Chapter 17

Graph-Theoretic Analysis of Finite Markov Chains

J. P. Jarvis
D. R. Shier

17.1 Introduction

Markov chains arise frequently in the modeling of physical and conceptual processes that evolve over time. For example, the diffusion of liquids across a semiporous membrane, the spread of disease within a population, and the flow of personnel within the ranks of an organization can all be modeled using Markov chains. In each of these cases, the system can be found in any of a finite number of states, and transitions between states occur at discrete instants according to specified probabilities.

As one illustration, suppose that there are M molecules in a vessel, separated into two chambers by a membrane, across which molecules can pass. A typical configuration of the system at any instant can be described by the distribution of the M molecules between the two chambers. If there are k_1 molecules in the first chamber, then there will be $k_2 = M - k_1$ molecules in the second chamber. Transitions from the current state (k_1, k_2) can occur by the movement of a single molecule from the first chamber to the second, or from the second chamber to the first. These two new states are represented by $(k_1 - 1, k_2 + 1)$ and $(k_1 + 1, k_2 - 1)$, respectively. In one possible model of this process, the probability of transition from (k_1, k_2) to $(k_1 - 1, k_2 + 1)$ is given by k_1/M, whereas the probability of transition to $(k_1 + 1, k_2 - 1)$ is $k_2/M = 1 - k_1/M$. This quantifies the idea that if more molecules are present in (say) chamber 1, then it is more likely for some molecule to transfer next from chamber 1 to chamber 2. Using this mathematical model, one can answer questions such as: (a) under what conditions do the molecules achieve an equilibrium configuation, (b) what are the (probabilistic) characteristics of this configuration, and (c) at what rate is this equilibrium approached?

The above is an instance of a finite-state Markov chain, which is the topic of the present chapter. Normally, this subject is presented in terms of the (finite) matrix describing the Markov chain. Our objective here is to supplement this viewpoint with a graph-theoretic approach, which provides a useful visual representation of the process. A number of important properties of the Markov chain (typically derived using matrix manipulations) can be deduced from this pictorial representation. Moreover, certain concepts from modern algebra will also be illuminated by developing this approach. In addition, the graph-theoretic rep-

resentation immediately suggests several computational schemes for calculating important structural characteristics of the underlying problem. This chapter indicates how appropriate data structures and algorithms enable such calculations to be carried out in an efficient manner.

17.2 State Classification

In this section some basic concepts of finite Markov chains are defined, leading to the notion of classifying states of the chain as either "recurrent" or "transient." Both algebraic and graph-theoretic approaches turn out to be useful in identifying such states. Algorithmic techniques for carrying out state classification are also discussed.

17.2.1 Preliminaries

Suppose that $\mathcal{M}$ is a finite-state Markov chain with states $\{1, 2, ..., n\}$. At every (discrete) instant t the chain $\mathcal{M}$ will be in one of these states. The quantity p_{ij} denotes the conditional probability that $\mathcal{M}$ will be in state j at time $t+1$, given that it was observed in state i at time t. Implicitly, we are assuming that the Markov chain is *homogeneous*: namely, the values p_{ij} are independent of time t. These (one-step) state transition probabilities define the *transition probability matrix* $P = [p_{ij}]$. In general, let p_{ij}^k denote the probability that $\mathcal{M}$ proceeds from state i to state j after k transitions. Then the *k-step transition probability matrix* $P^{(k)} = [p_{ij}^k]$ is given by $P^{(k)} = P^k$, the k-th matrix power of the transition probability matrix P. Notice that when $k = 0$ this produces $P^{(0)} = P^0 = I_n$, the $n \times n$ identity matrix, agreeing with the observation that after $k = 0$ steps the Markov chain is still in its initial state. The Markov chain is called *irreducible* if, for every pair of states i and j, there exist $r, s \geq 0$ with $p_{ij}^r > 0$ and $p_{ji}^s > 0$.

Figure 1 gives the transition probability matrix P for a five-state Markov chain, on the states $1, 2, 3, 4, 5$. Also shown is the third power P^3 of P. Accordingly, the conditional probability of being in state 4 at time 5, given that the system is observed in state 3 at time 2, is $p_{34}^3 = 0.186$. As will be seen later, this chain is not irreducible.

$$P = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ .5 & 0 & .2 & .3 & 0 \\ .4 & .1 & 0 & .2 & .3 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}, \quad P^3 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} .220 & .030 & .540 & .120 & .090 \\ 0 & 0 & 0 & 0 & 1 \\ .318 & .102 & .328 & .186 & .066 \\ .216 & .164 & .160 & .128 & .332 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

FIGURE 1 A transition probability matrix P

Associated with the Markov chain $\mathcal{M}$ is a *digraph* (directed graph) $G = G_{\mathcal{M}}$ having the set of nodes $N = \{1, 2, ..., n\}$ and the set of edges E. Each node corresponds to a state of $\mathcal{M}$, and G contains edge $(i, j) \in E$ if and only if $p_{ij} > 0$. Thus the digraph, or *state transition diagram*, G captures the structure of the possible one-step state transitions; the actual numerical values of the state transition probabilities are ignored. In graph-theoretic terms, $p_{ij}^k > 0$ means there is a directed *path* Q of *length* $l(Q) = k$ (number of edges) from node i to node j in G. If this holds for some $k \geq 0$, then node j is *accessible* from node i, written $i \rightarrow j$. Observe that $i \rightarrow i$ since we consider node i to be reachable from itself by a path of length 0. If there is no path in G from i to j, then we write $i \not\rightarrow j$. If both $i \rightarrow j$ and $j \rightarrow i$ hold, then we say that states i and j *communicate*, written $i \leftrightarrow j$. A path joining a node to itself is called a *circuit*. If this circuit contains no repeated nodes, then it is a *cycle*. Figure 2 shows the state transition diagram G for the Markov chain in Figure 1. Notice that there is a path from node 1 to node 2, but no path from node 2 to node 1; thus $1 \rightarrow 2$ whereas $2 \not\rightarrow 1$. The node sequence $[1, 3, 4, 1]$ defines a cycle in G. It is also seen that $1 \leftrightarrow 3$, $1 \leftrightarrow 4$ and $3 \leftrightarrow 4$.

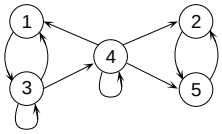

FIGURE 2 The state transition diagram G

An important concept in the analysis of Markov chains is the categorization of states as either *recurrent* or *transient*. The Markov chain, once started in a recurrent state, will return to that state with probability 1. However, for a transient state there is some positive probability that the chain, once started in that state, will never return to it. This same concept can be illuminated using graph-theoretic concepts, which do not involve the numerical values of

probabilities. Namely, we define node i to be transient if there exists some node j for which $i \rightarrow j$ but $j \not\rightarrow i$. Otherwise, node i is called recurrent. (These definitions apply only to finite Markov chains.) Let $\mathcal{T}$ denote the set of all transient nodes, and let $\mathcal{R} = N - \mathcal{T}$ be the set of all recurrent nodes.

17.2.2 Algebraic and Graph-Theoretic Considerations

Certain states in a Markov chain behave in a similar fashion. The following algebraic fact underlies this type of "state classification."

> *Lemma 1*: The relation $\leftrightarrow$ is an equivalence relation: namely, (a) $i \leftrightarrow i$ for all $i \in N$; (b) if $i \leftrightarrow j$ then $j \leftrightarrow i$ for all $i, j \in N$; and (c) if $i \leftrightarrow j$ and $j \leftrightarrow k$ then $i \leftrightarrow k$ for all $i, j, k \in N$.

Consequently, the equivalence relation $\leftrightarrow$ partitions N into a number of equivalence classes (communicating classes): disjoint sets whose union is N. Notice that $\mathcal{M}$ is irreducible precisely when there is just a single equivalence class under $\leftrightarrow$. A useful result is that nodes within a given equivalence class do indeed behave similarly.

> *Lemma 2*: If node i is recurrent and $i \leftrightarrow j$ then node j is recurrent. If node i is transient and $i \leftrightarrow j$ then node j is transient.

In the theory of directed graphs, G is called *strongly connected* if there is a path between any pair of nodes i, j in G. In other words, $i \rightarrow j$ holds for all i, j, meaning that $i \leftrightarrow j$ for all i, j. Thus an irreducible Markov chain $\mathcal{M}$ is simply one whose digraph G is strongly connected. In general, the communicating classes of $\mathcal{M}$ are just the maximal strongly connected subgraphs of G — the *strong components* of G. It is known that if nodes i and j lie on a common circuit, then they belong to the same strong component, and conversely. As a consequence, when the nodes within each strong component are combined into a new "supernode," then the *condensed* graph $\hat{G}$ governing these supernodes can contain no cycles — $\hat{G}$ is an *acyclic* graph.

For example, the state transition diagram in Figure 2 has two strong components: $K_1 = \{1, 3, 4\}$ and $K_2 = \{2, 5\}$. Thus the Markov chain $\mathcal{M}$ is not irreducible. For the larger state transition diagram shown in Figure 3, the four strong components are $K_1 = \{2, 5, 6\}$, $K_2 = \{3, 4\}$, $K_3 = \{1, 7\}$ and $K_4 = \{8\}$. The condensed graph $\hat{G}$ on these components is also displayed. All nodes in components K_2 and K_4 are recurrent, whereas all nodes in components K_1 and K_3 are transient. Recall that Lemma 2 assures us that all nodes within a strong component have the same classification. This example suggests that the recurrent components are precisely those with no leaving edges in the graph $\hat{G}$. This is true in general.

Lemma 3: The recurrent nodes of graph G are precisely those nodes whose corresponding supernodes have no leaving edges in $\hat{G}$.

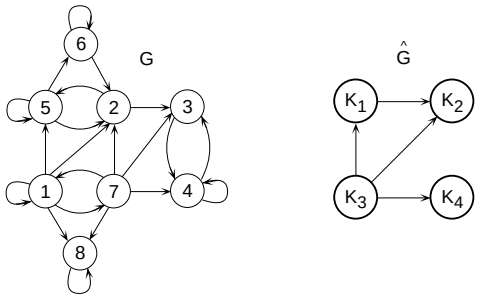

FIGURE 3 A sample digraph G with its condensed graph

In view of Lemma 3, inspection of the state transition diagram G of the finite chain $\mathcal{M}$ allows one to classify each node as either recurrent or transient. In addition, these concepts are important in connection with *stationary distributions* $\pi = [\pi_1, \pi_2, \ldots, \pi_n]$ for a Markov chain having states $N = \{1, 2, ..., n\}$. These probabilities represent the long run proportion of time the chain $\mathcal{M}$ spends in each state. Such probabilities can be found by solving the linear system $\pi = \pi P$, $\sum_{j \in N} \pi_j = 1$. The following standard results in the theory of Markov chains are stated in terms of the state transition diagram G for $\mathcal{M}$.

Theorem 1: If G is strongly connected then there is a unique stationary distribution π for $\mathcal{M}$. Moreover, this distribution satisfies $\pi_j > 0$ for all $j \in N$.

Theorem 2: If the condensed graph $\hat{G}$ for G has a single supernode with no leaving edges then there is a unique stationary distribution π for $\mathcal{M}$. Moreover, this distribution satisfies $\pi_j > 0$ for all $j \in \mathcal{R}$, and $\pi_j = 0$ for all $j \in \mathcal{T}$.

17.2.3 Depth-First Search Algorithm

A natural question concerns computational methods for identifying the recurrent and transient nodes of G. Fortunately, this can be done very efficiently by use of a depth-first search (or, pre-order traversal) of the digraph $G = (N, E)$. A depth-first search produces a numbering or labeling of the nodes of G, where each node is numbered in the order in which it is first encountered. This numbering can then be used in conjunction with the depth-first search to produce the

strong components of G. In general, the entire procedure has time complexity $O(n + m)$ where $n = |N|$ and $m = |E|$.

To carry out a depth-first search on G, an arbitrary node $v_0 \in N$ is selected as the starting node and is labeled as $\text{lab}(v_0) = 1$. The algorithm will end up labeling all vertices that are accessible from v_0. When the algorithm terminates, there may be unlabeled nodes (not accessible from the starting node v_0). In that case, the algorithm is restarted with an unlabeled node. Continuing in this manner, eventually all nodes will be labeled. For simplicity, the algorithm is now described for a graph G in which all nodes are accessible from the initially selected node v_0.

The algorithm labels the nodes of G with the numbers $1, 2, \ldots, n$ according to the order in which they are first encountered, starting with $\text{lab}(v_0) = 1$. In addition, edges are partitioned into two sets: *tree* and *non-tree* edges. As each newly labeled node v is processed, an edge (v, w) is considered. If node w has not yet been labeled, then w becomes labeled with $\text{lab}(w) = \text{lab}(v) + 1$, edge (v, w) is classified as a tree edge, and processing continues with the newly labeled node w. If w has already been labeled, (v, w) is a non-tree edge and can be further classified as a *back edge*, *forward edge*, or *cross edge*. Back edges (v, w) have $\text{lab}(v) > \text{lab}(w)$ with v a descendant of w in the tree; forward edges (v, w) have $\text{lab}(v) < \text{lab}(w)$ with w a descendant of v; cross edges (v, w) have $\text{lab}(v) > \text{lab}(w)$ with w neither a descendant nor ancestor of v. (See Figure 4.)

When all edges incident from v have been considered, processing is continued using the node u that was used to label node v. (Note that there is a unique tree edge incident to v: namely, (u, v).) Assuming that all nodes are accessible from the initial node v_0, the depth-first search algorithm will terminate when all nodes have been labeled and all edges have been classified.

Figure 4 shows a depth-first search tree rooted at node 1 for the state transition diagram given in Figure 3. Self-loops on nodes have been ignored, and the labels of nodes in Figure 4 turn out to be the original node numbers.

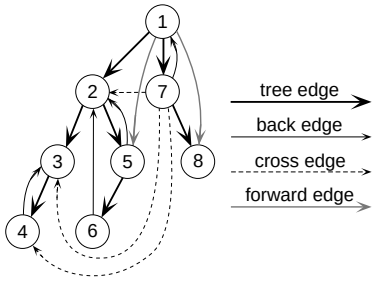

FIGURE 4 Types of edges in a depth-first search of a digraph

At the same time as a depth-first search is carried out on G, additional information can be collected to enable identification of the strong components of G. Our objective is not to describe the intricacies of this modification; the reader is referred to Aho et al. [1] for details. Rather we briefly describe here the overall strategy of this approach for finding strong components.

It is not difficult to see that all nodes in the same strong component will have a common ancestor (within that component) relative to a depth-first search tree. The minimum label common ancestor (for a particular depth-first search tree) is referred to as the *root* of the strong component. Suppose G has strong components $G_1, G_2, \ldots, G_k$ and let r_i denote the root associated with G_i. The components are numbered in the order that the depth-first search from each root node is completed. The following lemma identifies the nodes in the strong components of G.

> *Lemma 4*: The nodes in strong component G_i consist of those nodes that are descendants of r_i but are not in $G_k, 1 \leq k < i$.

In Figure 4, the root nodes turn out to be $3, 2, 8, 1$. Notice that the descendants of node 3 are $\{3, 4\}$ while the descendants of node 2 are $\{2, 3, 4, 5, 6\}$, producing the strong components $G_1 = \{3, 4\}$ and $G_2 = \{2, 5, 6\}$. Similarly, $G_3 = \{8\}$ is the strong component derived from root node 8, and $G_4 = \{1, 7\}$ is the strong component derived from root node 1. These components correspond to supernodes K_2, K_1, K_4, and K_3 respectively in Figure 3.

Thus the real task is to identify the roots of each strong component. The algorithm in [1] does this by executing a depth-first search and keeping auxiliary information for each node (which is updated whenever non-tree edges are encountered). Overall, the strong components can be found (and states classified as recurrent or transient) very efficiently, in $O(n + m)$ time. By using appropriate data structures for representing G, the storage required by the entire algorithm is also $O(n + m)$. Recall that n is the number of states of the Markov chain and m is the number of edges in G or equivalently the number of non-zero transition probabilities in P. Accordingly, the sparsity of the transition probability matrix P can be exploited computationally by this algorithmic approach.

17.3 Periodicity

In this section, the important concept of periodicity is explored. This concept is quite useful in understanding the "limiting behavior" of a Markov chain $\mathcal{M}$. Suppose that there is positive probability that $\mathcal{M}$, started in state i, will return to state i in a finite number of steps. That is, there is some $k > 0$ such that

$p_{ii}^k > 0$. Then the *period* of state i is the largest integer d such that $p_{ii}^k = 0$ whenever k is not a positive integer multiple of d (that is, $k \neq d, 2d, 3d, \ldots$). Note that p_{ii}^k is allowed to be positive only when k is a multiple of d, but it is not necessary that all such p_{ii}^k be positive. A state with period $d = 1$ is called *aperiodic*.

For the example of Figure 5(a), returns to state 1 can only occur at steps $k = 3, 6, 9, 12, \ldots$, so $p_{11}^k = 0$ for k not a multiple of 3; consequently state 1 has period $d = 3$. In Figure 5(b), returns to state 1 can only occur at steps $k = 8, 10, 16, 18, 20$ and even $k \geq 24$; state 1 has period $d = 2$. Notice that in this case, p_{ii}^k is not positive for all even $k > 0$; namely, $p_{ii}^k = 0$ for $k = 2, 4, 6, 12, 14, 22$.

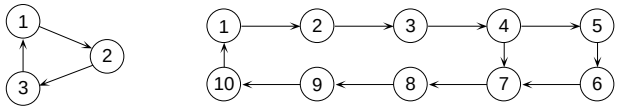

FIGURE 5 Examples of periodic Markov chains

17.3.1 Analytical Results

Here we state some standard facts concerning the period of a Markov chain, indicative of the importance of this concept in analyzing the long term behavior of the chain. The remainder of this subsection focuses on setting up the machinery needed for the efficient computation of the period d. The following result shows that all states in the same communicating class have the same period.

Lemma 5: If state i has period d and $i \leftrightarrow j$ then state j has period d.

When viewed in terms of the state transition diagram G, the period of strong component K_i is just the greatest common divisor (gcd) of the lengths of all directed circuits in the subgraph induced by the nodes of K_i. Equivalently, the period of K_i is the gcd of the lengths of all directed cycles in this subgraph. For the state transition diagram of Figure 2, all states of $K_1 = \{1, 3, 4\}$ are aperiodic while all those of $K_2 = \{2, 5\}$ have period 2.

Let $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_n]$ be the vector of *initial probabilities* of being in any of the n states at time $t = 0$. Then the vector αP^k gives the absolute probability of being in each state at time $t = k$. Under certain conditions, this probability vector approaches a vector of *limiting probabilities* as $t \to \infty$.

Theorem 3: Suppose that the condensed graph $\hat{G}$ for G has a single supernode K with no leaving edges and that all states of K are aperiodic.

Then limiting probabilities exist and are independent of the initial probability vector α. Moreover, these limiting probabilities coincide with the (unique) stationary distribution for the chain.

A special but important case of Theorem 3 occurs when G is strongly connected and all states are aperiodic. Indeed, for the remainder of this section it will be assumed that G is strongly connected ($\mathcal{M}$ is irreducible). Lemma 5 then assures us that every state of $\mathcal{M}$ has the same period d. If $d = 1$ then $\mathcal{M}$ will be called *aperiodic*; if $d \geq 2$ then $\mathcal{M}$ will be called *periodic*.

The period d has been defined in terms of the gcd of the lengths of all cycles in G. Now we explore another graph-theoretic characterization of the period that will be useful in developing an efficient algorithm for computing d. The following result characterizes the period d of $\mathcal{M}$ through a decomposition of G into "cyclically moving classes."

Theorem 4: $\mathcal{M}$ has period d if and only if its digraph G can be partitioned into d sets $C_0, C_1, \ldots, C_{d-1}$ such that (a) if $i \in C_k$ and $(i, j) \in E$ then $j \in C_{(k+1) \bmod d}$; and (b) d is the largest integer with this property.

Property (a) says that starting with any state in set C_k, the next transition must be to a state in C_{k+1}, then to a state in C_{k+2}, and so on, where the succeeding set index is taken modulo d. This situation is depicted in Figure 6. If $\mathcal{M}$ is aperiodic ($d = 1$), then there is a single set C_0 containing all states.

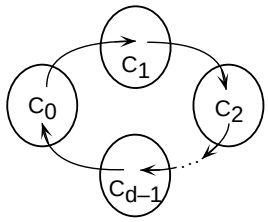

FIGURE 6 State transitions in a Markov chain with period *d*

It is easy to see that if the digraph G is partitioned into d sets with properties (a) and (b) then d must be the period of $\mathcal{M}$. Since all paths in G proceed from C_k to C_{k+1} to C_{k+2}, and so forth, every path from i to i must have length 0 modulo d. In $\mathcal{M}$ this means that if $p_{ii}^r > 0$, then $r \bmod d = 0$. Hence the period of the chain must be a multiple of d. The maximality of d for the partition in G implies that the period is exactly d.

To establish the converse, it is instructive to define a relation on the nodes of the strongly connected graph G associated with $\mathcal{M}$, assumed to have period d. Namely, define $i \sim j$ if every (i, j)-path has length 0 modulo d. Note that

since G is strongly connected, a path exists between every pair of nodes in G. The following result asserts that we again have an equivalence relation.

Lemma 6: $\sim$ is an equivalence relation.

While it is conceivable that some paths between nodes i and j might have length 0, while others might have positive length (modulo d), this cannot occur. In fact the following general result holds concerning path lengths, modulo d.

Lemma 7: For every pair of nodes i and j in G, all (i, j)-paths in G have the same length modulo d.

The equivalence classes associated with $\sim$ form the desired partition when labeled appropriately. Choose any node i_0 and a cycle containing i_0. Since $\mathcal{M}$ has period d, this cycle must contain at least d edges. Let the first d nodes of this cycle be denoted $i_0, i_1, \ldots, i_{d-1}$. Notice that no two of these nodes can be elements of the same equivalence class because this would identify a path between those two nodes of length less than d, whereas all such paths must have length 0 modulo d. Label the equivalence class containing node i_k as C_k.

Lemma 8: Let $\{C_0, C_1, \ldots, C_{d-1}\}$ be the equivalence classes induced by $\sim$ and numbered according to any (i_0, i_0) cycle. If (i, j) is an edge of G with $i \in C_k$, then $j \in C_{(k+1) \bmod d}$.

Corollary 1: Let Q_1 and Q_2 be (i, j_1) and (i, j_2)-paths in G, respectively. If $l(Q_1) \bmod d = l(Q_2) \bmod d$, then j_1 and j_2 are elements of the same equivalence class.

Lemma 8 shows that property (a) of Theorem 4 holds. Property (b) follows from the maximality of the period d. Moreover, by Corollary 1 and the fact that there are exactly d remainders modulo d, it is seen that $\{C_0, C_1, \ldots, C_{d-1}\}$ is indeed a partition of the nodes of G. This establishes the stated characterization for the period d of an irreducible Markov chain. Consequently, determining d can be reduced to finding an appropriate partition of the nodes in the associated digraph G. Although this might seem to be a more difficult problem, it can be accomplished efficiently in conjunction with a breadth-first search of G.

17.3.2 Breadth-First Search Algorithm

In the irreducible Markov chain $\mathcal{M}$, every transition moves from a node in C_k to a node in $C_{(k+1) \bmod d}$. This property can be used to identify the sets of nodes C_k in the associated digraph G by examining the nodes of G in order of non-decreasing path length from an arbitrary starting node. A breadth-first

search of G provides a systematic examination of such paths.

A breadth-first search of G from node v produces a collection of *level* sets. The kth level set consists of those nodes j that can be reached from node v by a shortest path of length k; in this case we define level$(j) = k$. Since all nodes within a level set share this common path length, every level set is contained in a single equivalence class (by Corollary 1). This property is independent of the node from which the breadth-first search is started. The difficulty lies in determining which level sets belong to the same equivalence class. Lemma 7 shows that certain information about d can be gleaned from the knowledge of path lengths. Sufficient information of this type can be gathered as we carry out the breadth-first search to determine d.

A breadth-first search of the (strongly connected) digraph G visits all nodes of G in order of non-decreasing level. In addition, this search partitions the edges of G into *tree* and *non-tree* edges. The breadth-first search is started from an arbitrary node v, setting level$(v) = 0$. As the search progresses, all edges incident from nodes on each level are successively examined in order of increasing level. Suppose edge (i, j) emanates from a node i with level$(i) = r$. Edge (i, j) is a tree edge if node j has not yet been encountered in the traversal of G. In this case, node j is assigned level$(j) = r + 1$. The collection of all such tree edges induces a directed tree T (the *breadth-first search tree*) rooted from the starting node v. If j has already been encountered, then (i, j) is a non-tree edge and j already belongs to some level set, with $s = \text{level}(j) \leq r+1$. If j is an ancestor of i in the search tree, the edge is called a *back edge* $(s < r)$; otherwise, the edge is called a *cross edge* $(s \leq r + 1)$. See Figure 7. In both cases, these non-tree edges provide information regarding the period of $\mathcal{M}$.

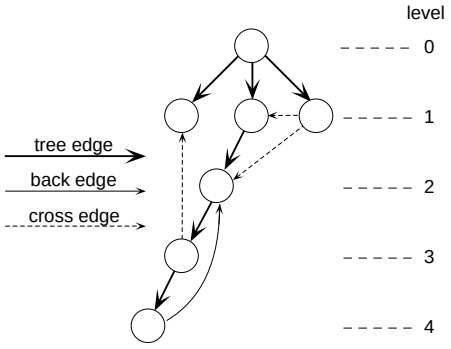

FIGURE 7 Tree, back, and cross edges in a breadth-first search

If (i, j) is a back edge, then the tree edges from j to i plus the edge (i, j) form a cycle in G. The length of the cycle, $r - s + 1$, is divisible by the period of the Markov chain. If (i, j) is a cross edge, two distinct paths from the root

to j have been identified. One of these follows tree edges directly to j and has length s. The other path follows tree edges to i and then uses edge (i, j), giving a length of $r + 1$. Since (by Lemma 7) alternative paths between any pair of nodes have the same length modulo d, the difference in path lengths, $r - s + 1$, is again divisible by the period d of the Markov chain. When $s = r + 1$, the paths have the same length and this provides no additional information about the period. Accordingly, we define the *value* of any edge $e = (i, j)$ by $\text{val}(e) = \text{val}(i, j) = \text{level}(i) - \text{level}(j) + 1 \geq 0$. Note that $\text{val}(e) = 0$ when $e \in T$. Also d must divide $\text{val}(e)$ for $e \notin T$, so d must divide $g = \gcd\{\text{val}(e) > 0 : e \notin T\}$. In fact we claim that $g = d$, which will be established after first stating the implied algorithm for determining the period of a finite irreducible Markov chain $\mathcal{M}$. This approach was first developed by Denardo [3].

Algorithm 1: Finding the period d of $\mathcal{M}$

1. From an arbitrary root node, perform a breadth-first search of G producing the rooted tree T.
2. The period g is given by $\gcd\{\text{val}(e) > 0 : e \notin T\}$.

As a practical matter, it is not necessary to keep all of the values generated by non-tree edges; rather, we maintain only the current gcd g, initialized to be the first positive edge value encountered. Whenever a new $\text{val}(e) > 0$ is found for $e \notin T$, then g is updated using $g := \gcd\{g, \text{val}(e)\}$. If at any step $\text{val}(e) = 1$ is generated (from a cross edge within the same level set), then the gcd is 1 and the Markov chain is aperiodic; the algorithm can be terminated immediately in such an instance.

To illustrate Algorithm 1, consider the digraph G with six nodes shown in Figure 8. A breadth-first search tree for G rooted at node 1 is shown in Figure 9. The first non-tree edge encountered is $(2, 1)$, a back edge. Then g is initialized to be $\text{level}(2) - \text{level}(1) + 1 = 3 - 0 + 1 = 4$. The only other non-tree edge is $(6, 4)$, a cross edge. It has value $\text{level}(6) - \text{level}(4) + 1 = 3 - 2 + 1 = 2$, giving the final $g = \gcd\{4, 2\} = 2$. Notice that G contains no cycle of length 2, but it does have cycles of lengths 4 and 6, yielding a period of 2.

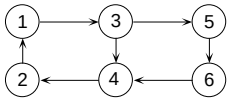

FIGURE 8 A Markov chain with period 2

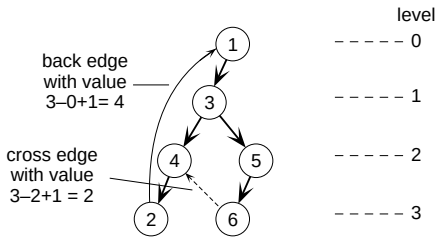

FIGURE 9 Breadth-first search for the Markov chain in Figure 8

To establish the correctness of Algorithm 1, it is sufficient to show that $g = \gcd\{\text{val}(e) > 0 : e \notin T\}$ divides the length of an arbitrary cycle W in G. If so it must divide the period d, the gcd of all cycle lengths in G. Since we have already seen that d divides g, then we must have $g = d$.

Relative to the breadth-first search tree T, let the edges E of G be partitioned into sets D and R. Set D consists of *down* edges (i, j), in which $\text{level}(j) = \text{level}(i) + 1$, and set R contains the *remaining* edges. Notice that all tree edges are in D, all back edges are in R, whereas cross edges can be in either set. Also edge e has $\text{val}(e) > 0$ precisely when $e \in R$. Now let $Q = [i_0, i_1, \ldots, i_k]$ be any path of length $l(Q) = k$ in G and define $\text{val}(Q) = \sum_{s=0}^{k-1} \text{val}(i_s, i_{s+1})$. Since $\text{val}(i_s, i_{s+1}) = \text{level}(i_s) - \text{level}(i_{s+1}) + 1$, it then follows that $\text{val}(Q) = \text{level}(i_0) - \text{level}(i_k) + l(Q)$. In particular if W is any cycle from node i to itself in G, then $\text{val}(W) = \text{level}(i) - \text{level}(i) + l(W) = l(W)$. This gives $l(W) = \text{val}(W) = \sum \{\text{val}(e) : e \in W\} = \sum \{\text{val}(e) : e \in W \cap R\}$. However, each term in the last sum is positive and divisible by g (the gcd of all positive non-tree values), so g divides $l(W)$, as required.

The computational effort associated with Algorithm 1 can be split into two parts: executing the breadth-first search and then finding the greatest common divisor associated with certain non-tree edges. Let $m = |E|$ denote the number of edges in the digraph G. Then the time complexity associated with the breadth-first search is $\text{O}(n + m) = \text{O}(m)$ [7]. There are $m - n + 1$ non-tree edges, hence a greatest common divisor must be found at most $m - n$ times (some non-tree edges are down edges and are thus not considered). Since $\text{val}(e) = \text{level}(i) - \text{level}(j) + 1 \leq \text{level}(i) + 1 \leq n$, the complexity of carrying out $m - n$ gcd's turns out to be $\text{O}(\log n + m) = \text{O}(m)$ [2]. Hence the overall complexity of the algorithm is $\text{O}(m)$. Using appropriate data structures, the algorithm requires $\text{O}(n + m) = \text{O}(m)$ space. As in the depth-first search algorithm for finding strong components, this algorithm for determining d can exploit sparsity present in the transition probability matrix P, which often has relatively few non-zero entries.

17.4 Conclusion

Our aim in this chapter is to provide a parallel development to the standard, matrix-based analysis of finite-state Markov chains [6, 8]. These graph-theoretic interpretations not only maintain a visual representation of the model but also reinforce a number of algebraic concepts. Specifically, the idea of an equivalence relation turns out to be quite useful in carrying out state classification as well as in determining periodicity. Moreover, an important step in any modeling effort is the development of efficient algorithms and data structures that "solve" the model in an effective way. This is especially important for the successful completion of large-scale modeling projects. Interestingly, certain of the Markov chain computations are most easily carried out using a depth-first search of G (state classification), while others involve a breadth-first search of G (determining the period). Overall, a mathematical sciences approach (blending discrete, algebraic, and computational elements) is illustrated here.

17.5 Exercises

1. Prove that the relation $\leftrightarrow$ is an equivalence relation (Lemma 1).

2. Prove that all states in a communicating class have the same character with regard to transience and recurrence (Lemma 2).

3. Consider the Markov chain whose non-zero transition probabilities are indicated by '+' in the matrix below:

$$P = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} + & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & + & 0 \\ 0 & 0 & + & + & + & 0 \\ + & 0 & 0 & 0 & 0 & + \\ 0 & + & 0 & 0 & + & 0 \\ + & 0 & 0 & + & 0 & 0 \end{pmatrix} \end{array}$$

(a) Draw the state transition diagram G for this Markov chain and determine the communicating classes.

(b) Classify the states of this Markov chain (recurrent or transient).

(c) Does the chain have a unique stationary distribution?

(d) Draw a depth-first search tree obtained by starting the search at node 3. Show the node labels and the non-tree edges produced by the search.

(e) What are the root nodes associated with the strong components of G?

4. The *node adjacency matrix* $A = [a_{ij}]$ for a digraph $G = (N, E)$ with $n = |N|$ is the $n \times n$ matrix with $a_{ij} = 1$ if $(i, j) \in E,\ a_{ij} = 0$ otherwise.

(a) Show that the (i, j) entry of A^k is the number of distinct (i, j)-paths in G having length k. (Such paths are allowed to contain repeated nodes and edges.)

(b) The *reachability matrix* $R = [r_{ij}]$ for G is the 0-1 matrix in which $r_{ij} = 1$ if $i \to j$, $r_{ij} = 0$ otherwise. Show that $R = B((I + A)^{n-1})$ where $B(\cdot)$ is the (entrywise) Boolean function taking on the value 1 for non-zero arguments and 0 for zero arguments.

(c) Show that node j is in the strong component of G containing node i if and only if $r_{ij} \cdot r_{ji}$ is non-zero. Compare the complexity of this approach for finding strong components with the graph-theoretic approach outlined in Section 17.2.3.

5. Let i be a state of a Markov chain such that there is positive probability of a return to state i in a finite number of steps. Let D denote the set of all positive integers r such that $p_{ii}^k = 0$ for all k not divisible by r.

(a) Show that $1 \in D$.

(b) Show that D is bounded above (and hence contains a largest positive member).

6. Prove that if i and j are communicating states, then i and j have the same period (Lemma 5). Hint: Consider paths P, Q that make i accessible from j and j accessible from i respectively. Then the union of P and Q is a circuit through both i and j. Now apply the definition of periodicity.

7. Prove that the relation $\sim$ is an equivalence relation (Lemma 6).

8. Prove Lemma 7.

9. Consider the Markov chain whose non-zero transition probabilities are indicated by '+' in the matrix below:

$$P = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array} \begin{array}{c} \begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\ \left(\begin{array}{cccccccccc} 0 & 0 & 0 & + & 0 & 0 & 0 & 0 & 0 & 0 \\ + & 0 & 0 & 0 & + & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & + & 0 & 0 & 0 \\ 0 & + & + & 0 & 0 & 0 & 0 & + & 0 & 0 \\ 0 & 0 & 0 & + & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & + & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & + & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & + & 0 & + & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & + \\ 0 & 0 & 0 & 0 & 0 & + & 0 & 0 & 0 & 0 \end{array} \right) \end{array}$$

(a) Draw the state transition diagram for this chain. Explain why the chain is irreducible.

(b) Show the breadth-first search tree obtained by starting with state (node) 1. Also indicate the level of each node and display the non-tree edges.

(c) Use the techniques described in Section 17.3.2 to determine the period of this Markov chain.

(d) Use the level sets obtained from the breadth-first search to identify the partition of states induced by periodicity (see Figure 6).

10. The *Euclidean algorithm* can be used to find the greatest common divisor of two integers. Find a reference to this algorithm and then implement the method in some programming language.

11. Suppose that an irreducible finite-state Markov chain with period d has the associated cyclic partition of states $\{C_0, C_1, \ldots, C_{d-1}\}$. Suppose that the states are numbered so that states from C_i receive smaller numbers than those from $C_{i+1}, 0 \leq i < d-1$.

(a) Show that the state transition probability matrix in block form is

$$\begin{pmatrix} 0 & P_0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & P_1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & P_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & P_{d-2} \\ P_{d-1} & 0 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

(b) Show that P^d is block diagonal and hence the Markov chain with *one-step* transition probability matrix given by P^d has d irreducible, aperiodic classes of states.

(c) Show that the ith diagonal block of P^d is $(P_i P_{i+1} \ldots P_{d-1} P_0 \ldots P_{i-1})$ for $0 \leq i \leq d-1$.

(d) Let π^i denote the limiting probabilities associated with the ith class of states in the Markov chain with one-step transition matrix P^d. This distribution is unique because the classes are irreducible and aperiodic (Theorem 3). Show that $\pi^{i+1} = \pi^i P_i$ and that the unique stationary distribution of the original Markov chain is given by $\pi = (\pi^0, \pi^1, \ldots, \pi^{d-1})/d$.

(e) By part (d), the stationary distribution for a periodic Markov chain with n states can be obtained by solving a system of equations with fewer than n equations. Which set of equations should be chosen?

17.6 References

Feller [4] discusses many classic probability models, including finite Markov chains. A more recent treatment of Markov chains and other probability models is in similar volumes by Ross [6] and Taylor and Karlin [8]. Traditional matrix-based computational methods for Markov chains are presented by Isaacson and Madsen [5]. Tarjan [7] develops a number of the most efficient graph algorithms and their associated data structures. An extensive treatment of topics in computational methods for discrete structures including graphs is given by Aho, Hopcroft, and Ullman [1]. The particular techniques for determining periodicity discussed in Section 17.3.2 were first presented by Denardo [3].

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[2] G. H. Bradley, "Algorithm and bound for the greatest common divisor of n integers", *Commun. ACM* **13**, 433–436 (1970).

[3] E. V. Denardo, "Periods of connected networks and powers of nonnegative matrices", *Math. Oper. Res.* **2**, 20–24 (1977).

[4] W. Feller, *An Introduction to Probability Theory and Its Applications*, Vol. I, 3rd ed., Wiley, New York, 1968.

[5] D. L. Isaacson and R. W. Madsen, *Markov Chains*, Wiley, New York, 1976.

[6] S. M. Ross, *Introduction to Probability Models*, 5th ed., Academic Press, Boston, 1993.

[7] R. E. Tarjan, *Data Structures and Network Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, 1983.

[8] H. M. Taylor and S. Karlin, *An Introduction to Stochastic Modeling*, Academic Press, San Diego, 1994.