

# Optimality and Dynamic Programming

CMPUT 261: Introduction to Artificial Intelligence

S&B §3.6, §4.0-4.4

# Lecture Outline

1. Recap & Logistics
2. Policy Evaluation
3. Optimality
4. Policy Improvement

*After this lecture, you should be able to:*

- justify why one policy is weakly better than another
- trace an execution of iterative policy evaluation
- state the Policy Improvement Theorem and describe why it is important
- trace an execution of the Value Iteration algorithm

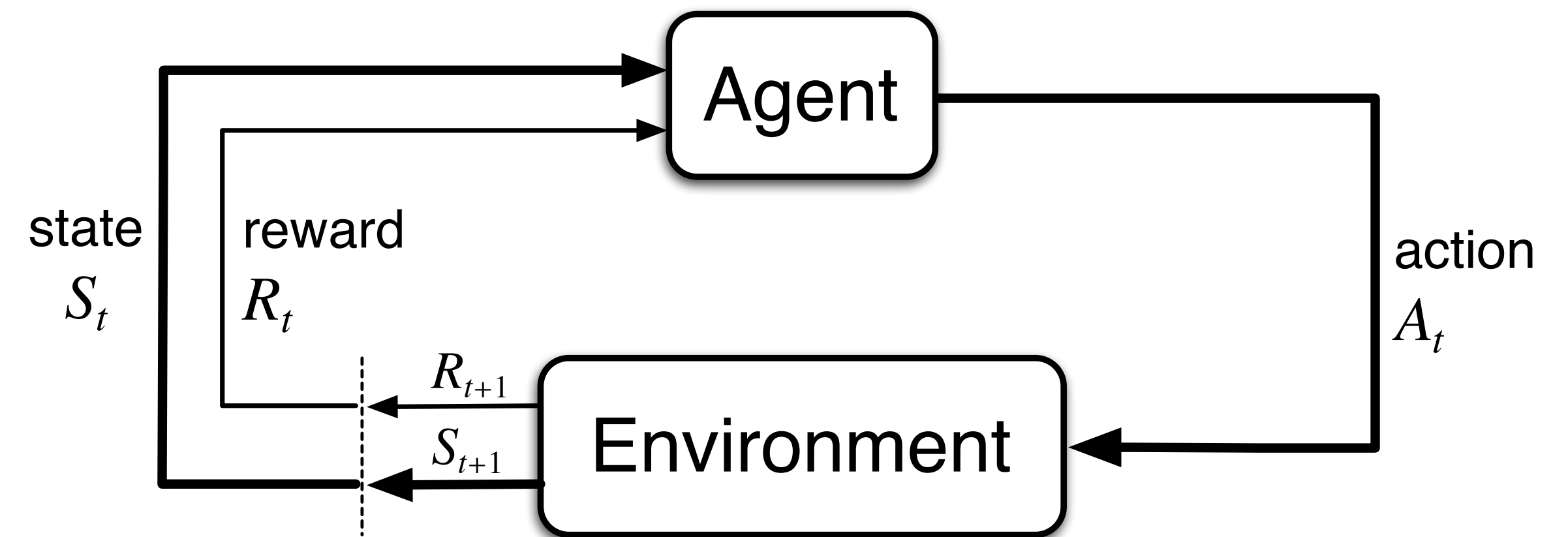
# Logistics

- **Assignment #3** is **due tomorrow** at 11:59pm
  - late submissions accepted until Friday at 11:59pm
- **Assignment #4** will be released Thursday
  - Due **Thursday April 11** at 11:59pm
- Reminder: TAs are available during labs to help

# Recap: Markov Decision Process

At each time  $t = 1, 2, 3, \dots$

1. Agent receives input denoting **current state**  $S_t$
2. Agent chooses **action**  $A_t$
3. Next time step, agent receives **reward**  $R_{t+1}$  and **new state**  $S_{t+1}$ , chosen according to a distribution  $p(s', r \mid s, a)$



This interaction between agent and environment produces a **trajectory**:  
 $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \dots$

# Recap: Policies & Value Functions

**Policy:** A function  $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$  that maps states to a distribution over actions

- For every state  $s$ , agent following policy  $\pi$  will play action  $a$  with probability  $\pi(a \mid s)$

**State-value function:**

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

**Action-value function:**

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

# Recap: Bellman Equations

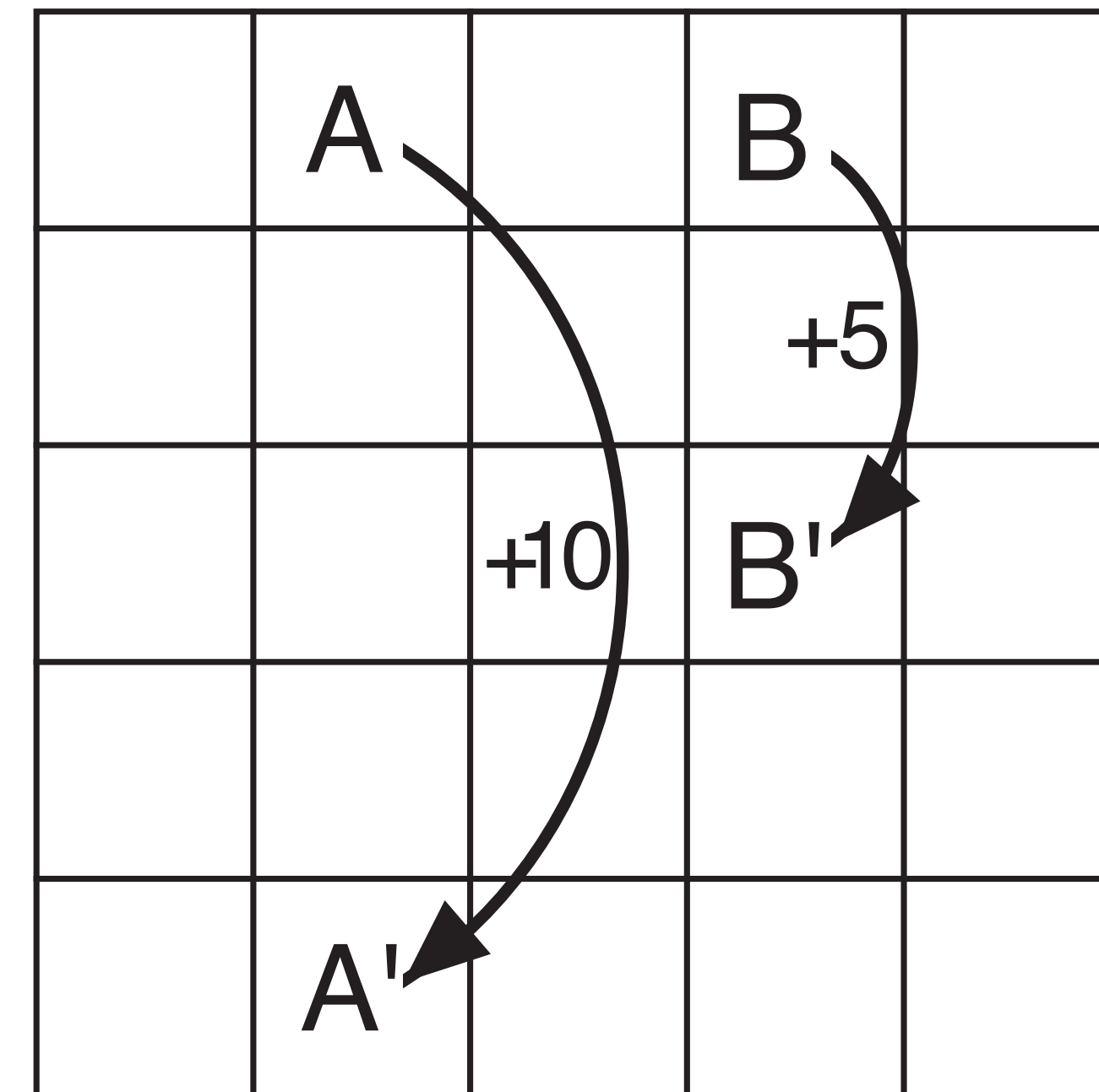
Value functions satisfy a **recursive consistency condition** called the **Bellman equation**:

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

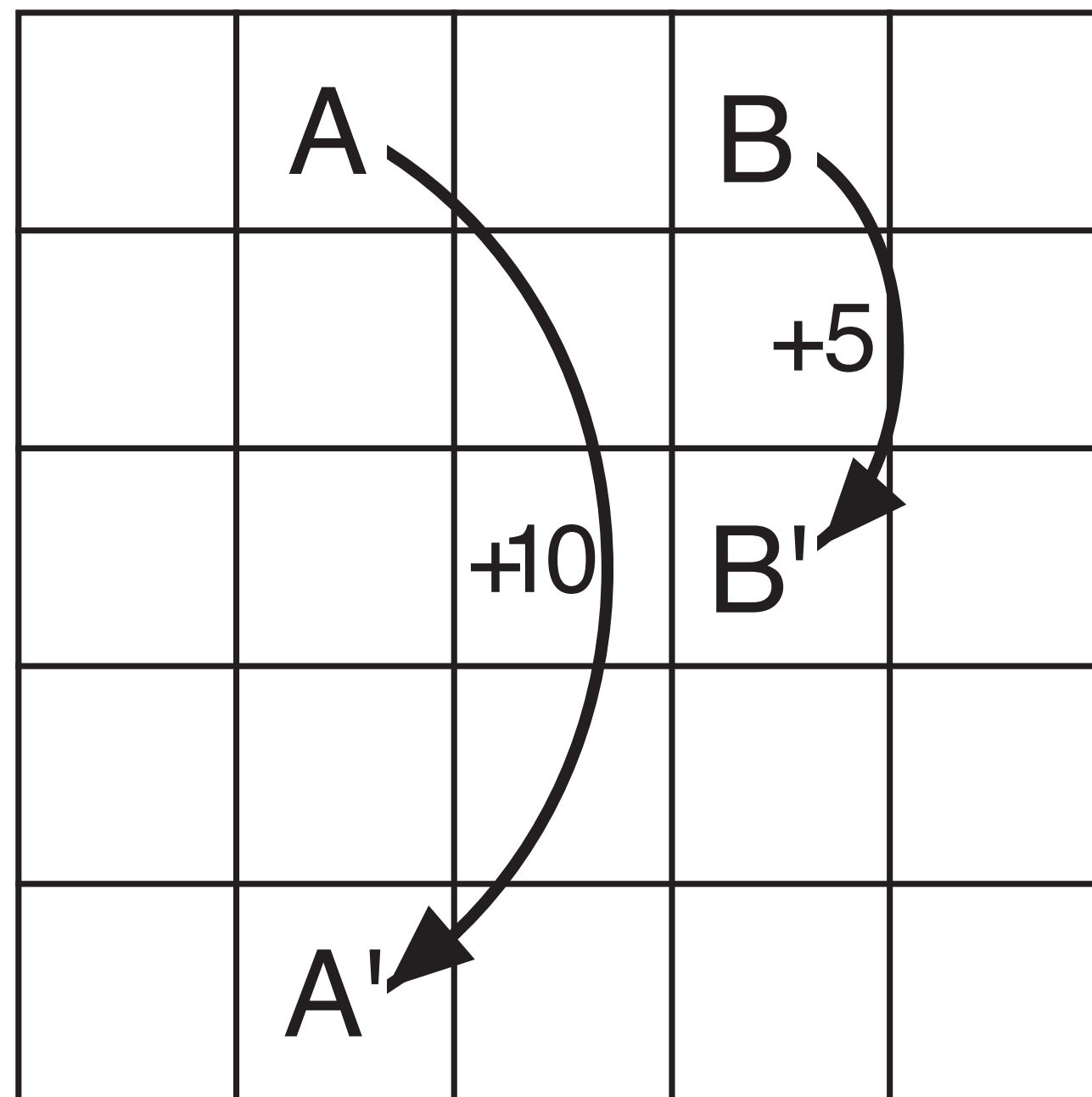
- $v_{\pi}$  is the **unique solution** to  $\pi$ 's (state-value) Bellman equation
- There is also a Bellman equation for  $\pi$ 's **action-value function**

# Example: GridWorld

- At each cell, can go north, south, east, west
- Try to go off the **edge**: reward of **-1**
- Leaving state **A**: takes you to state **A'**, reward of **+10**
- Leaving state **B**: takes you to state **B'**, reward of **+5**



# GridWorld



Reward dynamics

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

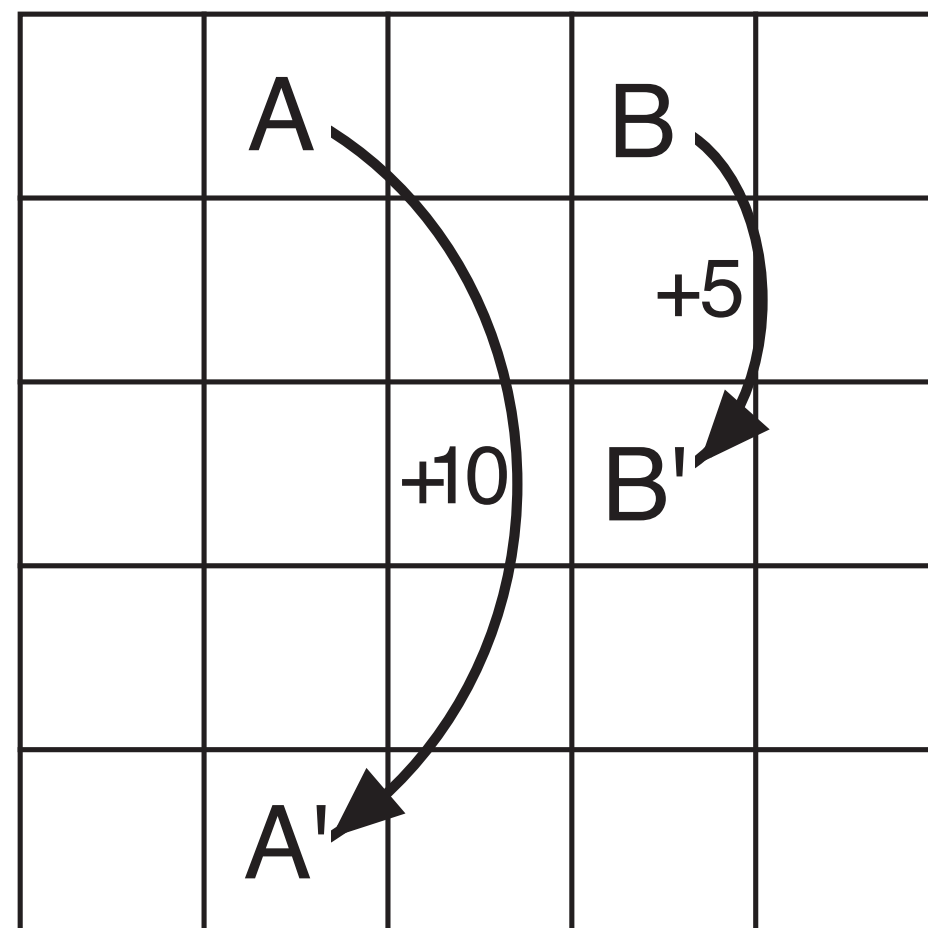
State-value function  $v_\pi$  for random policy

$$\pi(a | s) = 0.25$$



# GridWorld with Bounds Checking

What about a policy where we never try to go over an edge?



Reward dynamics

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

State-value function  $v_\pi$  for  
random policy  
 $\pi(a | s) = 0.25$

6.7	10.8	6.4	6.7	4.3
4.2	4.7	3.7	3.4	2.8
2.4	2.4	2.1	1.9	1.7
1.5	1.4	1.3	1.2	1.1
1.1	1.0	0.9	0.9	0.9

State-value function  $v_{\pi^B}$  for  
**bounded** random policy  $\pi^B$

# Policy Evaluation

**Question:** How can we **compute**  $v_\pi$ ?

1. We know that  $v_\pi$  is the unique solution to the Bellman equations, so we could just **solve them** (treating  $v_\pi(s_1), \dots, v_\pi(s_{|\mathcal{S}|})$  as variables)
  - but that is tedious and annoying and slow  
(it's a system of  $|\mathcal{S}|$  linear equations in  $|\mathcal{S}|$  unknowns)
  - Also requires a complete model of the dynamics
2. **Iterative policy evaluation**
  - Takes advantage of the recursive formulation

# Iterative Policy Evaluation

- Iterative policy evaluation uses the Bellman equation as an **update rule**:

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

- $v_{\pi}$  is a **fixed point** of this update, by definition
- Furthermore, starting from an **arbitrary**  $v_0$ , the sequence  $\{v_k\}$  will **converge** to  $v_{\pi}$  as  $k \rightarrow \infty$ 
  - *(nontrivial to prove)*

# In-Place Iterative Policy Evaluation

Iterative Policy Evaluation, for estimating  $V \approx v_\pi$

Input  $\pi$ , the policy to be evaluated

Algorithm parameter: a small threshold  $\theta > 0$  determining accuracy of estimation

Initialize  $V(s)$ , for all  $s \in \mathcal{S}^+$ , arbitrarily except that  $V(\text{terminal}) = 0$

Loop:

$\Delta \leftarrow 0$

Loop for each  $s \in \mathcal{S}$ :

$v \leftarrow V(s)$

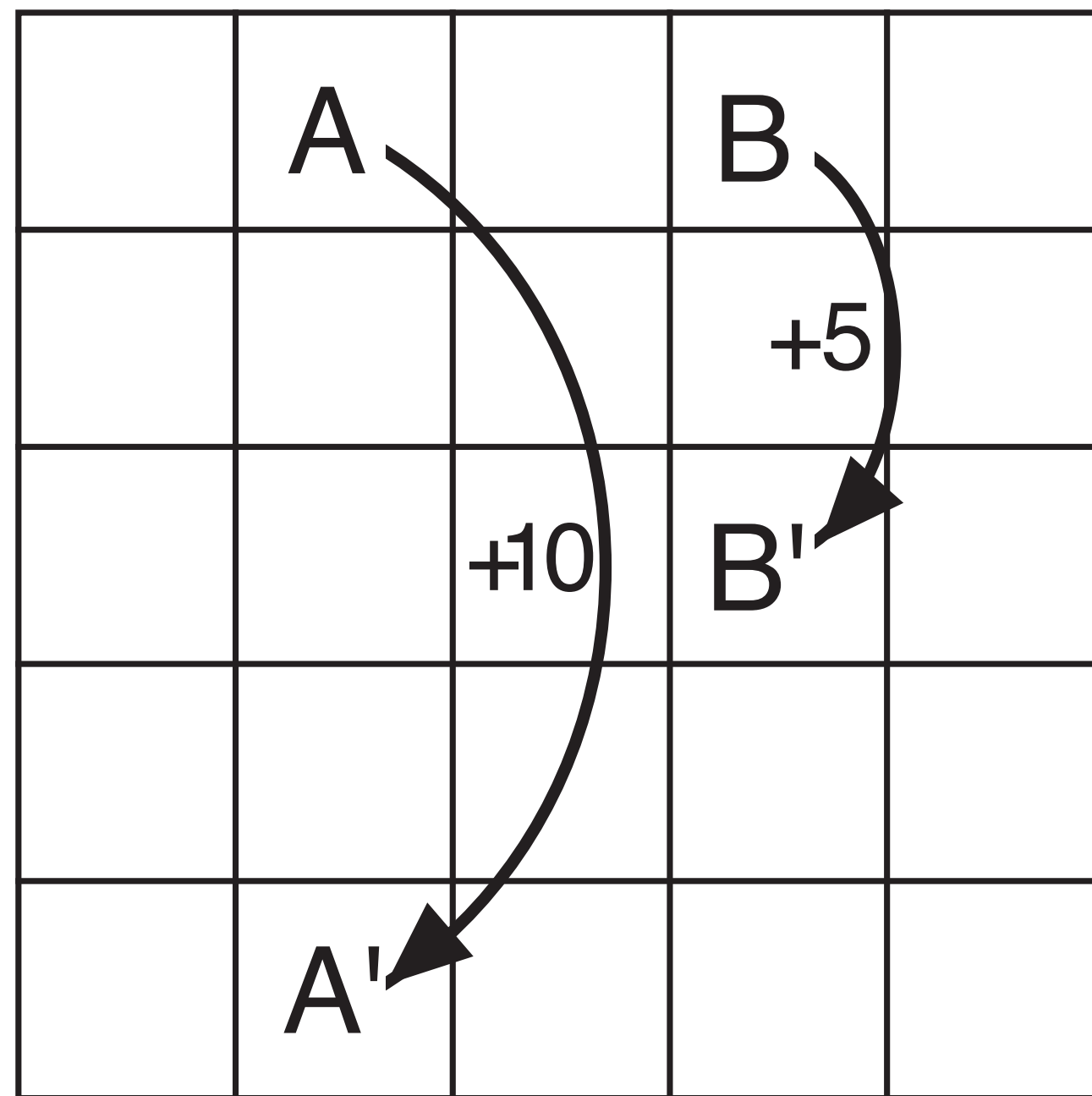
$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s', r} p(s', r | s, a) [r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until  $\Delta < \theta$

- The updates are **in-place**: we use new values for  $V(s)$  **immediately** instead of waiting for the current sweep to complete (**why?**)
- These are **expected updates**: Based on a weighted average (expectation) of **all possible next states** (**instead of what?**)

# Iterative Policy Evaluation



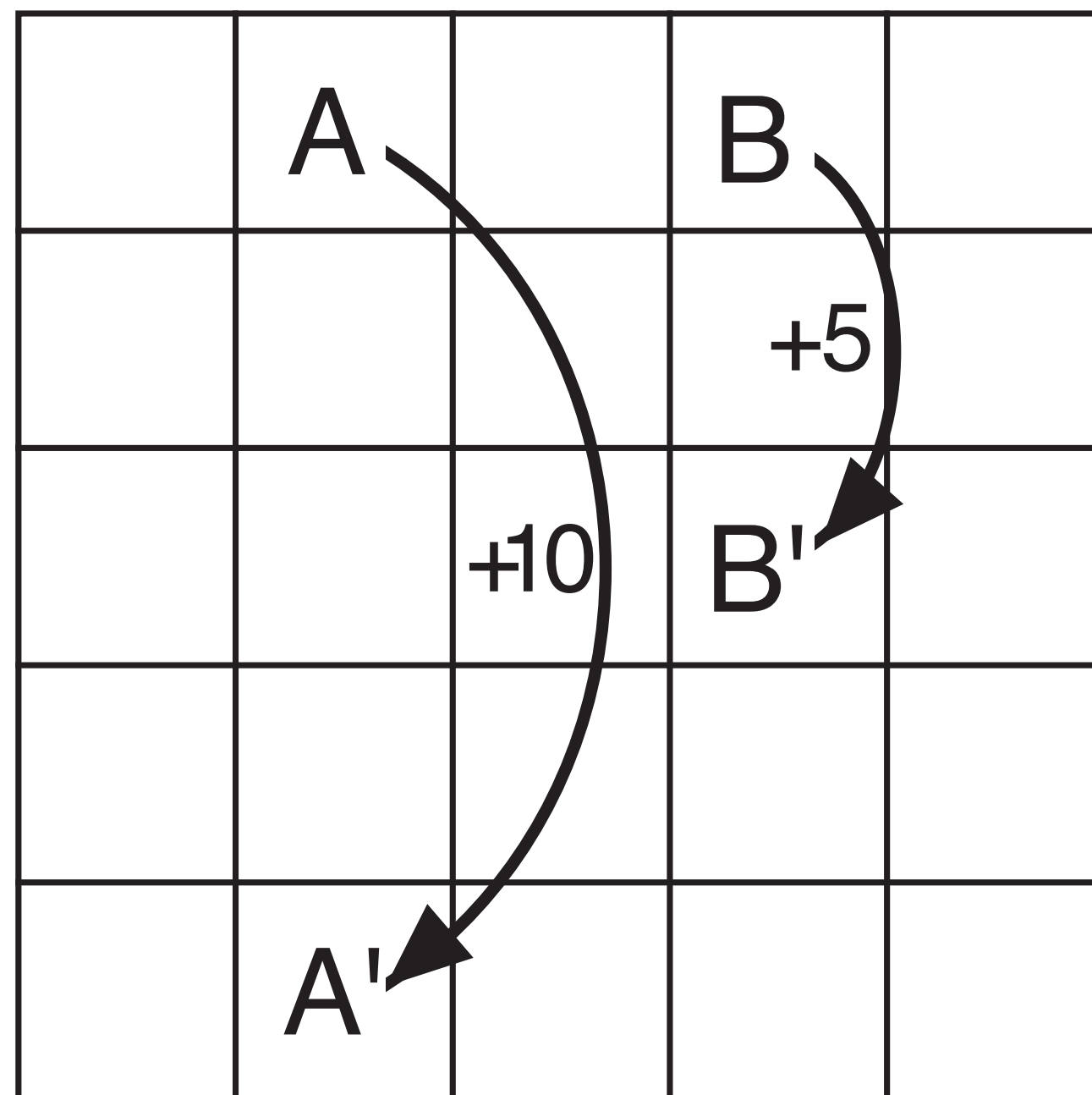
Reward dynamics

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	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0

$V$  at  $k = 0$

# Iterative Policy Evaluation

$$\begin{aligned} V(s_{1,1}) &= \pi(n)[-1 + \gamma V(s_{1,1})] + \pi(w)[-1 + \gamma V(s_{1,1})] + \\ &\quad \pi(s)[0 + \gamma V(s_{1,2})] + \pi(e)[0 + \gamma V(s_{2,1})] \\ &= 0.25(-1) + 0.25(-1) + 0.25(0) + 0.25(0) \end{aligned}$$



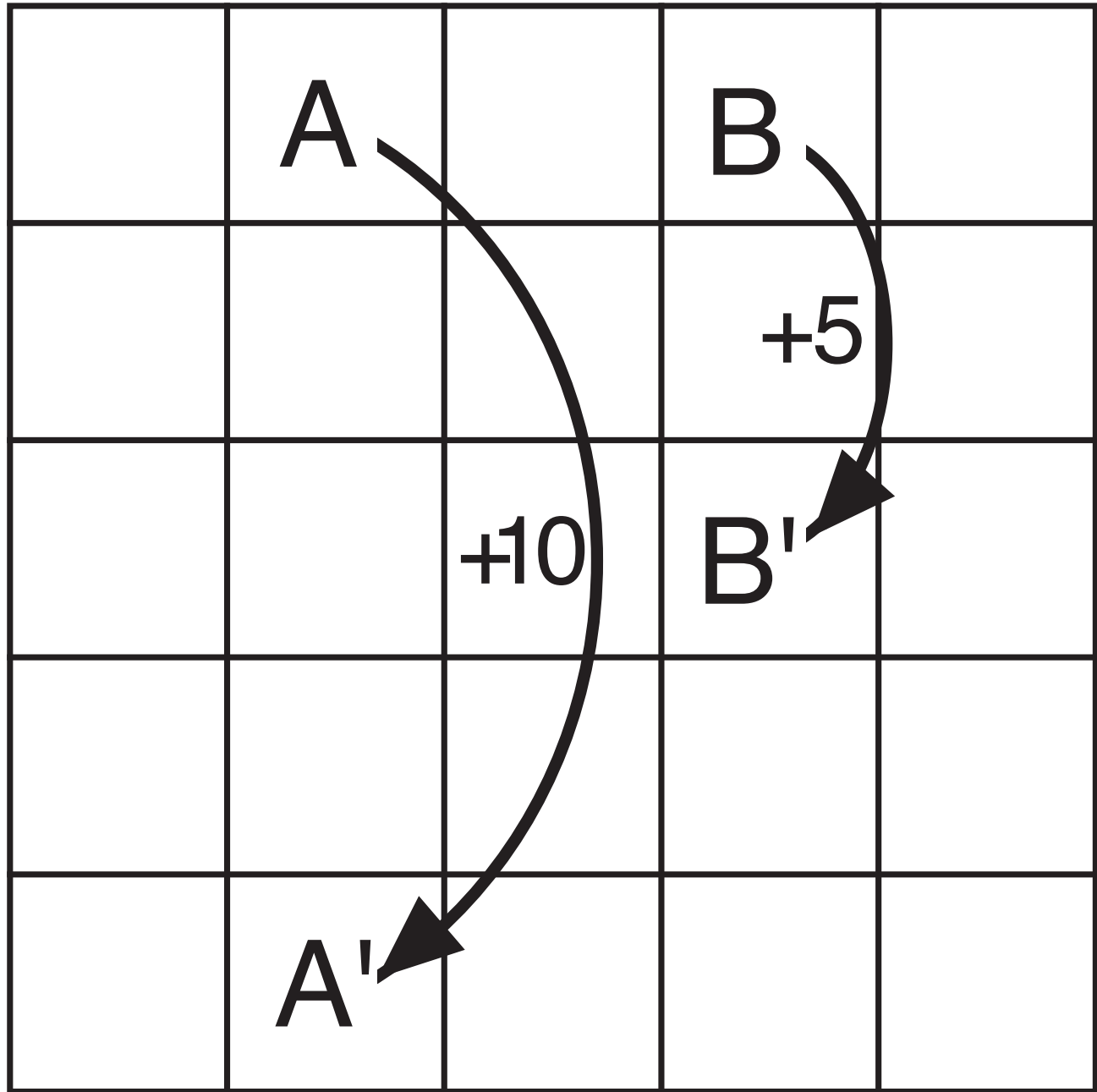
Reward dynamics

<b>-0.5</b>	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	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0

$V$  at  $k = 0$

# Iterative Policy Evaluation

$$\begin{aligned} V(s_{1,2}) &= \pi(n)[10 + \gamma \mathbf{V}(s_{2,5})] + \pi(w)[10 + \gamma \mathbf{V}(s_{2,5})] + \\ &\quad \pi(s)[10 + \gamma \mathbf{V}(s_{2,5})] + \pi(e)[10 + \gamma \mathbf{V}(s_{2,5})] \\ &= 0.25[10 + 0.9(\mathbf{0})] + 0.25[10 + 0.9(\mathbf{0})] + \\ &\quad 0.25[10 + 0.9(\mathbf{0})] + 0.25[10 + 0.9(\mathbf{0})] \end{aligned}$$



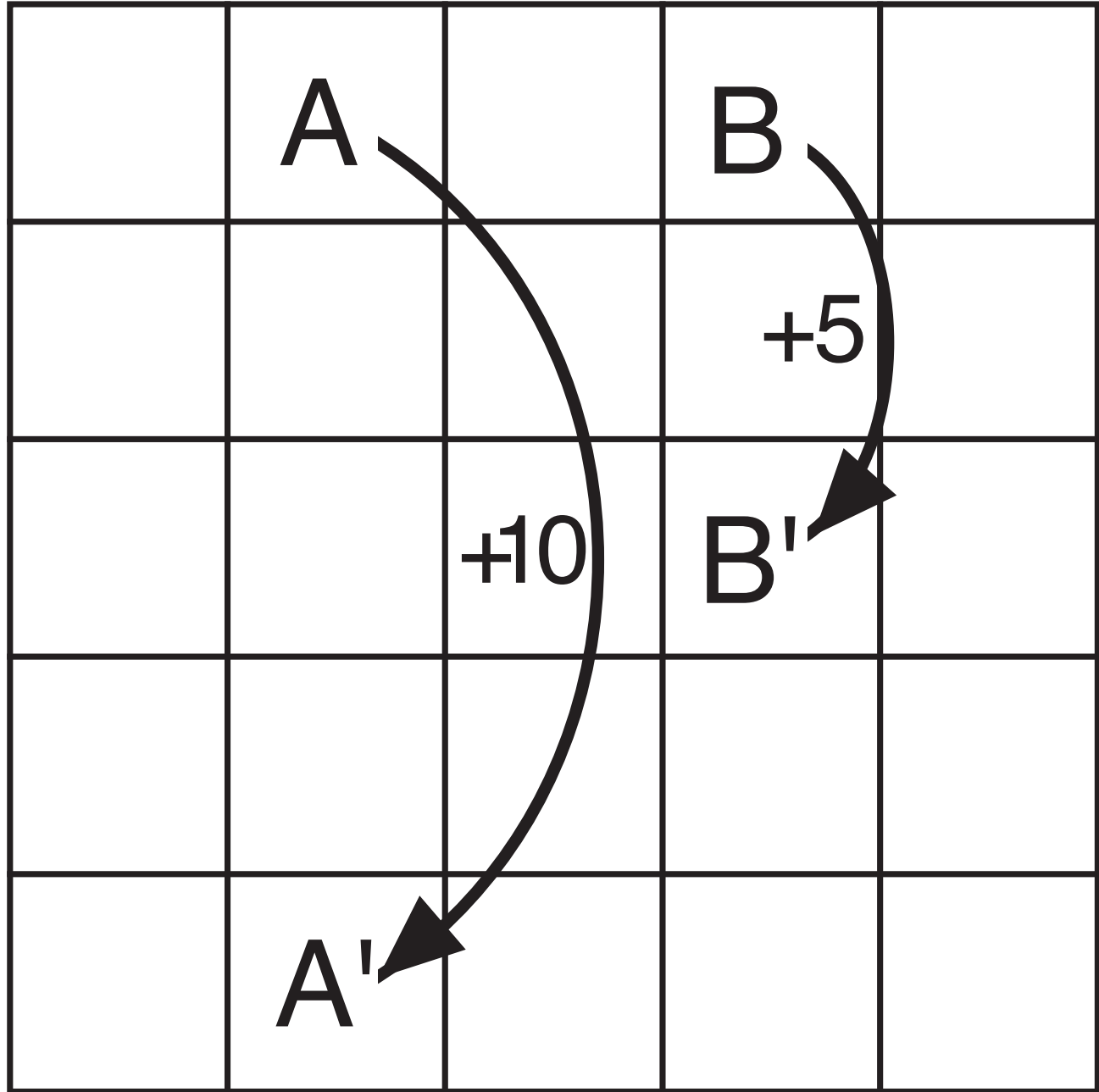
Reward dynamics

-0.5	10	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	0.0	0.0
0.0	0.0	0.0	0.0	0.0

$V$  at  $k = 0$

# Iterative Policy Evaluation

$$\begin{aligned} V(s_{3,1}) &= \pi(n)[-1 + \gamma V(s_{3,1})] + \pi(w)[-1 + \gamma \mathbf{V(s_{2,1})}] + \\ &\quad \pi(s)[0 + \gamma V(s_{3,2})] + \pi(e)[0 + \gamma V(s_{4,1})] \\ &= 0.25[-1 + 0.9(0)] + 0.25[0 + 0.9(\mathbf{10})] + \\ &\quad 0.25[0 + 0.9(0)] + 0.25[0 + 0.9(0)] \end{aligned}$$



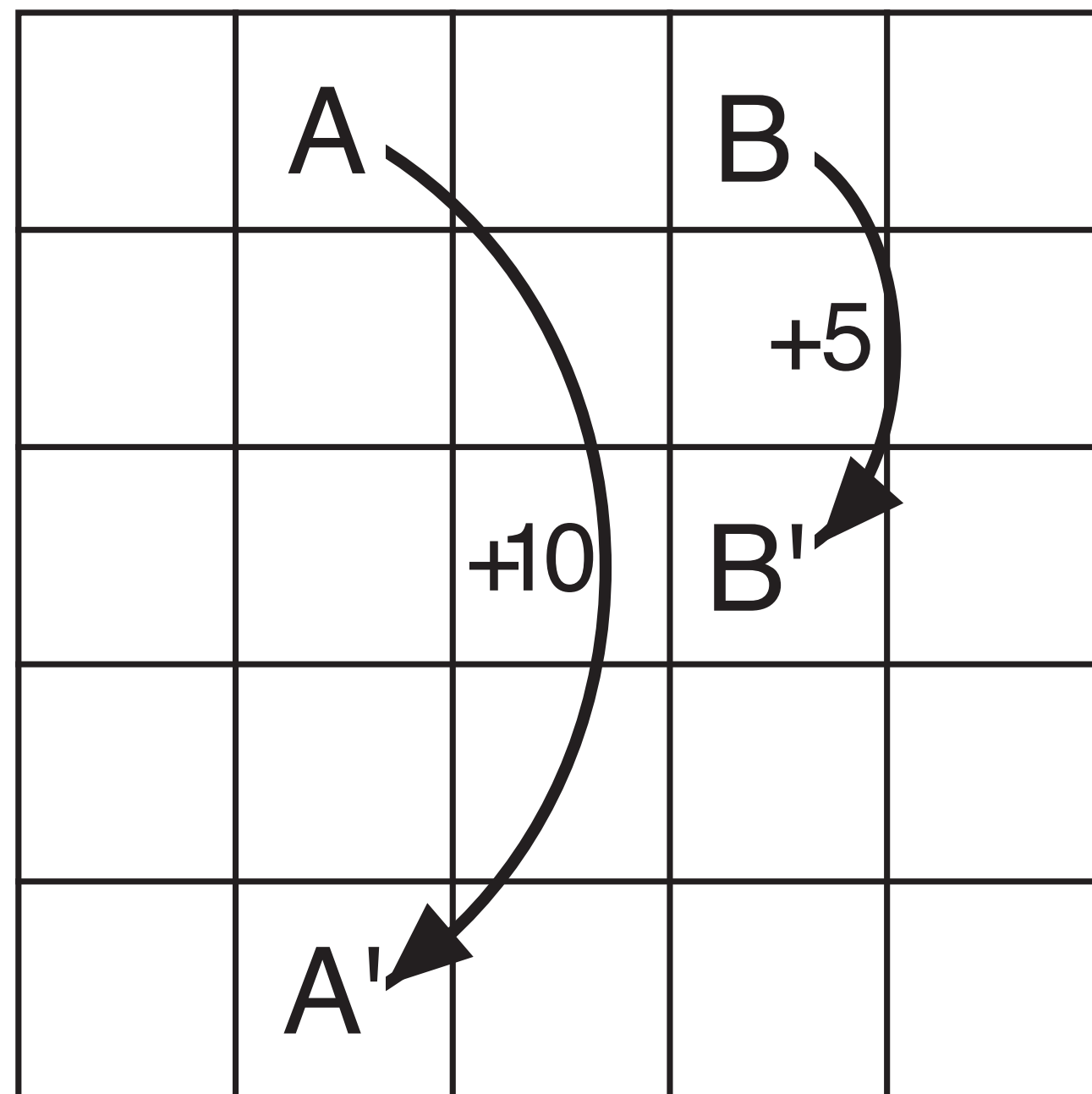
Reward dynamics

-0.5	10	2	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	0.0
0.0	0.0	0.0	0.0	0.0

$V$  at  $k = 0$



# Iterative Policy Evaluation in GridWorld

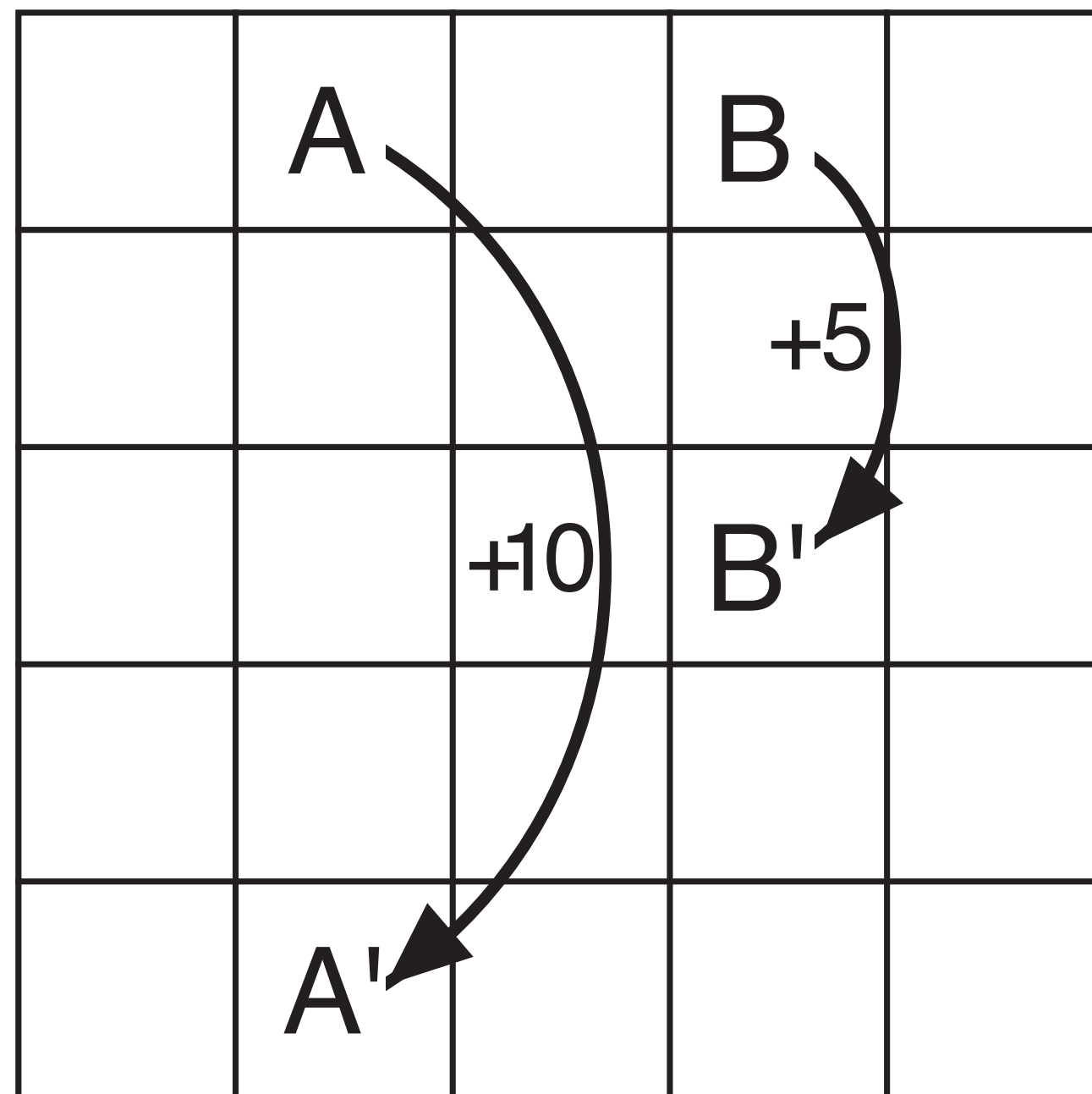


Reward dynamics

-0.5	10	2	5	0.6
-0.3	2.1	0.9	1.3	0.2
-0.3	0.4	0.3	0.4	-0.1
-0.3	0.0	0.0	0.1	-0.2
-0.5	-0.3	-0.3	-0.3	-0.6

$V$  at  $k = 1$

# Iterative Policy Evaluation in GridWorld

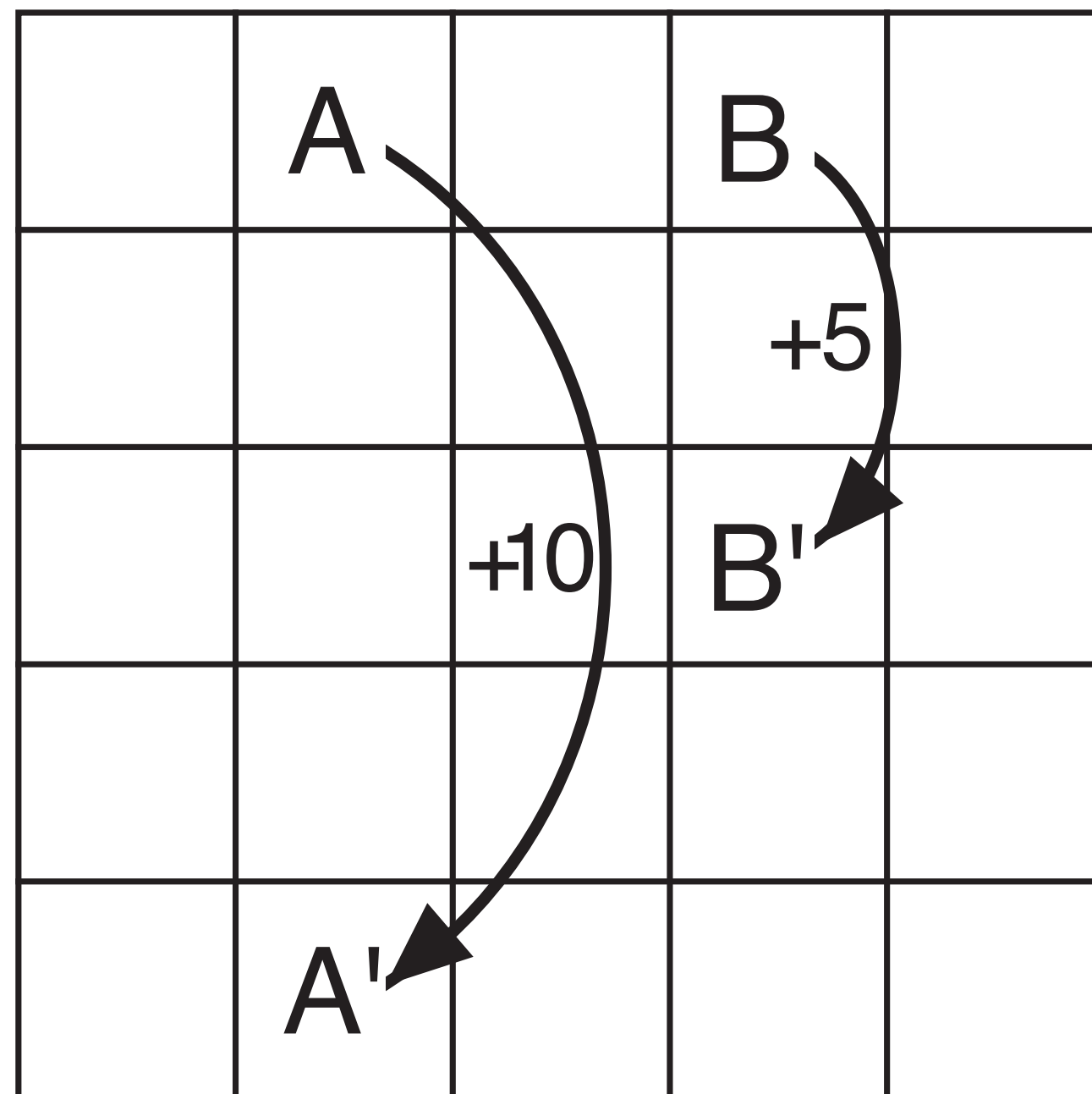


Reward dynamics

1.4	9.7	3.7	5.3	1.0
0.4	2.5	1.8	1.7	0.4
-0.2	0.6	0.6	0.5	-0.1
-0.5	0.0	0.0	0.0	-0.5
-1.0	-0.6	-0.5	-0.5	-1.0

$V$  at  $k = 2$

# Iterative Policy Evaluation in GridWorld



Reward dynamics

3.4	8.9	4.5	5.3	1.5
1.6	3.0	2.3	1.9	0.6
0.1	0.8	0.7	0.4	-0.4
-1.0	-0.4	-0.3	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

$V$  at  $k = 10\,000$

# Optimality

- **Question:** What is an **optimal** policy?
- A policy  $\pi$  is (weakly) **better** than a policy  $\pi'$  if it is better **for all**  $s \in \mathcal{S}$  :

$$\pi \geq \pi' \iff v_{\pi}(s) \geq v_{\pi'}(s) \quad \forall s \in \mathcal{S}.$$

- An **optimal** policy  $\pi_*$  is weakly better than **every other policy**
- **Question:** Is an optimal policy guaranteed to exist for a given MDP?
- All optimal policies share the **same state-value function**: (**why?**)

$$v_*(s) \doteq \max_{\pi} v_{\pi}(s)$$

- Also the same **action-value function**:

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

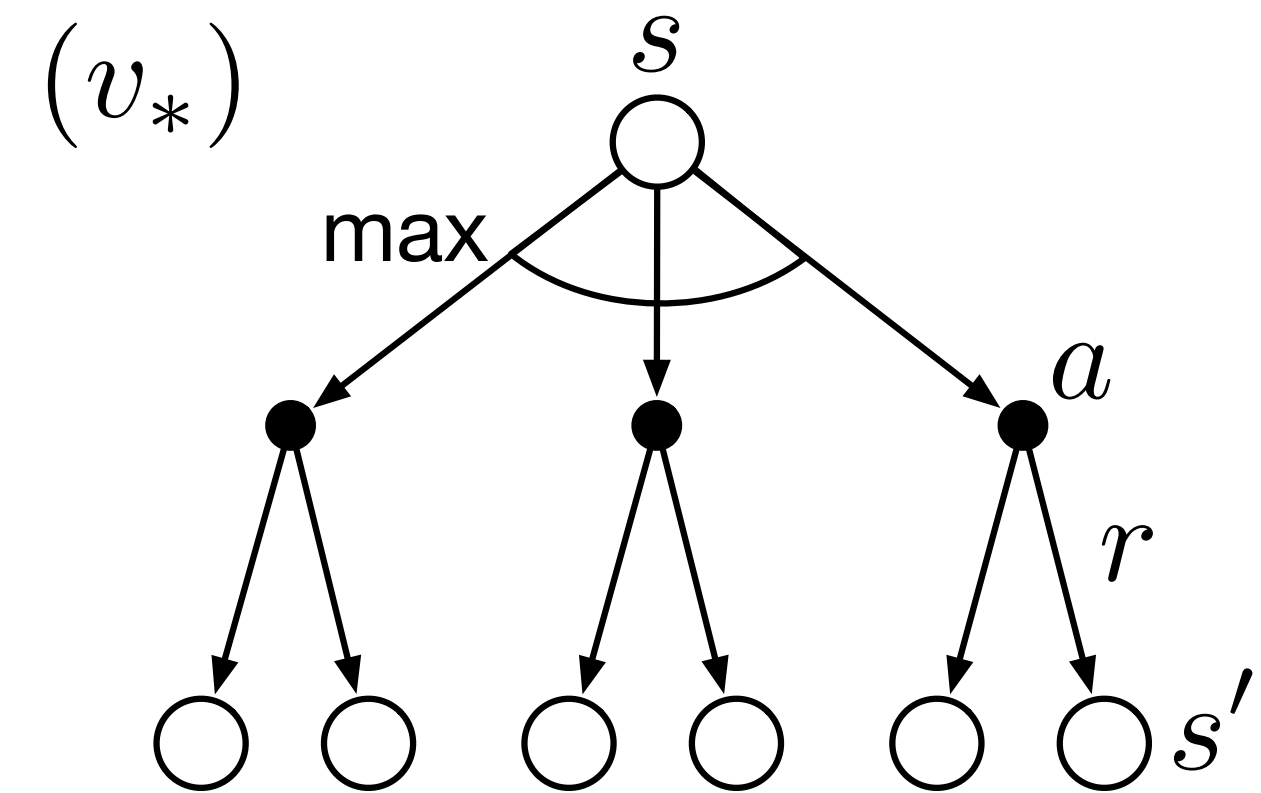
# Bellman Optimality Equations

- $v_*$  must satisfy the Bellman equation too
- In fact, it can be written in a special, **policy-free** way because we know that every state value is **maximized** by  $\pi_*$ :

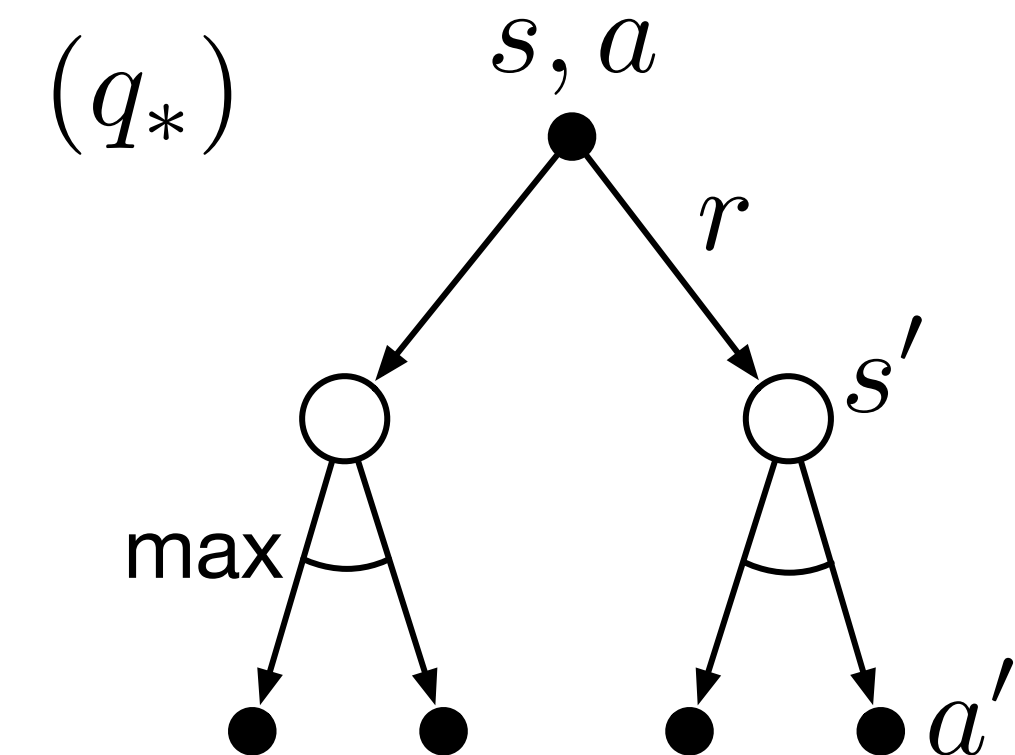
$$\begin{aligned} v_*(s) &= \max_a q_{\pi_*}(s, a) \\ &= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] \\ &= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\ &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \max_a \sum_{s', r} p(s', r \mid s, a)[r + \gamma v_*(s')] \end{aligned}$$

# Bellman Optimality Equations

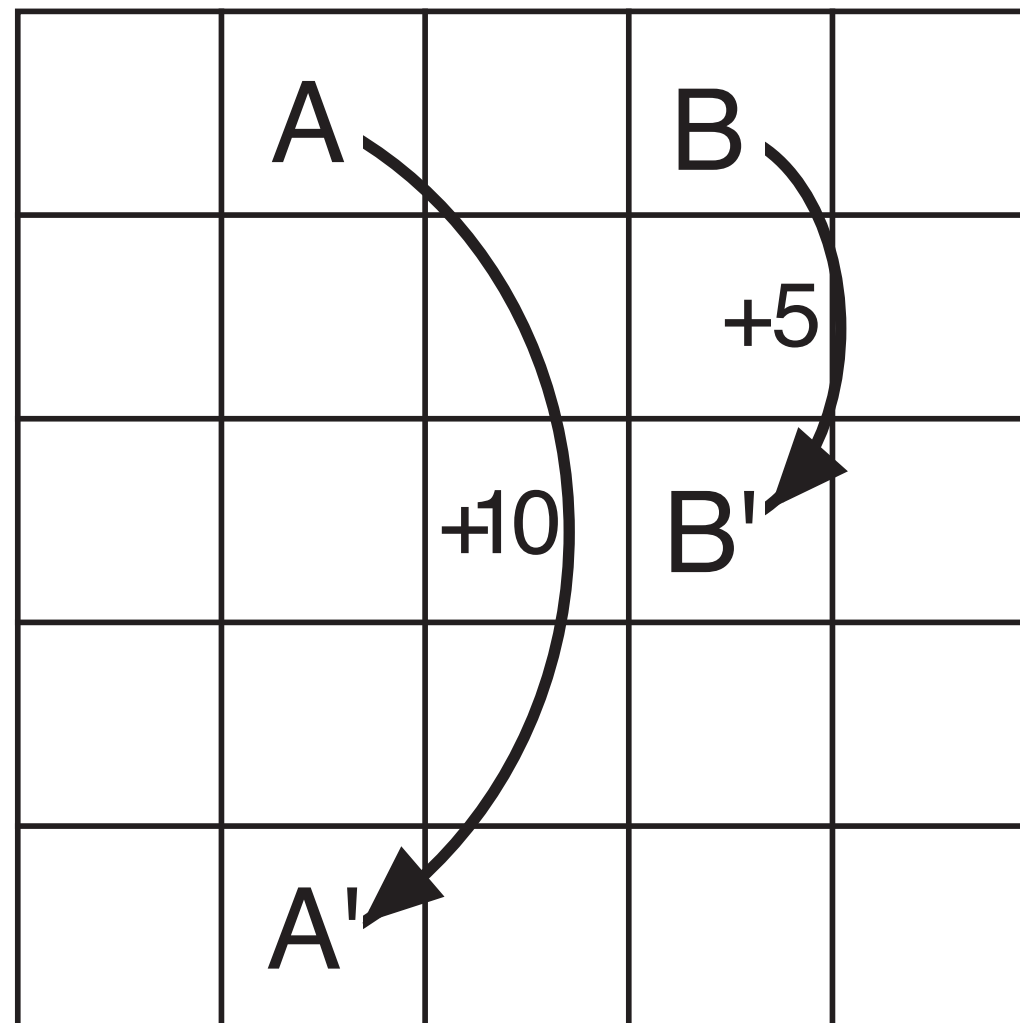
$$\begin{aligned}
 v_*(s) &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\
 &= \max_a \sum_{s', r} p(s', r \mid s, a) [r + \gamma v_*(s')]
 \end{aligned}$$



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



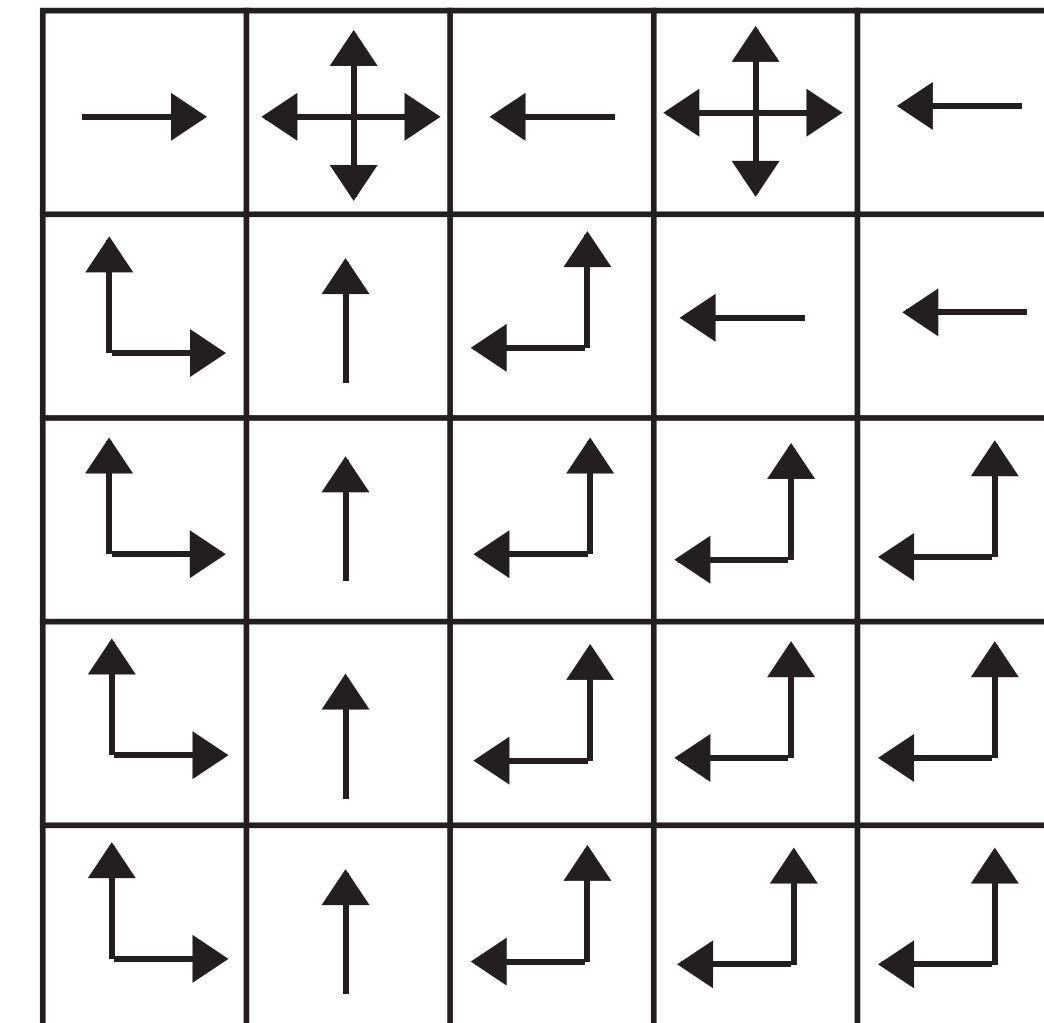
# Optimal GridWorld



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

$v_*$



$\pi_*$

# Policy Improvement Theorem

## Theorem:

Let  $\pi$  and  $\pi'$  be any pair of deterministic policies.

If  $q_{\pi}(s, \pi'(s)) \geq v_{\pi}(s) \quad \forall s \in \mathcal{S}$ ,

then  $v_{\pi'}(s) \geq v_{\pi}(s) \quad \forall s \in \mathcal{S}$ .

If you are never worse off **at any state** by following  $\pi'$  for **one step** and then following  $\pi$  forever after, then following  $\pi'$  **forever** has a higher expected value **at every state**.



# Policy Improvement Theorem Proof

$$v_{\pi}(s) \leq q_{\pi}(s, \pi'(s)) \quad \forall s$$

$$= \mathbb{E}_{\pi}[R_{t+1} + \gamma v_{\pi}(S_{t+1}) \mid S_t = s, A_t = \pi'(s)]$$

$$= \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_{\pi}(S_{t+1}) \mid S_t = s]$$

$$\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma q_{\pi}(S_{t+1}, \pi'(S_{t+1})) \mid S_t = s]$$

$$= \mathbb{E}_{\pi'}[R_{t+1} + \gamma \mathbb{E}_{\pi'}[R_{t+2} + \gamma v_{\pi}(S_{t+2}) \mid S_{t+1}, A_{t+1} = \pi'(S_{t+1})] \mid S_t = s]$$

$$= \mathbb{E}_{\pi'}[R_{t+1} + \gamma \mathbb{E}_{\pi'}[R_{t+2}] + \gamma^2 \mathbb{E}_{\pi'}[v_{\pi}(S_{t+2})] \mid S_t = s]$$

$$= \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_{\pi}(S_{t+2}) \mid S_t = s]$$

$$\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 v_{\pi}(S_{t+3}) \mid S_t = s]$$

$\vdots$

$$\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \mid S_t = s]$$

$$= v_{\pi'}(s).$$

# Greedy Policy Improvement

Given any policy  $\pi$ , we can construct a new greedy policy  $\pi'$  that is guaranteed to be **at least as good**:

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

- If this new policy is not **strictly** better than the old policy, then  $v_\pi(s) = v_{\pi'}(s)$  for all  $s \in \mathcal{S}$  (**why?**)
- Also means that the new (and old) policies are **optimal** (**why?**)

# Policy Iteration

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

Policy Iteration (using iterative policy evaluation) for estimating  $\pi \approx \pi_*$

1. Initialization

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

2. Policy Evaluation

Loop:

$\Delta \leftarrow 0$

Loop for each  $s \in \mathcal{S}$ :

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s)) [r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until  $\Delta < \theta$  (a small positive number determining the accuracy of estimation)

3. Policy Improvement

$policy-stable \leftarrow true$

For each  $s \in \mathcal{S}$ :

$old-action \leftarrow \pi(s)$

$\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s',r|s,a) [r + \gamma V(s')]$

If  $old-action \neq \pi(s)$ , then  $policy-stable \leftarrow false$

If  $policy-stable$ , then stop and return  $V \approx v_*$  and  $\pi \approx \pi_*$ ; else go to 2

This is a lot of iterations!  
Is it necessary to run to completion?

# Value Iteration

**Value iteration interleaves** the estimation and improvement steps:

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

**Value Iteration, for estimating  $\pi \approx \pi_*$**

Algorithm parameter: a small threshold  $\theta > 0$  determining accuracy of estimation  
Initialize  $V(s)$ , for all  $s \in \mathcal{S}^+$ , arbitrarily except that  $V(\text{terminal}) = 0$

Loop:

```
|  $\Delta \leftarrow 0$ 
| Loop for each  $s \in \mathcal{S}$ :
|    $v \leftarrow V(s)$ 
|    $V(s) \leftarrow \max_a \sum_{s', r} p(s', r \mid s, a) [r + \gamma V(s')]$ 
|    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$ 
until  $\Delta < \theta$ 
```

Output a deterministic policy,  $\pi \approx \pi_*$ , such that

$$\pi(s) = \operatorname{argmax}_a \sum_{s', r} p(s', r \mid s, a) [r + \gamma V(s')]$$

# Summary

- An **optimal policy** has higher state value than any other policy **at every state**
- A policy's state-value function can be computed by **iterating** an **expected update** based on the Bellman equation
- Given any policy  $\pi$ , we can compute a **greedy improvement**  $\pi'$  by choosing highest expected value action based on  $v_\pi$
- **Policy iteration:** Repeat:  
Greedy improvement using  $v_\pi$ , then recompute  $v_\pi$
- **Value iteration:** Repeat:  
Recompute  $v_\pi$  by assuming greedy improvement at every update