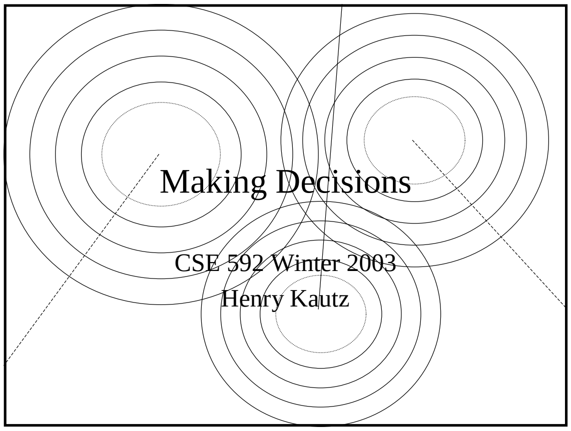
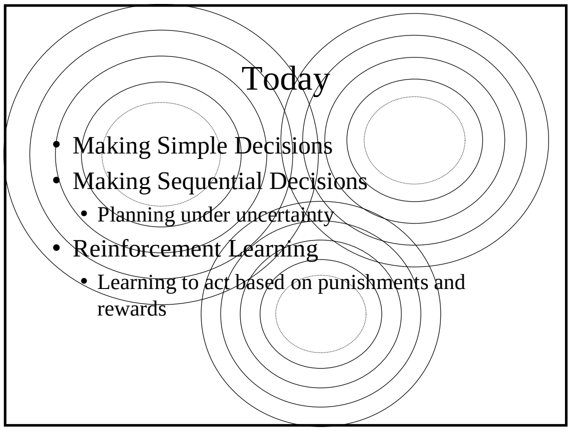


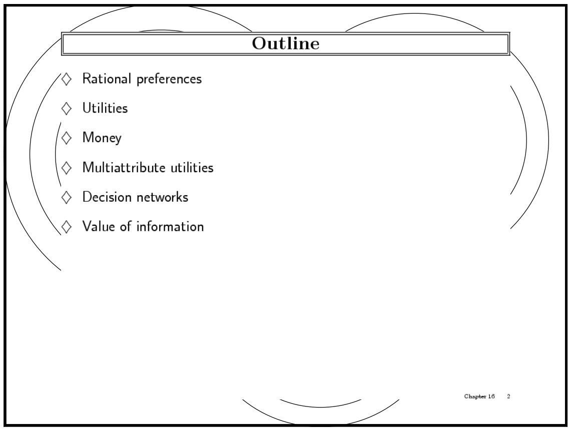
Making Decisions

CSE 592 Winter 2003
Henry Kautz



Today

- Making Simple Decisions
- Making Sequential Decisions
  - Planning under uncertainty
- Reinforcement Learning
  - Learning to act based on punishments and rewards



Outline

◇ Rational preferences

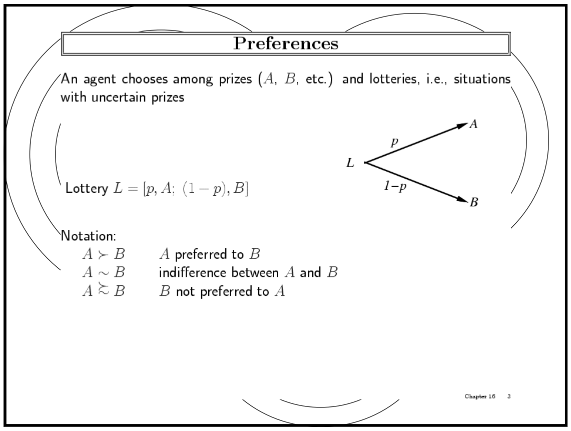
◇ Utilities

◇ Money

◇ Multiattribute utilities

◇ Decision networks

◇ Value of information

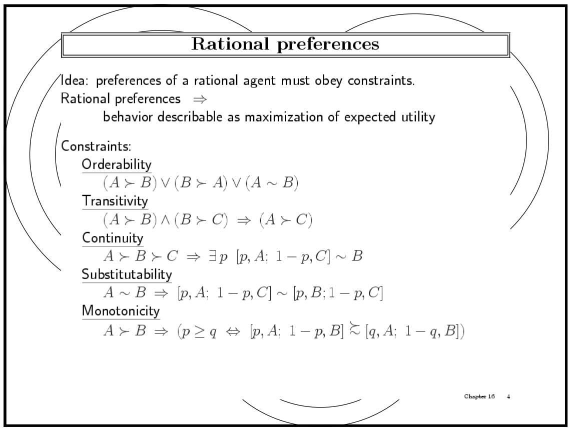


Preferences

An agent chooses among prizes (A, B, etc.) and lotteries, i.e., situations with uncertain prizes

Lottery $L = [p, A;\ (1-p), B]$

Notation:
$A \succ B$ A preferred to B
$A \sim B$ indifference between A and B
$A \succsim B$ B not preferred to A



Rational preferences

Idea: preferences of a rational agent must obey constraints.
Rational preferences $\Rightarrow$
behavior describable as maximization of expected utility

Constraints:

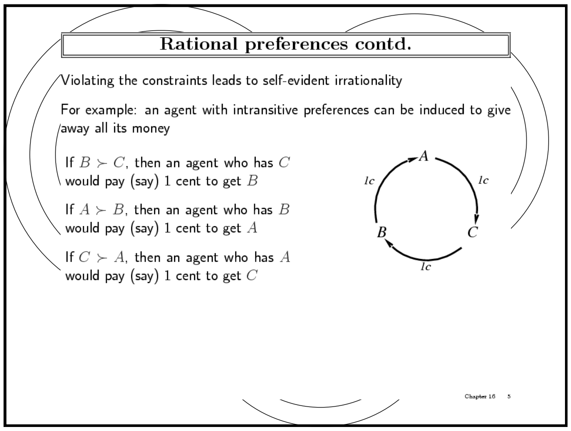
Orderability
$(A \succ B) \vee (B \succ A) \vee (A \sim B)$

Transitivity
$(A \succ B) \wedge (B \succ C) \Rightarrow (A \succ C)$

Continuity
$A \succ B \succ C \Rightarrow \exists p\ [p, A;\ 1-p, C] \sim B$

Substitutability
$A \sim B \Rightarrow [p, A;\ 1-p, C] \sim [p, B; 1-p, C]$

Monotonicity
$A \succ B \Rightarrow (p \geq q \Leftrightarrow [p, A;\ 1-p, B] \succsim [q, A;\ 1-q, B])$



Rational preferences contd.

Violating the constraints leads to self-evident irrationality

For example: an agent with intransitive preferences can be induced to give away all its money

If $B \succ C$, then an agent who has C would pay (say) 1 cent to get B

If $A \succ B$, then an agent who has B would pay (say) 1 cent to get A

If $C \succ A$, then an agent who has A would pay (say) 1 cent to get C

Maximizing expected utility

Theorem (Ramsey, 1931; von Neumann and Morgenstern, 1944):
Given preferences satisfying the constraints
there exists a real-valued function U such that

$$U(A) \geq U(B) \quad \Leftrightarrow \quad A \succsim B$$
$$U([p_1, S_1;\ \ldots\ ;\ p_n, S_n]) = \Sigma_i\ p_i U(S_i)$$

MEU principle:
Choose the action that maximizes expected utility

Note: an agent can be entirely rational (consistent with MEU)
without ever representing or manipulating utilities and probabilities

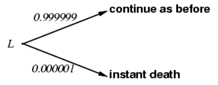
E.g., a lookup table for perfect tictactoe

Utilities

Utilities map states to real numbers. Which numbers?

Standard approach to assessment of human utilities:
- compare a given state A to a standard lottery L_p that has
  - "best possible prize" $u_\top$ with probability p
  - "worst possible catastrophe" $u_\perp$ with probability $(1-p)$
- adjust lottery probability p until $A \sim L_p$

pay $30 ~



Utility scales

Normalized utilities: $u_\top = 1.0$, $u_\perp = 0.0$

Micromorts: one-millionth chance of death
useful for Russian roulette, paying to reduce product risks, etc.

QALYs: quality-adjusted life years
useful for medical decisions involving substantial risk

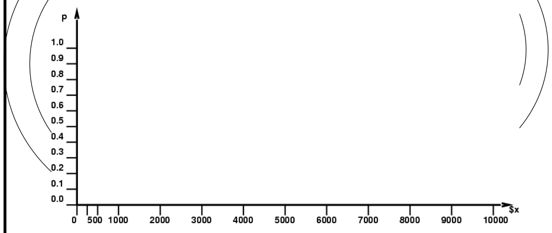
Note: behavior is **invariant** w.r.t. +ve linear transformation

$$U'(x) = k_1 U(x) + k_2 \quad \text{where } k_1 > 0$$

With deterministic prizes only (no lottery choices), only
ordinal utility can be determined, i.e., total order on prizes

Student group utility

For each x, adjust p until half the class votes for lottery (M=10,000)



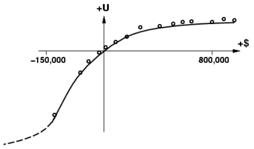
Money

Money does **not** behave as a utility function

Given a lottery L with expected monetary value $EMV(L)$,
usually $U(L) < U(EMV(L))$, i.e., people are risk-averse

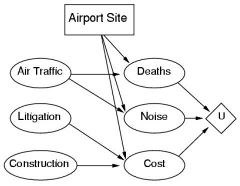
Utility curve: for what probability p am I indifferent between a prize x and a lottery $[p, \$M;\ (1-p), \$0]$ for large M?

Typical empirical data, extrapolated with risk-prone behavior:

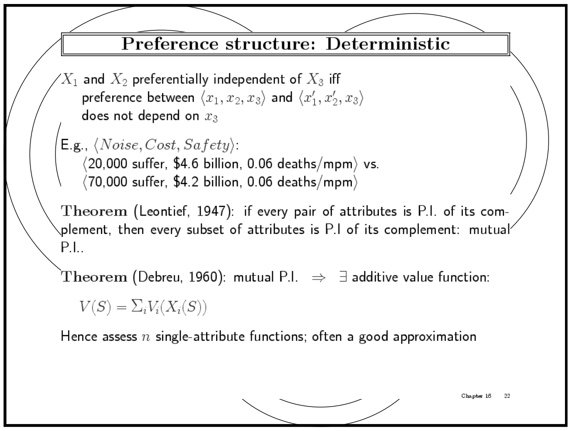


Decision networks

Add action nodes and utility nodes to belief networks
to enable rational decision making



Algorithm:
```
For each value of action node
    compute expected value of utility node given action, evidence
Return MEU action
```



Preference structure: Deterministic

X_1 and X_2 preferentially independent of X_3 iff
preference between $\langle x_1, x_2, x_3\rangle$ and $\langle x'_1, x'_2, x_3\rangle$
does not depend on x_3

E.g., $\langle Noise, Cost, Safety\rangle$:
⟨20,000 suffer, $4.6 billion, 0.06 deaths/mpm⟩ vs.
⟨70,000 suffer, $4.2 billion, 0.06 deaths/mpm⟩

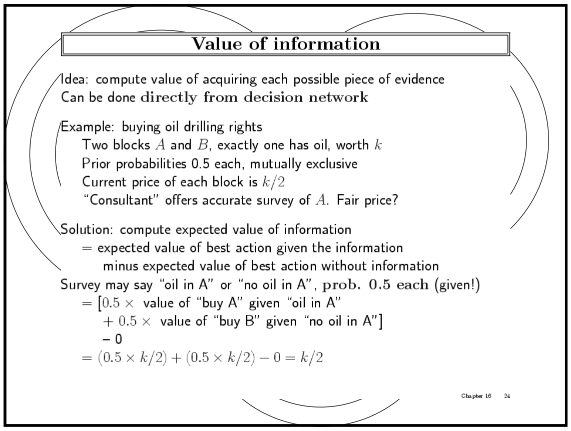
Theorem (Leontief, 1947): if every pair of attributes is P.I. of its complement, then every subset of attributes is P.I of its complement: mutual P.I..

Theorem (Debreu, 1960): mutual P.I. ⇒ ∃ additive value function:

$$V(S) = \Sigma_i V_i(X_i(S))$$

Hence assess n single-attribute functions; often a good approximation

Chapter 16 22



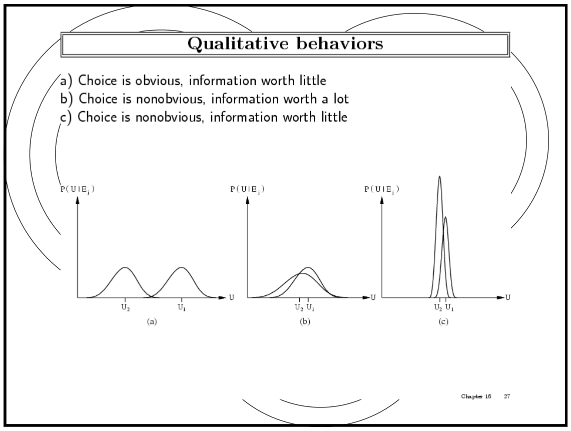
Value of information

Idea: compute value of acquiring each possible piece of evidence
Can be done **directly from decision network**

Example: buying oil drilling rights
Two blocks A and B, exactly one has oil, worth k
Prior probabilities 0.5 each, mutually exclusive
Current price of each block is $k/2$
"Consultant" offers accurate survey of A. Fair price?

Solution: compute expected value of information
= expected value of best action given the information
minus expected value of best action without information
Survey may say "oil in A" or "no oil in A", **prob. 0.5 each** (given!)
= [$0.5 \times$ value of "buy A" given "oil in A"
$+ 0.5 \times$ value of "buy B" given "no oil in A"]
− 0
$= (0.5 \times k/2) + (0.5 \times k/2) - 0 = k/2$

Chapter 16 24

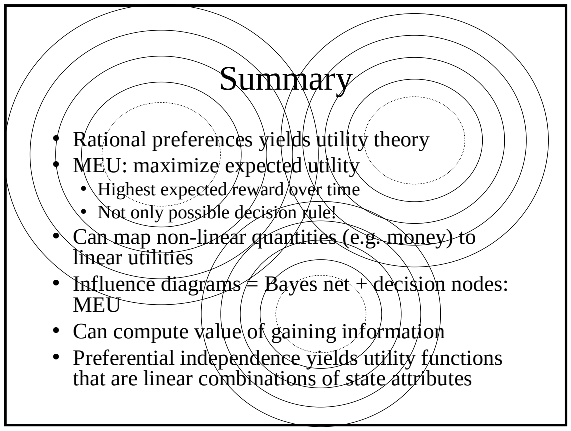


Qualitative behaviors

a) Choice is obvious, information worth little
b) Choice is nonobvious, information worth a lot
c) Choice is nonobvious, information worth little

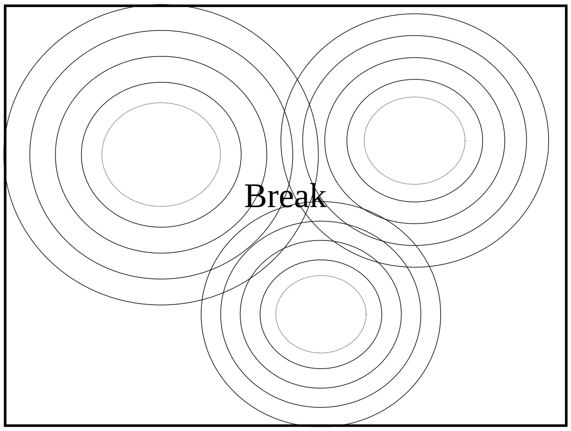
(a) (b) (c)

Chapter 16 27

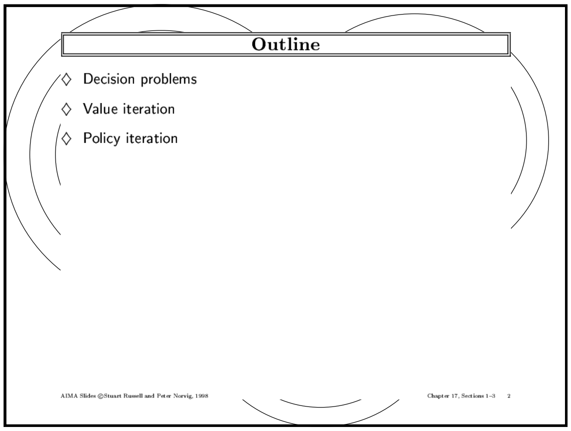


Summary

- Rational preferences yields utility theory
- MEU: maximize expected utility
  - Highest expected reward over time
  - Not only possible decision rule!
- Can map non-linear quantities (e.g. money) to linear utilities
- Influence diagrams = Bayes net + decision nodes: MEU
- Can compute value of gaining information
- Preferential independence yields utility functions that are linear combinations of state attributes



Break

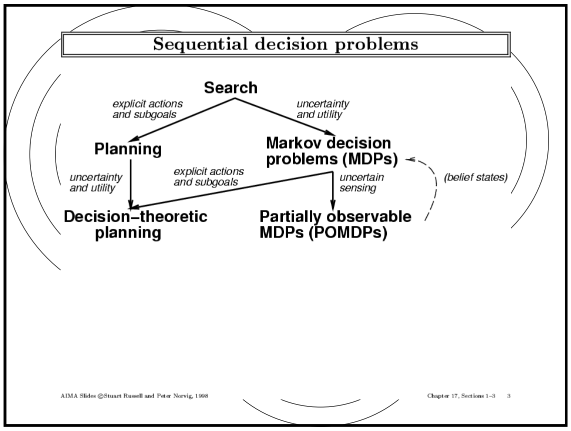


Outline

◇ Decision problems
◇ Value iteration
◇ Policy iteration

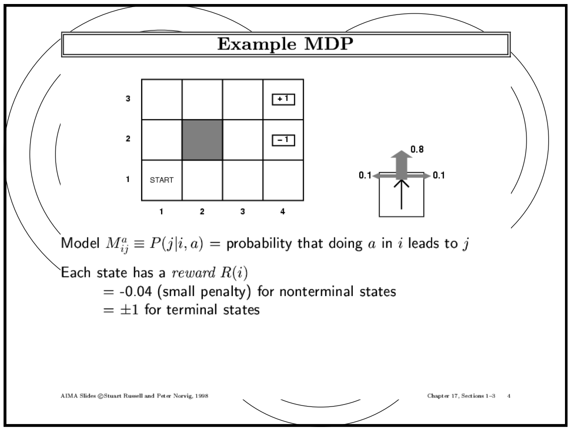
AIMA Slides ©Stuart Russell and Peter Norvig, 1998

Chapter 17, Sections 1–3 2



Sequential decision problems

Search
explicit actions and subgoals
uncertainty and utility
Planning
Markov decision problems (MDPs)
uncertainty and utility
explicit actions and subgoals
uncertain sensing
(belief states)
Decision–theoretic planning
Partially observable MDPs (POMDPs)

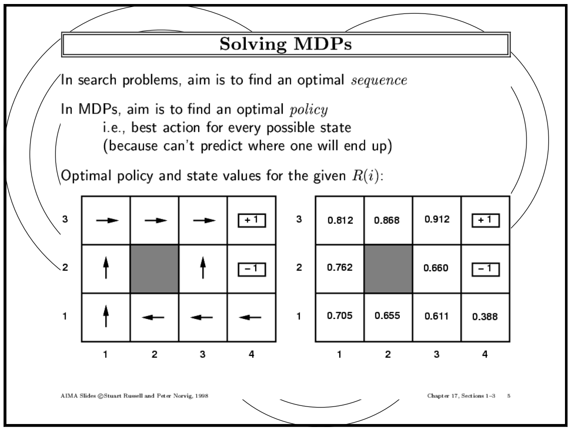


Example MDP

3				+1
2		▨		−1
1	START			
	1	2	3	4

Model $M_{ij}^a \equiv P(j|i,a)$ = probability that doing a in i leads to j

Each state has a *reward* $R(i)$
= -0.04 (small penalty) for nonterminal states
= ±1 for terminal states



Solving MDPs

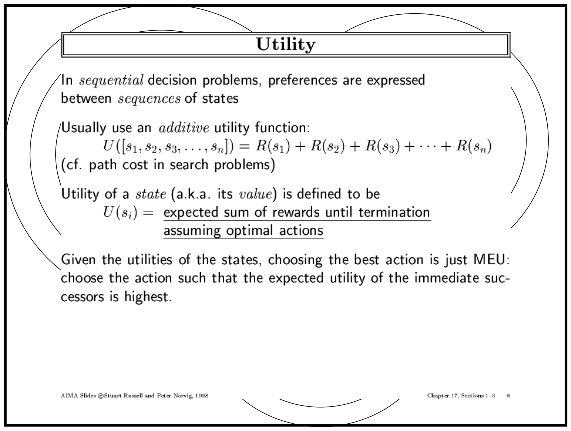
In search problems, aim is to find an optimal *sequence*

In MDPs, aim is to find an optimal *policy*
i.e., best action for every possible state
(because can't predict where one will end up)

Optimal policy and state values for the given $R(i)$:

3	→	→	→	+1
2	↑	▨	↑	−1
1	↑	←	←	←
	1	2	3	4

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



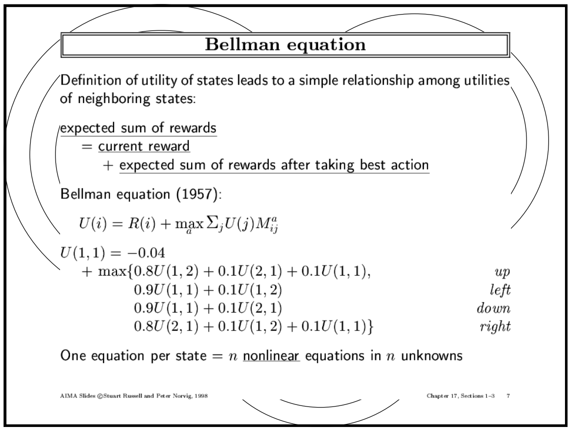
Utility

In *sequential* decision problems, preferences are expressed between *sequences* of states

Usually use an *additive* utility function:
$U([s_1, s_2, s_3, \ldots, s_n]) = R(s_1) + R(s_2) + R(s_3) + \cdots + R(s_n)$
(cf. path cost in search problems)

Utility of a *state* (a.k.a. its *value*) is defined to be
$U(s_i) = \underline{\text{expected sum of rewards until termination}}$
$\underline{\text{assuming optimal actions}}$

Given the utilities of the states, choosing the best action is just MEU: choose the action such that the expected utility of the immediate successors is highest.



Bellman equation

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

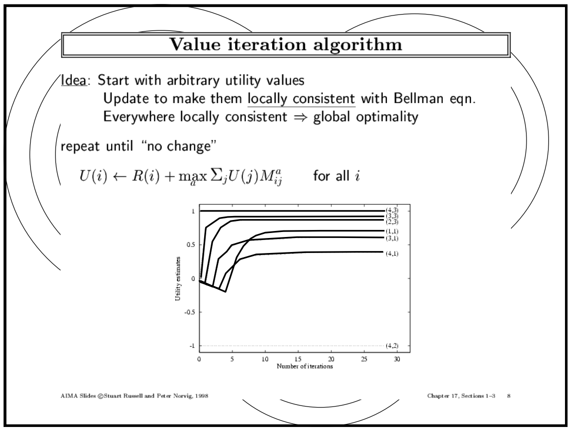
$\underline{\text{expected sum of rewards}}$
$= \underline{\text{current reward}}$
$+ \underline{\text{expected sum of rewards after taking best action}}$

Bellman equation (1957):

$$U(i) = R(i) + \max_a \Sigma_j U(j) M_{ij}^a$$

$$\begin{aligned}
U(1,1) = -0.04 & \\
+ \max\{ & 0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), && \textit{up} \\
& 0.9U(1,1) + 0.1U(1,2) && \textit{left} \\
& 0.9U(1,1) + 0.1U(2,1) && \textit{down} \\
& 0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\} && \textit{right}
\end{aligned}$$

One equation per state = n <u>nonlinear</u> equations in n unknowns

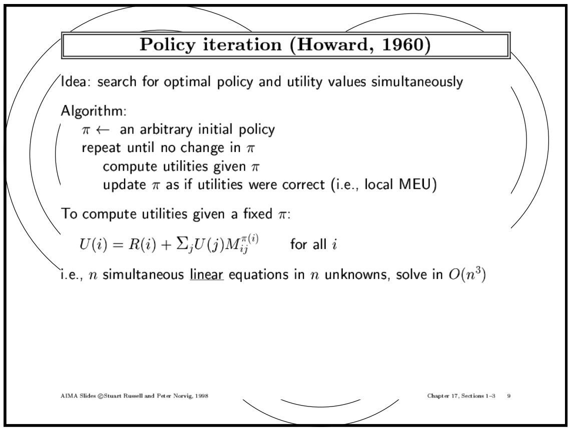


Value iteration algorithm

<u>Idea</u>: Start with arbitrary utility values
Update to make them <u>locally consistent</u> with Bellman eqn.
Everywhere locally consistent ⇒ global optimality

repeat until "no change"

$U(i) \leftarrow R(i) + \max_a \Sigma_j U(j) M_{ij}^a$ for all i



Policy iteration (Howard, 1960)

Idea: search for optimal policy and utility values simultaneously

Algorithm:

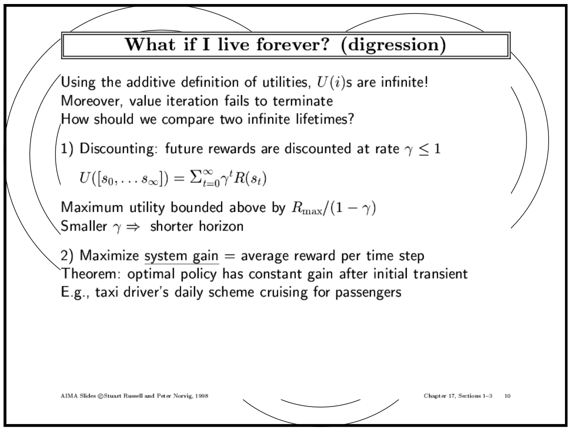
```
π ← an arbitrary initial policy
repeat until no change in π
    compute utilities given π
    update π as if utilities were correct (i.e., local MEU)
```

To compute utilities given a fixed π:

$$U(i) = R(i) + \Sigma_j U(j) M_{ij}^{\pi(i)} \quad \text{for all } i$$

i.e., n simultaneous linear equations in n unknowns, solve in $O(n^3)$

AIMA Slides ©Stuart Russell and Peter Norvig, 1998 Chapter 17, Sections 1–3 9



What if I live forever? (digression)

Using the additive definition of utilities, $U(i)$s are infinite!
Moreover, value iteration fails to terminate
How should we compare two infinite lifetimes?

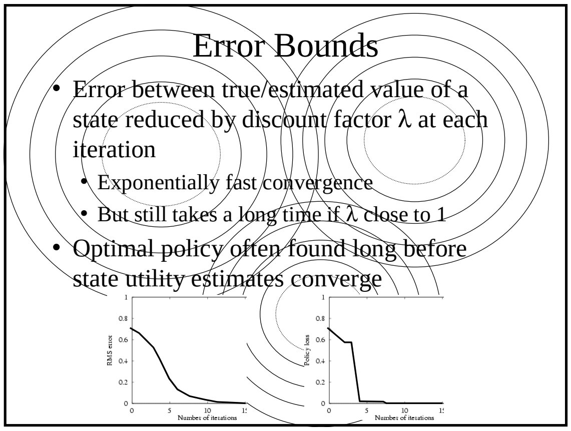
1) Discounting: future rewards are discounted at rate $\gamma \leq 1$

$$U([s_0, \ldots s_\infty]) = \Sigma_{t=0}^{\infty} \gamma^t R(s_t)$$

Maximum utility bounded above by $R_{\max}/(1-\gamma)$
Smaller $\gamma \Rightarrow$ shorter horizon

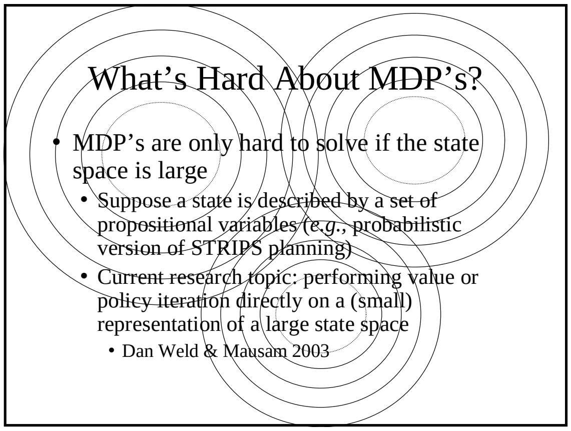
2) Maximize system gain = average reward per time step
Theorem: optimal policy has constant gain after initial transient
E.g., taxi driver's daily scheme cruising for passengers

AIMA Slides ©Stuart Russell and Peter Norvig, 1998 Chapter 17, Sections 1–3 10



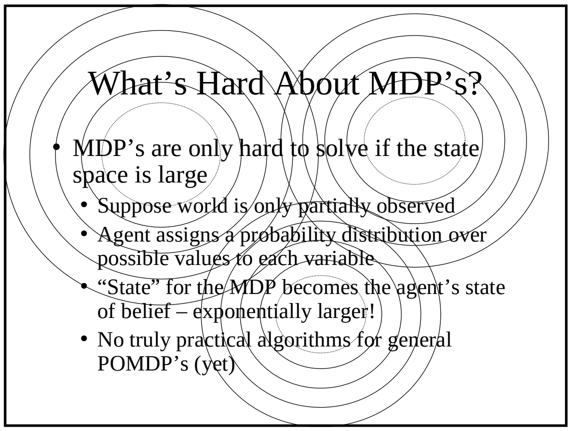
Error Bounds

- Error between true/estimated value of a state reduced by discount factor λ at each iteration
  - Exponentially fast convergence
  - But still takes a long time if λ close to 1
- Optimal policy often found long before state utility estimates converge



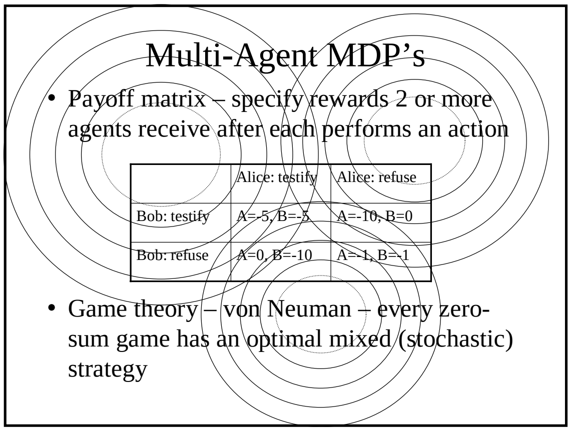
What's Hard About MDP's?

- MDP's are only hard to solve if the state space is large
  - Suppose a state is described by a set of propositional variables (*e.g.*, probabilistic version of STRIPS planning)
  - Current research topic: performing value or policy iteration directly on a (small) representation of a large state space
    - Dan Weld & Mausam 2003



What's Hard About MDP's?

- MDP's are only hard to solve if the state space is large
  - Suppose world is only partially observed
  - Agent assigns a probability distribution over possible values to each variable
  - "State" for the MDP becomes the agent's state of belief – exponentially larger!
  - No truly practical algorithms for general POMDP's (yet)



Multi-Agent MDP's

- Payoff matrix – specify rewards 2 or more agents receive after each performs an action

	Alice: testify	Alice: refuse
Bob: testify	A=-5, B=-5	A=-10, B=0
Bob: refuse	A=0, B=-10	A=-1, B=-1

- Game theory – von Neuman – every zero-sum game has an optimal mixed (stochastic) strategy

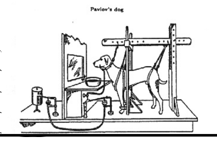
Summary

- Markov Decision Processes provide a general way of reasoning about sequential decision problems
- Solved by linear programming, value iteration, or policy iteration
- Discounting future rewards guarantees convergence of value/policy iteration
- Requires complete model of the world (*i.e.* the state transition function)
  - MPD – complete observations
  - POMDP – partial observations
- Large state spaces problematic

Break

Reinforcement Learning

- "Of several responses made to the same situation, those which are accompanied or closely followed by satisfaction to the animal will, other things being equal, be more firmly connected with the situation, so that, when it recurs, they will be more likely to recur; those which are accompanied or closely followed by discomfort to the animal will, other things being equal, have their connections with that situation weakened, so that, when it recurs, they will be less likely to occur. The greater the satisfaction or discomfort, the greater the strengthening or weakening of the bond." (Thorndike, 1911, p. 244)



The Reinforcement Learning Scenario

- How is learning to act possible when…
  - Actions have non-deterministic effects, that are initially unknown
  - Rewards or punishments come infrequently, at the end of long sequences of actions
  - The learner must decide what actions to take
  - The world is large and complex

RL Techniques

- Temporal-difference learning
  - Learns a utility function on states or on [state,action] pairs
  - Similar to backpropagation – treats the difference between expected / actual reward as an error signal, that is propagated backward in time
- Exploration functions
  - Balance exploration / exploitation
- Function approximation
  - Compress a large state space into a small one
  - Linear function approximation, neural nets, …
  - Generalization

Passive RL

- Given policy π, estimate $U^{\pi}(s)$
- Not given transition matrix or reward function!
- Epochs: training sequences

(1,1)→(1,2)→(1,3)→(1,2)→(1,3)→(1,2)→(1,1)→(1,2)→(2,2)→(3,2) <u>–1</u>
(1,1)→(1,2)→(1,3)→(2,3)→(2,2)→(2,3)→(3,3) <u>+1</u>
(1,1)→(1,2)→(1,1)→(1,2)→(1,1)→(2,1)→(2,2)→(2,3)→(3,3) <u>+1</u>
(1,1)→(1,2)→(2,2)→(1,2)→(1,3)→(2,3)→(1,3)→(2,3)→(3,3) <u>+1</u>
(1,1)→(2,1)→(2,2)→(2,1)→(1,1)→(1,2)→(1,3)→(2,3)→(2,2)→(3,2) <u>-1</u>
(1,1)→(2,1)→(1,1)→(1,2)→(2,2)→(3,2) <u>-1</u>

Approaches

- Direct estimation
  - Estimate *U^π(s)* as average total reward of epochs containing s (calculating from s to end of epoch)
  - Requires huge amount of data – does not take advantage of Bellman constraints!
    - Expected utility of a state = its own reward + expected utility of its successor states

Approaches

- Adaptive Dynamic Programming
  - Requires fully observable environment
  - Estimate transition function *M* from training data
  - Apply modified policy iteration to solve Bellman equation:

$$U^{\pi} = R(s) + \lambda \sum_{s'} M^{\pi}_{s,s'} U^{\pi}(s')$$

  - Drawbacks: requires complete observations, and you don't usually need value of all states

Temporal Difference Learning

- Ideas
  - Do backups on a per-epoch basis
  - Don't even try to estimate entire transition function!
  - For each transition from s to s', update:

$$U^{\pi}(s) \leftarrow U^{\pi}(s) + \alpha(R(s) + \lambda U^{\pi}(s') - U^{\pi}(s))$$

Example:

Q-Learning

- Version of TD-learning where instead of learning a value function on states, we learn one on [state,action] pairs

$$U^{\pi}(s) \leftarrow U^{\pi}(s) + \alpha(R(s) + \lambda U^{\pi}(s') - U^{\pi}(s))$$

$$Q(a,s) \leftarrow Q(a,s) + \alpha(R(s) + \max_{a'} Q(a',s') - Q(a,s))$$

- Why do this?

Active Reinforcement Learning

- Suppose agent has to create its own policy while learning
- First approach:
  - Start with arbitrary policy
  - Apply Q-Learning
  - New policy: in state *s*, choose action *a* that maximizes *Q(a,s)*
  - *Problem?*

Exploration Functions

- Too easily stuck in non-optimal space
- Simple fix: with fixed probability perform a random action
- Better: increase estimated expected value of states that have been rarely explored
- “Exploration versus exploitation tradeoff”

Function Approximation

- Problem of large state spaces remain
  - Never enough training data!
  - Want to generalize what has been learned to new situations
- Idea:
  - Replace large state table by a smaller, parameterized function
  - Updating the value of state will change the value assigned to many other similar states

Linear Function Approximation

- Represent *U(s)* as a weighted sum of features (basis functions) of s

$$\hat{U}_\theta(s) = \theta_1 f_1(s) + \theta_2 f_2(s) + \ldots + \theta_n f_n(s)$$

- Update each parameter separately, *e.g:*

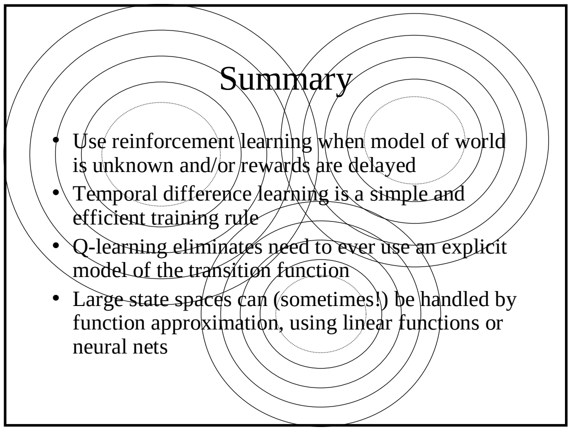
$$\theta_i \leftarrow \theta_i + \alpha(R(s) + \lambda\hat{U}_\theta(s') - \hat{U}_\theta(s))\frac{\partial\hat{U}_\theta(s)}{\partial\theta_i}$$

Neural Nets

- Neural nets can be used to create powerful function approximators
- Can become unstable (unlike linear functions)
- For TD-learning, apply difference signal to neural net output and perform back-propagation

Example

Demo



Summary

- Use reinforcement learning when model of world is unknown and/or rewards are delayed
- Temporal difference learning is a simple and efficient training rule
- Q-learning eliminates need to ever use an explicit model of the transition function
- Large state spaces can (sometimes!) be handled by function approximation, using linear functions or neural nets