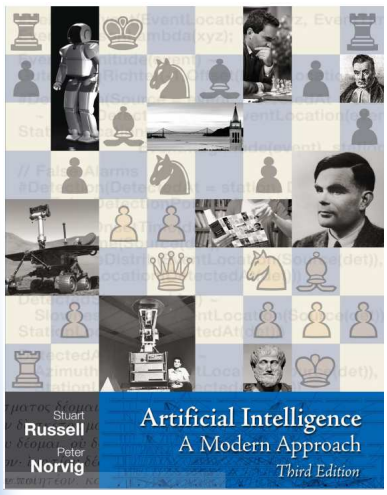
Intelligent Autonomous Agents and Cognitive Robotics

Topic 8: Decision-Making under Uncertainty Complex Decisions

Ralf Möller, Rainer Marrone
Hamburg University of Technology

Literature



- Chapter 17

Material from Lise Getoor, Jean-Claude Latombe, Daphne Koller, and Stuart Russell

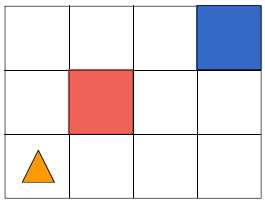
2

Sequential Decision Making

- Finite Horizon
  - Fixed time N after that nothing happens
- Infinite Horizon
  - N not fixed

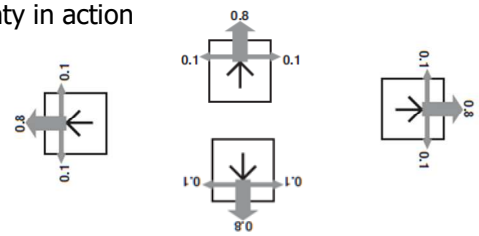
3

Simple Robot Navigation Problem

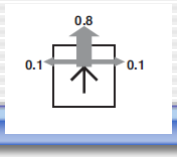


- In each state, the possible actions are U, D, R, L

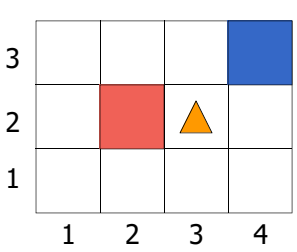
Uncertainty in action



4



Sequence of Actions

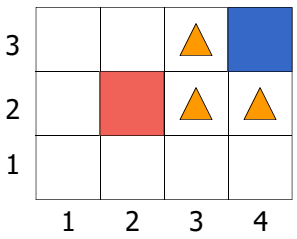


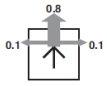
[3,2]

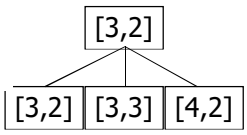
- Planned sequence of actions: (U, R)

5

Sequence of Actions



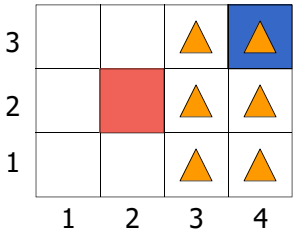


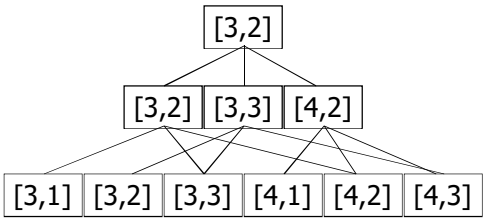


- Planned sequence of actions: (U, R)
- U is executed

6

Histories





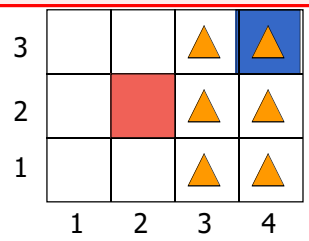
- Planned sequence of actions: (U, R)
- U has been executed
- R is executed

- There are 9 possible sequences of states – called histories – and 6 possible final states for the robot!

7

Probability of Reaching the Goal

Note importance of Markov property in this derivation



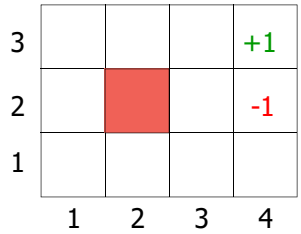
P([4,3] | (U,R).[3,2]) = **P**([3,3] | U.[3,2]) x **P**([4,3] | R.[3,3])
+ **P**([4,2] | U.[3,2]) x **P**([4,3] | R.[4,2])

= **0.8** x **0.8** + **0.1** x **0.1**

= 0.65

8

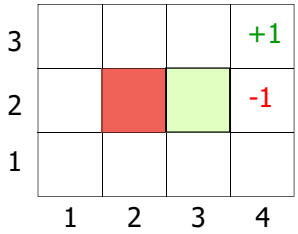
Utility of a History

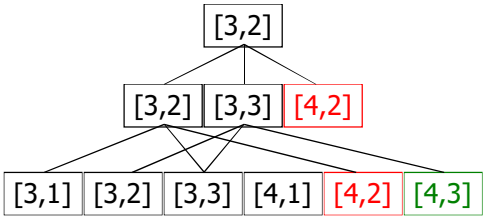


- [4,3] provides power supply
- [4,2] is a sand area from which the robot cannot escape
- The robot needs to recharge its batteries
- [4,3] or [4,2] are terminal states
- The utility of a history is defined by the utility of the last state (+1 or –1) minus n/25, where n is the number of moves

9

Utility of an Action Sequence



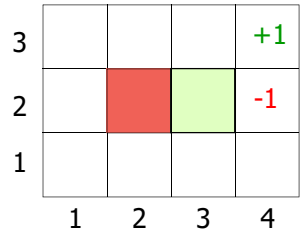


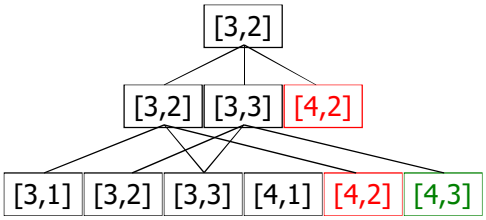
- Consider the action sequence (U,R) from [3,2]
- A run produces one among 7 possible histories, each with some probability
- The utility of the sequence is the expected utility of the histories:

$$\mathcal{U} = \Sigma_h \mathcal{U}_h \mathbf{P}(h)$$

10

Optimal Action Sequence

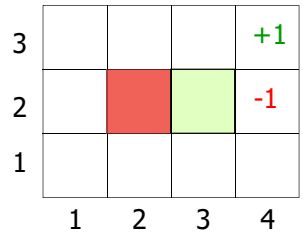


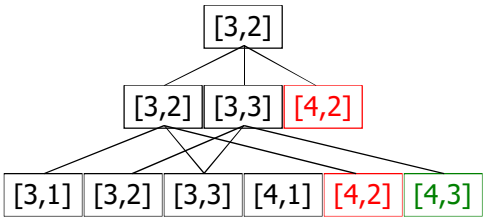


- Consider the action sequence (U,R) from [3,2]
- A run produces one among 7 possible histories, each with some probability
- The utility of the sequence is the expected utility of the histories
- The optimal sequence is the one with maximal utility

11

Optimal Action Sequence

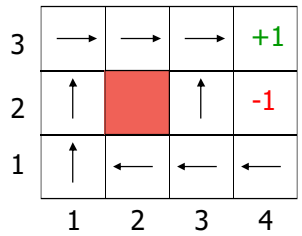




- Consider the action sequence (U,R) from [3,2]
- A run prod only if the sequence is executed blindly! me probability
- The utility of the sequence is the expected utility of the histories
- The optimal sequence is the one with maximal utility
- **But is the optimal action sequence what we want to compute?**

12

Policy (Reactive/Closed-Loop Strategy)



- A policy Π is a complete mapping from states to actions

13

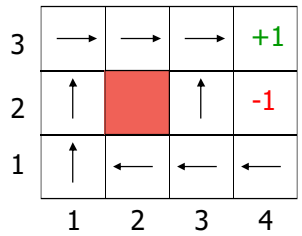
Reactive Agent Algorithm

Repeat:

- s ← sensed state
- If s is terminal then exit
- a ← Π(s)
- Perform a

14

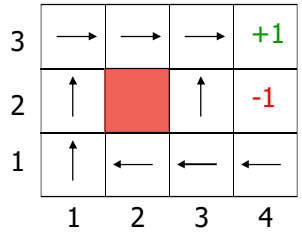
Optimal Policy



- A policy Π is a complete mapping from states to actions
- The optimal policy Π^* is the one that always yields a history (ending at a terminal state) with maximal expected utility

15

Optimal Policy



- A policy Π is a complete mapping from states to actions
- The optimal policy Π^* is the one that always yields a history with maximal expected utility

How to compute Π^*?

This problem is called a Markov Decision Problem (MDP)

16

Additive Utility: Stationarity

- History H = $(s_0,s_1,\ldots,s_n)$
- The utility of H is additive iff:

$$\mathcal{U}(s_0,s_1,\ldots,s_n) = \mathcal{R}(0) + \mathcal{U}(s_1,\ldots,s_n) = \Sigma\, \mathcal{R}(i)$$

Reward

17

Additive Utility

- History H = $(s_0,s_1,\ldots,s_n)$
- The utility of H is additive iff:

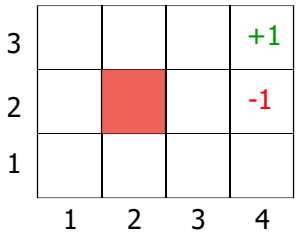
$$\mathcal{U}(s_0,s_1,\ldots,s_n) = \mathcal{R}(0) + \mathcal{U}(s_1,\ldots,s_n) = \Sigma\, \mathcal{R}(i)$$

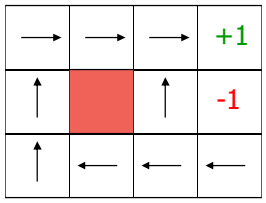
- Robot navigation example:
  - $\mathcal{R}(n) = +1$ if $s_n = [4,3]$
  - $\mathcal{R}(n) = -1$ if $s_n = [4,2]$
  - $\mathcal{R}(i) = -1/25$ if $i = 0, \ldots, n-1$

18

Principle of Max Expected Utility

- History H = $(s_0,s_1,\ldots,s_n)$
- Utility of H: $\mathcal{U}(s_0,s_1,\ldots,s_n) = \sum \mathcal{R}(i)$





First-step analysis →

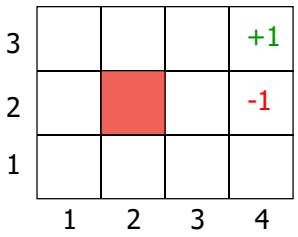
- $\mathcal{U}(i) = \mathcal{R}(i) + \max_a \sum_j \mathbf{P}(j \mid a.i)\, \mathcal{U}(j)$
- $\Pi^*(i) = \arg\max_a \sum_j \mathbf{P}(k \mid a.i)\, \mathcal{U}(j)$

19

Value Iteration

- Initialize the utility of each non-terminal state s_i to $\mathcal{U}_0(i) = 0$
- For t = 0, 1, 2, …, do:

$$\mathcal{U}_{t+1}(i) \leftarrow \mathcal{R}(i) + \max_a \sum_k \mathbf{P}(k \mid a.i)\, \mathcal{U}_t(k) \quad \text{(Bellmann equation)}$$



20

Initialization

3	0	0	0	+1
2	0		0	-1
1	0	0	0	0
	1	2	3	4

Iteration 1

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

$$U^{(1)}(1,1) = -0.04 + 1 * \max \begin{bmatrix} 0.8U^{(0)}(1,2) + 0.1U^{(0)}(2,1) + 0.1U^{(0)}(1,1) & UP \\ 0.9U^{(0)}(1,1) + 0.1U^{(0)}(1,2) & LEFT \\ 0.9U^{(0)}(1,1) + 0.1U^{(0)}(2,1) & DOWN \\ 0.8U^{(0)}(2,1) + 0.1U^{(0)}(1,2) + 0.1U^{(0)}(1,1) & RIGHT \end{bmatrix}$$

$$U^{(1)}(1,1) = -0.04 + \max \begin{bmatrix} 0 & UP \\ 0 & LEFT \\ 0 & DOWN \\ 0 & RIGHT \end{bmatrix}$$

21

Initialization

3	0	0	0	+1
2	0		0	-1
1	0	0	0	0
	1	2	3	4

Iteration 1

3	0	0	0.76	+1
2	0		0	-1
1	-0.04	0	0	0
	1	2	3	4

$$U^{(1)}(3,3) = -0.04 + 1 * \max \begin{bmatrix} 0.8U^{(0)}(3,3) + 0.1U^{(0)}(2,3) + 0.1U^{(0)}(4,3) & UP \\ 0.8U^{(0)}(2,3) + 0.1U^{(0)}(3,3) + 0.1U^{(0)}(3,2) & LEFT \\ 0.8U^{(0)}(3,2) + 0.1U^{(0)}(2,3) + 0.1U^{(0)}(4,3) & DOWN \\ 0.8U^{(0)}(4,3) + 0.1U^{(0)}(3,3) + 0.1U^{(0)}(3,2) & RIGHT \end{bmatrix}$$

$$U^{(1)}(3,3) = -0.04 + \max \begin{bmatrix} 0.1 & UP \\ 0 & LEFT \\ 0.1 & DOWN \\ 0.8 & RIGHT \end{bmatrix}$$

22

After a Full Iteration

- Only the state one step away from a positive reward (3,3) has gained value, all the others are losing value because of the cost of moving

Iteration 1

3	-.04	-.04	0.76	+1
2	-.04		-.04	-1
1	-.04	-.04	-.04	-.04
	1	2	3	4

24

Value Iteration: from state utilities to *policy*

- Now the agent can chose the action that implements the MEU principle: maximize the expected utility of the subsequent state

$$\pi^*(s) = \arg\max_a \sum_{s'} P(s' \mid s,a) U(s')$$

expected value of following policy π* in s'

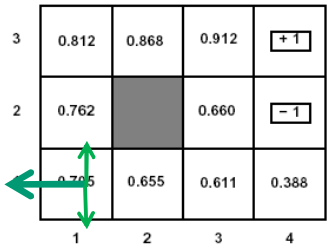
states reachable from s by doing a

Probability of getting to *s'* from *s* via *a*

25

Example

$$\pi^*(s) = \arg\max_a \sum_{s'} P(s'|s,a)U(s')$$



- To find the best action in (1,1)

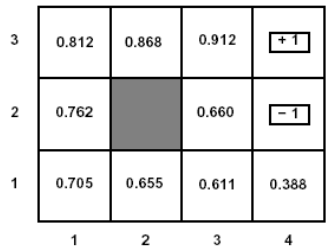
$$\pi^*(1,1) = \arg\max \begin{bmatrix} 0{,}7456 & UP \\ 0{,}7107 & LEFT \\ 0.9U(1,1) + 0.1U(2,1) & DOWN \\ 0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1) & RIGHT \end{bmatrix}$$

- We have to do this for all fields!!!!

26

Example

$$\pi^*(s) = \arg\max_a \sum_{s'} P(s'|s,a)U(s')$$



- To find the best action in (1,1)

$$\pi^*(1,1) = \arg\max \begin{bmatrix} 0{,}7456 & UP \\ 0{,}7107 & LEFT \\ 0{,}7 & DOWN \\ 0.6707 & RIGHT \end{bmatrix}$$

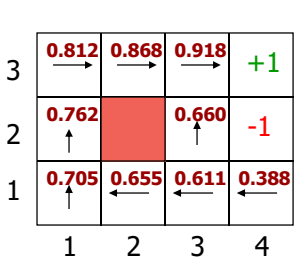
- give *Up* as best action

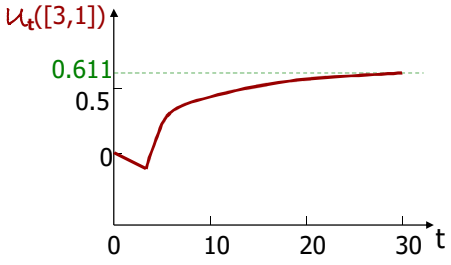
27

Value Iteration: the result

- Initialize the utility of each non-terminal state s_i to $\mathcal{U}_0(i) = 0$
- For t = 0, 1, 2, …, do:

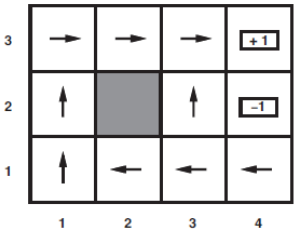
$$\mathcal{U}_{t+1}(i) \leftarrow \mathcal{R}(i) + \max_a \Sigma_k \mathbf{P}(k \mid a.i)\, \mathcal{U}_t(k)$$

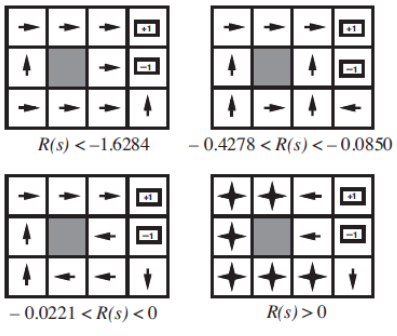




28

The Reward is important

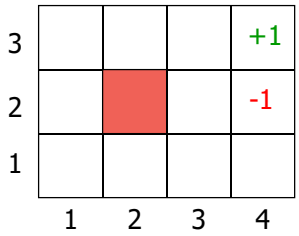




29

Infinite Horizon

In many problems, e.g., the robot navigation example, histories are potentially unbounded and the same state can be reached many times



$$U(i) = R(i) + \gamma \max_a \sum_j P(j \mid a.i)\ U(j)$$

One trick:
Use discounting to make infinite Horizon problem mathematically tractable

30

Value Iteration (finite and non-finite)

function VALUE-ITERATION(mdp, ϵ) **returns** a utility function
inputs: 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
local variables: U, U', vectors of utilities for states in S, initially zero
δ, the maximum change in the utility of any state in an iteration

repeat
$U \leftarrow U'$; $\delta \leftarrow 0$
for each state s **in** S **do**
$U'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s']$
if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
until $\delta < \epsilon(1-\gamma)/\gamma$
return U

31

Bellmann eq. is a contraction

- two important properties of contractions:
  - A contraction has **only one fixed point**; if there were two fixed points they would not get closer together when the function was applied, so it would not be a contraction.
  - **When the function is applied to any argument, the value must get closer to the fixed point**, so repeated application of a contraction always reaches the fixed point in the limit.

32

Value iteration

- Let U_i *denote the vector of utilities for all* the states at the ith iteration. Then the Bellman update equation can be written as

$$U_{i+1} \leftarrow BU_i$$

33

Value iteration

- use the **max norm**, which measures the "length" of a vector by the absolute value of its biggest component:

$$||U|| = max_s |U(s)|$$

- *Let U_i and U'_i be **any** two utility vectors. Then we have*

$$||BU_i - BU'_i|| \leq \gamma ||U_i - U'_i|| \qquad 17.7$$

34

Value iteration

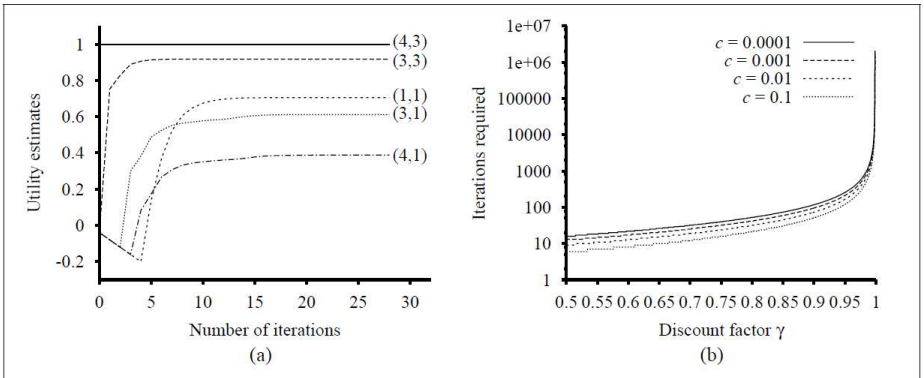


Figure 17.5 **FILES: .** (a) Graph showing the evolution of the utilities of selected states using value iteration. (b) The number of value iterations k required to guarantee an error of at most $\epsilon = c \cdot R_{\max}$, for different values of c, as a function of the discount factor γ.

36

Value iteration

- From the contraction, it can be shown that if the update is small (i.e., no state's utility changes by much), then the error, compared with the true utility function, also is small. More precisely,

$$\text{if } \|U_{i+1} - U_i\| < \varepsilon(1-\gamma)/\gamma \text{ then } \|U_{i+1} - U\| \leq \varepsilon \quad (17.8)$$

This is the stopping criteria for value iteration

37

Value iteration

- But the crucial question is!!!! How well will I do using this utility function?
- **policy loss**

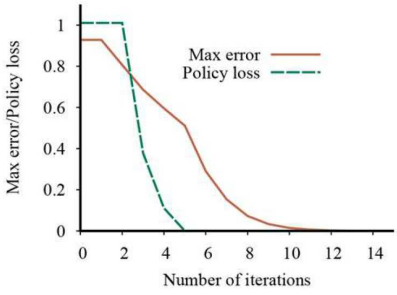
  $U^{\pi i}(s)$ is the utility obtained if πi is executed starting in *s,*
  policy loss $\|U^{\pi i} - U\|$ is the most the agent can lose by executing πi *instead of the optimal* policy π^*

38

Value iteration

- The policy loss of πi is connected to the error in U_i *by the following inequality:*

  if $||U_i - U|| < \varepsilon$ then $||U^{\pi_i} - U|| < 2\varepsilon$ (17.9)



The maximum error $\|U_i - U\|$ of the utility estimates and the policy loss $\|U^{\pi_i} - U\|$, as a function of the number of iterations of value iteration on the 4 × 3 world.

39

Policy Iteration

- Pick a policy Π at random

40

Policy Iteration

- Pick a policy Π at random
- Repeat:
  - Policy evaluation
    Compute the utility of each state for Π
    $\mathcal{U}_t(i) \leftarrow \mathcal{R}(i) + \sum_k \mathbf{P}(k \mid \Pi(i).i)\, \mathcal{U}_t(k)$

41

Policy Iteration

- Pick a policy Π at random
- Repeat:
  - Policy evaluation
    Compute the utility of each state for Π
    $\mathcal{U}_t(i) \leftarrow \mathcal{R}(i) + \sum_k \mathbf{P}(k \mid \Pi(i).i)\, \mathcal{U}_t(k)$
  - Policy improvement:
    Compute the policy Π' given these utilities
    $\Pi'(i) = \arg\max_a \sum_k \mathbf{P}(k \mid a.i)\, \mathcal{U}(k)$

42

Policy Iteration

- Pick a policy Π at random
- Repeat:
  - Policy evaluation:
    Compute the utility of each state for Π:
    $\mathcal{U}_t(i) \leftarrow \mathcal{R}(i) + \sum_k \mathbf{P}(k \mid \Pi(i).i)\ \mathcal{U}_t(k)$
  - Policy improvement:
    Compute the policy Π' given these utilities
    $\Pi'(i) = \arg\max_a \sum_k \mathbf{P}(k \mid a.i)\ \mathcal{U}(k)$
  - If $\Pi' = \Pi$ then return Π

43

Policy Iteration

function POLICY-ITERATION(mdp) **returns** a policy
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$
  local variables: U, a vector of utilities for states in S, initially zero
    π, a policy vector indexed by state, initially random

  repeat
    $U \leftarrow$ POLICY-EVALUATION(π, U, mdp)
    $unchanged? \leftarrow$ true
    for each state s **in** S **do**
      if $\max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s'] > \sum_{s'} P(s' \mid s, \pi[s])\ U[s']$ **then do**
        $\pi[s] \leftarrow \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s']$
        $unchanged? \leftarrow$ false
  until $unchanged?$
  return π

44

Linear equations

- By removing the max operator (Value Iteration) we can *also* solve the set of linear equations:

  $\mathcal{U}(i) = \mathcal{R}(i) + \sum_k \mathbf{P}(k \mid \Pi(i).i)\ \mathcal{U}(k)$

  (often a sparse system)
- Suppose we have $\Pi(1).1 =$ Up $\Pi(2).2$=Up
  U(1,1)= -0.04 + 0.8U(1,2)+0.1U(1,1)+0.1U(2,1)
  U(1,2)= -0.04 + 0.8U(1,3)+0.2U(1,2)
  …
- Can be solved in $O(n^3)$ by standard linear algebra methods
- For large state spaces we can mix value iteration and policy iteration

45

Further optimization

- All algorithms require updating the utility or policy for all states at once.
- At each step we can also select a subset for updating
  asynchronous policy iteration/mod. value iter.
  (can show it will converge if some conditions for initial policy and utility function hold)
- Leads to heuristic algorithms that concentrate on states that are likely to be reached by a good policy.
  - "if one has no intention of throwing oneself off a cliff, one should not spend time worrying about the exact value of the resulting state"

46

Summary

- Decision making under uncertainty
- Sequential decision making
  - Utility function
  - Value iteration
  - Policy iteration

47