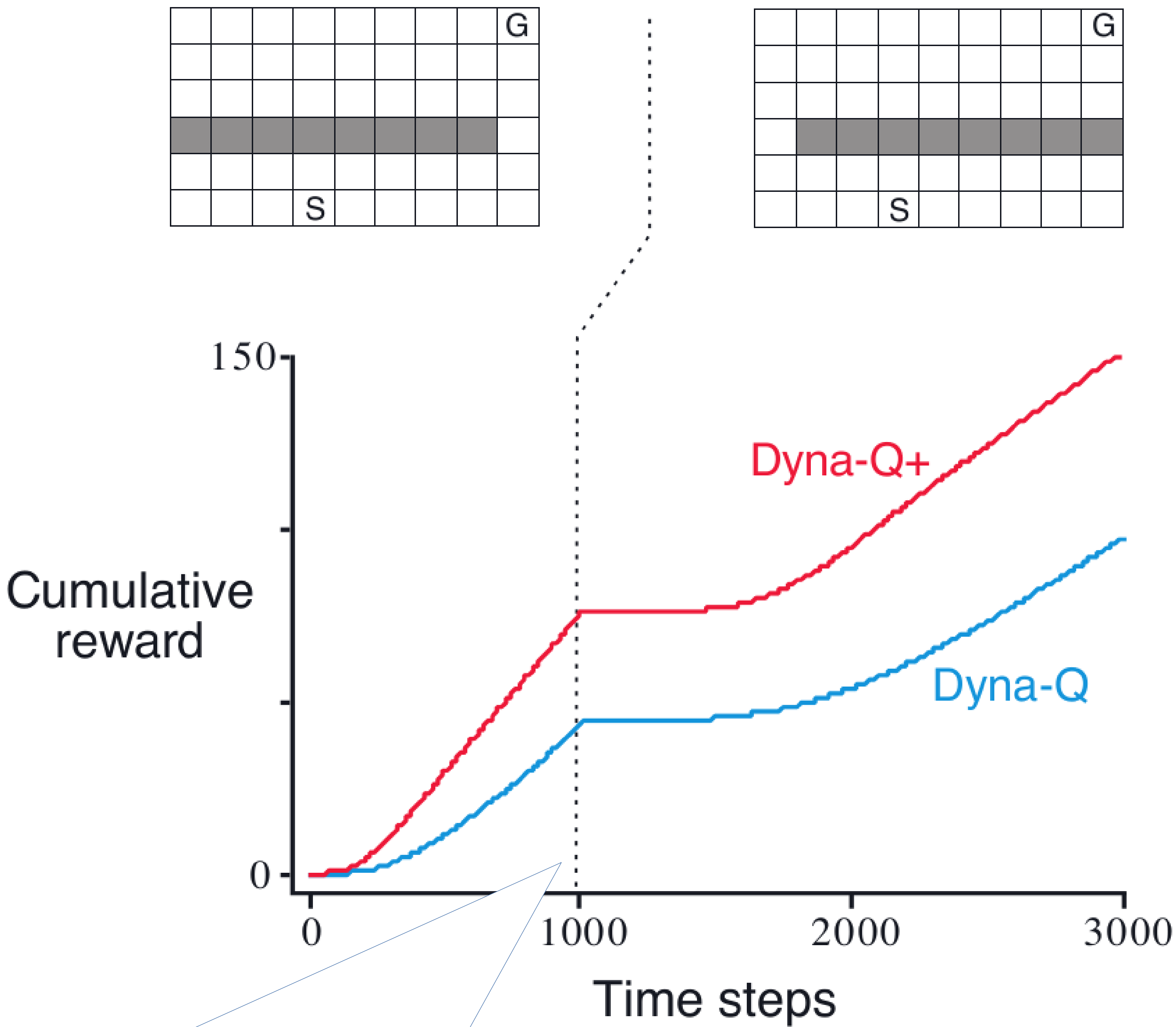
Think-pair-share



Exercise 8.1 The nonplanning method looks particularly poor in Figure 8.3 because it is a one-step method; a method using multi-step bootstrapping would do better. Do you think one of the multi-step bootstrapping methods from Chapter 7 could do as well as the Dyna method? Explain why or why not. □

What happens if model changes or is mis-estimated?

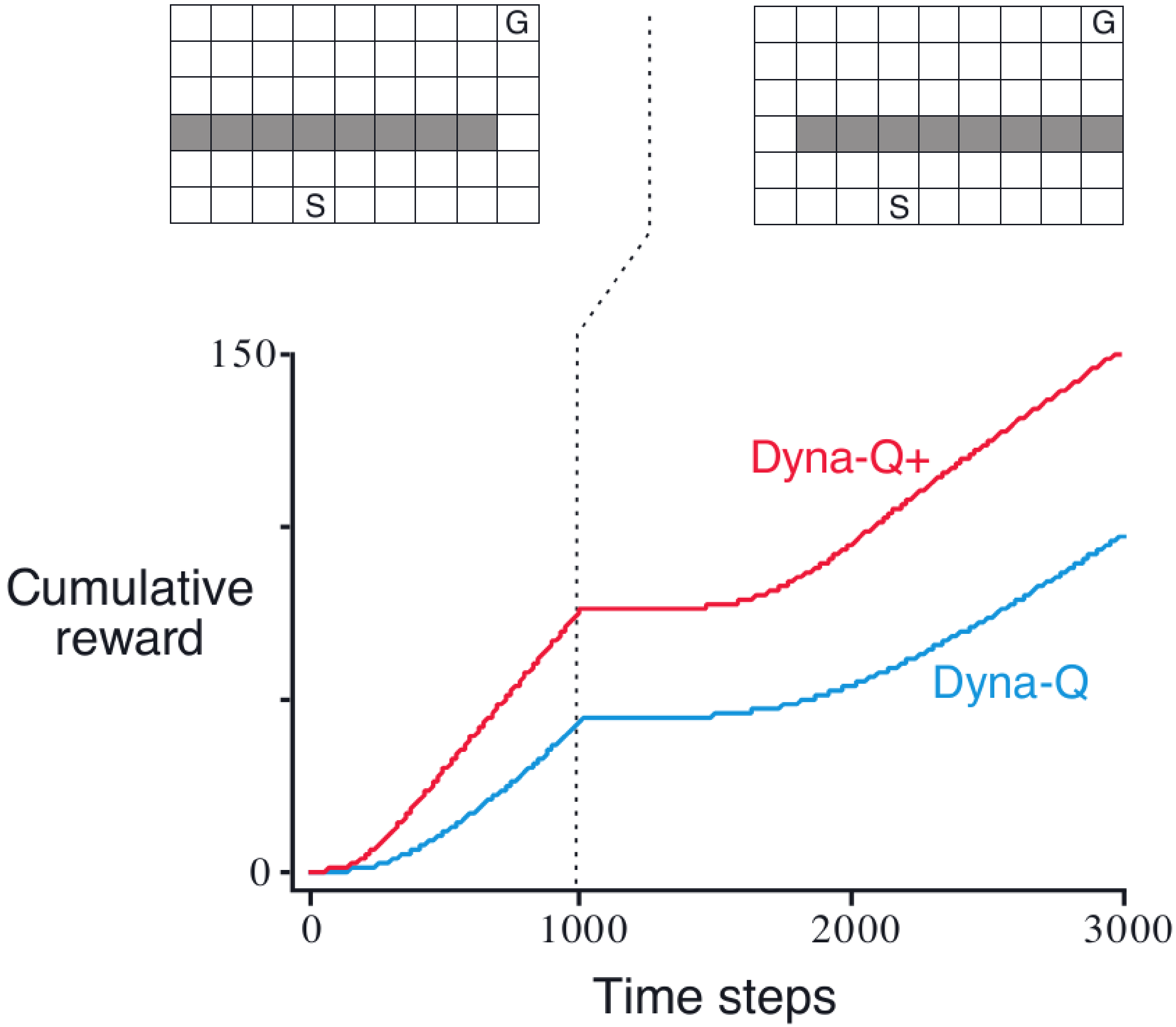
(SB, Example 8.2)



Environment changes here

Think-pair-share

(SB, Example 8.2)



Questions:

– why does dyna-q stop getting reward?

– why does it start again?

What is dyna-Q+?

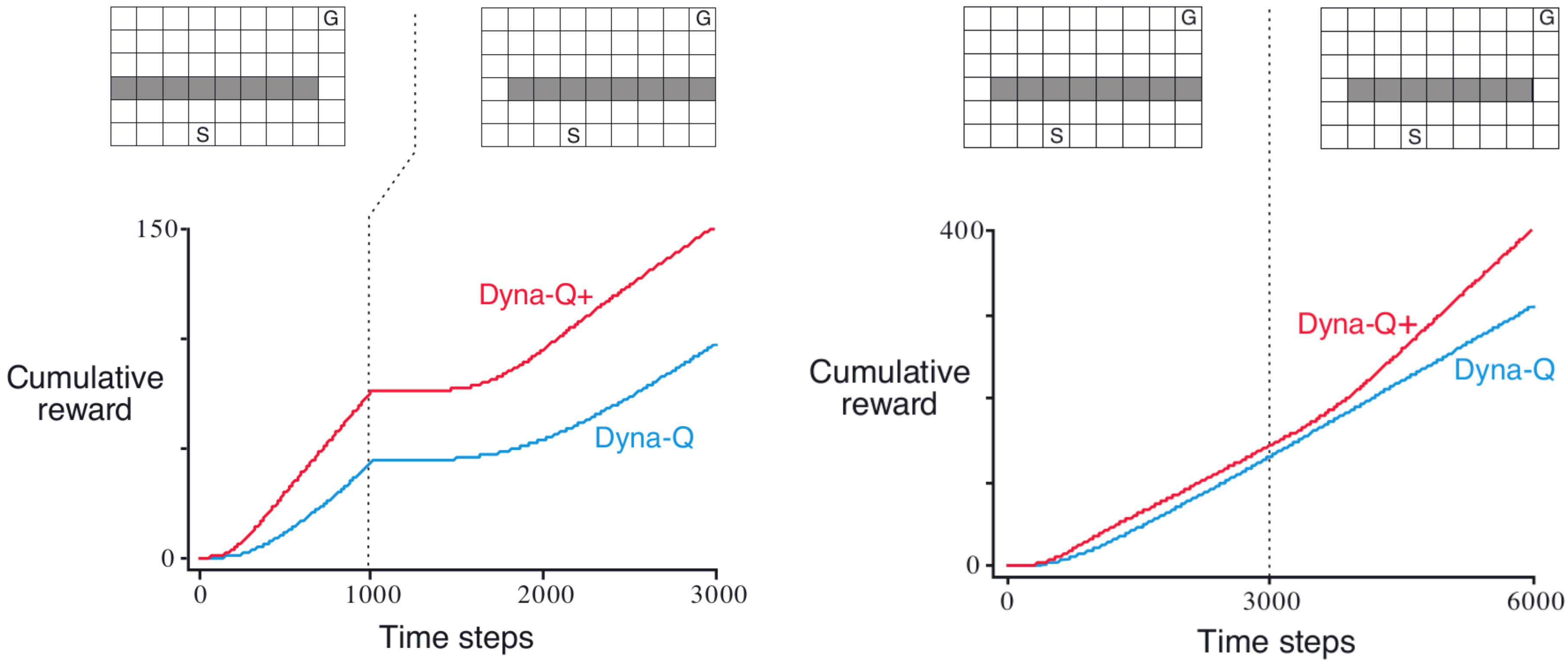
- Uses an "exploration bonus":
  - Keeps track of time since each state-action pair was tried for real
  - An extra reward is added for transitions caused by state-action pairs related to how long ago they were tried: the longer unvisited, the more reward for visiting

$$R + \kappa\sqrt{\tau}$$

time since last visiting the state-action pair

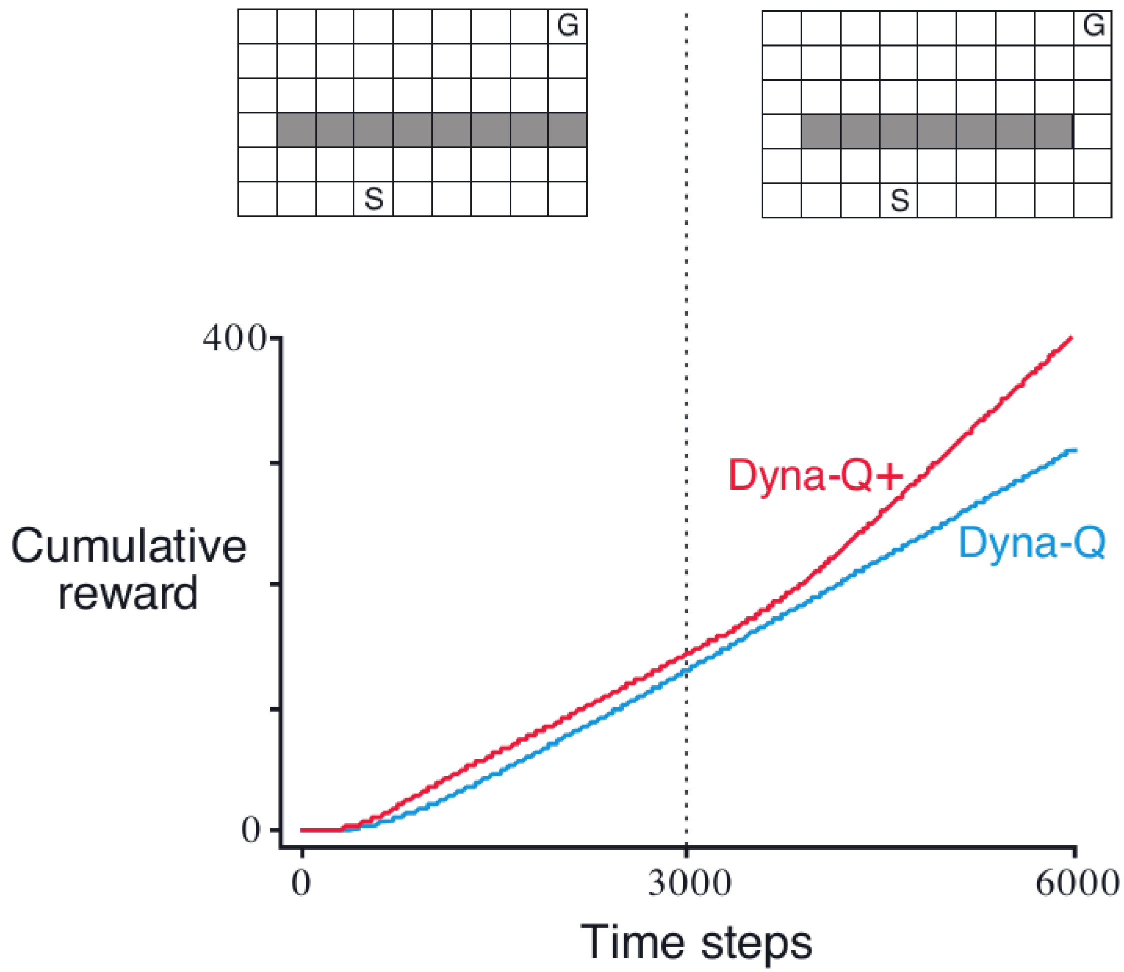
  - The agent actually "plans" how to visit long unvisited states

Think-pair-share



Exercise 8.2 Why did the Dyna agent with exploration bonus, Dyna-Q+, perform better in the first phase as well as in the second phase of the blocking and shortcut experiments? □

Dyna-Q



Exercise 8.3 Careful inspection of Figure 8.5 reveals that the difference between Dyna-Q+ and Dyna-Q narrowed slightly over the first part of the experiment. What is the reason for this? □

Prioritized Sweeping

Tabular Dyna-Q

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Loop forever:
(a) $S \leftarrow$ current (nonterminal) state
(b) $A \leftarrow \varepsilon$-greedy(S, Q)
(c) Take action A; observe resultant reward, R, and state, S'
(d) $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
(e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)
(f) Loop repeat n times:
$S \leftarrow$ random previously observed state
$A \leftarrow$ random action previously taken in S
$R, S' \leftarrow Model(S,A)$
$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

Unfocused replay from model

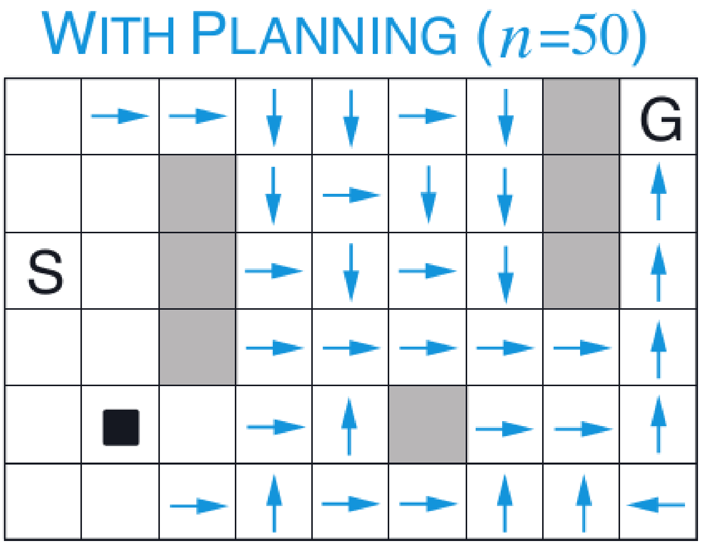
Prioritized Sweeping

Tabular Dyna-Q

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Loop forever:
(a) $S \leftarrow$ current (nonterminal) state
(b) $A \leftarrow \varepsilon$-greedy(S,Q)
(c) Take action A; observe resultant reward, R, and state, S'
(d) $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
(e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)
(f) Loop repeat n times:
$S \leftarrow$ random previously observed state
$A \leftarrow$ random action previously taken in S
$R, S' \leftarrow Model(S,A)$
$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

Unfocused replay from model
– can we do better?

Prioritized Sweeping



Instead of replaying **all** of these transitions on each iteration, just replay the important ones…

– Which states or state-action pairs should be generated during planning?

– Work backward from states who's value has just changed
  – Maintain a priority queue of state-action pairs whose values would change a lot if backed up, prioritized by the size of the change
  – When a new backup occurs, insert predecessors according to their priorities

Prioritized Sweeping

Prioritized sweeping for a deterministic environment

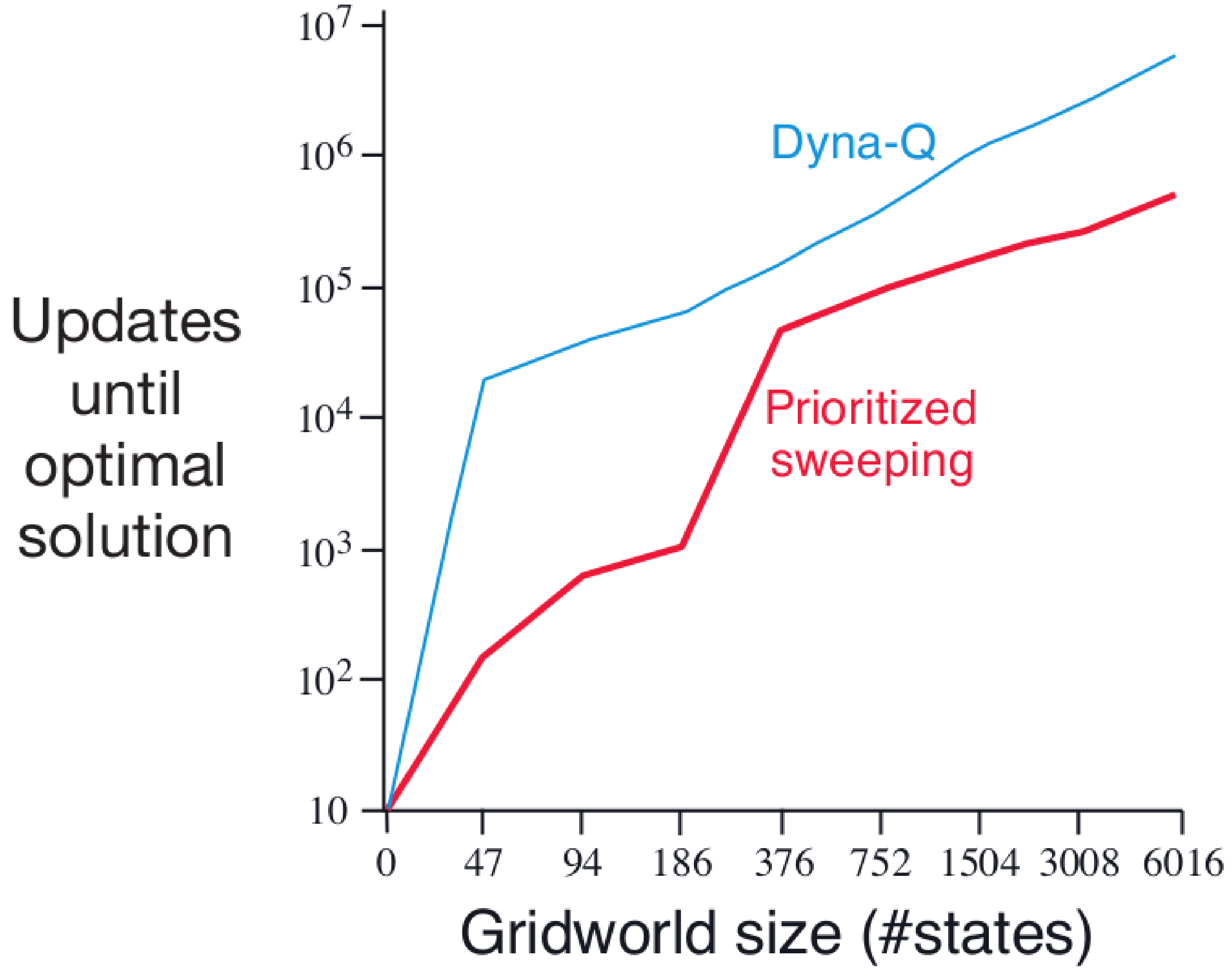
Initialize $Q(s,a)$, $Model(s,a)$, for all s, a, and $PQueue$ to empty
Loop forever:
- (a) $S \leftarrow$ current (nonterminal) state
- (b) $A \leftarrow policy(S, Q)$
- (c) Take action A; observe resultant reward, R, and state, S'
- (d) $Model(S, A) \leftarrow R, S'$
- (e) $P \leftarrow |R + \gamma \max_a Q(S', a) - Q(S, A)|$.

TD error

- (f) if $P > \theta$, then insert S, A into $PQueue$ with priority P
- (g) Loop repeat n times, while $PQueue$ is not empty:
  - $S, A \leftarrow first(PQueue)$
  - $R, S' \leftarrow Model(S, A)$
  - $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
  - Loop for all $\bar{S}, \bar{A}$ predicted to lead to S:
    - $\bar{R} \leftarrow$ predicted reward for $\bar{S}, \bar{A}, S$
    - $P \leftarrow |\bar{R} + \gamma \max_a Q(S, a) - Q(\bar{S}, \bar{A})|$.
    - if $P > \theta$ then insert $\bar{S}, \bar{A}$ into $PQueue$ with priority P

what's this part doing?

Prioritized Sweeping: Performance



Both use n=5 backups per environmental interaction

Trajectory sampling

Idea: dyna-Q while sampling experiences from a trajectory rather than uniformly, i.e. from the on-policy distribution

– is it better to sample uniformly or from the on-policy distribution?