The Spanning Tree Enumeration Problem for Digraphs

Nathaniel Dean
Bellcore
Morristown, NJ 07960
email: nate@bellcore.com

A. K. Kelmans
Rutgers University
New Brunswick, NJ 08903
email: kelmans@dimacs.rutgers.edu

Keh-Wei Lih
Bellcore
Red Bank, NJ 07701
email: lih@cc.bellcore.com

William A. Massey
AT&T Bell Labs
Murray Hill, NJ 07974
email: will@research.att.com

Peter Winkler
Bellcore
Morristown, NJ 07960
email: pw@bellcore.com

Abstract

Counting spanning trees is a fundamental problem in enumerative combinatorics. The conventional approach uses determinants, and explicit formulas have been found for several families of graphs, undirected and directed. We focus on the families of directed graphs that evolve naturally from the study of certain longstanding unsolved problems in queueing theory, i.e., networks of queues with finite buffers. We derive a new method for counting the number of spanning trees rooted at any given node, and we show how this in turn yields the steady state queue length probabilities. To demonstrate this method, we derive explicit formulas for the number of trees and, hence, the queue length probabilities for the case of two queues in tandem with smallest buffer size at most two.

1 Introduction

Counting spanning trees is a fundamental problem in enumerative combinatorics dating at least back to 1889 with Cayley's formula [2] for the number of spanning trees in a complete graph. Of course, now several proofs exist for this result, and connections with other combinatorial objects such as Prüfer sequences are well-known. This paper concentrates on certain families of directed graphs that evolve naturally from the study of some longstanding unsolved problems in queueing theory, i.e., networks of queues with finite buffers. We use the fact that the steady state queue length probabilities are related by a constant factor to the number of spanning trees rooted at the nodes of the associated digraph. We derive a new method for counting these trees and demonstrate this method by deriving an explicit formula for the number of trees (hence, the queue length probabilities) for the case of two queues in tandem with smallest buffer size at most two. Our method so far seems to be limited by a dependence on computing eigenvalues and eigenvectors of certain matrices.

A **directed graph** D (or **digraph**) consists of a nonempty set $V(D)$ of vertices and a set $A(D)$ of ordered pairs of vertices, called **arcs**. A digraph is said to be **strict** if it contains no loops, i.e. arcs of the form (v, v). For our purposes a **rooted tree** is a digraph in which every vertex has outdegree exactly 1, except one vertex called the **root** which has outdegree 0.

Let $X \subseteq V(D)$, and let C_X be the set of arcs joining X and $V(D)-X$. Then C_X is called an X-**cut**. For any vertex v of D we define the neighbor sets $N_X^+(v) = \{w \in V(D) : vw \in C_X\}$ and $N_X^-(v) = \{w \in V(D) : wv \in C_X\}$ and the corresponding outdegree $d_X^+(v) = |N_X^+(v)|$ and indegree $d_X^-(v) = |N_X^-(v)|$. We drop the subscript only when $X = \{v\}$.

For any function f the sum $\sum_{y \in W} f(y)$ is defined to be 0 when $W = \emptyset$.

2 Queueing Networks

We consider queueing networks that arise in the following way. A *customer* arrives at the first *server* and demands service. If the server is busy with another customer, the newly arrived customer must wait

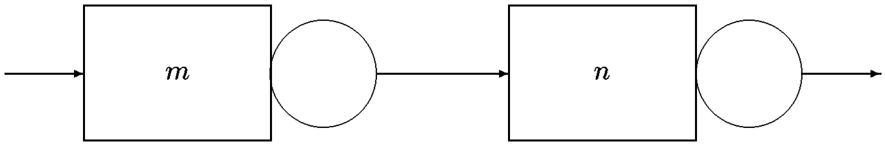

Figure 1: The 2-server queue with finite buffers

until the server is free. The customers that arrive when the server is busy form the first *queue* which we assume has a finite capacity (i.e., buffer) of size m. Customers leaving this queue must then be serviced by a second server having a queue of capacity n. Customers exit the system when they leave the second queue. Thus, we have two queues in tandem (Figure 1).

The *state* of the system is completely specified by the numbers i, j of customers in the first and second queues, respectively. There are exactly $(m+1)(n+1)$ possible states, and these states together with the possible transitions to other states can be modeled in an obvious way as the vertices and arcs of a digraph D as shown in Figure 2. Actually, D is a weighted digraph, since associated with each pair of vertices x, y there is the probability p_{xy} of a transition from state x to state y. Of course, this probability is zero if xy is not an arc of D, and so we think of D as an arc-weighted digraph. The **weight** of any subdigraph H of D is defined to be $\omega(H) = \Pi_{e \in A(D)} p_e$. Let $T_x(D)$ denote the set of spanning trees rooted at x, and let $\tau(x) = |T_x(D)|$.

The behavior of a queueing system (i.e., service times, throughput, utilization, etc.) is determined by the *steady state queue length probabilities*, that is, the probabilities of being in each of the many possible states. Thus, the ideal objective of queueing theorists is to obtain a closed form expression for these probabilities—usually an impossible task. The behavior of the queueing system is also affected by the rate at which customers enter and leave each queue. In this paper we assume that these rates are all the same. In the more general setting of

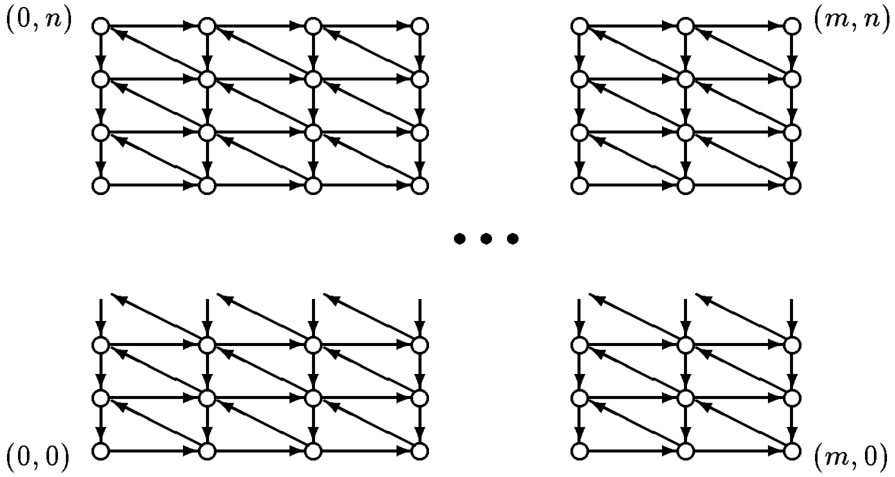

Figure 2: The m, n-queueing network

discrete Markov chains the steady state probabilities determine the stationary distribution of the chain. Instead of probabilities, our analysis will concentrate on the derivation of formulas for the number of trees rooted at specified nodes of the associated digraph. This approach is justified by the following somewhat old but neglected probabalistic result [5].

Theorem 1 *The steady state probabilities for any discrete time Markov chain are related by a constant factor to the number of spanning trees rooted at the nodes of the associated digraph.*

In the remainder of this section we present some general results about digraphs, and so the arc-weights are arbitrary real numbers, not just probabilities. In the next section we show how these results relate directly to queueing theory. We note that our approach does not exploit the well-known Matrix Tree Theorem (e.g., see [1]), and in fact we found it too cumbersome to use for this application.

Theorem 2 *Let X be a set of vertices of a digraph D. Then*

$$\sum_{i \in X} \sum_{ij \in C_X^+} \omega_i(D) p_{ij} = \sum_{j \in V(D)-X} \sum_{ji \in C_X^-} \omega_j(D) p_{ji}$$

where $\omega_i(D) = \sum_{T\in T_i(D)} \omega(T)$.

Proof. Let $P(W)$ be the set of pairs consisting of a spanning tree rooted at some vertex w of W, and an arc of C_X originating at w. It suffices to give a bijection from $P(X)$ to $P(V(D)-X)$. Accordingly, let (T,e) be in $P(X)$, where the root of T (and origin of e) is x. Add e to T, creating a cycle C. Moving forward from arc to arc along C, beginning at e, let e' be the *last* arc of C_X encountered. Then e' must originate at some vertex y in $V(D)-X$; delete e' from the union of T and e to get a spanning tree T' rooted at y. Let H denote this transformation from $P(X)$ to $P(V(D)-X)$, i.e., $H(T,e) = (T',e')$. The transformation H' is defined similarly for a pair in $P(V(D)-X)$, except that this time the deleted arc is the *first* cut-arc after e' encountered in the cycle instead of the last. But then H' is the inverse of H; for, if we begin with the pair (T',e') above then adding e' to T' produces the same cycle C we had before. Since the arc e will now be the first cut-arc encountered in C beginning from e', the original pair (T,e) will be recovered. It follows that H is a bijection. Since $T\cup e = T'\cup e'$, we have

$$\sum_{(T,e)\in P(X)} \omega(T\cup e) = \sum_{(T',e')\in P(X)} \omega(T'\cup e').$$

Evaluating the left hand side yields

$$\begin{aligned}\sum_{(T,e)\in P(X)} \omega(T\cup e) &= \sum_{(T,e)\in P(X)} \omega(T)\omega(e) = \sum_{i\in X}\sum_{ij\in C_X^+}\sum_{T\in T_i(D)} \omega(T)\omega(ij)\\ &= \sum_{i\in X}\sum_{ij\in C_X^+} p_{ij} \sum_{T\in T_i(D)} \omega(T) = \sum_{i\in X}\sum_{ij\in C_X^+} \omega_i(D)p_{ij}.\end{aligned}$$

Similarly,

$$\sum_{(T',e')\in P(X)} \omega(T'\cup e') = \sum_{j\in V(D)-X}\sum_{ji\in C_X^-} \omega_j(D)p_{ji}$$

which completes the proof. ∎

Corollary 3 (Balancing Cut Lemma) *Let X be a set of vertices of a digraph D. Then* $\sum_{x\in X} d_X^+(x)\tau(x) = \sum_{y\in V(D)-X} d_{V(D)-X}^+(y)\tau(y)$.

Proof. In Theorem 2 let $p_{ij} = 1$ for all arcs $ij \in A(D)$. Then for each vertex $i \in X$, we see that

$$\begin{aligned}\sum_{ij \in C_X^+} \omega_i(D) p_{ij} &= \sum_{ij \in C_X^+} \omega_i(D) = \sum_{ij \in C_X^+} \sum_{T \in T_i(D)} \omega(T) \\ &= \sum_{ij \in C_X^+} \tau(i) = d_X^+(i)\tau(i).\end{aligned}$$

Similarly,

$$\sum_{ji \in C_X^-} \omega_j(D) p_{ji} = d^+_{V(D)-X}(j)\tau(j). \blacksquare$$

Theorem 4 (Averaging Theorem) *For every vertex x of a strict digraph, $d^+(x)\tau(x) = \sum_{y \in N^-(x)} \tau(y)$.*

Proof. Let $X = \{x\}$. ∎

An **eulerian digraph** is a connected digraph with $d^+(x) = d^-(x)$ for every vertex x. A digraph G is called **regular** if each component is eulerian and every vertex of G has the same outdegree. The Averaging Theorem provides an easy proof of the following result for eulerian digraphs which was first proved by Tutte [7]. The same result restricted to connected regular digraphs was proved independently by Fellows and Wojciechowski [4].

Corollary 5 *In an eulerian digraph, $\tau(x)$ is the same for every vertex x.*

Proof. Otherwise, the set M of vertices for which τ is maximum has an inneighbor $y \in V(G) - M$. Let $x \in M \cap N^+(y)$. Since $\tau(x) > \tau(y)$, there must be a vertex $z \in N^-(x)$ with $\tau(z) > \tau(x)$, a contradiction. ∎

3 Two Finite Queues in Tandem

Recall that we have a 2-server queueing network (Figure 2) with finite buffers of size m and n, and we want to find a formula for the number $\tau(i,j)$ of spanning trees rooted at any node (i,j). We begin by deriving a formula for the elements of column $k-1$ in terms of those in columns

k and $k+1$. From the Averaging Theorem we get the following system of linear recurrences where $1 \le k \le m-2$.

$$
\begin{aligned}
\tau(k-1,n) &= 2\tau(k,n) - \tau(k+1,n-1) \\
\tau(k-1,j) &= 3\tau(k,j) - \tau(k,j+1) - \tau(k+1,j-1),\ 1 \le j \le n-1 \\
\tau(k-1,0) &= 2\tau(k,0) - \tau(k,1)
\end{aligned}
$$

In matrix form we have $\vec{u}_{k-1} = M\vec{u}_k$ where $\vec{u}_k = [\tau(k,n), \tau(k,n-1), \ldots, \tau(k,0), \tau(k+1,n-1), \ldots, \tau(k+1,0),]^T$ and M is the $2n+1 \times 2n+1$ matrix shown in Figure 3. Hence, $\vec{u}_k = M^{m-k}\vec{u}_m$. Now we must

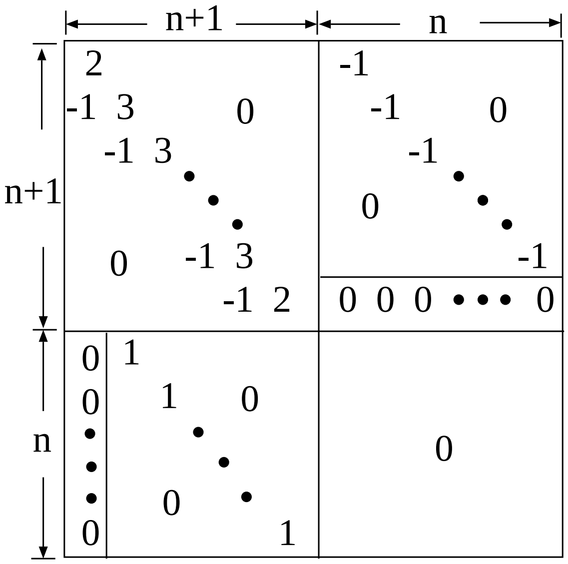

Figure 3: The matrix M

solve this equation treating $\vec{u}_m$ as the initial condition. The Averaging Theorem can be used to express the terms $\tau(m-1,n), \tau(m-1,n-1), \ldots, \tau(m-1,0)$ as a function of the terms $\tau(m,n-1), \ldots, \tau(m,0)$; in particular,

$$
\begin{aligned}
\tau(m-1,n) &= \tau(m,n) && (1) \\
\tau(m-1,j) &= 2\tau(m,j) - \tau(m,j+1),\ 1 \le j \le n-1 && (2) \\
\tau(m-1,0) &= \tau(m,0) - \tau(m,1) && (3)
\end{aligned}
$$

So it remains to be seen how to derive an expressions for $\tau(m, n-1), \ldots, \tau(m, 0)$. The derivation involves a series of reductions which will be explained in the next section. Our method for solving recurrences involves computing the eigenvalues and eigenvectors of M.

4 The Case n=2

The purpose of this section is two-fold. We use it to illustrate the technique of the previous section explicitly by way of example, and we use it to explain our derivation of an expression for $\vec{u}_m$. The derivation requires a number of reductions similar to those used in Kelmans [6].

We recursively reduce the size of the queueing network by expressing the number of spanning trees rooted at the nodes of the m'th column of the original network in terms of the number of spanning trees with the same root but in a smaller network. The networks are constructed by selecting certain edges to be in or not in the tree. We must consider both possibilities as part of a recurrence relation. For example, Figure 4 shows how we can eliminate one column of the network. We use T_k^0 to represent the portion of the m, n-network composed of columns k through m. Clearly, any spanning tree rooted in the mth column must contain the bold edge in the top diagram T_k^0, since the outdegree of every vertex except the root is exactly one. If arc a is contained in the spanning tree, then no other arc with the same origin as a can be in the tree. Further, the tree can contain no cycles. This allows us to eliminate two arcs from T_k^0 as shown. Continuing this way we can reduce the number of columns in the digraph T_k^0, but we create some new digraphs with restrictions that certain specific arcs must be contained in the spanning trees. The number of spanning trees in these reduced digraphs are denoted by variables of the form T_{k+1}^i where we abuse the notation by letting T_{k+1}^i also serve as a name for that portion of the queueing network. We apply the same process to all T_*^i, $i = 0, 1, 2, 3, 4$, and end up with the five different types of networks shown in Figure 5. However, $T_{k+1}^3 = T_{k+1}^0 - T_{k+1}^1$.

$$\begin{array}{lcllll}
T_k^0 & = & 6T_{k+1}^0 & -T_{k+1}^1 & +3T_{k+1}^2 & \\
T_k^1 & = & 2T_{k+1}^0 & +T_{k+1}^1 & +T_{k+1}^2 & \\
T_k^2 & = & T_{k+1}^0 & & +2T_{k+1}^2 & +T_{k+1}^4 \\
T_k^4 & = & 5T_{k+1}^0 & -2T_{k+1}^1 & +4T_{k+1}^2 & +T_{k+1}^4
\end{array}$$

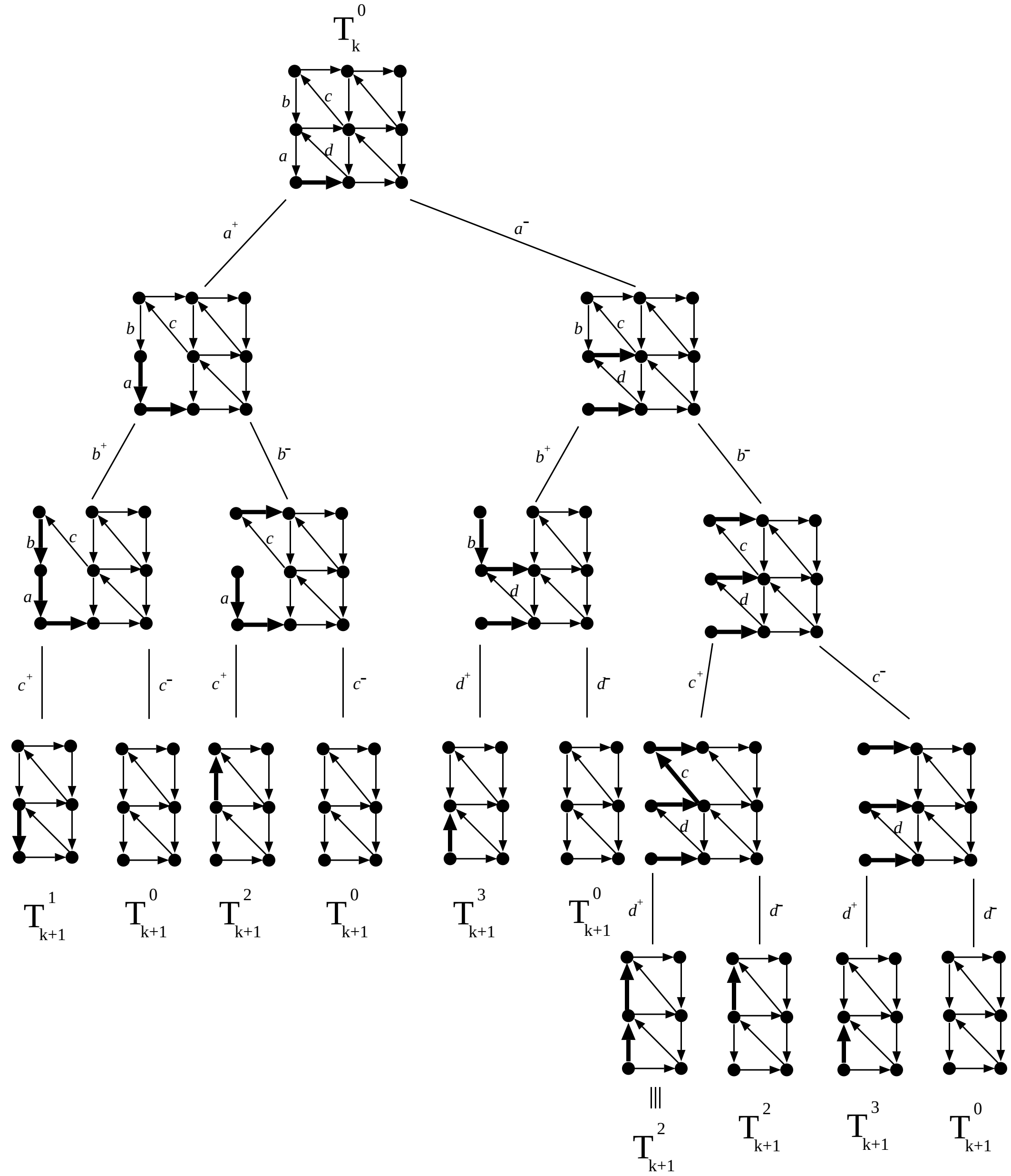

Figure 4: The Tree Generation for T_k^0

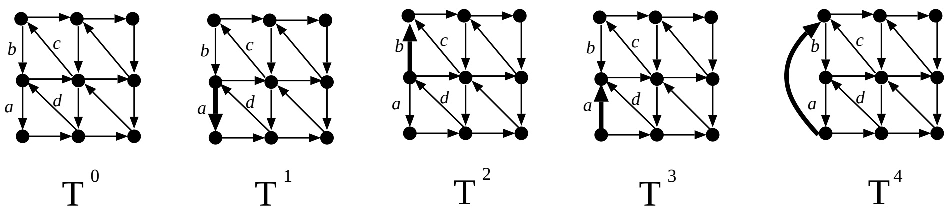

Figure 5: The Tree Generation Patterns

Again by using these recurrence relations and Equations 1–3 we find a formula for $\vec{u}_m$ which essentially represents the number of spanning trees rooted in the last two columns of the network.

$$\vec{u}_m = 2^{m-3.5}\begin{bmatrix} (2+\sqrt{2})^{m+1} - (2-\sqrt{2})^{m+1} - 4\sqrt{2} \\ 3\sqrt{2}((2+\sqrt{2})^m + (2-\sqrt{2})^m) \\ 3((2+\sqrt{2})^m - (2-\sqrt{2})^m + 2\sqrt{2}) \\ (2\sqrt{2}+1)(2+\sqrt{2})^m + (2\sqrt{2}-1)(2-\sqrt{2})^m - 2\sqrt{2} \\ 2(2+\sqrt{2})^{m+1} - 2(2-\sqrt{2})^{m+1} + 4\sqrt{2} \end{bmatrix}.$$

Recall that the matrix M is used to find the number of spanning trees rooted at other vertices of the network. For the case $n = 2$, M has the following form.

$$M = \begin{bmatrix} 2 & 0 & 0 & -1 & 0 \\ -1 & 3 & 0 & 0 & -1 \\ 0 & -1 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Now we are ready to present the general solution for an arbitrary column k.

$$\vec{u}_k = 3 \cdot 2^{2m-k-4.5} \begin{bmatrix} (2+\sqrt{2})^{k+1} - (2-\sqrt{2})^{k+1} - 2\sqrt{2} \\ 2\sqrt{2}((2+\sqrt{2})^k + (2-\sqrt{2})^k) \\ 2((2+\sqrt{2})^k - (2-\sqrt{2})^k + 2\sqrt{2}) \\ \sqrt{2}((2+\sqrt{2})^{k+1} + (2-\sqrt{2})^{k+1}) \\ (2+\sqrt{2})^{k+1} - (2-\sqrt{2})^{k+1} + 2\sqrt{2} \end{bmatrix}.$$

References

[1] C. Berge. *Graphs*, North-Holland, 2nd ed., 1985.

[2] A. Cayley, A theorem on trees. *Quart. J. Math.* (1889) **23** 376–378.

[3] R. L. Disney, W. Giffin and B. Wong, Two finite M/M/1 queues in tandem: a matrix solution for the steady state. *OPSEARCH* **14** (1977) 1-18.

[4] M. Fellows and J. M. Wojciechowski, Counting spanning trees in regular connected multigraphs. *J. Franklin Inst.* **326** (1989) 889–896.

[5] M. I. Freidlin and A. D. Ventsel', On small random perturbations of dynamical systems. *Russian Mathematical Surveys* **25** (1970) 1–56.

[6] A. K. Kelmans, Probabilistic networks and randomly deleted elements. In *Dokl. Vses. Sovesc. po Statist. Metodam Teorii Upravlenija*, Moscow (1970) 18–27 [in Russian].

[7] W. Tutte, The dissection of equilateral triangles into equilateral triangles. *Proc. Cambridge Philos. Soc.* **44** (1948) 463–482.