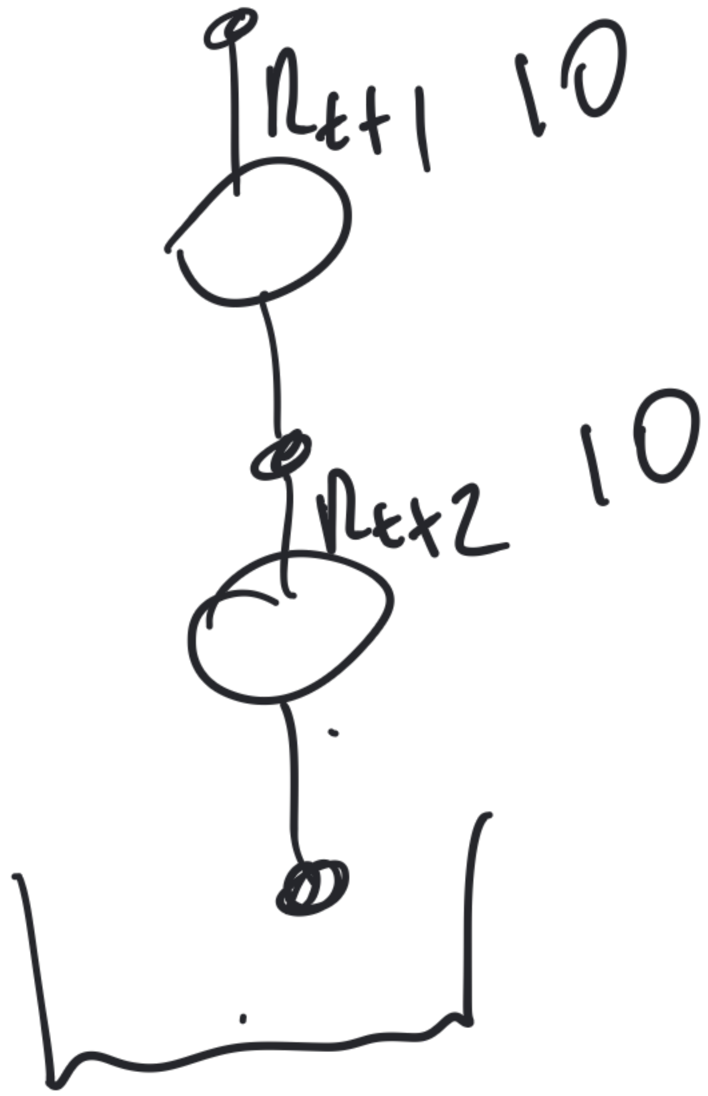
RL: Lecture 17

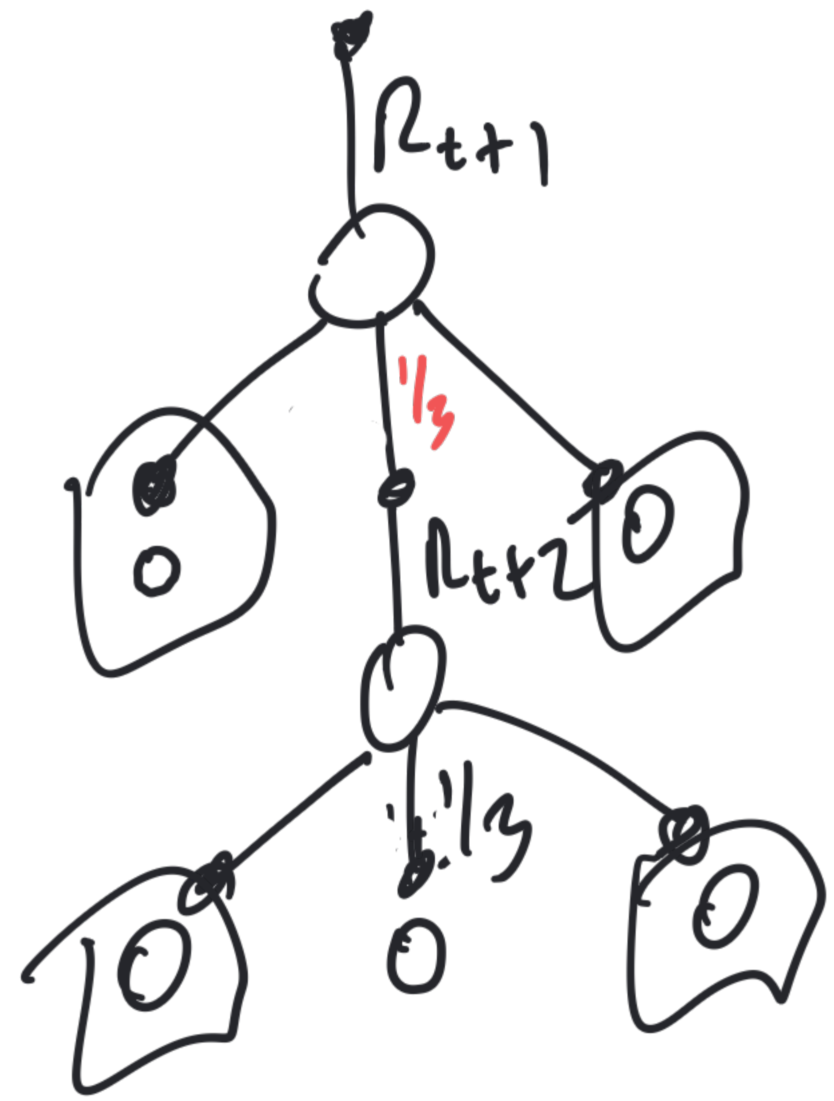
Harvey Mudd College

March 30, 2020

Neil Rhodes

on-policy   tree backup





tree backup

on-policy update

assum initially $Q(s,a) = 0 \ \forall s, \forall a$

equiprobable policy

$$Q(S_t, A_t) \mathrel{+}= \alpha\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 Q(S_{t+2}, A_{t+2}) - Q(S_t, A_t)\right]$$

$$Q(S_t, A_t) = \alpha[R_{t+1}$$

new Q = ~~R_t ≠ 0~~ $\alpha \cdot 20$

tree backup Q $= \alpha[R_{t+1} + \frac{1}{3} R_{t+2}$

Types of RL algorithms

- Model-based
- Model-free

RL algorithms

model-based

Dynamic Programming

model-free

TD

SARSA

Q-learning

expected sarsa

MC

$$\sum \underbrace{p(s, r \mid s', a')}_{\text{MDP environment}}$$

Environment

learn

model

Types of Models

- Distributional model
- Sample model

– actually have all the probs given $S, A \rightarrow$ prob $\left\{ (S', R) \ldots \right.$

more constrained model

given an (S, A) it returns a sample S', R

Given a distribution model, easy to come up w/ samples

What is planning?

given model

may be imperfect

take imaginary actions

w/o taking actions in env

get imaginary rewards & states

update policy

imagine taking an action

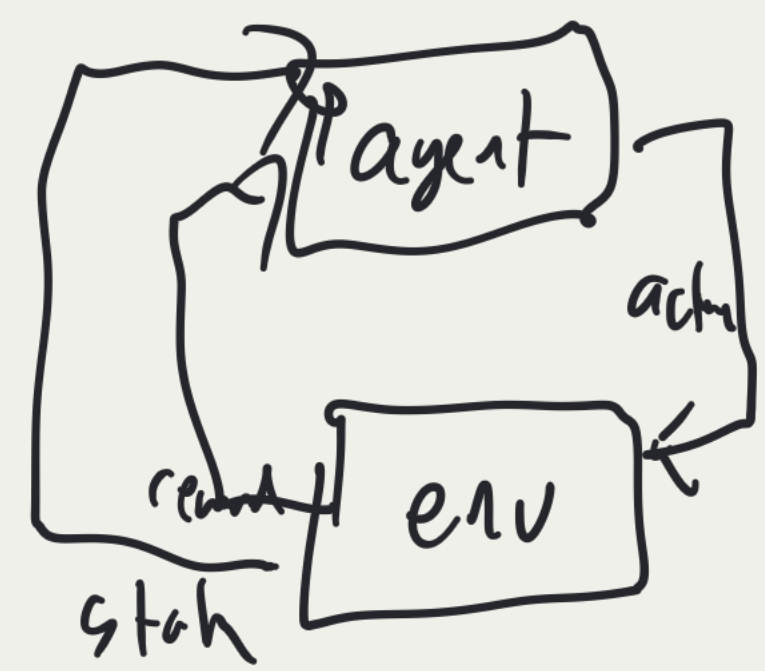
---

w/o model

Direct RL

use estimates of state values

take actions in env.



Random-sample one-step tabular Q-planning

Not learning planning ↖ uses model, not env

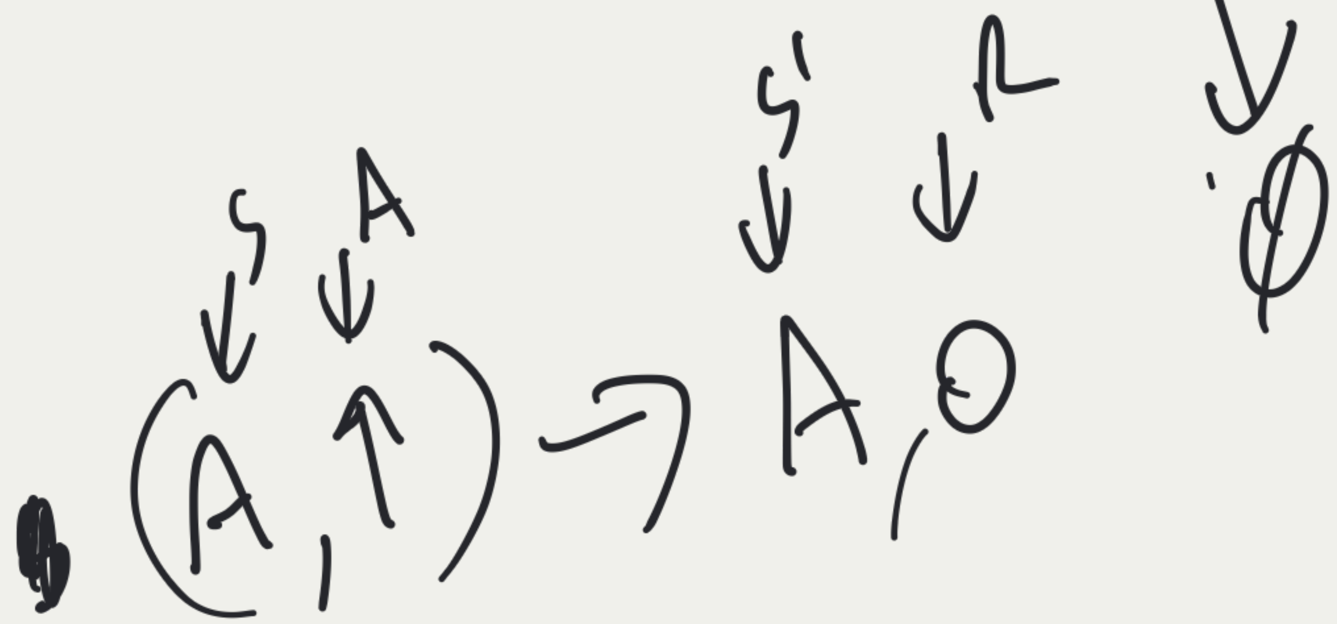
not the environment

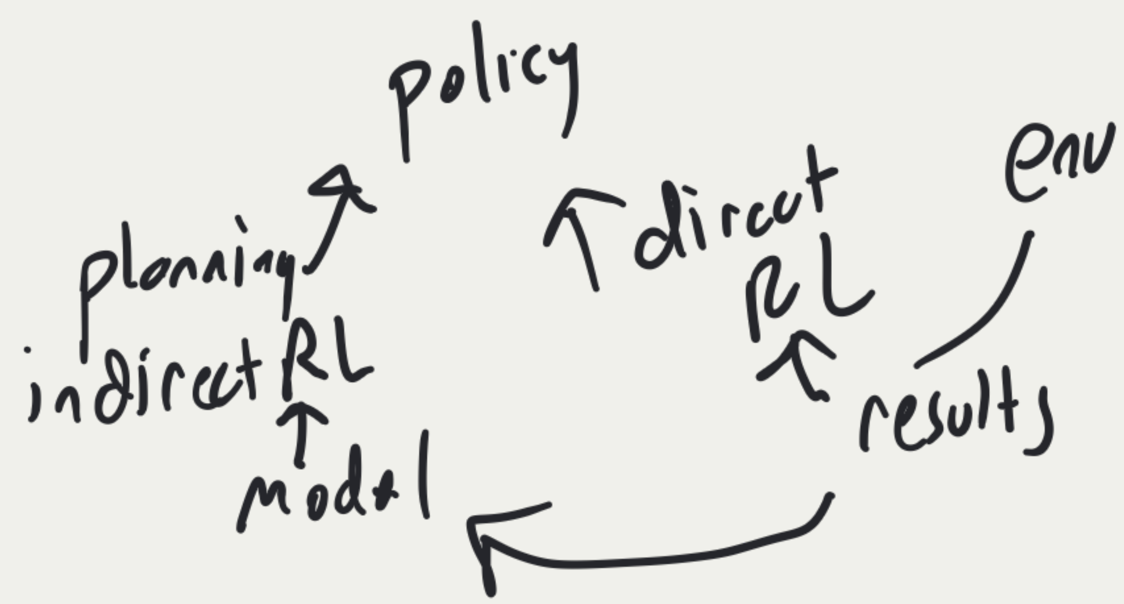
Loop forever:

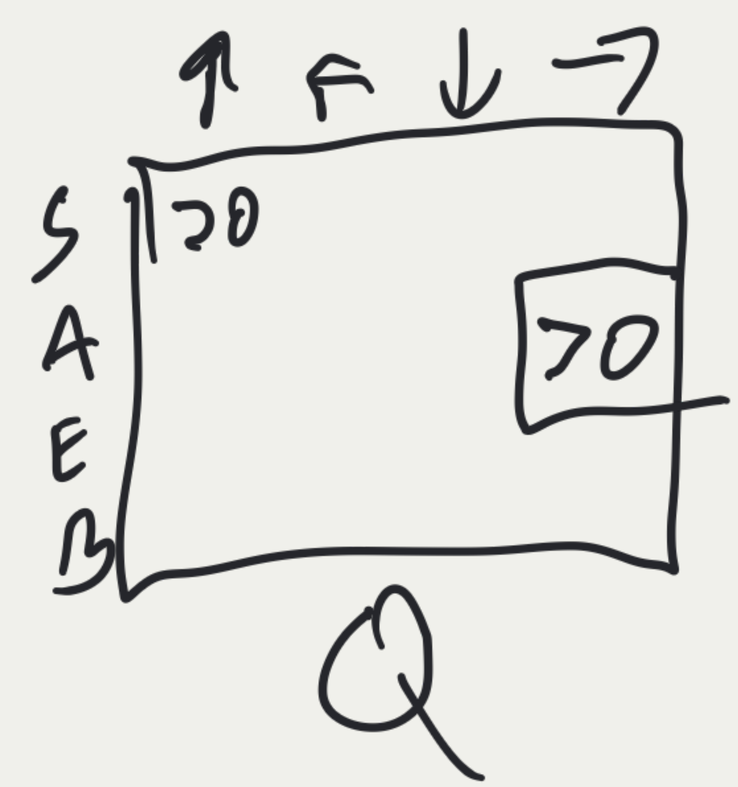
1. Select a state, S, and an action A at random
2. Send S, A to a $\boxed{\text{sample model}}$ and obtain a sample next reward R, and a sample next state, S' } ~~imagined~~ modeled S', R
3. Apply one-step tabular Q-learning to S, A, R, S'

$$Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$$

updating actual policy







Stochastic

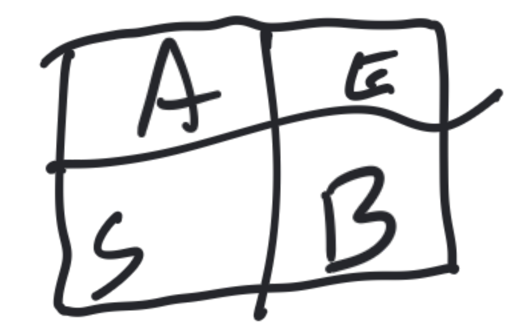
Real episodes

A —
0 B —
0

1) A 0 B 0
2) B 1
3) B 1
4) B 0
5) B 1
6) B 1
7) B 1

deterministic

Maze



$(S, \uparrow) \to A$

$\overset{S,\ A}{(A, \uparrow)} \to \overset{S',\ R}{(A, 0)}$

model

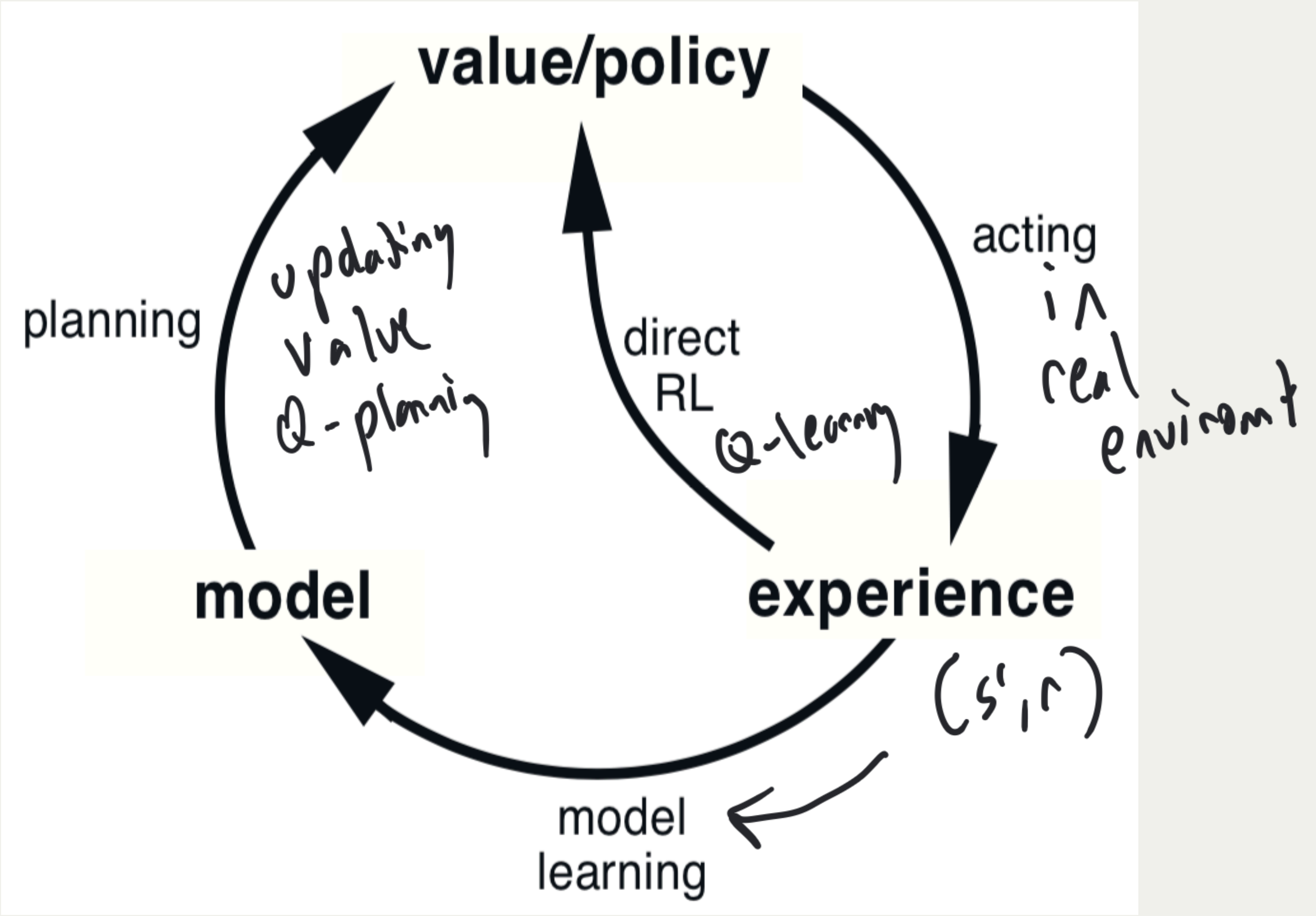
$S \times A \to S \times R$

	$\uparrow$	$\leftarrow$	$\downarrow$	$\rightarrow$
S	A, 0			
A	A, 0	A, 0		E, 1
E				
B				

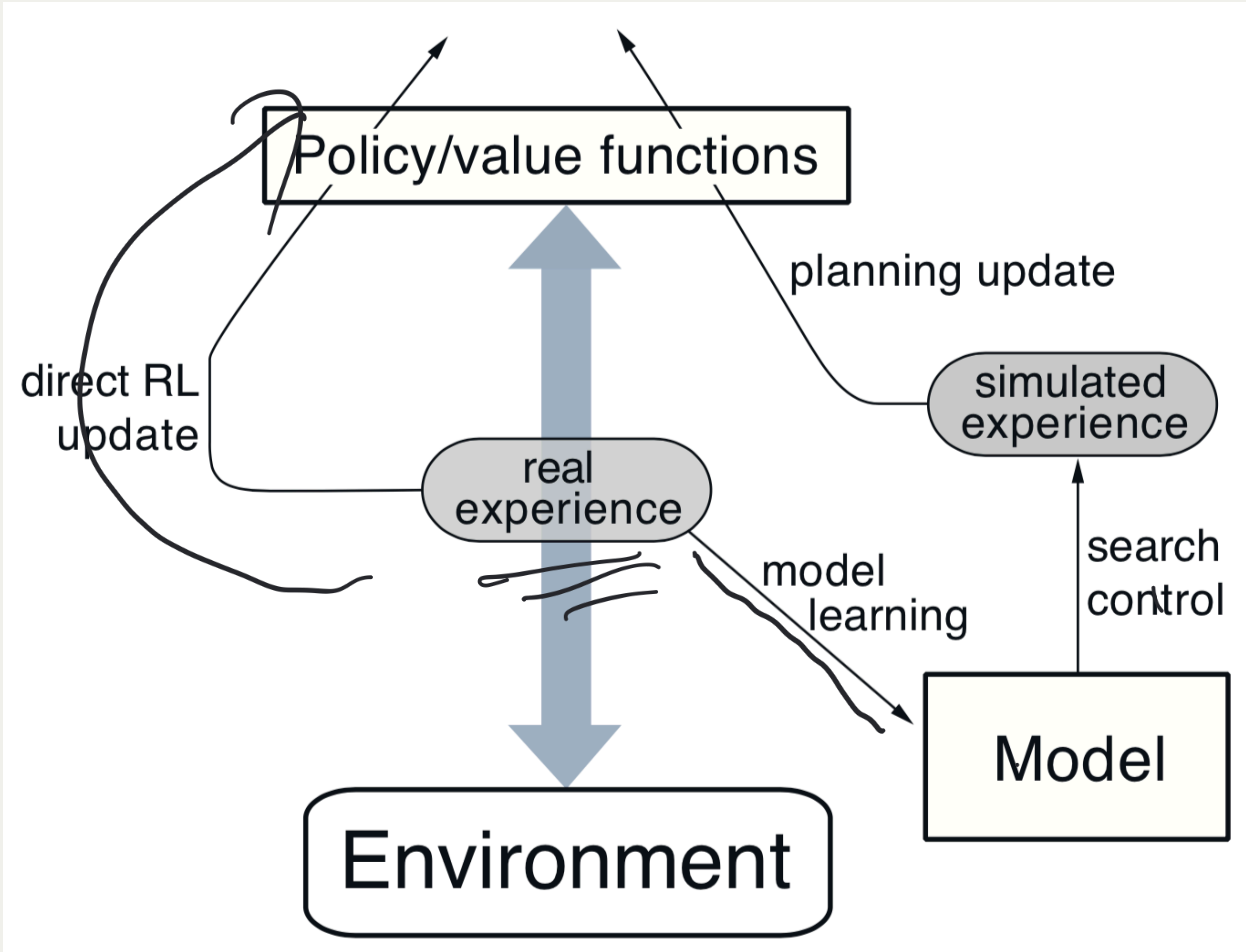
← model of maze env

S, ↑
0, A, ↑
0, A, ←
0, A, →
1, E

Online planning



Dyna Architecture



Tabular Dyna-Q Overview

- Direct RL method: one-step tabular Q-learning
- Model-learning method:
  - Assumes environment is deterministic
    - Table-based
    - Given $A_t, S_t \rightarrow R_{t+1}, S_{t+1}$, stores model[$(S_t, A_t)$] = (R_{t+1}, S_{t+1})

Tabular Dyna-Q Algorithm

Initialize $Q(s, a)$ and $Model(s, a)$ for all a, s

Loop forever:

1. $S \leftarrow$ current (nonterminal) state
2. $A \leftarrow \epsilon - \mathrm{greedy}(S, Q)$
3. Take action A; observe resultant reward R and state S'
4. $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$

direct RL

5. $\mathrm{Model}(S, A) \leftarrow (R, S')$ (assumes deterministic envirnoment)

update model

6. Loop repeat n times

   $S \leftarrow$ random previously observed state

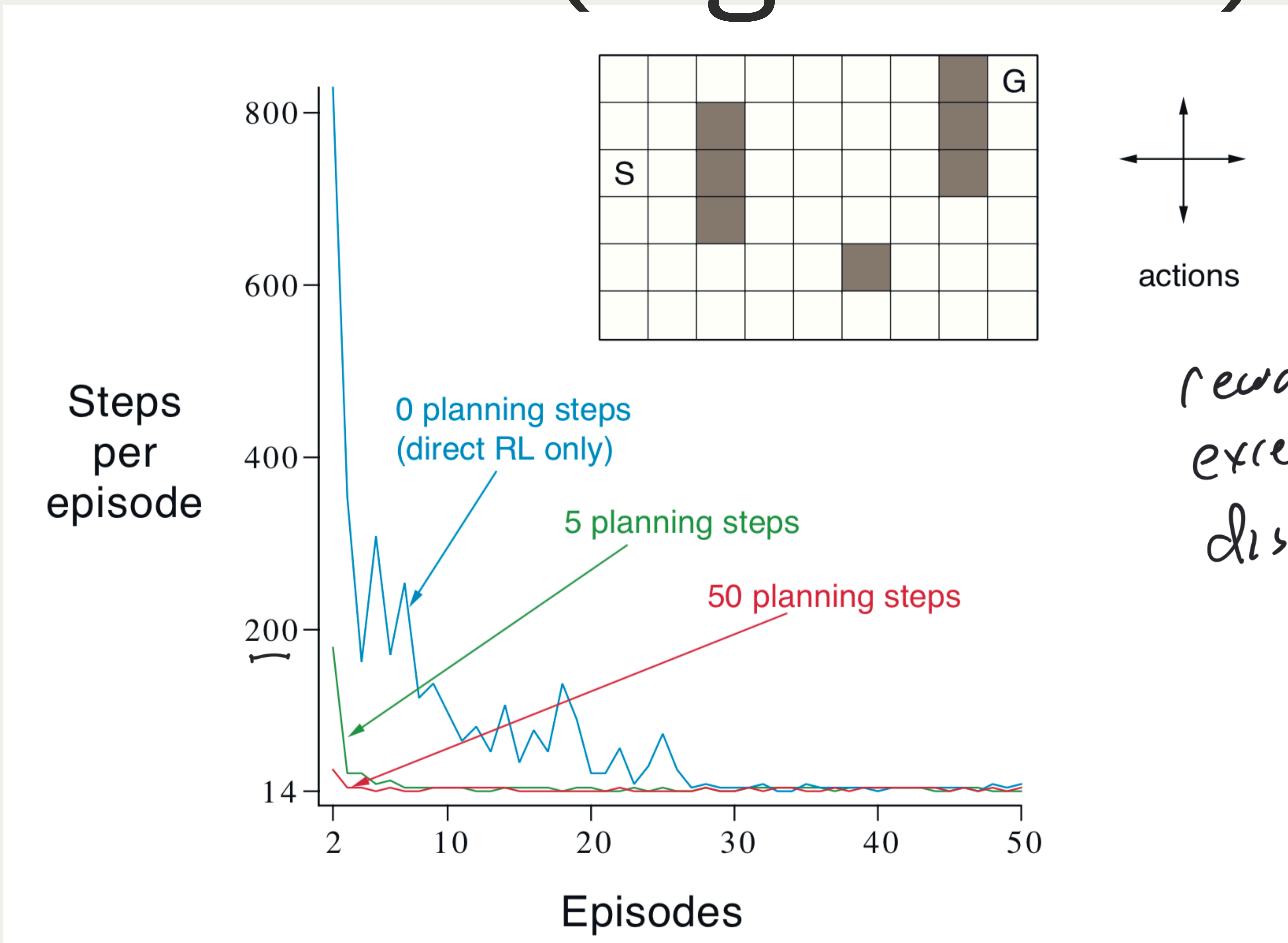
   $A \leftarrow$ random action previously taken in S

   $R, S' \leftarrow \mathrm{Model}(S, A)$

   $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$

planning

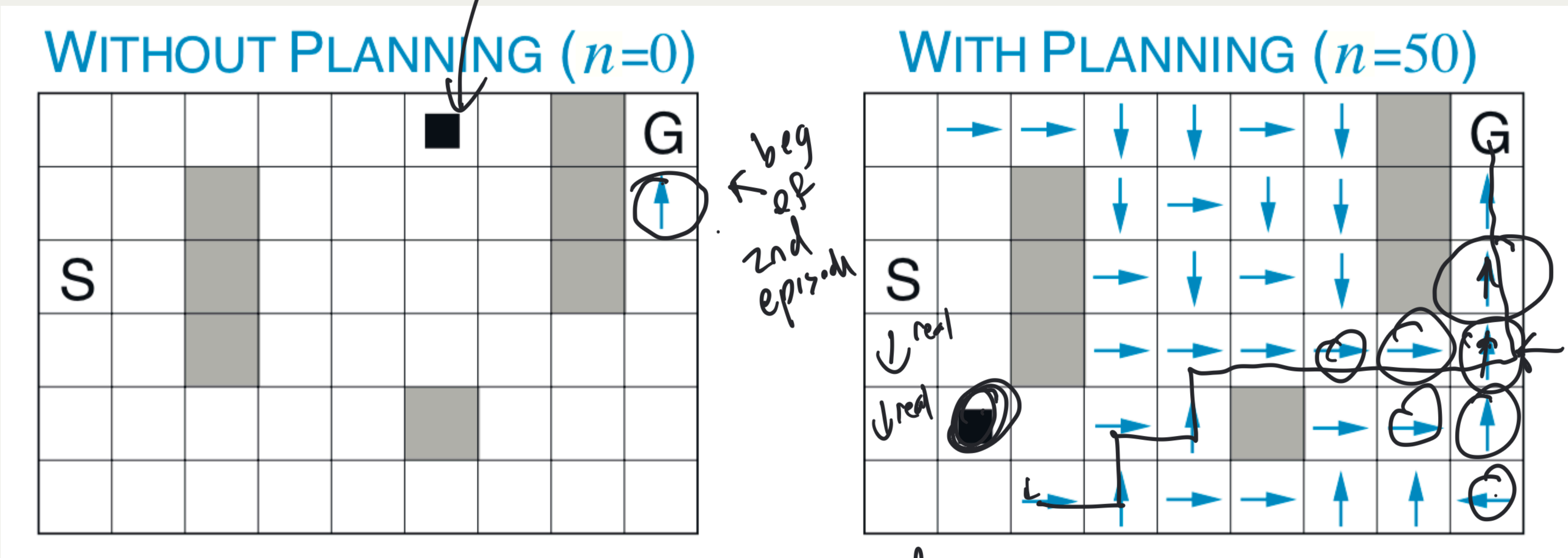
Dyna Maze (Figure 8.2)



reward = 0
except @ goal
discount = .9

Dyna Maze (Figure 8.3)

where we are

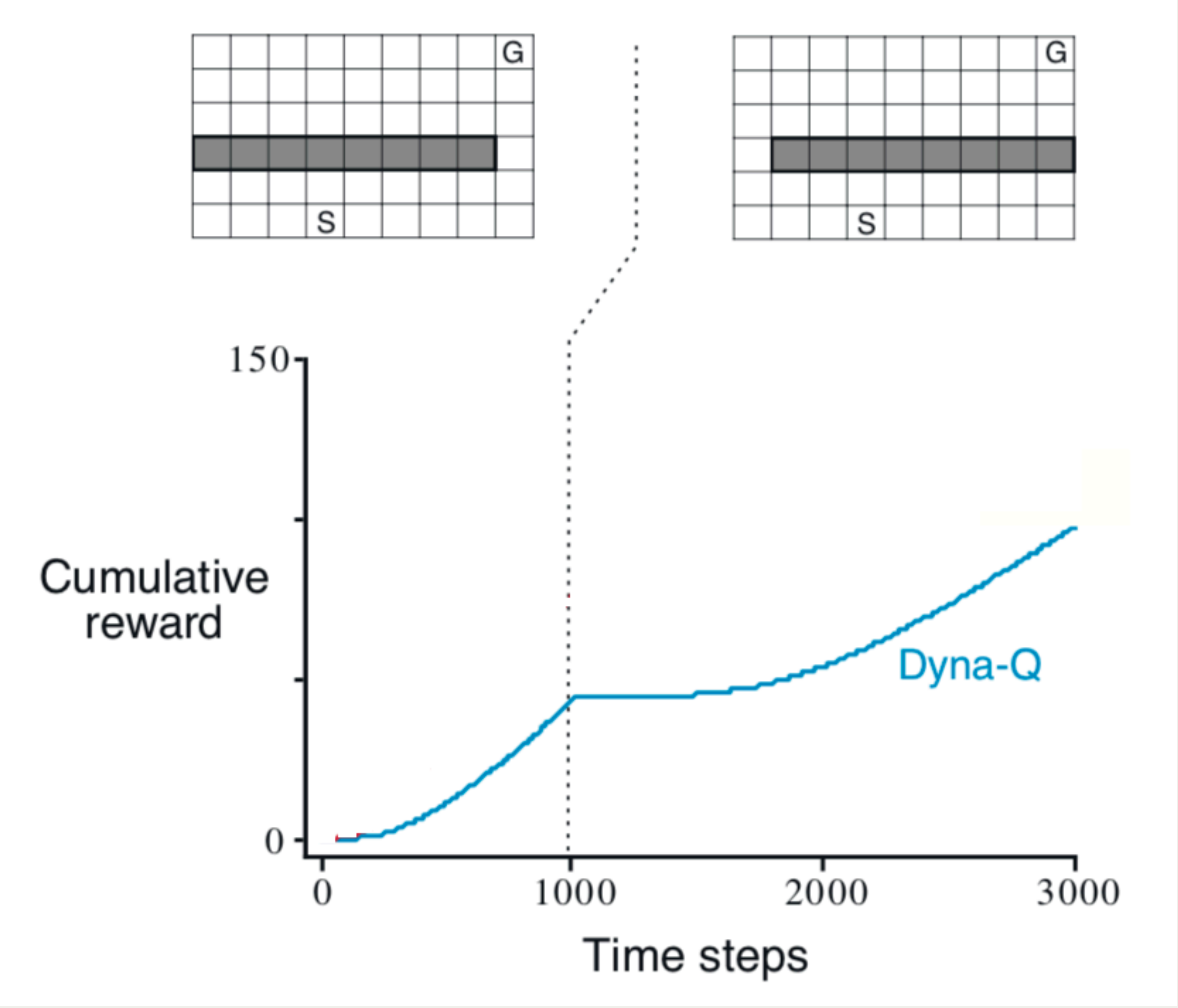


1/2 way through 2nd episode

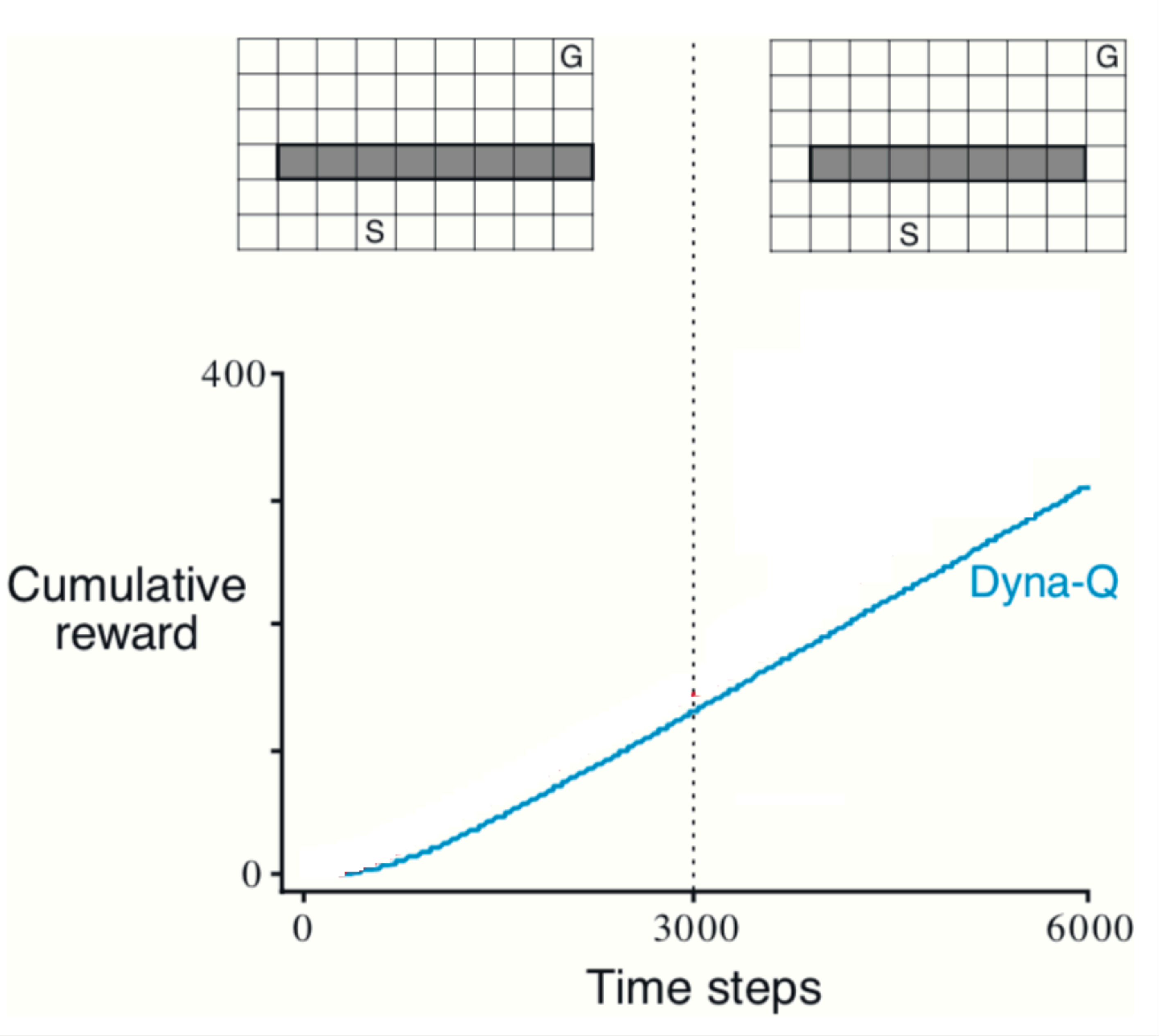
model-free 1 source of uncertainty: Q-values

model-based + additional source of uncertainty: model

When the Model is Wrong: Optimistic Model



When the Model is Wrong: Pessimistic Model



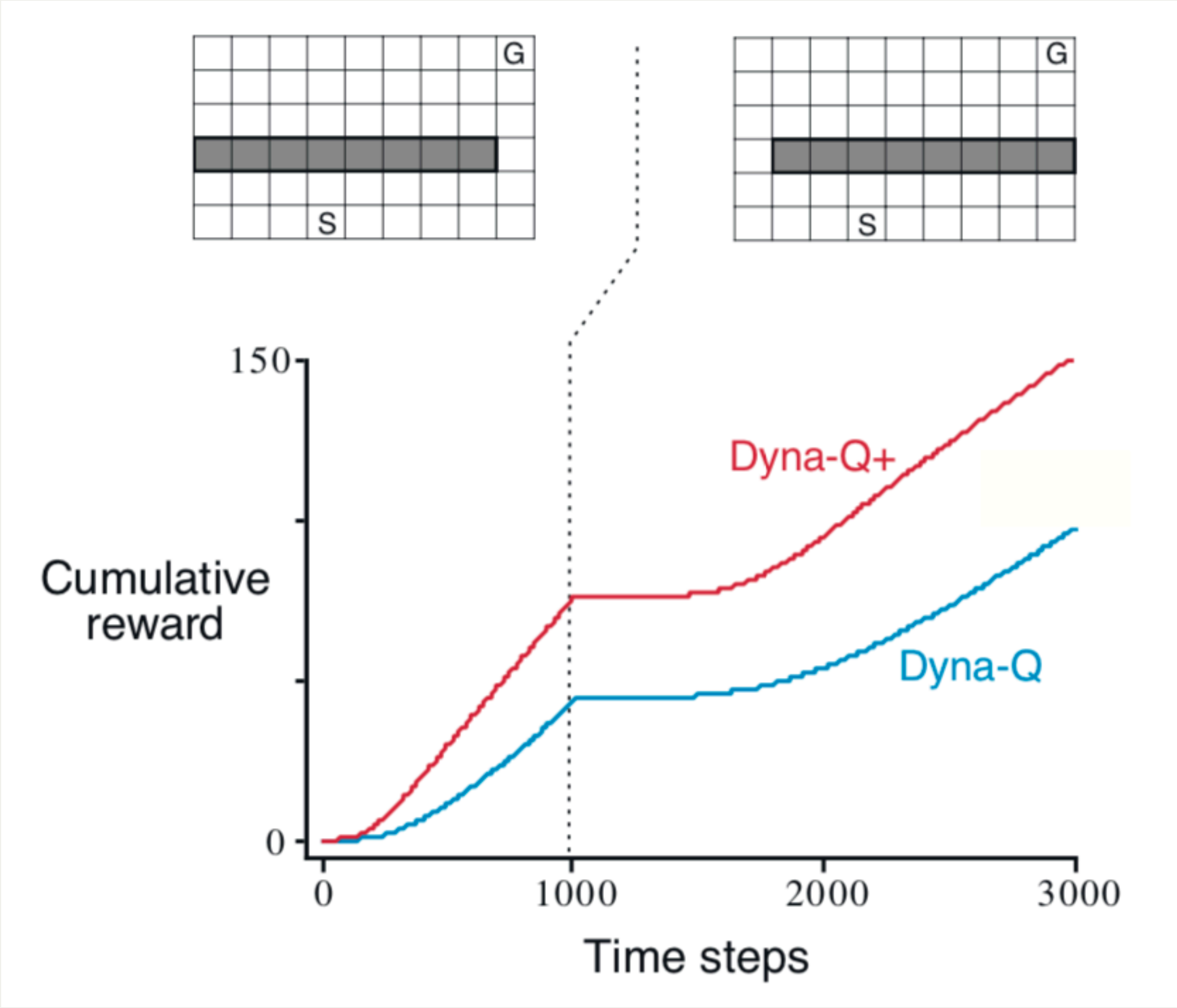
Dyna-Q+: Dyna-Q + heuristics for encouraging

- Provide an implicit reward to exploring stale transitions

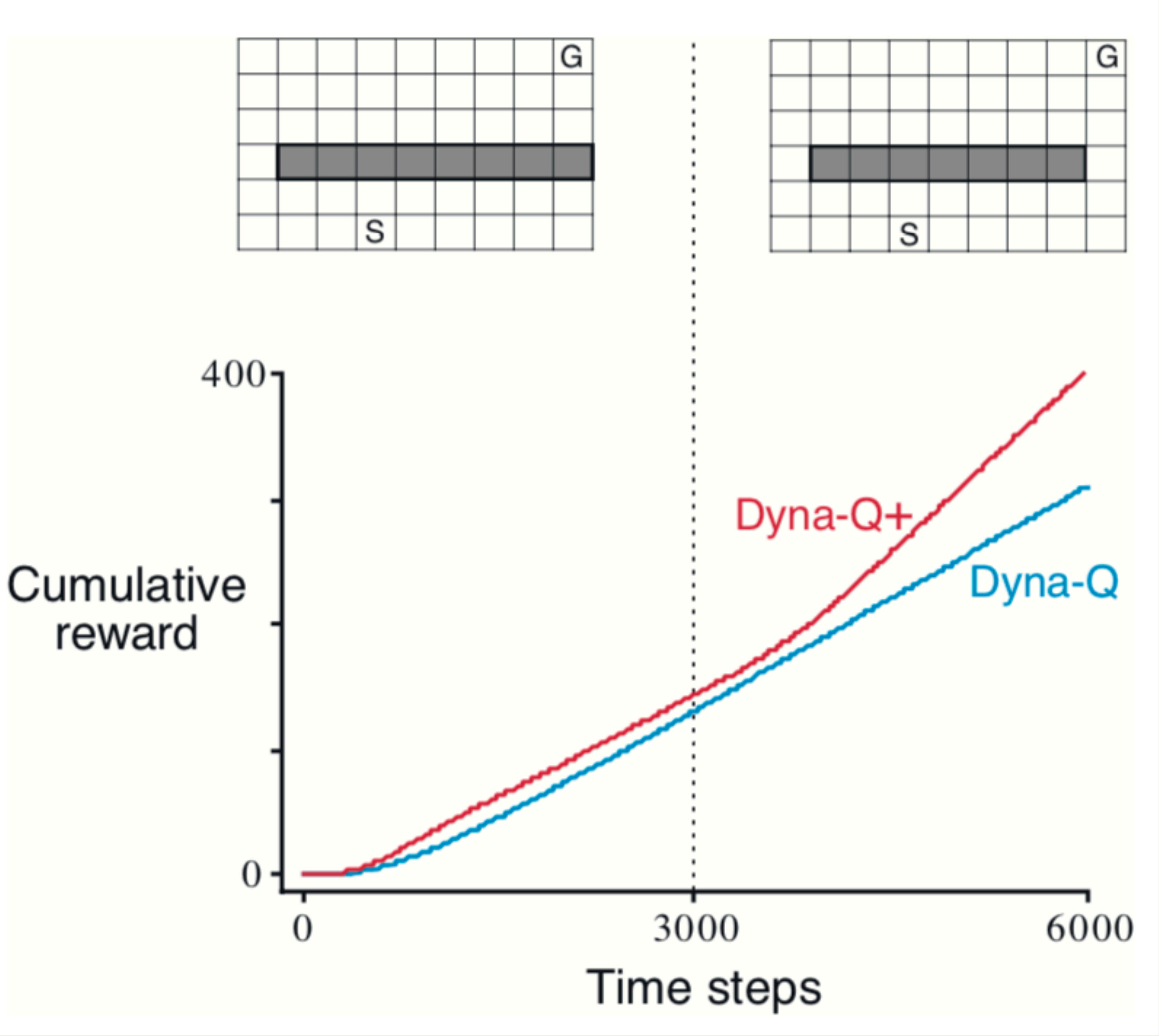
$$Q(S, A) \leftarrow Q(S, A) + \alpha[R + k\sqrt{\tau(S, A)} + \gamma \mathbf{max}_a Q(S', a) - Q(S, A)]$$

- Allow actions that had never been tried from a state to be considered in planning (initial model was that such an action led back to the same state with a reward of 0)

When the Model is Wrong: Optimistic Model

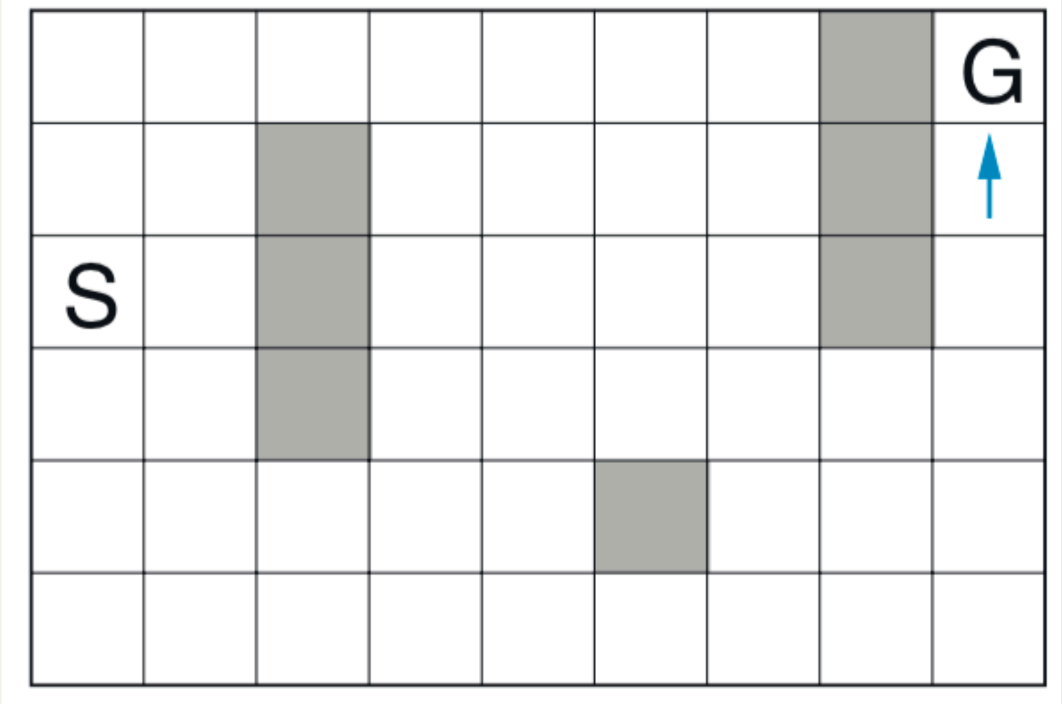


When the Model is Wrong: Pessimistic Model



Dyna-Q tries all state-action pairs uniformly.

Is there a better way?



Prioritized Sweeping (Det. Env.)

Initialize $Q(s, a)$ and $Model(s, a)$ for all a, s, $PQueue$ to empty
Loop forever:
1. $S \leftarrow$ current (nonterminal) state; 2. $A \leftarrow \text{policy}(S, Q)$
3. Take action A; observe resultant reward R and state S'
4. $\text{Model}(S, A) \leftarrow (R, S')$ (assumes deterministic envirnoment)
5. $P = R + \gamma \text{max}_a Q(S', a) - Q(S, A)$
6. If $P > \Theta$, insert (S, A) into $PQueue$ with priority P
7. Loop repeat n times while $PQueue$ is not empty
   a. $S, A = first(PQueue); R, S' \leftarrow \text{Model}(S, A)$
   b. $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \text{max}_a Q(S', a) - Q(S, A)]$
   c. Loop for all $\bar{S}, \bar{A}$ predicted to lead to S:
      i. $\bar{R} =$ pred. reward for $\bar{S}, \bar{A}, S$
      ii. $P = \hat{R} + \gamma \text{max}_a Q(S', a) - Q(\bar{S}, \bar{A})$
      iii. If $P > \Theta$, insert $(\bar{S}, \bar{A})$ into $PQueue$ with priority P