# Dynamic Distributed Decision Making
# Project 1
# MIE567

Hao Tan 999735728
Xiali Wu 999011322
David Molina

University of Toronto — February 10, 2020

## 1 Modelling

First, you are tasked with modelling the Gridworld domain above as a Markov decision process. Then, you are asked to provide a complete programming description of the problem that will be used to solve it computationally.

**1** Explain how you would model this navigation problem as a Markov decision process. In particular:

**a)** Why is this problem an MDP?

This problem can be modeled as an MDP because it has a discrete state space (cells) and a set of actions (north, east, west, south)/(up, right, down, left), where the transition from one state to another is based on a given action.

We can model this problem in terms of components such as a reward function,actions, transition probability, states, time steps, discount factor. All these components describe a MDP type of question.This problem is infinite horizon MDP since we can take unlimited amounts of steps between cells, and possibly revisit the same cell. Unlimited amount steps are definitely not optimal , but can potentially happen to max reward. Therefore, for infinite horizon MDP, a discount factor accounts for future reward.

Based on the definition of Markovian property, a stochastic process has the Markov property if the conditional probability distribution of future states of the process (conditional on both past and present states) depends only upon the present state, not on the sequence of events that preceded it (ie. P[St+1 | St] = P[St+1 | S1, S2 ... St]). In the given problem description, since the agent can only move one cell for one action, the next cell is only dependent on the current cell; the next cell is not depending on all previous cell before the current cell.

**b)** What are suitable state and action spaces for this problem? Are these the only possible choices? Why or why not?

**c)** What is the transition probability matrix P? (You may describe just the non-zero entries.)

**d)** Is the reward function provided the only possible one? If so, explain why. If not, provide an example of a different reward function that would lead to the same optimal behaviour

**e)** Derive the discounted Bellman equation for the problem, and simplify as much as you can. (Hint: to avoid deriving a separate value for each state, try to find groups of states such that you can write a single expression for V for them) What do you think is/are the optimal policy/policies for this problem, and why (you do NOT need to solve the Bellman equations)?

**2** Now, in a Python file called Gridworld.py, create a class that replicates the behaviour of the MDP you formulated in the previous question. Your class should contain four functions: one to return the initial state of the MDP, one to return a view of all possible states, and two to return, respectively, the reward and probability of a transition (s; a; s0) from state s to state s0 when taking action a.

Listing 1: Function 1:

```
def initial_state(self):
    # randomly generate an initial state
    i = random.randint(0, len(self.states)−1)
    rand_state = self.states[i]
    return rand_state
```

Listing 2: Function 2:

```
def possible_states(self):
    # return the possible states
    return self.states
```

Listing 3: Function 3:

```
def reward(self, current_pos, action):
    # take action in current pos
    self.new_pos = np.array(current_pos) + np.array(action)
    # normally, reward = 0
    reward = 0
    # if new pos results in off the grid, return reward −1
    if −1 in self.new_pos or self.size in self.new_pos:
        reward = −1
    # if in state A, transition to state A'
    if current_pos == [0, 1]:
        reward = 10
    # if in state B, transition to state B'
    if current_pos == [0, 3]:
        reward = 5
    return reward
```

## 2 Policy Evaluation

Now, suppose the agent selects all four actions with equal probability. Use your answers to questions 1 and 2 to write a python function, in a new file policy evaluation.py to find the value function for this policy. You may use either an iterative method or solve the system of equations. Show the value function you obtained to at least four decimals.

## 3 Value Iteration

The goal now is to solve the MDP above using dynamic programming. In a separate file called value iteration.py, provide a complete implementation of the value iteration algorithm you learned in class for solving the Bellman equations you derived earlier.

## 4 Policy Iteration

Repeat the previous part of the assignment, but now implement the policy iteration algorithm from class in a file called policy iteration.py. Report your results and answer all questions as in the previous part.

## 5 Comparison of Algorithms

In the final section of your report, you must compare the two algorithms you implemented and their performance on the Gridworld domain, and comment on any differences you observed. In particular, please answer at least the following questions in your report: