Un
Uncertainty

Markov Decision Processes

CSE 415: Introduction to Artificial Intelligence
University of Washington
Winter, 2020

Outline

- Motivation
- Definitions
- "Unreliable Elevator" Example
- Representations
- Possible Applications

Motivation

Un
Uncertainty

We have seen that state-space search is a powerful methodology for problem solving and game playing.

But state changes in real-world situations don't always go as planned.

Nondeterminism can make our models more general.

We want to be able to take the expectimax idea from adversarial game playing and somehow apply it in problem solving.

Motivation (continued)

Therefore, we need a means to

- Take the state-space formalism, and
- add randomness to operator application.

- **M**arkov **D**ecision **P**rocesses give us this.

- The MDP formalism also adds one more idea – *rewards* for every move.

Definitions

A *Markov Decision Process* M = (S, A, T, R), consists of four components:

S is a set of states. (for us, finite, but not necessarily).

A is a set of actions. (These are essentially operators, with preconditions, but when applied, they may "choose" their next states randomly.)

T is a function that maps state-action-state triples to probabilities. T is called the "transition model" or the "dynamics" of the MDP. $T(s, a, s')$ tells what the probability of arriving in state s' is, given that an agent is in state s and takes action a. (This assumes it is legal to take action a in state s.)

R is a function that maps state-action-state triples to real numbers called rewards. R is called the reward function of the MDP. $R(s, a, s')$ gives the reward that an agent receives when making the transition from state s to state s' having chosen action a.

Example: An Unreliable Elevator

Consider an unreliable elevator that operates between 3 floors of a building: Ground, Parking 1, and Parking 2. At any point in time, it considers itself to be in one of these 3 states. (In this model, we are not considering traveling as a separate state; for us, traveling is sort of in the transition.) The table below shows T and R. Rows with disallowed actions or T(s,a,s')=0 are not shown.

$s \in S$	$a \in A$	$s' \in S$	$T(s,a,s')$	$R(s,a,s')$
Ground	Down	Ground	1/4	-2
Ground	Down	Parking 1	3/4	+1
Parking 1	Up	Ground	1/4	+2
Parking 1	Up	Parking 1	3/4	-2
Parking 1	Down	Parking 1	1/4	-2
Parking 1	Down	Parking 2	3/4	+1
Parking 2	Up	Parking 1	1/4	+2
Parking 2	Up	Parking 2	3/4	-1
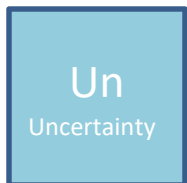
Elevator (continued)

Un
Uncertainty

The elevator usually goes down when the Down action is taken, but 25% of the times it stays at the same floor for 10 seconds.
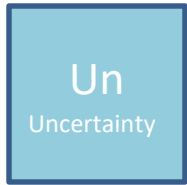When the Up action is taken, the elevator typically stays where it is for 10 seconds, but 25% of the times, it actually does go up.

Rewards are -2 whenever the elevator misbehaves and wastes time.
Rewards are +2 whenever the elevator goes up as requested.
Rewards are +1 whenever the elevator goes down as requested.

If you take one action from Ground, what is your expected reward for that action? If you decide to go from Ground to Parking 2 what might the total reward turn out to be? Note computing the expected total reward can sometimes be tricky, and we will introduce a special algorithm for it called Value Iteration.

Elevator (continued)

The elevator usually goes down when the Down action is taken, but 25% of the time it stays at the same floor for 10 seconds.
When the Up action is taken, the elevator typically stays where it is for 10 seconds, but 25% of the time, it actually does go up.

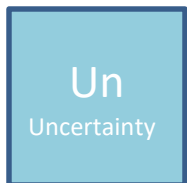Rewards are -2 whenever the elevator misbehaves and wastes time.
Rewards are +2 whenever the elevator goes up as requested.
Rewards are +1 whenever the elevator goes down as requested.

If you take one action from Ground, what is your expected reward for that action? If you decide to go from Ground to Parking 2 what might the total reward turn out to be? Note computing the expected total reward can sometimes be tricky, and we will introduce a special algorithm for it called Value Iteration.   (1/4) (-2) + (3/4) (+1) = + 1/4

Elevator (continued)

The elevator usually goes down when the Down action is taken, but 25% of the time it stays at the same floor for 10 seconds.
When the Up action is taken, the elevator typically stays where it is for 10 seconds, but 25% of the time, it actually does go up.

Rewards are -2 whenever the elevator misbehaves and wastes time.
Rewards are +2 whenever the elevator goes up as requested.
Rewards are +1 whenever the elevator goes down as requested.

If you take one action from Ground, what is your expected reward for that action? If you decide to go from Ground to Parking 2 what might the total reward turn out to be? Note computing the expected total reward can sometimes be tricky, and we will introduce a special algorithm for it called Value Iteration.   **If you are lucky and quick, it might be +2. It could be a lot less, if you are unlucky. But it could also be a lot more, if you decide to ride up and down a lot, and the elevator doesn't stall at all.**

What Do We Get From an MDP?

An enumeration of possibilities – the state space.
A definite specification of how actions can stochastically change the current state.
A clear reward structure for moving through the space.

A framework in which to design algorithms that can optimize choices of actions, predict reward totals, and that can learn about the T and R functions, if they are not explicitly given (i.e., machine learning).
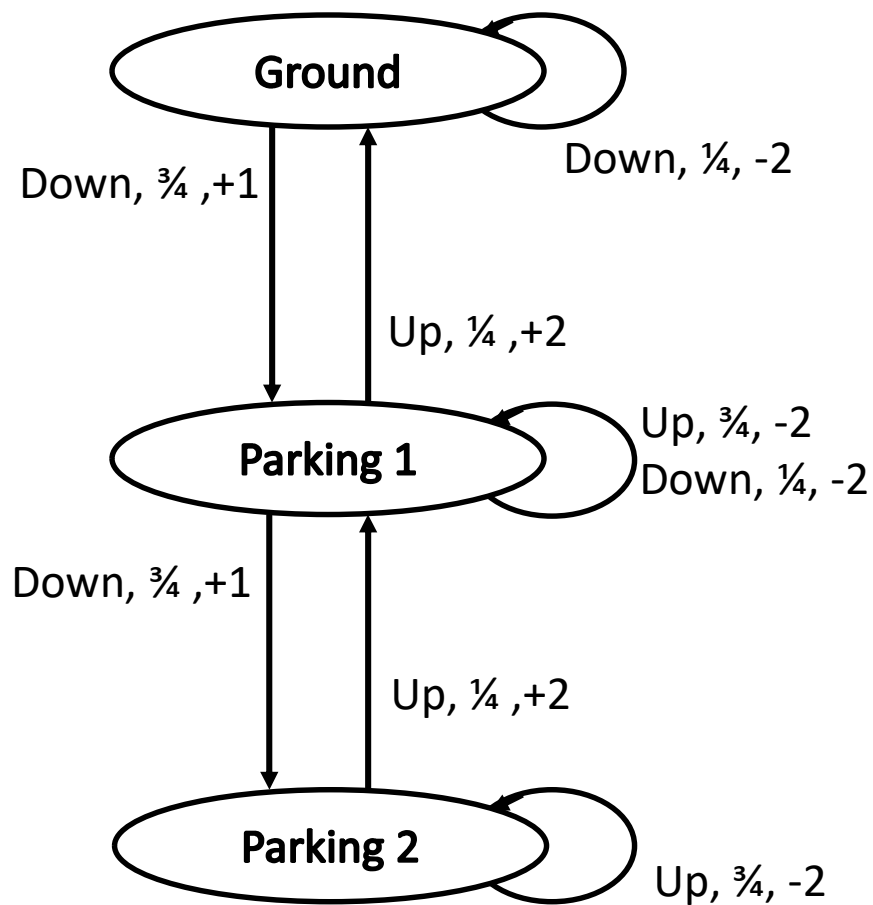
MDP Representations

Tabular: S, A, T, and R are given in one or more tables. (as in the Unreliable Elevator example, which used 1 table.)
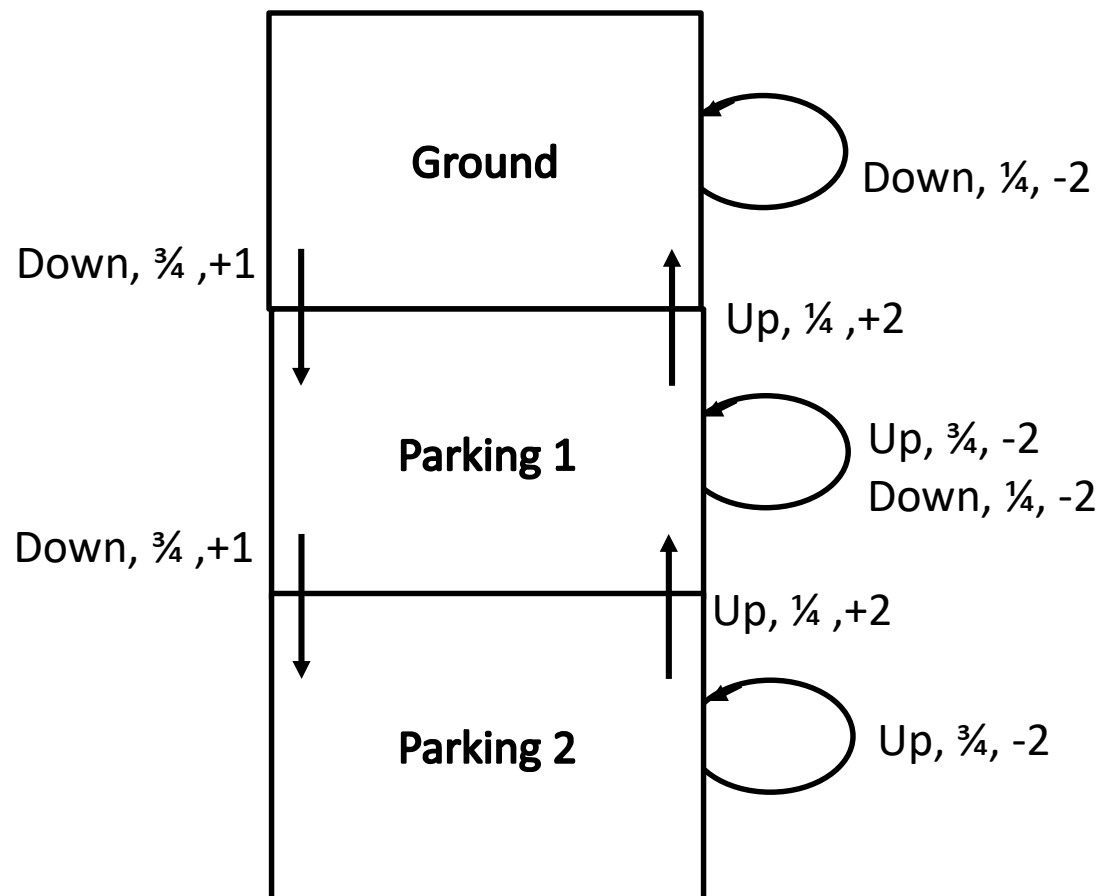
State Transition Diagram.
  (a) Simple type: Each state gets a node (and there are no other nodes)
       Each allowed transition (s, a, s') gets an edge, and
       the edge is labeled with a, T(s,a,s') and R(s,a,s').
  (b) Grid-style simple type. As in (a) but nodes are in abutting rectangles.
  (c) Type with nodes for both states and "q-states".
       Each state s gets a node.
       If a is an allowed action for state s, then (s,a) is called a q-state,
         and it also gets a node.
       There is an edge from each state node to each q-state node,
         corresponding to an allowed action for that state,
        and, there is an edge from each q-state node (s,a) to state node s'
         labeled with T(s,a,s') and R(s,a,s'), provided T(s,a,s') > 0.

Simple State Transition Graph

Ground

Down, ¾ ,+1

Down, ¼, -2

Up, ¼ ,+2

Parking 1

Up, ¾, -2
Down, ¼, -2

Down, ¾ ,+1

Up, ¼ ,+2

Parking 2

Up, ¾, -2

Grid-Style Simple State Transition Graph

Possible MDP Applications

- Modeling machines that can malfunction.
- Modeling robot motion through an environment.
- Modeling the progress of an agent solving a problem.
- Modeling a complex process that involves uncertain outcomes, such as social policies or economic policies that affect crime rates or the economy.
- Modeling users as they navigate a commercial website.
- Modeling security states of software systems.