An Introduction to Markov Decision Processes

Bob Givan
Purdue University

Ron Parr
Duke University

Outline

Markov Decision Processes defined (Bob)

- Objective functions
- Policies

Finding Optimal Solutions (Ron)

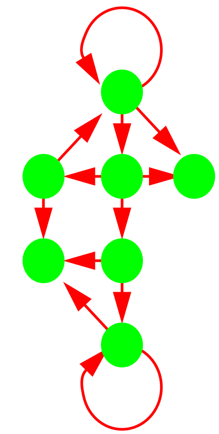
- Dynamic programming
- Linear programming

Refinements to the basic model (Bob)

- Partial observability
- Factored representations

Stochastic Automata with Utilities

A *Markov Decision Process* (MDP) model contains:

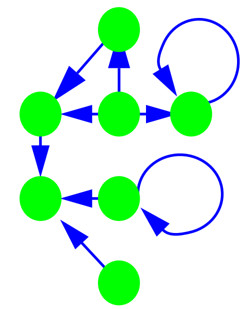


- A set of possible world states *S*
- A set of possible actions *A*
- A real valued reward function *R(s,a)*
- A description *T* of each action's effects in each state.

We assume the Markov Property**:** *the effects of an action taken in a state depend only on that state and not on the prior history.*

Stochastic Automata with Utilities

A *Markov Decision Process* (MDP) model contains:



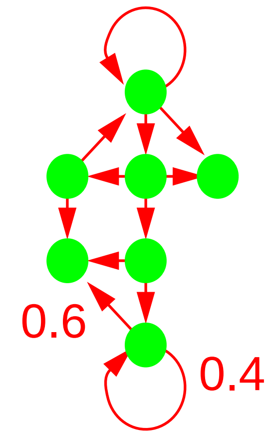
- A set of possible world states *S*
- A set of possible actions *A*
- A real valued reward function *R(s,a)*
- A description *T* of each action's effects in each state.

We assume the Markov Property**:** *the effects of an action taken in a state depend only on that state and not on the prior history.*

Representing Actions

Deterministic Actions:

- $T: S \times A \rightarrow S$ For each state and action we specify a new state.



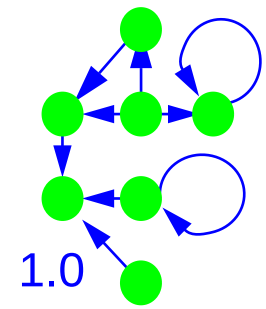
Stochastic Actions:

- $T: S \times A \rightarrow Prob(S)$ For each state and action we specify a probability distribution over next states. Represents the distribution $P(s' \mid s, a)$.

Representing Actions

Deterministic Actions:

- $T: S \times A \to S$ For each state and action we specify a new state.

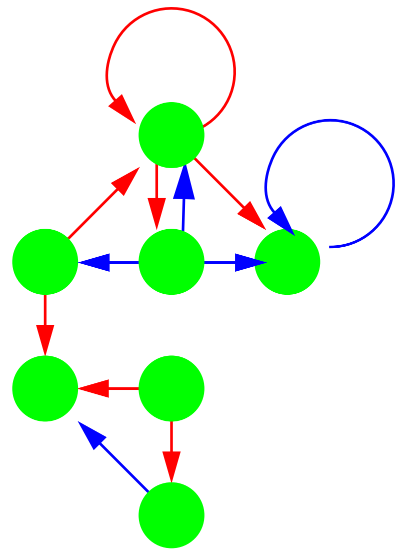


Stochastic Actions:

- $T: S \times A \to Prob(S)$ For each state and action we specify a probability distribution over next states. Represents the distribution $P(s' \mid s, a)$.

Representing Solutions

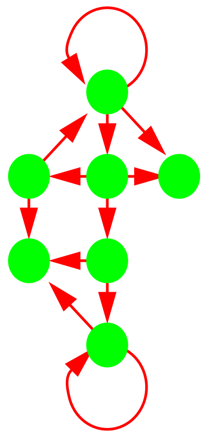
A *policy* π is a mapping from S to A



Following a Policy

Following a policy π:

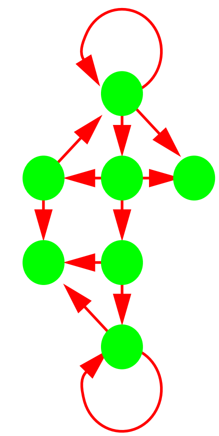
1. Determine the current state s
2. Execute action $\pi(s)$
3. Goto step 1.



Assumes full observability: the new state resulting from executing an action will be known to the system

Evaluating a Policy

How good is a policy π in a state s ?



For deterministic actions just total the rewards obtained... but result may be infinite.

For stochastic actions, instead *expected total reward* obtained–again typically yields infinite value.

How do we compare policies of infinite value?

Objective Functions

An objective function maps infinite sequences of rewards to single real numbers (representing utility)

Options:

1. Set a finite horizon and just total the reward
2. Discounting to prefer earlier rewards
3. Average reward rate in the limit

Discounting is perhaps the most analytically tractable and most widely studied approach

Discounting

A reward *n* steps away is discounted by γ^n for discount rate $0 < \gamma < 1$.

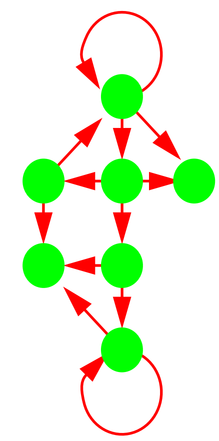
- models mortality: you may die at any moment
- models preference for shorter solutions
- a smoothed out version of limited horizon lookahead

We use *cumulative discounted reward* as our objective

(**Max value** <= $M + \gamma \cdot M + \gamma^2 \cdot M + ... = \frac{1}{1-\gamma} \cdot M$)

Value Functions

A value function $V_\pi : S \rightarrow \Re$ represents the expected objective value obtained following policy π from each state in S .



Value functions partially order the policies,

- but at least one optimal policy exists, and
- all optimal policies have the same value function, V^*

Bellman Equations

Bellman equations relate the value function to itself via the problem dynamics.

For the discounted objective function,

$$V_{\pi}(s) = R(s, \pi(s)) + \sum_{s' \in S} T(s, \pi(s), s') \cdot \gamma \cdot V_{\pi}(s')$$

$$V^{*}(s) = \underset{a \in A}{\mathbf{MAX}} \left(R(s, a) + \sum_{s' \in S} T(s, a, s') \cdot \gamma \cdot V^{*}(s') \right)$$

In each case, there is one equation per state in S

Finite-horizon Bellman Equations

Finite-horizon values at adjacent horizons are related by the action dynamics

$$V_{\pi,0}(s) = R(s, \pi(s))$$

$$V_{\pi,n}(s) = R(s, a) + \sum_{s' \in S} T(s, a, s') \cdot \gamma \cdot V_{\pi,n-1}(s')$$

Relation to Model Checking

Some thoughts on the relationship

- MDP solution focuses critically on expected value
- Contrast safety properties which focus on worst case
- This contrast allows MDP methods to exploit sampling and approximation more aggressively

- At this point, Ron Parr spoke on solution methods for about 1/2 an hour, and then I continued.

Large State Spaces

In AI problems, the "state space" is typically

- astronomically large
- described implicitly, not enumerated
- decomposed into factors, or aspects of state

Issues raised:

- How can we represent reward and action behaviors in such MDPs?
- How can we find solutions in such MDPs?

A Factored MDP Representation

- State Space *S* — assignments to state variables:

  `On-Mars?, Need-Power?, Daytime?,..etc...`

- Partitions — each block a DNF formula (or BDD, etc)

  Block 1: `not On-Mars?`
  Block 2: `On-Mars? and Need-Power?`
  Block 3: `On-Mars? and not Need-Power?`

- Reward function *R* — labelled state-space partition:

  Block 1: `not On-Mars?`. . . . . Reward=0
  Block 2: `On-Mars? and Need-Power?`. . . . . . . Reward=4
  Block 3: `On-Mars? and not Need-Power?` . . Reward=5

Factored Representations of Actions

- Assume: actions affect state variables independently.[1]

  e.g......Pr(Nd-Power? ^ On-Mars? | *x, a*)

  = Pr (Nd-Power? | *x, a*) * Pr (On-Mars? | *x, a*)

- Represent effect on each state variable as labelled partition:

Effects of Action `Charge-Battery` on variable `Need-Power?`

		Pr(Need-Power? | Block n)
Block 1:	<u>not</u> On-Mars?	0.9
Block 2:	On-Mars? <u>and</u> Need-Power?	0.3
Block 3:	On-Mars? <u>and</u> <u>not</u> Need-Power?	0.1

1. This assumption can be relaxed.

Representing Blocks

- Identifying “irrelevant” state variables
- Decision trees
- DNF formulas
- Binary/Algebraic Decision Diagrams

Partial Observability

System state can not always be determined

$\Rightarrow$ a Partially Observable MDP (POMDP)

- Action outcomes are not fully observable
- Add a set of observations *O* to the model
- Add an observation distribution $U(s,o)$ for each state
- Add an initial state distribution *I*

Key notion: belief state, a distribution over system states representing "where I think I am"

POMDP to MDP Conversion

Belief state Pr(x) can be updated to Pr(x'|o) using Bayes' rule:

$$\Pr(s'|s,o) = \Pr(o|s,s') \Pr(s'|s) / \Pr(o|s)$$

$$= U(s',o)\ T(s',a,s) \text{ normalized}$$

$$\Pr(s'|o) = \Pr(s'|s,o) \Pr(s)$$

A POMDP is Markovian and fully observable relative to the belief state.

$\Rightarrow$ a POMDP can be treated as a continuous state MDP

Belief State Approximation

Problem: When MDP state space is astronomical, belief states cannot be explicitly represented.

Consequence: MDP conversion of POMDP impractical

Solution: Represent belief state approximately

- Typically exploiting factored state representation
- Typically exploiting (near) conditional independence properties of the belief state factors