Homework 1: Markov Processes and Dynamic Programming

Collaboration in the sense of discussion is allowed, however, the work you turn in should be your own - you should not split parts of the assignments with other students and you should certainly not copy other students' code or papers. See the collaboration and academic integrity statement here: https://natanaso.github.io/ece276b. *Books may be consulted but not copied from.*

Submission

You should submit the following four files on **Gradescope** by the deadline shown on the top right corner.

1. **FirstnameLastname_HW1_P1-3.pdf**: upload your solutions to Problems 1-3. You may use latex, scanned handwritten notes (write legibly!), or any other method to prepare a pdf file. Do not just write the final result. Present your work in detail, explaining your approach at every step.
2. **FirstnameLastname_HW1_P4.zip**: upload the code you have written for Problem 4 in a zip file with the following strucutre:

   ```
   src/
   main.py
   README.txt
   ```

   The src directory should contain all files necessary to run your code. The README file should contain information about the python version and dependencies needed. The main.py file should run your program with the following syntax: "python main.py input.txt output.txt".
3. **FirstnameLastname_HW1_P5.zip**: upload the code you have written for Problem 5 in a zip file with the same structure as above.
4. **FirstnameLastname_HW1_P6.pdf**: upload the report for Problem 6. You are encouraged but not required to use an IEEE conference template[1] for your report.

Problems

1. [10 pts] USS Enterprise recently discovered a new planet with very strange seasonal changes as a result of its multiple suns. The planet has three seasons: *freezing*, *modest* and *burning*. Every season lasts roughly three months. The climatologist onboard reported that if the current season is *burning*, then it is equally likely to be *burning* and *modest* for the next season; if the current season is *modest*, half the time it is *modest* in three months, and if it changes it is just as likely to become *freezing* and *burning*; if the current season is *freezing* it has 2/3 probability of remaining so and 1/3 of becoming *modest*. We will model these seasonal changes via a Markov chain.
   (a) Write down the state transition matrix. Draw the state transition diagram (like the ones shown in the lecture).
   (b) Is the Markov chain periodic? If so, what is its period? If not, why not?
   (c) Suppose the starship enters the planet and the crew finds it *freezing*. If the crew plans to leave at the same time next year, what is the probability that the planet is in *freezing* at that time? How about *modest* and *burning*?
   (d) Suppose during the crew meeting that sets the departure plan, the onboard climatologist requests to stay until he can collect data in all three seasons. What is the probability that the ship cannot leave as scheduled?

[1] https://www.ieee.org/conferences_events/conferences/publishing/templates.html

2. [15 pts] Consider a system with the following motion model:

$$x_{t+1} = x_t + u_t + w_t$$

It has initial state $x_0 = 2$, stage cost $g(x_t, u_t) = x_t^2 + u_t^2$ and terminal cost $g_T(x_T) = x_T^2$ for horizon $T = 3$. The space of control inputs is $\mathcal{U} = \{-1, 1\}$. Answer the following questions:

   (a) Consider the case when $w_t = 0$ for all t. Identify an optimal open-loop policy and compute its associated expected cost.

   (b) Consider the case when w_t takes the values -1 and 1 with equal probability $\frac{1}{2}$. As usual, the noise terms are conditionally independent given the state. Identify an optimal open-loop policy and compute the associated expected cost. (Hint: Use the Markov property.)

   (c) Consider closed-loop policies for the noise model in (a) and (b). Use dynamic programming to find the optimal policies and compute the associated expected costs. Are the policies stationary?

   (d) Are the costs from open-loop and closed-loop policies the same? If not, why are they different?

3. [10 pts] You are controlling a (linear) system by selecting its modes of operation. For two modes, that is,

$$x_{t+1} = \begin{cases} U_1 x_t, & \text{if } u_t = 1, \\ U_2 x_t, & \text{if } u_t = 2 \end{cases}$$

where $x_t \in \mathbb{R}^n$ is the current state, and $U_1, U_2 \in \mathbb{R}^{n \times n}$ are two given matrices. Your costs at times $t = 0, \dots, T$ are $c_t = x_t^T Q x_t$ for some positive definite matrix Q. This setup has applications, for example, in sensor scheduling, or in embedded control systems with limited computation and communication resources.

   (a) Use dynamic programming to show that the optimal cost-to-go/value function $J_t^\star(x)$ of the problem is the minimum of 2^{T-t} positive definite quadratic functions. Describe the optimal policy $\pi_t^\star$ using these functions.

   (b) Consider the $T = 4$ horizon problem with matrices $U_1 = \begin{bmatrix} 0.5 & -1 \\ 1 & 0.5 \end{bmatrix}$, $U_2 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$ and Q the identity matrix. Plot the regions of the state space where it is optimal to select action 1 and 2 respectively at time $t = 0, \dots, 3$, for example on a discretized $[-1, 1] \times [-1, 1]$ square around the origin. Also plot the optimal value function $J_0^\star(x)$ on that space. Justify your plots through mathematical analysis.

4. [25 pts] **Programming Assignment** You are playing the game of tic-tac-toe against a not so good player. You are playing first. The opponent plays next and chooses uniformly randomly from all available positions. Then you play next, the opponent randomizes, and so on, until either you win, or you lose, or you tie. Suppose your goal is to find a policy to maximize the probability of winning, and you are indifferent between tie or losing. (Hint: Be reminded that for any event A, $\mathbb{P}(A) = \mathbb{E}\{\mathbb{1}(A)\}$.)

   (a) What is the maximum probability of winning? Formulate this as a finite-horizon dynamic programming problem and solve it numerically. Roughly, the number of states is all possible board arrangements, i.e., any of the 3×3 positions could be empty, occupied by you, or by the opponent, which makes 3^9 combinations (some of them will never occur but you do not have to go into the trouble of excluding them). The horizon can be 10, i.e., at most 9 plays for you and the opponent and at time 10 the outcome is decided.

   (b) Suppose now you care about losing too, so your goal is to maximize the probability of winning minus the probability of losing. What is the optimal such value? Is the optimal policy the same as before?

   (c) Suppose now the opponent plays first, and you play second. What are the optimal values for the settings in (a) and (b)?

   (d) Simulate the winning probability of the policies, and verify that the actual probabilities are close to the numbers you compute using dynamic programming.

Your code should save a `output_ttt.txt` file with four lines. The four lines should be: (1) Maximum probability of winning when you play first. (2) Maximum probability of winning minus the probability of losing when you play first. (3) Maximum probability of winning when your opponent plays first. (4) Maximum probability of winning minus the probability of losing when your opponent plays first. On each line there are two numbers separated by a white space. The first number is the value computed from dynamic programming, and the second number is the value from your simulation. In your report illustrate in tabular forms the policies for your first move (index the 3x3 grid by $1-9$, following a left-to-right, top-to-bottom order) and discuss your intuition.

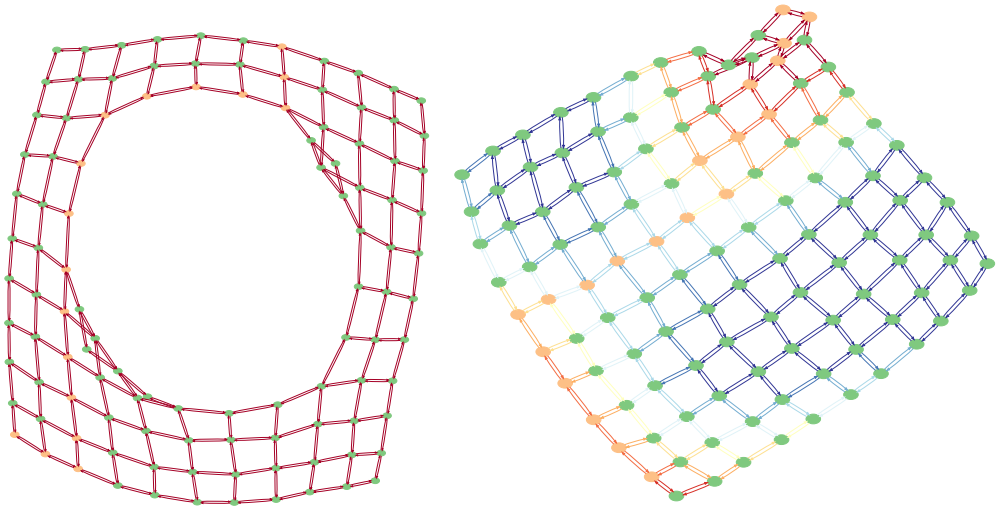
5. [25 pts] **Programming Assignment** Implement the dynamic programming algorithm to solve the minimum cost path planning problem on a given graph. The input to your algorithm should be a file called `input.txt`. This file has a specific format as given below:

   - The first line of the input file gives the total number of vertices, n, in the graph.
   - The second line gives the starting vertex index (indices go from 1 to n).
   - The third line gives the goal vertex index (indices go from 1 to n).
   - Starting from the fourth line, we have the edges in the graph specified in the form of an edge list. Each line specifies one edge: `i j wij` which indicates that there is a (directed) edge from `i` to `j` with a cost of `wij`. Note that this is a directed graph and the weights are always non-negative.

   Your code must produce a file called `output_dsp.txt`. This file must contain the vertices along the minimum cost path as well as the optimal value function. The output file must have the following format:

   - The first line must list the indices of the vertices on the shortest path – from the start to the goal vertex.
   - The second line must list all the optimal value functions, i.e., $J(x_0^\star), J(x_1^\star), \ldots, J(x_n^\star)$.

   Two sample input and output files are provided (also shown in Figure 1). The figures were generated using the provided plotting tool `vis_graph.py`. The `graphviz` package is required for this tool. Note that your code will be tested on instances other than the two input files. Make sure you follow the input/output conventions exactly.



(a) `input1.txt` (b) `input2.txt`

Figure 1: The two sample instances. The input is a directed graph given in the `input1.txt` and `input2.txt` files. The color of the edges indicates the cost (bluer edges have higher costs than redder edges). The minimum cost path from start to goal is shown in orange.

6. [15 pts] Write a project report describing your approach to the tic-tac-toe and the minimum cost path planning problem. Your report should include the following sections:
   - **Introduction**: present a brief overview of your approach
   - **Problem Formulation**: state the problem you are trying to solve in mathematical terms. This section should be short and clear and should define the quantities you are interested in. In particular, you should include the basic elements of the MDP problems you try to solve.
   - **Technical Approach**: describe your approach to finding optimal policies and computing the expected loss. Write down the dynamic programming equations.
   - **Results**: present the details of your simulation and your results and discuss them – what worked, what did not, and why.