Linköping University

TDDC17: Intro to Automated Planning

Planning under Uncertainty

Jonas Kvarnström
Artificial Intelligence and Integrated Computer Systems Division
Department of Computer and Information Science
Linköping University

jonas.kvarnstrom@liu.se – 2021

Multiple Outcomes

jonkv@ida

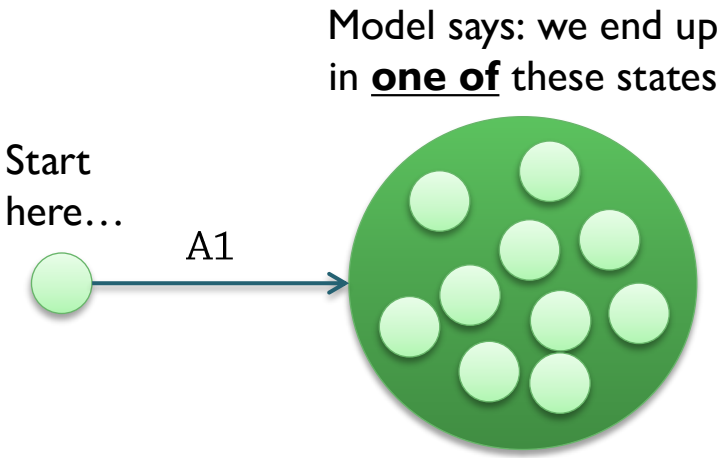
- **Classical planning** assumes we know outcomes in advance
  - State + action ➔ unique resulting state

- Sometimes we **must** deal with **multiple outcomes**
  - Due to **problems in execution**
    - **pick-up(object)**
      Intended outcome: carrying(object) is true
      Unintended outcome: carrying(object) is false
  - Due to **random** but clearly **desirable / undesirable** outcomes
    - Toss a coin – do I win?
  - Due to **random** outcomes with **unknown** long term effects
    - Do I end up in group A or B?
      No idea which one will turn out to be better for me

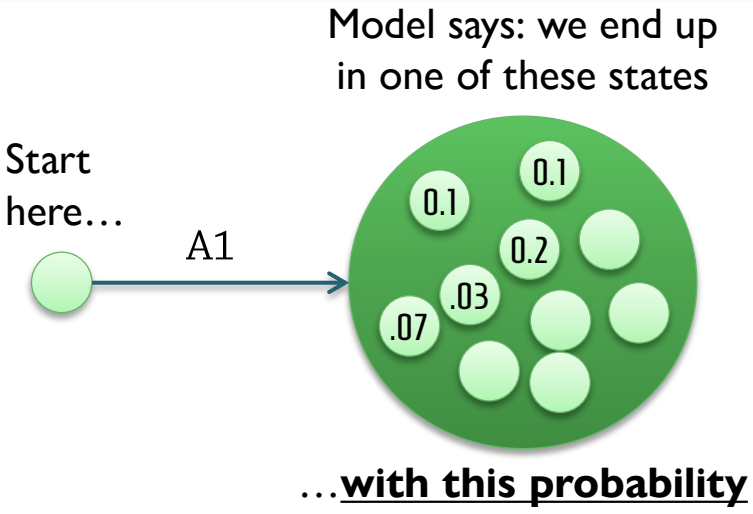
Information, while planning

jonkv@ida

Non-Deterministic Planning



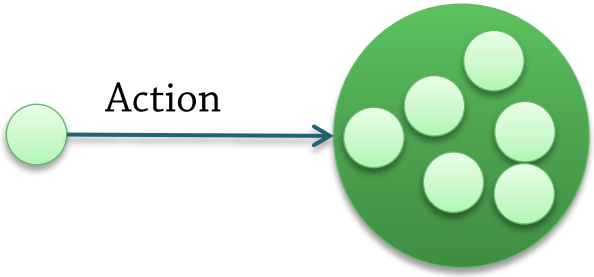
Probabilistic Planning



- First ”info dimension”:
  - What **do** we know about action outcomes **when we create the plan**?

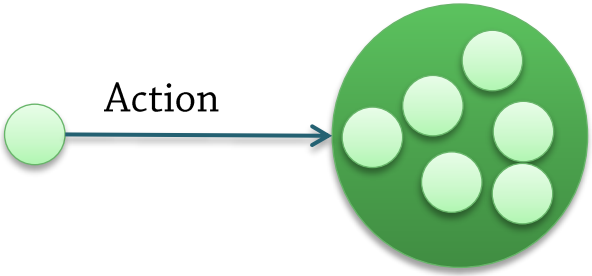
← Focus of this lecture!

Prediction during Planning



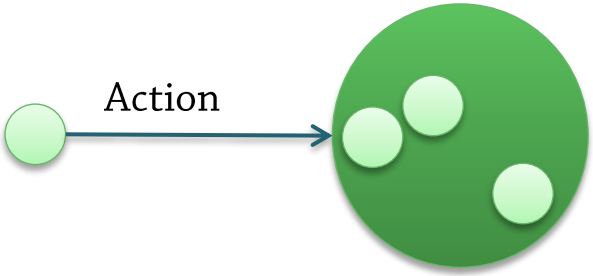
- Second "info dimension":
  - What do we **find out** about action outcomes **when we execute the plan**?

After execution: Non-Observable



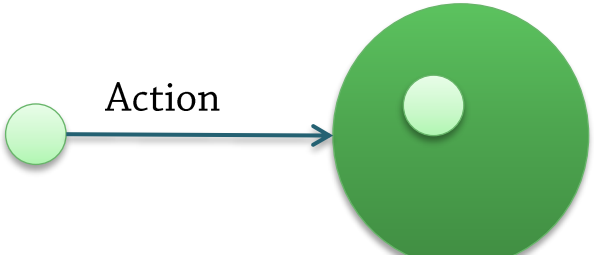
No new information sensed after executing an action
Only our initial predictions

After execution: Partially Observable



Can get some information
Some aspects are not observable
Still uncertain about the current state

After execution: Fully Observable



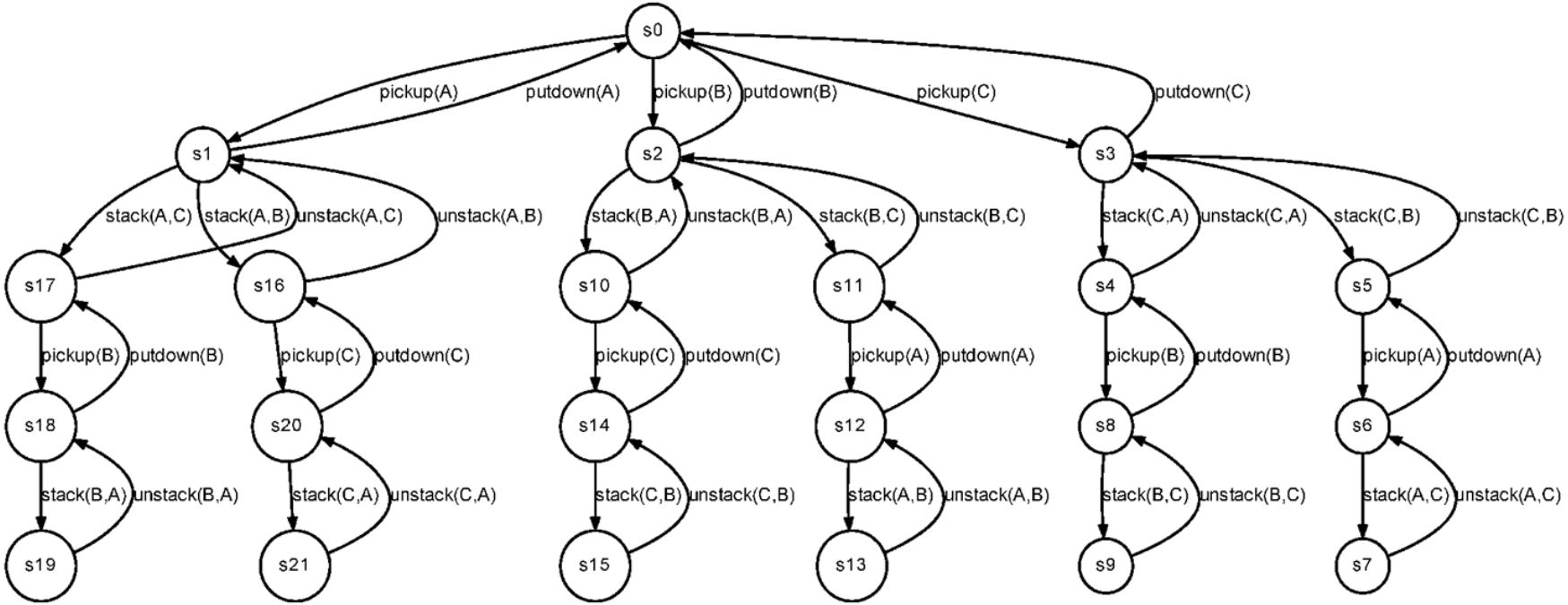
After executing an action
we will know the state we ended up in
← Focus of this lecture!

Probabilistic Planning:
Defining the World as a Stochastic System

State Transition System

jonkv@ida

- Classical planning: A **state transition system** $\Sigma = (S, A, \gamma)$
  - $S = \{ s_0, s_1, \ldots \}$: Finite set of **world states**
  - $A = \{ a_0, a_1, \ldots \}$: Finite set of **actions**
  - γ: S × A → 2^S: **State transition function**, specifying all "edges"



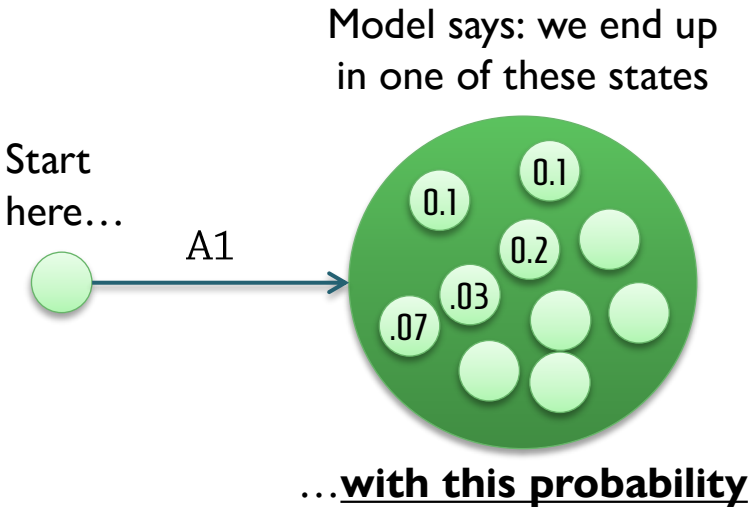
Stochastic System

jonkv@ida

- **Probabilistic planning** uses a **stochastic system** $\Sigma = (S, A, P)$
  - $S = \{ s_0, s_1, \ldots \}$: Finite set of **world states**
  - $A = \{ a_0, a_1, \ldots \}$: Finite set of **actions**
  - $P(s, a, s')$: Given that we are in s and execute a, the **probability** of ending up in s'

Replaces γ

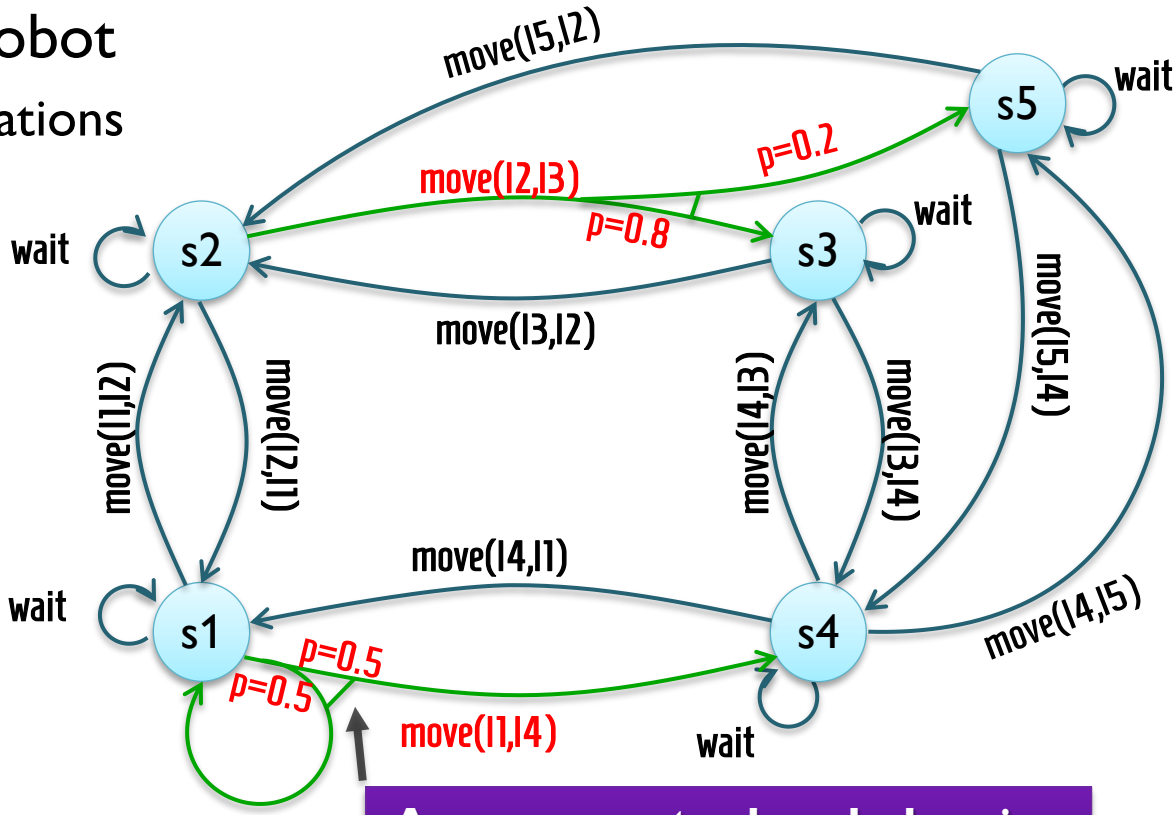
Planning



Stochastic System Example

jonkv@ida

- **Example**: A single robot
  - Moving between 5 locations
  - For simplicity, states correspond directly to locations
    - **s1**: at(rob, l1)
    - **s2**: at(rob, l2)
    - **s3**: at(rob, l3)
    - **s4**: at(rob, l4)
    - **s5**: at(rob, l5)

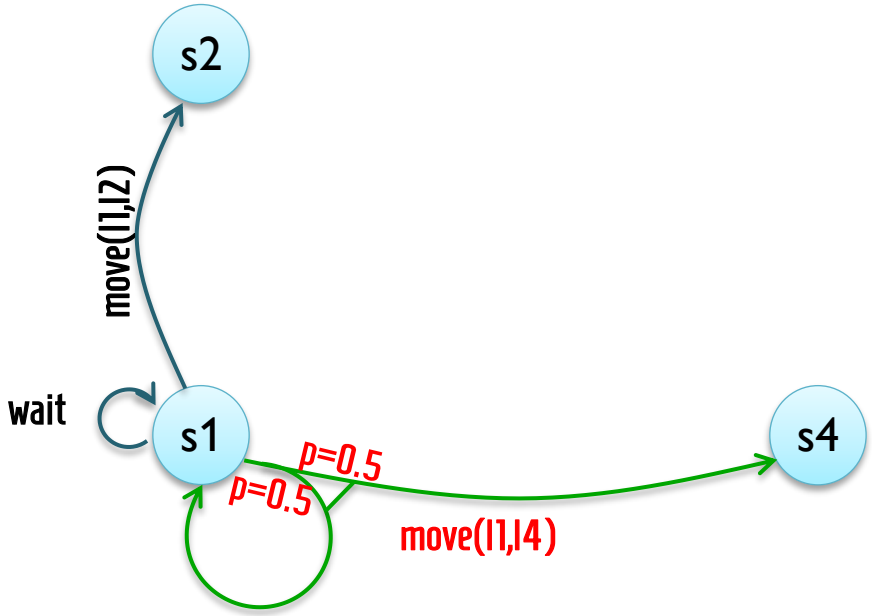


- Some transitions are **deterministic**, some are **stochastic**
  - Trying to move from l2 to l3: You may end up at l5 instead (20% risk)
  - Trying to move from l1 to l4: You may stay where you are instead (50% risk)

Stochastic System: Planning and Execution

jonkv@ida

- **Example**: In state **s1**, there are three possible actions
  - The planner chooses the **action**
    - For example, move(l1,l4)
  - Much later, when we actually execute this action:
    - We will find out the outcome (end up in s4 or s1), due to full observability
  - But when we're planning:
    - Must try to prepare for both potential outcomes!



Fully Observable Probabilistic Planning: Policies and Histories

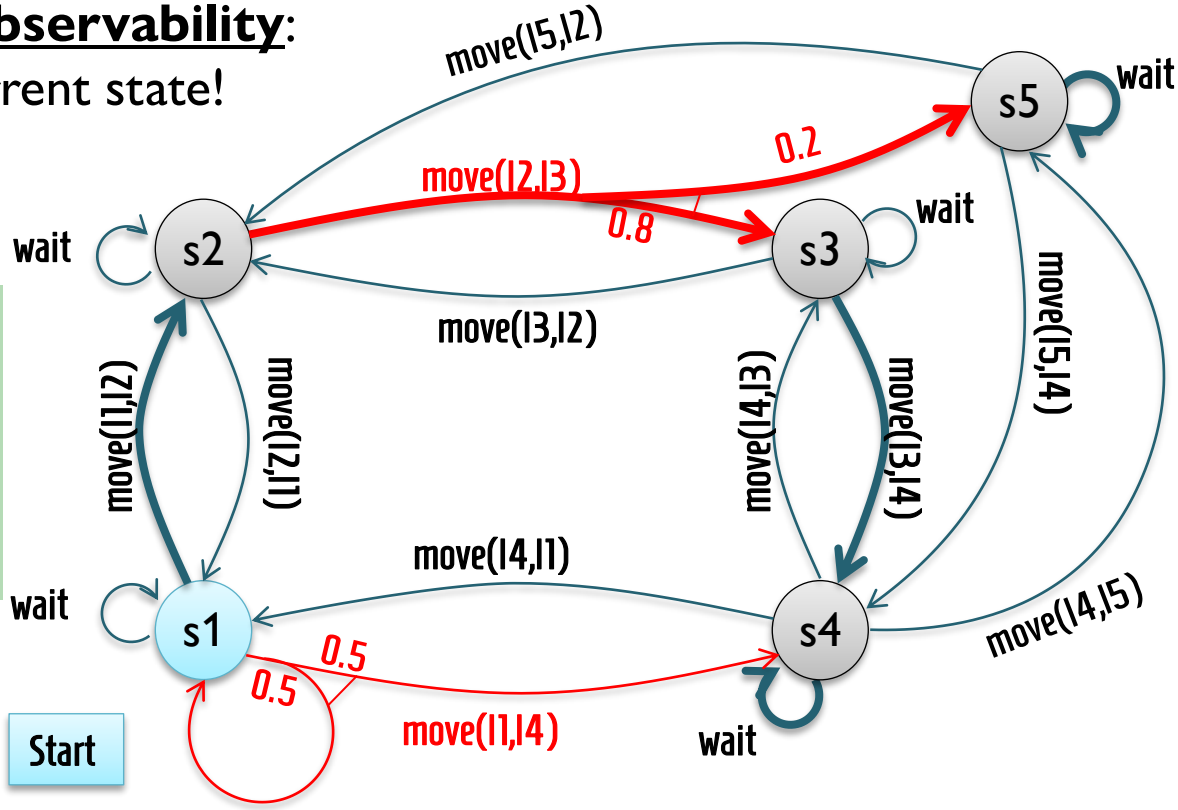
Important concepts,
before we define the planning problem itself!

Policies; Example 1

jonkv@ida

- One type of formal plan structure: **Policy** $\pi : S \rightarrow A$
  - Defining, **for each state**, which action to execute **whenever** we are there
  - Possible due to **full observability**:
    We **will** know the current state!

- Example 1
  - π1 = { (**s1**, move(l1,l2)),
    (**s2**, move(l2,l3)),
    (**s3**, move(l3,l4)),
    (**s4**, wait),
    (**s5**, wait)}

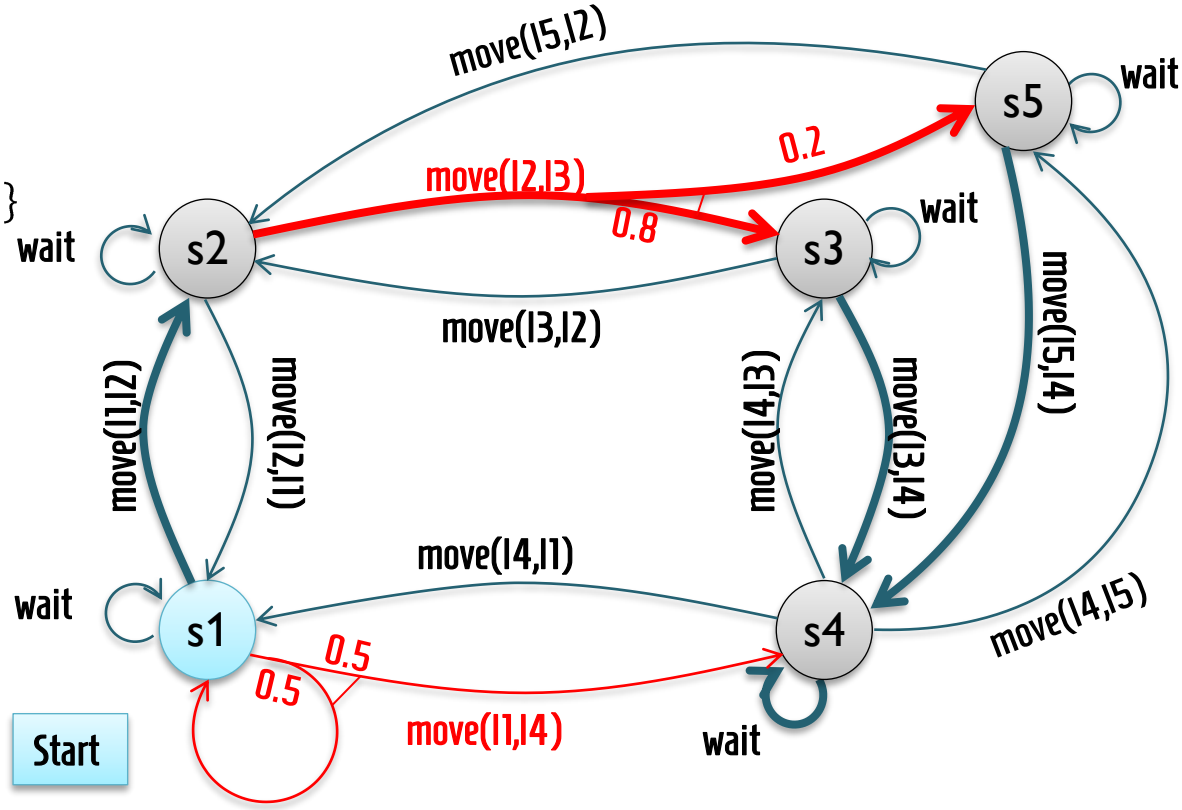


Reaches s4 or s5, waits there infinitely many times

Policy Example 2

jonkv@ida

- Example 2
  - π2 = { (s1, move(l1,l2)),
    (s2, move(l2,l3)),
    (s3, move(l3,l4)),
    (s4, wait),
    (s5, **move(l5,l4)**)}

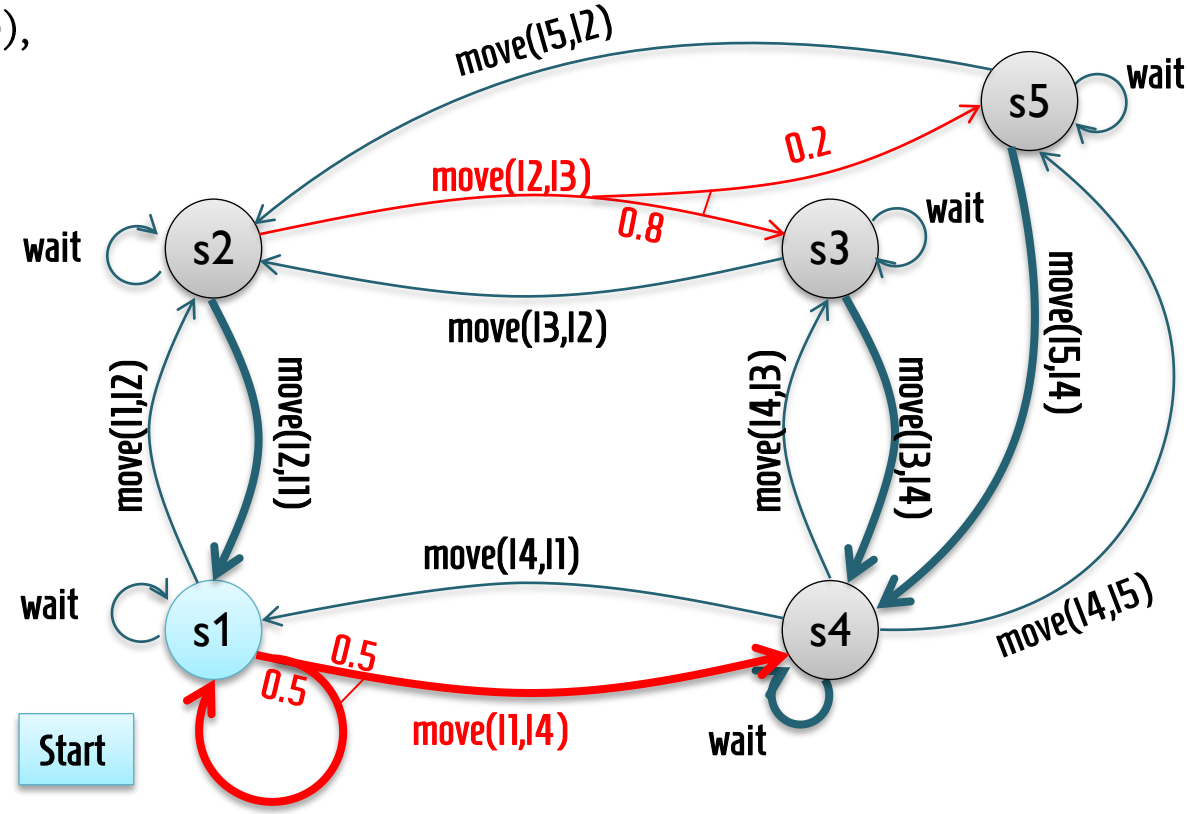


Always reaches state s4, waits there infinitely many times

Policy Example 3

jonkv@ida

- Example 3
  - π3 = { **(s1, move(l1,l4))**,
    (s2, move(l2,l1)),
    (s3, move(l3,l4)),
    (s4, wait),
    (s5, move(l5,l4)}



Reaches state s4 with 100% probability ”in the limit”
(the more steps, the greater the probability)

Policies and Histories

jonkv@ida

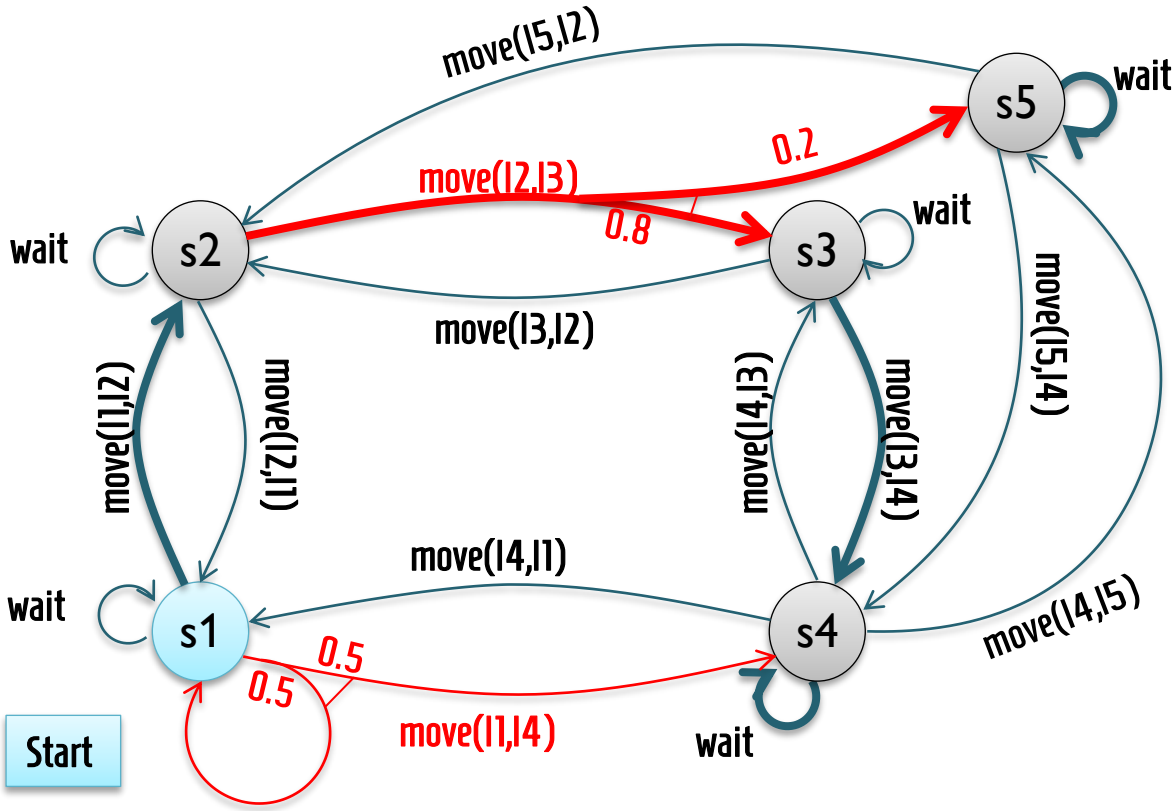
- The **outcome** of sequentially executing a policy:
  - A **state sequence** $h = \langle s_0, s_1, s_2, s_3, s_4, \ldots \rangle$, called a **history**
  - Infinite, since policies have no termination criterion

- For each policy, there can be **many potential histories**
  - Which one is the actual result?
    Gradually discovered at execution time!

History Example

jonkv@ida

- Example 1
  - π1 = { (s1, move(l1,l2)),
    (s2, move(l2,l3)),
    (s3, move(l3,l4)),
    (s4, wait),
    (s5, wait)}



- Even if we know we start in s1: **Two possible histories**
  - h_1 = ⟨s1, s2, s3, s4, s4, … ⟩ – Reached s4, waits indefinitely
    h_2 = ⟨s1, s2, s5, s5 … ⟩ – Reached s5, waits indefinitely

How probable are these histories?

Probabilities: Initial States, Transitions

jonkv@ida

- Each policy has a **probability distribution** over **histories/outcomes**
  - With known fixed initial state s_0:

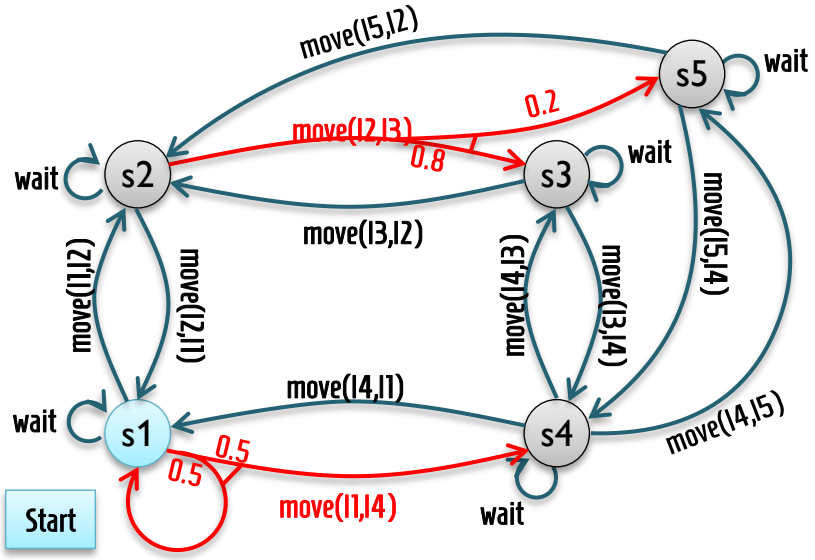
$$P(\langle s_0, s_1, s_2, s_3, \ldots \rangle \mid \pi) = \prod_{i \geq 0} P(s_i, \pi(s_i), s_{i+1})$$

Probabilities for each required state transition

  - With unknown initial state:

$$P(\langle s_0, s_1, s_2, s_3, \ldots \rangle \mid \pi) = P(s_0) \cdot \prod_{i \geq 0} P(s_i, \pi(s_i), s_{i+1})$$

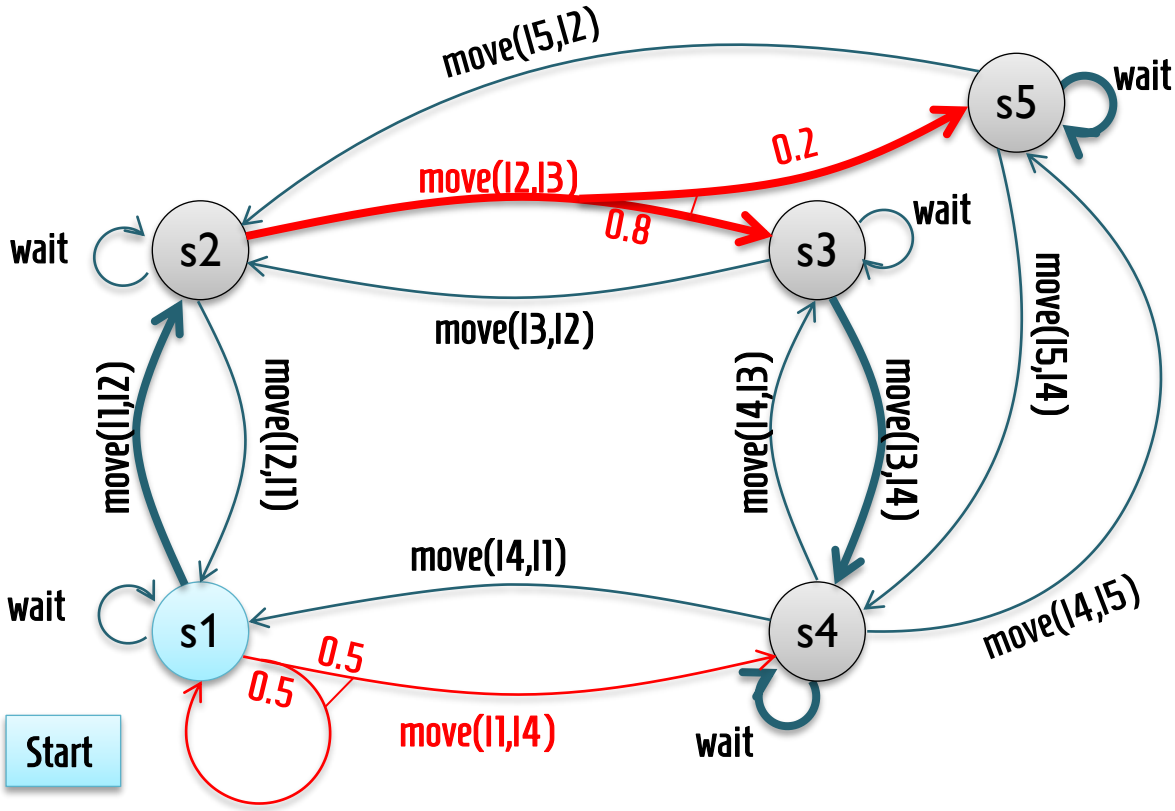
Probability of starting in this specific s_0



History Example 1

jonkv@ida

- Example 1
  - π1 = { (s1, move(l1,l2)),
    (s2, move(l2,l3)),
    (s3, move(l3,l4)),
    (s4, wait),
    (s5, wait)}

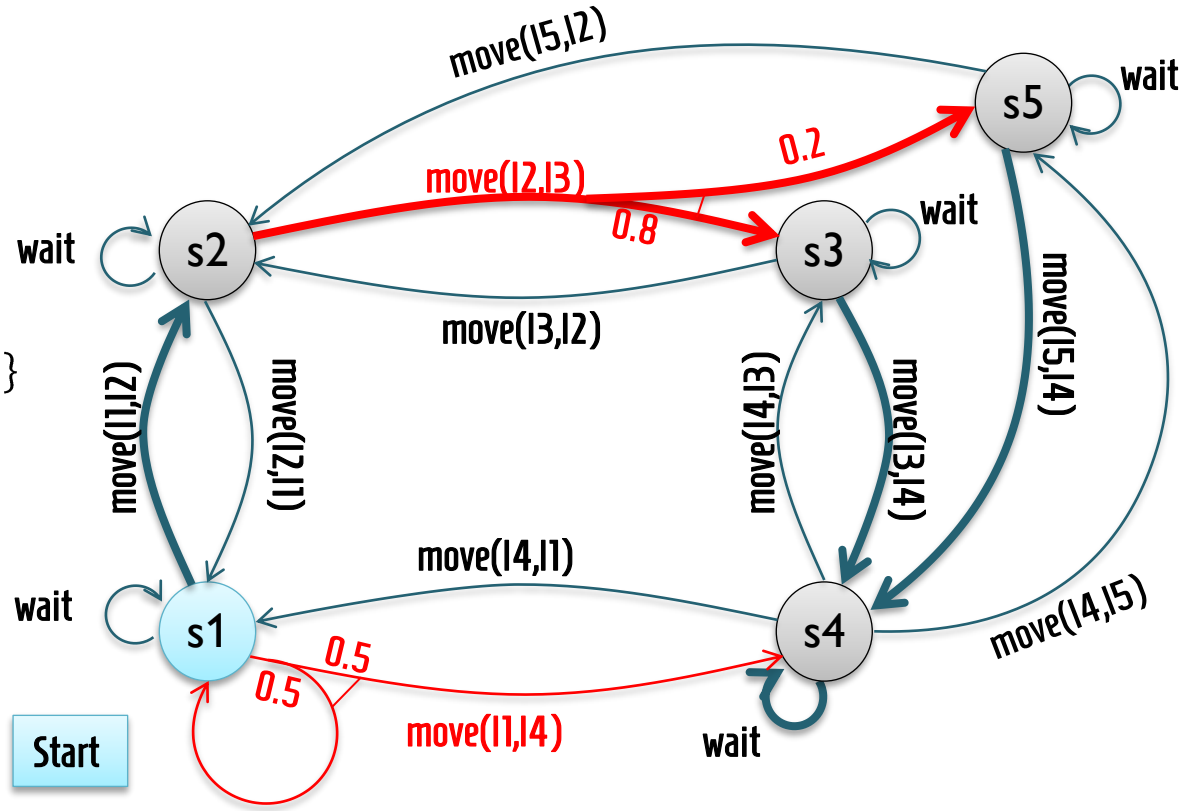


- Two possible histories, if $P(s1) = 1$:
  - h_1 = ⟨s1, s2, s3, s4, s4, … ⟩ – $P(h_1 \mid \pi_1) = 1 \times 1 \times 0.8 \times 1 \times \ldots = 0.8$
    h_2 = ⟨s1, s2, s5, s5 … ⟩ – $P(h_2 \mid \pi_1) = 1 \times 1 \times 0.2 \times 1 \times \ldots = 0.2$
    – $P(h \mid \pi_1) = 1 \times 0 = 0$ for all other h

History Example 2

jonkv@ida

- Example 2
  - π2 = { (s1, move(l1,l2)),
    (s2, move(l2,l3)),
    (s3, move(l3,l4)),
    (s4, wait),
    (s5, **move(l5,l4)**)}

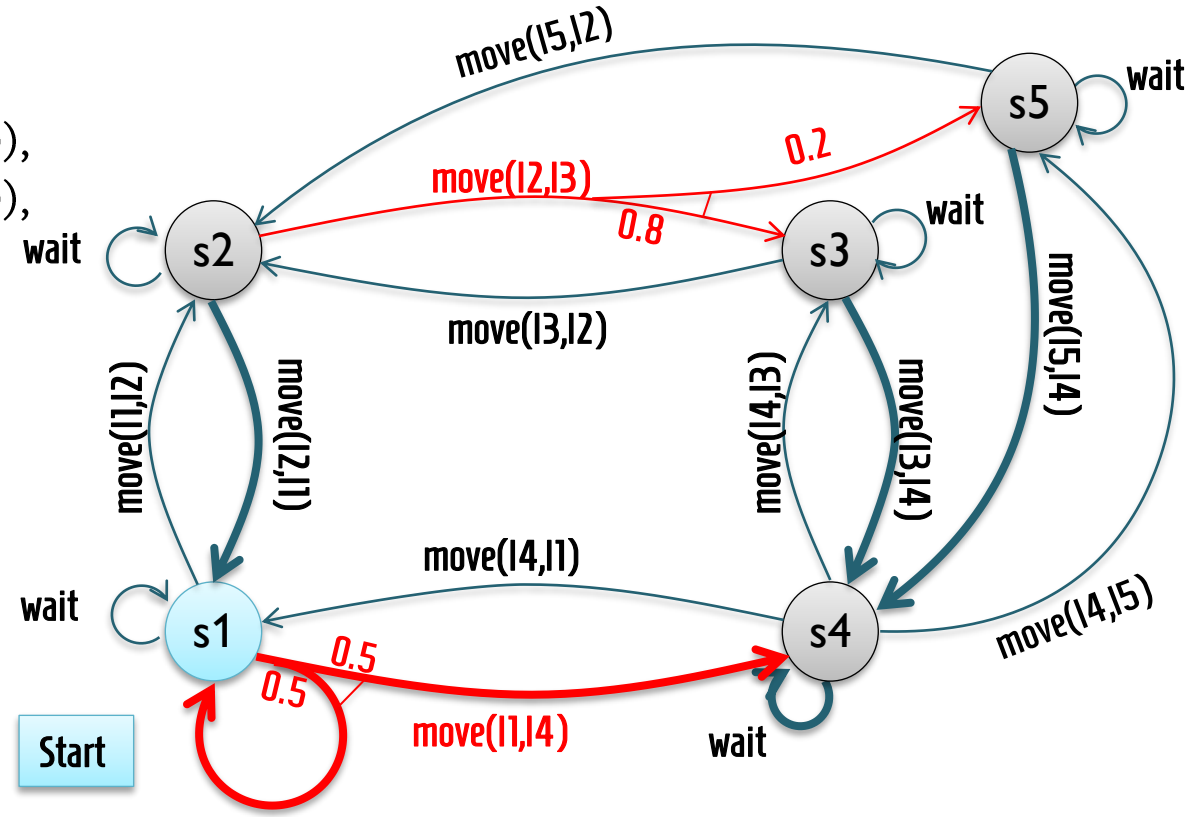


- h_1 = ⟨s1, s2, s3, s4, s4, … ⟩ $P(h_1 \mid \pi_2) = 1 \times 1 \times 0.8 \times 1 \times \ldots = 0.8$
  h_3 = ⟨s1, s2, s5, s4, s4, … ⟩ $P(h_3 \mid \pi_2) = 1 \times 1 \times 0.2 \times 1 \times \ldots = 0.2$
  $P(h \mid \pi_2) = 1 \times 0$ for all other h

History Example 3

jonkv@ida

- Example 3
  - π3 = { **(s1, move(l1,l4))**, **(s2, move(l2,l1))**, (s3, move(l3,l4)), (s4, wait), (s5, move(l5,l4)}



- $h_4 = \langle s1, s4, s4, \dots \rangle$ $P(h_4 \mid \pi_3) = 0.5 \times 1 \times 1 \times 1 \times 1 \times \dots = 0.5$
  $h_5 = \langle s1, s1, s4, s4, \dots \rangle$ $P(h_5 \mid \pi_3) = 0.5 \times 0.5 \times 1 \times 1 \times 1 \times \dots = 0.25$
  $h_6 = \langle s1, s1, s1, s4, s4, \dots \rangle$ $P(h_6 \mid \pi_3) = 0.5 \times 0.5 \times 0.5 \times 1 \times 1 \times \dots = 0.125$
  …
  $h_\infty = \langle s1, s1, s1, s1, s1, s1, \dots \rangle$ $P(h_\infty \mid \pi_3) = 0.5 \times 0.5 \times 0.5 \times 0.5 \times 0.5 \times \dots = 0$

Objectives:
Rewards for Indefinite Execution

Generalizating our Objectives

jonkv@ida

- Policies allow *indefinite execution*
  - No predetermined termination criterion – go on "forever"
    - **def** execute_policy(π):
      while True:
      s = sense_current_state()
      a = $\pi(s)$
      execute_action(a)

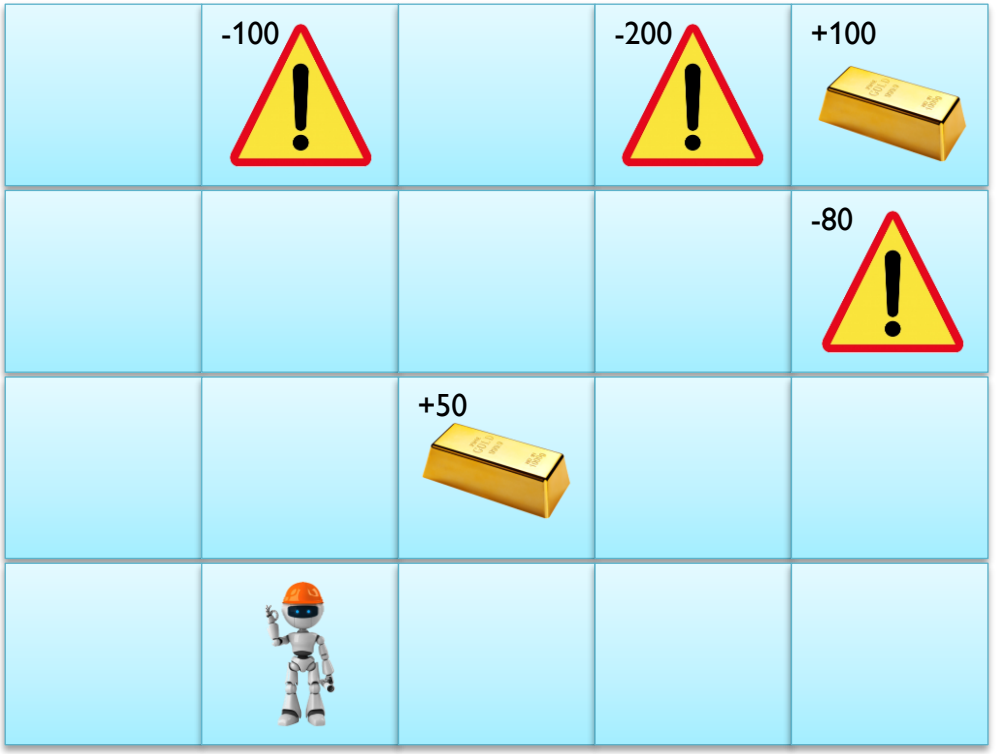
But without termination, there can't be goal states…
So what is the objective? What is a good policy?

- Combination of:
  - **Cost function** $c: (S, A) \rightarrow \mathbb{R}$
    - $c(s, a)$ = cost of **being in** state s and **executing** action a
  - **Reward function** $R: (S, A, S) \rightarrow \mathbb{R}$
    - $R(s, a, s')$ = Reward for **being in** s, **executing** a and **actually ending up** in s'

Example: Grid World

jonkv@ida

- Example: **Grid World**
  - **Actions**: North, South, West, East, NorthWest, …, TakeGold
    - Cost $c(s, a) = 15$ for all s, a
    - 90% chance: Go where you want
    - 10% risk: End up somewhere else
  - **Rewards** for some transitions
    - $R(s, a, s') = +100$
      for transitions when you
      take the gold in the top right cell
    - s = [top right, there is gold]
      a = TakeGold
      s' = [top right, there is no gold]
  - **Danger** in some cells
    - Try to go to the top right cell
    - $R(s, a, s') = 0$ usually
    - $R(s, a, s') = -200$
      if you accidentally end up
      in the danger cell

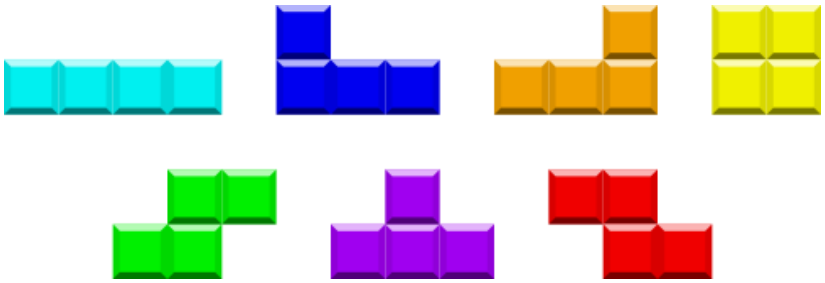


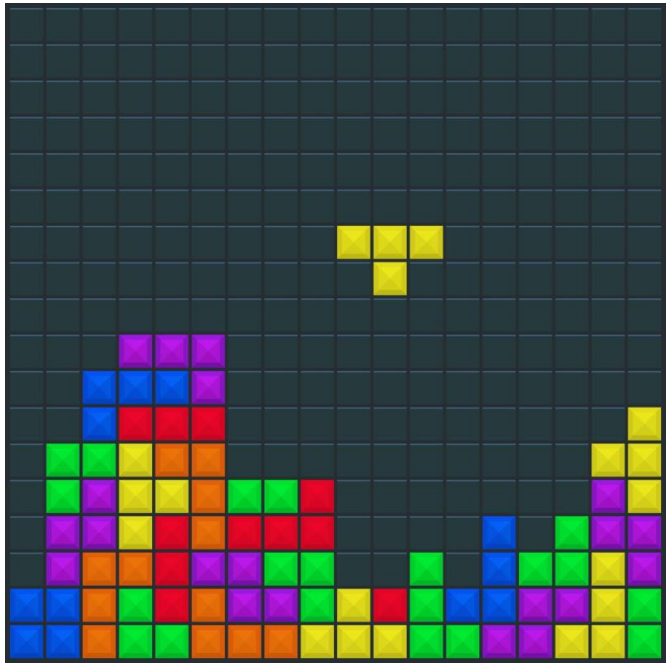
Important:
States != locations
Can't take the gold twice,
can't gain infinite rewards

Example: Tetris

jonkv@ida

- In each "step", **a piece falls one row** – and you execute **one action**
  - Guide the pieces left/right, rotate them, drop them
- If an action/step results in **filling a row**:
  - The line disappears
  - $R(old\ state, action, state\ with\ row\ removed) = 100$
- If an action/step results in **filling two rows**:
  - $R(old\ state, action, state\ with\ 2\ rows\ removed) = 400$
- When a piece has **fallen all the way**:
  - A new ***random*** piece falls from the top
  - Model piece probabilities using $P(s, a, s')$ according to (most types of) Tetris rules

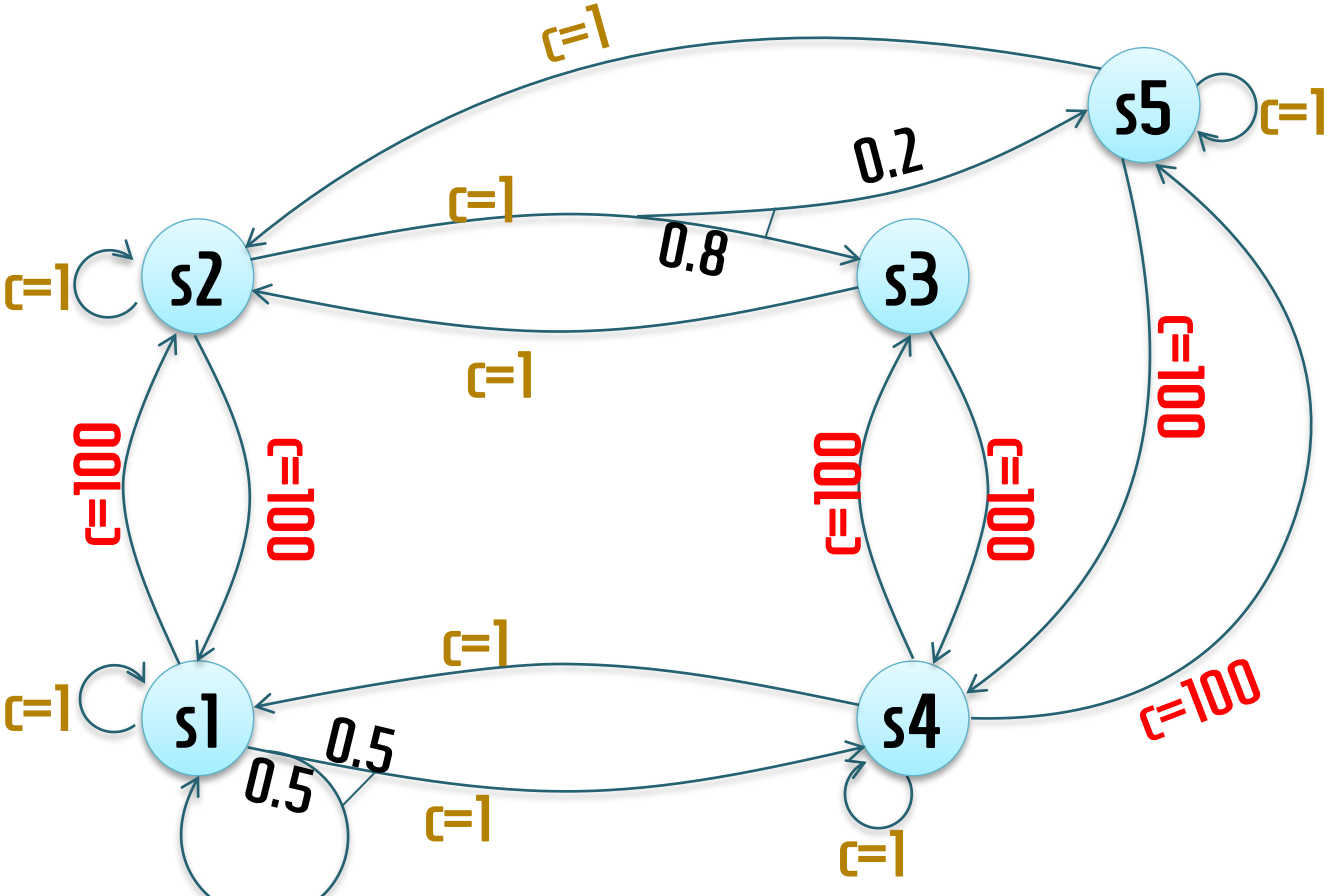




Example: Robot Navigation

jonkv@ida

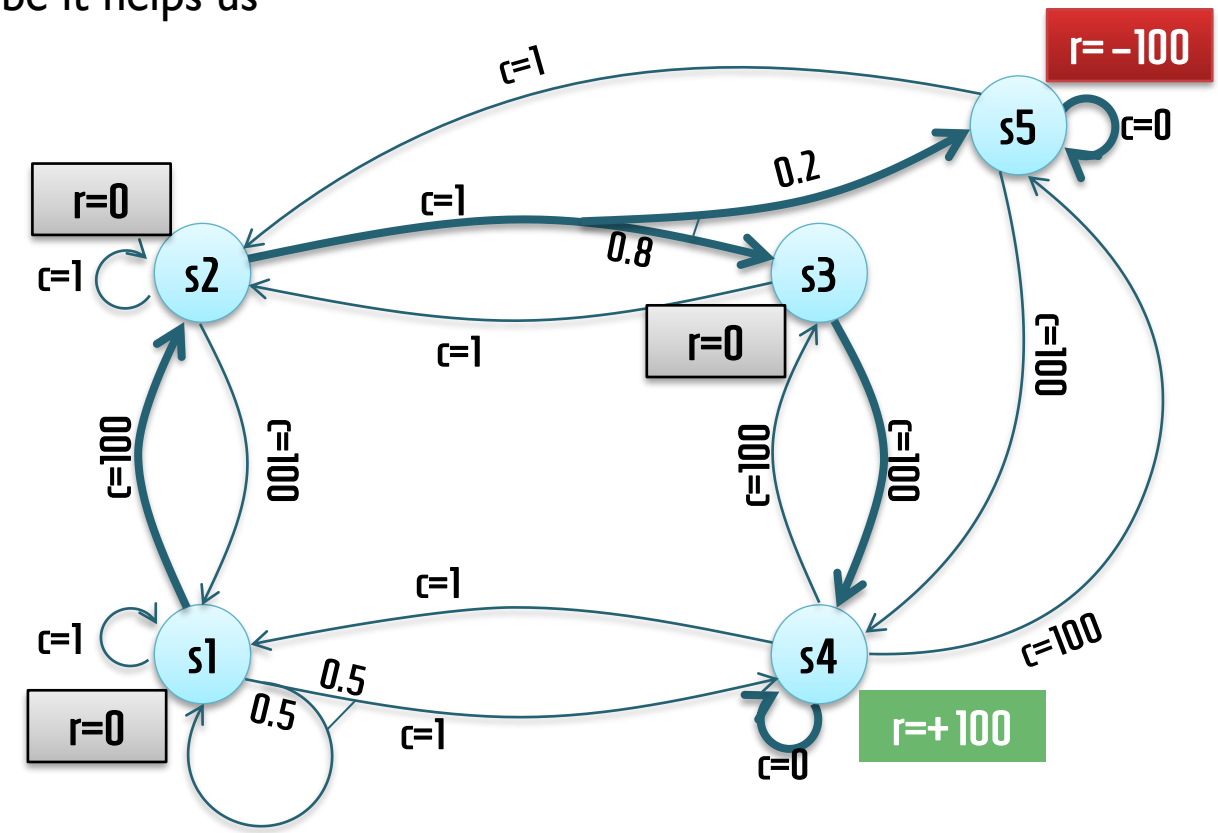
- Example costs in robot navigation:
  - $c(s,a) = 1$ for each "horizontal" action
  - $c(s,a) = 100$ for each "vertical" action: Far away, difficult, …
  - $c(s, wait) = 1$



Example: Robot Navigation (2)

jonkv@ida

- Example rewards in robot navigation
  - Every time you **end up in s5**:
    - Negative reward – maybe the robot is in our way
  - Every time you **end up in s4**:
    - Positive reward – maybe it helps us



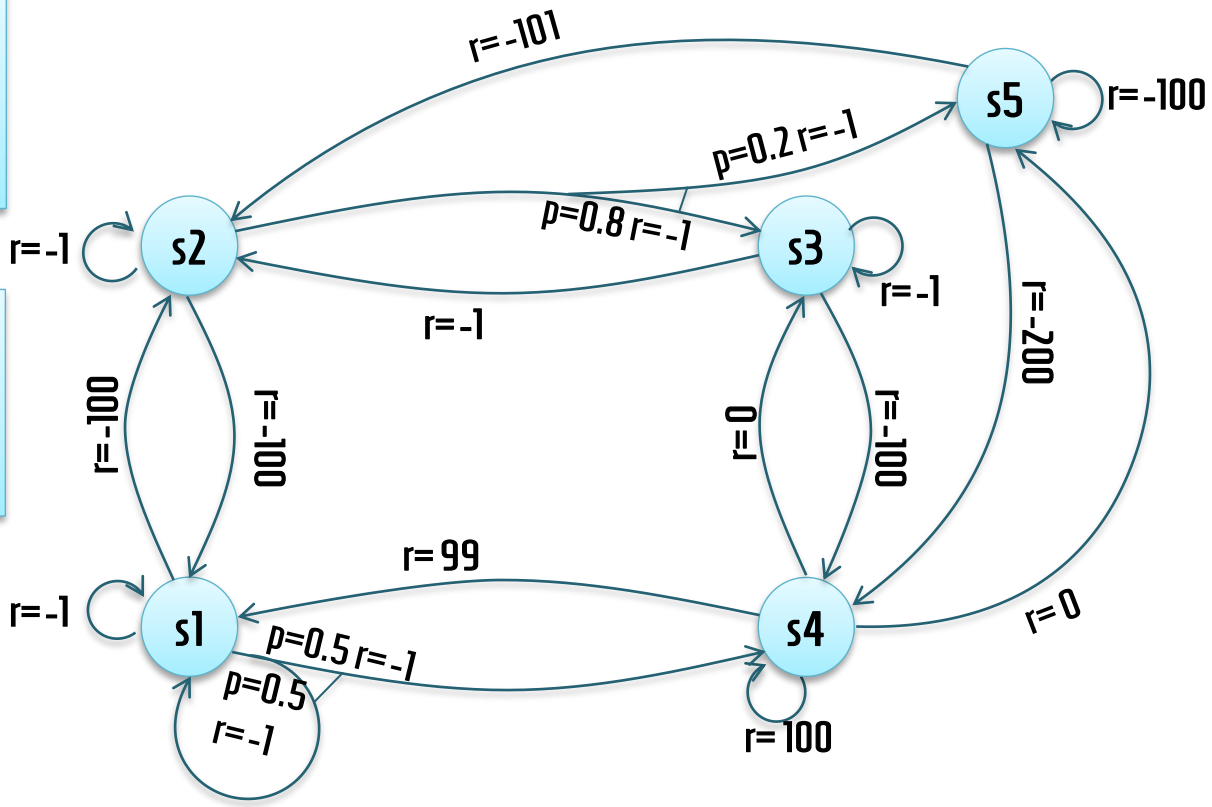
Simplification

jonkv@ida

- To simplify formulas, **include the cost in the reward**!
  - Decrease each $R(s_i, \pi(s_i), s_{i+1})$ by $C(s_i, \pi(s_i))$

C(s1, move(l1,l2)) = **100**
R(s1, move(l1,l2), s2) = **0**

R(s1, move(l1,l2), s2) = **-100**



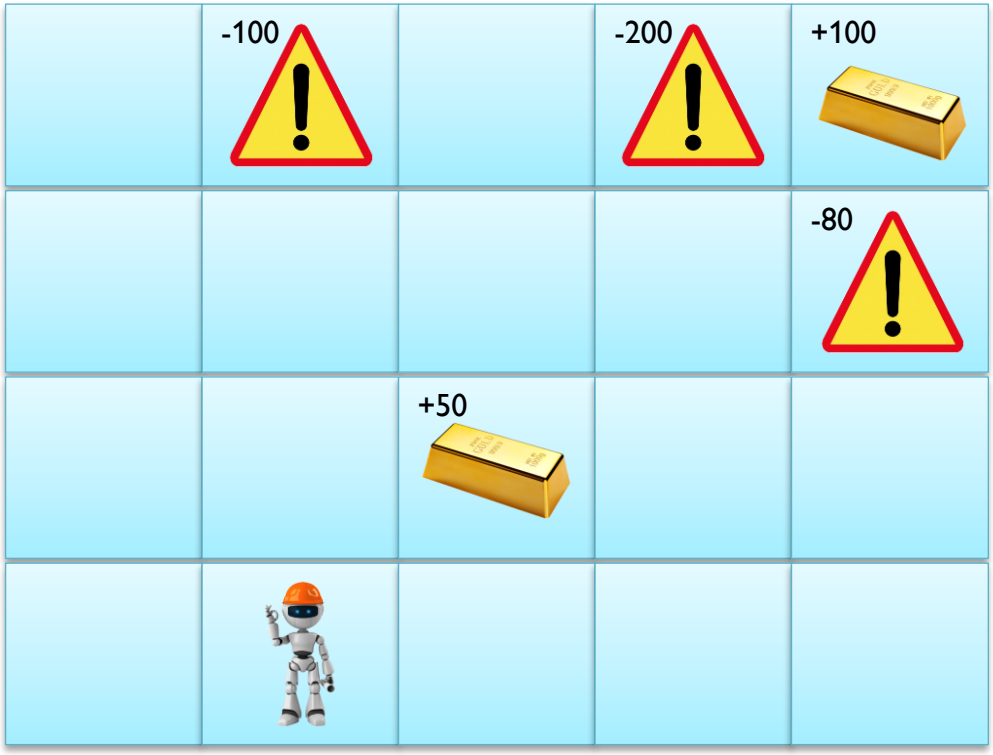
Utility Functions

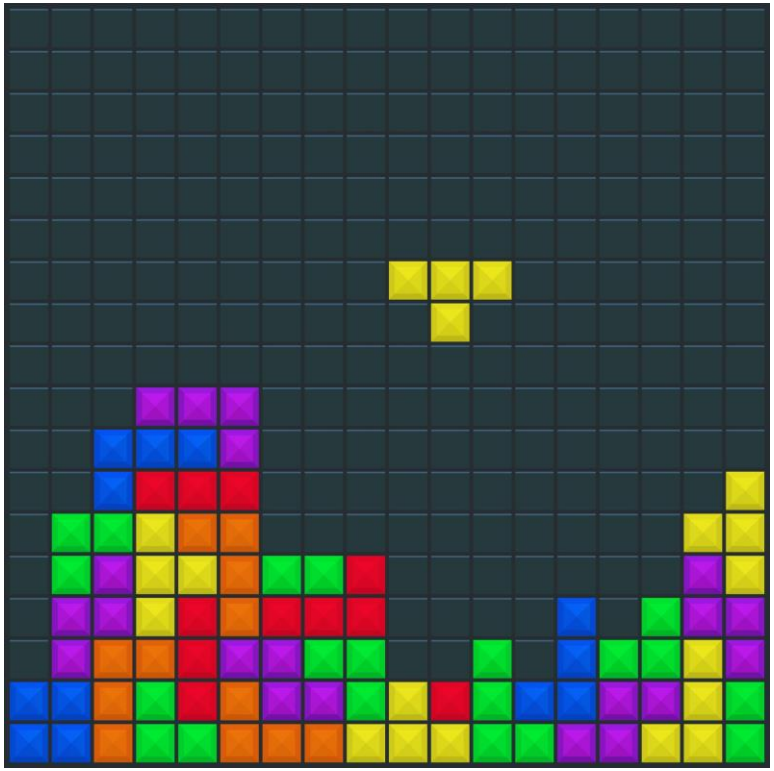
How useful is an outcome to us?

Total Rewards – In Advance?

jonkv@ida

- Given a policy π, what **will** our total rewards be?
  - Can't know in advance
    - *Will I reach the goal or end up in the danger zone?*
    - *Which pieces will I get?*





Total Rewards – After Executing?

jonkv@ida

- Given a policy π…
  - …**and** an **outcome**, a **history** (*infinite* state sequence) $h = \langle s_0, s_1, s_2, \ldots \rangle$ resulting from actually having executed π…

- …What **were** our total rewards?
  - *Undiscounted* **utility of a history**:

$$\mathrm{V}(h|\pi) = \sum_{i \geq 0} R(s_i, \pi(s_i), s_{i+1})$$

    - I was in s_0, executed $\pi(s_0)$, and ended up in s_1 -- reward!
    - I was in s_1, executed $\pi(s_1)$, and ended up in s_2 −− reward!
    - I was in s_2, executed $\pi(s_2)$, and ended up in s_3 −− reward!
    - …

Discount Factors and Discounted Utility

Utility in a Context

jonkv@ida

Policy = solution for **infinite** horizon

(**Infinite** actual execution)

Never ends – unrealistic

Indefinite execution

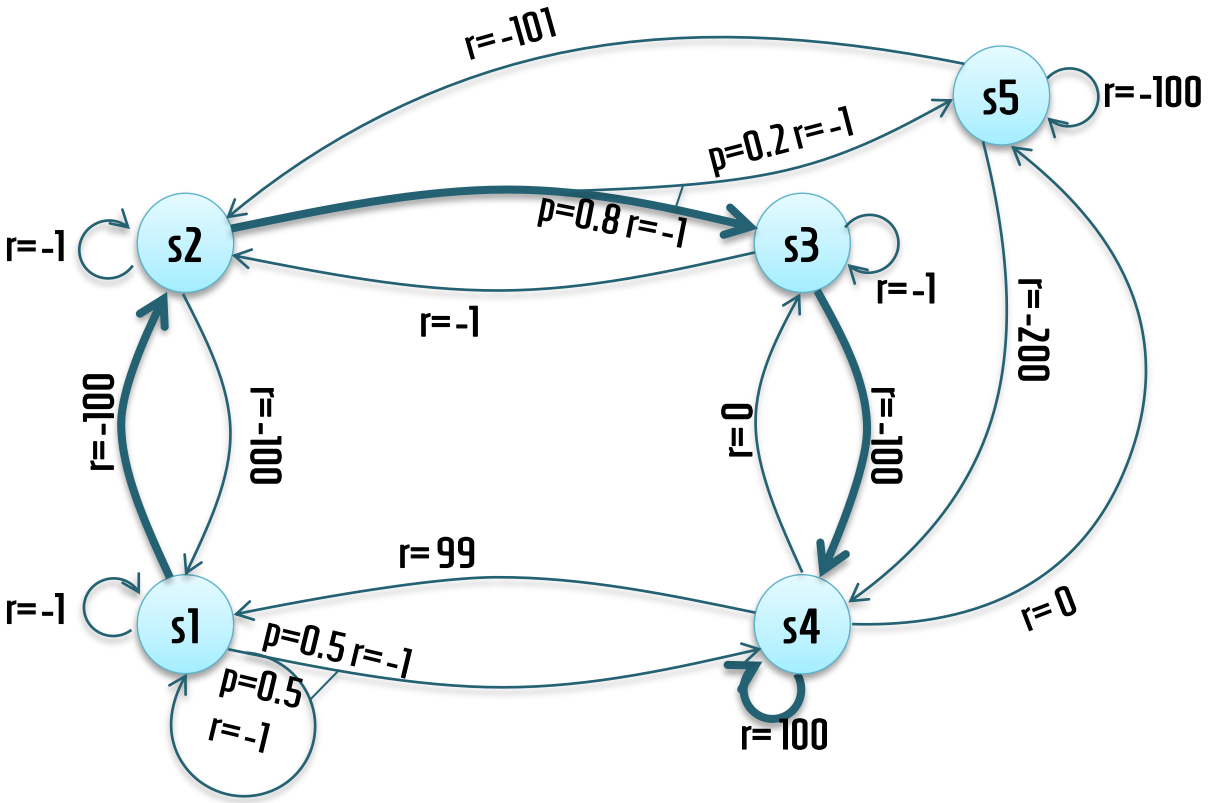
We *will* stop at some point
(the universe will end),
but we can't predict *when*

To find the best policy
for *long term execution*:
Consider the infinite case

Infinite Undiscounted Utility

jonkv@ida

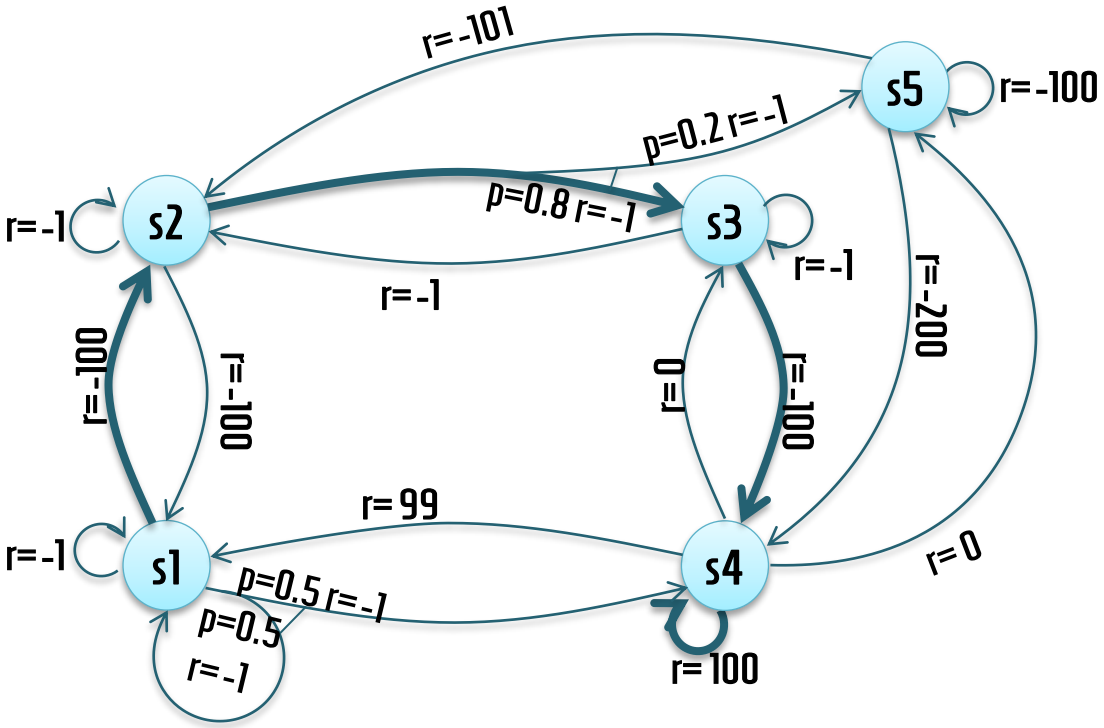
- If we use **undiscounted utility** for an **infinite history**:
  - π_1 **could** result in $h_1 = \langle s1, s2, s3, s4, s4, \ldots \rangle$
    - Stays at $s4$ forever, executing "wait" ➔ **infinite** amount of rewards!
    - $V(h_1 \mid \pi_1) = (-100) + (-1) + (-100) + 100 + 100 + 100 + 100 + 100 + \ldots$



Infinite Undiscounted Utility (2)

jonkv@ida

- **What's the problem**, if we "like" being in state s4?
  - Can't distinguish between **different ways** of getting there!
    - s1→s2→s3→s4: $-201 + \infty = \infty$
    - s1→s2→s1→s2→s3→s4: $-401 + \infty = \infty$
    - Both appear equally good…
  - Can't distinguish between infinite times **100** and and infinite times **1000**
  - Even without infinity, we can't see the difference between rewards **now** and rewards in the **far future**



Discounted Utility

jonkv@ida

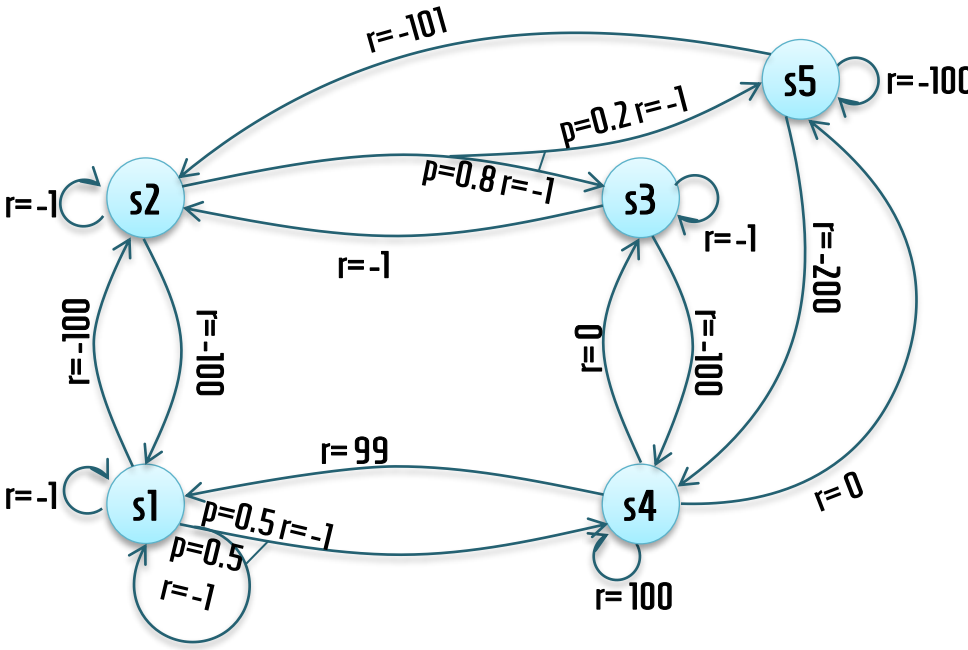
- Solution: **Discounted utility** for a history
  - Introduce a **discount factor**, γ, with $0 \leq \gamma \leq 1$
  - Let

$$V(h|\pi) = \sum_{i \geq 0} \gamma^i R(s_i, \pi(s_i), s_{i+1})$$

  - Distant rewards/costs have **less influence**
    - For example: 0.9, 0.81, 0.729, …
  - Discounted utility is **finite** as long as $0 \leq \gamma < 1$

Examples will use $\gamma = 0.9$

Only to simplify formulas!
Should choose carefully…



Expected Utility of a Policy

Expected Utility of a Policy

jonkv@ida

- We want to choose a **good policy**
  - We know, for **each** history (outcome) $h = \langle s_0, s_1, s_2, s_3, s_4, \dots \rangle$ of a policy π:
    - The ***probability*** that the history will occur: $P(h|\pi)$
    - The resulting *actual* ***discounted utility***: $V(h|\pi) = \sum_{i \geq 0} \gamma^i R(s_i, \pi(s_i), s_{i+1})$
  - Using this, calculate the statistically **expected utility** (~"average" utility) for the entire **policy**:

$$E(\pi) = \sum_{h \in \{\text{all possible histories for } \pi\}} P(h|\pi) V(h|\pi)$$

  - Or, the **expected utility** given that we **start execution in state s**:

$$E(\pi, s) = \sum_{h \in \{\text{all possible histories for } \pi\}} P(h\,|\pi, s_0 = s) V(h|\pi)$$

Strictly speaking, this is the expected discounted utility (but expected undiscounted utility is rarely used)

Remembering the Notation...

jonkv@ida

- How to remember the notation?
  - V – The *actual* **V**alue.
    Value depends on rewards,
    and rewards depend on the state we actually ended up in,
    so we can only know the *actual value* if we know the *actual outcome*,
    which is represented as a *history*.

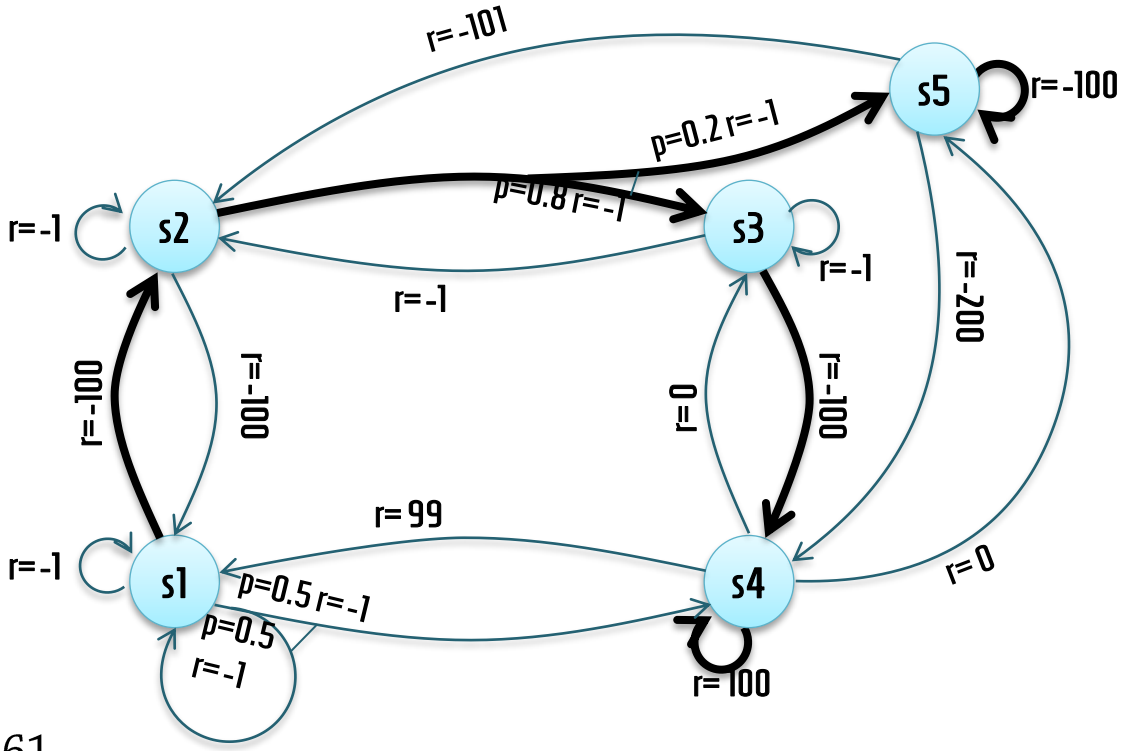
  - E – the ***E**xpected* value, in the statistical sense.
    Since it is only *expected*, it isn't certain yet,
    and that's because it takes into account *all possible outcomes* (histories)
    and the resulting rewards and probabilities.

Example 1

jonkv@ida

π_1 = {(s1, move(l1,l2)),
(s2, move(l2,l3)),
(s3, move(l3,l4)),
(s4, wait),
(s5, wait)}

Given that we start in s1,
π_1 can lead to only **two** histories:
80% chance of history h1,
20% chance of history h2



$\gamma = 0.9$

Factors 1, 0.9, 0.81, 0.729, 0.6561…

$h_1 = \langle s1, s2, s3, s4, s4, \dots \rangle$

$V(h_1 \mid \pi_1) = .9^0(-100) + .9^1(-1) + .9^2(-100) + .9^3 100 + .9^4 100 + \dots \approx 547.9$

$h_2 = \langle s1, s2, s5, s5 \dots \rangle$

$V(h_2 \mid \pi_1) = .9^0(-100) + .9^1(-1) + .9^2(-100) + .9^3(-100) + \dots \approx -910.1$

$E(\pi_1) = 0.8 * 547.9 + 0.2 (-910.1) = 256.3$

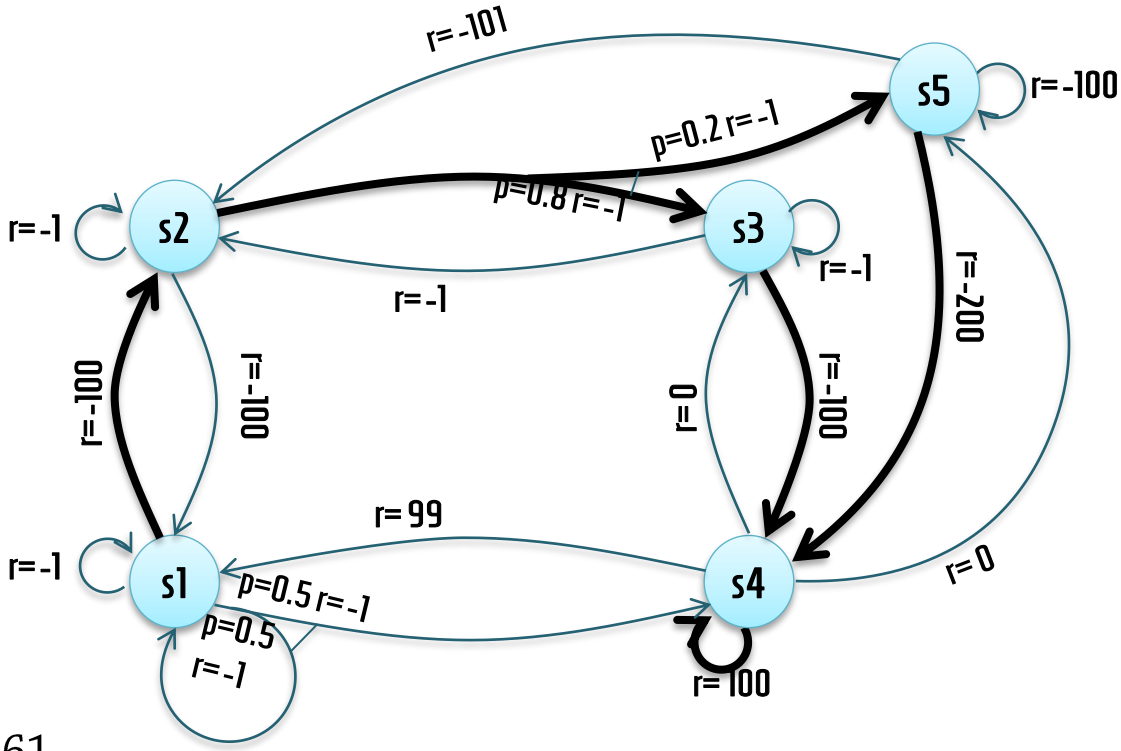
We expect a reward of 256.3 on average

Example 2

jonkv@ida

π_2 = {(s1, move(l1,l2)),
(s2, move(l2,l3)),
(s3, move(l3,l4)),
(s4, wait),
(s5, **move(l5,l4)**}

Given that we start in s1,
also **two** different histories…
80% chance of history h1,
20% chance of history h2



$\gamma = 0.9$

Factors 1, 0.9, 0.81, 0.729, 0.6561…

$h_1 = \langle s1, s2, s3, s4, s4, \dots \rangle$

$V(h_1 \mid \pi_1) = .9^0(100) + .9^1(-1) + .9^2(-100) + .9^3 100 + .9^4 100 + \dots = 547.9$

$h_2 = \langle s1, s2, s5, s4, s4, \dots \rangle$

$V(h_2 \mid \pi_1) = .9^0(-100) + .9^1(-1) + .9^2(-200) + .9^3 100 + \dots = 466.9$

$E(\pi_2) = 0.8 * 547.9 + 0.2 (466.9) = 531{,}7$

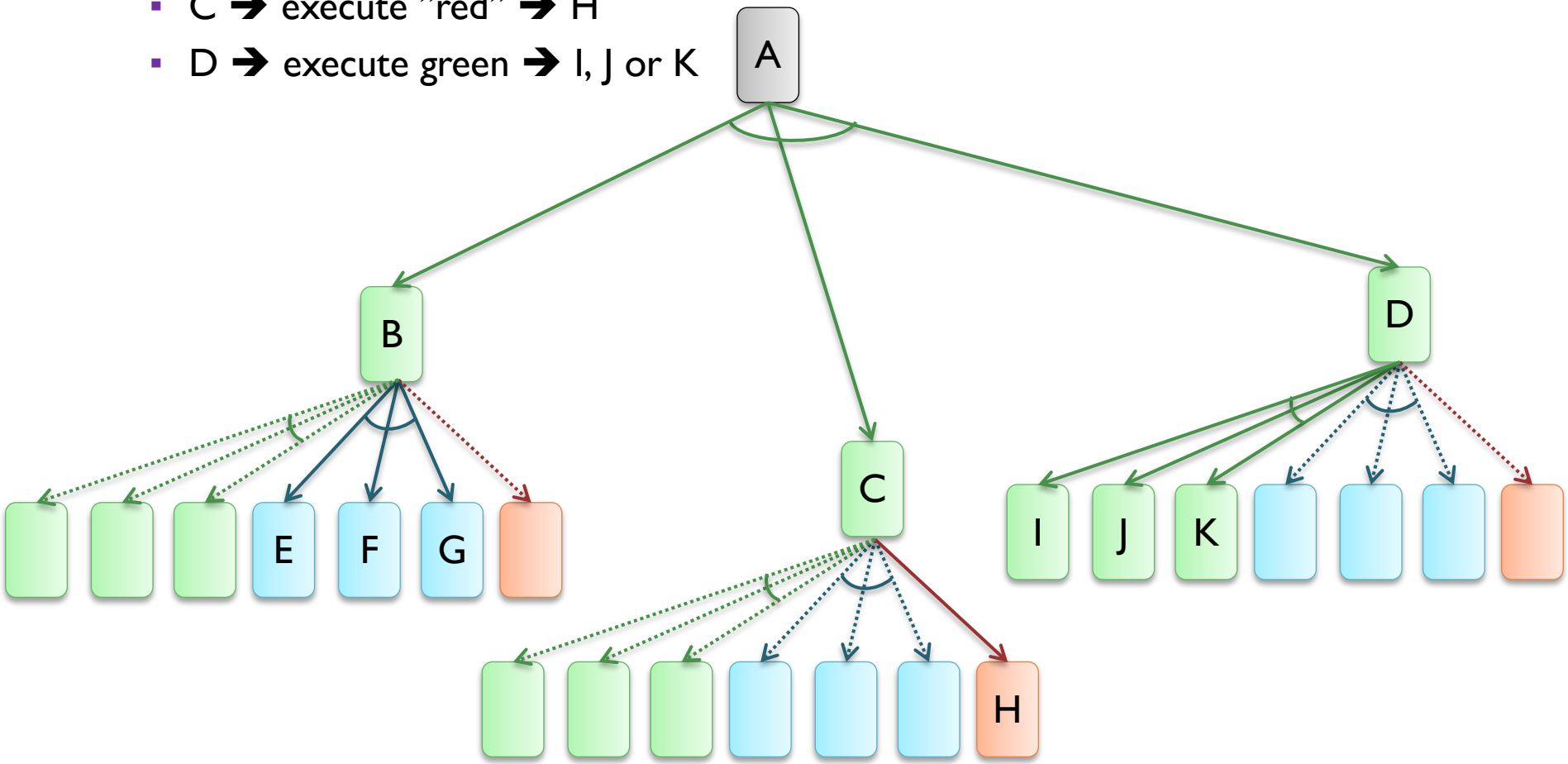
Expected reward 531.7 (π_1 gave 256.3)

Expected Utility of a Policy: Another Definition

Expected Utility: Example

jonkv@ida

- Consider a policy…
  - In state A, we should execute the ”green action”, which might lead to:
    - B ➔ execute ”blue” ➔ E, F or G
    - C ➔ execute ”red” ➔ H
    - D ➔ execute green ➔ I, J or K

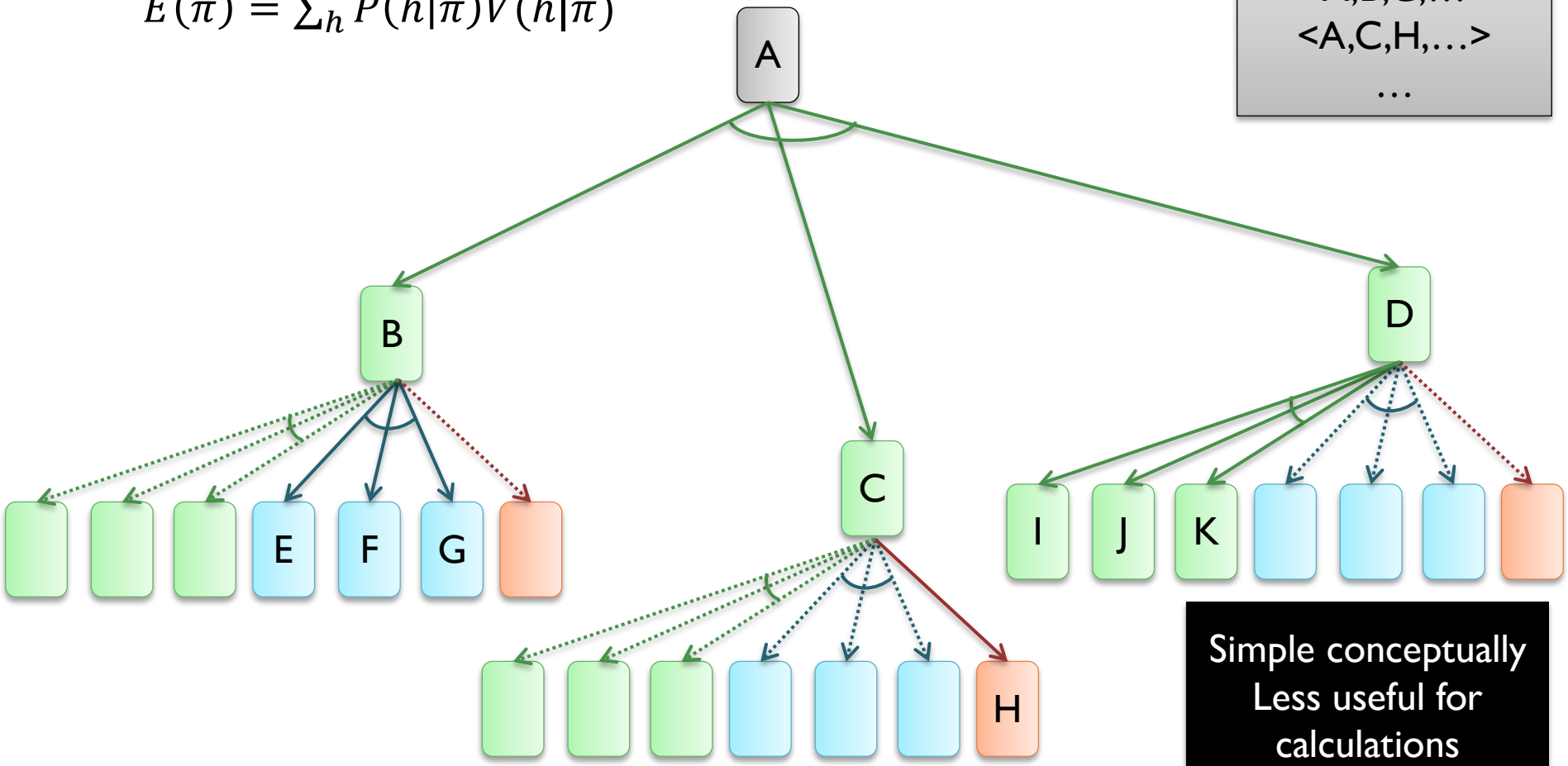


Expected Utility: History-based

jonkv@ida

- We calculated expected **utilities** based on **histories**
  - Find and iterate over ***all possible infinite histories:***
    $E(\pi) = \sum_h P(h|\pi)V(h|\pi)$

<A,B,E,…>
<A,B,F,…>
<A,B,G,…>
<A,C,H,…>
…

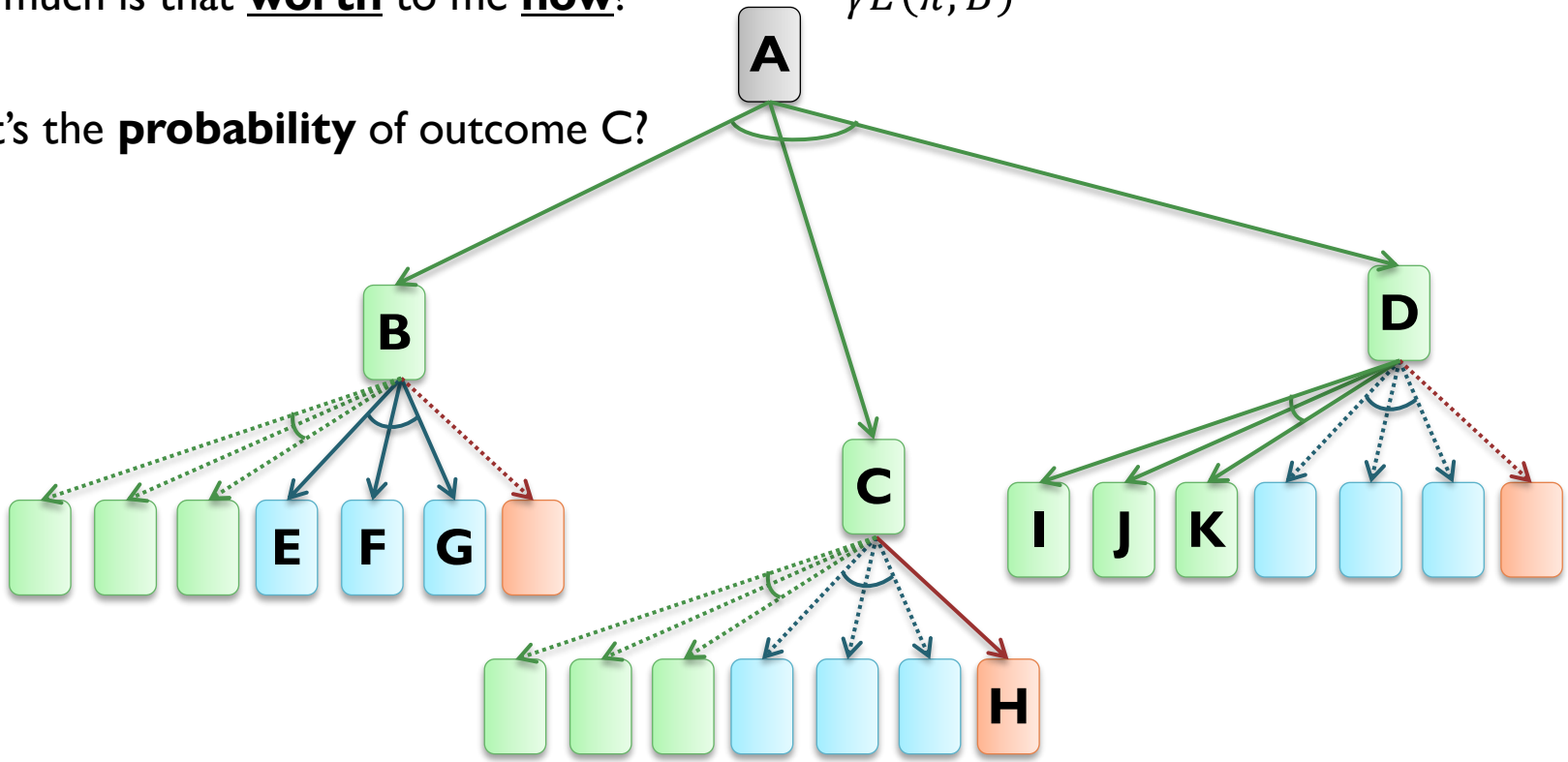


Simple conceptually
Less useful for calculations

Expected Utility: Step by Step

jonkv@ida

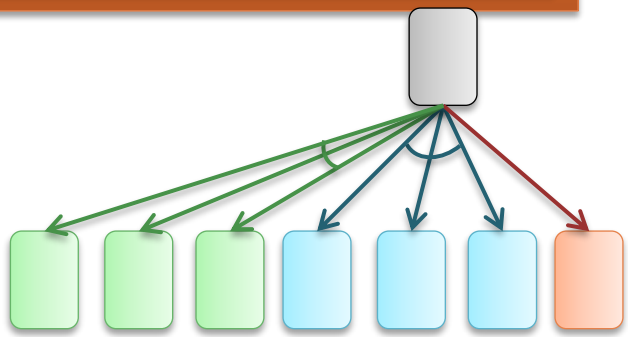
- Another computation method:
  - **We want** $E(\pi, A)$, and the selected action is $\pi(A) = green$
  - What's the **probability** of outcome B? $P(A, green, B)$
  - What's the **reward** for this outcome? $R(A, green, B)$
  - **How much more** will I get **after** arriving in B? $E(\pi, B)$, by definition!
  - How much is that **worth** to me **now**? $\gamma E(\pi, B)$
  - What's the **probability** of outcome C?
  - …
  - …



Expected Utility: Step by Step (2)

jonkv@ida

- If π is a policy, then
  - $E(\pi,s) = \sum_{s' \in S} P(s, \pi(s), s') * (R(s, \pi(s), s') + \gamma E(\pi,s'))$
  - The expected utility of continuing to execute π after having reached s
  - Is the sum, for all possible states $s' \in S$ that you might end up in (outcomes),
    - of the probability $P(s, \pi(s), s')$ of actually ending up in that state given the action $\pi(s)$ chosen by the policy, times
    - the reward you get for this transition
    - plus the discount factor times the expected utility $E(\pi, s')$ of **continuing** π from the new state s'

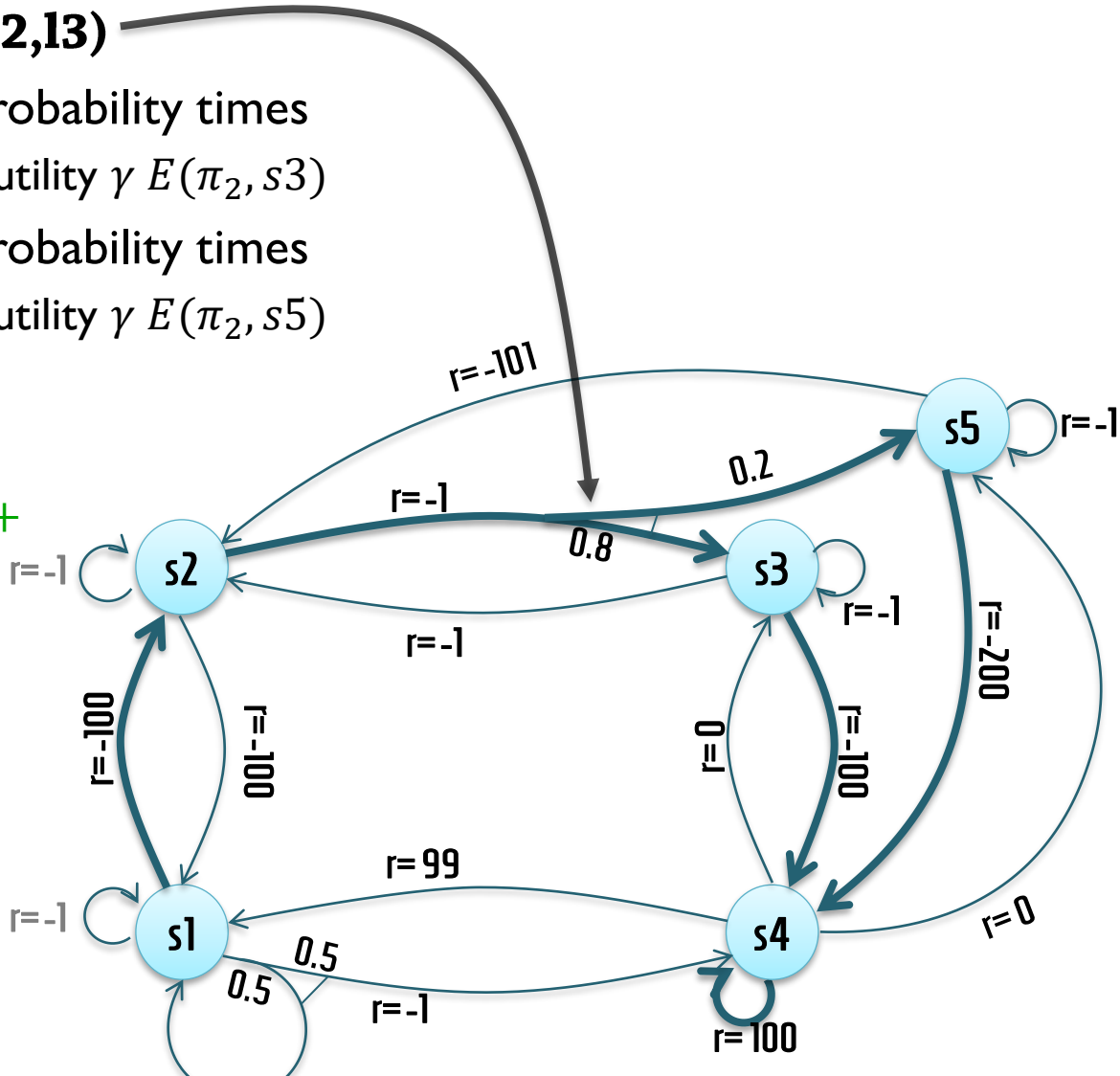


Example

jonkv@ida

- $E(\pi_2, s2)$ = the expected utility of executing $\boldsymbol{\pi_2}$ starting in **s2**:
  - Policy says: Use **move(l2,l3)**
  - Ending up in $s3$: 80% probability times
    - Reward -1 plus future utility $\gamma\ E(\pi_2, s3)$
  - Ending up in $s5$: 20% probability times
    - Reward -1 plus future utility $\gamma\ E(\pi_2, s5)$
  - So:
    - $E(\pi_2, s2) =$
      $0.8\big(-1 + \gamma E(\pi_2, s3)\big) +$
      $0.2\ \big(-1 + \gamma E(\pi_2, s5)\big)$

π_2 = {(s1, move(l1,l2)),
(s2, move(l2,l3)),
(s3, move(l3,l4)),
(s4, wait),
(s5, move(l5,l4)}

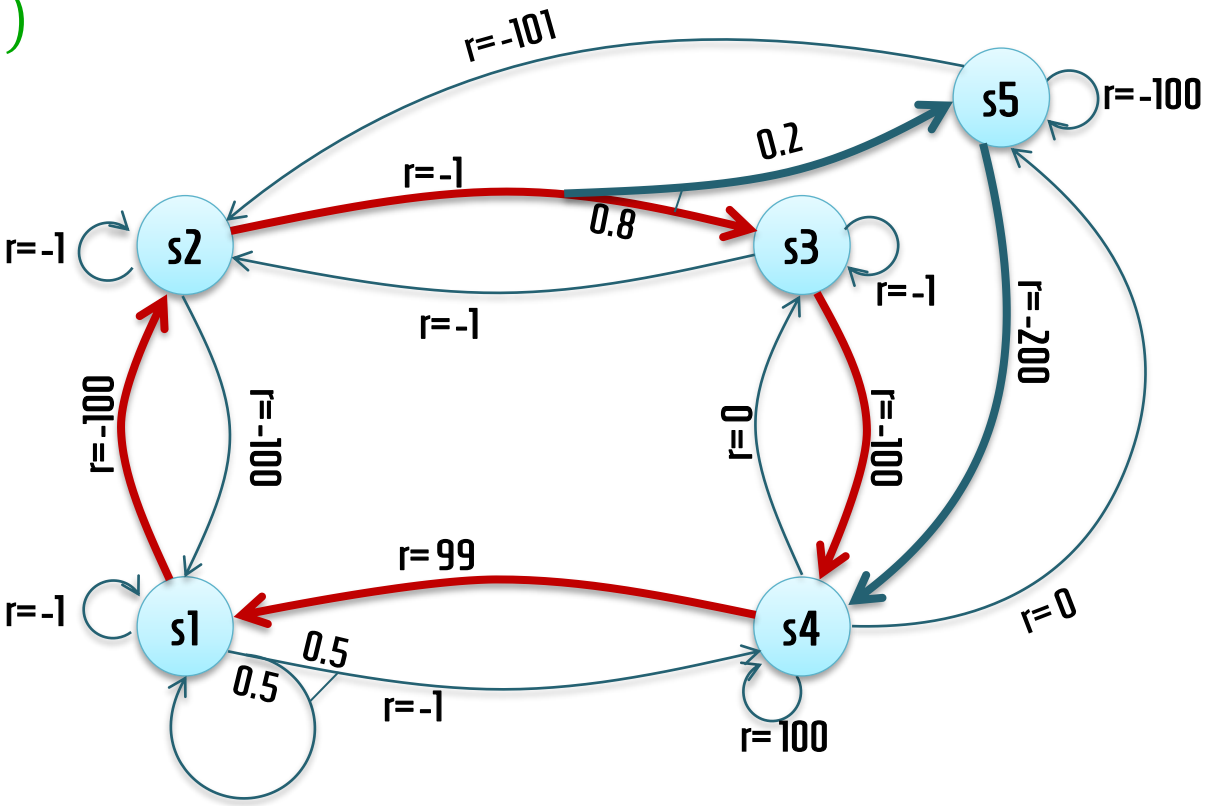


Trivial?

jonkv@ida

- Seems like a **trivial calculation**!
  - Each expected utility depends on a few others…
    so you'd just keep computing until you get to the end?
  - $E(\pi_2, s2) =$
    $0.8\big(-1 + \gamma E(\pi_2, s3)\big) +$
    $0.2\,\big(-1 + \gamma E(\pi_2, s5)\big)$

- **No: The graph often has cycles**
  - $E(\pi_2, s2)$ depends on
    $E(\pi_2, s3)$ ➔
    $E(\pi_2, s4)$ ➔
    $E(\pi_2, s1)$ ➔
    $E(\pi_2, s2)$ ➔ …



Equation System

jonkv@ida

- If π is a policy, then
  - $E(\pi,s) = \sum_{s' \in S} P(s, \pi(s), s') * (R(s, \pi(s), s') + \gamma E(\pi,s'))$

This is an **equation system**: |S| equations, |S| variables!

Use standard solution methods…

(Equation system example)

jonkv@ida

- Clarification: What do we mean by equation system?
  - Suppose you have 2 equations
    - $x = 5y$
    - $y = x - 20$
  - Can't solve recursively:
    - Start with $x = 5y$; you want the value of y
    - We know $y = x - 20$; we want the value of x, which is $5y$, which is …
  - Can solve as an equation system:
    - $x = 5(x - 20)$
    - $x = 5x - 100$
    - $0 = 4x - 100$
    - $100 = 4x$
    - $25 = x$

Fully Observable Probabilistic Planning: Markov Decision Processes

Markov Decision Processes

jonkv@ida

- **Markov Decision Processes**
  - Underlying world model: **Stochastic system**
  - Plan representation: **Policy** – which action to perform in **any** state
  - Goal representation: **Reward function**
  - Solution: An **optimal policy**

  - **Definition:**
    An optimal policy π^* **maximizes expected utility for all states**:
    For all states s and alternative policies π,
    $$E(\pi^*, s) \geq E(\pi, s)$$

Given a policy, we can compute expected utility.
How do we find a policy maximizing it?

Simplification

jonkv@ida

- In many formulations of MDPs (and our robotic example), rewards **do not depend on the outcome s'**!

Let's simplify the upcoming examples a bit…

$$E(\pi, s) = \sum_{s' \in S} P(s, \pi(s), s') \cdot \left(R(s, \pi(s), s') + \gamma E(\pi, s') \right)$$

$$\rightarrow E(\pi, s) = \sum_{s' \in S} P(s, \pi(s), s') \cdot \left(R(s, \pi(s)) + \gamma E(\pi, s') \right)$$

$$\rightarrow E(\pi, s) = R(s, \pi(s)) + \sum_{s' \in S} P(s, \pi(s), s') \cdot \gamma E(\pi, s')$$

Solving an MDP, step 1: Local modifications

Properties of Local Changes

jonkv@ida

- Given an MDP, suppose that:
  - You already have some arbitrary (even random) policy π
  - You select an arbitrary state s_k, make a **local change** to $\pi(s_k)$
    - Example: $\pi(s_k) = move(l1, l3)$ ➔ $\pi(s_k) = move(l1, l4)$
  - The change turns out to be a **local improvement**
    - It **increases** the expected utility $\mathrm{E}(\pi, s_k)$ for **this** particular state s_k

- Then the change **cannot decrease** $\mathrm{E}(\pi, s')$ for **any** s'!

A local improvement for one state is always a global improvement

Properties of Local Changes (2)

jonkv@ida

- Why?

We change $\pi(s_k)$ -- select another action…

$$E(\pi, s_k) = R\big(s, \pi(s_k)\big) + \sum_{s' \in S} P(s, \pi(s_k), s') \cdot \gamma E(\pi, s')$$

So expected utility $E(\pi, s_k)$ **increases**

How does this affect $E(\pi, s_m)$ for another state?

$$E(\pi, s_m) = R\big(s, \pi(s_m)\big) + \sum_{s' \in S} P(s, \pi(s_m), s') \cdot \gamma E(\pi, s')$$

All of these remain unchanged!

$E(\pi, s_k)$ may occur **here** ($s' = s_k$),
but only ***positively***:
Increase $E(\pi, s_k)$ ➔
may increase $E(\pi, s_m)$

Properties of Local Changes (3)

jonkv@ida

- **How many times** can you make local improvements?
  - There is a **finite number** of possible **policies**, $|S| \cdot |A|$
    - The number of states $|S|$ is finite
    - The number of actions $|A|$ is finite
  - You **can't** make the **same improvement** twice
    - Every improvement leads to strictly greater expected utility
    - Can't "loop around"
  - ➔ After a **finite number** of local improvements, no more local improvement will be possible

Properties of Local Changes (4)

jonkv@ida

- Also:
  - Every global improvement **can be reached** through such local improvements (no need to first make the policy worse, then better)

- ➔ We can **find optimal solutions** through **local** improvements
  - No need to "think **globally**"

But how do we find a local improvement?

Remember, finding expected utilities required solving an expensive equation system…

Is a Local Change an Improvement?

jonkv@ida

- To find out **if** a change is an improvement:
  - Take the current policy π, with an expected utility:

$$E(\pi, s) = R\big(s, \pi(s)\big) + \sum_{s' \in S} P(s, \pi(s), s') \cdot \gamma E(\pi, s')$$

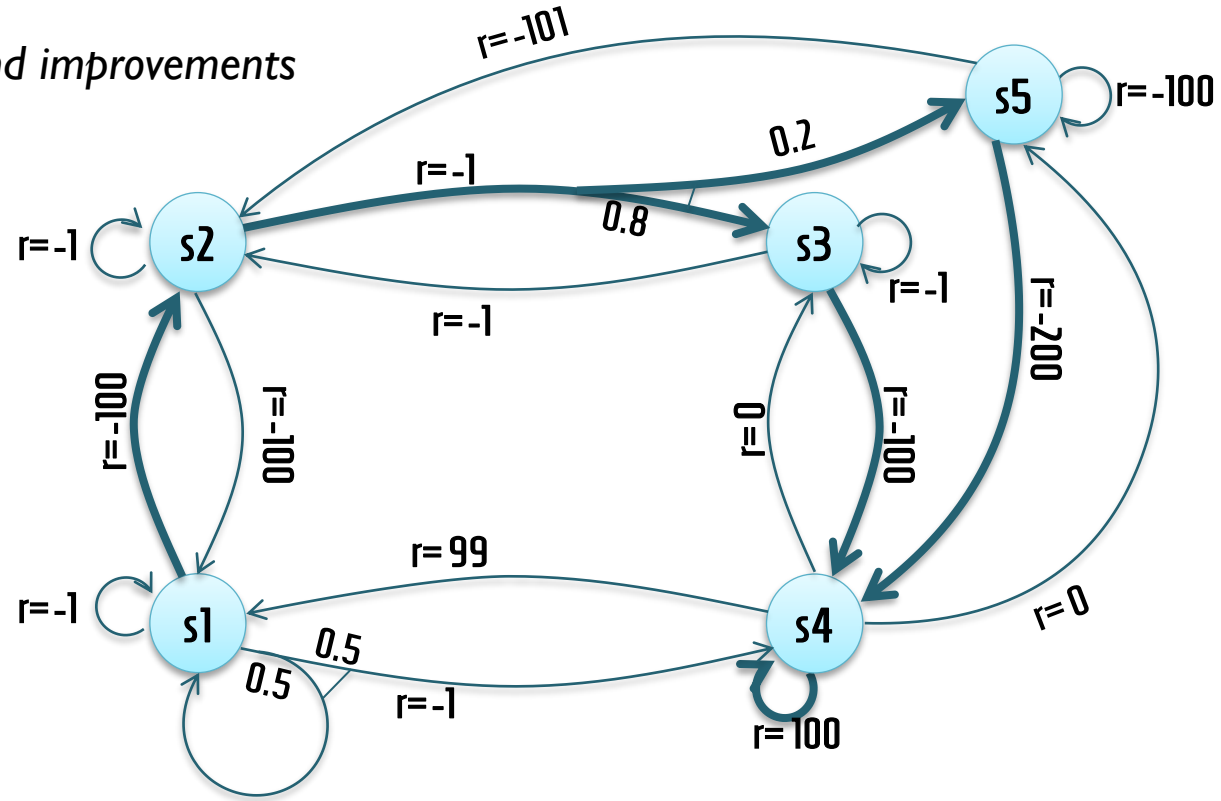
  - Define the Q function:
    - What is the expected utility if you **first execute action** a, regardless of what the policy says, but then continue executing the **old policy** for all other steps?
    - $Q(\pi, s, a) = R(s, a) + \sum_{s' \in S} P(s, a, s') \cdot \gamma E(\pi, s')$

If $Q(\pi, s, a) > E(\pi, s)$, then setting $\pi(s) = a$ would be an *improvement* to π.
We know this much without solving a full equation system…
Just not how large the improvement is!

Preliminaries 2: Example

jonkv@ida

- Example: $E(\pi, s1)$
  - The expected utility of following π
  - Starting in s1, beginning with move(l1,l2)
- $Q(\pi, s1, \text{move}(l1, l4))$
  - The expected utility of being in s1, first executing move(l1, l4), *then* following policy π
  - Only used to *quickly find improvements*



One Solution Method for MDPs: Policy Iteration

Policy Iteration

jonkv@ida

- General idea:
  - Start out with an **initial policy**, maybe randomly chosen
  - Calculate *and store* the **expected utility** of executing that policy for each state
  - **Update** the policy by making a **local** decision **for each state**: "Which action should my **improved** policy choose in this state?"
    - Use the actions that ***appear to be best*** according to the Q function, based on the actual expected utility for the **current** policy
    - For every state s:

$$\pi'(s) := \arg\max_{a \in A} Q(\pi, s, a)$$

  - **Iterate** until the policy no longer changes

But what if there was an **even better** choice,
which we don't see now because of our single step modification (Q)?

That's OK: We still have an *improvement*,
which cannot prevent *future* improvements in the next iteration

First Iteration

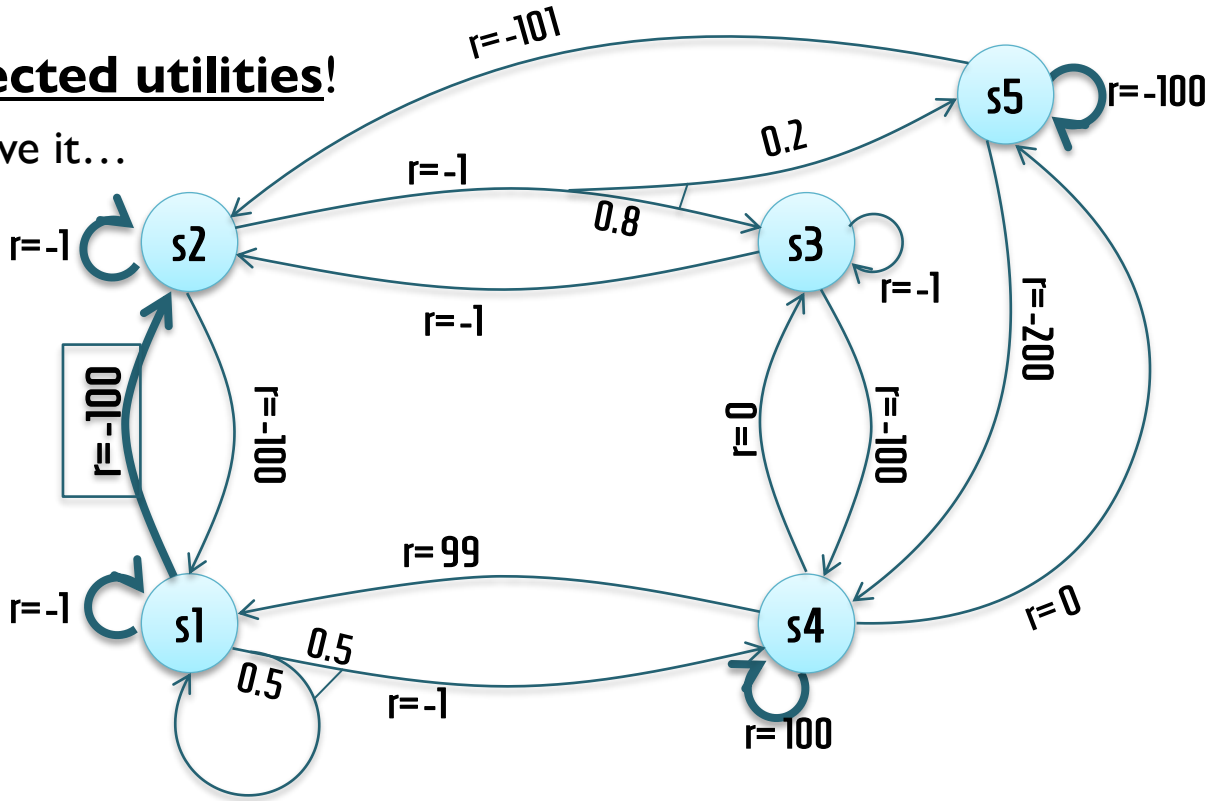
Policy Iteration 1: Initial Policy π_1

jonkv@ida

- Policy iteration requires an **initial policy**
  - Let's start by choosing "wait" in every state
  - Let's set a discount factor: $\gamma = 0.9$
    - Easy to use in calculations
      In reality we might use a larger factor
      (we're not **that** short-sighted!)

$$\pi_1 = \{(s1, \text{wait}), (s2, \text{wait}), (s3, \text{wait}), (s4, \text{wait}), (s5, \text{wait})\}$$

  - Need to know its **expected utilities**!
    - To know how to improve it…



Policy Iteration 2: Expected Utility for π_1

jonkv@ida

- Calculate expected utilities for the **current** policy π_1
  - Simple: Chosen transitions are deterministic ***and*** return to the same state!
    - $E(\pi,s) =$ $R(s, \pi(s))$ $+ \gamma$ $\sum_{s' \in S} P(s, \pi(s), s') E(\pi,s')$
    - $E(\pi1,s1) =$ $R(s1, wait)$ $+ \gamma$ $E(\pi1,s1)$ $= -1$ $+ 0.9$ $E(\pi1,s1)$
    - $E(\pi1,s2) =$ $R(s2, wait)$ $+ \gamma$ $E(\pi1,s2)$ $= -1$ $+ 0.9$ $E(\pi1,s2)$
    - $E(\pi1,s3) =$ $R(s3, wait)$ $+ \gamma$ $E(\pi1,s3)$ $= -1$ $+ 0.9$ $E(\pi1,s3)$
    - $E(\pi1,s4) =$ $R(s4, wait)$ $+ \gamma$ $E(\pi1,s4)$ $= +100$ $+ 0.9$ $E(\pi1,s4)$
    - $E(\pi1,s5) =$ $R(s5, wait)$ $+ \gamma$ $E(\pi1,s5)$ $= -100$ $+ 0.9$ $E(\pi1,s5)$
  - Simple equations to solve:
    - $0.1E(\pi1,s1) = -1$ ➔ $E(\pi1,s1) = -10$
    - $0.1E(\pi1,s2) = -1$ ➔ $E(\pi1,s2) = -10$
    - $0.1E(\pi1,s3) = -1$ ➔ $E(\pi1,s3) = -10$
    - $0.1E(\pi1,s4) = +100$ ➔ $E(\pi1,s4) = +1000$
    - $0.1E(\pi1,s5) = -100$ ➔ $E(\pi1,s5) = -1000$

Given this policy π_1:

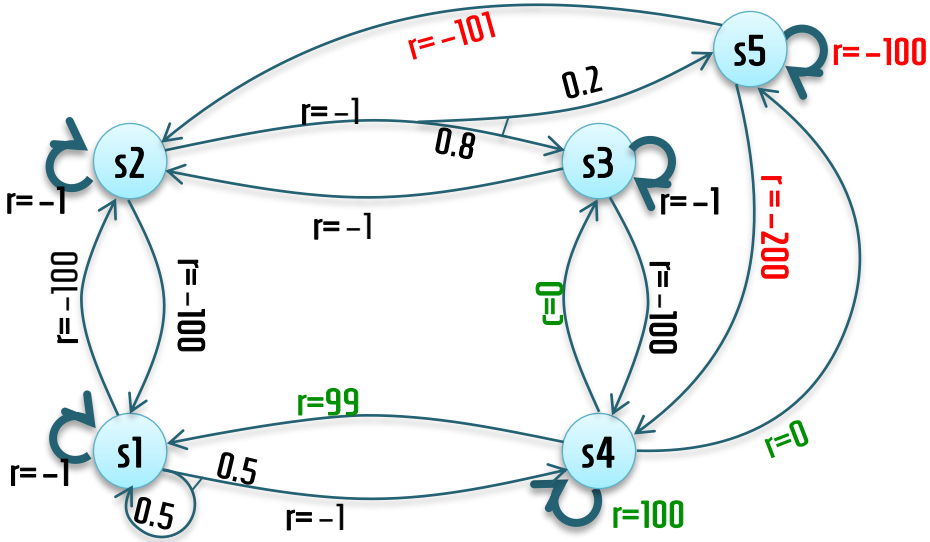
High rewards if we start in s4,
high costs if we start in s5

Policy Iteration 3: Update 1a

jonkv@ida

What is the best **local** modification according to the **expected utilities** of the **current** policy?

$E(\pi_1, s1) = -10$
$E(\pi_1, s2) = -10$
$E(\pi_1, s3) = -10$
$E(\pi_1, s4) = +1000$
$E(\pi_1, s5) = -1000$



- For every state s:
  - Let $\pi_2(s) = \text{argmax}_{a \in A}\, Q(\pi_1, s, a)$
  - That is, find the action a that maximizes $R(s, a) + \gamma \sum_{s' \in S} P(s, a, s')\ E(\pi 1, s')$
    - s1: wait
      move(l1,l2)
      move(l1,l4)

      Best improvement

−1	+ 0.9 * − 10	= − 10
− 100	+ 0.9 * − 10	= − 109
−1	+ 0.9 * (0.5* − 10 + 0.5*1000)	= +444.5

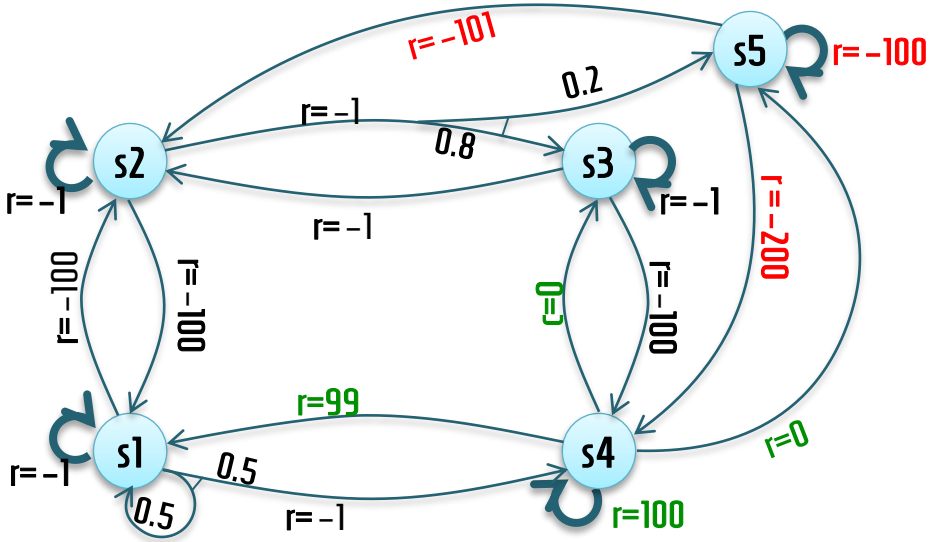
  - These are not the **true** expected utilities for starting in state $s1$!
    - But the values will yield good guidance to find **policy improvements**

Policy Iteration 4: Update 1b

jonkv@ida

What is the best **local** modification according to the **expected utilities** of the **current** policy?

$E(\pi_1, s1) = -10$
$E(\pi_1, s2) = -10$
$E(\pi_1, s3) = -10$
$E(\pi_1, s4) = +1000$
$E(\pi_1, s5) = -1000$



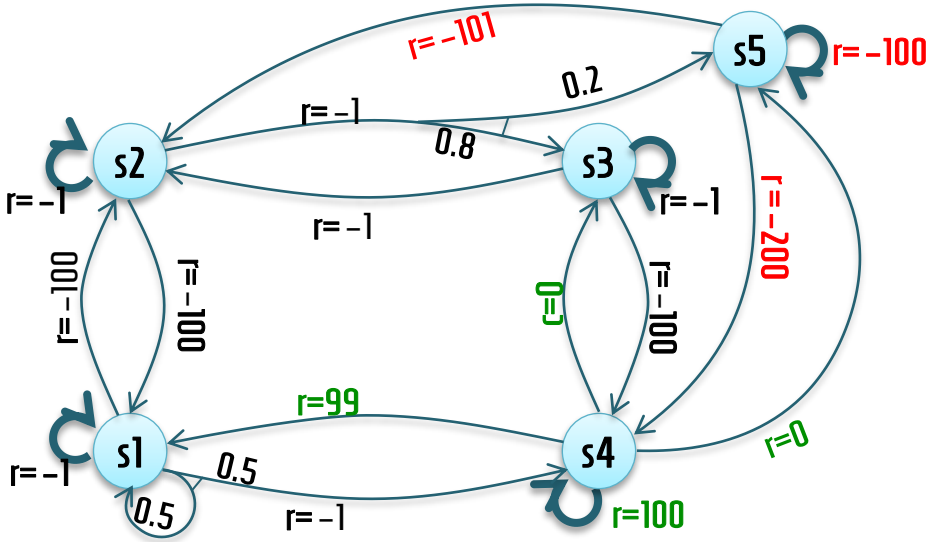
- For every state s:
  - Let $\pi_2(s) = \text{argmax}_{a \in A}\ Q(\pi 1, s, a)$
  - That is, find the action *a* that maximizes $R(s, a) + \gamma \sum_{s' \in S} P(s, a, s')\ E(\pi 1, s')$
    - s2: wait: $-1 + 0.9 * -10 = -10$
      move(l2,l1): $-100 + 0.9 * -10 = -109$
      move(l2,l3): $-1 + 0.9 * (0.8*-10 + 0.2*-1000) = -188{,}2$

Policy Iteration 5: Update 1c

jonkv@ida

What is the best **local** modification according to the **expected utilities** of the **current** policy?

$E(\pi_1, s1) = -10$
$E(\pi_1, s2) = -10$
$E(\pi_1, s3) = -10$
$E(\pi_1, s4) = +1000$
$E(\pi_1, s5) = -1000$



- For every state s:
  - Let $\pi_2(s) = \text{argmax}_{a \in A}\ Q(\pi 1, s, a)$
  - That is, find the action *a* that maximizes $R(s, a) + \gamma \sum_{s' \in S} P(s, a, s')\ E(\pi 1, s')$
    - s3: wait
      move(l3,l2)
      move(l3,l4)
    - s4: wait
      move(l4,l1)
      …
    - s5: wait
      move(l5,l2)
      move(l5,l4)

−1	+ 0.9 *	−10	= −10
−1	+ 0.9 *	−10	= −10
−100	+ 0.9 *	+1000	= +800
+100	+ 0.9 *	+1000	= +1000
+99	+ 0.9 *	−10	= +90
−100	+ 0.9 *	−1000	= −1000
−101	+ 0.9 *	−10	= −110
−200	+ 0.9 *	+1000	= +700

Policy Iteration 6: Second Policy

jonkv@ida

- This results in a **new policy**

π_1 = {(s1, wait), (s2, wait), (s3, wait), (s4, wait), (s5, wait)}	E(π1,s1) = –10 E(π1,s2) = –10 E(π1,s3) = –10 E(π1,s4) = +1000 E(π1,s5) = –1000

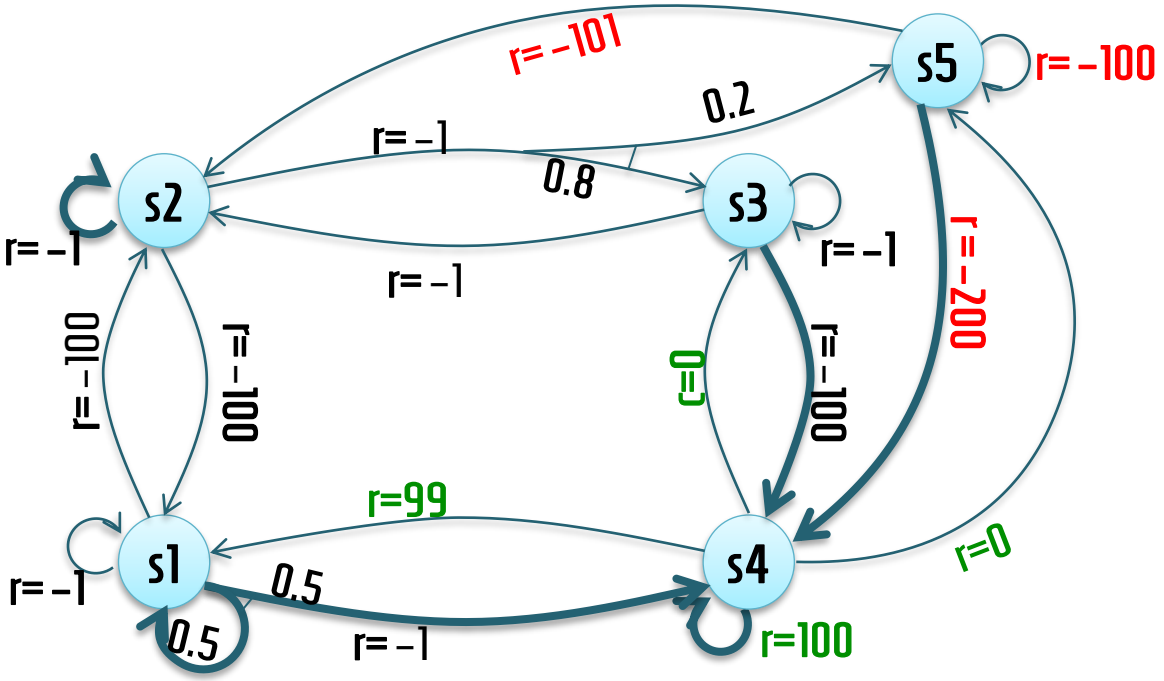
π_2 = { (s1, **move(11,14)**), (s2, wait), (s3, **move(13,14)**), (s4, wait), (s5, **move(15,14)**)}	>= +444,5 >= –10 >= +800 >= +1000 >= +700

Q-values based on *one* modified action, then following π_1 (can't decrease!)

Now we have made use of earlier indications that s4 seems to be a good state

➔ Try to go there from s1 / s3 / s5!

No change in s2 yet…



Second Iteration

Policy Iteration 7: Expected Utilities for π_2

- Calculate **true** expected utilities for the **new** policy π_2
  - $E(\pi2,s1) = R(s1, move(l1,l4)) + \gamma \ldots = -1 + 0.9\,(0.5E(\pi2,s1) + 0.5E(\pi2,s4))$
  - $E(\pi2,s2) = R(s2, wait) + \gamma E(\pi2,s2) = -1 + 0.9\,E(\pi2,s2)$
  - $E(\pi2,s3) = R(s3, move(l3,l4)) + \gamma E(\pi2,s4) = -100 + 0.9\,E(\pi2,s4)$
  - $E(\pi2,s4) = R(s4, wait) + \gamma E(\pi2,s4) = +100 + 0.9\,E(\pi2,s4)$
  - $E(\pi2,s5) = R(s5, move(l5,l4)) + \gamma E(\pi2,s4) = -200 + 0.9\,E(\pi2,s4)$
  - Equations to solve:
    - $0.1E(\pi2,s2) = -1$ ➔ **$E(\pi2,s2) = -10$**
    - $0.1E(\pi2,s4) = +100$ ➔ **$E(\pi2,s4) = +1000$**
    - $E(\pi2,s3) = -100 + 0.9E(\pi2,s4) = -100 + 0.9*1000 = +800$ ➔ **$E(\pi2,s3) = +800$**
    - $E(\pi2,s5) = -200 + 0.9E(\pi2,s4) = -200 + 0.9*1000 = +700$ ➔ **$E(\pi2,s5) = +700$**
    - $E(\pi2,s1) = -1 + 0.45 * E(\pi2,s1) + 0.45 * E(\pi2,s4)$ ➔ ➔ **$E(\pi2,s1) = +816,36$**

      $0.55\,E(\pi2,s1) = -1 + 0.45 * E(\pi2,s4)$ ➔

      $0.55\,E(\pi2,s1) = -1 + 450$ ➔

      $0.55\,E(\pi2,s1) = +449$ ➔

      $E(\pi2,s1) = +816,3636\ldots$

$\pi_2 = \{(s1, move(l1,l4)),$
$(s2, wait),$
$(s3, move(l3,l4)),$
$(s4, wait),$
$(s5, move(l5,l4))\}$

Policy Iteration 8: Second Policy

jonkv@ida

- Now we have the **true** expected utilities of the second policy…

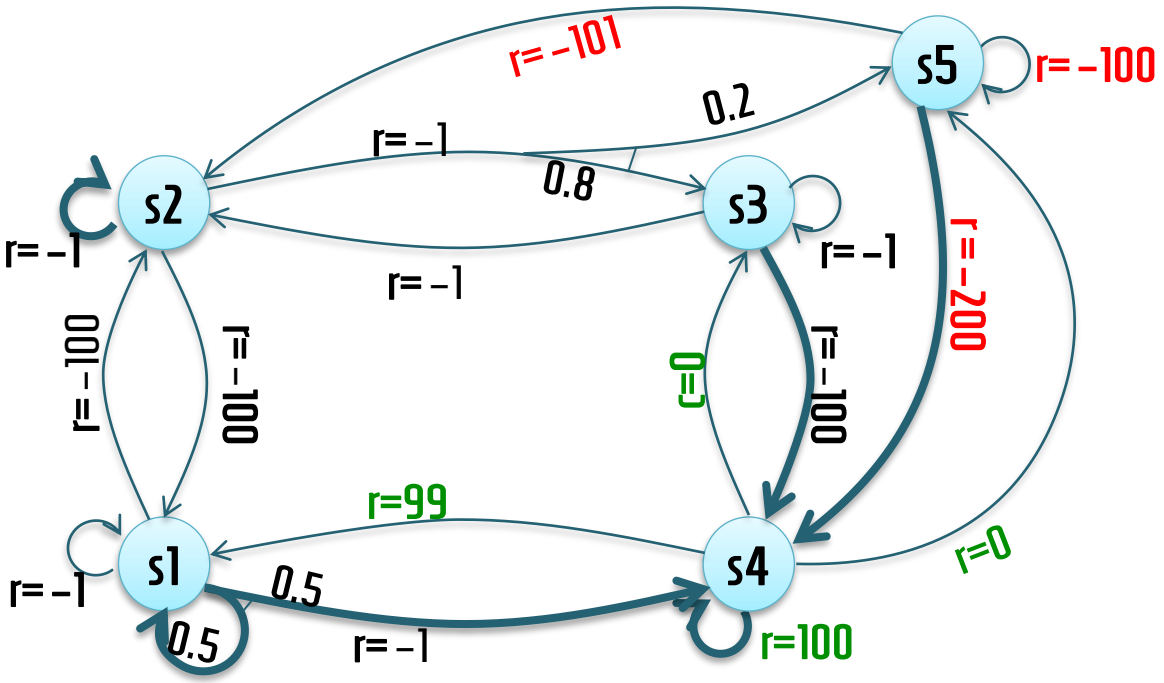
π_1 = {(s1, wait), (s2, wait), (s3, wait), (s4, wait), (s5, wait)}	E(π1,s1) = –10 E(π1,s2) = –10 E(π1,s3) = –10 E(π1,s4) = +1000 E(π1,s5) = –1000

π_2 = { (s1, move(l1,l4), (s2, wait), (s3, move(l3,l4)), (s4, wait), (s5, move(l5,l4))}	>= +444,5 >= –10 >= +800 >= +1000 >= +700	E(π2,s1) = +816,36 E(π2,s2) = – 10 E(π2,s3) = +800 E(π2,s4) = +1000 E(π2,s5) = +700

S5 wasn't so bad after all, since you can reach s4 in a single step!

S1 / s3 are even better.

S2 *seems* much worse in comparison, since the benefits of s4 haven't "propagated" that far.

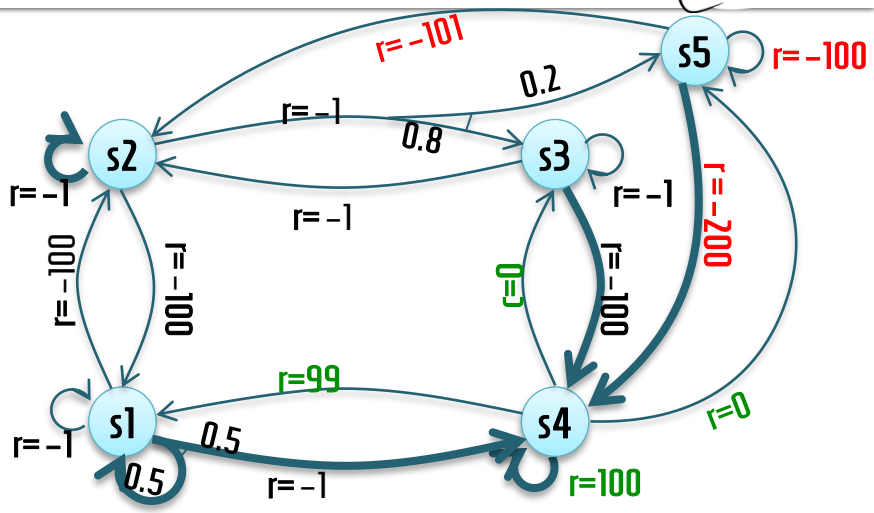


Policy Iteration 9: Update 2a

jonkv@ida

What is the best **local** modification according to the **expected utilities** of the **current** policy?

$E(\pi 2,s1) = +816{,}36$
$E(\pi 2,s2) = -10$
$E(\pi 2,s3) = +800$
$E(\pi 2,s4) = +1000$
$E(\pi 2,s5) = +700$



- For every state s:
  - Let $\pi_3(s) = \text{argmax}_{a \in A}\, Q(\pi_2,s,a)$
  - That is, find the action *a* that maximizes $R(s, a) + \gamma \sum_{s' \in S} P(s, a, s')\ E(\pi_2,s')$
    - s1: wait: $-1 + 0.9 * 816{,}36 = +733{,}72$
      move(l1,l2): $-100 + 0.9 * -10 = -109$
      move(l1,l4): $-1 + 0.9 * (.5*1000+.5*816.36) = +816{,}36$
      Seems best – chosen!
    - s2: wait: $-1 + 0.9 * -10 = -10$
      move(l2,l1): $-100 + 0.9 * 816{,}36 = +634{,}72$
      move(l2,l3): $-1 + 0.9 * (0.8*800 + 0.2*700) = +701$

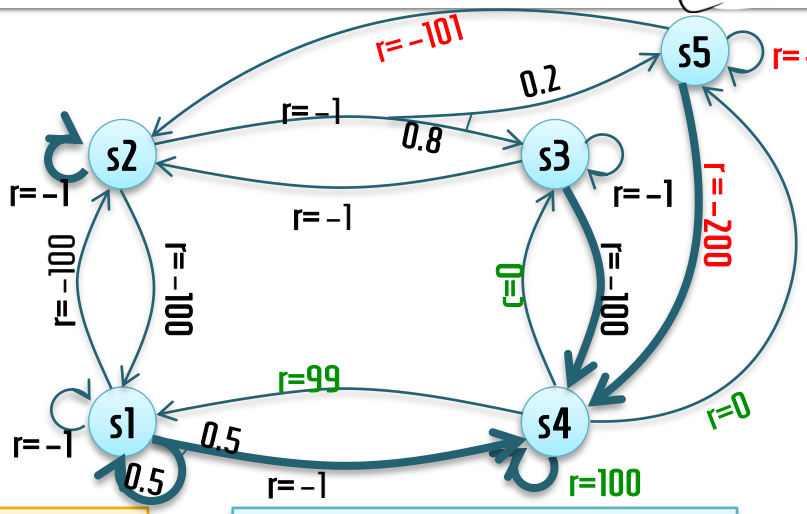
Now we will change the action taken at s2,
since the expected utilities for possible "next" states s1, s3, s5… have increased

Policy Iteration 10: Update 2b

jonkv@ida

What is the best **local** modification according to the **expected utilities** of the **current** policy?

$E(\pi 2,s1) = +816{,}36$
$E(\pi 2,s2) = -10$
$E(\pi 2,s3) = +800$
$E(\pi 2,s4) = +1000$
$E(\pi 2,s5) = +700$



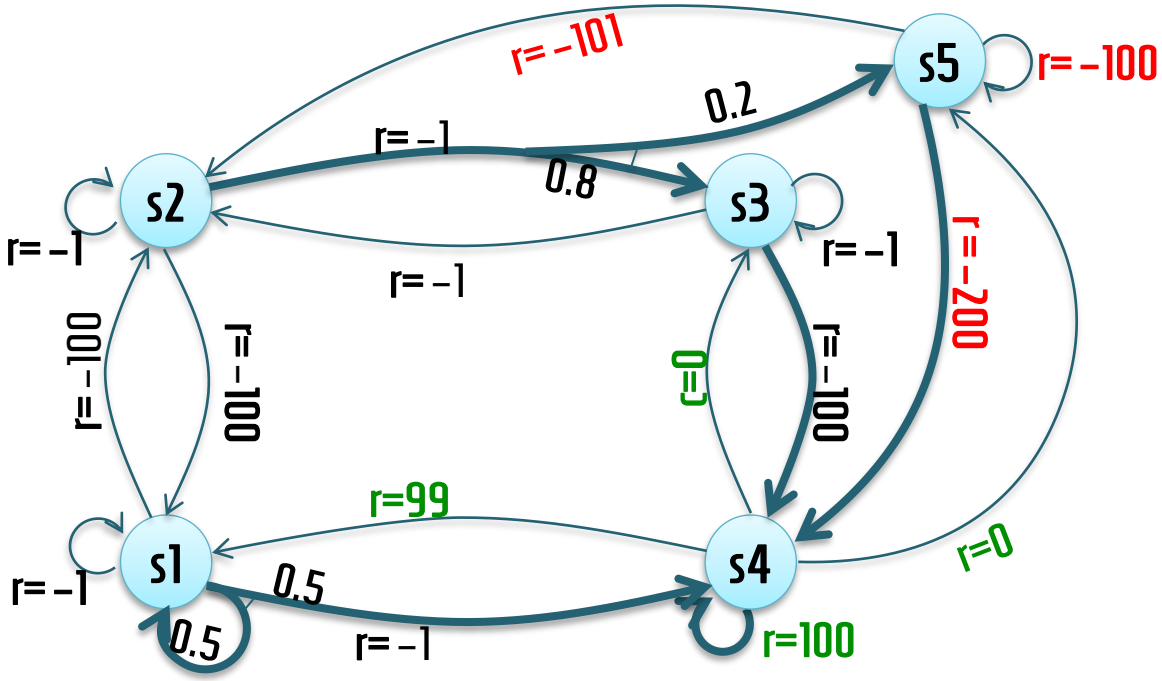
- For every state s:
  - Let $\pi_3(s) = \text{argmax}_{a \in A}\, Q(\pi_2,s,a)$
  - That is, find the action *a* that maximizes $R(s, a) + \gamma \sum_{s' \in S} P(s, a, s')\ E(\pi_2,s')$
    - s3: wait $-1 + 0.9 * 800 = +719$
      move(l3,l2) $-1 + 0.9 * -10 = -10$
      move(l3,l4) $-100 + 0.9 * 1000 = +800$
    - s4: wait $+100 + 0.9 * 1000 = +1000$
      move(l4,l1) $+99 + 0.9 * 816{,}36 = +833{,}72$
      …
    - s5: wait $-100 + 0.9 * 700 = +530$
      move(l5,l2) $-101 + 0.9 * -10 = -110$
      move(l5,l4) $-200 + 0.9 * -1000 = +700$

Policy Iteration 11: Third Policy

jonkv@ida

- This results in a **new policy** π_3
  - **True expected utilities** are updated by solving an equation system
  - The algorithm will iterate once more
  - No changes will be made to the policy
  - ➔ Termination with optimal policy!

π_3 = {(s1, move(l1,l4),
(s2, **move(l2,l3)**),
(s3, move(l3,l4)),
(s4, wait),
(s5, move(l5,l4))}



Policy Iteration Algorithm

Policy Iteration Algorithm

jonkv@ida

- **Policy iteration** is a way to find an optimal policy π^*
  - Start with an **arbitrary** initial policy π_1. Then, for $i = 1, 2, \ldots$
    - Compute expected utilities $E(\pi_i, s)$ for all s by **solving a system of equations**

Find utilities according to current policy

- System: For all s, $E(\pi_i, s) = Q(\pi_i, s, \pi_i(s))$
  $= R(s, \pi_i(s)) + \gamma \sum_{s' \in S} P(s, \pi_i(s), s') E(\pi_i, s')$
- Result: The **expected utilities** of the "current" policy in **every** state s

    - Not a simple recursive calculation – the state graph is generally cyclic!
    - Compute an improved policy π_{i+1} "locally" for every s

Find best local improvements

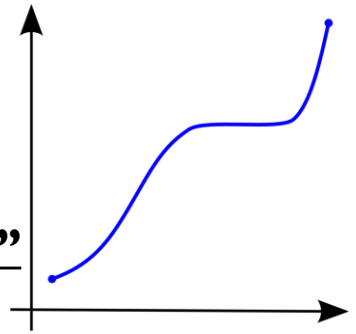
- $\pi_{i+1}(s) := \text{argmax}_{a \in A} Q(\pi_i, s, a)$
  $= \text{argmax}_{a \in A} R(s, a) + \gamma \sum_{s' \in S} P(s, a, s') E(\pi_i, s')$
- **Best action** in **any** given state s given expected utilities of **old** policy π_i

    - If $\pi_{i+1} = \pi_i$ then exit
      - No local improvement possible, so the solution is optimal
    - Otherwise
      - This is a new policy π_{i+1} – with **new** expected utilities!
      - Iterate, calculate **those** utilities, …

Policy Iteration Convergence

jonkv@ida

- **Converges** to a final answer in a finite number of iterations!
  1. Finite states, finite actions ➔ finite number of candidate policies
  2. An iteration can never **return** to a previous policy
     - We change which action to execute in state *s* only if this **improves expected (pseudo-)utility Q** for *s*
     - This can **never decrease** the utility for other states!
     - So utilities are **monotonically strictly improving "all over"** ➔ no circularity possible



- Actually: **Polynomial** number of iterations!
  - But polynomial in the **number of states** (huge) not the number of objects/actions
  - May take **many** iterations, and each iteration can be slow (solving equation system)

Alternatives

jonkv@ida

- Methods exist for **reducing the search space**,
  and for **approximating** optimal solutions (see the book)
  - Value iteration
  - Linear programming
  - Real Time Dynamic Programming
  - …

Conclusions

Example Questions

jonkv@ida

- **Example exam topics:**
  - **PDB heuristics**: The main ideas of patterns, how this results in a modified planning problem, why this is faster to *solv*e, how the results are used, …
    - Given a planning problem, can you apply a pattern and find the relaxed problem?
  - **Landmarks**: The main ideas, what a landmark *is*, how to find landmarks, how to use them in a heuristic function, …
    - Given a planning problem, can you find *n* unachieved fact landmarks using the means-ends analysis **algorithm?**
  - The **concepts** of histories, utility, discount factors, …
  - What a ***policy*** is / how it is defined, why we use it in some types of planning, and why a classical plan is not sufficient in these cases
    - Explain **policy iteration**, and apply 1-2 steps given a small problem instance

TDDD48 Automated Planning (1)

jonkv@ida

- **Deeper discussions** about all of these topics, and…
  - **Formal basis** for planning
    - Alternative representations of planning problems
    - Simple and complex state transition systems
  - Different **principles** for **heuristics**
  - Alternative **search spaces**
    - Partial order planning, …
  - Extended **expressivity**
    - Planning with non-classical goals
  - Planning with **domain knowledge**
    - Using what you know: Temporal control rules
    - Breaking down a task into smaller parts: Hierarchical Task Networks
  - **Combining planners** – portfolio planning, learning planning parameters, …
  - Alternative **types** of planning
    - Path planning
  - And so on…