ASEN 5264 Decision Making under Uncertainty
Homework 2: Markov Decision Processes

February 1, 2026

A submission should consist of two or three files:
- A single PDF file containing answers to questions (typed, handwritten, or exported notebook).
- JSON file output from `HW2.evaluate`.
- A code listing *if the code is not included in the PDF.*

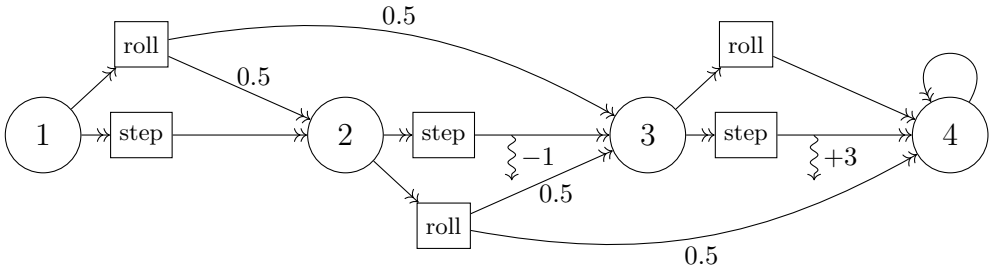
Conceptual Questions

Question 1. (5 pts)

a) Describe the difference between the reward function and the state-action (Q) value function.

b) Write down an equation for the state-action value in terms of R, T, and V.

Question 2. (15 pts) Consider a game with 4 squares (enumerated 1-4 from left to right) in a horizontal line drawn on paper, a token, and a die. Each turn, the player can either roll the die or step to the next square. If the player rolls and the die shows an odd number, the token is moved one square to the right, and if an even number is rolled, the token is moved two squares to the right. The game ends when the token is on the 4th square. If the player steps when the token is on square 2, one point is subtracted; if the player steps when the token is on square 3, three points are added.



a) Formulate this problem as an MDP (write down S, A, T, and R).

b) Use the Bellman backup algorithm to determine the optimal policy for this game, assuming a discount factor $\gamma = 0.95$?

Question 3. (15 pts) Consider a game with 3 squares in a horizontal line drawn on paper, a token, and a die. Each turn, the player can either reset or roll the die. The token always starts in the leftmost square. If the player rolls and the die shows an odd number, the token is moved one square to the right, and if an even number is rolled, the token is moved two squares to the right (in both cases stopping at the rightmost

square[1]). If the player resets, the token is always moved to the leftmost square. If the reset occurs when the token is in the middle square, two points are added; if the player resets when the token is on the right square, a point is subtracted.

a) Formulate this problem as an MDP (write down S, A, T, and R). Assume a discount of $\gamma = 0.95$ and that the die is fair.

b) Evaluate the following policy assuming a fair die:

$$\pi(s) = \begin{cases} roll & \text{if } s = L \\ reset & \text{if } s = M \\ reset & \text{if } s = R \end{cases}$$

c) Suppose you are not sure that the die is fair (i.e. whether it will yield odd and even with equal probability). Give finite upper and lower bounds for the accumulated discounted score that you can expect to receive with discount $\gamma = 0.95$.

Exercise

Question 4. (Value iteration for Grid World, 35 pts)

Solve the MDP `HW2.grid_world` with your own implementation of value iteration with a discount of $\gamma = 0.95$ and plot the resulting value function. All of the necessary information to solve this problem can be extracted with the `HW2.transition_matrices` and `HW2.reward_vectors` functions, and plotting can be accomplished with `POMDPTools.render(HW2.grid_world, color=V)` where `v` is the value function. See the starter code and function docstrings for more information.

Challenge Problem

Question 5. (Value iteration for ACAS, 30 pts)

Your task is to find the optimal value function for an Aircraft Collision Avoidance System (ACAS). The encounter model will be specified as a Markov decision process, and your task will be to compute the value function for discount $\gamma = 0.99$ using value iteration or another suitable algorithm that you implement. The continuous physical state space can be discretized at different levels of granularity, and the goal is to find the value function for the finest discretization possible.

A model with discretization level `n` can be constructed with

```
m = HW2.UnresponsiveACASMDP(n)
```

The higher `n` is, the finer the discretization and the larger the state space. Again, all of the information needed to solve this problem can be extracted with the `HW2.transition_matrices` and `HW2.reward_vectors` functions, so you can start with your code from Question 4.

The score received for solving the problem is `n`. You must submit your code for this problem along with the `results.json` file from executing `HW2.evaluate(v, "email@colorado.edu")` where `v` is the value function vector[2]. A score of `n` = 7 or higher will receive full credit[3].

[1] If the die is rolled from the middle square or right square, it will always end up in the right square.

[2] `HW2.evaluate` will check the value function with a tolerance of 1×10^{-6}. However, the probabilities and rewards contain some small numbers, so results can vary slightly depending on the way these are added and multiplied. Therefore we recommend using a tolerance of 1×10^{-8}.

[3] Hints: If you run out of memory, consider using the `sparse` keyword argument mentioned in the docstring of `transition_matrices`. By taking advantage of the structure of the problem, it is possible to attain a score of `n` = 20 with less than 10 minutes of computation time on a single core of a i7 laptop processor.