Finite Markov Decision Progresses
&
Dynamic Programming

Yan Yukun

Chapter 3

3.1 The Agent-Environment Interface

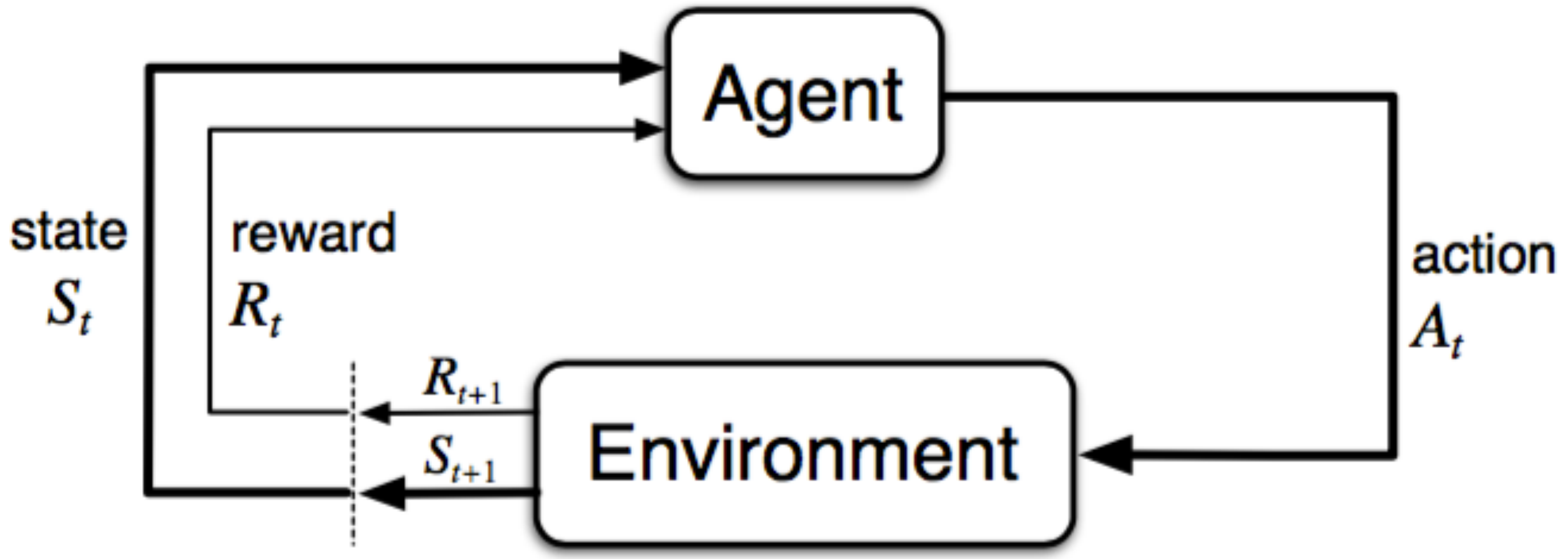


Figure 3.1: The agent–environment interaction in reinforcement learning.

agent: learner and decision-maker $\mathcal{A}(S_t)$

environment: provide states and rewards $S_t \in \mathcal{S}$ $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$

policy: a mapping from state to probabilities of selecting actions π_t,

3.1 The Agent-Environment Interface

Example 3.1: Bioreactor

state: thermocouple and other sensory readings

action: activate heating elements and motors

rewards: moment- by-moment measures of the rate at which the useful chemical is produced by the bioreactor

Example 3.2: Pick-and-Place Robot

state: thermocouple and other sensory readings

action: the latest readings of joint angles and velocities

rewards: +1 for each object successfully picked up and placed

3.1 The Agent-Environment Interface

Example 3.3: Recycling Robot

state: current charge level of the battery

action: search \ wait \ recharge

rewards: zero most of the time, but then become positive when the robot secures an empty can, or large and negative if the battery runs all the way down.

Exercise 3.1 Devise three example tasks of your own that fit into the reinforcement learning framework, identifying for each its states, actions, and rewards. Make the three examples as different from each other as possible. The framework is abstract and flexible and can be applied in many different ways. Stretch its limits in some way in at least one of your examples.

Chess game / Pingpang game / Question-Answering robot

*Exercise 3.2 Is the reinforcement learning framework adequate to usefully represent all goal-directed learning tasks? Can you think of any clear exceptions?

Exercise 3.3 Consider the problem of driving. You could define the actions in terms of the accelerator, steering wheel, and brake, that is, where your body meets the machine. Or you could define them farther out—say, where the rubber meets the road, considering your actions to be tire torques. Or you could define them farther in—say, where your brain meets your body, the actions being muscle twitches to control your limbs. Or you could go to a really high level and say that your actions are your choices of where to drive. **What is the right level, the right place to draw the line between agent and environment? On what basis is one location of the line to be preferred over another? Is there any fundamental reason for preferring one location over another, or is it a free choice?**

3.2 Goal & Reward

reward:received scalar signal

- the reward signal is not the place to impart to the agent prior knowledge about how to achieve what we want it to do.
- The reward signal is your way of communicating to the robot what you want it to achieve, not how you want it achieved.
- The rewards are computed in the environment rather than in the agent.

3.3 Returns

return: some specific function of the reward sequence

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T,$$

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \quad = \quad \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

3.3 Returns

Example 3.4: Pole-Balancing

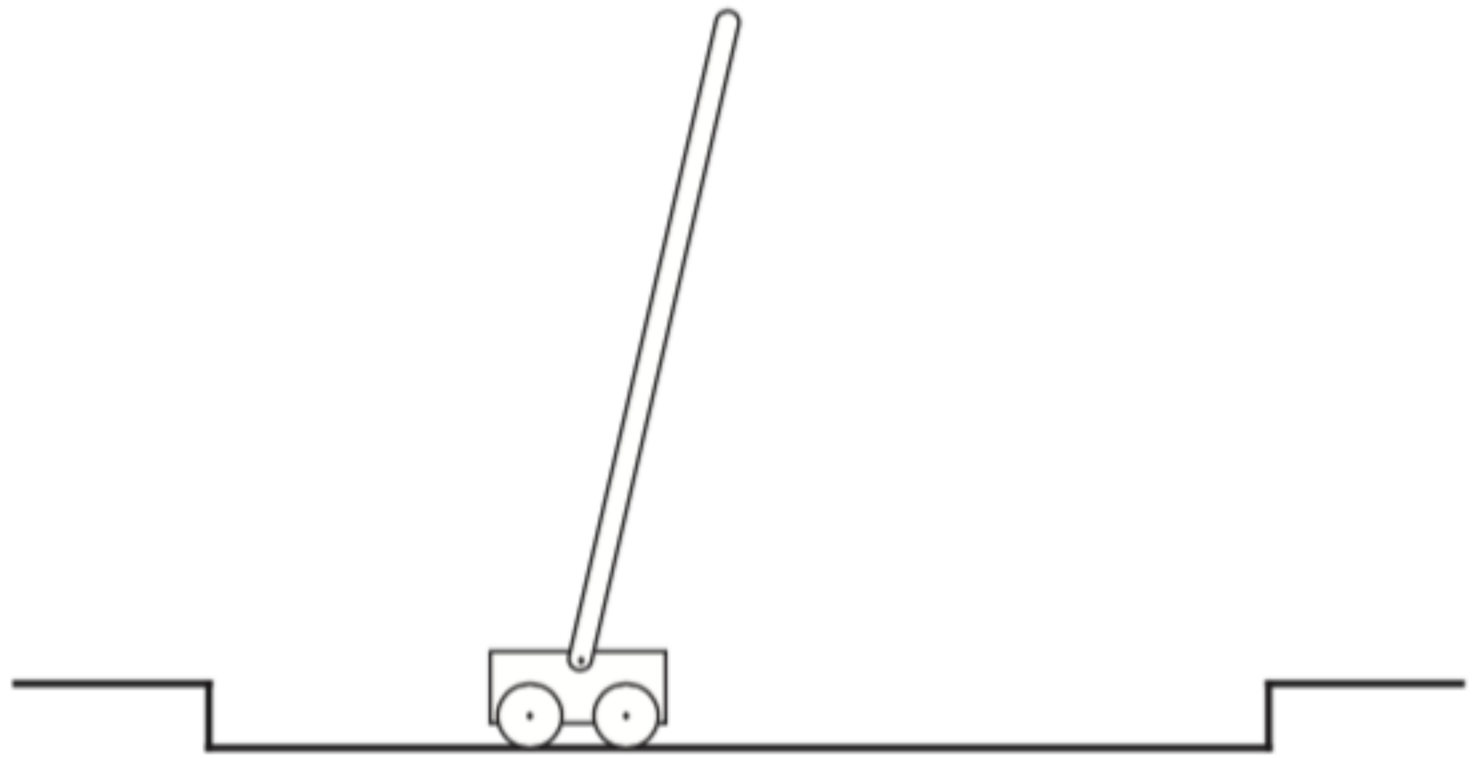
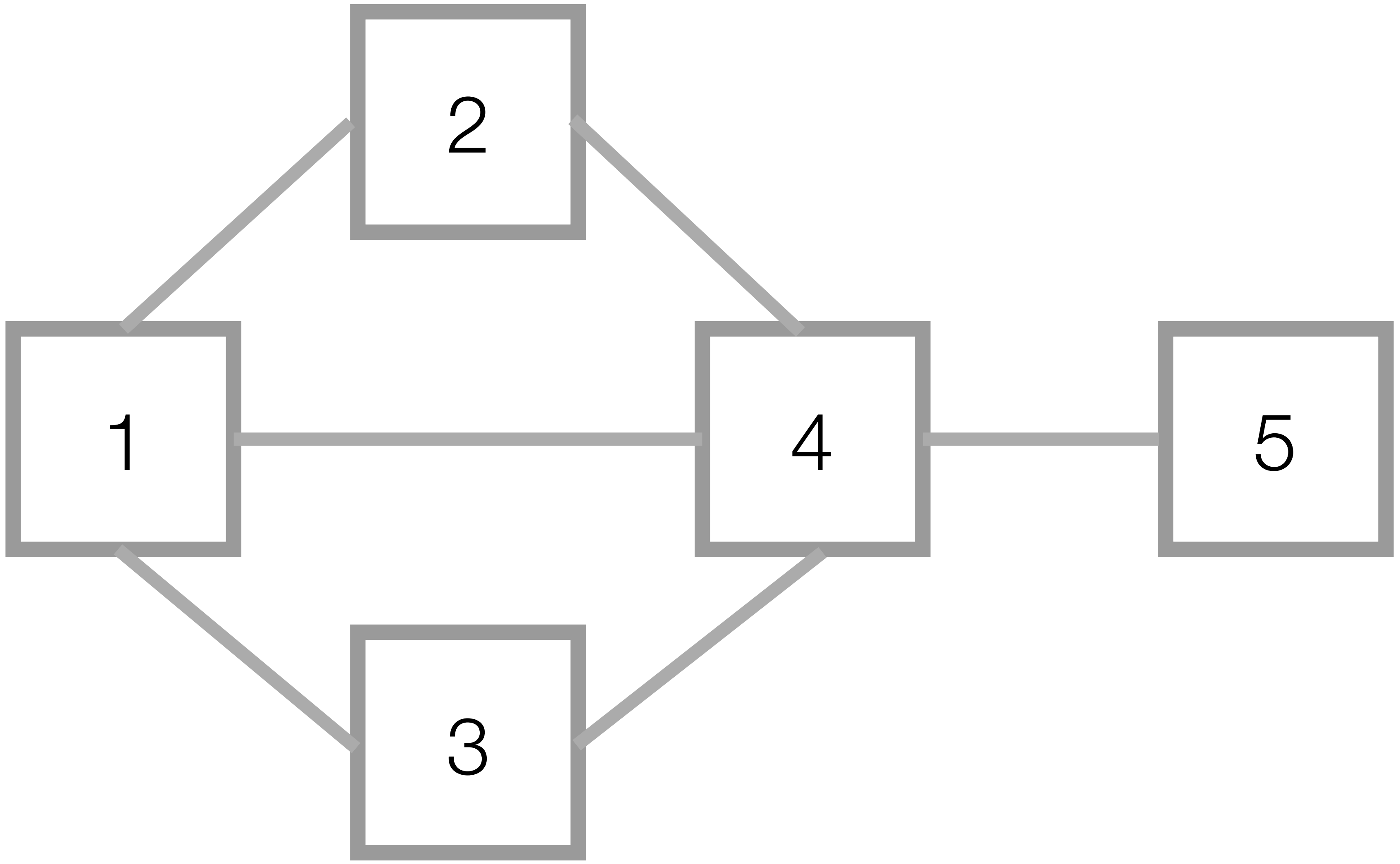


Figure 3.2: The pole-balancing task.

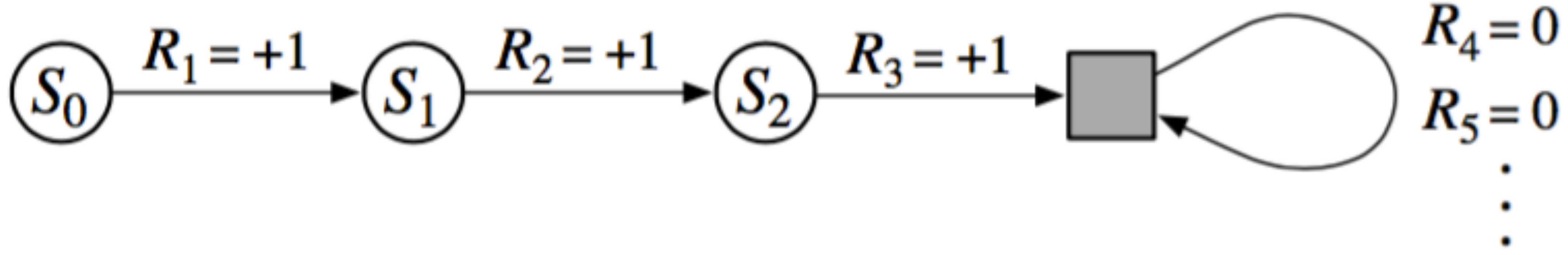
Exercise 3.4 Suppose you treated pole-balancing as an episodic task but also used discounting, with all rewards zero except for -1 upon failure. What then would the return be at each time? How does this return differ from that in the discounted, continuing formulation of this task?

Exercise 3.5 Imagine that you are designing a robot to run a maze. You decide to give it a reward of +1 for escaping from the maze and a reward of zero at all other times. The task seems to break down naturally into episodes—the successive runs through the maze—so you decide to treat it as an episodic task, where the goal is to maximize expected total reward (3.1). After running the learning agent for a while, you find that it is showing no improvement in escaping from the maze. What is going wrong? Have you effectively communicated to the agent what you want it to achieve?



state value update with Rt: if discounted, all state’s value will be the same:1

3.4 Unified Notation for Episodic and Continuing Tasks



$$G_t \doteq \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1},$$

3.4 The Markov Property

Markov property : A state signal that succeeds in retaining all relevant information is said to be Markov, or to have the Markov property

$$\Pr\{S_{t+1} = s', R_{t+1} = r \mid S_0, A_0, R_1, \ldots, S_{t-1}, A_{t-1}, R_t, S_t, A_t\},$$

$$p(s', r|s, a) \doteq \Pr\{S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a\},$$

one-step dynamics

3.4 The Markov Property

the assumption of Markov state representations

Example 3.5: Pole-Balancing State

state = the position and velocity of the cart along the track, the angle between the cart and the pole, and the rate at which this angle is changing (the angular velocity) ?

other effects ——- non-makos state

3.4 The Markov Property

the assumption of Markov state representations

Example 3.6: Draw Poker

in practice this is far too much to remember and analyze, and most of it will have no clear effect on one’s predictions and decisions.

Exercise 3.6: Broken Vision System: Imagine that you are a vision system. When you are first turned on for the day, an image floods into your camera. You can see lots of things, but not all things. You can't see objects that are occluded, and of course you can't see objects that are behind you. After seeing that first scene, do you have access to the Markov state of the environment? Suppose your camera was broken that day and you received no images at all, all day. Would you have access to the Markov state then?

3.6 Markov Decision Processes

Markov decision process : A reinforcement learning task that satisfies the Markov property is called a Markov decision process, or MDP.

A particular finite MDP is defined by its state and action sets and by the one-step dynamics of the environment.

$$p(s', r|s, a) \doteq \Pr\{S_{t+1}=s', R_{t+1}=r \mid S_t=s, A_t=a\}.$$

expected rewards for state–action pairs

$$r(s, a) \doteq \mathbb{E}[R_{t+1} \mid S_t=s, A_t=a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r|s, a),$$

3.6 Markov Decision Processes

state-transition probabilities

$$p(s'|s,a) \doteq \Pr\{S_{t+1}=s' \mid S_t=s, A_t=a\} = \sum_{r\in\mathcal{R}} p(s',r|s,a),$$

state–action–next-state triples

$$r(s,a,s') \doteq \mathbb{E}\left[R_{t+1} \mid S_t=s, A_t=a, S_{t+1}=s'\right] = \frac{\sum_{r\in\mathcal{R}} r\, p(s',r|s,a)}{p(s'|s,a)}.$$

3.6 Markov Decision Processes

Example 3.7: Recycling Robot MDP

$$\begin{aligned}\mathcal{A}(\texttt{high}) &\doteq \{\texttt{search}, \texttt{wait}\}\\ \mathcal{A}(\texttt{low}) &\doteq \{\texttt{search}, \texttt{wait}, \texttt{recharge}\}.\end{aligned}$$

s	s'	a	$p(s' \mid s,a)$	$r(s,a,s')$
high	high	search	α	$r_{\texttt{search}}$
high	low	search	$1-\alpha$	$r_{\texttt{search}}$
low	high	search	$1-\beta$	-3
low	low	search	β	$r_{\texttt{search}}$
high	high	wait	1	$r_{\texttt{wait}}$
high	low	wait	0	$r_{\texttt{wait}}$
low	high	wait	0	$r_{\texttt{wait}}$
low	low	wait	1	$r_{\texttt{wait}}$
low	high	recharge	1	0
low	low	recharge	0	0.

3.6 Markov Decision Processes

Example 3.7: Recycling Robot MDP

transition graph   state nodes and action nodes

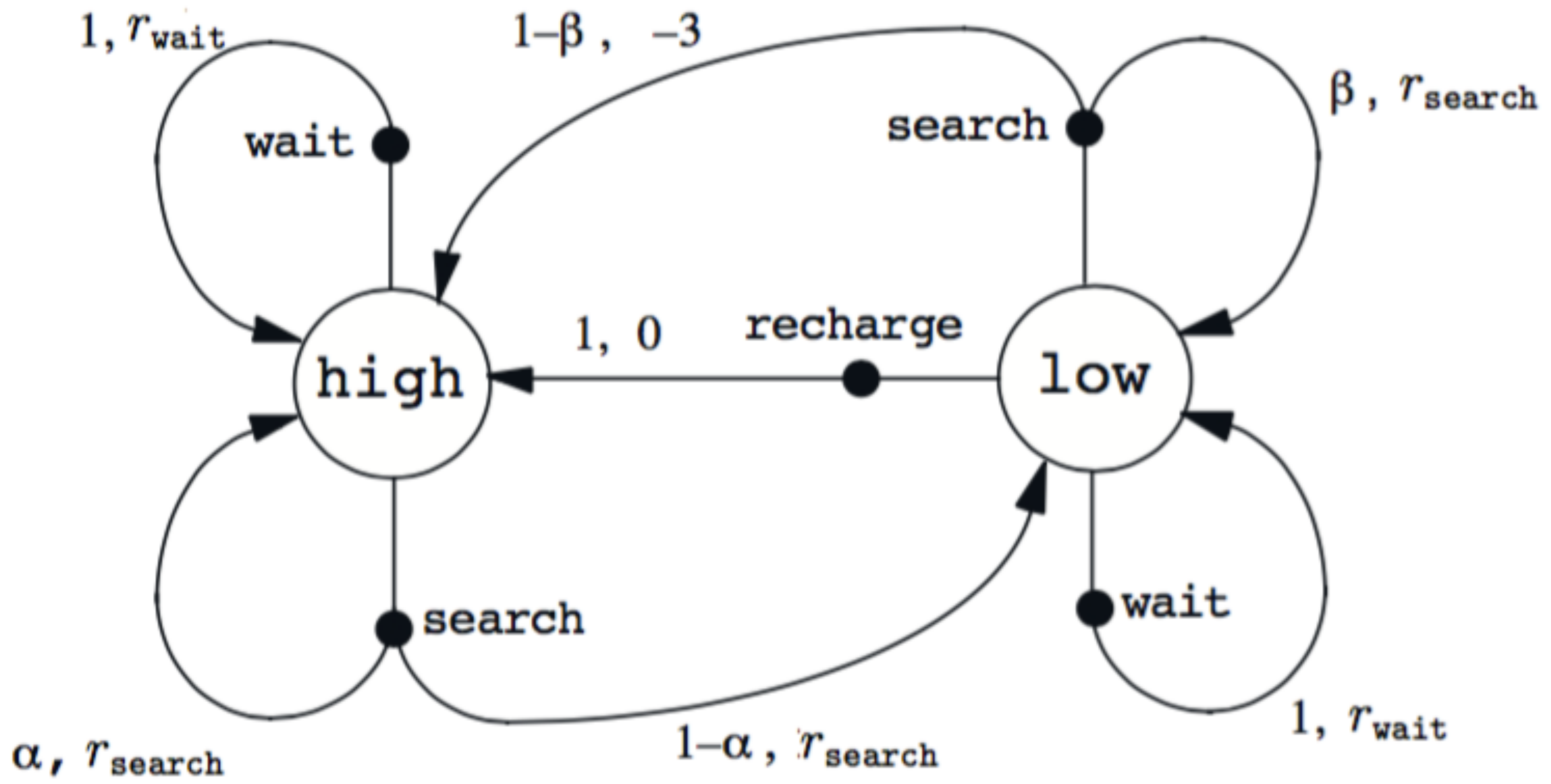


Figure 3.3: Transition graph for the recycling robot example.

3.7 Value Functions

Value Functions: "how good it is for the agent to be in a given state or to take a given action" (**expected return**)
defined with respect to particular policies

state-value function for policy

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right],$$

action-value function for policy

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right].$$

3.7 Value Functions

The value functions v_* and q_* can be estimated from experience.

Monte Carlo methods

or use a parameterized function approximator

3.7 Value Functions

they satisfy particular recursive relationships. **Bellman equation** for v\pi

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right] \\
&= \mathbb{E}_\pi\left[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \,\middle|\, S_t = s\right] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \left[r + \gamma \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \,\middle|\, S_{t+1} = s'\right]\right] \\
&= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\Big[r + \gamma v_\pi(s')\Big], \qquad (3.12)
\end{aligned}
$$

It states that the value of the start state must equal the (discounted) value of the expected next state, plus the reward expected along the way.

3.7 Value Functions

backup diagrams

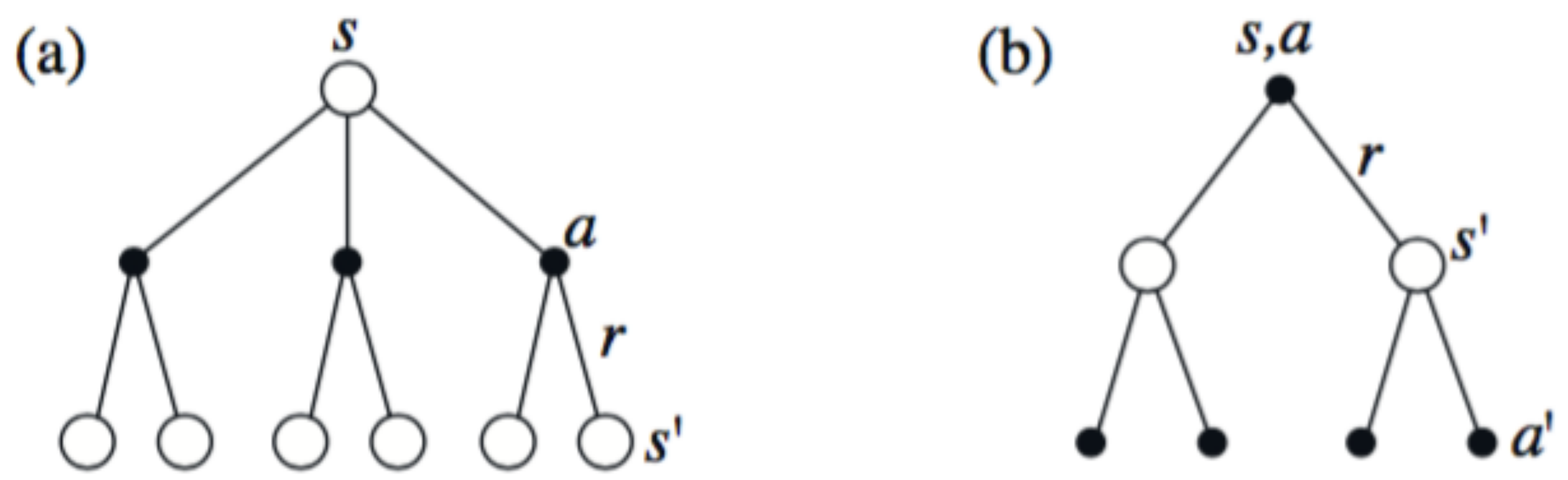
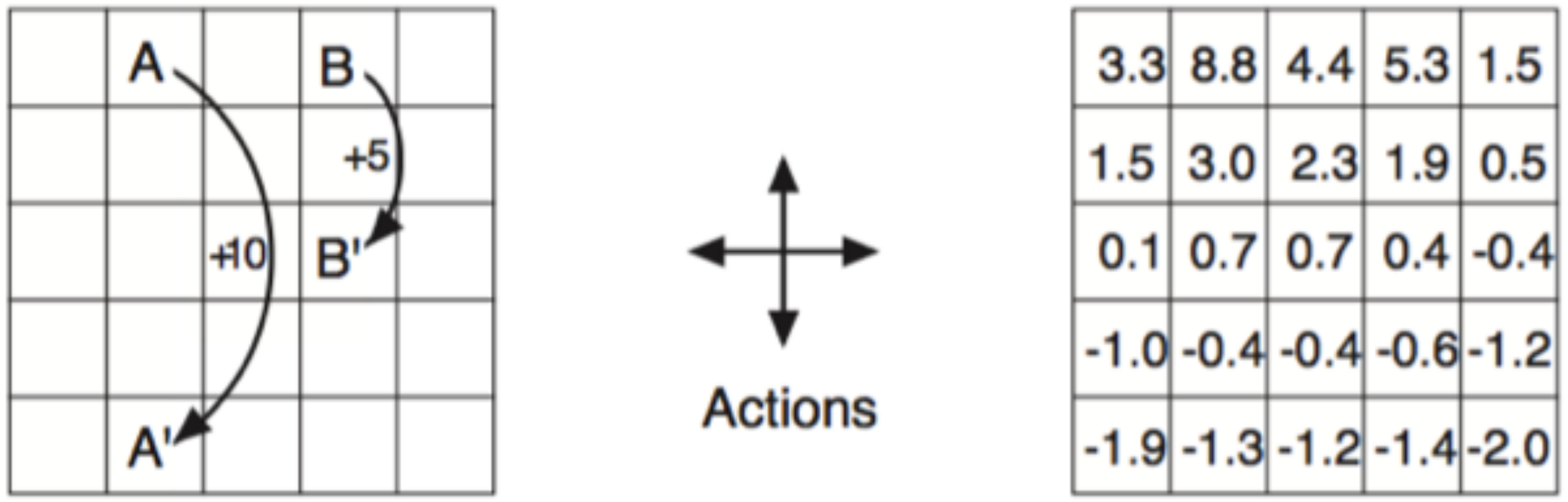


Figure 3.4: Backup diagrams for (a) v_π and (b) q_π.

3.7 Value Functions

Example 3.8: Gridworld

reward = 0/-1



3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

$\gamma = 0.9.$

Figure 3.5: Grid example: exceptional reward dynamics (left) and state-value function for the equiprobable random policy (right).

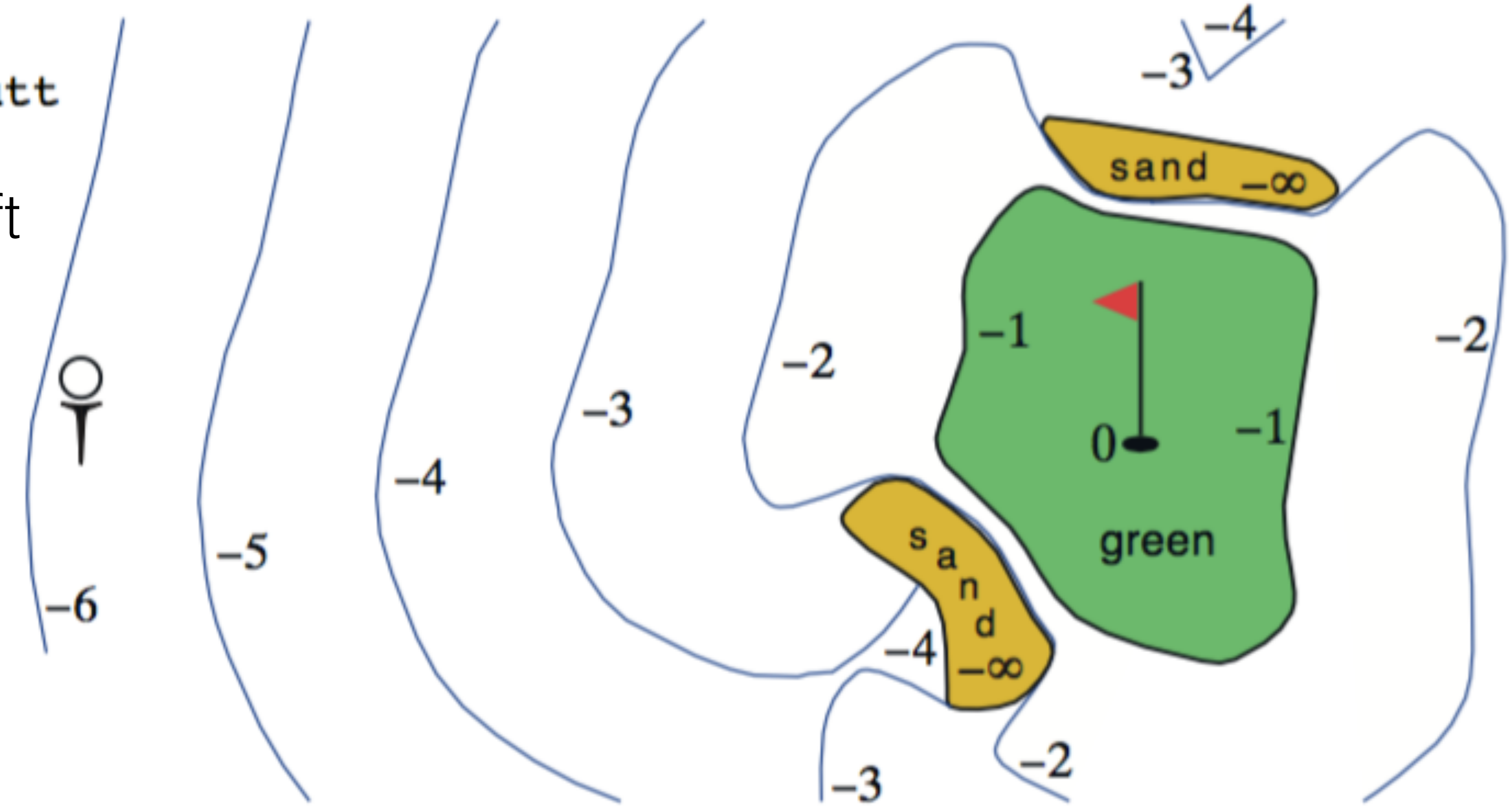
3.7 Value Functions

Example 3.9: Golf

actions:putt/driver

values:how many hits left

v_{putt}



Exercise 3.7 What is the Bellman equation for action values, that is, for q_π It must give the action value $q_\pi(s,a)$ in terms of the action values, $q_\pi(s',a')$, of possible successors to the state–action pair (s,a). As a hint, the backup diagram corresponding to this equation is given in Figure 3.4b. analogous to (3.12), but for action values.

Exercise 3.8 The Bellman equation (3.12) must hold for each state for the value function $v_{⇡}$ shown in Figure 3.5b. As an example, show numerically that this equation holds for the center state, valued at +0.7, with respect to its four neighboring states, valued at +2.3, +0.4, 0.4, and +0.7. (These numbers are accurate only to one decimal place.)

Exercise 3.9 In the gridworld example, rewards are positive for goals, negative for running into the edge of the world, and zero the rest of the time. Are the signs of these rewards important, or only the intervals between them? Prove, using (3.2), that adding a constant c to all the rewards adds a constant, v_c, to the values of all states, and thus does not affect the relative values of any states under any policies. What is v_c in terms of gamma ?

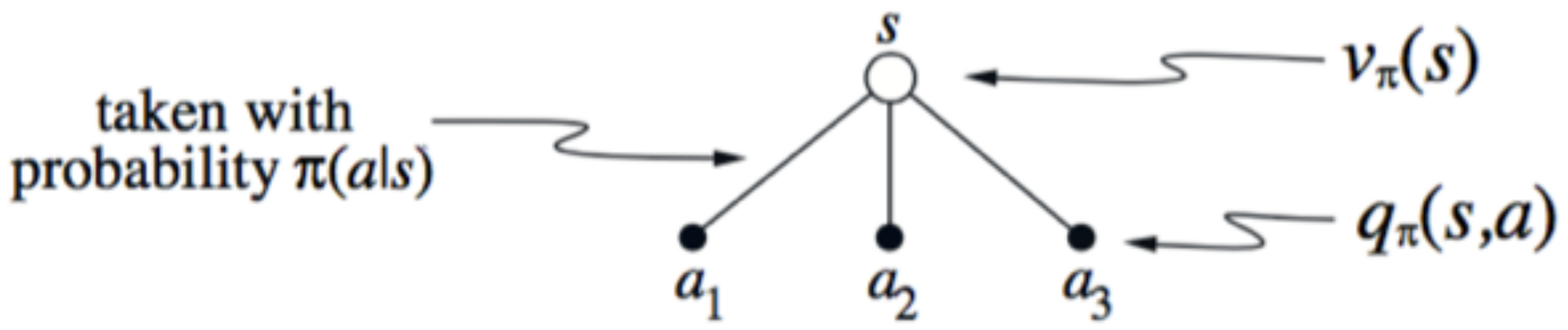
$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

Exercise 3.10 Now consider adding a constant c to all the rewards in an episodic task, such as maze running. Would this have any effect, or would it leave the task unchanged as in the continuing task above? Why or why not? Give an example.

have an effect that…

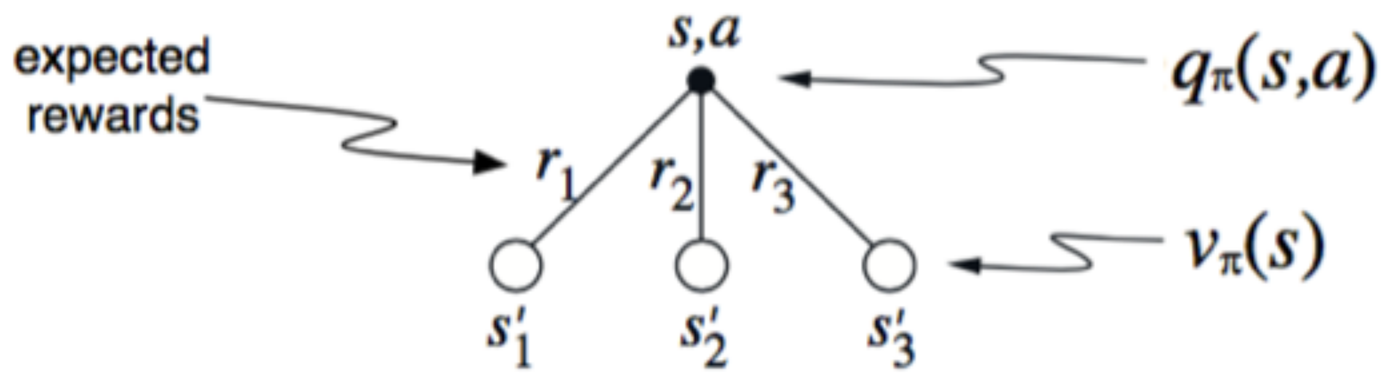
the maze task

Exercise 3.11 The value of a state depends on the the values of the actions possible in that state and on how likely each action is to be taken under the current policy. We can think of this in terms of a small backup diagram rooted at the state and considering each possible action:



Give the equation corresponding to this intuition and diagram for the value at the root node, $v_\pi(s)$, in terms of the value at the expected leaf node, $q_\pi(s, a)$, given $S_t = s$. This expectation depends on the policy, π. Then give a second equation in which the expected value is written out explicitly in terms of $\pi(a|s)$ such that no expected value notation appears in the equation.

Exercise 3.12 The value of an action, $q_\pi(s,a)$, depends on the expected next reward and the expected sum of the remaining rewards. Again we can think of this in terms of a small backup diagram, this one rooted at an action (state–action pair) and branching to the possible next states:



Give the equation corresponding to this intuition and diagram for the action value, $q_\pi(s,a)$, in terms of the expected next reward, R_{t+1}, and the expected next state value, $v_\pi(S_{t+1})$, given that $S_t = s$ and $A_t = a$. Then give a second equation, writing out the expected value explicitly in terms of $p(s', r|s,a)$ defined by (3.6), such that no expected value notation appears in the equation.

3.8 Optimal Value Functions

A policy **\pi** is defined to be better than or equal to a policy **\pi'** if its expected return is greater than or equal to that of **\pi'** for **all states**.

There is always at least one policy that is better than or equal to all other policies.

optimal state-value function

$$v_*(s) \doteq \max_\pi v_\pi(s),$$

optimal action-value function,

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a),$$

$$q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] .$$

3.8 Optimal Value Functions

A policy **\pi** is defined to be better than or equal to a policy **\pi'** if its expected return is greater than or equal to that of **\pi'** for **all states**.

There is always at least one policy that is better than or equal to all other policies.

optimal state-value function

$$v_*(s) \doteq \max_\pi v_\pi(s),$$

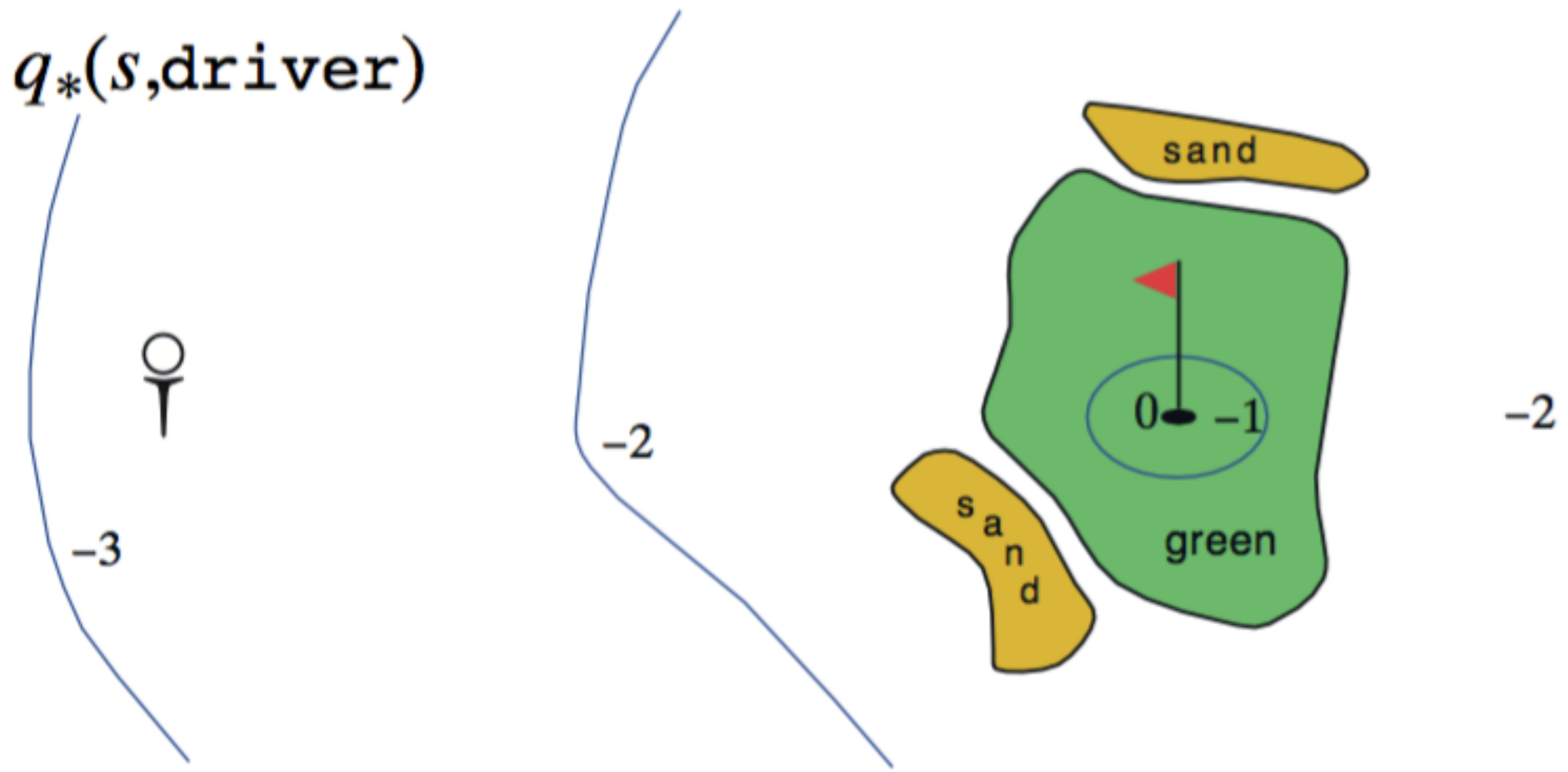
optimal action-value function,

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a),$$

$$q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \, .$$

3.8 Optimal Value Functions

Example 3.10: Optimal Value Functions for Golf



3.8 Optimal Value Functions

Bellman optimality equation.

$$
\begin{aligned}
v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\
&= \max_a \mathbb{E}_{\pi*}[G_t \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}_{\pi*}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right] \\
&= \max_a \mathbb{E}_{\pi*}\left[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \,\middle|\, S_t = s, A_t = a\right] \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \max_{a \in \mathcal{A}(s)} \sum_{s',r} p(s', r | s, a) \big[r + \gamma v_*(s')\big].
\end{aligned}
$$

3.8 Optimal Value Functions

Bellman optimality equation.

$$\begin{aligned} q_*(s,a) &= \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \,\Big|\, S_t = s, A_t = a\Big] \\ &= \sum_{s',r} p(s', r|s,a)\Big[r + \gamma \max_{a'} q_*(s', a')\Big]. \end{aligned}$$

N equations in N unknowns

once the optimal value functions are solved, any policy that is greedy with respect to the optimal evaluation function vı← is an optimal policy. a one-step-ahead search yields the long-term optimal actions.

3.8 Optimal Value Functions

Bellman optimality equation.

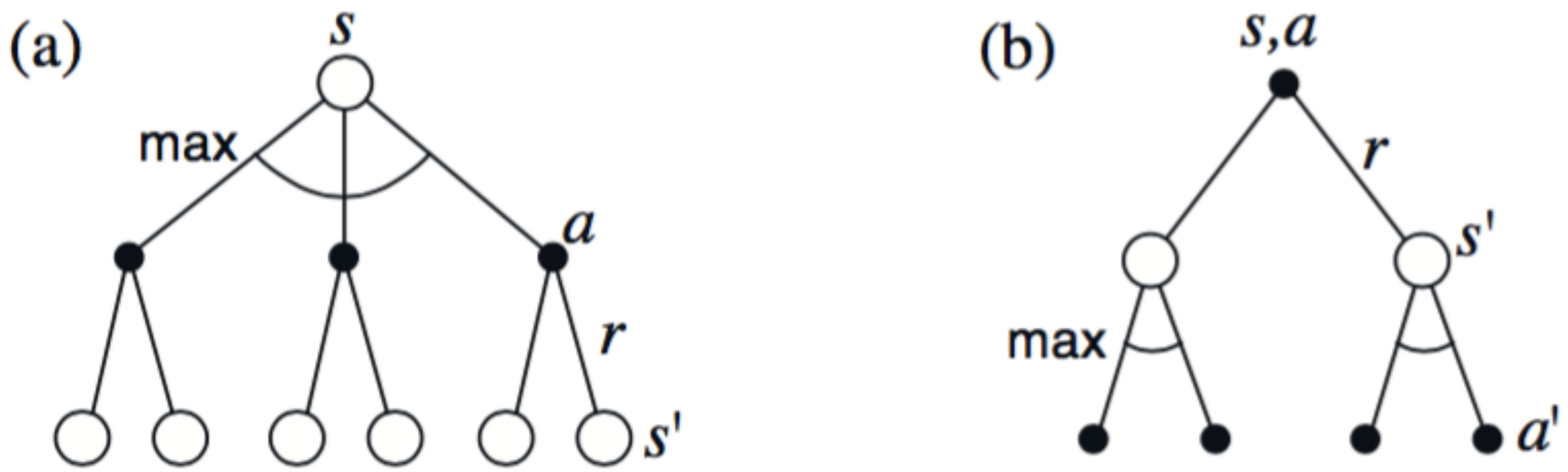


Figure 3.7: Backup diagrams for (a) v_* and (b) q_*

it is even better knowing q*

3.8 Optimal Value Functions

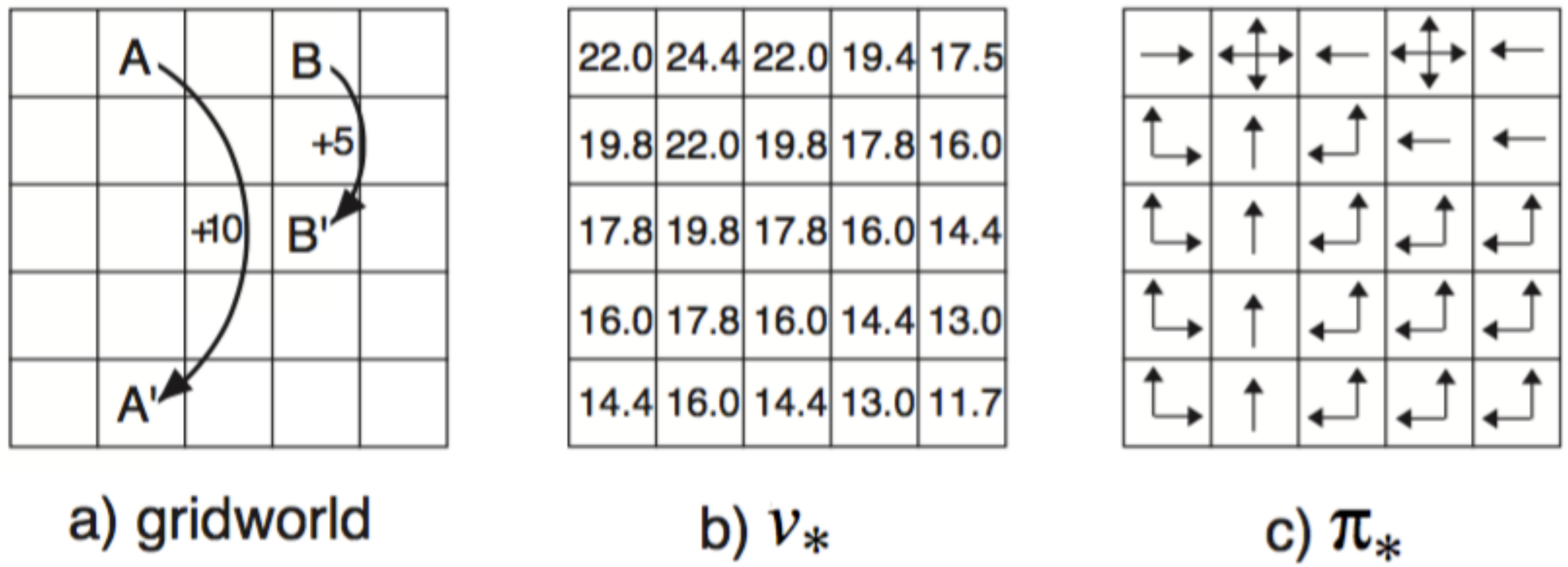
Example 3.11: Bellman Optimality Equations for the Recycling Robot

$$\begin{aligned}
v_*(\mathtt{h}) &= \max\left\{\begin{array}{l} p(\mathtt{h}|\mathtt{h},\mathtt{s})[r(\mathtt{h},\mathtt{s},\mathtt{h}) + \gamma v_*(\mathtt{h})] + p(\mathtt{l}|\mathtt{h},\mathtt{s})[r(\mathtt{h},\mathtt{s},\mathtt{l}) + \gamma v_*(\mathtt{l})], \\ p(\mathtt{h}|\mathtt{h},\mathtt{w})[r(\mathtt{h},\mathtt{w},\mathtt{h}) + \gamma v_*(\mathtt{h})] + p(\mathtt{l}|\mathtt{h},\mathtt{w})[r(\mathtt{h},\mathtt{w},\mathtt{l}) + \gamma v_*(\mathtt{l})] \end{array}\right\} \\
&= \max\left\{\begin{array}{l} \alpha[r_\mathtt{s} + \gamma v_*(\mathtt{h})] + (1-\alpha)[r_\mathtt{s} + \gamma v_*(\mathtt{l})], \\ 1[r_\mathtt{w} + \gamma v_*(\mathtt{h})] + 0[r_\mathtt{w} + \gamma v_*(\mathtt{l})] \end{array}\right\} \\
&= \max\left\{\begin{array}{l} r_\mathtt{s} + \gamma[\alpha v_*(\mathtt{h}) + (1-\alpha) v_*(\mathtt{l})], \\ r_\mathtt{w} + \gamma v_*(\mathtt{h}) \end{array}\right\}.
\end{aligned}$$

$$v_*(\mathtt{l}) = \max\left\{\begin{array}{l} \beta r_\mathtt{s} - 3(1-\beta) + \gamma[(1-\beta)v_*(\mathtt{h}) + \beta v_*(\mathtt{l})] \\ r_\mathtt{w} + \gamma v_*(\mathtt{l}), \\ \gamma v_*(\mathtt{h}) \end{array}\right\}.$$

3.8 Optimal Value Functions

Example 3.12: Solving the Gridworld



22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

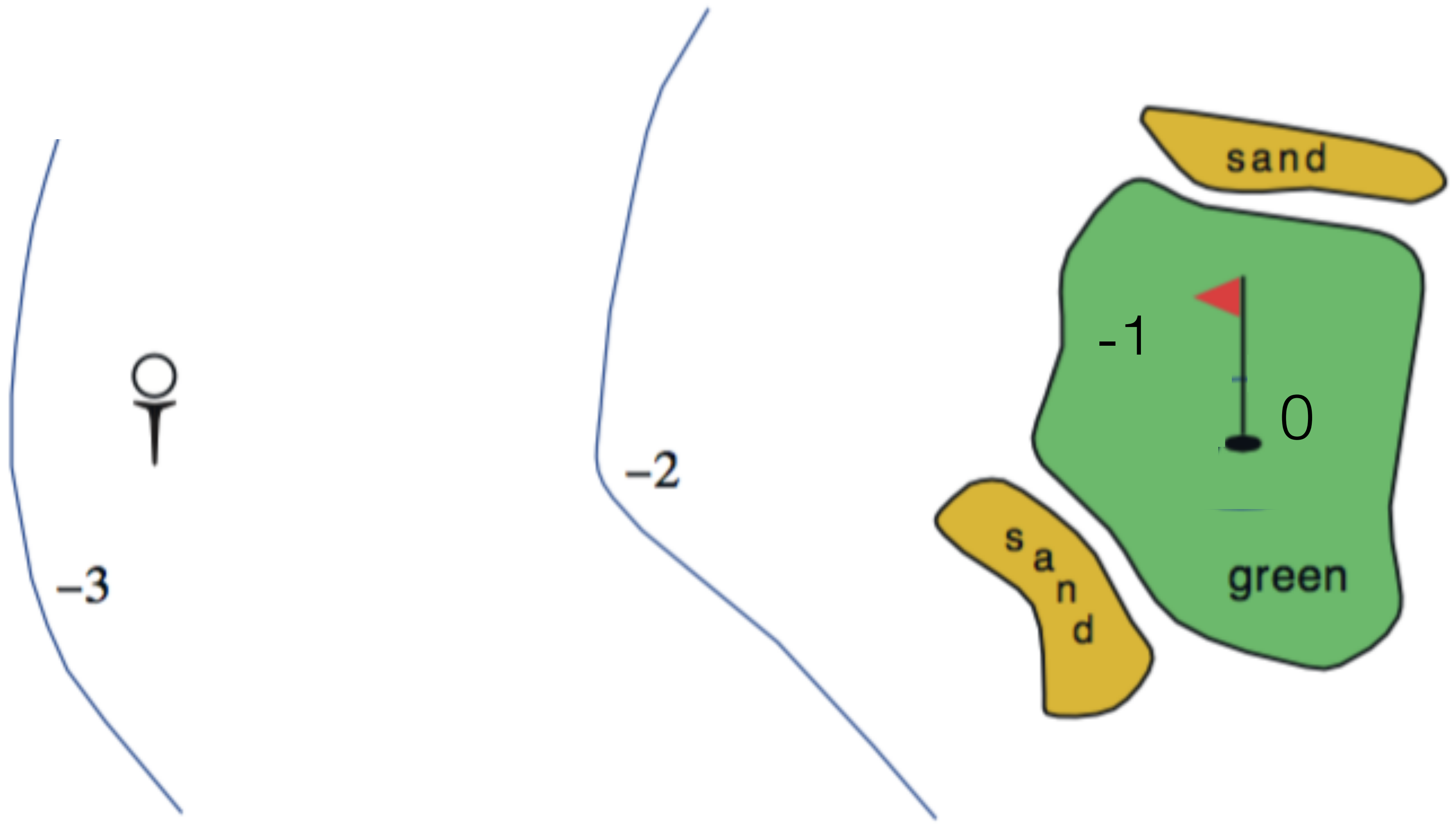
a) gridworld b) v_* c) π_*

Figure 3.8: Optimal solutions to the gridworld example.

3.8 Optimal Value Functions

Explicitly solving the Bellman optimality equation relies on at least three assumptions that are rarely true in practice: (1) we accurately know the dynamics of the environment; (2) we have enough computational resources to complete the computation of the solution; and (3) the Markov property.

Exercise 3.13 Draw or describe the optimal state-value function for the golf example.



optimal policy

Exercise 3.14 Draw or describe the contours of the optimal action-value function for putting, $q_*(s,\text{putter})$, for the golf example.

Exercise 3.15 Give the Bellman equation for q_* for the recycling robot.

Chapter 4

4.1 Policy Evaluation

policy evaluation: We compute the state-value function to evaluate a policy

The existence and uniqueness of v$_{\backslash pi}$ are guaranteed as long as either \gamma <
1 or eventual termination is guaranteed from all states under the policy

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s\right] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_\pi(s')\Big], \end{aligned}$$

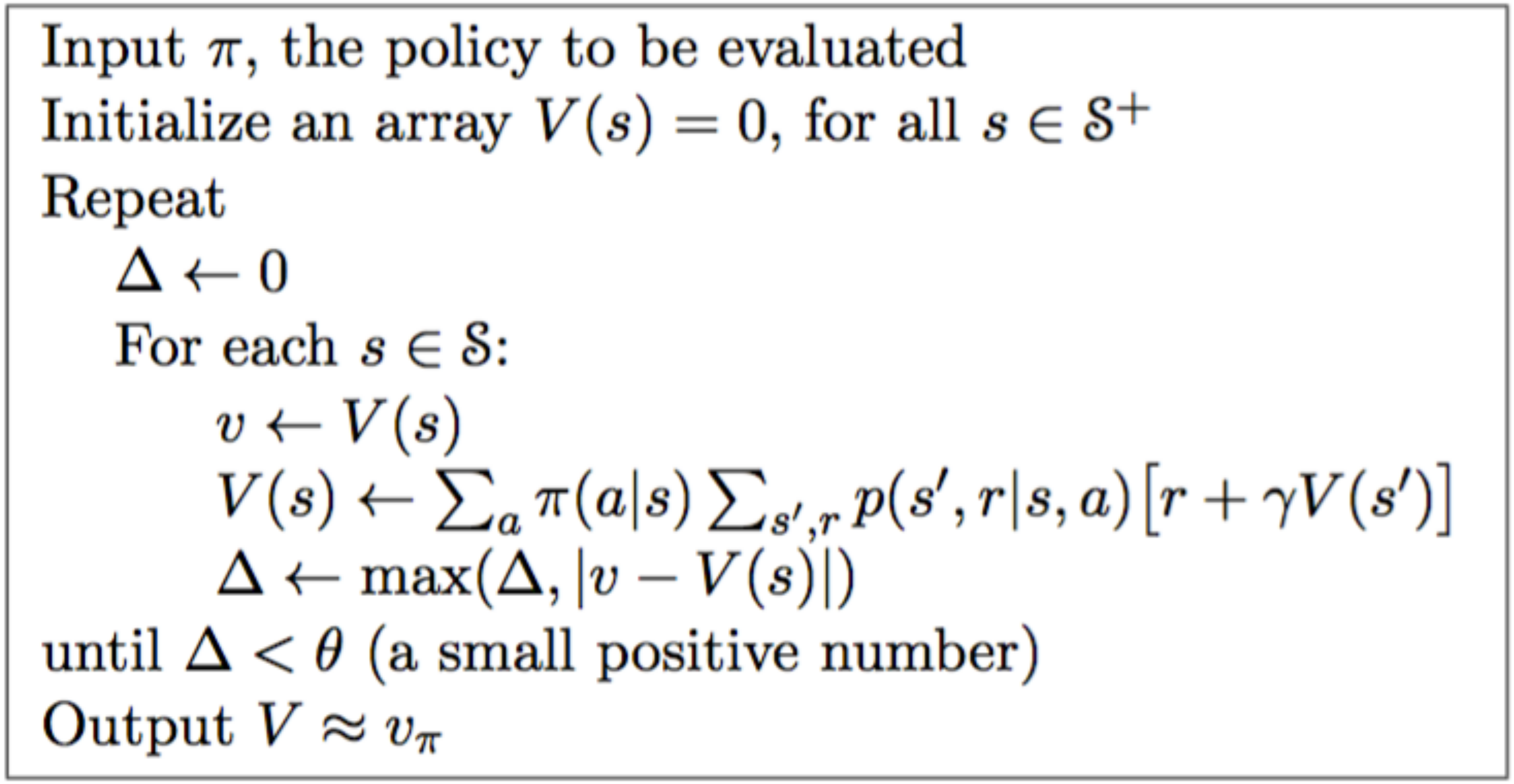
iterative policy evaluation.
full backup. Each iteration of iterative policy evaluation backs up the value of every state once to produce the new approximate value function v$_{k+1}$.

$$\begin{aligned} v_{k+1}(s) &\doteq \mathbb{E}_\pi[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_k(s')\Big], \end{aligned}$$

4.1 Policy Evaluation

two array VS “in place”

sweep(through the state space)

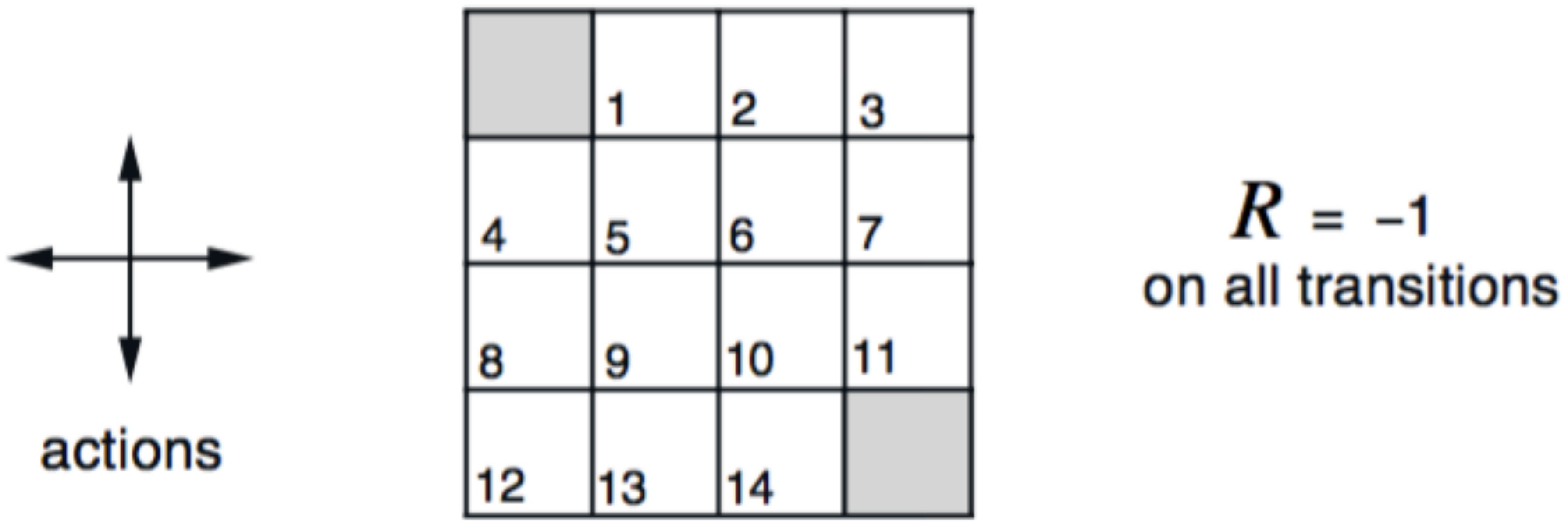


```
Input $\pi$, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$
```

Figure 4.1: Iterative policy evaluation.

4.1 Policy Evaluation

Example 4.1 Consider the 4→I4 gridworld shown below.



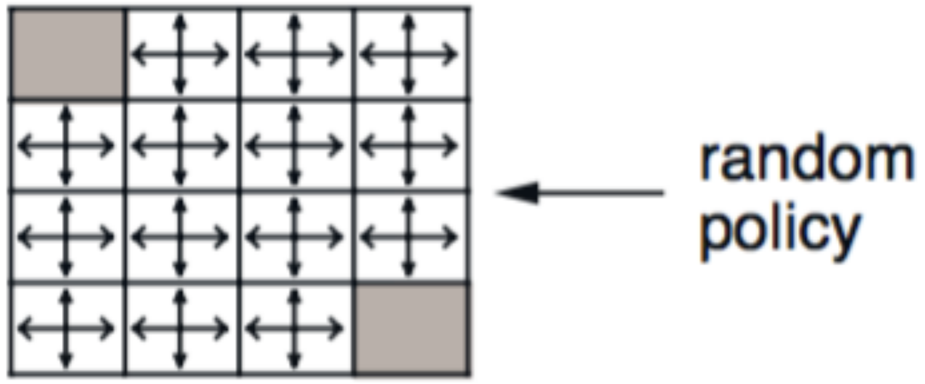
4.1 Policy Evaluation

v_k for the Random Policy

Greedy Policy w.r.t. v_k

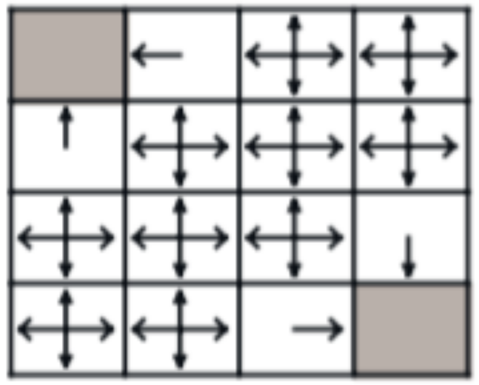
$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0



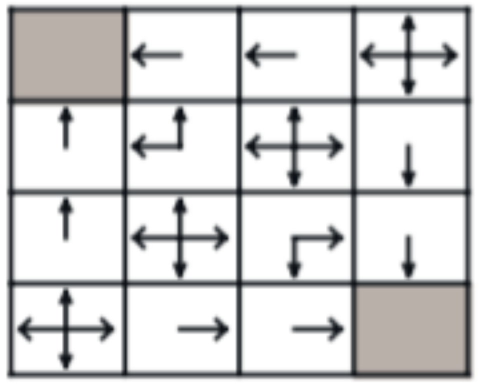
$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

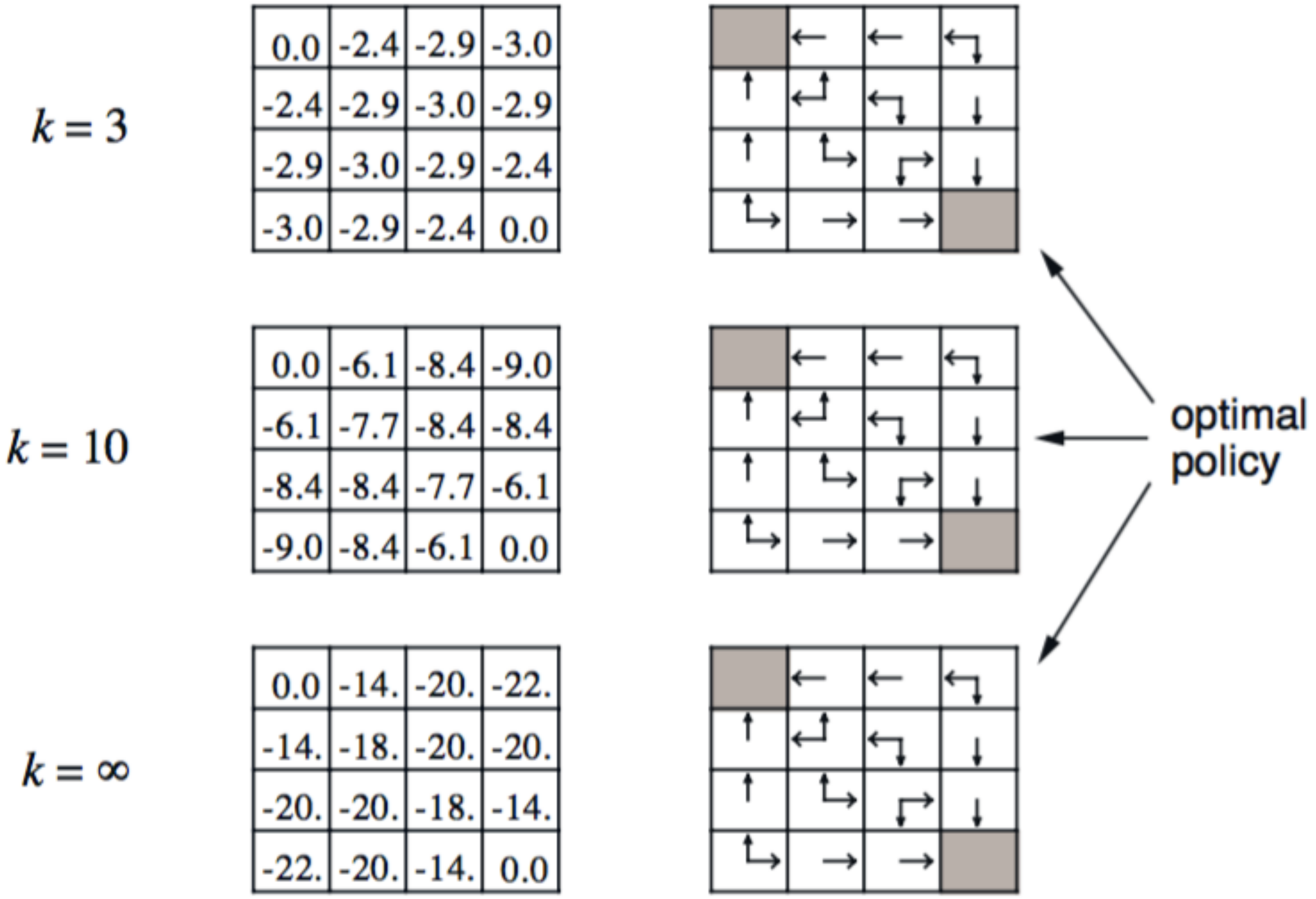


$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0



4.1 Policy Evaluation



$k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

Exercise 4.1 In Example 4.1, if **\pi** is the equiprobable random policy, what is q_{π}(11,down)? What is q_{π}(7,down)?

Exercise 4.2 In Example 4.1, suppose a new state 15 is added to the gridworld just below state 13, and its actions, left, up, right, and down, take the agent to states 12, 13, 14, and 15, respectively. Assume that the transitions from the original states are unchanged. What, then, is v_{π}(15) for the equiprobable random policy? Now suppose the dynamics of state 13 are also changed, such that action down from state 13 takes the agent to the new state 15. What is v_{π}(15) for the equiprobable random policy in this case?

1\ no state can take any actions to reach 15 we only need to compute v(15)

2\ iterate?

Exercise 4.3 What are the equations analogous to (4.3), (4.4), and (4.5) for the action-value function q_π and its successive approximation by a sequence of functions $q_0, q_1, q_2, \ldots$?

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi\big[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s\big] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\Big[r + \gamma v_\pi(s')\Big],
\end{aligned}
$$

Exercise 4.4 In some undiscounted episodic tasks there may be policies for which eventual termination is not guaranteed. For example, in the grid problem above it is possible to go back and forth between two states forever. In a task that is otherwise perfectly sensible, $v_{\pi}(s)$ may be negative infinity for some policies and states, in which case the algorithm for iterative policy evaluation given in Figure 4.1 will not terminate. As a purely practical matter, how might we amend this algorithm to as- sure termination even in this case? Assume that eventual termination is guaranteed

4.2 Policy Improvement

consider selecting a in s and thereafter following the existing policy, \pi.

$$
\begin{aligned}
q_\pi(s,a) &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_\pi(s')\Big].
\end{aligned}
$$

policy improvement theorem

two **deterministic** policies, for all s

$$q_\pi(s,\pi'(s)) \geq v_\pi(s). \quad \xrightarrow{\qquad\qquad} \quad v_{\pi'}(s) \geq v_\pi(s).$$

4.2 Policy Improvement

consider selecting a in s and thereafter following the existing policy, \pi.

$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \sum_{s',r} p(s', r|s,a)\Big[r + \gamma v_\pi(s')\Big]. \end{aligned}$$

policy improvement theorem

two **deterministic** policies, for all s

$$q_\pi(s, \pi'(s)) \geq v_\pi(s). \qquad \longrightarrow \qquad v_{\pi'}(s) \geq v_\pi(s).$$

a slightly changed policy is better than the original one

4.2 Policy Improvement

$$\begin{aligned}
v_\pi(s) &\le q_\pi(s, \pi'(s)) \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \\
&\le \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) \mid S_t = s\right] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma \mathbb{E}_{\pi'}[R_{t+2} + \gamma v_\pi(S_{t+2})] \mid S_t = s] \\
&= \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2}) \mid S_t = s\right] \\
&\le \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 v_\pi(S_{t+3}) \mid S_t = s\right] \\
&\vdots \\
&\le \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \mid S_t = s\right] \\
&= v_{\pi'}(s).
\end{aligned}$$

4.2 Policy Improvement

new greedy policy

$$\begin{aligned}
\pi'(s) &\doteq \arg\max_a q_\pi(s,a) \\
&= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \arg\max_a \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_\pi(s')\Big],
\end{aligned}$$

4.2 Policy Improvement

suppose the two policy are the same good

$$\begin{aligned}
\pi'(s) &\doteq \arg\max_a q_\pi(s,a) \\
&= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \arg\max_a \sum_{s',r} p(s',r|s,a)\Big[r + \gamma \boxed{v_\pi}(s')\Big],
\end{aligned}$$

$$\begin{aligned}
v_{\pi'}(s) &= \max_a \mathbb{E}[R_{t+1} + \gamma v_{\pi'}(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \max_a \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_{\boxed{\pi'}}(s')\Big].
\end{aligned}$$

therefore, $v_{\pi'}$ must be v_*, and both π and π' must be optimal policies.

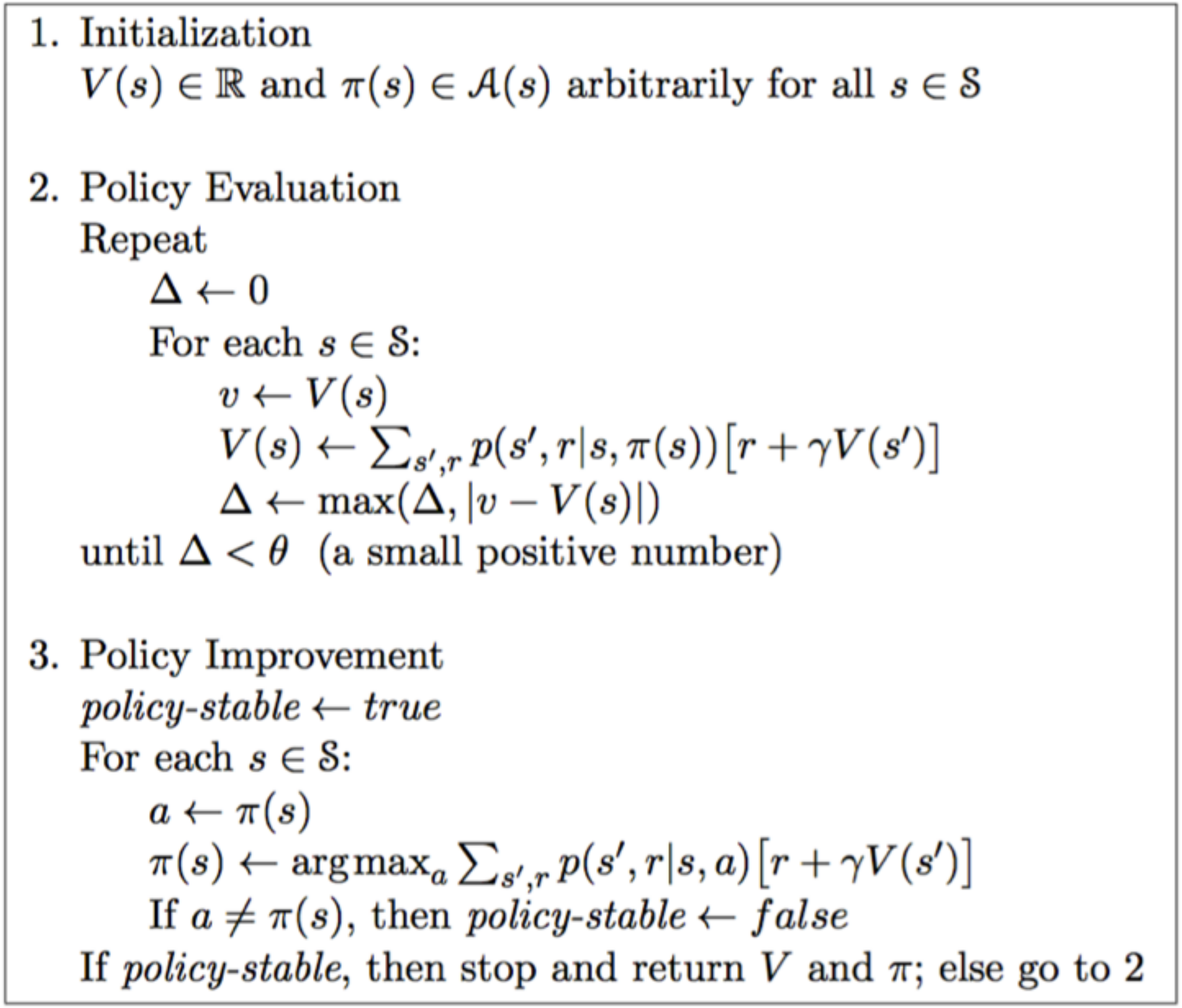
converge!

4.3 Policy Policy Iteration

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*,$$

This way of finding an optimal policy is called **policy iteration**

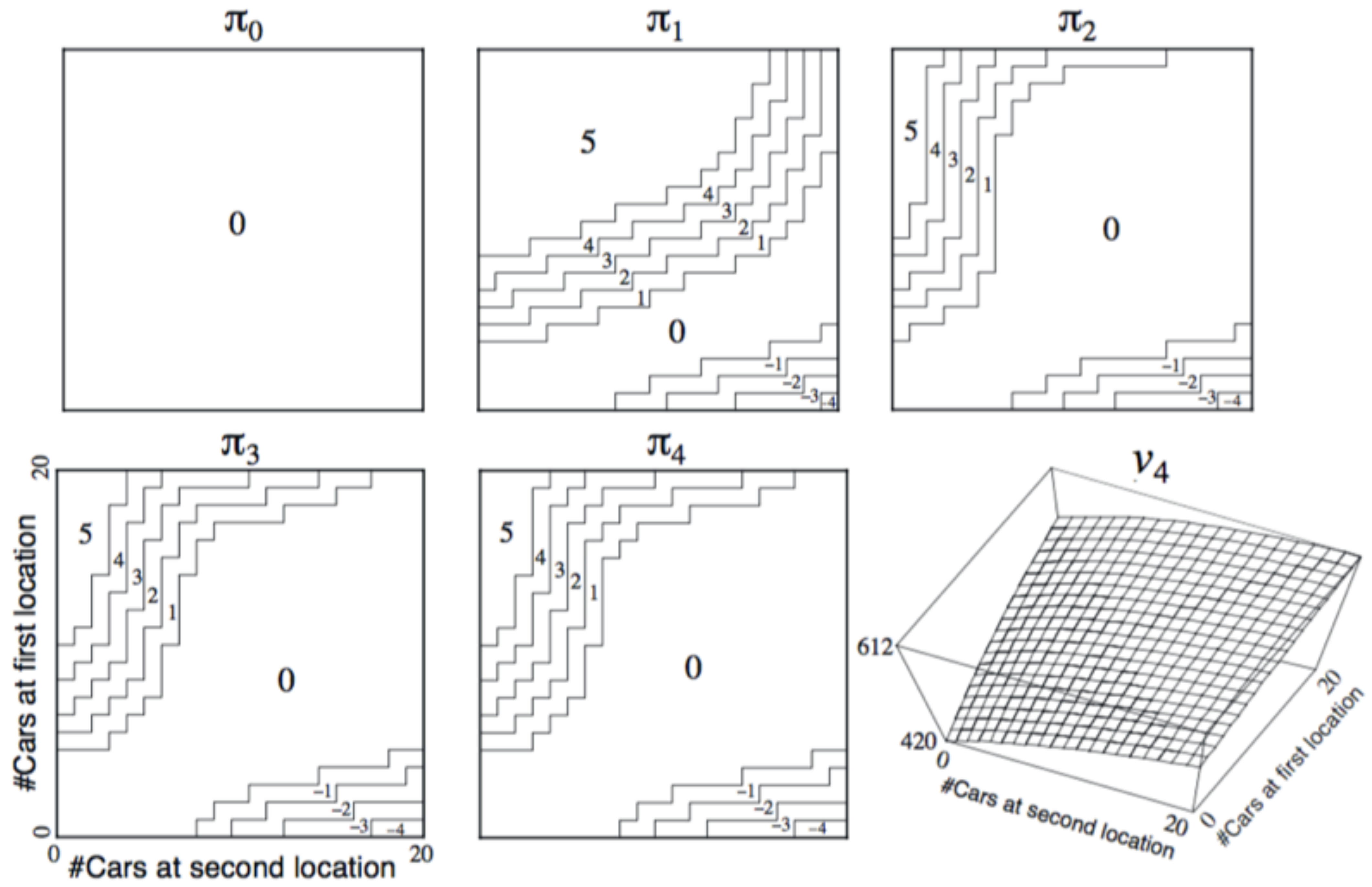
4.3 Policy Policy Iteration



```
1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
       $\Delta \leftarrow 0$
       For each $s \in \mathcal{S}$:
           $v \leftarrow V(s)$
           $V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))\big[r + \gamma V(s')\big]$
           $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$  (a small positive number)

3. Policy Improvement
   *policy-stable* $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
       $a \leftarrow \pi(s)$
       $\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
       If $a \neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V$ and $\pi$; else go to 2
```

4.3 Policy Policy Iteration

Example 4.2: Jack's Car Rental



Exercise 4.5 (programming) Write a program for policy iteration and re-solve Jack’s car rental problem with the following changes. One of Jack’s employees at the first location rides a bus home each night and lives near the second location. She is happy to shuttle one car to the second location for free. Each additional car still costs $2, as do all cars moved in the other direction. In addition, Jack has limited parking space at each location. If more than 10 cars are kept overnight at a location (after any moving of cars), then an additional cost of $4 must be incurred to use a second parking lot (independent of how many cars are kept there). These sorts of nonlinearities and arbitrary dynamics often occur in real problems and cannot easily be handled by optimization methods other than dynamic programming. To check your program, first replicate the results given for the original problem. If your computer is too slow for the full problem, cut all the numbers of cars in half.

Exercise 4.6 How would policy iteration be defined for action values? Give a complete algorithm for computing $q_{⇤}$, analogous to Figure 4.3 for computing $v_{⇤}$. Please pay special attention to this exercise, because the ideas involved will be used throughout the rest of the book.

Exercise 4.7 Suppose you are restricted to considering only policies that are ϵ-*soft*, meaning that the probability of selecting each action in each state, s, is at least $\epsilon/|\mathcal{A}(s)|$. Describe qualitatively the changes that would be required in each of the steps 3, 2, and 1, in that order, of the policy iteration algorithm for v_* (Figure 4.3).

4.4 Value Iteration

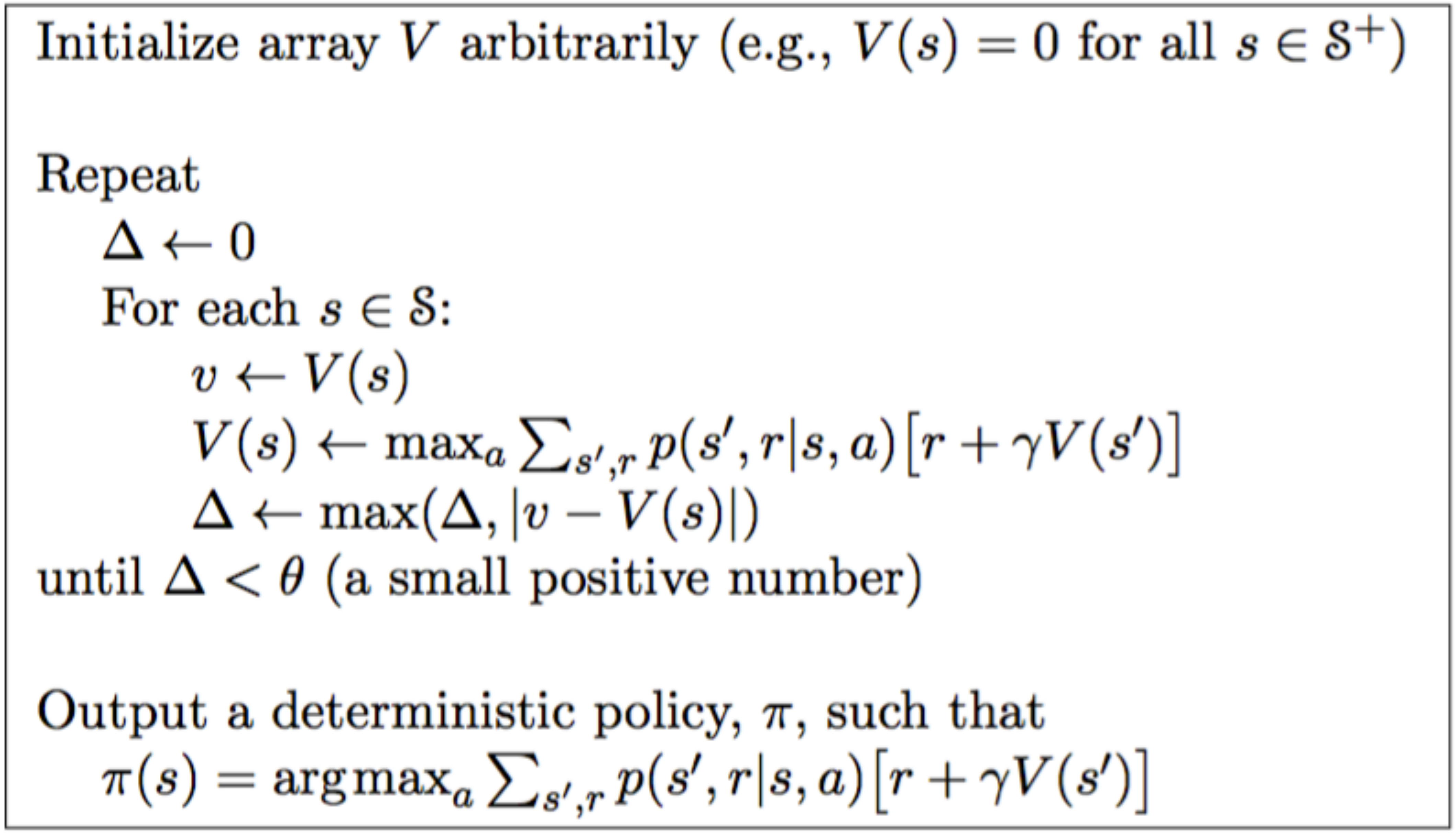
If policy evaluation is done iteratively, then convergence exactly to v_π occurs only in the limit. **Must we wait for exact convergence, or can we stop short of that?**

$$
\begin{aligned}
v_{k+1}(s) &\doteq \max_a \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \max_a \sum_{s',r} p(s', r|s, a)\Big[r + \gamma v_k(s')\Big],
\end{aligned}
$$

the sequence $\{v_k\}$ can be shown to converge to v_* under the same conditions that guarantee the existence of v_*.

n practice, we stop once the value function changes by only a small amount in a weep.

4.4 Value Iteration



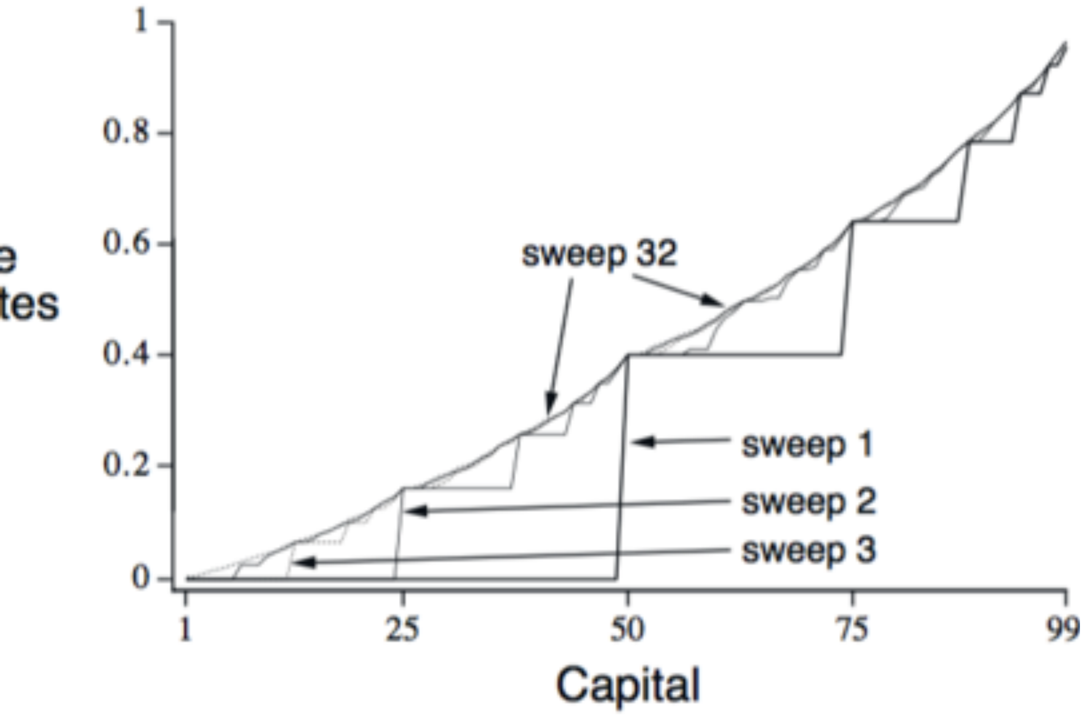
```
Initialize array $V$ arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, $\pi$, such that
    $\pi(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
```

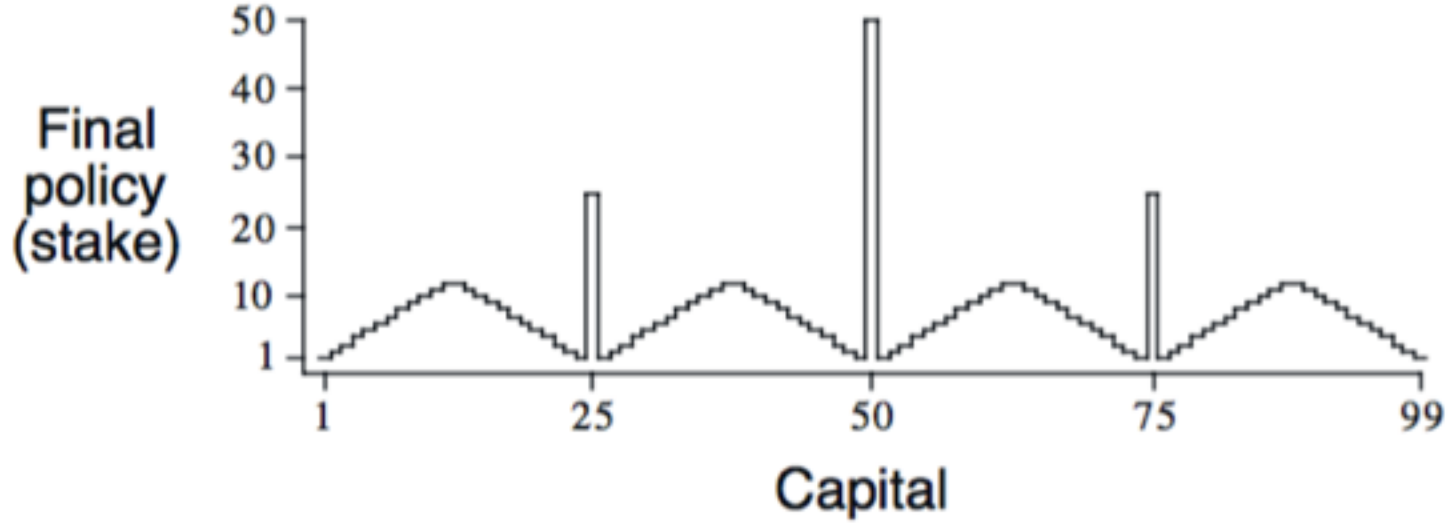
Figure 4.5: Value iteration.

All of these algorithms converge to an optimal policy for discounted finite MDPs.

4.4 Value Iteration

Example 4.3: Gambler's Problem





Exercise 4.8 Why does the optimal policy for the gambler's problem have such a curious form? In particular, for capital of 50 it bets it all on one flip, but for capital of 51 it does not. Why is this a good policy?

Exercise 4.10 What is the analog of the value iteration backup (4.10) for action values, $q_{k+1}(s, a)$?