Making (even more) Complex Decisions

Paolo Turrini

Department of Computing, Imperial College London

Introduction to Artificial Intelligence
2nd Part

AlphaGo beats World Go Champion

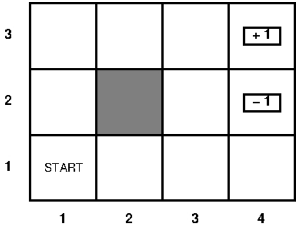
Outline

- Rewind
- The Value Iteration Algorithm

The main reference

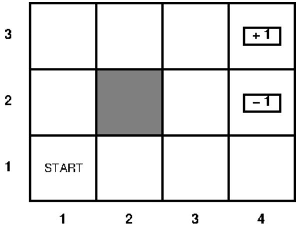
Stuart Russell and Peter Norvig
Artificial Intelligence: a modern approach
Chapters 17

The World



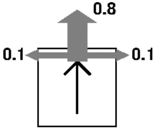
- Begin at the start state
- The game ends when we reach either goal state $+1$ or -1
- Collision results in no movement
- Rewards: $+1$ and -1 for terminal states respectively, -0.04 for all others

The World



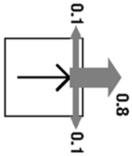
- Fully observable
- Markovian
- Discounted rewards
- Stochastic actions

The Agent



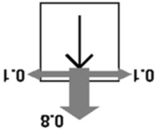
Each time it's like throwing an unfair dice

The Agent



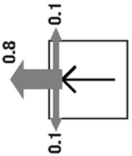
Each time it's like throwing an unfair dice

The Agent



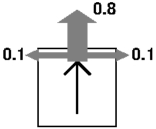
Each time it's like throwing an unfair dice

The Agent



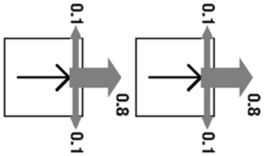
Each time it's like throwing an unfair dice

The Agent



Each time it's like throwing an unfair dice

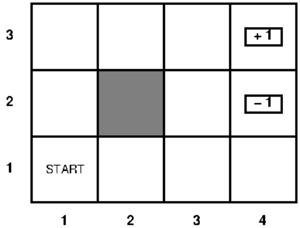
The Agent



Walking is a repetition of throws:

- The probability that I walk right the first time: 0.8
- The probability that I walk right the second time: 0.8
- It's a product! 0.8^2

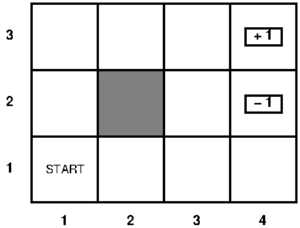
Plans and their value



Walking is a repetition of throws:

- A plan, e.g., $[\mathit{Up}, \mathit{Up}, \mathit{Right}, \mathit{Right}, \mathit{Right}]$, can bring us somewhere unintentionally

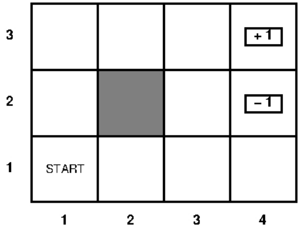
Plans and their value



Walking is a repetition of throws:

- A plan, e.g., [*Up*, *Up*, *Right*, *Right*, *Right*], can bring us somewhere unintentionally
- How much is a plan worth?

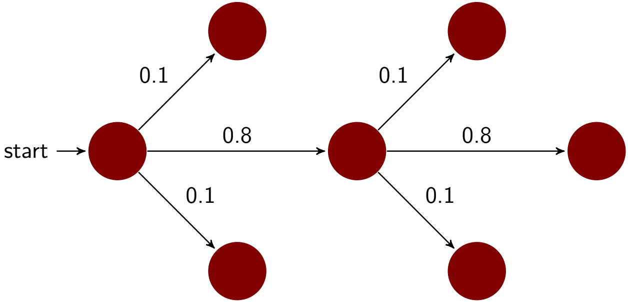
Plans and their value



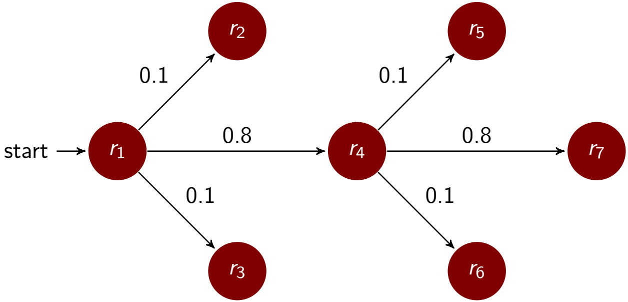
Walking is a repetition of throws:

- A plan, e.g., $[Up, Up, Right, Right, Right]$, can bring us somewhere unintentionally
- How much is a plan worth? $v^p(s) = E[\sum_{t=0}^{\infty} \gamma^t r(S_t)]$, the expected (discounted) sum of rewards.

Time and Risk

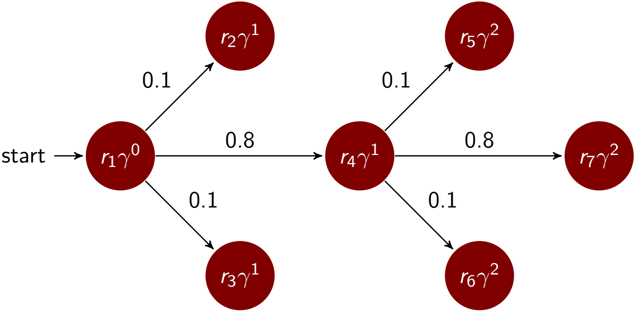


Time and Risk



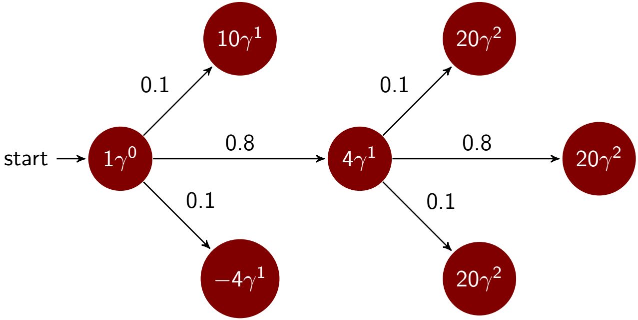
The real value of rewards depends on the agent's patience.
(as much as the real value of money depends on the attitude towards risk)

Time and Risk



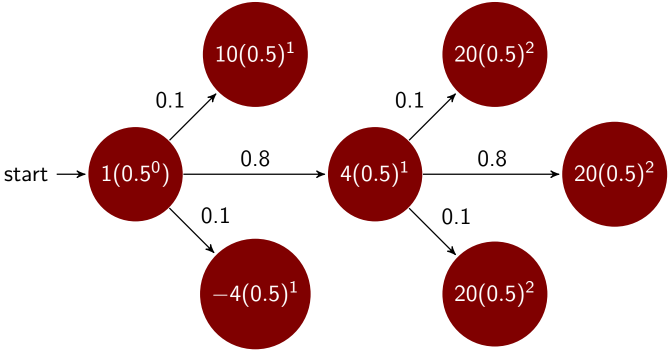
Multiplicative discounting γ^n after n steps.

Time and Risk



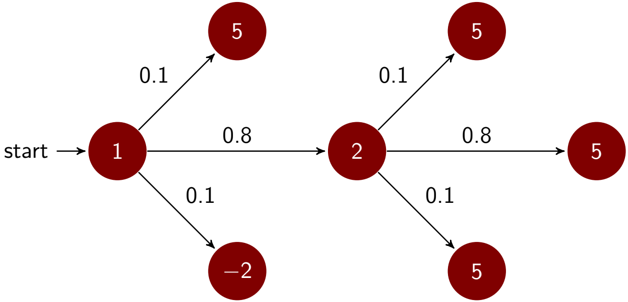
Multiplicative discounting: γ^n after n steps.

Time and Risk



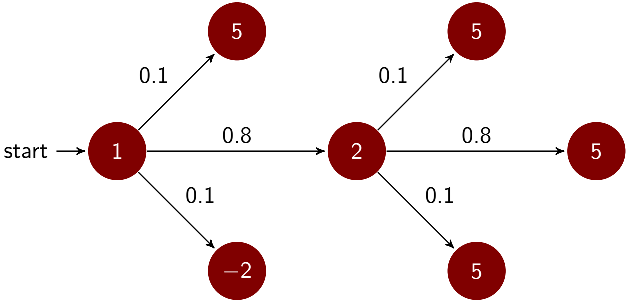
$\gamma = 0.5$

Time and Risk



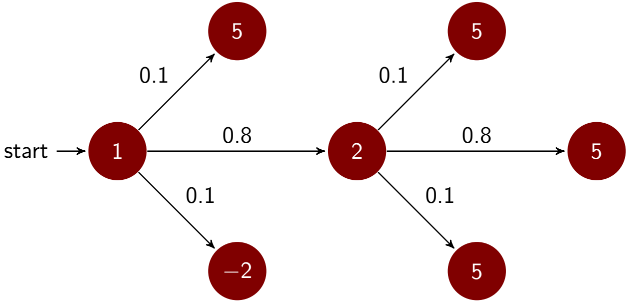
$\gamma = 0.5$

Time and Risk



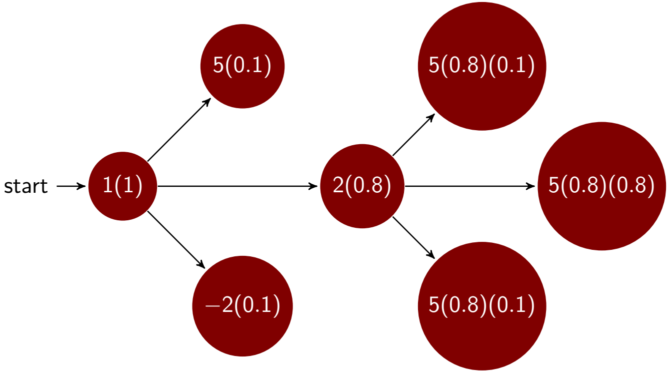
And now?

Time and Risk



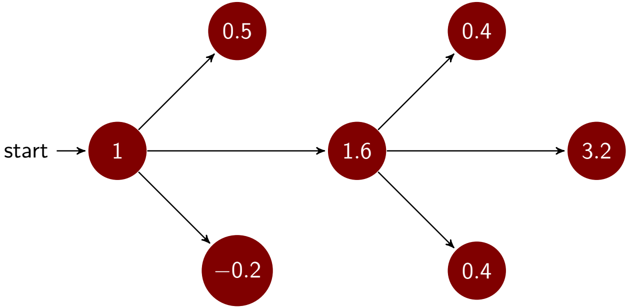
And now?
We include the probabilities...

Time and Risk



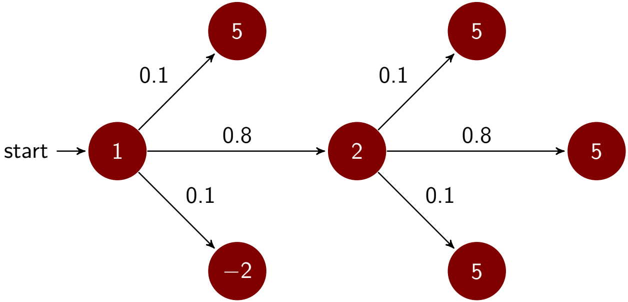
Probabilities of sequences:
to *discount* further the already discounted rewards!

Time and Risk



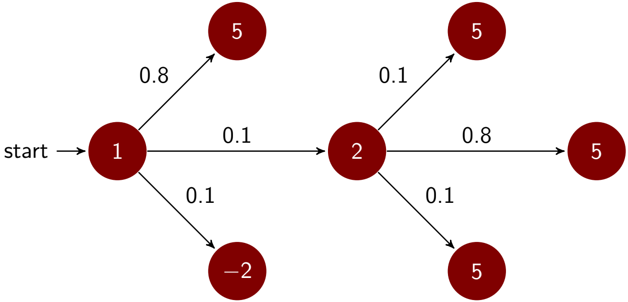
Expected utility of this intended course of actions (not considering the rest = assuming it's zero reward everywhere else) is:
6.9

Time and Risk



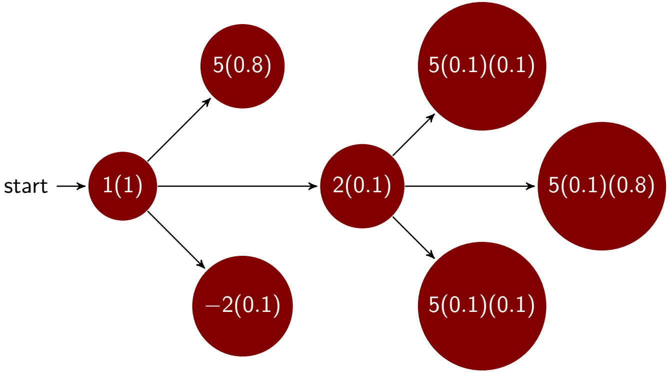
Let's see what happens if we go up instead...

Time and Risk



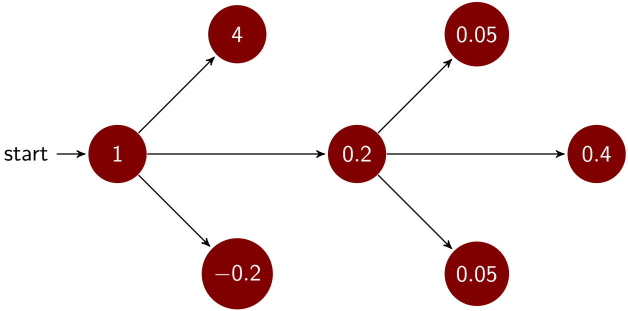
Let's see what happens if we go up instead...

Time and Risk



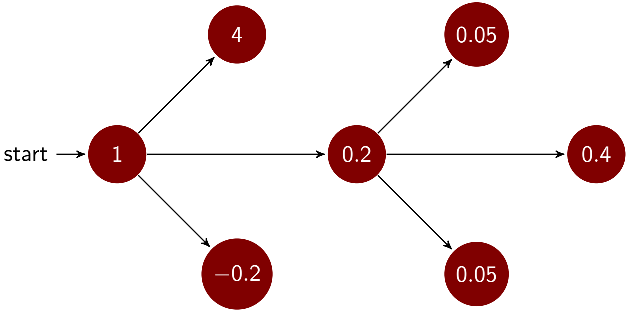
Including probabilities...

Time and Risk



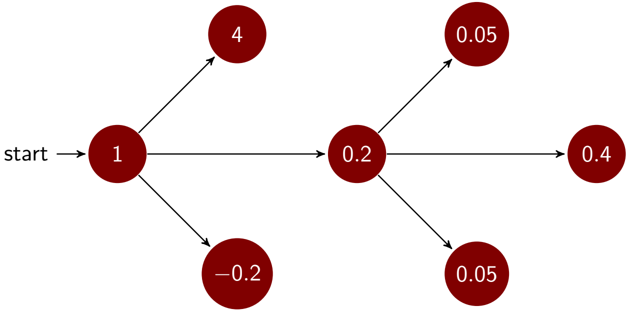
Including probabilities...

Time and Risk



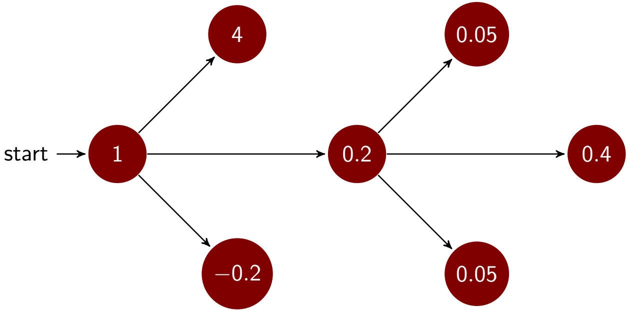
Summing up: 5.5

Time and Risk



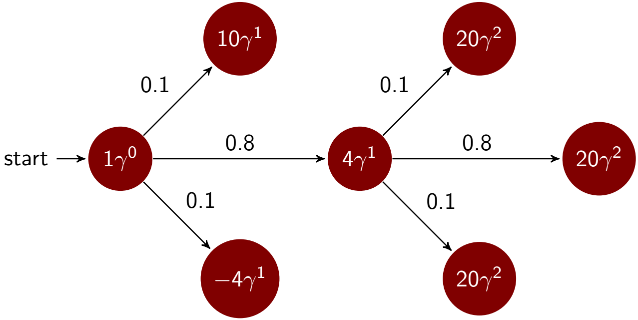
This means that switching to *Up* is dominated by going right.

Time and Risk



This means that switching to *Up* is dominated by going right.
Same reasoning for going down: lower expected utility!

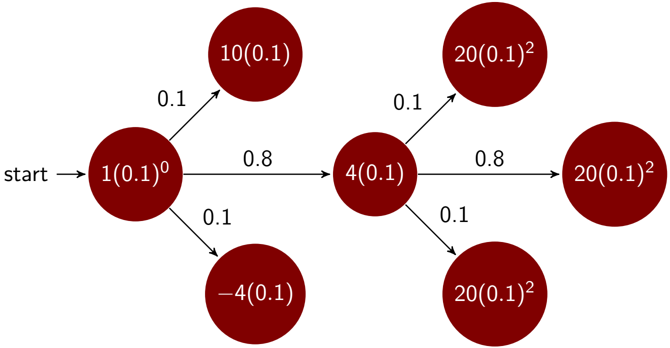
Time and Risk



Now I'm going to be very impatient.

$\gamma = 0.1$

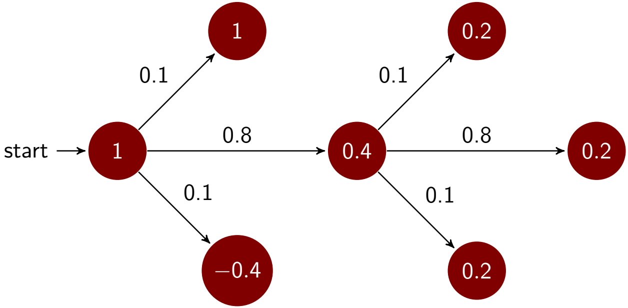
Time and Risk



Now I'm going to be very impatient.

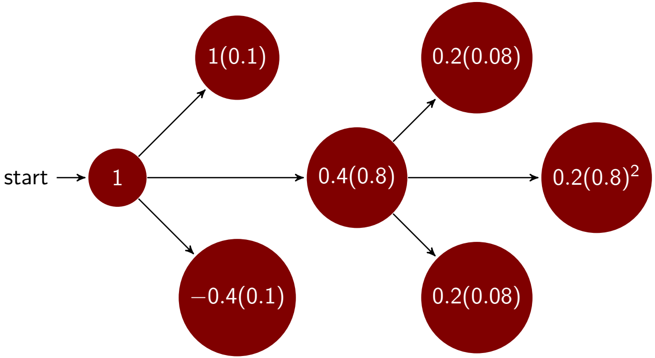
$\gamma = 0.1$

Time and Risk



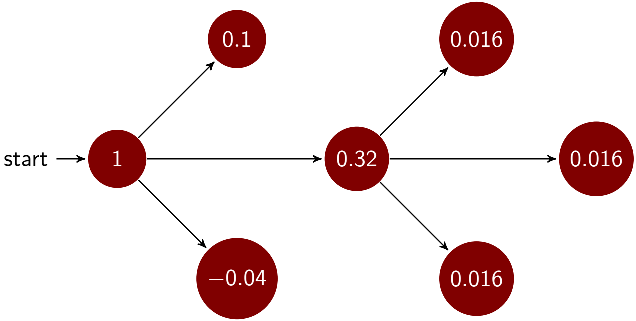
Can you already see what's going on?

Time and Risk



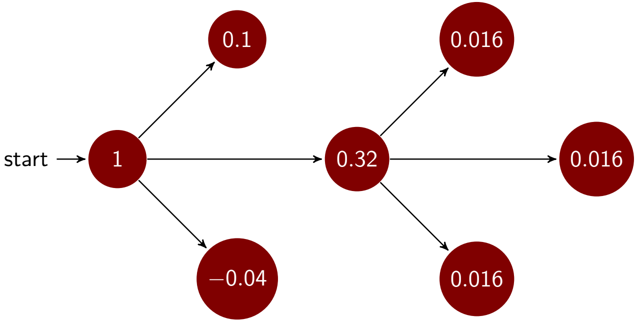
Let's include the probabilities

Time and Risk



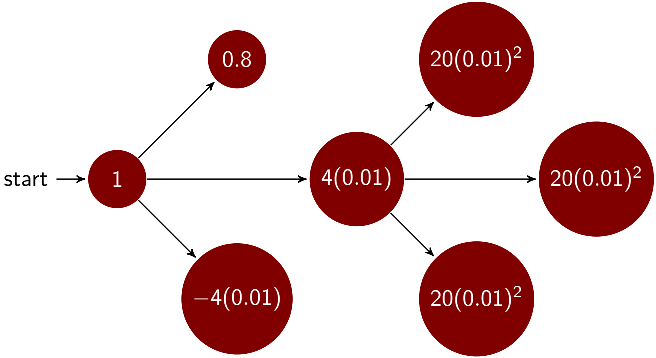
Notice the impact of discounting on negative rewards:
In the end, it's all gonna be zero!

Time and Risk



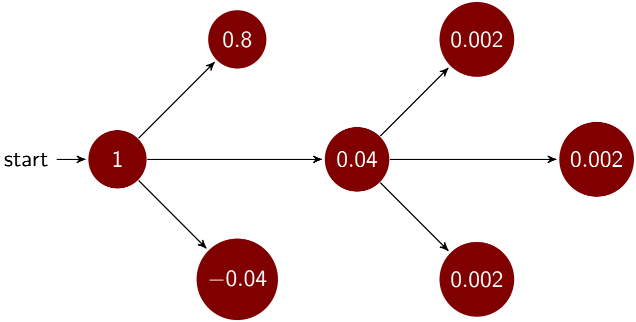
The expected utility at the starting state is: 1.428

Time and Risk



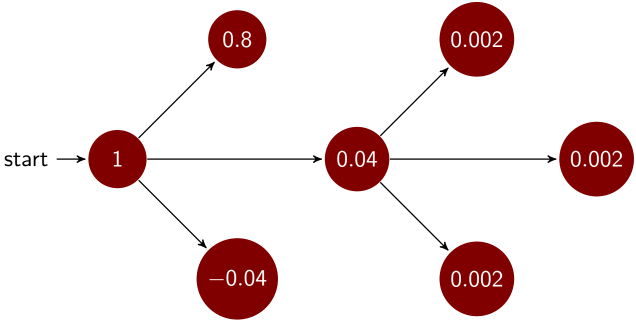
Let's go up

Time and Risk



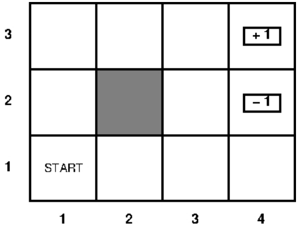
The expected utility at the starting state is: 1.806

Time and Risk

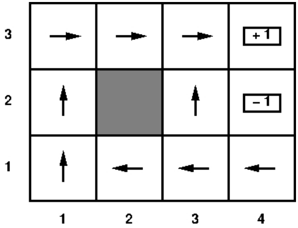


Now Up is dominant!

A policy



A policy



A policy is a specification of moves at each decision point

Expected utility of a policy

The expected utility (or value) of policy π, from state s is:

Expected utility of a policy

The expected utility (or value) of policy π, from state s is:

$$v^{\pi}(s) = E[\sum_{t=0}^{\infty} \gamma^{t} r(S_t)]$$

Expected utility of a policy

The expected utility (or value) of policy π, from state s is:

$$v^{\pi}(s) = E[\sum_{t=0}^{\infty} \gamma^t r(S_t)]$$

E, the probability distribution over the sequences is induced by:

- the policy π (the actions we are actually going to make)
- the initial state t (where we start)
- the transition model (where we can get to)

Expected utility of a policy

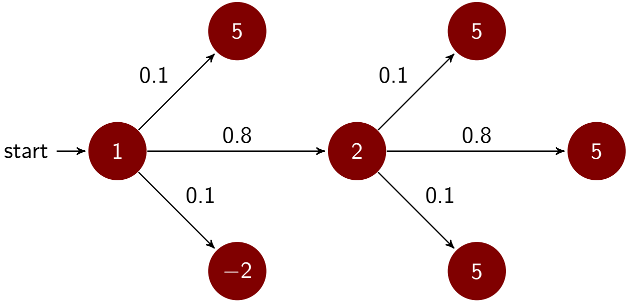
The expected utility (or value) of policy π, from state s is:

$$v^{\pi}(s) = E[\sum_{t=0}^{\infty} \gamma^{t} r(S_t)]$$

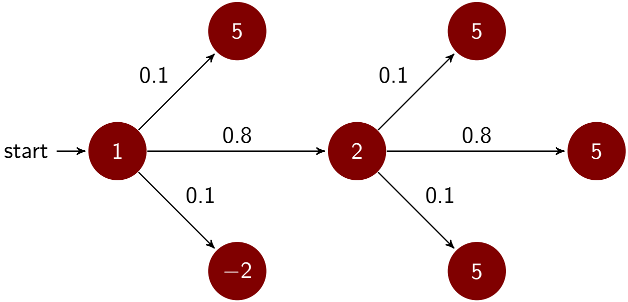
E, the probability distribution over the sequences is induced by:

- the policy π (the actions we are actually going to make)
- the initial state t (where we start)
- the transition model (where we can get to)

Optimal policies

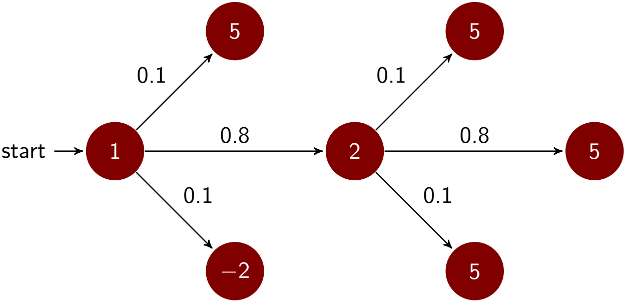


Optimal policies



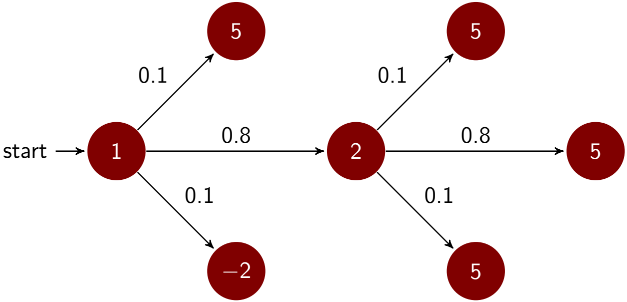
If this was the entire (relevant) world...

Optimal policies



If this was the entire (relevant) world...
and $\gamma = 0.5$

Optimal policies



If this was the entire (relevant) world...
and $\gamma = 0.5$
Going straight twice in a row would have value: 6.9

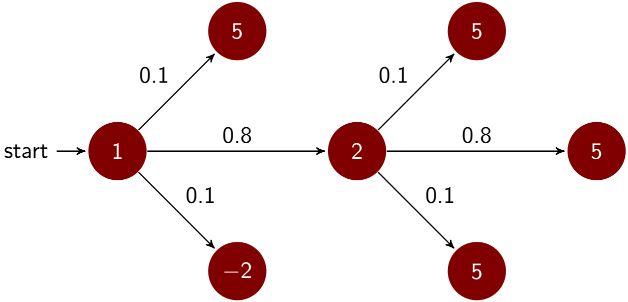
Optimal policies

We want the **optimal** policy:

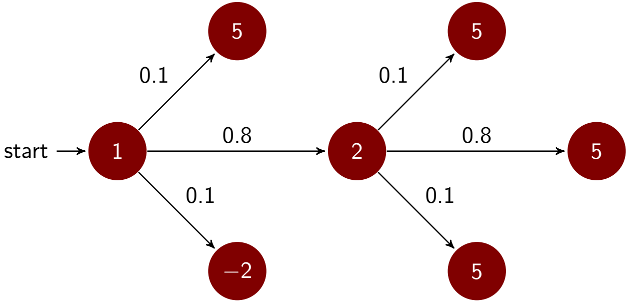
$$\pi_s^* = \underset{\pi}{\operatorname{argmax}}\, v^\pi(s)$$

And we know that it's unique no matter the starting state.

Optimal policies

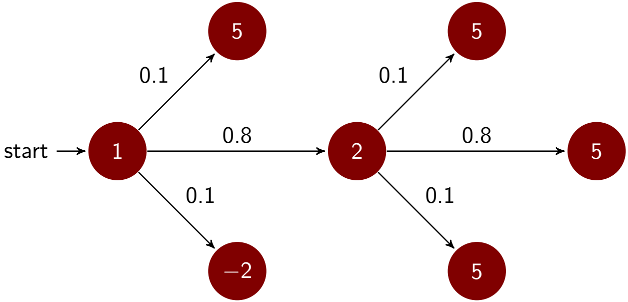


Optimal policies



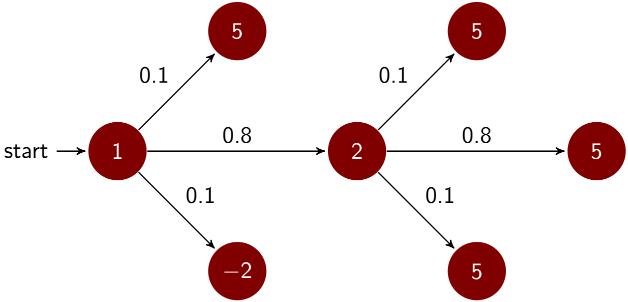
If this was the entire (relevant) world...

Optimal policies



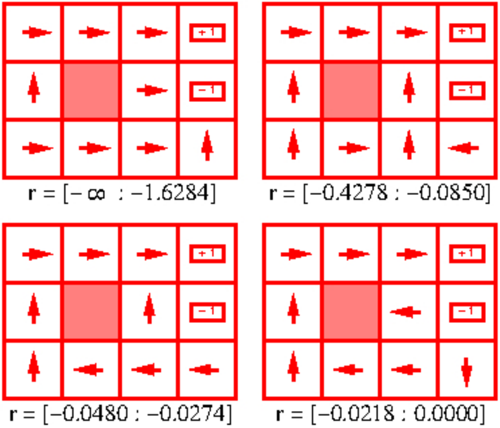
If this was the entire (relevant) world...
and $\gamma = 0.5$

Optimal policies



If this was the entire (relevant) world...
and $\gamma = 0.5$
Going straight twice in a row would be optimal

Risk and reward



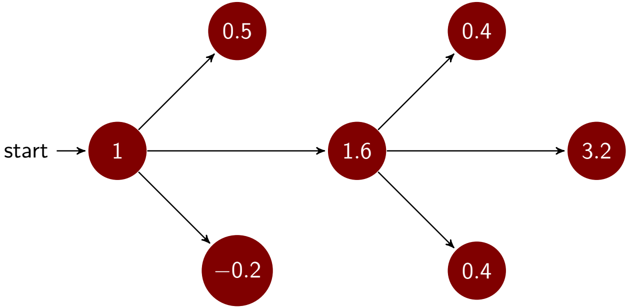
Value of states

The **value of a state** s is its value under the optimal policy.

In other words:

expected (discounted) sum of rewards
assuming optimal actions

Value of states



6.9 is the value of the starting state.

VERY VERY IMPORTANT

Given the values of the states, choosing the best action is just MEU: maximize the expected utility of the immediate successors

Value of states

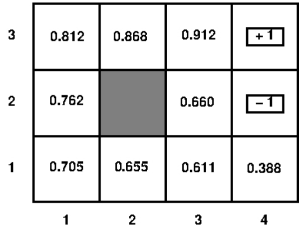


Figure: The values with $\gamma = 1$ and $R(s) = -0.04$

Value of states

3	0.812	0.868	0.912	+1
2	0.762		0.660	−1
1	0.705	0.655	0.611	0.388
	1	2	3	4

Figure: The optimal policy

$$\pi^*(s) = \underset{a \in A(s)}{\text{argmax}} \sum_{s'} P(s' \mid s, a) v(s')$$

Maximise the expected utility of the subsequent state

Value of states

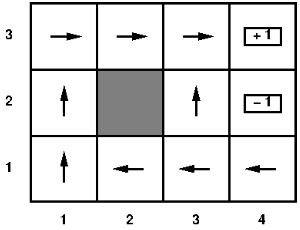


Figure: The optimal policy

$$\pi^*(s) = \underset{a \in A(s)}{\text{argmax}} \sum_{s'} P(s' \mid s, a) v(s')$$

Maximise the expected utility of the subsequent state

The Bellman equation

Definition of utility of states leads to a simple relationship among values of neighboring states:

The Bellman equation

Definition of utility of states leads to a simple relationship among values of neighboring states:

Definition (Rewards)

expected sum of rewards $=$ current reward $+\ \gamma \times$ expected sum of rewards after taking best action

The Bellman equation

Bellman equation (1957):

$$v(s) = r(s) + \gamma \max_a \sum_{s'} P(s' \mid (s, a)) v(s')$$

The Bellman equation

Bellman equation (1957):

$$v(s) = r(s) + \gamma \max_a \sum_{s'} P(s' \mid (s,a)) v(s')$$

We can use it to compute the optimal policy!

Value Iteration Algorithm

1. Start with arbitrary values
2. Repeat for every s simultaneously until "no change"

$$v(s) \leftarrow r(s) + \gamma \max_a \sum_{s'} v(s')P(s' \mid (s, a))$$

The Value Iteration Algorithm

Algorithm : VIA

Value Iteration(MDP, ϵ)

Input: MDP, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$, rewards $R(s)$, discount γ

ϵ, the maximum error allowed in the utility of any state

Output: A utility function

begin

  $v \leftarrow v', \delta \leftarrow 0$; `/* Using local variables to store information about values and value change */`

  while $\delta < \epsilon(\frac{(1-\gamma)}{\gamma})$ **do**

  for each state $s \in S$ **do**

$$v'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid (s,a))v[s'];$$

    if $|v'[s] - v[s]| > \delta$ **then**

      $\delta \leftarrow |v'[s] - v[s]|$

    Return v;

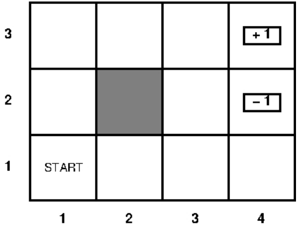
A fundamental fact

Theorem

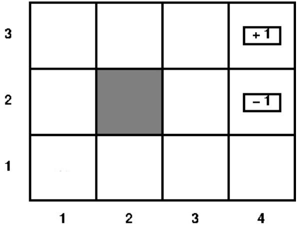
VIA:

- *terminates*
- *returns the unique optimal policy (for the input values).*

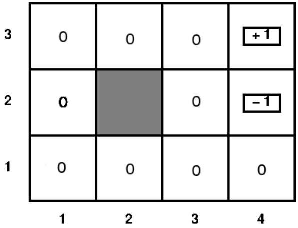
VIA in action



VIA in action

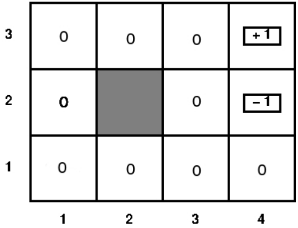


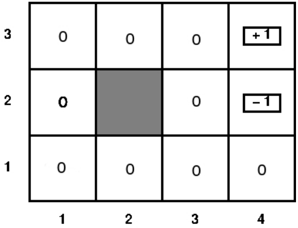
VIA in action



Initialise the values, for $\gamma = 1, r = -0.04$

VIA in action

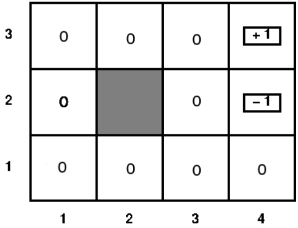


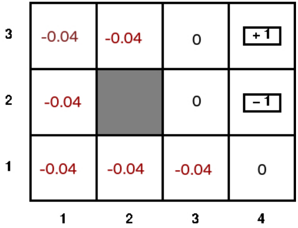


Simultaneously apply the Bellmann update to all states

$$v(s) = r(s) + \gamma \max_a \sum_{s'} P(s' \mid (s, a)) v(s')$$

VIA in action





Simultaneously apply the Bellmann update to all states

$$v(s) = r(s) + \gamma \max_a \sum_{s'} P(s' \mid (s, a))v(s')$$

VIA in action

3	0	0	0	+1
2	0		0	−1
1	0	0	0	0
	1	2	3	4

3	-0.04	-0.04	0	+1
2	-0.04		-0.04	−1
1	-0.04	-0.04	-0.04	-0.04
	1	2	3	4

$$v(s) = r(s) + \gamma \max_a \sum_{s'} P(s' \mid (s,a)) v(s')$$

VIA in action

	1	2	3	4
3	0	0	0	+1
2	0		0	−1
1	0	0	0	0

	1	2	3	4
3	-0.04	-0.04	0.76	+1
2	-0.04		-0.04	−1
1	-0.04	-0.04	-0.04	-0.04

$$v(s) = r(s) + \gamma \max_a \sum_{s'} P(s' \mid (s,a)) v(s')$$

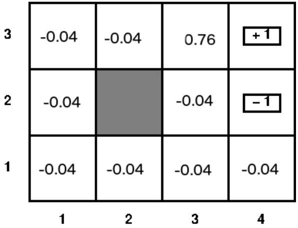
VIA in action

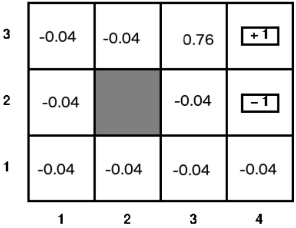
	1	2	3	4
3	0	0	0	+1
2	0		0	−1
1	0	0	0	0

	1	2	3	4
3	-0.04	-0.04	0.76	+1
2	-0.04		-0.04	−1
1	-0.04	-0.04	-0.04	-0.04

$$v(s) = r(s) + \gamma \max_a \sum_{s'} P(s' \mid (s, a)) v(s')$$

VIA in action





$$v(s) = r(s) + \gamma \max_a \sum_{s'} P(s' \mid (s, a)) v(s')$$

VIA in action

	1	2	3	4
3	-0.04	-0.04	0.76	$\boxed{+1}$
2	-0.04		-0.04	$\boxed{-1}$
1	-0.04	-0.04	-0.04	-0.04

	1	2	3	4
3	-0.08	-0.04	0.76	$\boxed{+1}$
2	-0.08		-0.04	$\boxed{-1}$
1	-0.08	-0.08	-0.08	-0.04

$$v(s) = r(s) + \gamma \max_a \sum_{s'} P(s' \mid (s,a)) v(s')$$

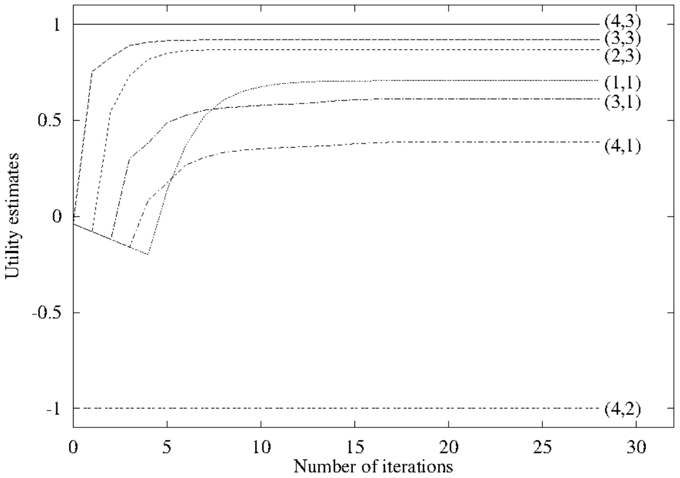
VIA in action

	1	2	3	4
3	-0.04	-0.04	0.76	+1
2	-0.04		-0.04	−1
1	-0.04	-0.04	-0.04	-0.04

	1	2	3	4
3	-0.08	0.56	0.832	+1
2	-0.08		0.46	−1
1	-0.08	-0.08	-0.08	-0.08

$$v(s) = r(s) + \gamma \max_a \sum_{s'} P(s' \mid (s,a)) v(s')$$

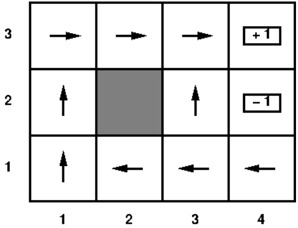
Value Iteration Algorithm



The state values

3	0.812	0.868	0.912	+1
2	0.762		0.660	−1
1	0.705	0.655	0.611	0.388
	1	2	3	4

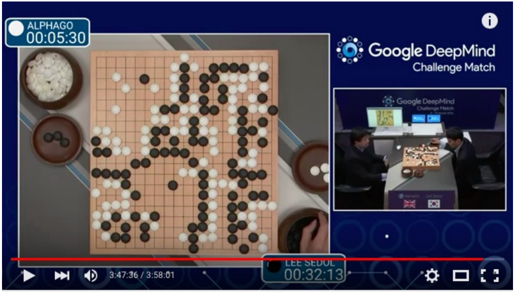
The optimal policy



Summary

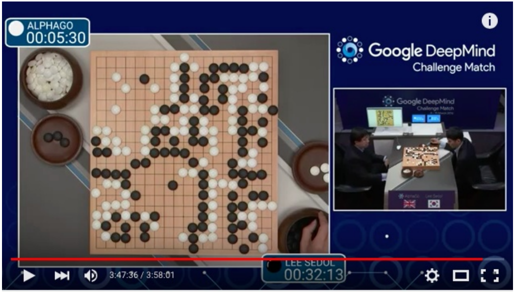
- Stochastic actions can lead to unpredictable outcomes
- But we can still find optimal “strategies”, exploiting probabilities

What's next



What if we don't know what game we are playing?

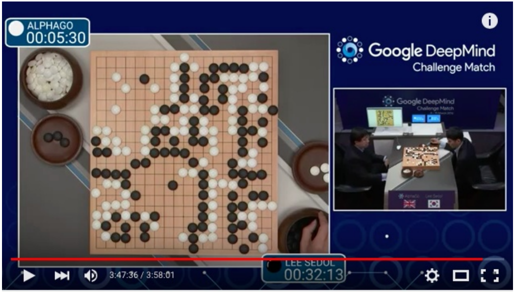
What's next



What if we don't know what game we are playing?

Play anyway and see what happens!

What's next



What if we don't know what game we are playing?

Play anyway and see what happens!
and play as much as possible!