Dynamic programming

Herke van Hoof

Plan for today

Last lecture, we talked about the exploration/exploitation tradeoffs in bandits and MDPs as model for decision making

Today, we'll define **optimal behaviour**, and see how we can find optimal policies for **known MDPs**

Markov decision processes

Last lecture, we introduced MDPs, formally consisting of:

- A finite set of states
- A finite set of actions for each state (often the same in all states)
- A dynamics function

$$p(s', r|s, a) \doteq \Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}$$

Sometimes written as: transition function $p(s'|s, a)$

reward function $p(r|s, a, s')$

- A discount factor $\gamma \in [0,1)$

Returns

We also introduced the notion of **return**

(only for episodic tasks)

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

As well as **discounted return**

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

(continuing or episodic tasks)

Unified notation

Different notation for episodic and continuing tasks is a bit awkward.

We will always use the discounted return, but allow $\gamma=1$ if every episode terminates

How to find optimal policy & value fcs?

Thanks to Jan Peters

How to find optimal policy & value fcs?

You have won an award in Madrid!

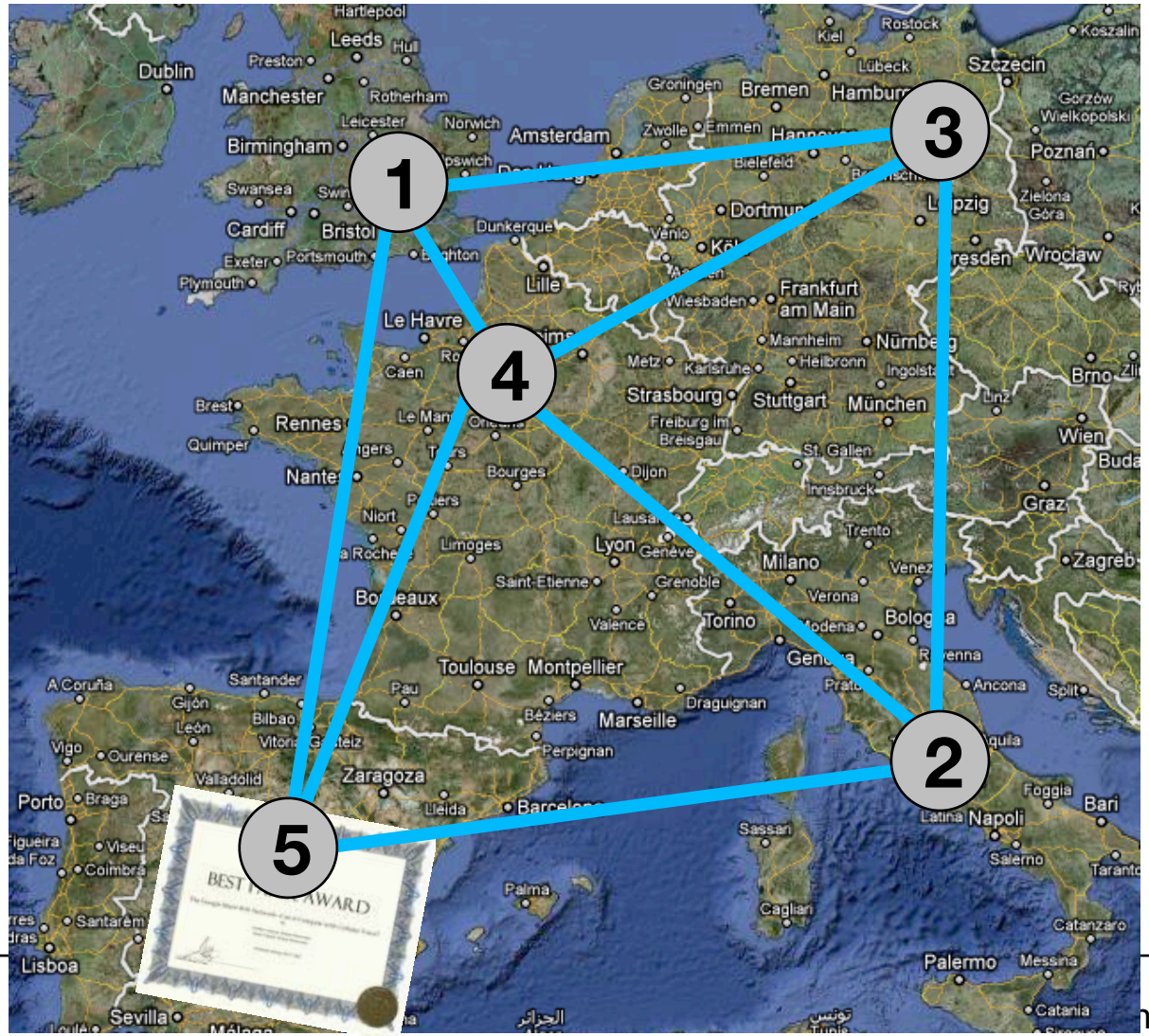
What is the optimal policy to collect it?

Thanks to Jan Peters

How to find optimal policy & value fcs?

You have won an award in Madrid!

What is the optimal policy to collect it?

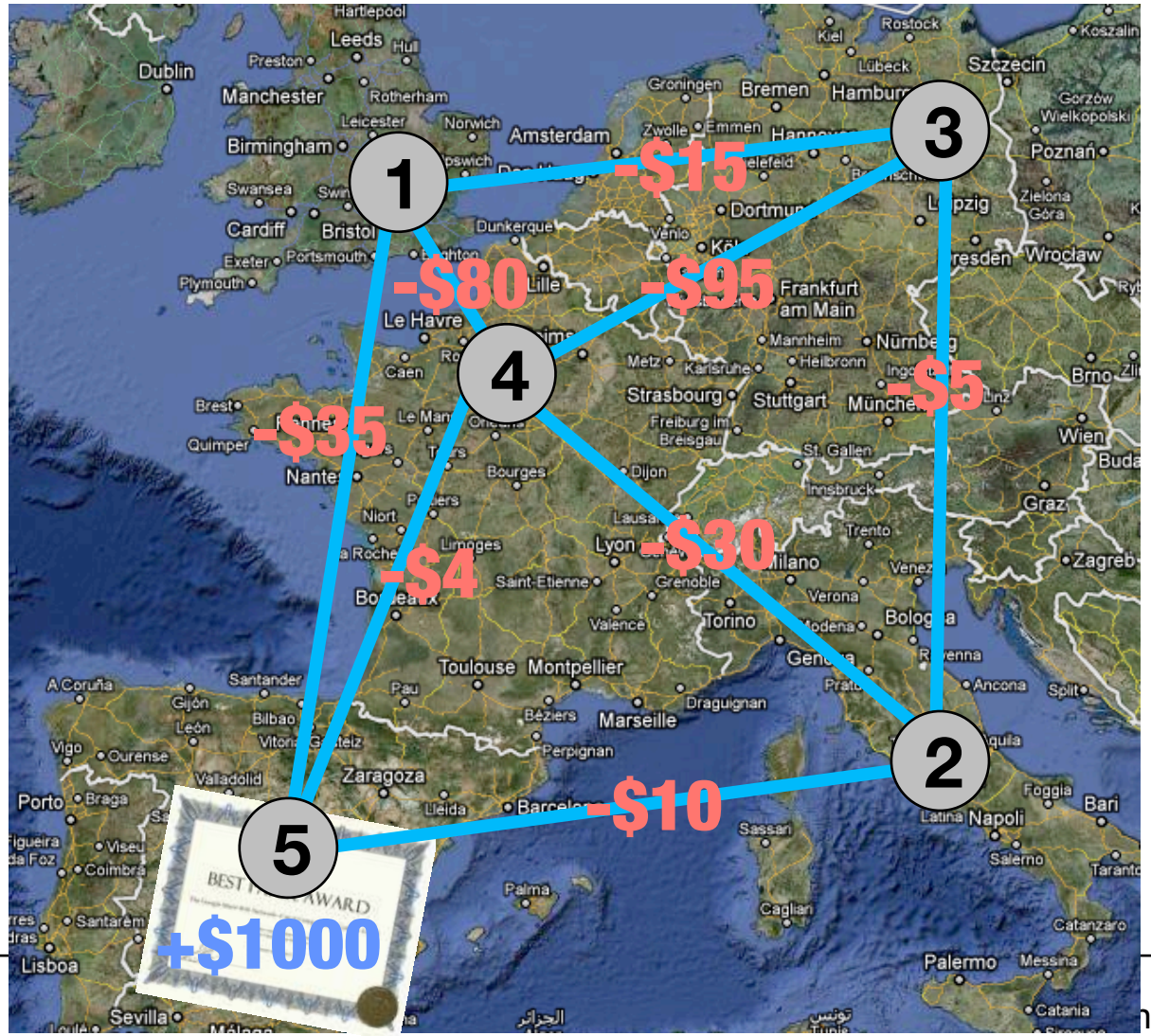


Thanks to Jan Peters

How to find optimal policy & value fcs?

You have won an award in Madrid!

What is the optimal policy to collect it?



Thanks to Jan Peters

Policies and value functions

Our goal is to figure out how to take good actions

The way actions are selected is called the policy $\pi(a|s)$

For every state, the policy specifies a probability distribution over actions

Of course, there are many possible policies. **Which is best?**

Policies and value functions

A policy should get high expected (discounted) returns

The state-value function expresses how good π is from *s*:

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

So, if we start from *s*, we might prefer π over π' if

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

We'll see later how *v* can be computed or learned!

Policies and value functions

Of course, for a fixed policy the value function also measures how good it is for the agent to be in a **state**

Similarly, we might wonder how good it is to be in a (state,action) pair. We'll express this with a state-action value function

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

Similar to the bandit case, we could use *q* to select actions e.g.: greedy policy: $a^* = \arg\max_a q_\pi(s,a)$

Policies and value functions

Finally we can define optimal policies

A policy is optimal if

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

Multiple policies can be optimal, share optimal value functions:

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

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

We'll see how to learn these, too…

Bellman's principle of optimality

"An optimal sequence of controls in a multistage optimization problem has the property that whatever the initial stage, state and controls are, the remaining controls must constitute an optimal sequence of decisions for the remaining problem with stage and state resulting from previous controls considered as initial conditions"

Richard Bellman, Dynamic Programming, 1957

Thanks to Jan Peters

Bellman's principle of optimality

"An optimal sequence of controls in a multistage optimization problem has the property that whatever the initial stage, state and controls are, the remaining controls must constitute an optimal sequence of decisions for the remaining problem with stage and state resulting from previous controls considered as initial conditions"

Richard Bellman, Dynamic Programming, 1957

Bellman's quote suggests trying to first find the last controls, and working backwards.
A solution for the last step will be part of any solution for more than one step. Then 2 steps, etc.
This follows the "dynamic programming" principle.

Policies and value functions

To follow Bellman's recipe, consider relationship between subsequent v-functions:

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$$

$$=$$

$$=$$

$$=$$

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$$

$$=$$

$$=$$

Similar identities hold for q

Policies and value functions

To follow Bellman's recipe, consider relationship between subsequent v-functions:

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

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi[G_t|S_t = s] \\
&= \mathbb{E}_{a\sim\pi}\left[\mathbb{E}_\pi[G_t|S_t = s, A_t = a]\right] \\
&= \mathbb{E}_{a\sim\pi} q_\pi(s, a)
\end{aligned}
$$

Similar identities hold for q

Policies and value functions

Optimal value functions are also related:

$$q_*(s,a) = \qquad v_*(S_{t+1})$$

$$v_*(s) = \quad q_*(s,a)$$

$$= \qquad\qquad v_*(S_{t+1})$$

$$q_*(s,a) = \qquad\qquad q_*(S_{t+1}, a')$$

Policies and value functions

Optimal value functions are also related:

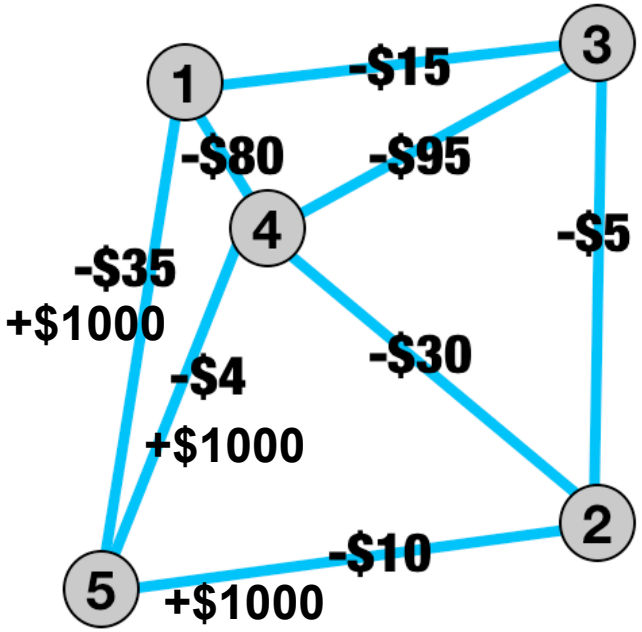
$$q_*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma v_*\left(S_{t+1}\right) | S_t = s, A_t = a\right]$$

$$\begin{aligned} v_*(s) &= \max_a q_*(s,a) \\ &= \max_a \mathbb{E}\left[R_{t+1} + \gamma v_*\left(S_{t+1}\right) | S_t = s, A_t = a\right] \end{aligned}$$

$$q_*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*\left(S_{t+1}, a'\right) | S_t = s, A_t = a\right]$$

Last two equations called *Bellman optimality equations*, and show how value of subsequent states / (s,a) pairs is related

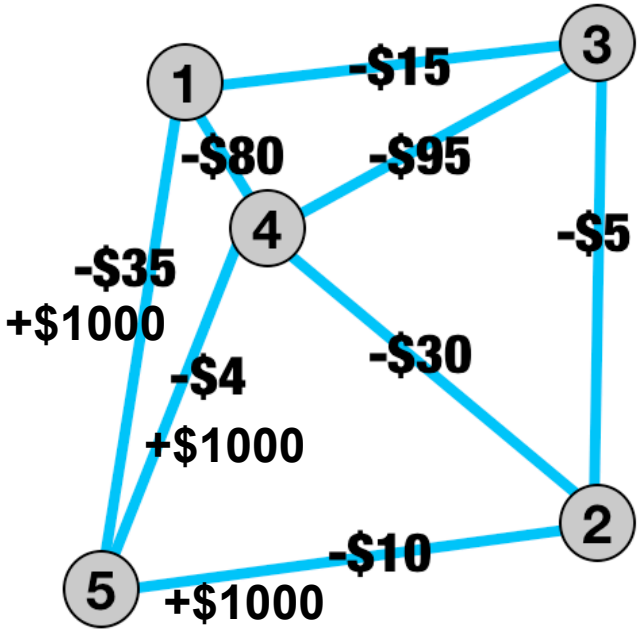
Let's try



	V_4	V_3	V_2	V_1	V_0	
						1
						2
						3
						4
						5

Thanks to Jan Peters

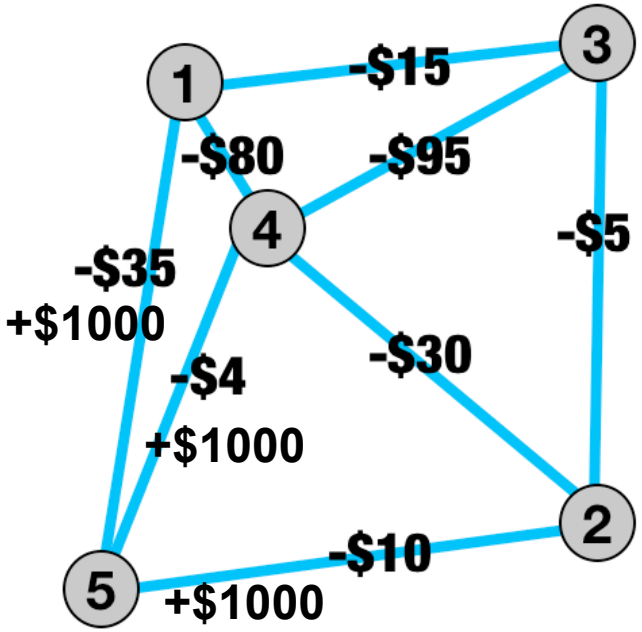
Let's try



	V_4	V_3	V_2	V_1	V_0	
					0	1
					0	2
					0	3
					0	4
					0	5

Thanks to Jan Peters

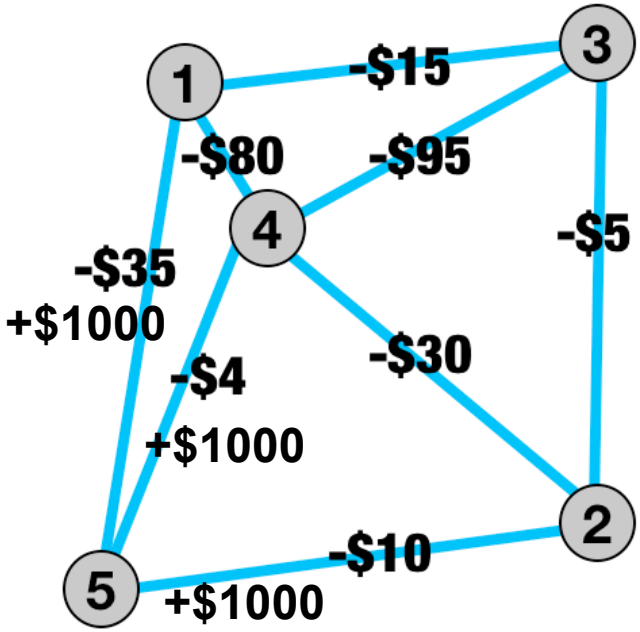
Let's try



	V_4	V_3	V_2	V_1	V_0	
				965	0	1
					0	2
					0	3
					0	4
					0	5

Thanks to Jan Peters

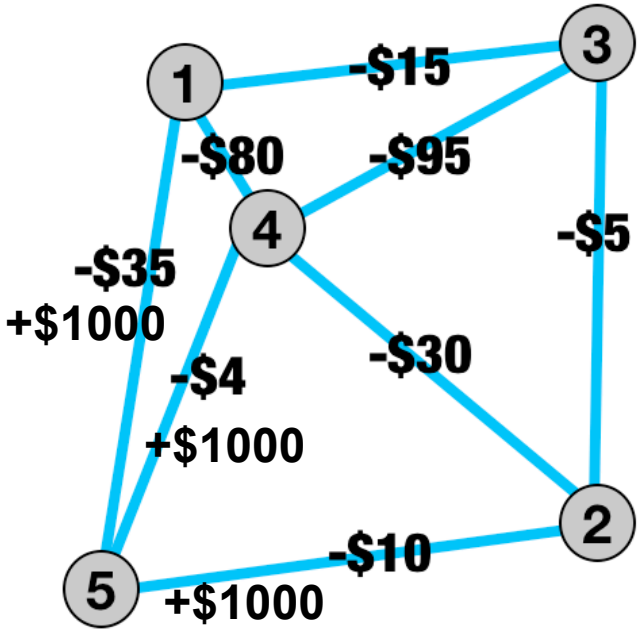
Let's try



	V_4	V_3	V_2	V_1	V_0	
				965	0	1
				990	0	2
					0	3
					0	4
					0	5

Thanks to Jan Peters

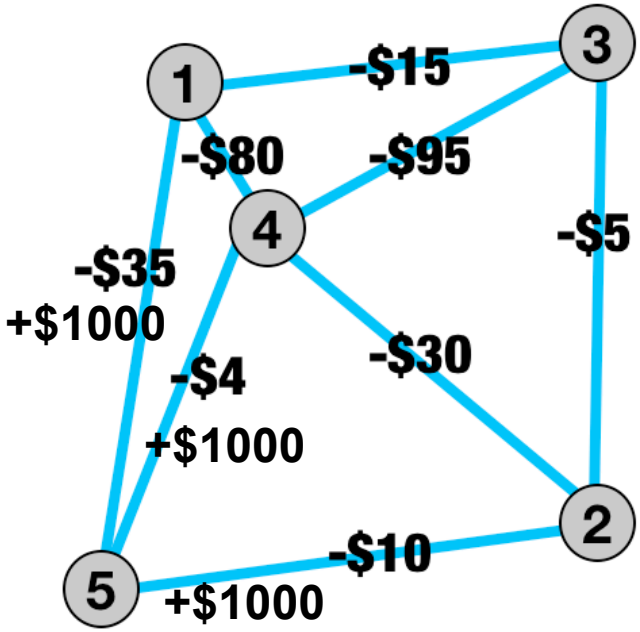
Let's try



	V_4	V_3	V_2	V_1	V_0	
				965	0	1
				990	0	2
				0	0	3
					0	4
					0	5

Thanks to Jan Peters

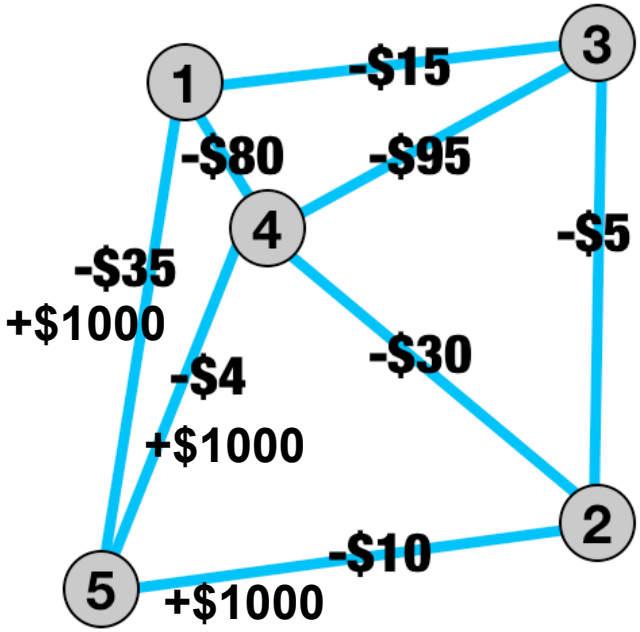
Let's try



	V_4	V_3	V_2	V_1	V_0	
				965	0	1
				990	0	2
				0	0	3
				996	0	4
				0	0	5

Thanks to Jan Peters

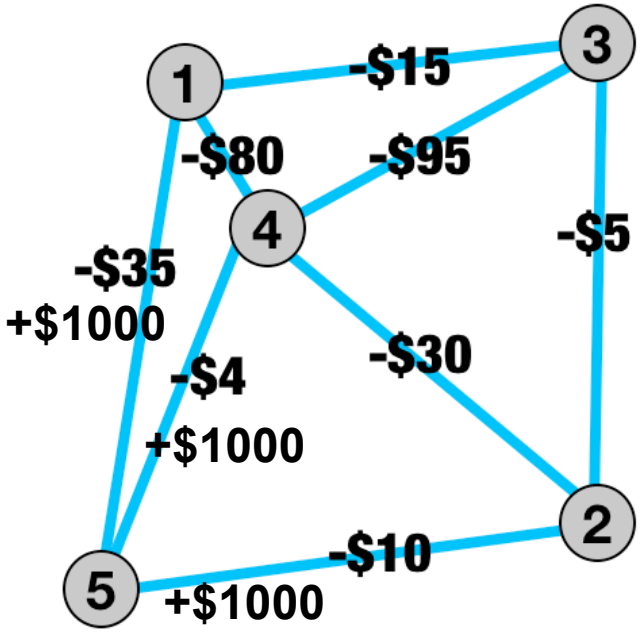
Let's try



	V_4	V_3	V_2	V_1	V_0	
			965	965	0	1
				990	0	2
				0	0	3
				996	0	4
				0	0	5

Thanks to Jan Peters

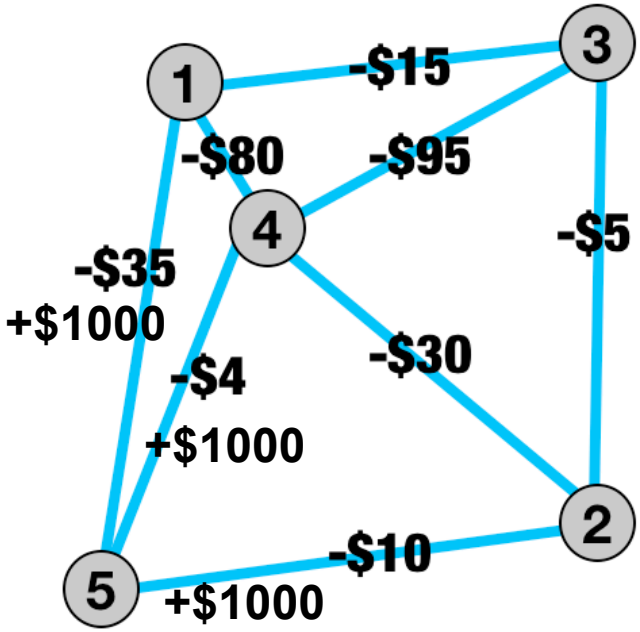
Let's try



	V_4	V_3	V_2	V_1	V_0	
			965	965	0	1
			990	990	0	2
				0	0	3
				996	0	4
				0	0	5

Thanks to Jan Peters

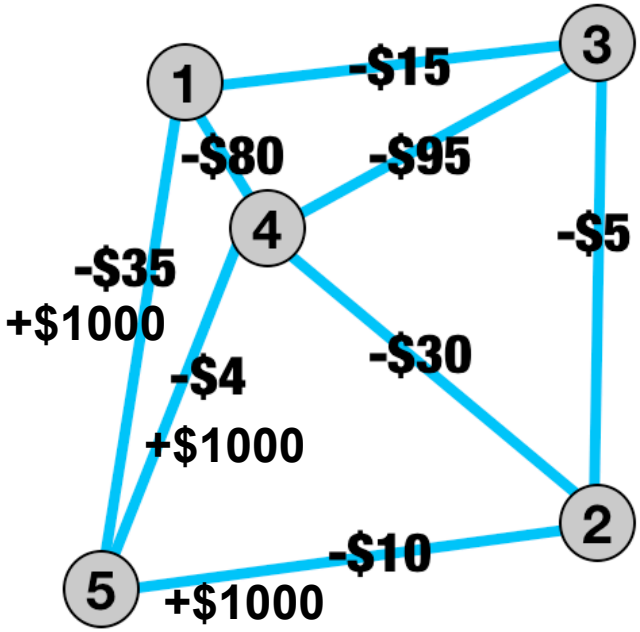
Let's try



	V_4	V_3	V_2	V_1	V_0	
			965	965	0	1
			990	990	0	2
			985	0	0	3
				996	0	4
				0	0	5

Thanks to Jan Peters

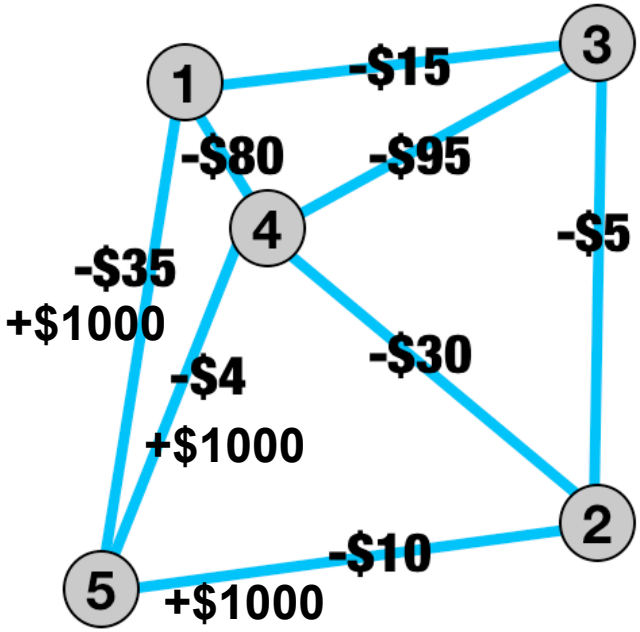
Let's try



	V_4	V_3	V_2	V_1	V_0	
			965	965	0	1
			990	990	0	2
			985	0	0	3
			996	996	0	4
			0	0	0	5

Thanks to Jan Peters

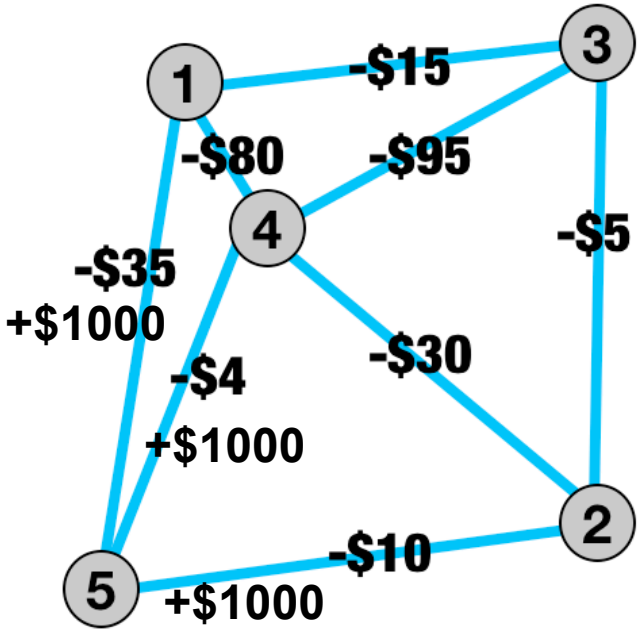
Let's try



	V_4	V_3	V_2	V_1	V_0	
		970	965	965	0	1
			990	990	0	2
			985	0	0	3
			996	996	0	4
			0	0	0	5

Thanks to Jan Peters

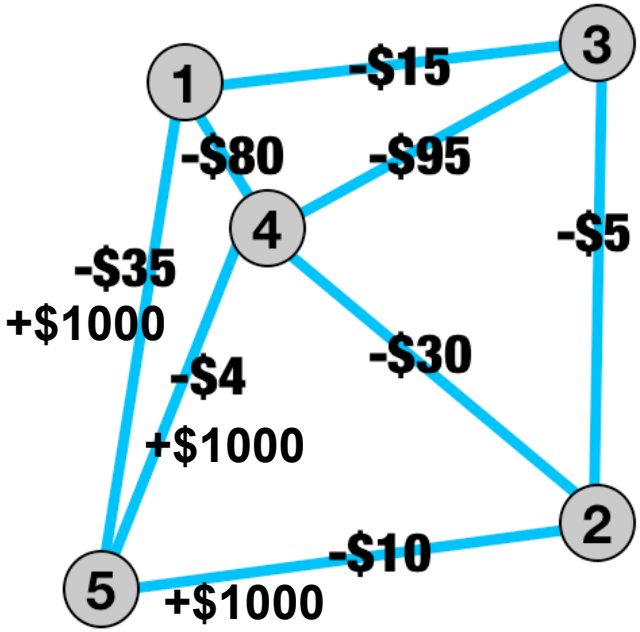
Let's try



	V_4	V_3	V_2	V_1	V_0	
		970	965	965	0	1
		990	990	990	0	2
		985	985	0	0	3
		996	996	996	0	4
		0	0	0	0	5

Thanks to Jan Peters

Let's try



	V_4	V_3	V_2	V_1	V_0	
	970	970	965	965	0	1
	990	990	990	990	0	2
	985	985	985	0	0	3
	996	996	996	996	0	4
	0	0	0	0	0	5

Thanks to Jan Peters

Formalization

1.At the last step, we have the value function

$$V_0^*(\mathbf{x}) = 0$$

2.We compute the optimal policy such that

$$\pi_{t+1}^*(\cdot|\mathbf{s}) = \operatorname{argmax}_\pi \quad [r + V_t^*(\mathbf{s}')]$$
$$\mathbf{a} = \pi(\mathbf{s}), (r, \mathbf{s}') = f(\mathbf{s}, \mathbf{a})$$

3.Obtain next value function

$$V_{t+1}^*(\mathbf{s}) = \max_\pi \quad [r(\mathbf{s}, \mathbf{a}) + V_t^*(\mathbf{s}')]$$

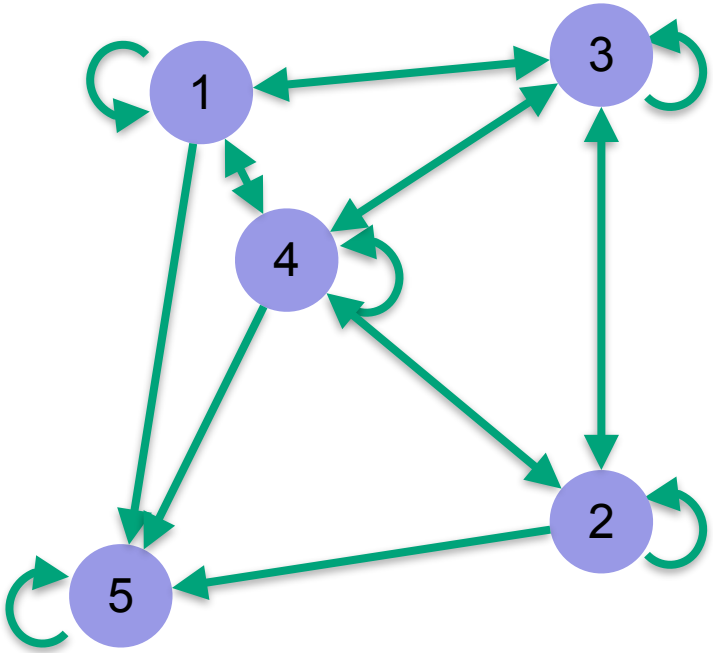
4.If not converged, go back to Step 2.

Thanks to Jan Peters

V_2	V_1	V_0	
$V_2(1)$	$V_1(1)$	$V_0(1)$	1
$V_2(2)$	$V_1(2)$	$V_0(2)$	2
$V_2(3)$	$V_1(3)$	$V_0(3)$	3
$V_2(4)$	$V_1(4)$	$V_0(4)$	4
$V_2(5)$	$V_1(5)$	$V_0(5)$	5

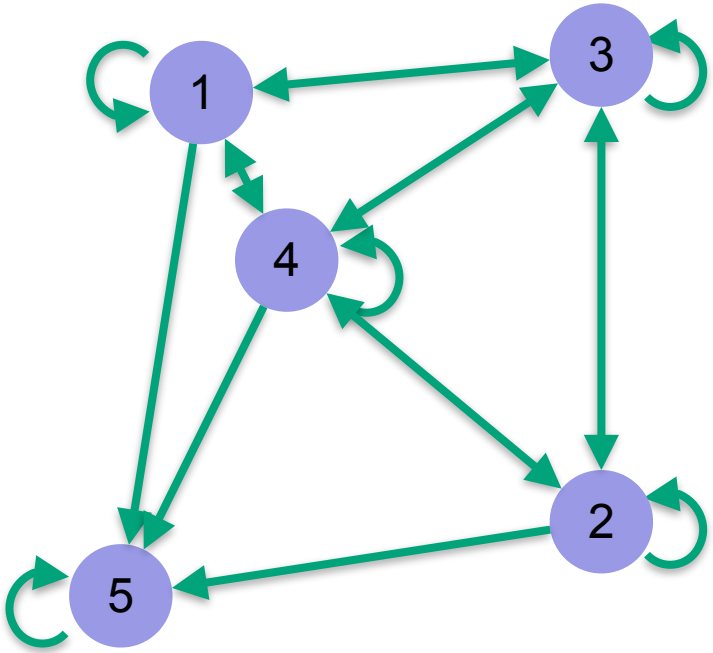
Sidenotes

To simplify, self-loops were not shown in the example and edges were not directed. We can rectify as follows:



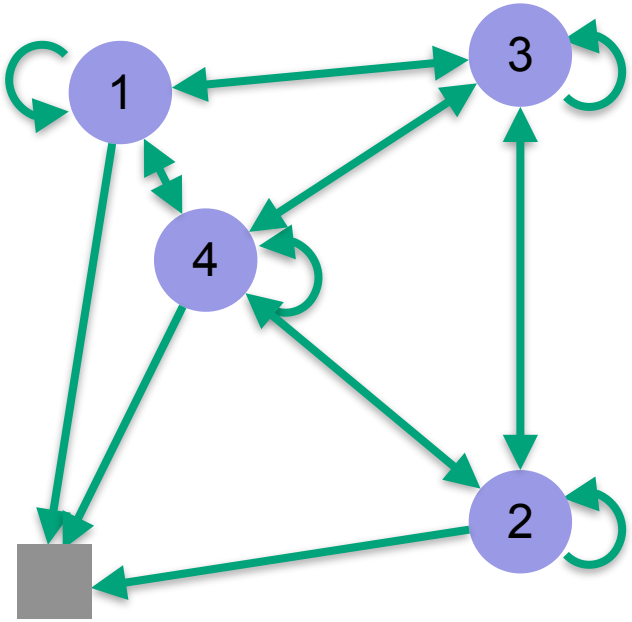
Sidenotes

From state 5 nothing of interest (transition, reward) will happen.
Terminal state, often indicated by square.



Sidenotes

From state 5 nothing of interest (transition, reward) will happen.
Terminal state, often indicated by square.



An episode ‘terminates’ if the terminal state is reached!

Sidenotes

Actions can have stochastic outcomes!
Actions and stochastic transitions can be explicitly shown in the transition graph of an MDP as follows:

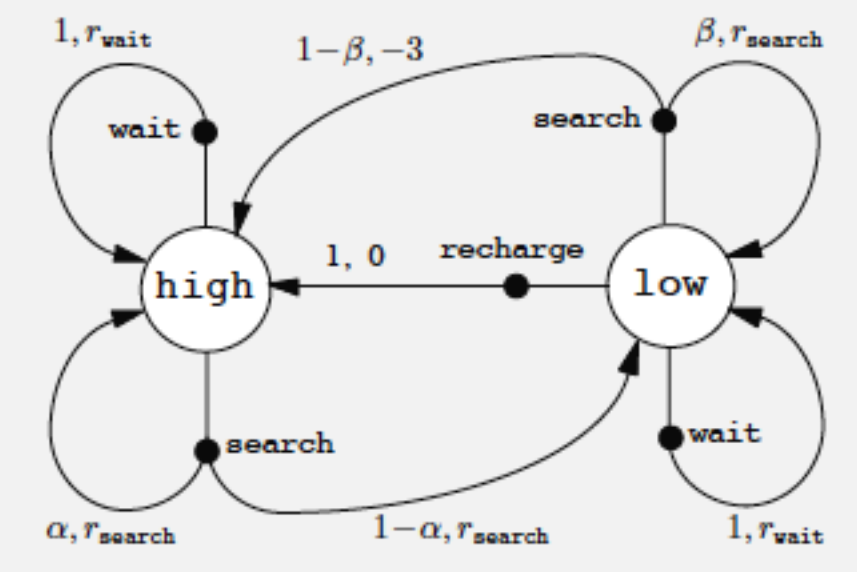
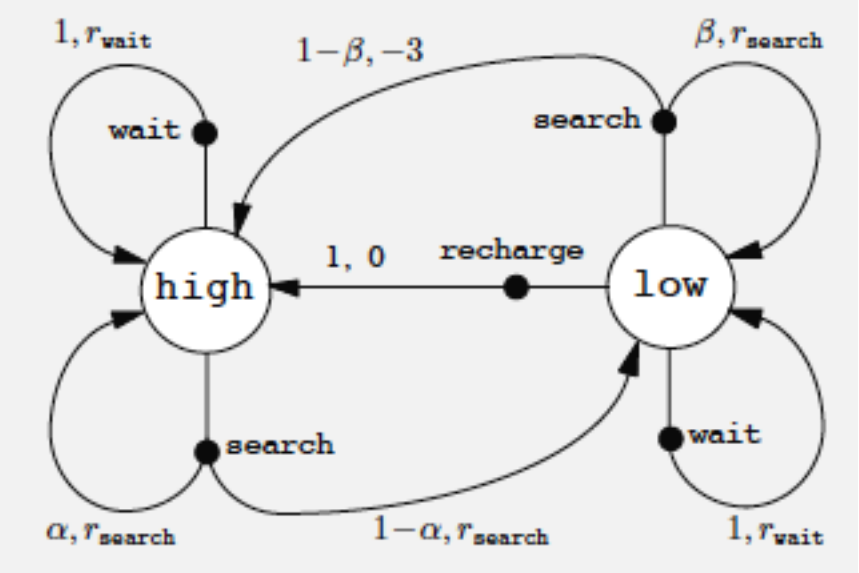


Figure: Sutton&Barto; RL:AI

Sidenotes

Actions can have stochastic outcomes!
Actions and stochastic transitions can be explicitly shown in the transition graph of an MDP as follows:



Can we still find the optimal policy with stochastic outcomes?

Figure: Sutton&Barto; RL:AI

Formalization

1.At the last step, we have the value function

$$V_0^*(\mathbf{x}) = 0$$

2.We compute the optimal policy such that

$$\pi_{t+1}^*(\cdot|\mathbf{s}) = \text{argmax}_\pi \quad [r + V_t^*(\mathbf{s}')]$$

3.Obtain next value function

$$V_{t+1}^*(\mathbf{s}) = \max_\pi \quad [r(\mathbf{s}, \mathbf{a}) + V_t^*(\mathbf{s}')]$$

4.If not converged, go back to Step 2.

Thanks to Jan Peters

V_2	V_1	V_0	
$V_2(1)$	$V_1(1)$	$V_0(1)$	1
$V_2(2)$	$V_1(2)$	$V_0(2)$	2
$V_2(3)$	$V_1(3)$	$V_0(3)$	3
$V_2(4)$	$V_1(4)$	$V_0(4)$	4
$V_2(5)$	$V_1(5)$	$V_0(5)$	5

Formalization

1.At the last step, we have the value function

$$V_0^*(\mathbf{x}) = 0$$

2.We compute the optimal policy such that

$$\pi_{t+1}^*(\cdot|\mathbf{s}) = \text{argmax}_\pi \, \mathbb{E}\left[r + V_t^*(\mathbf{s}')\right]$$

$$\mathbf{a} \sim \pi(\cdot|\mathbf{s}), (r, \mathbf{s}') \sim f(\mathbf{s}, \mathbf{a})$$

3.Obtain next value function

$$V_{t+1}^*(\mathbf{s}) = \max_\pi \mathbb{E}\left[r(\mathbf{s}, \mathbf{a}) + V_t^*(\mathbf{s}')\right]$$

4.If not converged, go back to Step 2.

Just add expectation operator over stochastic action, reward and next state!

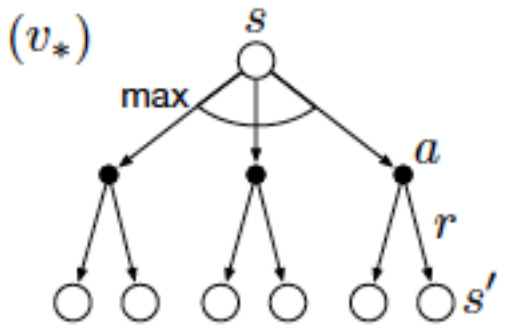
V_2	V_1	V_0	
$V_2(1)$	$V_1(1)$	$V_0(1)$	1
$V_2(2)$	$V_1(2)$	$V_0(2)$	2
$V_2(3)$	$V_1(3)$	$V_0(3)$	3
$V_2(4)$	$V_1(4)$	$V_0(4)$	4
$V_2(5)$	$V_1(5)$	$V_0(5)$	5

Value iteration

This is the value iteration algorithm

At convergence, we have found v* (as v no longer changes until “infinity”, this is solution infinite-horizon problem)

In the tutorial sessions, we’ll go through key steps in the convergence proof



backup diagram

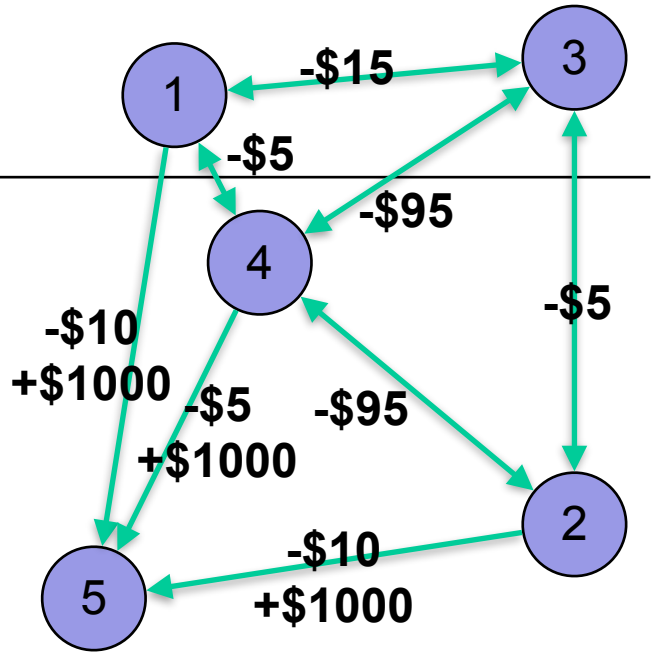
Figure: Sutton&Barto; RL:AI

Policy evaluation

Value iterations finds best policy π^* and its value function V^*

What if we are given some (possibly sub-optimal) policy π and want to determine its value function V^π?

Policy evaluation



Let's try the policy that goes to a **random** other state (except in state 5)

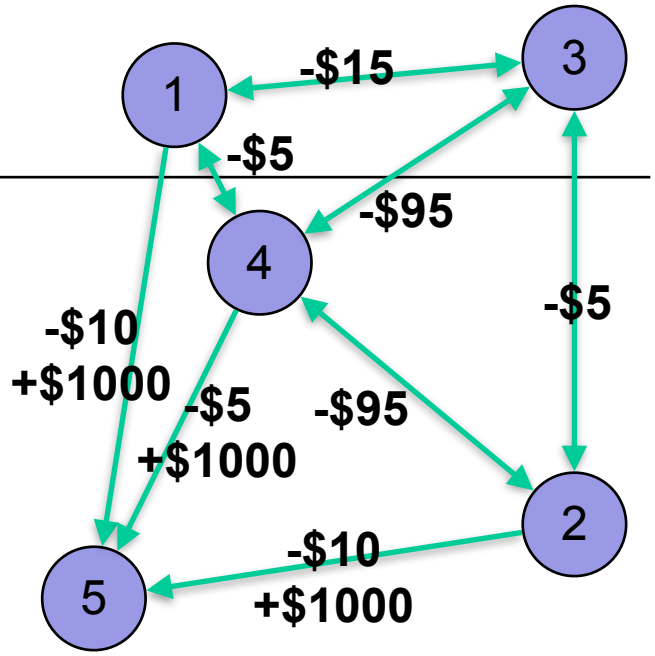
Start considering fixed-length episodes

Again: work back from last step

Bellman equation: $v_\pi(s) = \mathbb{E}_{a\sim\pi}\mathbb{E}_{s',r}[r + \gamma v_\pi(s')|s, a]$

$v_{\pi,T-1}(4) =$

Policy evaluation



Let's try the policy that goes to a **random** other state (except in state 5)

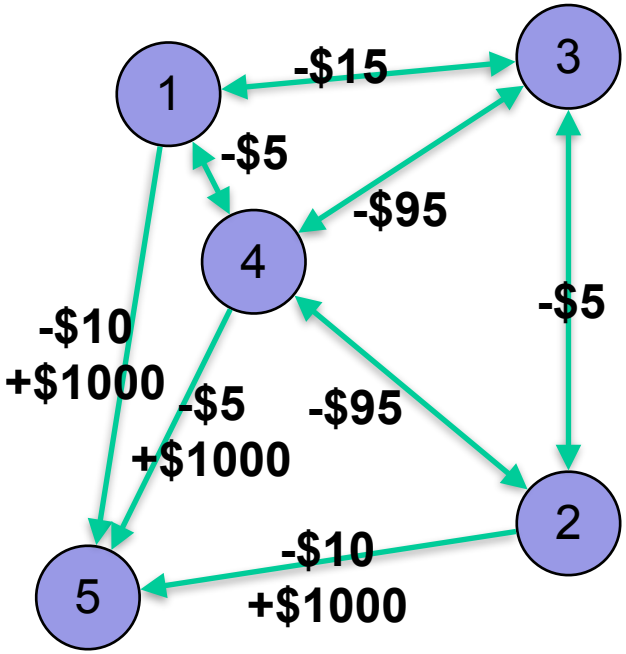
Start considering fixed-length episodes

Again: work back from last step

Bellman equation: $v_\pi(s) = \mathbb{E}_{a\sim\pi}\mathbb{E}_{s',r}[r + \gamma v_\pi(s')|s, a]$

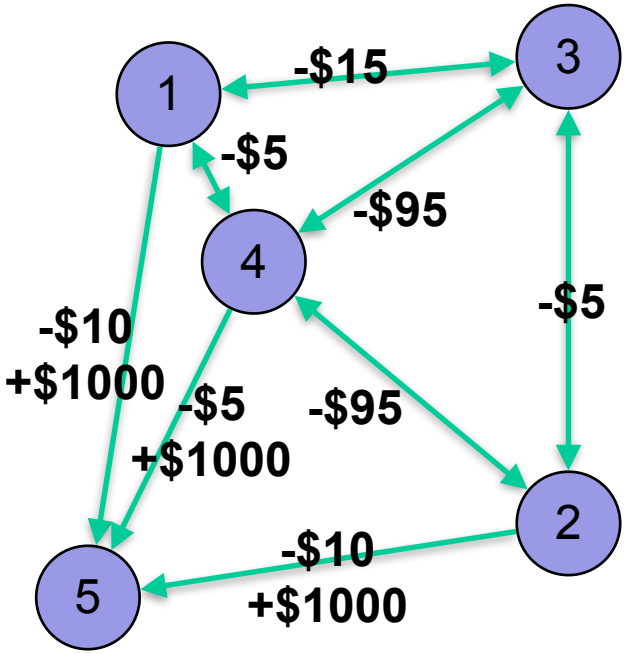
$$v_{\pi,T-1}(4) = \frac{-5+0}{4} + \frac{-95+0}{4} + \frac{-95+0}{4} + \frac{-5+1000}{4} = 200$$

Policy evaluation



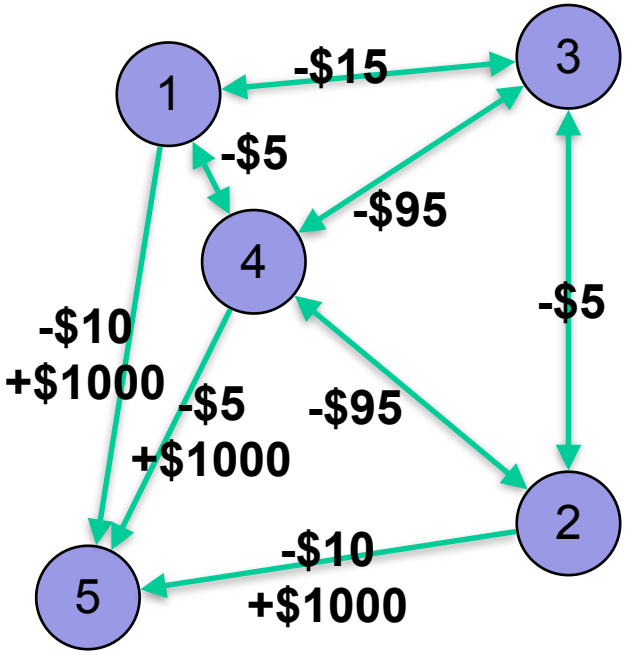
$V_{\pi,3}$	$V_{\pi,2}$	$V_{\pi,1}$	
		0	1
		0	2
		0	3
	200	0	4
		0	5

Policy evaluation



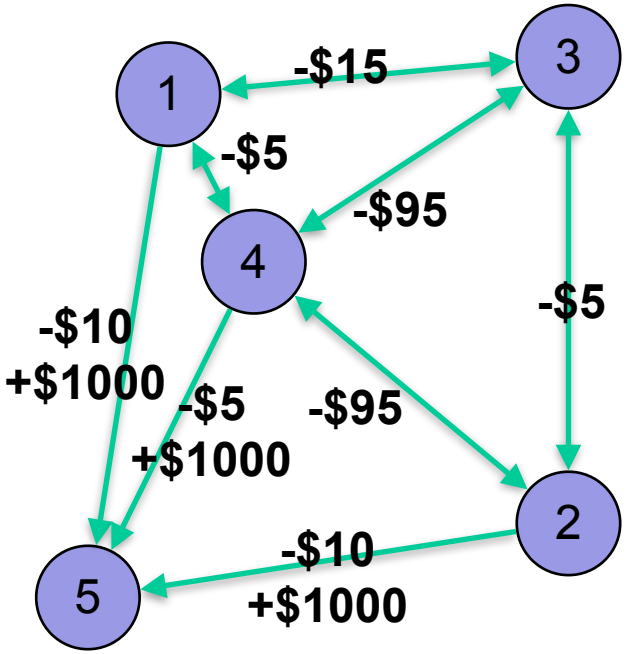
$V_{\pi,3}$	$V_{\pi,2}$	$V_{\pi,1}$	
	323,33	0	1
	296,67	0	2
	-38.33	0	3
	200	0	4
	0	0	5

Policy evaluation



$V_{\pi,3}$	$V_{\pi,2}$	$V_{\pi,1}$	
	323,33	0	1
	296,67	0	2
	-38.33	0	3
345,41	200	0	4
	0	0	5

Policy evaluation



$V_{\pi,n+1}$	$V_{\pi,n}$	.	$V_{\pi,2}$	$V_{\pi,1}$
870,33	870,33		323,33	0
843,67	843,67		296,67	0
810	810		-38.33	0
831	831		200	0
0	0		0	0

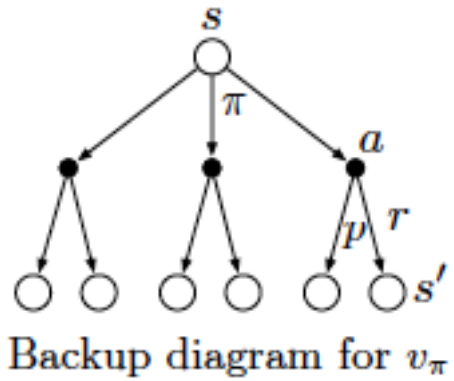
Policy evaluation

Policy evaluation converges to v_π

Again, at convergence equal to solution of infinite-horizon problem

Can we also use v_π to select better actions?

That is, can we define a **policy improvement** step**?**



Backup diagram for v_π

Figure: Sutton&Barto; RL:AI

Policy improvement

Which policy changes improve overall performance?

Policy improvement

Which policy changes improve overall performance?

Policy improvement theorem:
If π' equal to π *except* at s_t, then:

$$\mathbb{E}_{a_t \sim \pi'(s_t)} q_\pi(s_t, a_t) > v_\pi(s_t) \quad \Rightarrow \quad v_{\pi'}(s) \geq v_\pi(s) \forall s$$

How can we use this to suggest an improved policy?

Policy improvement

Which policy changes improve overall performance?

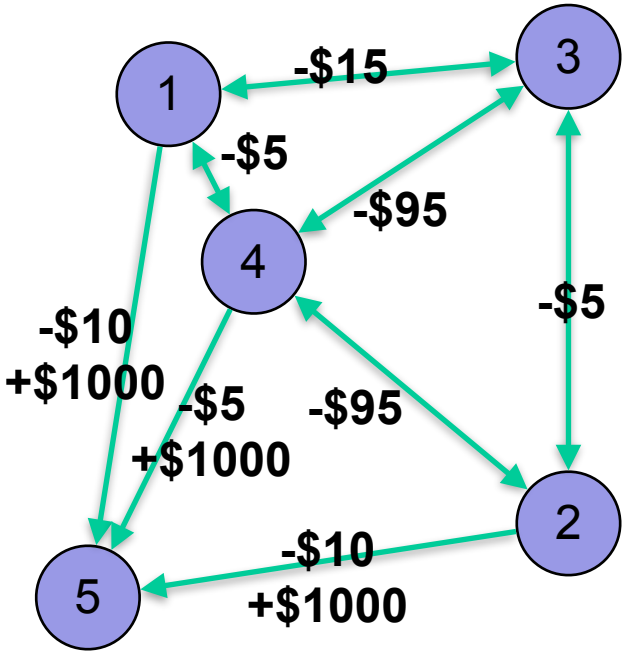
Policy improvement theorem:
If π' equal to π *except* at s_t, then:

$$\mathbb{E}_{a_t \sim \pi'(s_t)} q_\pi(s_t, a_t) > v_\pi(s_t) \quad \Rightarrow \quad v_{\pi'}(s) \geq v_\pi(s) \forall s$$

How can we use this to suggest an improved policy?

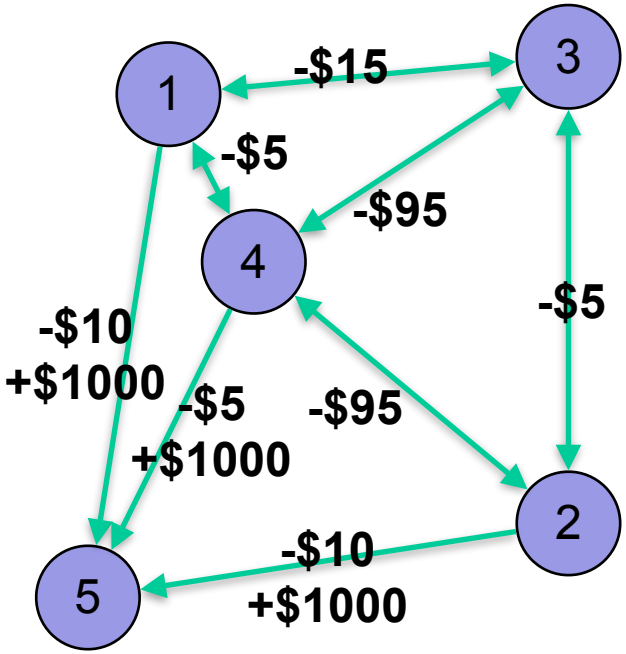
- We can act greedily at s_t (select action maximising $q_\pi(s_t, \cdot)$)
- We can repeat this argument for each possible state

Policy improvement



$V_{\pi,n+1}$	$V_{\pi,n}$	.	$V_{\pi,2}$	$V_{\pi,1}$
870,33	870,33		323,33	0
843,67	843,67		296,67	0
810	810		-38.33	0
831	831		200	0
0	0		0	0

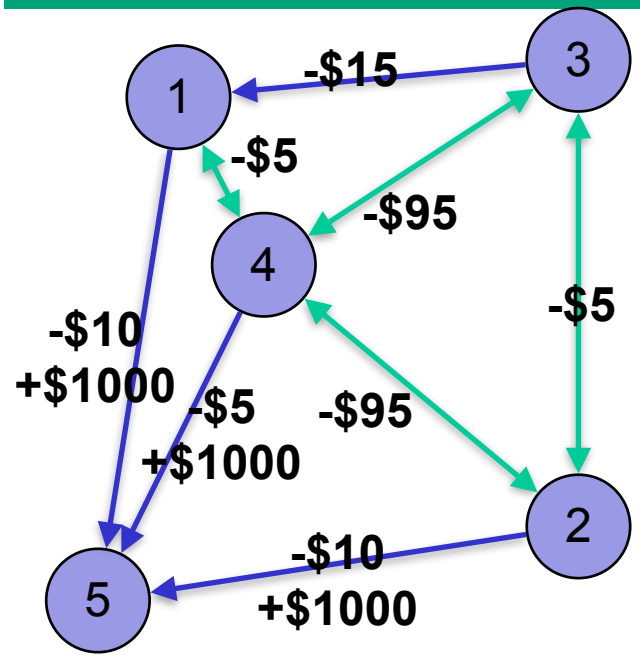
Policy improvement



$V_{\pi,n+1}$	$V_{\pi,n}$	.	$V_{\pi,2}$	$V_{\pi,1}$
870,33	870,33		323,33	0
843,67	843,67		296,67	0
810	810		-38.33	0
831	831		200	0
0	0		0	0

Policy improvement

The new policy is better than the old one
Can the policy be improved further? Why or why not?



$V_{\pi,n+1}$	$V_{\pi,n}$	.	$V_{\pi,2}$	$V_{\pi,1}$
870,33	870,33		323,33	0
843,67	843,67		296,67	0
810	810		-38.33	0
831	831		200	0
0	0		0	0

Policy iteration

Iterating the two steps is called policy iteration:

- Policy evaluation
- Policy improvement

Again, this will converge to the optimal policy & value function

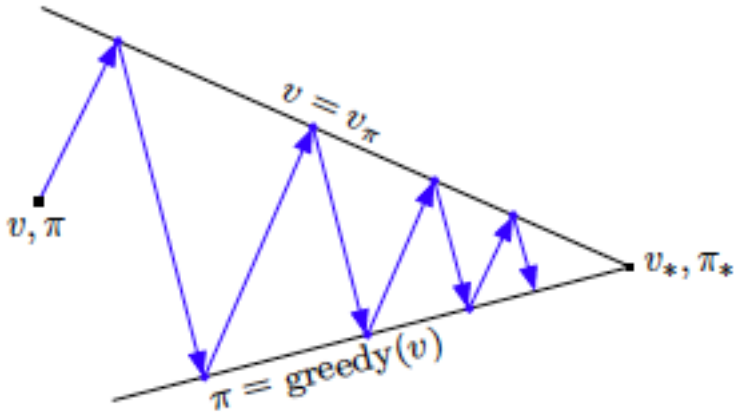


Figure: Sutton&Barto; RL:AI

Policy iteration and value iteration

Value iteration*	Policy iteration
Update v(s) once for each s	**Update v(s) until convergence**
Update policy	**Update policy** ***(policy improvement)***

Both examples of generalised policy iteration. Many others:

- Asynchronous updates (update single states)
- Do several value updates (but not until convergence)
- Start from any value function (e.g. value function from last iteration)
- Very similar procedures to compute state-action value function (q)

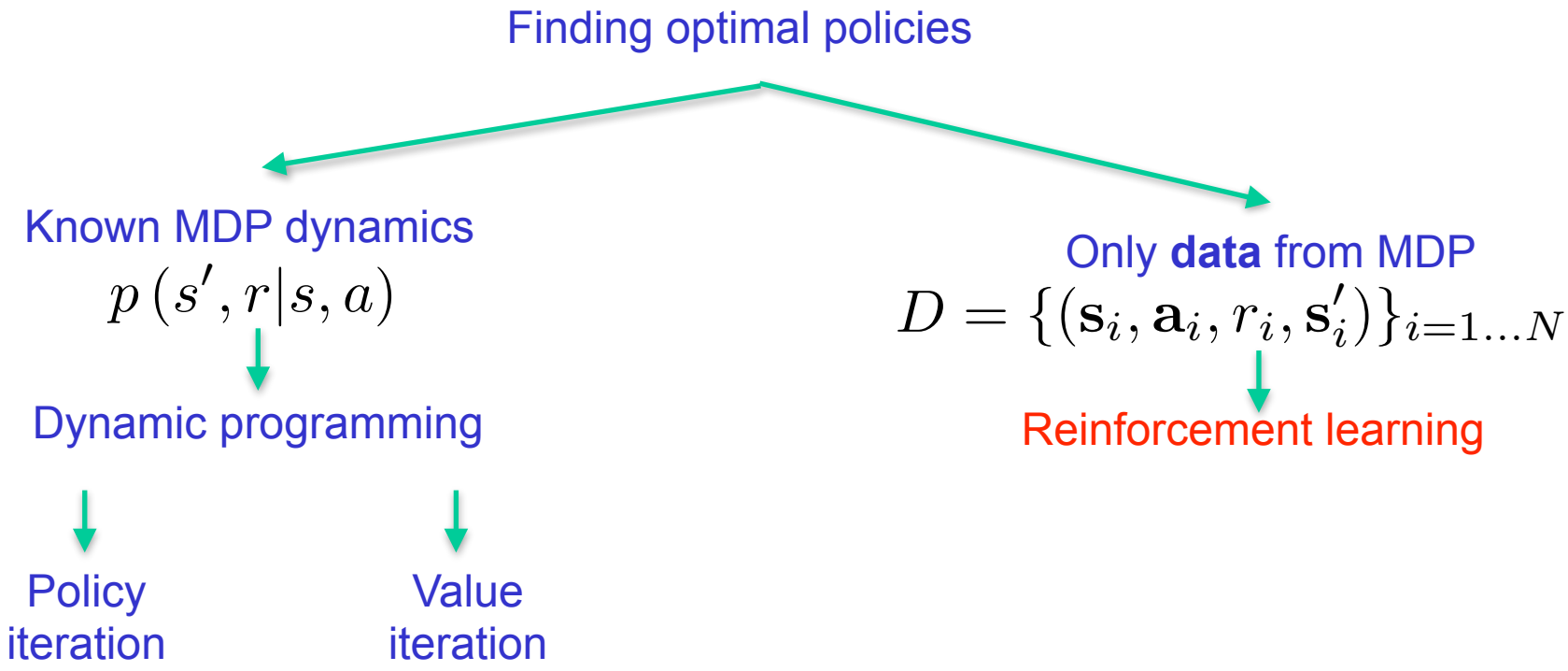
* In algorithms the policy update often happens implicitly, but it can be written out with a separate value- and policy update step as e.g. in this previous slide

Policy iteration and value iteration

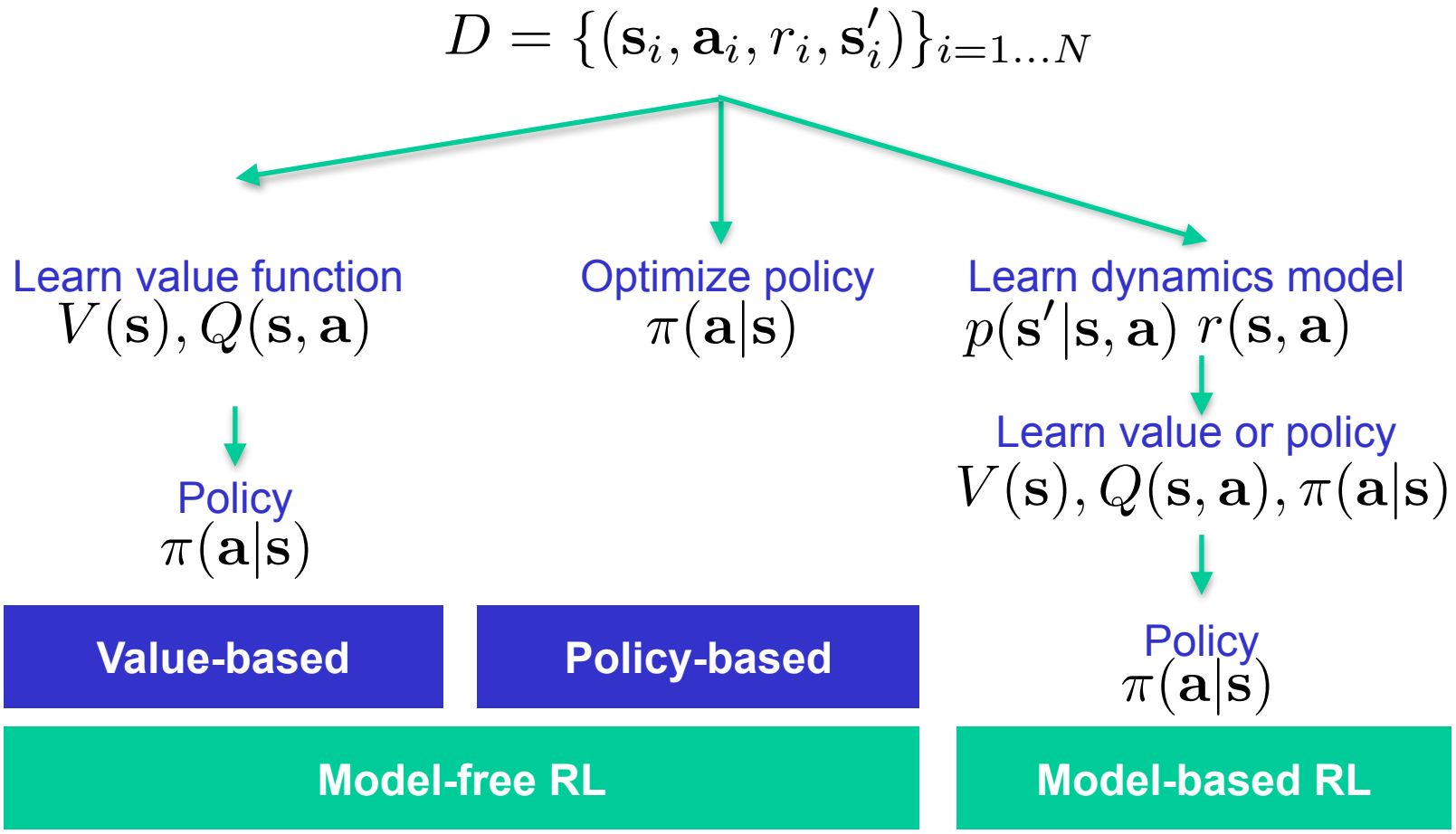
Dynamic programming requires knowing the transition probabilities!

- In RL, we typically assume we do not have them…
- We thus need to **learn** something about the environment
- We could learn the transition probabilities
- Can be more effective to learn the value function directly
- And we can even learn a policy directly, without value function

Big picture

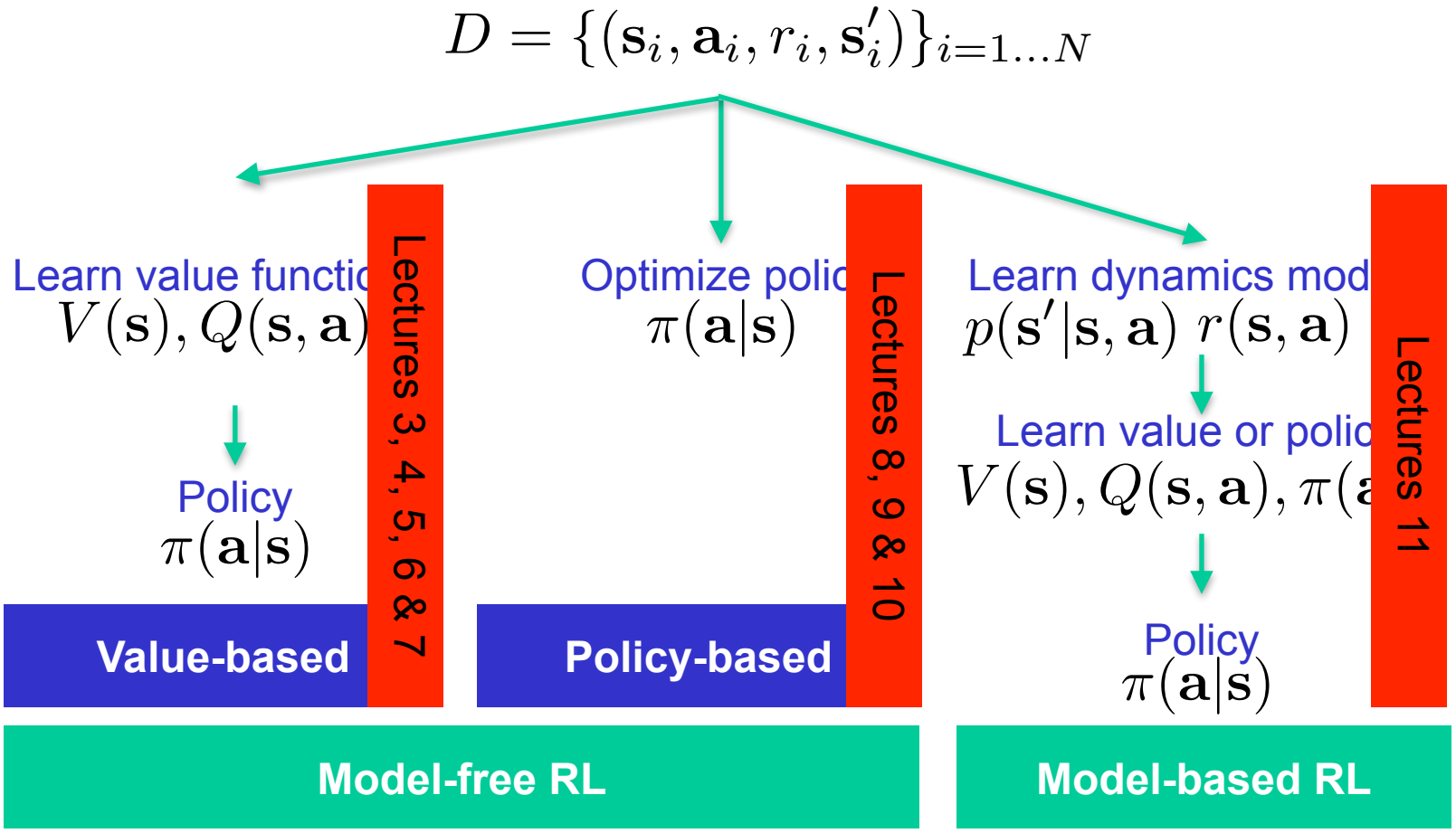

Finding optimal policies
Known MDP dynamics
$p(s', r|s, a)$
Dynamic programming
Policy iteration
Value iteration
Only **data** from MDP
$D = \{(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i)\}_{i=1...N}$
Reinforcement learning

Big picture: How to learn policies



Thanks to Jan Peters

Big picture: How to learn policies



Thanks to Jan Peters

What you should know

What is an optimal value function and an optimal policy?

What are the value iteration (VI), policy evaluation (PE) and policy iteration (PI) algorithms and what do they compute?

How do iterations of VI and PI relate to different time horizons?

What is a terminal state?

Thanks for your attention!

Feedback?

h.c.vanhoof@uva.nl