EE365 Homework 3

1. *Managing a data center.* You are the manager of a data center offering a particular service to customers (*e.g.*, computing power, file retrieval, serving web pages). In this problem we consider a very simple model with only one server.

   At each time t, the server receives a number of job requests w_t that is a random variable with a Poisson distribution with parameter $\lambda = 2$. As jobs arrive, they are placed in a buffer. Jobs can be processed in the same period that they arrive, but the buffer can hold at most 25 jobs, and any additional jobs in the buffer at the end of a time period are dropped.

   The server has three modes of operation: `slow`, `normal`, and `fast`. The numbers of jobs that can be processed in a time period for the different modes of operation are 1, 2 and 3, respectively, and the costs in dollars per time step are 1, 3.5, and 5.75, respectively. These costs are incurred even when the buffer is empty, as one cannot shut down the server at such times. (Note that we may end up processing fewer jobs in a time period than we are capable of processing in that time period; for example, if there are no jobs in the buffer, and no new jobs arrive, then we will not process any jobs.)

   At each time t, the server examines the buffer before any new jobs arrive, and decides on the mode of operation based on x_t, the number of jobs in the buffer. The policy for choosing the mode of operation is as follows. If $x_t \leq 15$, operate the server in `slow` mode; if $15 < x_t \leq 22$, operate the server in `normal` mode; and if $x_t > 22$, operate the server in `fast` mode. The mode of operation at time t is denoted u_t.

   The data center is paid \$2 for every job processed. The profit of the data center is the amount the data center is paid for serving jobs minus the amount the data center pays to operate the server. Initially the buffer is empty: *i.e.*, $x_0 = 0$.

   You must compute the requested quantities exactly (i.e., not using Monte Carlo). Of course, you may wish to check your answers using Monte Carlo (and this is highly recommended). As always you must submit all code used in your solution.

   (a) Having fixed the policy, (x_t) is a Markov chain. Find the transition matrix P for this Markov chain. Print out, and turn in `round(10*P)`.

   (b) Find a function $g_t(x, u, w)$ that gives the profit in time period t if there are x jobs in the buffer at the beginning of the period, the server operates in mode u, and w new jobs arrive.

   (c) Simulate one trajectory of the system for $t = 0, \ldots, T$, where $T = 100$. Plot x and w as functions of time.

   (d) Fix some large value of the time horizon T, and let v_t be the value function at time t: *i.e.*, the expected profit over the time segment $t, \ldots, T$. Plot the value function $v_0(x)$ vs. x. You should observe that

   $$v_0 - v_1 \approx \alpha \mathbf{1}$$

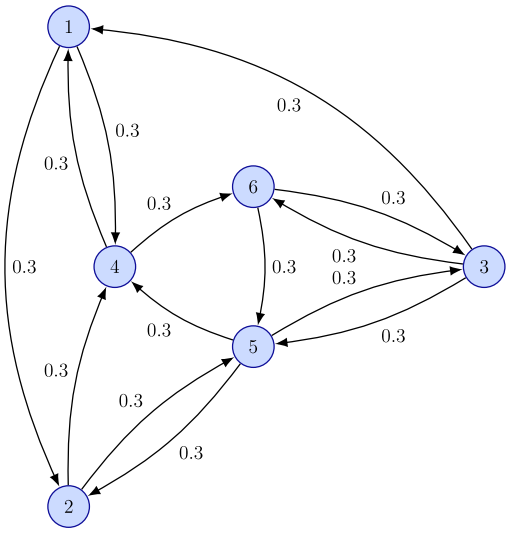
for some $\alpha \in \mathbb{R}$. In other words, the difference between the value functions at times 0 and 1 is approximately constant across states. Explain why this approximate relation holds, and give an interpretation of α.

(e) What is the mean fraction of time (for large T) that the server spends in each of the three operating modes?

(f) What is the mean profit per time step (for large T)?

(g) Now suppose that the data center is penalized \$10 for each job that is dropped. What is the mean profit per time step (for large T) for this modified problem?

2. *Second passage time.* In this problem we will consider the following Markov chain.



Note that self-loops are omitted from this figure. The transition matrix for this chain is

$$P = \begin{bmatrix} 0.4 & 0.3 & 0 & 0.3 & 0 & 0 \\ 0 & 0.4 & 0 & 0.3 & 0.3 & 0 \\ 0.3 & 0 & 0.1 & 0 & 0.3 & 0.3 \\ 0.3 & 0 & 0 & 0.4 & 0 & 0.3 \\ 0 & 0.3 & 0.3 & 0.3 & 0.1 & 0 \\ 0 & 0 & 0.3 & 0 & 0.3 & 0.4 \end{bmatrix}.$$

This matrix is given in `passage_time_data.m` so you don't have to type it into `MATLAB`. We will compute several things about this Markov chain. Let the destination set be the single state $E = \{1\}$, and let the initial state be $i = 2$. Recall that the first passage time to the set E is

$$\tau_E = \min\{t > 0 \,|\, x_t \in E\}$$

The kth passage time for a Markov chain is defined recursively as

$$\tau_E^{(k)} = \min\{t > \tau_E^{(k-1)} \,|\, x_t \in E\},$$

where $\tau_E^{(1)} = \tau_E$.

(a) For the Markov chain above, compute and plot the distribution of the first passage time conditioned on the initial state, *i.e.*, compute

$$f(t) = \mathbf{Prob}(\tau_E = t \,|\, x_0 = i)$$

as a function of t.

(b) One may calculate the distribution of the second passage time by constructing a new Markov chain, whose first passage time is equal to the second passage time of the original chain. What is the transition graph of this chain?

(c) For $t = 1, 2, \ldots$, use your construction to compute the distribution of the second passage time, which is

$$s(t) = \mathbf{Prob}(\tau_E^{(2)} = t \,|\, x_0 = i).$$

Use your method to compute and plot this distribution also.

(d) Plot the sum of $s(t)$ and $f(t)$, and also plot $\mathbf{Prob}(x_t \in E)$. Explain what you see.

(e) For a given state j, the *recurrence time* is the first passage time to the state j given the initial condition $x_0 = j$. This has distribution

$$r(t) = \mathbf{Prob}(\tau_{\{j\}} = t \,|\, x_0 = j).$$

Let $j = 1$. Explain how to compute this, compute it, and plot it.

(f) Show that $s = f * r$, where $*$ denotes convolution. Verify this numerically, and interpret this result.

(g) Give a method to compute the distribution of the kth passage time, for any k. How does your method scale with k?

3. *Class decomposition of a Markov chain.* In lecture we claimed that the states of a Markov chain can be ordered so that the probability-transition matrix has the form

$$P = \begin{bmatrix} P_{11} & P_{12} \\ 0 & P_{22} \end{bmatrix},$$

where P_{11} is block upper triangular with irreducible blocks on the diagonal, and P_{22} is block diagonal with irreducible blocks. Each of the blocks on the diagonal of P_{11} represents a transient class, while each of the blocks on the diagonal of P_{22} represents a recurrent class. In this problem you will write `MATLAB` code to find an ordering of the states that puts P in this form. We will use standard graph-theory terminology throughout the problem, so please consult Wikipedia if you encounter any unfamiliar concepts. The file `class_decomposition_data.m` defines a specific probability-transition matrix that you should use throughout this problem. However, your code must work for any probability-transition matrix.

(a) Let G be a graph with adjacency matrix $A \in \mathbb{R}^{n \times n}$. We write $i \to j$ if there is a path from node i to node j in G. We define the reachability matrix $R \in \mathbb{R}^{n \times n}$ such that

$$R_{ij} = \begin{cases} 1 & i \to j, \\ 0 & \text{otherwise.} \end{cases}$$

```
R = A /* A is the adjacency matrix                                    */
repeat
    for i = 1 to n do
        /* iterate over all states k we know are reachable from i     */
        for k such that R_ik = 1 do
            /* iterate over all states j we know are reachable from k */
            for j such that R_kj = 1 do
                /* j is reachable from i through k                    */
                R_ij = 1
            end
        end
    end
until no changes are made to R_ij
```

Algorithm 1: Computing the reachability matrix

An algorithm for constructing R is given in algorithm 1.

Write a `MATLAB` function that implements algorithm 1; use the following function header.

```
function R = reachable_states(A)
```

For reference, our implementation of `reachable_states` is less than twenty lines. *Hint.* The function `find` may be useful.

(b) We write $i \leftrightarrow j$ if $i \to j$ and $j \to i$, and we define the communication matrix $C \in \mathbb{R}^{n \times n}$ such that

$$C_{ij} = \begin{cases} 1 & i = j \text{ or } i \leftrightarrow j, \\ 0 & \text{otherwise.} \end{cases}$$

Explain how to construct the communication matrix from the reachability matrix.

(c) We define the transience vector $t \in \mathbb{R}^n$ such that

$$t_i = \begin{cases} 1 & i \text{ is a transient state,} \\ 0 & \text{otherwise.} \end{cases}$$

Explain how to construct the transience vector from the communication and reachability matrices.

(d) Implement one of the algorithms given on the Wikipedia page for "topological sorting." use the following function header.

```
function L = topological_sort(A)
```

For reference, our implementation of `topological_sort` is less than twenty lines.

(e) Suppose that duplicate rows of C have been removed, and the rows have been sorted so that the first n_t rows represent the transient classes. (You need to write code to do this; the commands `unique` and `sort` may be useful.) Note

that there is now a unique row of C corresponding to each class. For two classes $C_i, C_j \subseteq \{1, \ldots, n\}$, we write $C_i \to C_j$ if $x \to y$ for some $x \in C_i$ and $y \in C_j$. The adjacency matrix $A_t \in \mathbb{R}^{n_t \times n_t}$ for the set of transient classes is defined such that

$$(A_t)_{ij} = \begin{cases} 1 & C_i \to C_j, \\ 0 & \text{otherwise.} \end{cases}$$

Explain how to use the reachability matrix R of the Markov chain to find the adjacency matrix A_t for the set of transient classes.

(f) You should now have all the tools you need to construct an ordering of the states that puts the probability transition matrix in the desired form. Apply your method to the matrix in `class_decomposition.data.m`. Attach a plot of calling `spy` on your reordered matrix (the data file generates a plot of the original matrix). For reference, our solution has thirty-five lines (not including `reachable_states` and `topological_sort`).

4. *Markov web surfing model.* A set of n web pages labeled $1, \ldots, n$ contain (directed) links to other pages. We define the link matrix $L \in \mathbb{R}^{n \times n}$ as

$$L_{ij} = \begin{cases} 1 & \text{if page } i \text{ links to page } j \\ 0 & \text{otherwise.} \end{cases}$$

We define $o \in \mathbb{R}^n$ as $o = L\mathbf{1}$, which gives the number of outgoing links from each page.

A very crude model of a web surfer is a Markov chain on the pages, with transitions described as follows. The model includes a parameter $\theta \in (0, 1)$, which (roughly) gives the probability that the surfer follows a link from the current page. For a page with $o_i > 0$ (*i.e.*, with at least one outgoing link) the surfer moves to each of the linked-to pages with probability θ/o_i, and jumps to a page not linked to i with probability $(1 - \theta)/(n - o_i - 1)$. For a page with no outgoing links (*i.e.*, $o_i = 0$) the surfer jumps to a random page, chosen from a uniform distribution on (the other) pages.

We will assume that web surfer starts at a random page, uniformly distributed.

We earn a payment when the surfer follows (*i.e.*, clicks on) a link, given by $R_{ij} \geq 0$. This payment matrix satisfies $R_{ij} = 0$ when $L_{ij} = 0$ (*i.e.*, we are not paid for random jumps; only following links).

The following questions concern the specific instance of the problem with data given in `link_matrix_data.m`.

(a) What is the most likely page the surfer is on, at time $t = 10$? at $t = 100$?

(b) Let J denote the expected total payment over $t = 0, \ldots, 50$. Compute J three ways:

- Monte Carlo simulation (which gives an estimate of J, not the exact value).
- Distribution propagation.
- Value iteration.

Be sure to check that the values are consistent.

Remark. The Markov model described in this problem leads to Google's famous PageRank, which corresponds to the fraction of time spent at each site, when $T \to \infty$. (The current version of PageRank is based on far more than just the link topology, but the first versions really did make heavy use of the Markov surfing model.)