

Chapter 2

Queueing Systems and the Web

In this chapter, we first discuss some more Markovian queueing systems. The queueing system is a classical application of continuous Markov chains. We then present an important numerical algorithm based on the computation of Markov chains for ranking webpages. This is a modern application of Markov chains though the numerical methods used are classical.

2.1 Markovian Queueing Systems

An important class of queueing networks is the Markovian queueing systems. The main assumptions of a Markovian queueing system are the Poisson arrival process and exponential service time. The one-server system discussed in Sect. 1.2.1 of Chap. 1 is a queueing system without waiting space. This means that when a customer arrives and finds the server is busy, the customer has to leave the system. In the following sections, we will introduce some more Markovian queueing systems. A queueing system is a classical application of a continuous time Markov chain. We will further discuss its applications in re-manufacturing systems in Chap. 3. For more details about numerical solutions for queueing systems and Markov chains, we refer the readers to the books by Ching [46], Leonard [144], Neuts [167, 168] and Stewart [191].

In the following, we begin our discussion with an $M/M/1/n - 2$ queue, a Markovian queueing system with one server and $n - 2$ waiting spaces. Here the first 'M' representing the arrival process is a Poisson process and the second 'M' represents the service time following the exponential distribution.

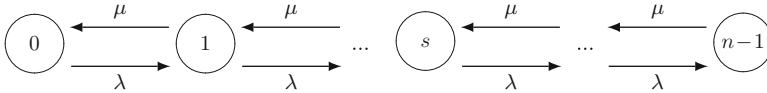


Fig. 2.1 The Markov chain for the one-queue system (one server)

2.1.1 An $M/M/1/n - 2$ Queueing System

Now let us consider a more general queueing system with customer arrival rate being λ . Suppose the system has one exponential server with service rate being μ and there are $n - 2$ waiting spaces in the system. The queueing discipline is *First-come-first-serve*. When a newly arrived customer finds the server is busy, the customer can still wait in the queue provided that there is a waiting space available. Otherwise, the customer has to leave the queueing system. To describe the queueing system, we use the number of customers in the queue to represent the state of the system. There are n states, namely $0, 1, \dots, n - 1$. The Markov chain for the queueing system is given in Fig. 2.1. The number of customers in the system is used to represent the states in the Markov chain. Clearly it is an irreducible Markov chain.

If we order the states of the system in increasing number of customers, it is not difficult to show that the generator matrix for this queueing system is given by the following $n \times n$ tri-diagonal matrix $A_1 = A_{(n,1,\lambda,\mu)}$ where

$$A_1 = \begin{pmatrix} \lambda & -\mu & & & & & 0 \\ -\lambda & \lambda + \mu & -\mu & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & -\lambda & \lambda + \mu & -\mu & & \\ & & & -\lambda & \lambda + \mu & -\mu & \\ & & & & \ddots & \ddots & \ddots \\ & & & & & -\lambda & \lambda + \mu & -\mu \\ 0 & & & & & & -\lambda & s\mu \end{pmatrix} \quad (2.1)$$

and transient solution $\mathbf{p}(t)$ satisfies the following system of differential equations:

$$\frac{d\mathbf{p}(t)}{dt} = A_1 \mathbf{p}(t).$$

The underlying Markov chain is irreducible and the solution for the steady-state probability distribution, i.e.,

$$\lim_{t \rightarrow \infty} \mathbf{p}(t) = \mathbf{p}_{(n,1,\lambda,\mu)} \quad \text{and} \quad \mathbf{0} = \lim_{t \rightarrow \infty} \frac{d\mathbf{p}(t)}{dt} = A_1 \mathbf{p}_{(n,1,\lambda,\mu)}$$

can be shown to be $\mathbf{p}_{(n,1,\lambda,\mu)} = (p_0, p_1, \dots, p_{n-1})^T$ where

$$p_i = \alpha \prod_{k=1}^{i+1} \frac{\lambda}{\mu} \quad \text{and} \quad \alpha^{-1} = \sum_{i=0}^n p_i. \quad (2.2)$$

Here p_i is the probability that there are i customers in the queueing system in the steady-state and α is the *normalization constant*.

Example 2.1. In the one-server system, the steady-state probability distribution is given by

$$p_i = \frac{\rho^i (1 - \rho)}{1 - \rho^n} \quad \text{where} \quad \rho = \frac{\lambda}{\mu}.$$

When the system has no limit on waiting space, we have a $M/M/1/\infty$ queue (or simply $M/M/1$ queue). Suppose that $\rho < 1$, the steady-state probability becomes

$$\lim_{n \rightarrow \infty} p_i = \rho^i (1 - \rho).$$

The expected number of customers in the system is given by

$$\begin{aligned} L_c &= \sum_{i=0}^{\infty} i p_i = \sum_{i=0}^{\infty} i \rho^i (1 - \rho) \\ &= \frac{\rho(1 - \rho)}{(1 - \rho)^2} = \frac{\rho}{1 - \rho}. \end{aligned} \quad (2.3)$$

The expected number of customers waiting in the queue is given by

$$\begin{aligned} L_q &= \sum_{i=1}^{\infty} (i - 1) p_i = \sum_{i=1}^{\infty} (i - 1) \rho^i (1 - \rho) \\ &= \frac{\rho}{1 - \rho} - \rho = \frac{\rho^2}{1 - \rho}. \end{aligned} \quad (2.4)$$

Moreover the expected number of customers in service is given by

$$L_s = 0 \cdot p_0 + 1 \cdot \sum_{i=1}^{\infty} p_i = 1 - (1 - \rho) = \rho.$$

2.1.2 An $M/M/s/n - s - 1$ Queueing System

Now let us consider a more general queueing system with customer arrival rate being λ . Suppose the system has s parallel and identical exponential servers with

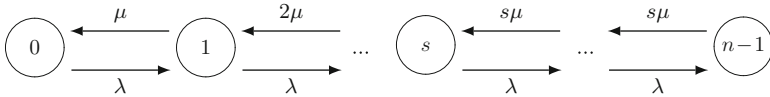


Fig. 2.2 The Markov chain for the one-queue system (s servers)

service rate being μ and there are $n - s - 1$ waiting spaces in the system. The queueing discipline is *First-come-first-serve*. Again when a customer arrives and finds all the servers are busy, the customer can still wait in the queue provided that there is a waiting space available. Otherwise, the customer has to leave the system. To apply the continuous time Markov chains for modeling this queueing system, one has to obtain the waiting time for the departure of one customer when there are more than one customer (let us say k customers) in the queueing system. We need the following lemma.

Lemma 2.2. *Suppose that X_1, X_2, \dots, X_k are independent, identical, exponential random variables with mean μ^{-1} , and consider the corresponding order statistics*

$$X_{(1)} \leq X_{(2)} \leq \dots \leq X_{(k)}.$$

Then $X_{(1)}$ is again exponentially distributed with mean $\frac{1}{k}$ times the mean of the original random variables.

Proof. We observe that

$$X_{(1)} = \min(X_1, X_2, \dots, X_k).$$

Hence $X_{(1)} > x$ if and only if all $X_i > x$ ($i = 1, 2, \dots, k$). We therefore have

$$\begin{aligned} P\{X_{(1)} > x\} &= P\{X_1 > x\}P\{X_2 > x\} \cdots P\{X_k > x\} \\ &= (e^{-\mu x})^k \\ &= e^{-k\mu x}. \end{aligned}$$

Again it is still exponentially distributed with mean $1/(k\mu)$. If we use the number of customers in the queue to represent the state of the system, then there are n states, namely $0, 1, \dots, n - 1$. The Markov chain for the queueing system is given in Fig. 2.2. The number of customers in the system is used to represent the states in the Markov chain. Clearly it is an irreducible Markov chain.

If we order the states of the system in increasing number of customers, it is not difficult to show that the generator matrix for this queueing system is given by the following $n \times n$ tri-diagonal matrix $A_2 = A_{(n,s,\lambda,\mu)}$ where

$$A_2 = \begin{pmatrix} \lambda & -\mu & & & & & & 0 \\ -\lambda & \lambda + \mu & -2\mu & & & & & \\ & \ddots & \ddots & \ddots & & & & \\ & & -\lambda & \lambda + (s-1)\mu & -s\mu & & & \\ & & & -\lambda & \lambda + s\mu & -s\mu & & \\ & & & & \ddots & \ddots & \ddots & \\ & & & & & -\lambda & \lambda + s\mu & -s\mu \\ 0 & & & & & & -\lambda & s\mu \end{pmatrix} \quad (2.5)$$

and the underlying Markov chain is irreducible. The solution for the steady-state probability distribution can be shown to be

$$\mathbf{p}_{(n,s,\lambda,\mu)} = (p_0, p_1, \dots, p_{n-1})^T \quad (2.6)$$

where

$$p_i = \alpha \prod_{k=1}^{i+1} \frac{\lambda}{\mu \min\{k, s\}}$$

and

$$\alpha^{-1} = \sum_{i=0}^n p_i.$$

Here p_i is the probability that there are i customers in the queueing system in steady-state and α is the *normalization constant*.

2.1.3 Allocation of the Arrivals in a System of M/M/1/ ∞ Queues

In this subsection, we consider a queueing system consisting of n independent M/M/1/ ∞ queues. The service rate of the server at the i th queue is μ_i . Again we assume that the arrival process is a Poisson process with rate λ . The allocation of arrivals is an important decision process in a queueing system, see for instance [193] and the references therein. Here we consider an allocation process proposed in [193], which is implemented such that it diverts an arrived customer to queue i with a probability given by

$$\frac{\lambda_i}{\lambda_1 + \dots + \lambda_n} = \frac{\lambda_i}{\lambda}.$$

Then the input process of Queue i is a Poisson process with rate λ_i . The objective here is to find the parameters λ_i such that some system performance is optimized. Here we must assume $\lambda_i < \mu_i$.

There are many system performance indicators. In what follows, our main objective is to minimize the total expected number of customers in the system. We first obtain the expected number of customers in Queue i by the result in (2.3) as follows:

$$\frac{\lambda_i / \mu_i}{1 - \lambda_i / \mu_i}.$$

Then the total expected number of customers in the system is

$$\sum_{i=1}^n \frac{\lambda_i / \mu_i}{1 - \lambda_i / \mu_i}.$$

The optimization problem is then given as follows:

$$\min_{\lambda_1, \dots, \lambda_n} \left\{ \sum_{i=1}^n \frac{\lambda_i / \mu_i}{1 - \lambda_i / \mu_i} \right\}$$

subject to

$$\sum_{i=1}^m \lambda_i = \lambda$$

and

$$0 \leq \lambda_i < \mu_i \quad \text{for } i = 1, 2, \dots, n.$$

The Lagrangian function is given by

$$L(\lambda_1, \dots, \lambda_n, m) = \sum_{i=1}^n \frac{\lambda_i / \mu_i}{1 - \lambda_i / \mu_i} - m \left(\sum_{i=1}^n \lambda_i - \lambda \right).$$

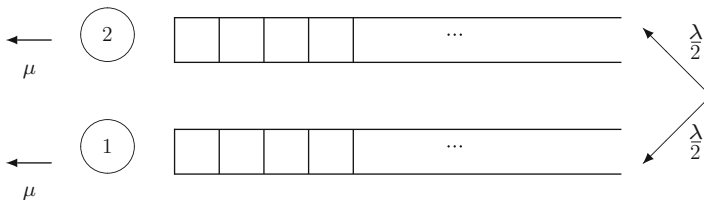
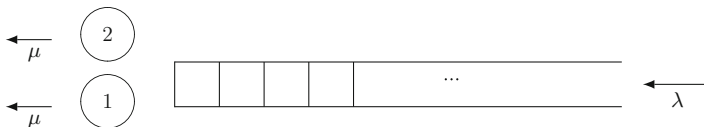
Solving the following equations

$$\frac{\partial L}{\partial \lambda_i} = 0, \quad i = 1, 2, \dots, n \quad \text{and} \quad \frac{\partial L}{\partial m} = 0$$

we get the optimal solution

$$\lambda_i = \mu_i \left(1 - \frac{1}{\sqrt{m \mu_i}} \right) < \mu_i, \quad i = 1, 2, \dots, n$$

where

**Fig. 2.3** Case (i) Two M/M/1/∞ Queues**Fig. 2.4** Case (ii) One M/M/2/∞ Queue

$$m = \left(\frac{\sum_{i=1}^n \sqrt{\mu_i}}{\sum_{i=1}^n \mu_i - \lambda} \right)^2.$$

Another possible system performance indicator is the total expected number of customers waiting in the system. One can carry out a similar analysis and we leave the problem to the readers as an exercise.

2.1.4 Two M/M/1 Queues or One M/M/2 Queue?

In an M/M/1/ queueing system with service rate μ and arrival rate λ (we assume that $\lambda < \mu$), suppose one extra identical server can be added, then which of following situations is better (we assume that $\lambda < \mu$)? (i) Separate the two operators. Therefore we have two M/M/1/∞ queues. In this case, we assume that an arriving customer can either join the first queue or the second with equal chance. (ii) Join the two operators together. Therefore we have an M/M/2/∞ queue (Figs. 2.3 and 2.4).

To determine which case is better, one can calculate the expected number of customers in both cases. Very often, in our consideration, the smaller the better. In case (i), using the result in (2.3), the expected number of customers in any one of the queueing systems will be given by

$$\frac{\left(\frac{\lambda}{2\mu}\right)}{1 - \left(\frac{\lambda}{2\mu}\right)}.$$

Hence the total expected number of customers in the queueing system is

$$S_1 = 2 \times \frac{\left(\frac{\lambda}{2\mu}\right)}{1 - \left(\frac{\lambda}{2\mu}\right)} = \frac{\left(\frac{\lambda}{\mu}\right)}{1 - \left(\frac{\lambda}{2\mu}\right)}.$$

Now we examine the second case. In Case (ii), the expected number of customers in the queueing system will be given by

$$S_2 = \frac{\left(\frac{\lambda}{\mu}\right)}{1 - \left(\frac{\lambda}{2\mu}\right)^2} \quad (2.7)$$

and this is left as an exercise. It is straightforward to see that $S_2 < S_1$. We then conclude that Case (ii) is better. We should put all the servers together if our concern is to minimize the total number of customers in the system.

2.1.5 The Two-Queue Free System

In this subsection, we introduce a higher dimensional queueing system. Suppose that there are two one-queue systems as discussed in Sect. 2.1.2. This queueing system consists of two independent queues with the number of identical servers and waiting spaces being s_i and $n_i - s_i - 1$ ($i = 1, 2$) respectively.

If we let the arrival rate of customers in Queue i be λ_i and the service rate of the servers be μ_i ($i = 1, 2$) then the states of the queueing system can be represented by the elements in the following set:

$$S = \{(i, j) | 0 \leq i \leq n_1, 0 \leq j \leq n_2\}$$

where (i, j) represents the state that there are i customers in Queue 1 and j customers in Queue 2. Thus this is a two-dimensional queueing model. If we order the states lexicographically, then the generator matrix can be shown to be the following $n_1 n_2 \times n_1 n_2$ matrix in *tensor product* form [37, 46]:

$$A_3 = I_{n_1} \otimes A_{(n_2, s_2, \lambda_2, \mu_2)} + A_{(n_1, s_1, \lambda_1, \mu_1)} \otimes I_{n_2}. \quad (2.8)$$

Here \otimes is the Kronecker tensor product [108, 119]. The Kronecker tensor product of two matrices A and B of sizes $p \times q$ and $m \times n$ respectively is a $(pm) \times (qn)$ matrix given as follows:

$$A \otimes B = \begin{pmatrix} a_{11}B & \cdots & \cdots & a_{1q}B \\ a_{21}B & \cdots & \cdots & a_{2q}B \\ \vdots & \vdots & \vdots & \vdots \\ a_{p1}B & \cdots & \cdots & a_{pq}B \end{pmatrix}.$$

The Kronecker tensor product is a useful tool for representing generator matrices in many queueing systems and stochastic automata networks [37, 46, 141, 142, 191]. For this two-queue free queueing system, it is also not difficult to show that the steady-state probability distribution is given by the probability distribution vector

$$\mathbf{P}(\mathbf{n}_1, s_1, \lambda_1, \mu_1) \otimes \mathbf{P}(\mathbf{n}_2, s_2, \lambda_2, \mu_2). \quad (2.9)$$

2.1.6 The Two-Queue Overflow System

Now let us add the following system dynamics to the two-queue free system discussed in Sect. 2.1.5. In this queueing system, we allow overflow of customers from Queue 2 to Queue 1 whenever Queue 2 is full and there is still waiting space in Queue 1; see for instance Fig. 2.5 (Taken from [46]). This is called the two-queue overflow system; see Kaufman [37, 46, 130].

In this case, the generator matrix is given by the following matrix:

$$A_4 = I_{n_1} \otimes A_{(n_2, s_2, \lambda_2, \mu_2)} + A_{(n_1, s_1, \lambda_1, \mu_1)} \otimes I_{n_2} + R \otimes \mathbf{e}_{n_2}^T \mathbf{e}_{n_2}. \quad (2.10)$$

Here \mathbf{e}_{n_2} is the unit vector $(0, 0, \dots, 0, 1)$ and

$$R = \begin{pmatrix} \lambda_2 & & & & 0 \\ -\lambda_2 & \lambda_2 & & & \\ & -\lambda_2 & \ddots & & \\ & & \ddots & \lambda_2 & \\ 0 & & & -\lambda_2 & 0 \end{pmatrix}. \quad (2.11)$$

In fact

$$A_4 = A_3 + R \otimes \mathbf{e}_{n_2}^T \mathbf{e}_{n_2},$$

where $R \otimes \mathbf{e}_{n_2}^T \mathbf{e}_{n_2}$ is the matrix describing the overflow of customers from Queue 2 to Queue 1. Unfortunately, there is no analytical solution for the generator matrix A_4 .

For an overflow queueing system, a closed form solution of the steady-state probability distribution may not always be available. In fact, there are a lot applications related to queueing systems whose problem size are very large [26–28, 36, 37, 46, 77, 152]. Direct methods for solving the probability distribution such as the Gaussian elimination and LU factorization can be found in [132, 191]. Another popular method is called the matrix analytic methods [141, 142]. Apart from these direct methods, another class of popular numerical methods called iterative methods, exists. This includes the classical iterations introduced in Chap. 1, such as the Jacobi method, Gauss-Seidel method and SOR method. Sometimes when the generator matrix has a block structure, then the block Jacobi method, block Gauss-Seidel method and block SOR method are also popular methods [108].

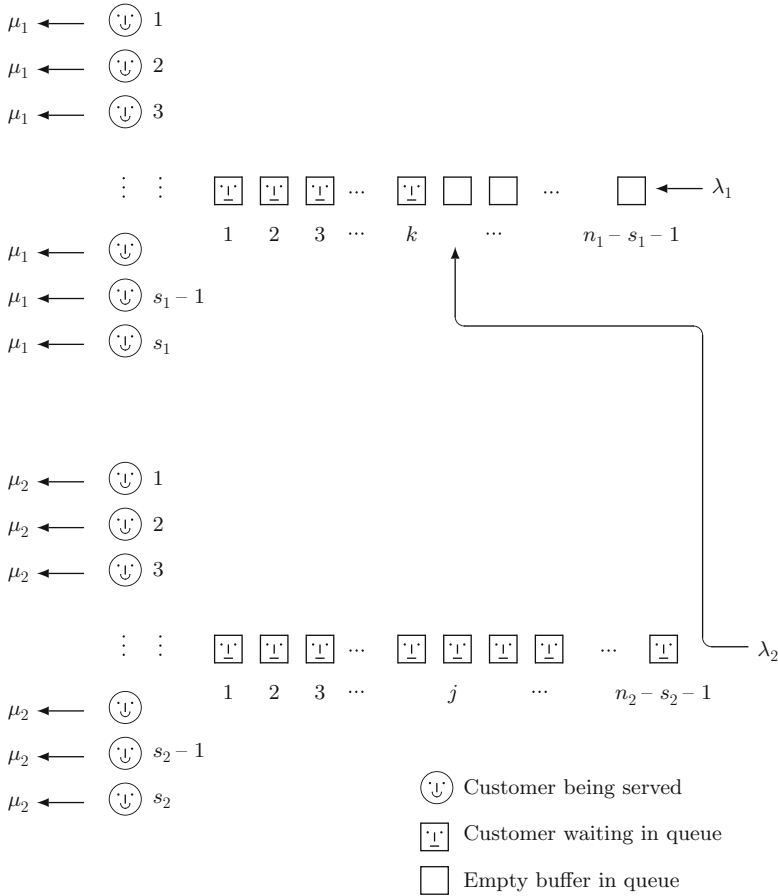


Fig. 2.5 The two-queue overflow system

A hybrid numerical algorithm which combines both SOR and a genetic algorithm has also been introduced by Ching et al. [213] for solving queueing systems. Conjugate gradient methods with circulant-based preconditioners are efficient solvers for a class of Markov chains having near-Toeplitz generator matrices. We will briefly discuss this in the following subsection.

2.1.7 The Preconditioning of Complex Queueing Systems

In many complex queueing systems, one observes both block structure, near-Toeplitz structure and sparsity in the generator matrices. Therefore an iterative method such as the CG method can be a good solver with a suitable preconditioner.

2.1.7.1 Circulant-Based Preconditioners

In this subsection, we illustrate how to get a circulant preconditioner from a generator matrix of a queueing system. The generator matrices of the queueing networks can be written in terms of the sum of tensor products of matrices. Very often, a key block structure of a queueing system is the following: $(n + s + 1) \times (n + s + 1)$ tri-diagonal matrix:

$$Q = \begin{pmatrix} \lambda & -\mu & & & & & 0 \\ -\lambda & \lambda + \mu & -2\mu & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & -\lambda & \lambda + (s-1)\mu & -s\mu & & \\ & & & -\lambda & \lambda + s\mu & -s\mu & \\ & & & & \ddots & \ddots & \ddots \\ & & & & & -\lambda & \lambda + s\mu & -s\mu \\ 0 & & & & & & -\lambda & s\mu \end{pmatrix}. \quad (2.12)$$

This is the generator matrix of an M/M/s/n queue. In this queueing system there are s independent exponential servers, the customers arrive according to a Poisson process of rate λ and each server has a service rate of μ .

One can observe that if s is fixed and n is large, then Q is close to the following tridiagonal Toeplitz matrix $\text{Tri}[\lambda, -\lambda - s\mu, s\mu]$. In fact, if one considers the following circulant matrix $c(Q)$:

$$c(Q) = \begin{pmatrix} \lambda + s\mu & -s\mu & & & -\lambda \\ -\lambda & \lambda + s\mu & -s\mu & & \\ & \ddots & \ddots & \ddots & \\ & & -\lambda & \lambda + s\mu & -s\mu \\ -s\mu & & & -\lambda & \lambda + s\mu \end{pmatrix} \quad (2.13)$$

it is easy to see that

$$\text{rank}(c(Q) - Q) \leq s + 1$$

is independent of n for fixed s . Therefore, for fixed s and large value of n , the approximation is a good one. Moreover, $c(Q)$ can be diagonalized by the discrete Fourier Transformation and a closed form solution of its eigenvalues can be easily obtained. This is important in the convergence rate analysis of the CG method. By applying this circulant approximation to the blocks of the generator matrices, effective preconditioners are constructed and the preconditioned systems are also proved to have singular values clustered around one, see for instance Chan and Ching [37]. A number of related applications can also be found in [36, 37, 41, 43, 45, 46, 49].

2.1.7.2 Toeplitz-Circulant-Based Preconditioners

Another class of queueing systems with batch arrivals have been discussed by Chan and Ching in [36]. The generator matrices of queueing systems of s identical exponential servers with service rate μ take the form

$$A_n = \begin{pmatrix} \lambda & -\mu & 0 & 0 & 0 & \dots & 0 \\ -\lambda_1 & \lambda + \mu & -2\mu & 0 & 0 & \dots & 0 \\ -\lambda_2 & -\lambda_1 & \lambda + 2\mu & \ddots & \ddots & & \vdots \\ \vdots & -\lambda_2 & \ddots & \ddots & -s\mu & \ddots & \\ & \vdots & \ddots & \ddots & \lambda + s\mu & \ddots & 0 \\ -\lambda_{n-2} & -\lambda_{n-3} & \dots & & \ddots & \ddots & -s\mu \\ -r_1 & -r_2 & -r_3 & \dots & -r_{s+1} & \dots & s\mu \end{pmatrix}, \quad (2.14)$$

where r_i are such that each column sum of A_n is zero, i.e.

$$r_i = \lambda - \sum_{k=n-i}^{\infty} \lambda_k.$$

Here λ is the arrival rate and $\lambda_i = \lambda p_i$ where p_i is the probability that an arrived batch is of size i . It is clear that the matrix is dense and the method of circulant approximation does not work directly in this case. A Toeplitz-circulant type of preconditioner was proposed to solve this queueing system by Chan and Ching [36]. The idea is that the generator matrix is close to a Toeplitz matrix whose generating function has a zero on the unit circle of order one. By factoring the zero, the quotient has no zero on the unit circle. Using this fact, a Toeplitz-circulant preconditioner is then constructed for the queueing system. Both the construction cost and the preconditioner system can be solved in $n \log(n)$ operations. Moreover, the preconditioned system was proved to have singular values clustered around one. Hence a very fast convergence rate is expected when the CG method is applied to solving the preconditioned system.

This idea was further applied to queueing systems with batch arrivals and negative customers, see Ching [48]. The term “negative customer” was first introduced by Gelenbe et al. [102–104] in the modelling of neural networks. Here the role of a negative customer is to remove a number of customers waiting in the queueing system. For example, one may consider a communication network in which messages are transmitted in a packet-switching mode. When a server fails (this corresponds to an arrival of a negative customer) during a transmission, parts of the messages will be lost. One may also consider a manufacturing system where a negative customer represents a cancellation of a job. These lead to many practical applications in the modelling of physical systems.

In the queueing system, we assume that the arrival process of the batches of customers follows a Poisson process of rate λ . The batch size again follows a stationary distribution of p_i ($i = 1, 2, \dots$). Here p_i is the probability that an arrived batch is of size i . It is also assumed that the arrival process of negative customers is a Poisson process with rate τ . The number of customers to be removed is assumed to follow a probability distribution

$$b_i (i = 1, 2, \dots).$$

Furthermore, if the arrived negative customer is supposed to remove i customers in the system, but the number of customers in the system is less than i , then the queueing system will become empty. The removal (killing) strategy here is to remove the customers in the front of the queue, i.e. “Remove the Customers at the Head” (RCH). For $i \geq 1$, we let

$$\tau_i = b_i \tau$$

where b_i is the probability that the number of customers to be removed is i and therefore we have

$$\tau = \sum_{k=1}^{\infty} \tau_k.$$

The generator matrices of the queueing systems take the following form:

$$A_n = \begin{pmatrix} \lambda & -u_1 & -u_2 & -u_3 & \dots & \dots & \dots & -u_{n-1} \\ -\lambda_1 & \lambda + \tau + \mu & -2\mu - \tau_1 & -\tau_2 & -\tau_3 & \dots & \dots & -\tau_{n-2} \\ -\lambda_2 & -\lambda_1 & \lambda + \tau + 2\mu & \ddots & \ddots & \ddots & & \vdots \\ \vdots & -\lambda_2 & \ddots & \ddots & -s\mu - \tau_1 & -\tau_2 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \lambda + \tau + s\mu & \ddots & \ddots & -\tau_3 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & -\tau_2 \\ -\lambda_{n-2} & -\lambda_{n-3} & -\lambda_{n-4} & \dots & \lambda_2 & -\lambda_1 & \lambda + \tau + s\mu & -s\mu - \tau_1 \\ -v_1 & -v_2 & -v_3 & \dots & \dots & -v_{n-2} & -v_{n-1} & \tau + s\mu \end{pmatrix}.$$

Here

$$\lambda = \sum_{i=1}^{\infty} \lambda_i \quad \text{and} \quad \lambda_i = \lambda p_i$$

and

$$u_1 = \tau \quad \text{and} \quad u_i = \tau - \sum_{k=1}^{i-1} \tau_k \quad \text{for } i = 2, 3, \dots$$

and v_i is defined such that the i th column sum is zero. The generator matrices enjoy the same near-Toeplitz structure. Toeplitz-circulant preconditioners can be constructed similarly and the preconditioned systems are proved to have singular values clustered around one, Ching [48].

Finally, we remark that there is another efficient iterative method for solving queueing systems which is not covered in this text, the multigrid methods. Interested readers may consult the following references Bramble [24], Chan et al. [38], Chang et al. [40] and McCormick [154].

2.2 Search Engines

In this section, we introduce a very important algorithm used by Google for ranking webpages on the Internet. In surfing the Internet, web surfers usually utilize search engines to find the related webpages satisfying their queries. Unfortunately, there are often thousands of webpages which are relevant to their queries. Therefore a proper list of the webpages in a certain order of importance is necessary. The list should also be updated regularly and frequently. Thus it is important to seek for a fast algorithm for updating the PageRank so as to reduce the time lag of updating. As it turns this problem is difficult. The reason is not just because of the huge size of the webpages on the Internet, but also their numbers keep on growing rapidly.

PageRank has been proposed by Page et al. [169] to reflect the importance of each webpage. Larry Page and Sergey Brin are the co-founders of Google. In fact, one can find the following statement at Google's website (<http://www.google.com/technology/>): "The heart of our software is PageRank™, a system for ranking web pages developed by our founders Larry Page and Sergey Brin at Stanford University. And while we have dozens of engineers working to improve every aspect of Google on a daily basis, PageRank continues to provide the basis for all of our web search tools."

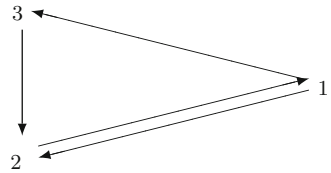
A similar idea of ranking journals has been proposed by Garfield [100, 101] as a measure of standing for journals, which is called the *impact factor*. The impact factor of a journal is defined as the average number of citations per recently published papers in that journal. By regarding each webpage as a journal, this was then extended to measure the importance of a webpage in the PageRank Algorithm.

The PageRank is defined as follows. Let N be the total number of webpages in the web and we define a matrix Q called the *hyperlink matrix*. Here

$$Q_{ij} = \begin{cases} 1/k & \text{if Webpage } i \text{ is an outgoing link of Webpage } j; \\ 0 & \text{otherwise;} \end{cases}$$

and k is the total number of outgoing links of Webpage j . For simplicity of discussion, here we assume that $Q_{ii} > 0$ for all i . This means that for each webpage there is a link pointing to itself. Hence Q can be regarded as a transition probability matrix of a Markov chain of a random walk. The analogy is that one may regard a web surfer as a random walker and the webpages as the states of the Markov chain. Assuming that this underlying Markov chain is irreducible, then the steady-state probability distribution

Fig. 2.6 An example of three webpages



$$(p_1, p_2, \dots, p_N)^T$$

of the states (webpages) exists. Here p_i is the proportion of time that the random walker (web surfer) spends visiting State (Webpage) i . The larger the value of p_i is, the more important Webpage i will be. Thus the PageRank of Webpage i is then defined as p_i . If the Markov chain is not irreducible then one can still follow the treatment in the next subsection.

An Example

We consider a web of an internet with 3 webpages: 1, 2, 3 such that

$1 \rightarrow 1, 1 \rightarrow 2, 1 \rightarrow 3$
 $2 \rightarrow 1, 2 \rightarrow 2,$
 $3 \rightarrow 2, 3 \rightarrow 3.$

One can represent the relationship by the following Markov chain (Fig. 2.6).

The transition probability matrix of this Markov chain is then given by

$$Q = \begin{matrix} & \begin{matrix} 1 & 2 & 3 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} & \begin{pmatrix} 1/3 & 1/2 & 0 \\ 1/3 & 1/2 & 1/2 \\ 1/3 & 0 & 1/2 \end{pmatrix} \end{matrix}.$$

The steady-state probability distribution of the Markov chain

$$\mathbf{p} = (p_1, p_2, p_3)$$

satisfies

$$\mathbf{p} = Q\mathbf{p} \quad \text{and} \quad p_1 + p_2 + p_3 = 1.$$

Solving the above linear system, we get

$$(p_1, p_2, p_3) = \left(\frac{3}{9}, \frac{4}{9}, \frac{2}{9}\right).$$

Therefore the ranking of the webpages is:

$$\text{Webpage 2} > \text{Webpage 1} > \text{Webpage 3}.$$

One can also interpret the result as follows. Both Webpages 1 and 3 point to Webpage 2 and therefore Webpage 2 is the most important. Since webpage 2 points to Webpage 1 but not Webpage 3, Webpage 1 is then more important than Webpage 3.

In terms of the actual Internet, since the size of the Markov chain is huge and the time for computing the PageRank required by Google for computing the Page Rank is just a few days, a direct method for solving the steady-state probability is not desirable. Iterative methods [10] and decomposition methods [7] have been proposed to solve the problem. Another pressing issue is that the number of webpages grows rapidly, and the PageRank of each webpage has to be updated regularly. Here we seek for adaptive and parallelizable numerical algorithms for solving the PageRank problem. One potential method is the hybrid iterative method proposed in Yuen et al. [213]. The hybrid iterative method was first proposed by He et al. [116] for solving the numerical solutions of PDEs and it has also been successfully applied to solving the steady-state probability distributions of queueing networks [213]. The hybrid iterative method combines the evolutionary algorithm and the Successive Over-Relaxation (SOR) method. The evolutionary algorithm allows the relaxation parameter w to be adaptive in the SOR method. Since the cost of the SOR method per iteration is more expensive and less efficient in the parallel computing for our problem (as the matrix system is huge), here we will also consider replacing the role of SOR method by the Jacobi Over-Relaxation (JOR) method [108, 132]. The JOR method is easier to be implemented in parallel computing environments. Here we present hybrid iterative methods based on SOR/JOR method, and the evolutionary algorithm. The hybrid method allows the relaxation parameter w to be adaptive in the SOR/JOR method. We give a brief mathematical discussion on the PageRank approach. We then briefly describe the power method, a popular approach for solving the PageRank problem.

2.2.1 The PageRank Algorithm

The PageRank Algorithm has been used successfully in ranking the importance of webpages by Google (<http://www.search-engine-marketing-sem.com/Google/GooglePageRank.html>). Consider a web of N webpages with Q being the hyper-link matrix. Since the matrix Q can be reducible, to tackle this problem, one can consider the revised matrix P :

$$P = \alpha \begin{pmatrix} Q_{11} & Q_{12} & \cdots & Q_{1N} \\ Q_{21} & Q_{22} & \cdots & Q_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ Q_{N1} & Q_{N2} & \cdots & Q_{NN} \end{pmatrix} + \frac{(1-\alpha)}{N} \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix} \quad (2.15)$$

where $0 < \alpha < 1$. In this case, the matrix P is irreducible and aperiodic, therefore the steady-state probability distribution exists and is unique [181]. Typical values

for α are 0.85 and $(1 - 1/N)$, see for instance [10, 113]. The value $\alpha = 0.85$ is a popular one and the power method works very well for this problem [113–115]. We remark that this value can affect the original ranking of the webpages, see for instance, the example in Sect. 2.2.3.

One can interpret (2.15) as follows. The idea of the algorithm is that for a network of N webpages, each webpage has an inherent importance of $(1 - \alpha)/N$. If a page P_i has an importance of p_i , then it will contribute an importance of αp_i which is shared among the webpages that it points to. The importance of webpage P_i can be obtained by solving the following linear system of equations, subject to the normalization constraint:

$$\begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_N \end{pmatrix} = \alpha \begin{pmatrix} Q_{11} & Q_{12} & \cdots & Q_{1N} \\ Q_{21} & Q_{22} & \cdots & Q_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ Q_{N1} & Q_{N2} & \cdots & Q_{NN} \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_N \end{pmatrix} + \frac{(1 - \alpha)}{N} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}. \quad (2.16)$$

Since

$$\sum_{i=1}^N p_i = 1,$$

(2.16) can be re-written as

$$(p_1, p_2, \dots, p_N)^T = P(p_1, p_2, \dots, p_N)^T.$$

2.2.2 The Power Method

The power method is a popular method for solving the PageRank problem. The power method is an iterative method for solving the largest eigenvalue in modulus (the dominant eigenvalue) and its corresponding eigenvector [108]. The idea of the power method can be briefly explained as follows. Given an $n \times n$ matrix A and suppose that (a) there is a single eigenvalue of maximum modulus and the eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$ are labelled such that

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|;$$

(b) there is a linearly independent set of n unit eigenvectors. This means that there is a basis

$$\{\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \dots, \mathbf{u}^{(n)}\}$$

such that

$$A\mathbf{u}^{(i)} = \lambda_i \mathbf{u}^{(i)}, \quad i = 1, 2, \dots, n, \quad \text{and} \quad \|\mathbf{u}^{(i)}\| = 1.$$

Then beginning with an initial vector $\mathbf{x}^{(0)}$, one may write

$$\mathbf{x}^{(0)} = a_1 \mathbf{u}^{(1)} + a_2 \mathbf{u}^{(2)} + \dots + a_n \mathbf{u}^{(n)}.$$

Now we iterate the initial vector with the matrix A as follows:

$$\begin{aligned} A^k \mathbf{x}^{(0)} &= a_1 A^k \mathbf{u}^{(1)} + \dots + a_n A^k \mathbf{u}^{(n)} = a_1 \lambda_1^k \mathbf{u}^{(1)} + \dots + a_n \lambda_n^k \mathbf{u}^{(n)} \\ &= \lambda_1^k \left\{ a_1 \mathbf{u}^{(1)} + \left(\frac{\lambda_2}{\lambda_1} \right)^k a_n \mathbf{u}^{(2)} + \dots + \left(\frac{\lambda_n}{\lambda_1} \right)^k a_n \mathbf{u}^{(n)} \right\}. \end{aligned}$$

Since

$$\frac{|\lambda_i|}{|\lambda_1|} < 1 \quad \text{for } i = 2, \dots, n,$$

we obtain

$$\lim_{k \rightarrow \infty} \frac{|\lambda_i|^k}{|\lambda_1|^k} = 0 \quad \text{for } i = 2, \dots, n.$$

Hence we have

$$A^k \mathbf{x}^{(0)} \approx a_1 \lambda_1^k \mathbf{u}^{(1)}.$$

To get an approximation for $\mathbf{u}^{(1)}$ we introduce a normalization in the iteration:

$$\mathbf{r}_{k+1} = \frac{A^{k+1} \mathbf{x}^{(0)}}{\|A^k \mathbf{x}^{(0)}\|_2}$$

and we have

$$\lim_{k \rightarrow \infty} \mathbf{r}_{k+1} = \lim_{k \rightarrow \infty} \frac{a_1 \lambda_1^{k+1} \mathbf{u}^{(1)}}{\|a_1 \lambda_1^k \mathbf{u}^{(1)}\|_2} = \lambda_1 \mathbf{u}^{(1)}.$$

It turns out that for the PageRank problem, the largest eigenvalue of P is 1, and the corresponding eigenvector in normalized form is the PageRank vector. The main computational cost of this method comes from the matrix-vector multiplications. The convergence rate of the power method depends on the ratio of $|\lambda_2/\lambda_1|$ where λ_1 and λ_2 are respectively the largest and the second largest eigenvalues of the matrix P . It was proved by Haveliwala and Kamvar [113] that for the second largest eigenvalue of P , we have

$$|\lambda_2| \leq \alpha \quad \text{for } 0 \leq \alpha \leq 1.$$

Since $\lambda_1 = 1$, the convergence rate of the power method is α , see for instance [108]. A popular value for α is 0.85. With this value, it is mentioned in Kamvar et al. [129] that the power method on a web data set of over 80 million pages converges in about 50 iterations.

2.2.3 An Example

In this subsection, we consider a small example of six webpages. This example demonstrates that the value of $\alpha = 0.85$ can affect the original ranking ($\alpha = 1$) of the webpages even if the number of webpages is small. In the example, the webpages are organized as follows:

Webpage 1 \rightarrow 1, 3, 4, 5

Webpage 2 \rightarrow 2, 3, 5, 6

Webpage 3 \rightarrow 1, 2, 3, 4, 5, 6

Webpage 4 \rightarrow 2, 3, 4, 5

Webpage 5 \rightarrow 1, 3, 5

Webpage 6 \rightarrow 1, 6

From the given structure of the webpages, we have the hyperlink matrix as follows:

$$Q = \begin{pmatrix} 0.2500 & 0.0000 & 0.1667 & 0.0000 & 0.3333 & 0.5000 \\ 0.0000 & 0.2500 & 0.1667 & 0.2500 & 0.0000 & 0.0000 \\ 0.2500 & 0.2500 & 0.1667 & 0.2500 & 0.3333 & 0.0000 \\ 0.2500 & 0.0000 & 0.1667 & 0.2500 & 0.0000 & 0.0000 \\ 0.2500 & 0.2500 & 0.1667 & 0.2500 & 0.3333 & 0.0000 \\ 0.0000 & 0.2500 & 0.1667 & 0.0000 & 0.0000 & 0.5000 \end{pmatrix}$$

then the steady-state probability distribution is given by

$$(0.2260, 0.0904, 0.2203, 0.1243, 0.2203, 0.1186)^T$$

and the ranking should be $1 > 3 \geq 5 > \underline{4} > \underline{6} > 2$. For $\alpha = 0.85$, we have

$$P = \begin{pmatrix} 0.2375 & 0.0250 & 0.1667 & 0.0250 & 0.3083 & 0.4500 \\ 0.0250 & 0.2375 & 0.1667 & 0.2375 & 0.0250 & 0.0250 \\ 0.2375 & 0.2375 & 0.1667 & 0.2375 & 0.3083 & 0.0250 \\ 0.2375 & 0.0250 & 0.1667 & 0.2375 & 0.0250 & 0.0250 \\ 0.2375 & 0.2375 & 0.1667 & 0.2375 & 0.3083 & 0.0250 \\ 0.0250 & 0.2375 & 0.1667 & 0.0250 & 0.0250 & 0.4500 \end{pmatrix}.$$

In this case, the steady-state probability distribution is given by

$$(0.2166, 0.1039, 0.2092, 0.1278, 0.2092, 0.1334)^T$$

and the ranking should be $1 > 3 \geq 5 > \underline{6} > \underline{4} > 2$. We observe that the ranking of states 6 and 4 are inter-changed in the two approaches.

2.2.4 The SOR/JOR Method and the Hybrid Method

In this section, we present a hybrid algorithm for solving the steady-state probability of a Markov chain, Yuen et al. [213, 214]. We first give a review on the JOR method for solving linear systems, in particular solving the steady-state probability distribution of a finite Markov chain. We then introduce the hybrid algorithm based on the SOR/JOR method and the evolutionary algorithm. For the SOR method, it has been discussed in Chap. 1. When we consider a non-singular linear system $B\mathbf{x} = \mathbf{b}$, the JOR method is a classical iterative method. The idea of the JOR method can be explained as follows. We write $B = D - (D - B)$ where D is the diagonal part of the matrix B . Given an initial guess of the solution, \mathbf{x}_0 , the JOR iteration scheme reads:

$$\begin{aligned}\mathbf{x}_{n+1} &= (I - wD^{-1}B)\mathbf{x}_n + wD^{-1}\mathbf{b} \\ &\equiv B_w\mathbf{x}_n + wD^{-1}\mathbf{b}.\end{aligned}\tag{2.17}$$

The parameter w is called the relaxation parameter and it lies between 0 and 1 [8]. Clearly if the scheme converges, the limit will be the solution of $B\mathbf{x} = \mathbf{b}$. The choice of the relaxation parameter w affects the convergence rate of the SOR/JOR method significantly, see for instance [213, 214]. In general, the optimal value of w is unknown. For more details about the SOR/JOR method and its properties, we refer readers to [8, 108].

The generator matrix P of an irreducible Markov chain is singular and has a null space of dimension one (the null vector corresponds to the steady-state probability distribution). One possible way to solve the steady-state probability distribution is to consider the following revised system:

$$A\mathbf{x} = (P + \mathbf{e}_n^T \mathbf{e}_n)\mathbf{x} = \mathbf{e}_n^T \tag{2.18}$$

where $\mathbf{e}_n = (0, 0, \dots, 0, 1)$ is a unit vector. The steady-state probability distribution is then obtained by normalizing the solution \mathbf{x} , see for instance, Ching [46]. We remark that the linear system (2.18) is irreducibly diagonal dominant. The hybrid method based on He et al. [116] and Yuen et al. [213] consists of four major steps: *initialization, mutation, evaluation and adaptation* [174].

In the initialization step, we define the size of the population k of the approximate steady-state probability distribution. This means that we also define k approximates to initialize the algorithm. Then we use the JOR iteration in (2.17) as the “mutation step”. In the evaluation step, we evaluate how “good” each member in the population is by measuring their residuals. In this case, it is clear that the smaller the residual the better the approximation and therefore the better the member in the population. In the adaptation step, the relaxation parameters of the “weak” members are migrated (with certain probability) towards the best relaxation parameter. The hybrid algorithm reads:

Step 1: Initialization: We first generate an initial population of k ($2 \leq k \leq n$) identical steady-state probability distributions as follows:

$$\{\mathbf{e}_i : i = 1, 2, \dots, k\}$$

where $\mathbf{e}_i = (0, \dots, 0, \underbrace{1}_{ith\ entry}, 0, \dots, 0)^T$. We then compute

$$r_i = \|B\mathbf{e}_i - \mathbf{b}\|_2$$

and define a set of relaxation parameters $\{w_1, w_2, \dots, w_k\}$ such that

$$w_i = \tau + \frac{(1 - 2\tau)(k - i)}{k - 1}, \quad i = 1, 2, \dots, k.$$

Here $\tau \in (0, 1)$ and therefore $w_i \in [\tau, 1 - \tau]$. We set $\tau = 0.01$ in our numerical experiments. We then obtain a set of ordered triples

$$\{(\mathbf{e}_i, w_i, r_i) : i = 1, 2, \dots, k\}.$$

Step 2: Mutation: The mutation step is carried out by doing a SOR/JOR iteration on each member \mathbf{x}_i (\mathbf{x}_i is used as the initial in the SOR/JOR) of the population with their corresponding w_i . We then get a new set of approximate steady-state probability distributions: \mathbf{x}_i for $i = 1, 2, \dots, k$. Hence we have a new set of

$$\{(\mathbf{x}_i, w_i, r_i) : i = 1, 2, \dots, k\}.$$

Go to Step 3.

Step 3: Evaluation: For each \mathbf{x}_i , we compute and update its residual

$$r_i = \|B\mathbf{x}_i - \mathbf{b}\|_2.$$

This is used to measure how “good” an approximate \mathbf{x}_i is. If $r_j < tol$ for some j then stop and output the approximate steady-state probability distribution \mathbf{x}_j . Otherwise we update r_i of the ordered triples

$$\{(\mathbf{x}_i, w_i, r_i) : i = 1, 2, \dots, k\}$$

and go to Step 4.

Step 4: Adaptation: In this step, the relaxation factors w_k of the weak members (relatively large r_i) in the population are moving towards the best one with certain probability. This process is carried out by first performing a linear search on $\{r_i\}$ to find the best relaxation factor, w_j . We then adjust all the other w_k as follows:

$$w_k = \begin{cases} (0.5 + \delta_1) * (w_k + w_j) & \text{if } (0.5 + \delta_1) * (w_k + w_j) \in [\tau, 1 - \tau] \\ w_k & \text{otherwise,} \end{cases}$$

where δ_1 is a random number in $[-0.01, 0.01]$. Finally the best w_j is also adjusted by

$$w_j = \delta_2 * w_j + (1 - \delta_2) * \frac{(w_1 + w_2 + \dots + w_{j-1} + w_{j+1} + \dots + w_k)}{k - 1}$$

where δ_2 is a random number in $[0.99, 1]$. A new set of $\{w_i\}$ is then obtained and hence

$$\{(\mathbf{x}_i, w_i, r_i) : i = 1, 2, \dots, k\}.$$

Go to Step 2.

2.2.5 Convergence Analysis

In this section, we consider the linear system $B\mathbf{x} = \mathbf{b}$ where B is strictly diagonal dominant, i.e.

$$|B_{ii}| > \sum_{j=1, j \neq i}^N |B_{ij}| \quad \text{for } i = 1, 2, \dots, N$$

where N is the size of the matrix.

We first prove that the hybrid algorithm converges for a range of w with the SOR method. We begin with the following lemma.

Lemma 2.3. *Let B be a strictly diagonal dominant square matrix and*

$$K = \max_i \left\{ \sum_{j=1, j \neq i}^m \frac{|B_{ij}|}{|B_{ii}|} \right\} < 1,$$

then

$$\|B_w\|_{M_\infty} < 1 \quad \text{for } 0 < w < 2/(1 + K)$$

where B_w is defined in (2.15). Here the definition of the matrix $\|\cdot\|_{M_\infty}$ can be found in Sect. 1.3.1.

Proof. Let \mathbf{x} be an $n \times 1$ vector such that $\|\mathbf{x}\|_\infty = 1$. We are going to prove that

$$\|B_w \mathbf{x}\|_\infty \leq 1 \quad \text{for } 0 < w < 2/(1 + K).$$

Consider

$$\mathbf{y} = (D - wL)^{-1}((1 - w)D + wU)\mathbf{x}$$

and we have

$$(D - wL)\mathbf{y} = ((1 - w)D + wU)\mathbf{x}$$

i.e.,

$$\begin{pmatrix} B_{11} & 0 & \cdots & \cdots & 0 \\ -wB_{21} & B_{22} & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & 0 \\ -wB_{m1} & \cdots & \cdots & -wB_{m,m-1} & B_{mm} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} (1-w)B_{11} & wB_{12} & \cdots & \cdots & wB_{1m} \\ 0 & (1-w)B_{22} & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & wB_{m-1,m} \\ 0 & \cdots & \cdots & 0 & (1-w)B_{mm} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_m \end{pmatrix}.$$

Case 1: $1 \leq w < 2/(K + 1)$.

For the first equation, we have

$$B_{11}y_1 = (1 - w)B_{11}x_1 + w \sum_{j=2}^m B_{1j}x_j.$$

Since

$$|x_i| \leq 1 \quad \text{and} \quad \sum_{j=2}^m |B_{1j}| < K|B_{11}|,$$

we have

$$|y_1| \leq |1 - w| + wK = w(1 + K) - 1 < 1.$$

For the second equation, we have

$$B_{22}y_2 = (1 - w)B_{22}x_2 + wB_{21}y_1 + w \sum_{j=3}^m B_{2j}x_j.$$

Since

$$|y_1| \leq 1, \quad |x_i| \leq 1$$

and

$$\sum_{j=1, j \neq 2}^m |B_{2j}| < K|B_{22}|,$$

we have

$$|y_2| \leq |1 - w| + wK = w(1 + K) - 1 < 1.$$

Inductively, we have $|y_i| < 1$ and hence $\|\mathbf{y}\|_\infty < 1$. Therefore we have proved that

$$\|B_w\|_{M_\infty} < 1 \quad \text{for} \quad 1 \leq w < 2/(1 + K).$$

Case 2: $0 < w < 1$.

For the first equation, we have

$$B_{11}y_1 = (1 - w)B_{11}x_1 + w \sum_{j=2}^m B_{1j}x_j.$$

Since

$$|x_i| \leq 1 \quad \text{and} \quad \sum_{j=2}^m |B_{1j}| < |B_{11}|,$$

we have

$$|y_1| < 1 - w + w = 1.$$

For the second equation, we have

$$B_{22}y_2 = (1 - w)B_{22}x_2 + wB_{21}y_1 + w \sum_{j=3}^m B_{2j}x_j.$$

Since

$$|y_1| \leq 1, |x_i| \leq 1 \quad \text{and} \quad \sum_{j=1, j \neq 2}^m |B_{2j}| < |B_{22}|,$$

we have

$$|y_2| < 1 - w + w = 1.$$

Inductively, we have $|y_i| < 1$ and hence $\|\mathbf{y}\|_\infty < 1$. Therefore

$$\|B_w\|_{M_\infty} < 1 \quad \text{for} \quad 0 < w < 1.$$

Combining the results, we have

$$\|B_w\|_{M_\infty} < 1 \quad \text{for} \quad 0 < w < 2/(1 + K).$$

Proposition 2.4. *The hybrid algorithm converges for $w \in [\tau, 2/(1 + K) - \tau]$ where $0 < \tau < 1/(1 + K)$.*

Proof. We note that

$$f(\tau) = \max_{w \in [\tau, 2/(1+K) - \tau]} \{ \|B_w\|_{M_\infty} \}$$

exists and is less than one. Let us denote it by $0 \leq f(\tau) < 1$. Therefore in each iteration of the hybrid method, the matrix norm ($\|\cdot\|_{M_\infty}$) of the residual is decreased by a fraction not less than $f(\tau)$. By using the fact that

$$\|ST\|_{M_\infty} \leq \|S\|_{M_\infty} \|T\|_{M_\infty},$$

the hybrid algorithm is convergent.

We then prove that the hybrid algorithm with the JOR method converges for a range of w . We have the following lemma.

Lemma 2.5. *Let B be a strictly diagonal dominant square matrix and*

$$K = \max_i \left\{ \sum_{j=1, j \neq i}^N \frac{|B_{ji}|}{|B_{ii}|} \right\} < 1,$$

then

$$\|B_w\|_{M_1} \leq 1 - (1 - K)w < 1 \quad \text{for } \tau < w < 1 - \tau$$

where B_w is defined in (2.15). Here the definition of the matrix norm $\|\cdot\|_{M_1}$ can be found in Sect. 1.3.1.

By using the similar approach as in Proposition 2.4, one can prove the following proposition.

Proposition 2.6. *The hybrid iterative method converges for $w \in [\tau, 1 - \tau]$.*

Proof. We observe that

$$f(\tau) = \max_{w \in [\tau, 1 - \tau]} \{ \|B_w\|_{M_1} \}$$

exists and is less than one. Let us denote it by $0 \leq f(\tau) < 1$. Therefore in each iteration of the hybrid method, the matrix norm ($\|\cdot\|_{M_1}$) of the residual is decreased by a fraction not less than $f(\tau)$. By using the fact that

$$\|ST\|_{M_1} \leq \|S\|_{M_1} \|T\|_{M_1},$$

the hybrid algorithm is convergent.

We note that the matrix A in (2.16) is irreducibly diagonal dominant only, but not strictly diagonal dominant. Therefore the conditions in Lemmas 2.3 and 2.5 are not satisfied. However, one can always consider a regularized linear system as follows:

$$(A + \epsilon I)\mathbf{x} = \mathbf{b}.$$

Table 2.1 Number of iterations for convergence ($\alpha = 1 - 1/N$)

JOR	Data set 1				Data set 2				Data set 3			
N	100	200	300	400	100	200	300	400	100	200	300	400
$k = 2$	41	56	42	42	57	95	58	70	31	26	32	25
$k = 3$	56	60	42	42	56	75	57	61	31	35	43	25
$k = 4$	46	59	42	42	55	72	58	62	31	32	38	25
$k = 5$	56	60	43	43	56	68	57	60	32	30	36	26

SOR	Data set 1				Data set 2				Data set 3			
N	100	200	300	400	100	200	300	400	100	200	300	400
$k = 2$	20	18	17	17	16	15	16	15	18	14	19	15
$k = 3$	30	27	17	25	16	23	16	23	18	21	29	15
$k = 4$	25	24	19	22	17	21	16	21	18	19	26	18
$k = 5$	30	28	19	23	17	21	16	20	20	20	25	17

Table 2.2 Number of iterations for convergence ($\alpha = 0.85$)

JOR	Data set 1				Data set 2				Data set 3			
N	100	200	300	400	100	200	300	400	100	200	300	400
$k = 2$	42	56	44	47	61	82	66	64	18	28	32	26
$k = 3$	55	60	45	52	62	81	63	62	18	36	42	26
$k = 4$	53	59	45	49	58	71	62	62	18	33	38	26
$k = 5$	53	65	45	49	61	70	64	62	18	32	37	26

SOR	Data set 1				Data set 2				Data set 3			
N	100	200	300	400	100	200	300	400	100	200	300	400
$k = 2$	19	17	17	16	16	14	15	15	15	14	19	16
$k = 3$	28	26	17	24	16	22	15	23	15	23	29	16
$k = 4$	24	23	19	21	16	20	16	21	17	20	25	16
$k = 5$	28	26	19	21	17	21	16	20	16	20	23	16

Here I is the identity matrix and $\epsilon > 0$ can be chosen as small as desired. Then the matrix $(A + \epsilon I)$ is strictly diagonal dominant, although this will introduce a small error of $O(\epsilon)$ to the linear system. Numerical results in Yuen et al. [213,214] indicate that the hybrid algorithm is very efficient in solving for the steady-state probability distribution of queueing systems and ranking webpages in the Web. Here we present some small scale numerical results (three different data sets) for two typical values of α in Tables 2.1 and 2.2 (Taken from [214]). Here k is the size of population and N is the number of webpages.

2.3 Summary

In this chapter, we discussed two important applications of Markov chains: the classical Markovian queueing networks and the Modern PageRank algorithm. The latter application comes from the measurement of prestige in a network. The

computation of prestige in a network is an important issue [22, 23] and has many other applications, such as social networks [206] and disease transmission [12]. A number of methods based on the computation of eigenvectors have been proposed in the literature, see for instance Langville and Meyer [139, 140] and the references therein. Further research can be done in developing models and algorithms for the case when there are negative relations in the network [192]. In a network, being chosen or nominated by a popular or powerful person (webpage) would add to one's popularity. Instead of supporting a member, a negative relation means being rejected by a member in the network.

2.4 Exercise

1. We consider an $M/M/1/\infty$ queue. The mean service time of the exponential server is μ^{-1} and the customers arrive according to a Poisson process with mean rate λ where $\lambda < \mu$. An arrived customer may leave the system directly with a probability of

$$\frac{i}{i+1} \quad i = 0, 1, \dots,$$

when they find there are already i customers in the system.

- (a) Find the steady-state probability distribution of the queueing system in terms of λ and μ .
 - (b) Assuming that the queueing system is in steady-state, find the loss probability (probability that an arrived customer chooses not to join the queueing system).
 - (c) If the mean service time μ^{-1} can be assigned as fast as possible (i.e. $\lambda < \mu$), recommend the service rate μ if the loss probability has to be less than 1 %.
2. Consider an exponential server system with service rate μ having no waiting space. Customers arrive according to a Poisson process with mean rate λ . As the server is failure-prone, the normal time of the server follows the exponential distribution with mean θ^{-1} . When it breaks down, it will be repaired at once. The repair time of the server follows an exponential distribution with mean γ^{-1} . The following are the four possible states of the system

$$\{(N, 0), (D, 0), (N, 1), (D, 1)\}$$

where N represents the server is normal, D represents the server is down, 0 represents the system is idle and 1 represents the system is busy.

- (a) Write down the generator matrix of this Markov process.
- (b) Obtain the steady-state probability distribution of the system.

- (c) Assuming that the system is in steady-state, find the conditional probability that the server is under repair given that the system is idle.
3. (a) We consider a system of n identical failure-prone machines. A repairman is assigned to repair the broken machines. The mean normal time and the repair time of a machine are exponentially distributed with means λ^{-1} and μ^{-1} respectively. The system is said to be in state i ($i = 0, 1, \dots, n$) if there are i broken machines.
- (a) Write down the Markov chain and the generator matrix for the states of this machine repairing model.
- (b) Find the steady-state probability p_i that there are i broken machines in the system.
- (c) Find the steady-state probability that the repairman is idle.
4. A company is going to hire a repairman to look after four identical unreliable machines. The machines break down randomly according to a Poisson process of mean rate of 3 per day. The non-productive time of any machine costs the firm 200 dollars per day. The firm can hire a slow, cheap repairman charging 500 dollars per day who repairs at an average rate of 4 machines per day. Alternatively the firm can hire a fast, experienced repairman charging 1,000 dollars per day who repairs at an average rate of 5 machines per day. In either case repair time is assumed to be exponentially distributed. Compute the expected running costs in both cases and determine which repairman should be hired.
5. Consider the allocation problem discussed in Sect. 2.1.3.
- (a) Apply the result in (2.4) to show that the expected number of customers waiting in Queue i is given by

$$\frac{(\lambda_i / \mu_i)^2}{1 - \lambda_i / \mu_i}.$$

Hence show that the total expected number of customers waiting in the system is given by

$$\sum_{i=1}^n \frac{(\lambda_i / \mu_i)^2}{1 - \lambda_i / \mu_i}.$$

- (b) Solve the following optimization problem:

$$\min_{\lambda_1, \dots, \lambda_n} \left\{ \sum_{i=1}^n \frac{(\lambda_i / \mu_i)^2}{1 - \lambda_i / \mu_i} \right\}.$$

subject to

$$\sum_{i=1}^m \lambda_i = \lambda$$

and

$$0 \leq \lambda_i < \mu \quad \text{for } i = 1, 2, \dots, n$$

and get the optimal allocation.

6. Prove the formula (2.7).
7. In a post office, postmen must retrieve their assignments from the office which is managed by an assistant. Suppose the mean number of postmen retrieving their assignments per hour is 4 and they are paid 10 dollars per hour. There are two possible assistants A or B to be employed for managing the office. On average assistant A takes 11.5 min to handle one request and is paid 9.5 dollars per hour. While assistant B takes 12.5 min to handle one request and is paid 7 dollars per hour. Assume that the inter-arrival time of postmen and the processing time of the assistants are exponentially distributed. Which assistant should be employed?
8. Let

$$P = \begin{pmatrix} 0.75 & 0.25 & 0 & 0 \\ 0.2 & 0.5 & 0.2 & 0.1 \\ 0.3 & 0.25 & 0.25 & 0.2 \\ 0 & 0.15 & 0.25 & 0.6 \end{pmatrix}.$$

Apply the power method for 10 iterations to get an approximate of the steady-state probability distribution.

9. Apply the power method to the following $n \times n$ transition probability matrix

$$P = \begin{pmatrix} 0.75 & 0.25 & 0 & \cdots & \cdots & 0 \\ 0.25 & 0.50 & 0.25 & \ddots & & \vdots \\ 0 & 0.25 & 0.50 & 0.25 & \ddots & \vdots \\ 0 & & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & 0.25 & 0.50 & 0.25 \\ 0 & \cdots & \cdots & 0 & 0.25 & 0.75 \end{pmatrix} \quad (2.19)$$

to get an approximation of the steady-state probability distribution for $n = 10, 20, 40, 80, 160, 320$. Record the number of iterations required for the convergence under the following stopping criterion

$$\|\mathbf{x}_k - \mathbf{p}\|_1 < 10^{-6}.$$

Here \mathbf{p} is the true solution and \mathbf{x}_k is the approximate solution obtained in k th iteration. We remark that the true solution of the steady-state probability distribution is given by

$$\mathbf{p} = \frac{1}{n}(1, 1, \dots, 1)^T.$$

10. Consider a general PageRank matrix

$$P = (cQ + (1 - c)\mathbf{1}\mathbf{u}^T) \quad \text{for } c \in [0, 1)$$

where \mathbf{u} is a positive probability distribution vector.

- (a) By using Sherman-Morrison-Woodbury's formula, show that the stationary distribution is given by

$$\mathbf{x} = (1 - c)(I - cP)^{-1}\mathbf{u}.$$

- (b) Hence show that

$$\mathbf{x} = (1 - c) \sum_{n=0}^{\infty} c^n (P^n \mathbf{u}).$$



<http://www.springer.com/978-1-4614-6311-5>

Markov Chains

Models, Algorithms and Applications

Ching, W.-K.; Huang, X.; Ng, M.K.; Siu, T.K.

2013, XVI, 243 p., Hardcover

ISBN: 978-1-4614-6311-5