Solutions to Homework 6 - Markov Chains

1) Discrete-time queueing model for a service station. Suppose customers can arrive to a service station at times $n = 0, 1, 2, \ldots$ In any given period, independent of everything else, there is one arrival with probability p, and there is no arrival with probability $1 - p$. Suppose customers are served one-at-a-time on a first-come-first-served basis. If at the time of an arrival, there are no customers present, then the arriving customer immediately enters service. Otherwise, the arrival joins the back of the queue.

In a time period n, events happen in the following order: (i) arrivals, if any, occur; (ii) service completions, if any, occur; and (iii) service begins on a new customer if there has been an arrival to an empty queue or a service has just finished and there is another customer present.

Assume that service times are i.i.d. geometric RVs (each with parameter q) that are independent of the arrival process. Note that a customer who enters service in time n can complete service, at the earliest, in time $n + 1$ (in which case his service time is 1). Let X_n denote the number of customers at the station at the end of time period n; i.e., after the time-n arrivals and services. Note that X_n includes both customers waiting as well as any customer being served.

A) Let Y be a geometric-distributed RV, with parameter q. We want to show that $\mathrm{P}\left[Y = i \,\middle|\, Y \geq i\right] = q$ for $i = 1, 2, \ldots$. From the definition of conditional probability we obtain

$$\mathrm{P}\left[Y = i \,\middle|\, Y \geq i\right] = \frac{\mathrm{P}\left[Y = i, Y \geq i\right]}{\mathrm{P}\left[Y \geq i\right]} = \frac{\mathrm{P}\left[Y = i\right]}{\mathrm{P}\left[Y \geq i\right]} = \frac{\mathrm{P}\left[Y = i\right]}{1 - \mathrm{P}\left[Y < i\right]} = \frac{(1-q)^{i-1}q}{1 - \sum_{k=1}^{i-1}(1-q)^{k-1}q} \tag{1}$$

where in obtaining the last equality we have substituted the expression for the pmf of a geometric RV. To simplify the denominator, just sum the finite geometric series

$$1 - \sum_{k=1}^{i-1}(1-q)^{k-1}q = 1 - \frac{q}{1-q}\sum_{k=1}^{i-1}(1-q)^{k} = 1 - \frac{q}{1-q}\left(\frac{1-q-(1-q)^{i}}{q}\right) = (1-q)^{i-1}.$$

Substituting this result back in (1), we obtain

$$\mathrm{P}\left[Y = i \,\middle|\, Y \geq i\right] = q$$

which is what we wanted to show.

B) Service times are assumed i.i.d. geometric RVs with parameter q. Suppose that a particular customer begins service at time n, and does not complete service in periods $n + 1, n + 2$ or $n + 3$. If we let Y denote the service time, then we want to compute

$$\mathrm{P}\left[Y = (n+4) - n \,\middle|\, Y \neq (n+1) - n, Y \neq (n+2) - n, Y \neq (n+3) - n\right] = \mathrm{P}\left[Y = 4 \,\middle|\, Y \geq 4\right]$$

But this is exactly what we calculated in part *A*), and therefore it readily follows that

$$\mathrm{P}\left[Y = 4 \,\middle|\, Y \neq 1, Y \neq 2, Y \neq 3\right] = q$$

The conclusion in the context of the problem is that the probability that a user being served since time instant $n-1$ (or any other value in the past) completes service and exits the system at time n is equal to q. Note that this is the case conditioned on the information that we know that the user was still being served at time instant $n - 1$.

C) Given the current state, the next state is determined by whether or not we have an arrival and/or a service. The occurrence of an arrival has probability p and is independent of past events. Similarly, if the system is in state $i > 0$, there will be a service with probability q, conditionally independent of past events given the system is in state $i > 0$. In particular, because of the geometric service times it does not matter how long the customer currently being served has been in service.

More formally, let X_n be the number of customers at the station at the end of time period n, and let A_n be a Bernoulli-distributed RV with parameter p indicating the arrival of a customer at time n. Likewise, let D_n be a Bernoulli-distributed RV with parameter q indicating the departure of a customer that completed service at time n. Both $A_{\mathbb{N}}$ and $D_{\mathbb{N}}$ are i.i.d. and independent of each other. Also, for each n both A_n and D_n are independent of X_n.

(Notice that the fact that service times are geometric is crucial towards modeling departures as an i.i.d. sequence; see also the discussion at the end of the previous part.) From the problem description, we find that X_n satisfies the following recursion

$$X_n = X_{n-1} + A_n - \min(X_{n-1}, D_n).$$

Observe how the last term $\min(X_{n-1}, D_n)$ accounts for the fact that if $X_{n-1} = 0$ (the service station is empty), then no user is being served and consequently no user can eventually exit the system at time instant n. Because X_n obeys a recursion of the general form

$$X_n = f(X_{n-1}, A_n, D_n)$$

for the function $f(a, b, c) = a + b - \min(a, c)$, and the i.i.d. processes $A_{\mathbb{N}}$ and $D_{\mathbb{N}}$ (which are also independent of X_0), then we know that $X_{\mathbb{N}}$ is a MC.

D) The transition probabilities can be derived as follows. First we consider the case where the present state is $X_n = 0$, where the possibilities are:

1) If an arrival does not occur at time instant $n + 1$ then $X_{n+1} = 0$. Note that because $X_n = 0$, then it is also impossible that some user exits the system after being served because the system was empty. Then we conclude that $P_{00} = 1 - p$.
2) If an arrival occurs at time instant $n + 1$ then $X_{n+1} = 1$. Then we conclude that $P_{01} = p$.

All the rest of the transition probabilities from state 0 should be equal to zero, as they correspond to state transitions that cannot happen. The general case where the present state is $X_n = i > 0$, gives rise to the following possibilities:

1) If an arrival does not occur at time instant $n + 1$ and the user being served does not complete service, then $X_{n+1} = i$. This will happen with probability $(1 - p)(1 - q)$. Furthermore, if an arrival occurs at time instant $n+1$ and the user currently being served exits the system after service completion, then also $X_{n+1} = i$. This will happen with probability pq. The conclusion is that $P_{ii} = pq + (1 - p)(1 - q) = 1 + 2pq - p - q$.
2) If an arrival occurs at time instant $n+1$ and the user being served does not complete service, then $X_{n+1} = i+1$. Then we conclude that $P_{i,i+1} = p(1 - q)$.
3) If an arrival does not occur at time instant $n + 1$ and the user currently being served exits the system after service completion, then $X_{n+1} = i - 1$. Then we conclude that $P_{i,i-1} = (1 - p)q$.

All the rest of the transition probabilities will be zero. As a sanity check, note that $P_{ii} + P_{i,i-1} + P_{i,i+1} = 1$. The general form of the transition matrix is

$$\mathbf{P} = \begin{bmatrix} 1-p & p & 0 & 0 & \dots \\ (1-p)q & 1+2pq-p-q & p(1-q) & 0 & \dots \\ 0 & (1-p)q & 1+2pq-p-q & p(1-q) & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}$$

which has a banded tri-diagonal structure.

2) Probability of the instant when a state is visited for the first time. Suppose that $X_{\mathbb{N}} = X_0, X_1, \dots, X_n, \dots$ is a MC with state space S, and for $i, j \in S$ define

$$g_{ij}^1 = \mathrm{P}\left[X_1 = j \,\middle|\, X_0 = i\right],$$
$$g_{ij}^n = \mathrm{P}\left[X_n = j, X_{n-1} \neq j, \dots, X_2 \neq j, X_1 \neq j \,\middle|\, X_0 = i\right], \ n \geq 2.$$

A) To derive a recursion that relates g_{ij}^n to g_{ij}^{n-1}, start from g_{ij}^n and condition on the first transition to obtain

$$\begin{aligned} g_{ij}^n &= \mathrm{P}\left[X_n = j, X_{n-1} \neq j, \dots, X_1 \neq j \,\middle|\, X_0 = i\right] \\ &= \sum_{\substack{k \in S \\ k \neq j}} \mathrm{P}\left[X_n = j, X_{n-1} \neq j, \dots, X_1 = k \,\middle|\, X_0 = i\right] \\ &= \sum_{\substack{k \in S \\ k \neq j}} \mathrm{P}\left[X_n = j, X_{n-1} \neq j, \dots, X_2 \neq j \,\middle|\, X_1 = k, X_0 = i\right] \mathrm{P}\left[X_1 = k \,\middle|\, X_0 = i\right] \\ &= \sum_{\substack{k \in S \\ k \neq j}} \mathrm{P}\left[X_n = j, X_{n-1} \neq j, \dots, X_2 \neq j \,\middle|\, X_1 = k\right] g_{ik}^1 = \sum_{\substack{k \in S \\ k \neq j}} g_{kj}^{n-1} g_{ik}^1. \end{aligned}$$

Notice that the fourth equality follows from the Markov property, while the fifth equality is a consequence of the time-homogeneity of the MC so that $\mathrm{P}\left[X_n = j, X_{n-1} \neq j, \ldots, X_2 \neq j \,\middle|\, X_1 = k\right] = g_{kj}^{n-1}$.

B) The quantities g_{ii}^n represent the probabilities that starting from $X_0 = i$, the MC revisits i for the first time after exactly n transitions. Accordingly, it follows that the probability f_i that the MC ever revisits state i is given by

$$f_i = \mathrm{P}\left[\bigcup_{n=1}^{\infty} X_n = i \,\middle|\, X_0 = i\right] = \sum_{n=1}^{\infty} g_{ii}^n.$$

Hence, we have that state i in is recurrent if $f_i = \sum_{n=1}^{\infty} g_{ii}^n = 1$, and transient if $f_i = \sum_{n=1}^{\infty} g_{ii}^n < 1$.

3) Expected discounted costs. Suppose that $X_{\mathbb{N}} = X_0, X_1, \ldots, X_n, \ldots$ is a MC with state space S, and transition probabilities P_{ij} for $i, j \in S$. Suppose that $c(\cdot)$ is a cost function so that we are charged cost $c(i)$ when we are in state $i \in S$. Suppose that $0 < \lambda < 1$ is a discount factor. Let

$$v(k, i) = \mathbb{E}\left[\sum_{n=0}^{k} \lambda^n c(X_n) \,\middle|\, X_0 = i\right].$$

By first using the given conditioning on $X_0 = i$, and then conditioning on the first transition we obtain for $k \geq 0$

$$\begin{aligned}
v(k+1, i) &= \mathbb{E}\left[\sum_{n=0}^{k+1} \lambda^n c(X_n) \,\middle|\, X_0 = i\right] \\
&= \mathbb{E}\left[c(X_0) + \sum_{n=1}^{k+1} \lambda^n c(X_n) \,\middle|\, X_0 = i\right] = \mathbb{E}\left[c(i) + \sum_{n=1}^{k+1} \lambda^n c(X_n) \,\middle|\, X_0 = i\right] \\
&= c(i) + \sum_{j \in S} \mathbb{E}\left[\sum_{n=1}^{k+1} \lambda^n c(X_n) \,\middle|\, X_1 = j, X_0 = i\right] \mathrm{P}\left[X_1 = j \,\middle|\, X_0 = i\right] \\
&= c(i) + \lambda \sum_{j \in S} \mathbb{E}\left[\sum_{n=1}^{k+1} \lambda^{n-1} c(X_n) \,\middle|\, X_1 = j\right] P_{ij} \\
&= c(i) + \lambda \sum_{j \in S} P_{ij} v(k, j)
\end{aligned}$$

which is the desired result. In obtaining the last equality we used that $v(k, j) = \mathbb{E}\left[\sum_{n=1}^{k+1} \lambda^{n-1} c(X_n) \,\middle|\, X_1 = j\right]$, which follows from a change of variables $m = n - 1$.

4) Ranking of nodes in graphs.

A) Markov chain model. The random process of agent visits $A_{\mathbb{N}}$ is a MC because it satisfies the Markov property. Specifically, it follows from the description of the agent's actions that

$$\mathrm{P}\left[A_{n+1} = j \,\middle|\, A_n = i, \mathbf{A}_{n-1} = \mathbf{a}\right] = \mathrm{P}\left[A_{n+1} = j \,\middle|\, A_n = i\right] = \frac{1}{N_i}, \quad j \in n(i)$$

depends only on i and j, and is conditionally independent on the history of the process $\mathbf{A}_{n-1} = \mathbf{a}$.

Next, we give conditions under which the itemized statements are true:

- *State i, meaning visit of agent to node i, of this MC is transient.* Since the MC is finite (it has J states), the only possibility for a state i to be transient is that it can reach some other state j through a path of nonzero probabilities, but not vice versa. Hence assuming $n(i)$ is not empty, this could happen if there are no nodes in its incoming neighborhood $n^{-1}(i)$, or if the nodes in its incoming neighborhood have no incoming neighbors, and so forth.
- *All states of this MC are transient.* It is impossible that all states in a finite MC are transient.
- *All the states of this MC are recurrent.* All states in the MC are recurrent if the graph is strongly connected, meaning that there is a directed path from each node to each other node. This way, all states communicate and the MC is irreducible. Because the MC is finite, then the single class should be recurrent. On the other extreme, if there are no edges between different nodes in the graph, then each (absorbing) state is a class, and each class is recurrent. Something in between is obtained when the graph comprises $1 < K < J$ communicating

classes (strongly connected components using graph parlance), and there are no edges connecting states from the different classes. (Otherwise, if an edge connects two different communicating classes, then the class from which the edge originates has to be transient.) This way, all states in all classes are recurrent.

- *All states of this MC are aperiodic.* Although not said explicitly, it is reasonable to assume that the graph has no self loops, meaning that $P_{ii} = 0$ for all states i. (Note that we must have self loops in the extreme and uninteresting case that there are no edges among different nodes in the graph, otherwise we do not have a MC. In this trivial case, all states are aperiodic.) Because periodicity is a class property, then e.g., all states will be periodic if all connections are reciprocal (meaning the presence of edge (i, j) implies that (j, i) is present), and there is at least one loop of odd length within each class. This way $P_{ii}^2 \neq 0$ and $P_{ii}^{2n+1} \neq 0$ for some $n > 0$, and since the greatest common divisor between 2 and any odd number is 1, then all states must be aperiodic. This is just one set of conditions, and of course there are others.
- *All states are positive recurrent.* Because the MC is finite, recurrent states are positive recurrent. So the aforementioned conditions under which all states are recurrent also imply that all states are positve recurrent.
- *All states are ergodic.* Ergodic states are aperiodic and positive recurrent. Hence, joint satisfaction of the aforementioned conditions under which all states are positive recurrent and aperiodic imply that all states are ergodic.
- *The MC is irreducible.* As stated earlier, the MC is irreducible when the graph comprises a single strongly connected component. This way all states communicate since there is a directed path between any two nodes in the graph.

B) Implement random walk. The following Matlab function implements the random walk in the graph, and calculates the node ranks for a given network. The initial state is passed as a parameter.

```
function ranks=ranks_by_random_walk(graph,N,initial_state)

J=min(size(graph)); % J : total number of states
nr_neighbors = sum(graph); % number of neighbours of each state

% Here we construct a matrix which contains the indices of the states
% that each state is linked to.
% This matrix called "Neighbors" will serve as a lookup-table
% in order to find out to which state exactly should the random walker
% go. Specifically Neighbors(m,n)=f , if f is not zero, means that
% state m is linked to state f

k=max(nr_neighbors);
Neighbors=zeros(J,k);
for i=1:J
        temp=find(graph(i,:));
        Neighbors(i,1:length(temp))=temp;
end

i=initial_state; % Starting State
nr_visits=zeros(J,1);
for n=1:N
        nr_visits(i)=nr_visits(i)+1;
        i=Neighbors(i,randi(nr_neighbors(i)));
end
ranks=nr_visits/N;
end
```

The Matlab script below calls the function for the provided homework collaboration network. This network is reducible, meaning there are disjoint "collaboration communities" amongst the whole class. For two initial states, namely $A_0 = 1$ and $A_0 = 6$ (belonging to different communication classes), the results are depicted in Fig. 1. As

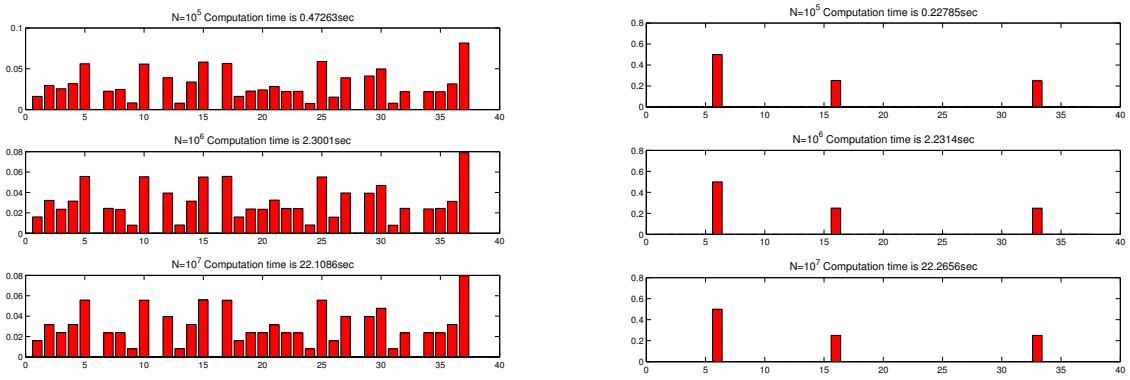

Fig. 1. Ranks calculated based on the random walk on the graph, and taking the time average of the number of visits to each state. Estimated fraction of visits are reported for $N = 10^5, 10^6$, and 10^7, where N is the total duration of the experiment. Initial states are $A_0 = 1$ (left), and $A_0 = 6$ (right). Both states belong to different communication classes, and as expected the obtained ranks are different. (Part *B*)

expected, since the MC is reducible the ranks depend on the initial condition, and in this particular case they are quite different. The length of the experiment is also varied, using values $N = 10^i$ for $i = 5, 6$, and 7.

We can investigate the disjoint classes by looking at the resulting ranks in Fig. 1. We observe from Fig. 1 (left) that some of the states receive zero rank when $A_0 = 1$ (see e.g., states 6, 11, 16, 28, 33). The conclusion is that these states do not belong to the same communication class as state 1. Starting from state 6, we obtain the ranks in Fig. 1 (right) which shows that states 6, 16 and 33 comprise a separate class. Starting from states 11 and 28 shows that each of them are singleton (absorbing) states, and hence separate classes.

```
clc;close all;clear all;
collaboration_matrix % Loads the collaboration graph

J=min(size(graph));

figure
for j=1:3
        N=10^(4+j);
        tic
        ranks=ranks_by_random_walk(graph,N,1);
        t=toc

        subplot(3,1,j)
        bar(1:J,ranks,'r')
        title(['N=10^',num2str(j+4), ' Computation time is ',...
                        ...num2str(t),'sec'],'FontSize',12)
end
```

To obtain an irreducible MC so that ranks do not depend on the initial state A_0, we add an artificial node (the professor) that is connected to all other nodes. The following Matlab script implements the required change; see the 3rd line which adds an extra row and column consisting of all ones. The resulting ranks are depicted in Fig. 2.

```
clc;close all;clear all;
collaboration_matrix % Loads the collaboration graph
graph=[graph ones(J,1);ones(1,J) 0]; % Augmenting with the
                                     % fully connected node (professor)

J=min(size(graph));
```

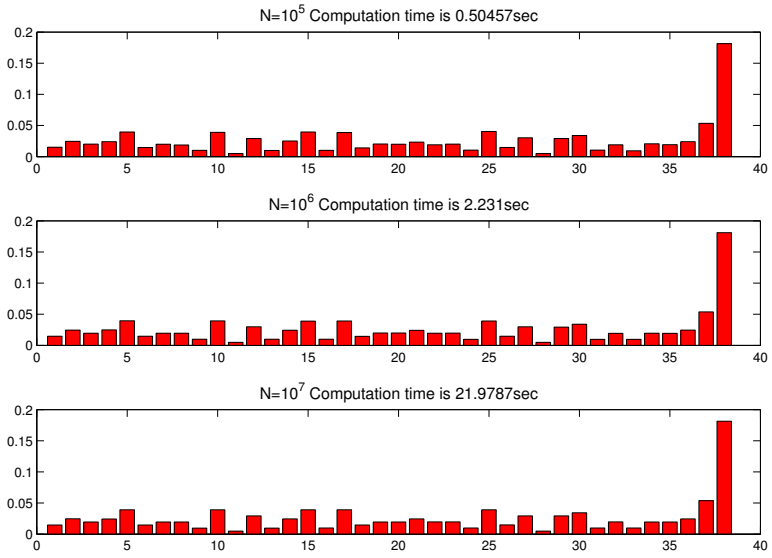

Fig. 2. Ranks calculated based on the random walk on the graph, and taking the time average of the number of visits to each state. Estimated fraction of visits are reported for $N = 10^5, 10^6$, and 10^7, where N is the total duration of the experiment. Initial state is $A_0 = 1$, and the MC is irreducible. (Part *B*)

```
figure
for j=1:3
        N=10^(4+j);
        tic
        ranks=ranks_by_random_walk(graph,N,1);
        t=toc

        subplot(3,1,j)
        bar(1:J,ranks,'r')
        title(['N=10^',num2str(j+4), ' Computation time is ',...
                        ...num2str(t),'sec'],'FontSize',12)
end
```

C) Probability update. Let $p_i(n) := \mathrm{P}[A_n = i]$ denote the unconditional probability that the outside agent is at node i at time n. It is possible to express $p_i(n+1)$ in terms of the probabilities at time n of those nodes that can transition into i. In fact, using the law of total probability after conditioning on the state at time n, one obtains

$$\mathrm{P}[A_{n+1} = i] = \sum_{j \in n^{-1}(i)} \mathrm{P}\left[A_{n+1} = i \,\middle|\, A_n = j\right] \mathrm{P}[A_n = j]$$

which is

$$p_i(n+1) = \sum_{j \in n^{-1}(i)} P_{ji} p_j(n).$$

Defining the column vector $\mathbf{p}(n) := [p_1(n), \ldots, p_J(n)]^T$, the update equation can be compactly expressed in matrix form as

$$\mathbf{p}(n+1) = \mathbf{P}^T \mathbf{p}(n).$$

D) Find ranks using the probability update. The limiting probabilities $\lim_{n\to\infty} p_i(n)$ exist provided the MC is aperiodic (when it is periodic, the probabilities oscillate and do not settle into steady-state values). Since the MC

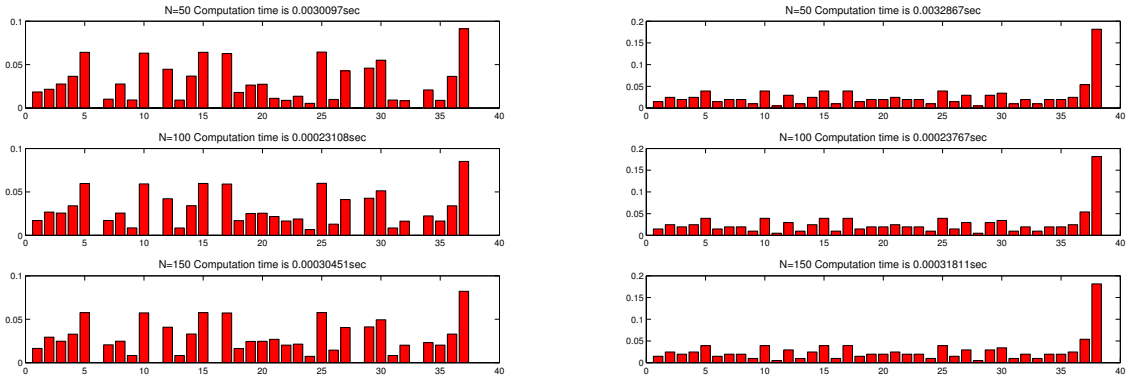

Fig. 3. Ranks calculated based on the probability updates for the reducible network (left), and the irreducible case (right). Estimated times for $N = 50, 100$, and 150 are reported, where N is the total number of iterations. The initial distribution is $\mathbf{p}(0) = [1, 0, \ldots, 0]^T$. (Part *D*)

under study is aperiodic, these limits always exist. Suppose $p_{A_0}(0) = 1$, that is the outside agent begins at node A_0 almost surely. Then from the conditional version of the ergodic theorem (that is, restricting the evolution of the MC to the ergodic class to which A_0 belongs), the rank of state i can be obtained as

$$r_i(A_0) = \lim_{n\to\infty} \frac{1}{n} \sum_{m=1}^{n} \mathbb{I}(A_m = i) = \lim_{n\to\infty} P^n_{A_0,i} = \lim_{n\to\infty} p_i(n). \tag{2}$$

Notice that if state i belongs to a different class than A_0, then $r_i(A_0) = 0$ because state i is never visited. The Matab function to compute the ranks using the probability updates as suggested by (2) follows.

```
function ranks=ranks_by_probability_update(graph,N,initial_distribution)

J=min(size(graph)); % J : total number of states
pi_D=initial_distribution; % given initial distribution

nr_neighbors = sum(graph);

transition_probabilities = graph;
for k=1:J
        if nr_neighbors(k)>0
                transition_probabilities(k,:)=graph(k,:)/nr_neighbors(k);
        else
                transition_probabilities(k,k)=1;
        end
end

for n=1:N
        pi_D=transition_probabilities'*pi_D; % Probability updates
end
ranks=pi_D;
end
```

The results for the reducible MC are shown in Fig. 3 (left). The calculated ranks coincide with those obtained in part *B*, see also Fig. 1 (left). Interestingly, convergence is attained after only around 100 iterations, whereas using the random walk model it takes at least 10^6 iterations (you can also check the running times from the plots).

For the limiting probabilities to be independent of the initial distribution, the MC must be ergodic. By adding a fully connected node (the professor), the MC becomes irreducible and also ergodic because all states are aperiodic.

In this case, we know from the ergodic theorem that

$$r_i = \lim_{n\to\infty} \frac{1}{n} \sum_{m=1}^{n} \mathbb{I}(A_m = i) = \lim_{n\to\infty} P^n_{ji} = \lim_{n\to\infty} p_i(n) \tag{3}$$

independent of the initial condition, i.e., for all $1 \leq j \leq J$ in (3). So (3) justifies why the ranks can be computed using probability updates. To do so, one can run the Matlab function above with the irreducible graph as an input. The results are shown in Fig. 3 (right), and again they coincide with those obtained in part *B*, see also Fig. 2.

E) Recast as system of linear equations. Focusing on the modified, irreducible graph we can compute the ranks from the system of linear equations defining the MC's unique stationary distribution $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_J]^T$. Again, notice that in the ergodic case one has

$$r_i = \lim_{n\to\infty} \frac{1}{n} \sum_{m=1}^{n} \mathbb{I}(A_m = i) = \pi_i$$

where the stationary distribution satisfies

$$\boldsymbol{\pi} = \mathbf{P}^T \boldsymbol{\pi}, \quad \boldsymbol{\pi}^T \mathbf{1} = 1 \tag{4}$$

and $\mathbf{1}$ is the $J \times 1$ vector of all ones. It follows from (4) that $\boldsymbol{\pi}$ is the unique solution to the following system of linear equations

$$\begin{pmatrix} \mathbf{I} - P^T \\ \mathbf{1}^T \end{pmatrix} \boldsymbol{\pi} = \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}$$

where $\mathbf{I}$ is the $J \times J$ identity matrix, and $\mathbf{0}$ is the $(J+1) \times 1$ vector of all ones. In a nutshell, the above system implies that one can find $\boldsymbol{\pi}$ as a normalized vector (entries should sum up to one) spanning the null-space of the matrix $(\mathbf{I} - \mathbf{P}^T)$. The script to implement such procedure follows. The resulting ranks are shown in Fig. 4 (left).

```
clc;close all;clear all;
collaboration_matrix % Loads the collaboration graph
graph=[graph ones(J,1);ones(1,J) 0]; % Augmenting with the
                                     % fully connected node (professor)

J=min(size(graph)); % J : total number of states
nr_neighbors = sum(graph);
transition_probabilities = graph;
for k=1:J
        if nr_neighbors(k)>0
                transition_probabilities(k,:)=graph(k,:)/nr_neighbors(k);
        else
                transition_probabilities(k,k)=1;
        end
end

figure
tic
pi=null(eye(J)-transition_probabilities'); % Find vector in the nullspace
ranks=pi/sum(pi); % Normalize
t=toc

% Plot results
bar(1:J,ranks,'r')
title(['Recasting as Linear system problem,', ' Computation time is ',...
              ...num2str(t),'sec'],'FontSize',12)
xlabel('states','Fontsize',12)
ylabel('ranks','Fontsize',12)
```

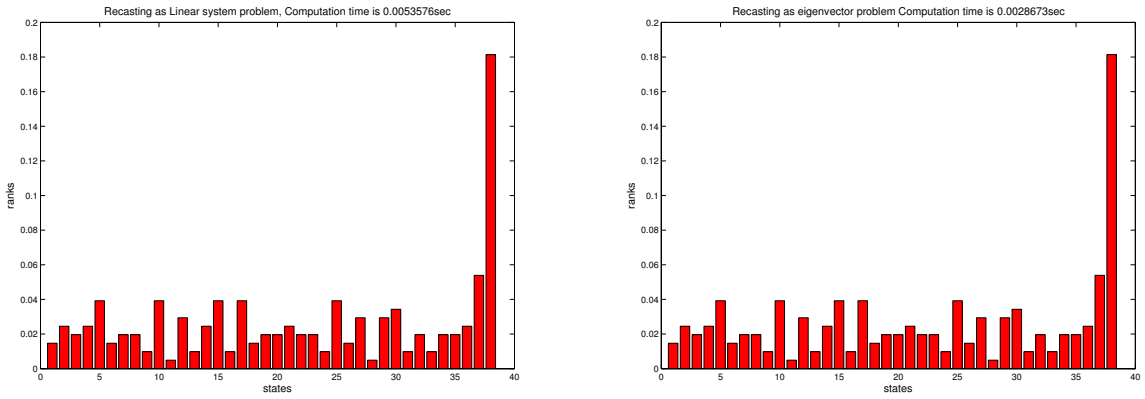

Fig. 4. Ranks calculated as the solution to a system of linear equations (left), and an eigenvalue problem (right). (Parts *E* and *F*.)

F) Recast as eigenvalue problem. From (4) it follows that $\boldsymbol{\pi}$ can also be computed as the normalized eigenvector (entries should sum up to one) of matrix $\mathbf{P}^T$, associated with the eigenvalue 1. The Matlab script to implement such procedure follows, and the obtained ranks are shown in Fig. 4 (right).

```
clc;close all;clear all;
collaboration_matrix % Loads the collaboration graph
graph=[graph ones(J,1);ones(1,J) 0]; % Augmenting with the
                                     % fully connected node (professor)

J=min(size(graph)); % J : total number of states
nr_neighbors = sum(graph);
transition_probabilities = graph;
for k=1:J
        if nr_neighbors(k)>0
                transition_probabilities(k,:)=graph(k,:)/nr_neighbors(k);
        else
                transition_probabilities(k,k)=1;
        end
end

figure
tic
[V,D]=eig(transition_probabilities'); % Perform eigendecomposition of P'
ranks=V(:,1)/sum(V(:,1)); % Normalize dominant eigenvector
t=toc

% Plot results
bar(1:J,ranks,'r')
title(['Recasting as Linear system problem,', ' Computation time is ',...
...num2str(t),'sec'],'FontSize',12)
xlabel('states','Fontsize',12)
ylabel('ranks','Fontsize',12)
```

G) Discuss advantages of each method. Restricting our attention to the modified (irreducible) graph containing the fully connected node, we have seen that all four methods in parts *B-F* yield the same results. But each of them has particular advantages that make them suitable for different applications, as summarized next.

- *Random walk on the graph.* The random walk approach is preferable in that it is secure, meaning that rank or

graph topology information is not shared between nodes. For a node to determine its own rank, it only needs to know its outgoing neighborhood and the state of a global synchronization clock (yielding the value of n). This also means that implementation can be distributed, i.e., global network information need not be compiled and processed in a central location.

- *Probability update.* The probability update, motivated by probability propagation in a MC is similar to the random walk in that implementation can be distributed and it is fairly secure (but less secure than the random walk algorithm, since probability updates require exchanging current ranks with neighbors). The main advantage is that it converges to the true ranks markedly faster than the random walk implementation. If the MC is very large, the iterations can be halted at any time to save computations, thus providing an approximation for the ranks of all J states. This should be compared with the algorithms that either solve a linear system or an eigenvalue problem, where in principle one does not have the flexibility to control the accuracy of the solution. We also observed that probability updates boil down to simple matrix times vector multiplications, for which fast algorithms are available. (This is at the heart of Matlab's success story, since its routines are designed and optimized specifically to carry out such matrix computations remarkably fast.)
- *System of linear equations.* Solving the system of linear equations eliminates the problem of slow convergence, because (in theory) it does require an iterative procedure. In practice, be aware that Matlab's routines to solve linear systems are iterative. As a disadvantage, solving the linear system requires having the whole network topology centrally available to carry out the computations, which is the least secure and robust. The worst case complexity of solving a linear system is cubic in the input size (J), so for moderate-to-large graphs this approach could become infeasible.
- *Eigenvalue problem.* For the most part, the eigendecompostion approach shares the advantages and disadvantage of its system of linear equations counterpart.