



ELTE

FACULTY OF
INFORMATICS

DYNAMIC PROGRAMMING

Deep Reinforcement Learning
Balázs Nagy, PhD



ELTE | IK

DEPARTMENT OF
ARTIFICIAL
INTELLIGENCE

Fibonacci series

Write a function that returns the n^{th} Fibonacci number

$$\text{Fib}(0) = 0$$

$$\text{Fib}(1) = 1$$

$$\text{Fib}(2) = 1$$

$$\text{Fib}(3) = 2$$

$$\text{Fib}(4) = 3$$

$$\text{Fib}(5) = 5$$

$$\text{Fib}(6) = 8$$

General rule:

$$\text{Fib}(0) = 0$$

$$\text{Fib}(1) = 1$$

$$\text{Fib}(n) = \text{Fib}(n-1) + \text{Fib}(n-2)$$

Naive approach

```
def fib(n):  
    if n <= 1:  
        result = 1  
  
    else:  
        result = fib(n - 1) + fib(n - 2)  
  
    return result
```

Naive approach

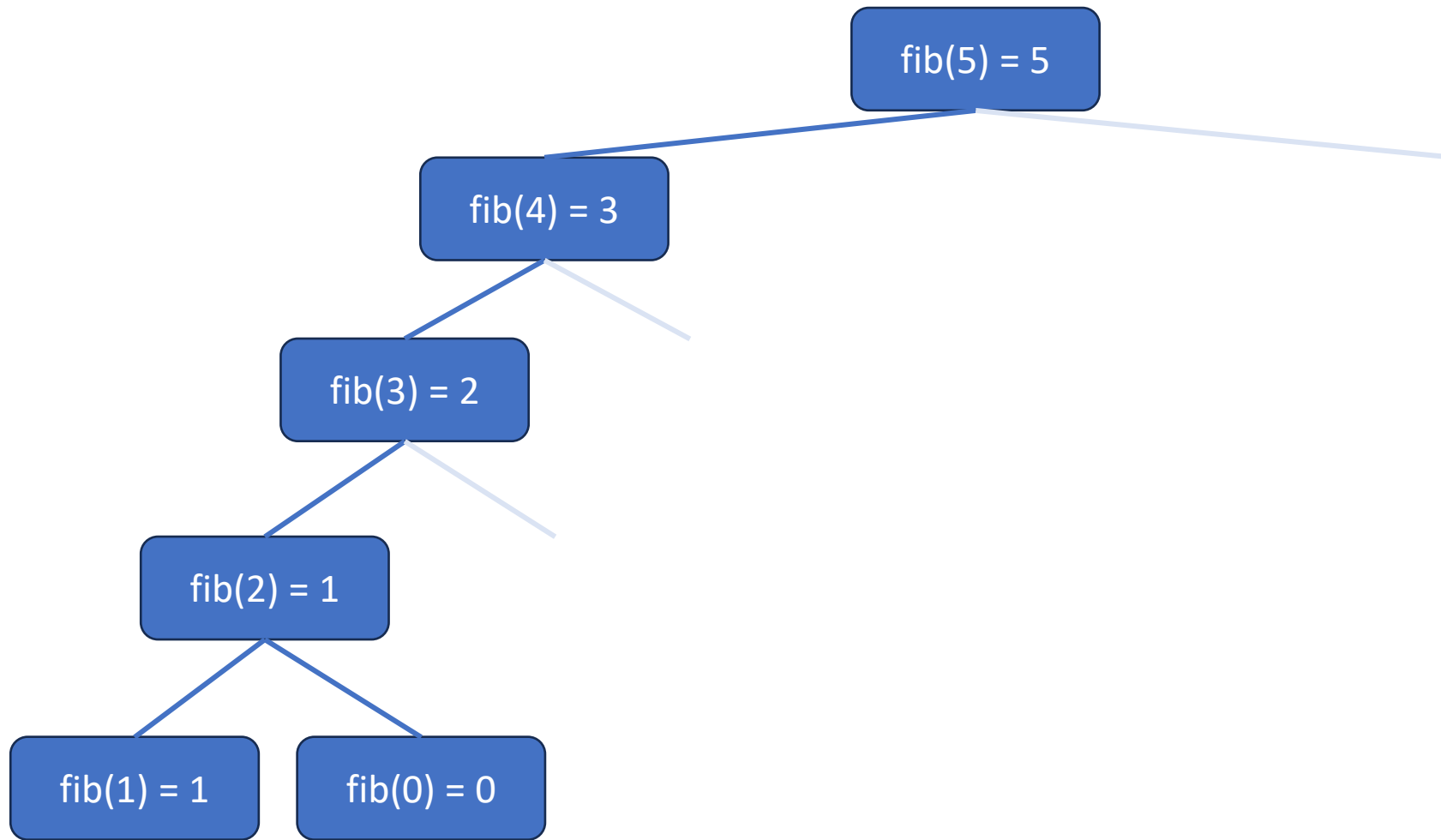
```
def fib(n):  
    if n <= 1:  
        result = 1  
  
    else:  
        result = fib(n - 1) + fib(n - 2)  
  
    return result
```

```
print(fib(7))    Output: 13
```

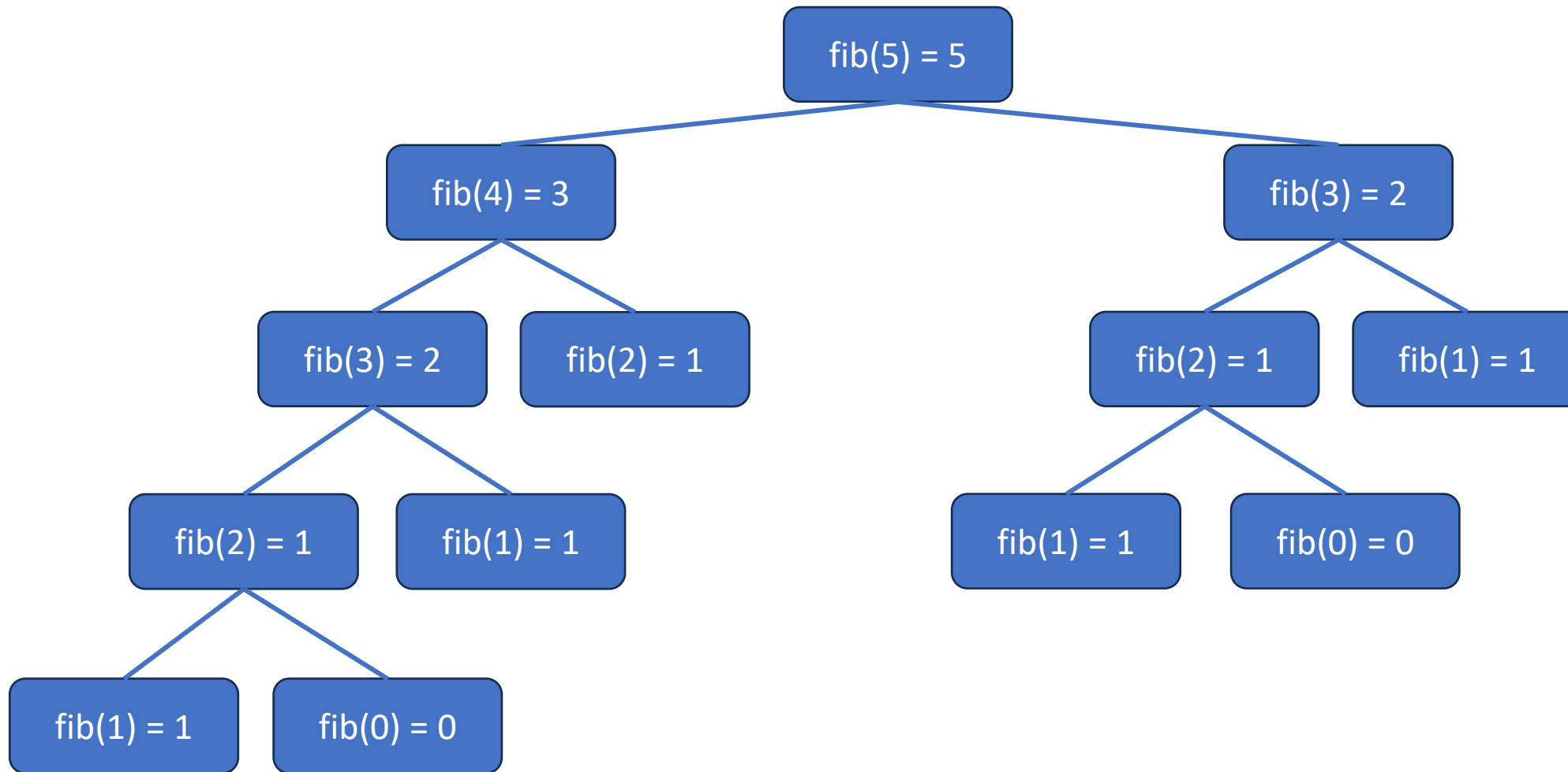
```
print(fib(13))   Output: TIMEOUT
```

Time complexity: $O(2^{\frac{n}{2}})$

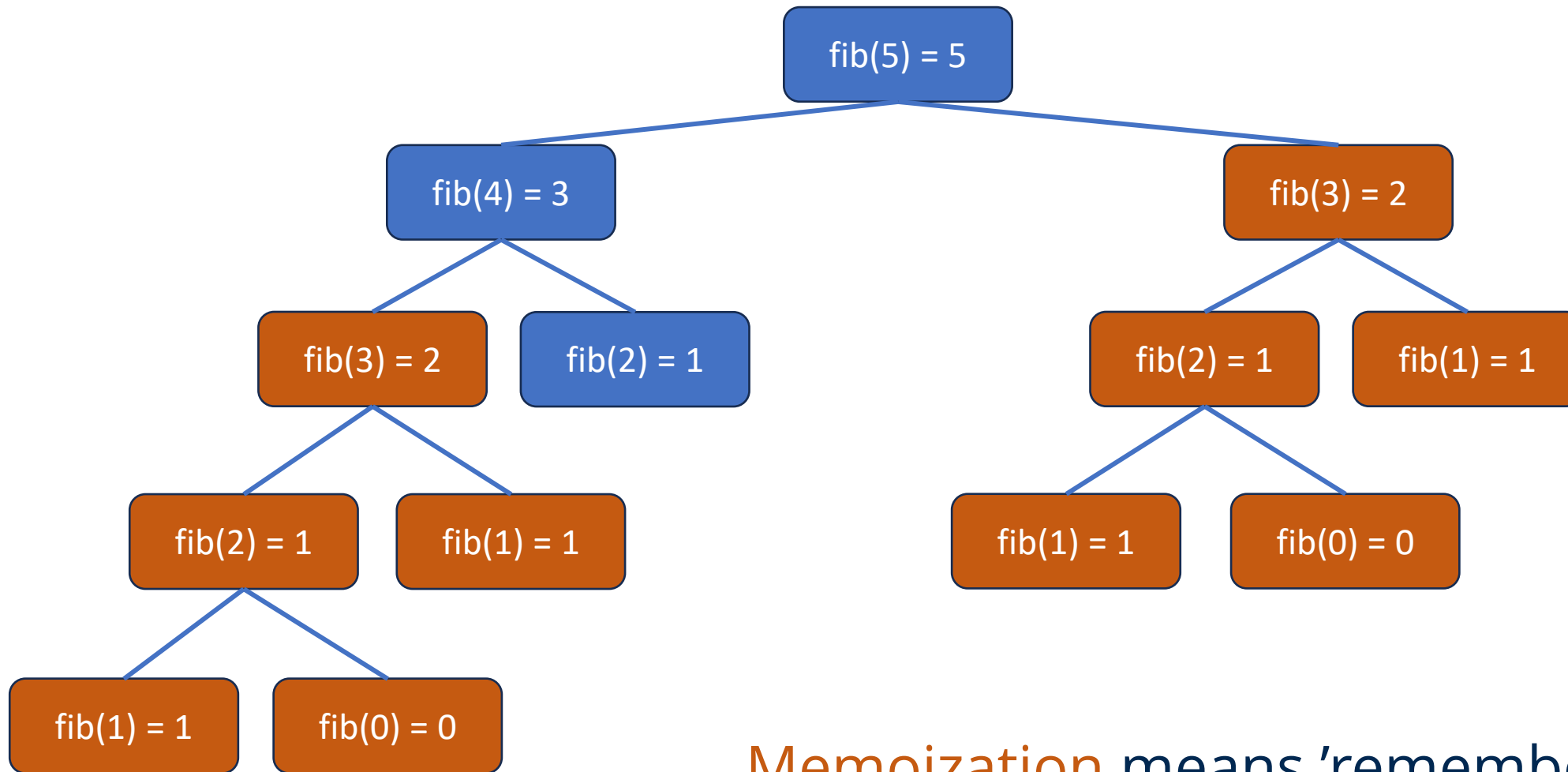
Naive approach



Naive approach



Naive approach



Memoization means 'remember' in Latin

DP approach

```
memo = {}

def fib(n):
    if n in memo:
        return memo[n]

    if n <= 1:
        result = 1
    else:
        result = fib(n - 1) + fib(n - 2)

    memo[n] = result
    return result
```


DP approach

```
memo = {}

def fib(n):
    if n in memo:
        return memo[n]

    if n <= 1:
        result = 1
    else:
        result = fib(n - 1) + fib(n - 2)

    memo[n] = result
    return result
```

```
print(fib(7))    Output: 13
print(fib(13))   Output: 12586269025
```

Time complexity: $O(n)$

Dynamic Programming (DP)

DP = Recursion + Memoization

Dynamic Programming (DP)

DP = Recursion + Memoization

In Reinforcement Learning:

- Collection of algorithms to compute optimal policies
- Perfect model of the environment is given (MDP)
- Great computational expense



DP + RL

- Use value functions to search for good policy

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

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

Policy Evaluation

- The “prediction problem”
- Compute the state value v_π for an arbitrary policy π

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

where $\pi(a|s)$ is the probability of taking action a in state s under policy π , and the expectations are subscripted by π to indicate that they are conditional on π being followed

Policy Evaluation

- The “prediction problem”
- Compute the state value v_π for an arbitrary policy π

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

If the dynamics of the environment is known the solution is straight forward

where $\pi(a|s)$ is the probability of taking action a in state s under policy π , and the expectations are subscripted by π to indicate that they are conditional on π being followed

Iterative Policy Evaluation

- Chose a random v_0 (Terminal states value is 0)
- Use Bellman equitation as update rule

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

where $\pi(a|s)$ is the probability of taking action a in state s under policy π , and the expectations are subscripted by π to indicate that they are conditional on π being followed

Pseudo code

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

Input π , 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 $\Delta < \theta$

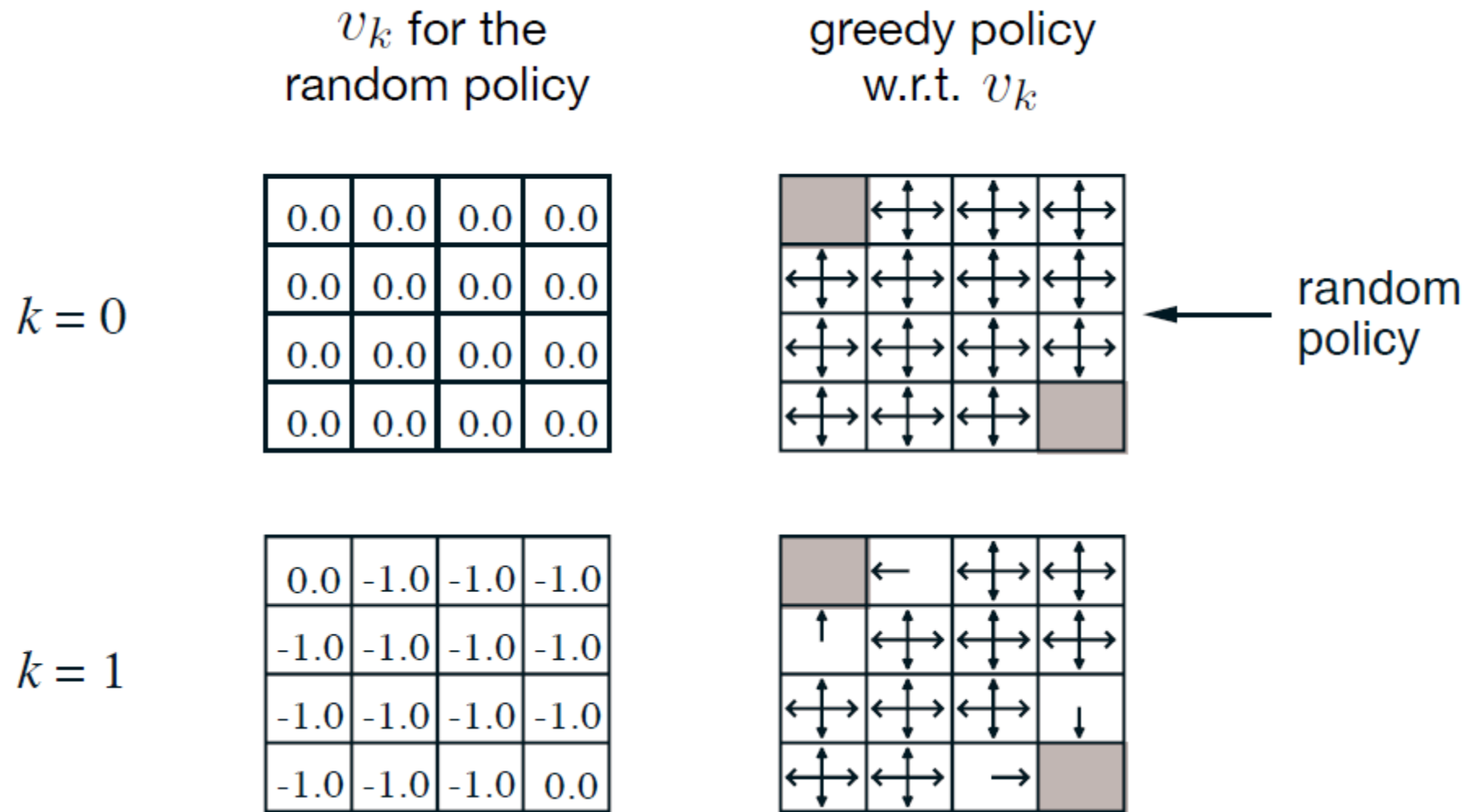
Update rule

- Iterative policy evaluation

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

$$v_k \rightarrow v_{\pi} \text{ if } k \rightarrow \infty$$

Gridworld example




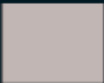
Gridworld example

$k = 2$

v_k for the
random policy



| | | | |
|------|------|------|------|
| 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 |

greedy policy
w.r.t. v_k

| | | | |
|---|-----|-----|---|
|  | ← | ← | ↔↕↔ |
| ↑ | ↖ | ↔↕↔ | ↓ |
| ↑ | ↔↕↔ | ↘ | ↓ |
| ↔↕↔ | → | → |  |

$k = 3$

| | | | |
|------|------|------|------|
| 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 |

| | | | |
|---|-----|-----|---|
|  | ← | ← | ↖ |
| ↑ | ↖ | ↔↕↔ | ↓ |
| ↑ | ↔↕↔ | ↘ | ↓ |
| ↖ | → | → |  |

Gridworld example

$k = 10$

v_k for the
random policy

| | | | |
|------|------|------|------|
| 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 |

greedy policy
w.r.t. v_k

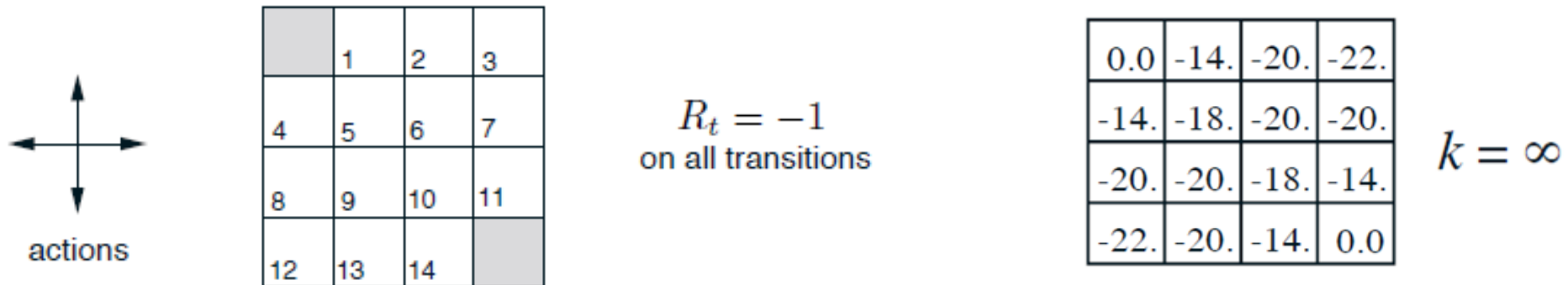
| | | | |
|---|---|---|---|
| | ← | ← | ↙ |
| ↑ | ↖ | ↙ | ↓ |
| ↑ | ↗ | ↘ | ↓ |
| ↖ | → | → | |

| | | | |
|---|---|---|---|
| | ← | ← | ↙ |
| ↑ | ↖ | ↙ | ↓ |
| ↑ | ↗ | ↘ | ↓ |
| ↖ | → | → | |

optimal
policy

Gridworld example

Undiscounted, deterministic, episodic task



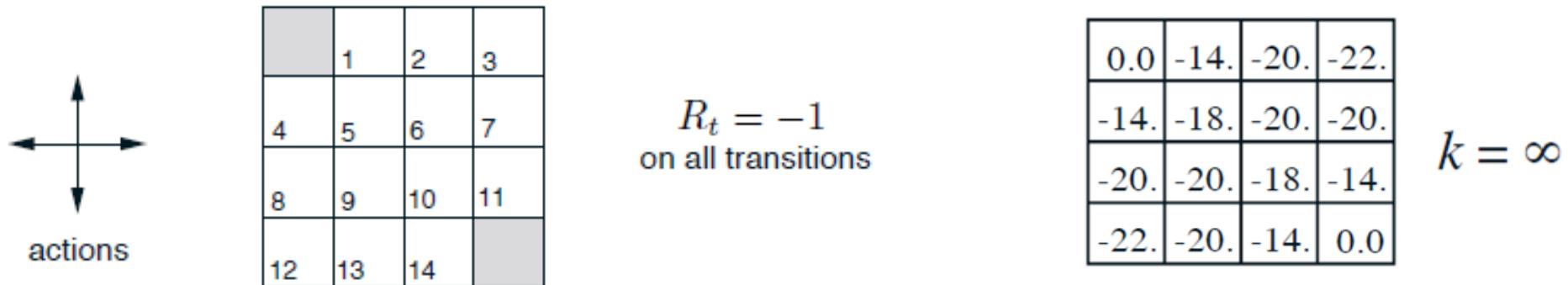
If π is the equiprobable random policy, what is

$$q_{\pi}(11, \text{down}) =$$

$$q_{\pi}(7, \text{down}) =$$

Gridworld example

Undiscounted, deterministic, episodic task



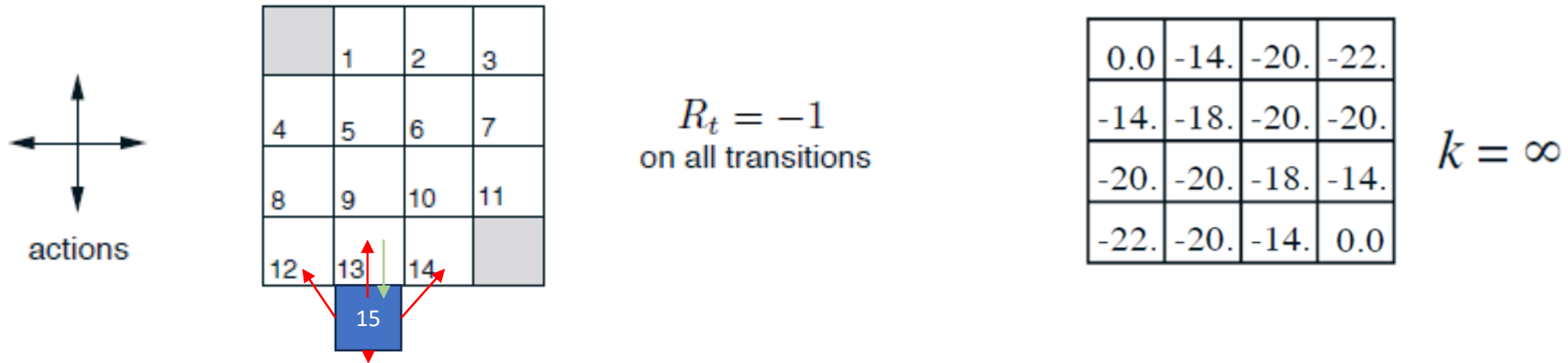
If π is the equiprobable random policy, what is

$$q_{\pi}(11, \text{down}) = -1 + v_{\pi}(\mathcal{T}) = -1 + 0 = -1$$

$$q_{\pi}(7, \text{down}) = -1 + v_{\pi}(11) = -1 + (-14) = -15$$

Gridworld example

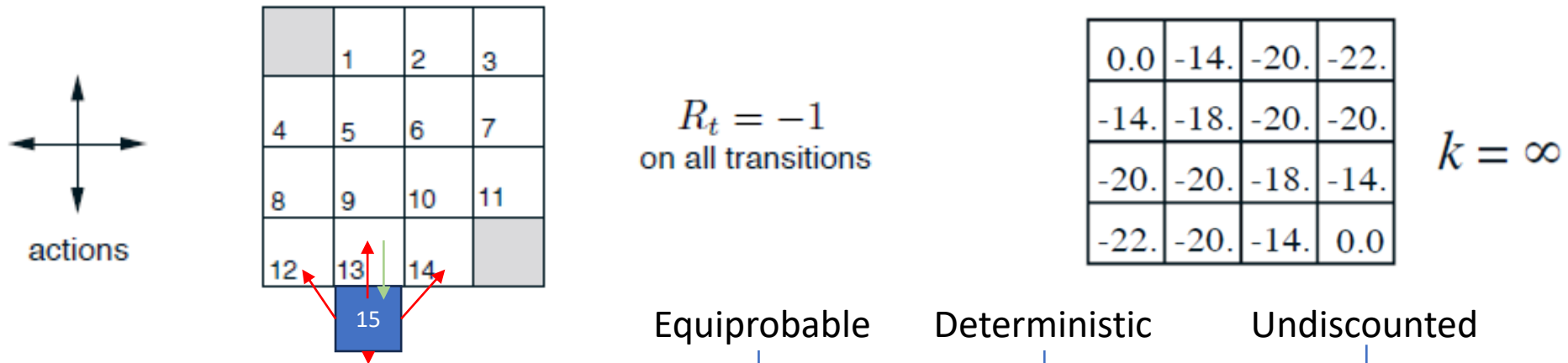
Undiscounted, deterministic, episodic task



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 $v_\pi(15)$ 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 $v_\pi(15)$ for the equiprobable random policy in this case?

Gridworld example

Undiscounted, deterministic, episodic task



Equiprobable Deterministic Undiscounted

$$v_{\pi}(15) = \sum_a \underbrace{\pi(a|s)}_{0.25} \sum_{s', r} \underbrace{p(s', r | s, a)}_1 [\underbrace{r}_{-1} + \underbrace{\gamma}_1 v_{\pi}(s')] = 0.25(-1 - 22 - 1 - 20 - 1 - 14 - 1 + v_{\pi}(15))$$

Gridworld example

$$\begin{aligned}v_{\pi}(15) &= 0.25 [(-1 + -20) + (-1 + -22) + (-1 + -14) + (-1 + v_{\pi}(15))] \\&= 0.25 [(-60 + v_{\pi}(15))] \\&= -15 + 0.25 [v_{\pi}(15)]\end{aligned}$$















and we know $v_{\pi}(15) == v_{\pi}(13) == -20$ as the state transitions and value functions at next state are identical. Therefore we get:

$$\begin{aligned}v_{\pi}(15) &= -15 + 0.25 [-20] \\&= -20\end{aligned}$$

If the dynamics are changed such one can transition from state 13 into 15, the characteristic of the MDP are unchanged. Moving into state 15, which has the same value as state 13 and the same subsequent dynamics, is identical to returning back to state 13 - as was the case previously. The value function is therefore unchanged and $v_{\pi}(15) = -20$

Policy Improvement

Policy π















| | | | |
|---|---|---|---|
| |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  | |

$k = 2$

$v_{\pi}(s)$

| | | | |
|------|------|------|------|
| 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 |

New policy

| | | | |
|---|---|---|---|
| |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  | |

- We know how good to follow the current policy π
- Would it be better to change to a new?

Policy Improvement

- If we select a in s according to π

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

- What if we select a in s according to π'
 - π' is identical to π except that $\pi'(s) = a \neq \pi(s)$

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

$$v_{\pi'}(s) \geq v_{\pi}(s)$$

If it is true we
have found a
better policy

Proof of the policy improvement theorem

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

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

Take one action according to π'

Redefine the expected value

Recursion

Recursion q def

Expand state value with discounted cumulative reward

Policy Improvement

- So far, we have evaluated a change in the policy
- Extend the idea, to all states and all possible actions
- Select a that is best according to $q_{\pi}(s,a)$
- Consider this the new greedy policy π'

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

Policy π and $v_{\pi}(s)$

| | | | | |
|--|------|------|------|------|
| | 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 |



New greedy π'

| | | | |
|---|---|---|---|
| | ← | ← | ↕ |
| ↑ | ↖ | ↕ | ↓ |
| ↑ | ↕ | ↗ | ↓ |
| ↕ | → | → | |

Policy Improvement

- The process of making a new policy that improves on an original policy, by making it greedy with respect to the value function of the original policy is called **policy improvement**
- Suppose if the new policy is as good as, but not better than the original policy

$$v_{\pi} = v_{\pi'} = v_{*}$$

Must be optimal policies

Policy Iteration

- Sequence of monotonically improving policies and value functions

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

E – evaluation

I – improvement

Finite MDP has only a finite number of policies, so this process must converge to an optimal policy and optimal value function

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

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 $s \in \mathcal{S}$:

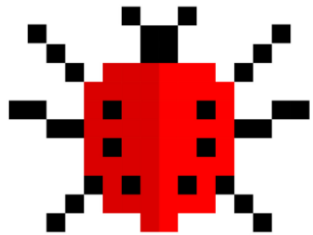
old-action $\leftarrow \pi(s)$

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

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

Where is
the bug?



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

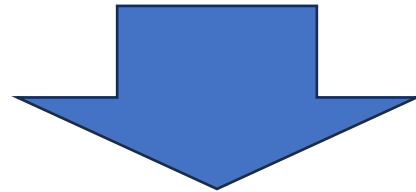
If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Problem:
Oscillation may occur

if $v_{\pi'}(s) = v_{\pi}$; *policy-stable* \leftarrow true

Policy Iteration - drawback

- Use policy evaluation in each step
 - Computational and time consuming
- Truncate the evaluation step
 - Stop after one sweep of evaluation



Value Iteration

Value Iteration

- Policy evaluation in the iterative policy improvement is stopped after one sweep (one update of each state)

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

Pseudo code

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

Output a deterministic policy, $\pi \approx \pi_*$, such that
$$\pi(s) = \operatorname{argmax}_a \sum_{s',r} p(s',r|s,a) [r + \gamma V(s')]$$

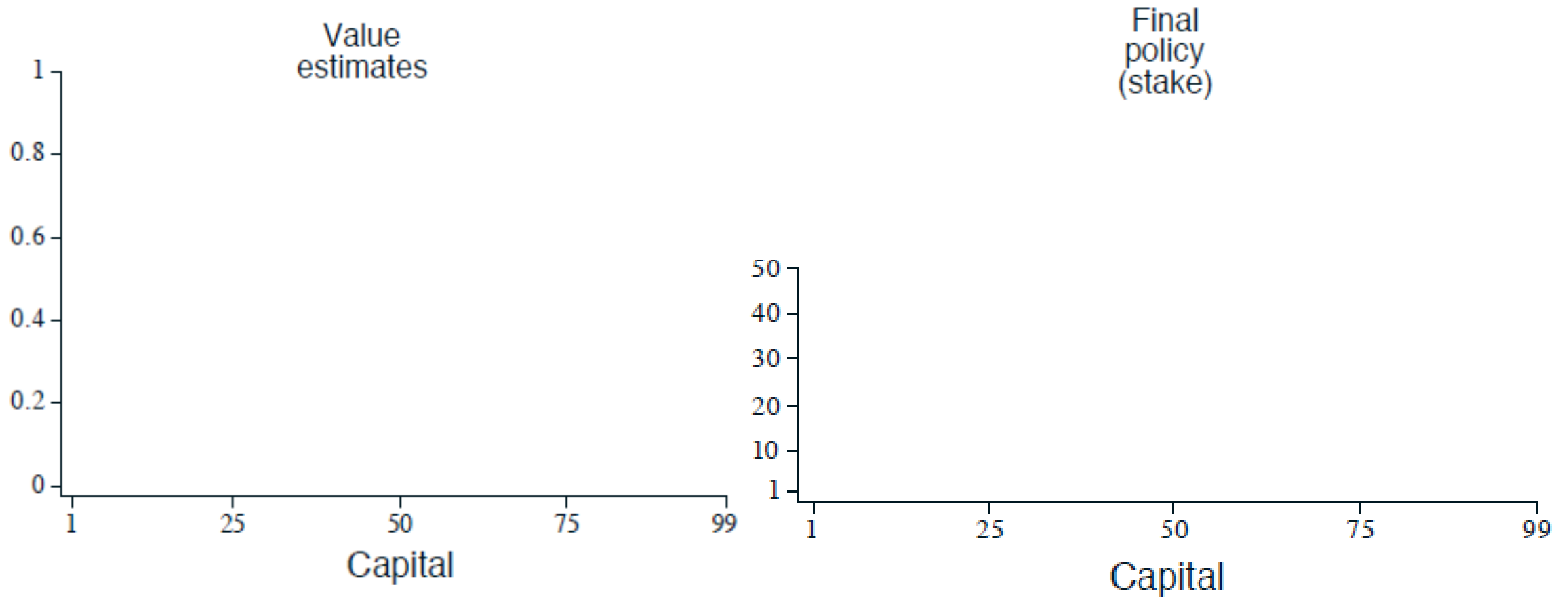
Gambler's problem

A gambler has the opportunity to make bets on the outcomes of a sequence of coin flips. If the coin comes up heads, he wins as many dollars as he has staked on that flip; if it is tails, he loses his stake. The game ends when the gambler wins by reaching his goal of \$100, or loses by running out of money. On each flip, the gambler must decide what portion of his capital to stake, in integer numbers of dollars. This problem can be formulated as an undiscounted, episodic, finite MDP. The state is the gambler's capital's $s \in \{1, 2, \dots, 99\}$ and the actions are stakes, $a \in \{0, 1, \dots, \min(s, 100-s)\}$. The reward is zero on all transitions except those on which the gambler reaches his goal, when it is +1. The state-value function then gives the probability of winning from each state. A policy is a mapping from levels of capital to stakes. The optimal policy is a mapping from levels of capital to stakes. The optimal policy maximizes the probability of reaching the goal. Let p_h denote the probability of the coin coming up heads. If p_h is known, then the entire problem is known and it can be solved, for instance, by value iteration.

Can you guess what the optimal value function and optimal policy looks like?

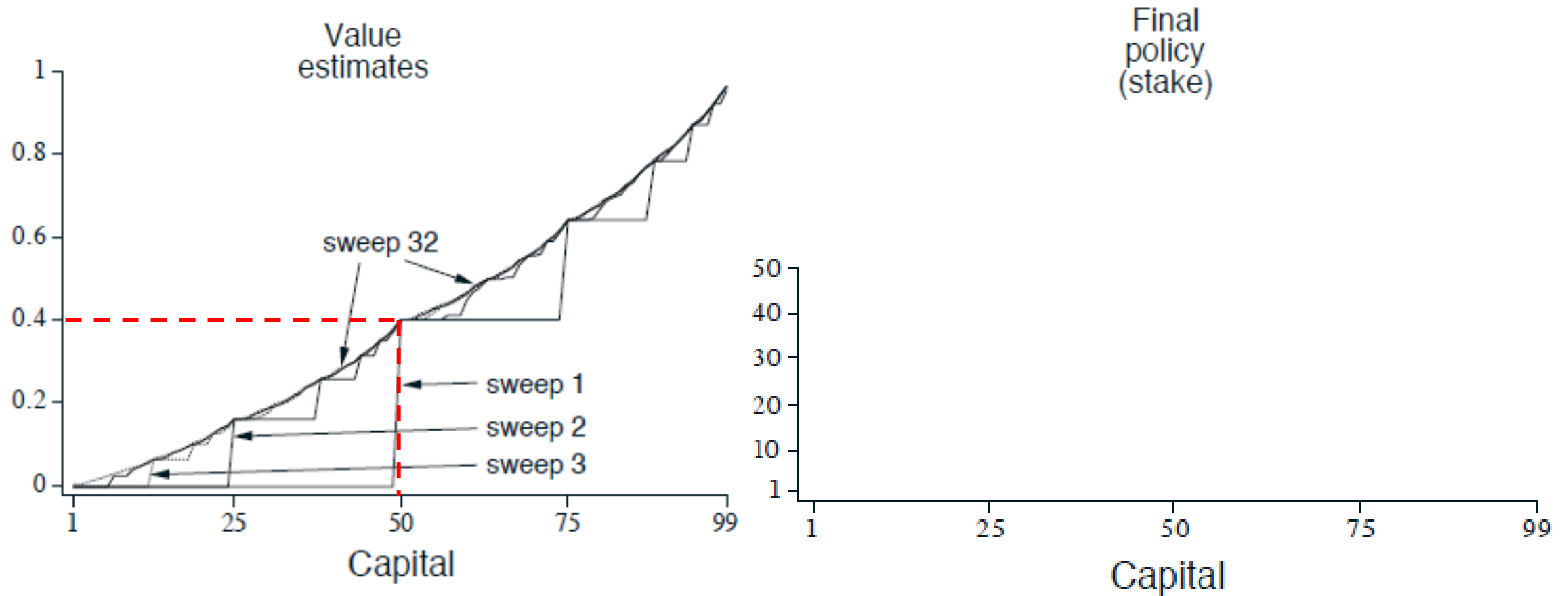
Gambler's problem

- $P(H) = 0.4$



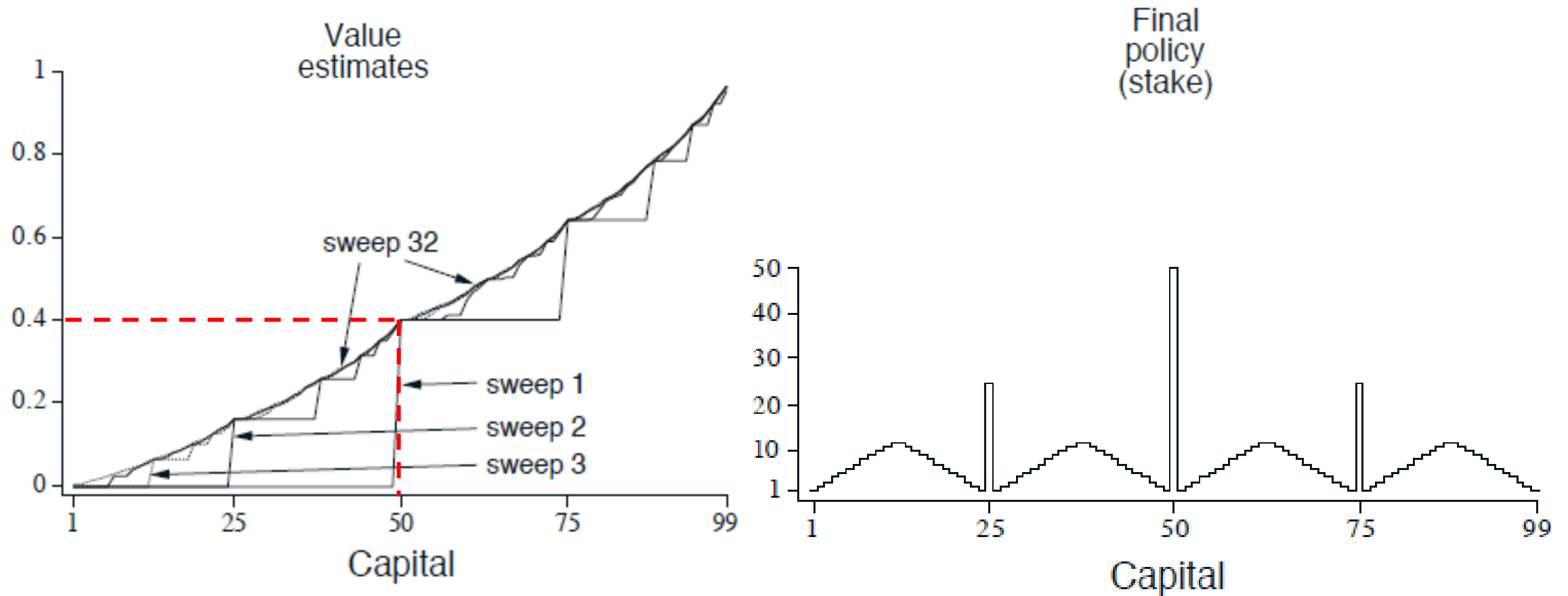
Gambler's problem

- $P(H) = 0.4$



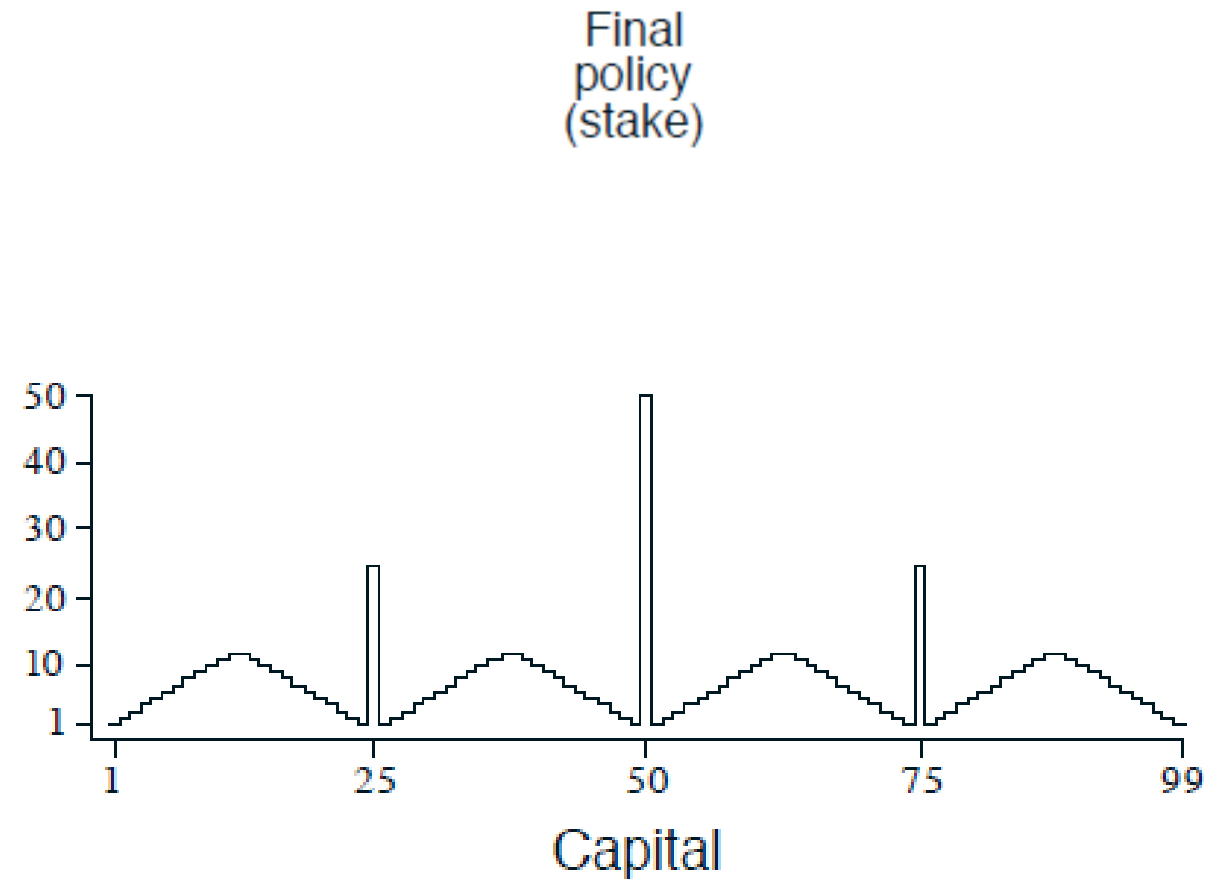
Gambler's problem

- $P(H) = 0.4$



Gambler's problem

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?



Gambler's problem

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?

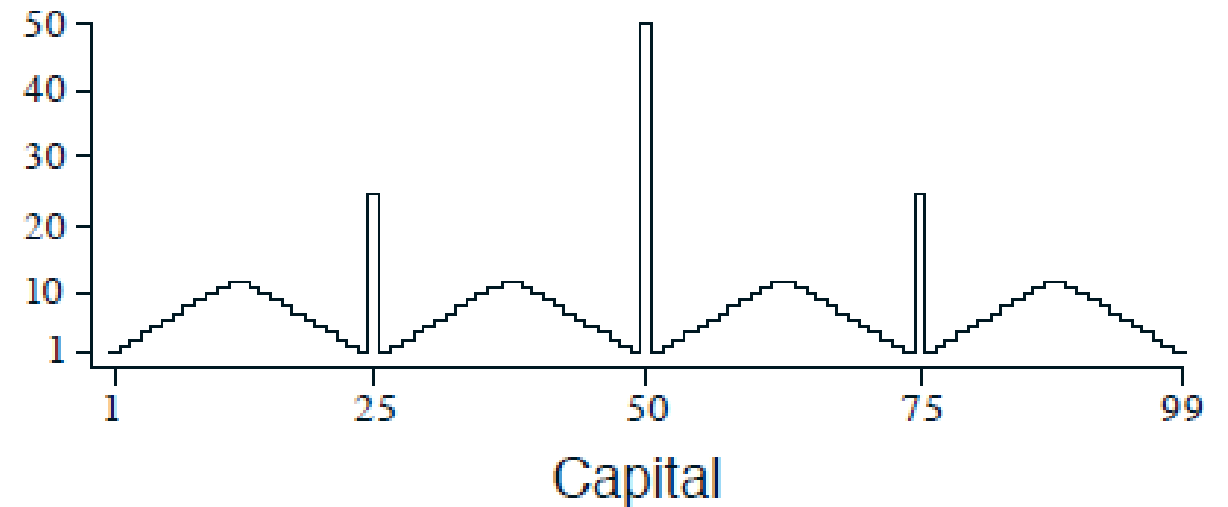
At 50 it has a chance to win it all with p_h probability.

Thinking capital of 51 as $50 + 1$.

The best policy is to see if we can earn much from that extra 1 dollar.

If we bet 50 out of 51 first the win probability is p_h and we lose the extra chance. (to reach 75, and 100).

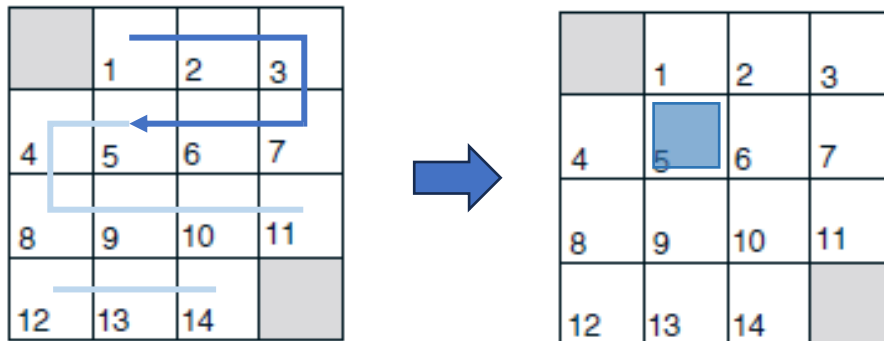
Final
policy
(stake)



Asynchronous DP

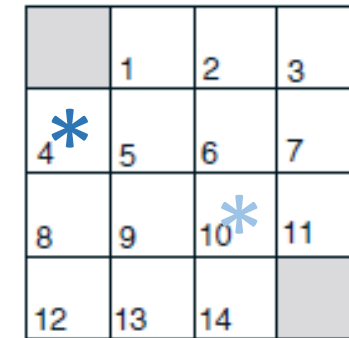
Synchronous update

- Require sweeps of the state set (*expected update*)
- Time consuming



Asynchronous update

- In place iterative DP
- Not organized into systematic sweeps
- Flexible
- Can not ignore states



It can even focus on specific state sets...

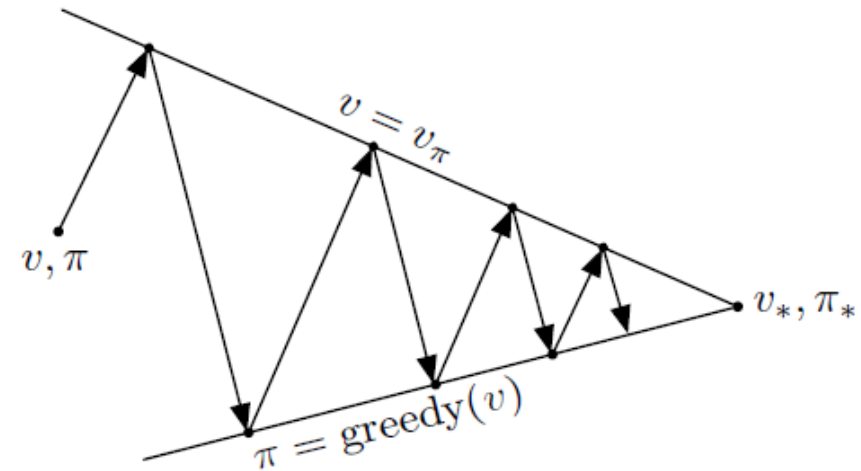
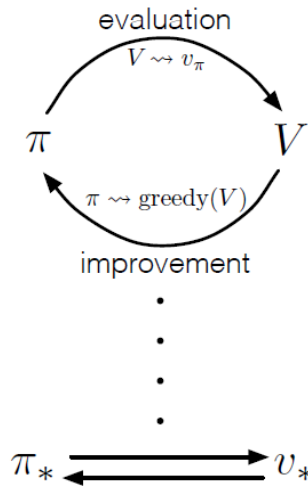
Generalized Policy Iteration (GPI)

- **Policy evaluation:** making the value function consistent with the current policy
- **Policy improvement:** making the policy greedy with respect to the current value function

Generalized Policy Iteration (GPI)

- **Policy evaluation:** making the value function consistent with the current policy
- **Policy improvement:** making the policy greedy with respect to the current value function

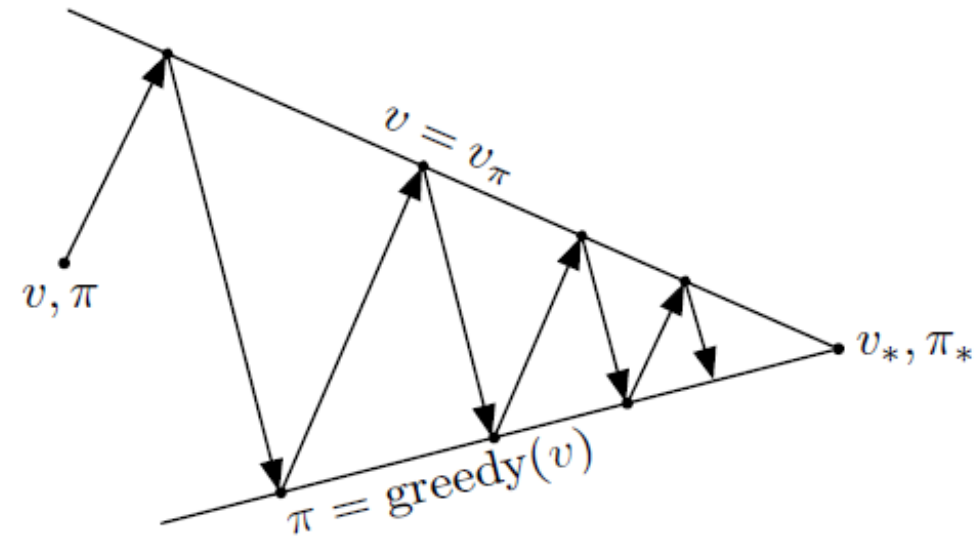
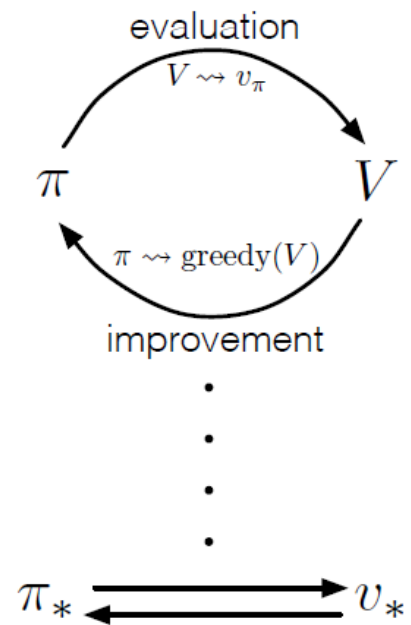
2 interacting processes revolving around an approximate policy and an approximate value function



If evaluation process and improvement process stabilize (no longer produce changes)
then the value function and policy must be optimal

Generalized Policy Iteration (GPI)

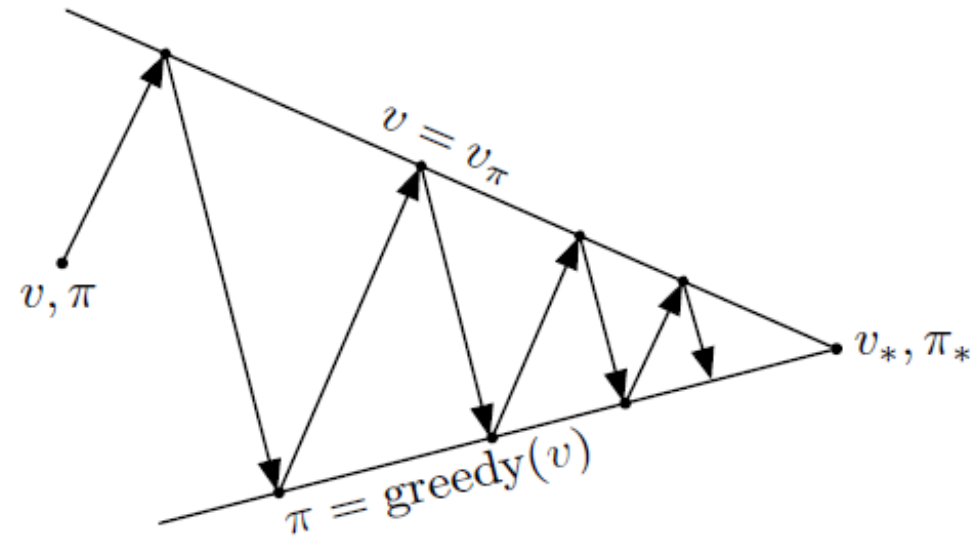
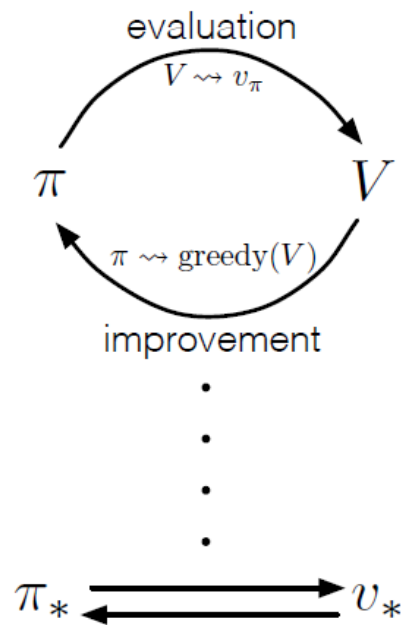
- Does it necessary to complete one before the other?



If evaluation process and improvement process stabilize (no longer produce changes)
then the value function and policy must be optimal

Generalized Policy Iteration (GPI)

- Does it necessary to complete one before the other?
 - NO, It can be approximation (or Asynchronous)



If evaluation process and improvement process stabilize (no longer produce changes)
then the value function and policy must be optimal

Efficiency of DP

- Not practical for large problems (curse of dimensionality)
- Faster than direct search or linear programming
- Update estimates on the basis of other estimates = **bootstrapping**
- **Require model** of the environment



ELTE

FACULTY OF
INFORMATICS

Thank you for your attention!