Lecture 12: MDP1

Victor R. Lesser

CMPSCI 683

Fall 2010

Biased Random GSAT - WalkSat

function WALKSAT(*clauses*, *p*, *max_flips*) **returns** a satisfying model or *failure*
  inputs: *clauses*, a set of clauses in propositional logic
    p, the probability of choosing to do a "random walk" move, typically around 0.5
    max_flips, number of flips allowed before giving up

  model ← a random assignment of *true*/*false* to the symbols in *clauses*
  for *i* = 1 **to** *max_flips* **do**
    if *model* satisfies *clauses* **then return** *model*
    clause ← a randomly selected clause from *clauses* that is false in *model*
    with probability *p* flip the value in *model* of a randomly selected symbol from *clause*
    else flip whichever symbol in *clause* maximizes the number of satisfied clauses
  return *failure*

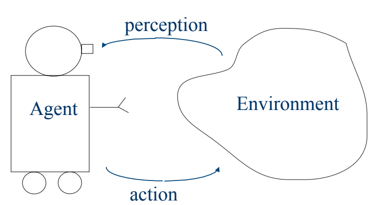
Notice no random restart

Figure 7.18 The WALKSAT algorithm for checking satisfiability by randomly flipping the values of variables. Many versions of the algorithm exist.

Today's lecture

- Search where there is Uncertainty in Operator Outcome --Sequential Decision Problems
  - Planning Under Uncertainty
  - Markov Decision Processes (MDP)

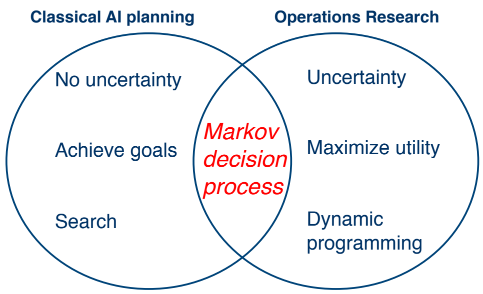
Planning under uncertainty



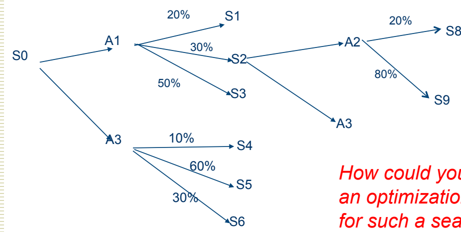
Utility depends on a sequence of decisions

Actions have unpredictable outcomes

Approaches to planning



Search with Uncertainty



How could you define an optimization criteria for such a search?

What is the output of the search?

Stochastic shortest-path problems

- Given a start state, the objective is to minimize the expected cost of reaching a goal state.
- S: a finite set of states
- $A(i)$, $i \in S$: a finite set of actions available in state i
- $P_{ij}(a)$: probability of reaching state j after action a in state i
- $C_i(a)$: expected cost of taking action a in state i

Markov decision process

- A model of sequential decision-making developed in operations research in the 1950's.
- Allows reasoning about actions with uncertain outcomes.
- MDPs have been adopted by the AI community as a framework for:
  - Decision-theoretic planning (e.g., [Dean et al., 1995])
  - Reinforcement learning (e.g., [Barto et al., 1995])

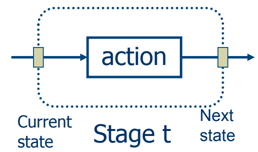
Markov Decision Processes (MDP)

- S - finite set of domain states
- A - finite set of actions
- $P(s' \mid s, a)$ - state transition function
- $R(s)$, $R(s, a)$, or $R(s, a, s')$ - reward function
  - *Could be negative to reflect cost*
- S_0 - initial state
- The Markov assumption:

$$P(s_t \mid s_{t-1}, s_{t-2}, \ldots, s_1, a) = P(s_t \mid s_{t-1}, a)$$

The MDP Framework (cont)

- Agent *fully observes its current state*
- Markovian property: the state contains enough information to pick the optimal action
- Objective: maximize the expected reward of the start state
- Policy: π : S --> A; how to find optimal policy



Policy vs. Plan

A Finite MDP with Loops

Recycling Robot

- At each step, robot has to decide whether it should
  - (1) actively search for a can.
  - (2) wait for someone to bring it a can.
  - (3) go to home base and recharge.
- Searching is better but runs down the battery; if runs out of power while searching, has to be rescued (which is bad and represented as a penalty).
- Decisions made on basis of current energy level: `high`, `low`.
- Reward = number of cans collected

Recycling Robot MDP

$S = \{\texttt{high}, \texttt{low}\}$

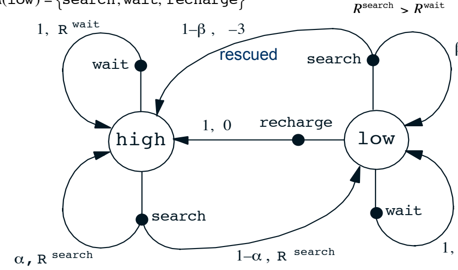
$A(\texttt{high}) = \{\texttt{search}, \texttt{wait}\}$

$A(\texttt{low}) = \{\texttt{search}, \texttt{wait}, \texttt{recharge}\}$

$R^{\texttt{search}}$ = expected no. of cans while searching

$R^{\texttt{wait}}$ = expected no. of cans while waiting

$R^{\texttt{search}} > R^{\texttt{wait}}$

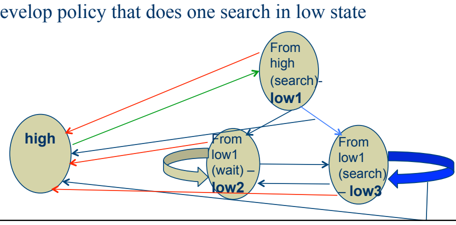


What is an example of a policy?

Where is there uncertainty?

Breaking the Markov Assumption to get a Better Policy

- Concerned about path to Low State (whether you came as a result of a search from a high state or a search or wait action from a low state (high, low1, low2, low3)
  - can more accurately reflect likelihood of rescue
  - develop policy that does one search in low state



Goals and Rewards

- Is a scalar reward signal an adequate notion of a goal?—maybe not, but it is surprisingly flexible.
- A goal should specify **what** we want to achieve, not **how** we want to achieve it.
  - It is not the path to a specific state but reaching a specific state – fits with Markov Assumption
- A goal must be outside the agent's direct control—thus outside the agent.
- The agent must be able to measure success:
  - Explicitly in terms of a reward;
  - frequently during its lifespan.

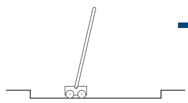
Performance criteria

- Specify how to combine rewards over multiple time steps or histories.
- Finite horizon problems involve a fixed number of steps.
- The best action in each state may depend on the number of steps left, hence it is **non-stationary**.
  - Finite horizon non-stationary problems can be solved by adding the number of steps left to the state – adds more states
- Infinite horizon policies depend only on the current state, hence the optimal policy is **stationary**.

Performance criteria cont.

- The assumption the agent's preferences between state sequences is stationary: $[s_0,s_1,s_2,\ldots] > [s_0,s_1',s_2',\ldots]$ iff $[s_1,s_2,\ldots] > [s_1',s_2',\ldots]$
  - *how you got to a state does not affect the best policy from that state*
- This leads to just two ways to define utilities of histories:
  - Additive rewards: utility of a history is $U([s_0,a_1,s_1,a_2,s_2,\ldots]) = R(s_0) + R(s_1) + R(s_2) + \ldots$
  - Discounted rewards: utility of a history is $U([s_0,a_1,s_1,a_2,s_2,\ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) \ldots$
- With a **proper policy** (guaranteed to reach a terminal state) no discounting is needed.
- An alternative to discounting in infinite-horizon problems is to optimize the average reward per time step.

An Example



Avoid **failure:** the pole falling beyond a critical angle or the cart hitting end of track.

As an **episodic task** where episode ends upon failure:

reward $= +1$ for each step before failure

$\Rightarrow$ return $=$ number of steps before failure

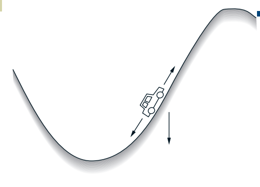
As a **continuing task** with discounted return:

reward $= -1$ upon failure; 0 otherwise

$\Rightarrow$ return $= -\gamma^k$, for k steps before failure

In either case, return is maximized by avoiding failure for as long as possible.

Another Example



Get to the top of the hill as quickly as possible.

reward $= -1$ for each step where **not** at top of hill

$\Rightarrow$ return $= -$ number of steps before reaching top of hill

Return is maximized by minimizing number of steps to reach the top of the hill.

Policies and utilities of states

- A policy π is a mapping from states to actions.
- An optimal policy π* maximizes the expected reward:

$$\pi^* = \arg\max_{\pi} E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi\right]$$

- The utility of a state

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi, s_0 = s\right]$$

A simple grid environment

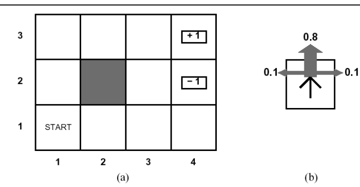


Figure 17.1 (a) A simple 4×3 environment that presents the agent with a sequential decision problem. (b) Illustration of the transition model of the environment: the "intended" outcome occurs with probability 0.8, but with probability 0.2 the agent moves at right angles to the intended direction.

Example: An Optimal Policy

A policy is a choice of what action to choose at each state

An Optimal Policy is a policy where you are always choosing the action that maximizes the “return”/”utility” of the current state

→	→	→	+1
↑		↑	−1
↑	←	←	←

.812	.868	.912	+1
.762		.660	−1
.705	.655	.611	.388

Actions succeed with probability 0.8 and *move at right angles with probability 0.1* (remain in the same position when there is a wall). Actions incur a small cost (0.04).

- What happens when cost increases?
- Why move from .611 to .655 instead of .660?

Policies for different $R(s)$

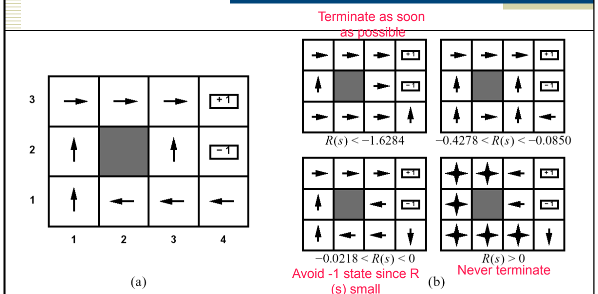


Figure 17.2 (a) An optimal policy for the stochastic environment with $R(s) = -0.04$ in the nonterminal states. (b) Optimal policies for four different ranges of $R(s)$.

Next Lecture

- Continuations with MDP
  - Value and policy iteration
- Search where is Uncertainty in Operator Outcome and Initial State
  - Partial Orderded MDP (POMDP)
- Hidden Markov Processes