Algorithms to Calculate the Distribution of the Longest Path Length of a Stochastic Activity Network with Continuous Activity Durations

Lawrence M. Leemis, Matthew J. Duggan, John H. Drew, Jeffrey A. Mallozzi, Kerry W. Connell
Department of Mathematics, The College of William & Mary, Williamsburg, VA 23187

We develop algorithms to calculate the probability distribution of the longest path of an arbitrary stochastic activity network with continuous activity durations by three techniques: recursive Monte Carlo simulation, series-parallel reduction, and conditioning. Examples illustrate the use of the three techniques. © 2006 Wiley Periodicals, Inc. NETWORKS, Vol. 48(3), 143–165 2006

Keywords: computer algebra system; conditional probability; recursion; series-parallel reduction

1. INTRODUCTION

Activity networks are used to plan projects by showing precedence relationships between the various activities that constitute such projects [1]. An activity network is a special case of a directed graph in which the nodes (or vertices) represent points in time and the arcs (or edges) represent activities with time values modeling activity durations. Figure 1 shows an example of a stochastic activity network, where the positive random variable Y_{ij} denotes the time to complete the activity associated with the arc from node i to node j. The Y_{ij}s are assumed to be mutually independent. Common-cause delays, for example, rain delays on an outdoor project, are not considered here. Activity start times are constrained in that no activity emanating from a given node can start until all activities that enter that node have been completed. We consider three techniques for calculating the distribution of the time to complete an activity network: simulation, series-parallel reduction, and conditioning. In some cases we are also able to calculate the probability that a path through the network has the greatest length, and the probability that an arc is on the path of greatest length.

The notation used to describe a stochastic activity network is introduced in Section 2. A recursive Monte Carlo simulation algorithm for estimating measures of performance associated with a stochastic activity network is described in Section 3. Two algorithms for finding exact values of these performance measures are developed and illustrated in Section 4; one for series-parallel networks, and one for more general networks. The algorithm for more general networks requires two additional restrictions on the network: each node has at most two incoming arcs, and each arc duration has a probability density function that is described by a single function on $(0, \infty)$. The final section outlines conclusions and further work.

2. NOTATION AND ASSUMPTIONS

Activity networks have n nodes and m arcs, with a single source node (labeled node 1), a single terminal node (labeled node n), and no loops. Nodes are labeled $1, 2, \ldots, n$ in topological order [7] so that if there is an arc a_{ij} from node i to node j, then i is less than j. Thus, the nodes along each path occur in ascending order.

For each node j, the "backward" set $\mathcal{B}(j)$ is the set of all nodes that are immediately before node j on some path, that is, $\mathcal{B}(j)$ is the set of all nodes i for which there is an arc from i to j. Similarly, for each node j, the "forward" set $\mathcal{A}(j)$ is the set of all nodes that are immediately after node j on some path, that is, $\mathcal{A}(j)$ is the set of all nodes k for which there is an arc from j to k.

For each arc a_{ij}, there is a continuous random activity duration Y_{ij} with positive support. The distribution of Y_{ij} is determined by its cumulative distribution function (CDF) $F_{Y_{ij}}(t)$ or its probability density function (PDF) $f_{Y_{ij}}(t)$. For each node j there is a random time value T_j, which is the time of completion of all activities entering node j. The quantity T_n is therefore the time of completion of the entire network. Shier ([12], pp. 122–123) gives an example of an activity network with discrete activity durations.

Received November 2003; accepted June 2006
Correspondence to: L.M. Leemis; e-mail: leemis@math.wm.edu
Contract grant sponsor: National Science Foundation (Educational Innovation grant); Contract grant number: CDA9712718 (to J.A.M and K.W.C)
Contract grant sponsor: CSEMS (Effective Transitions through Academe to Industry for Computers Scientists and Mathematics); Contract grant number: 0123022 (to M.J.D.)
DOI 10.1002/net.20125
Published online in Wiley InterScience (www.interscience.wiley.com).

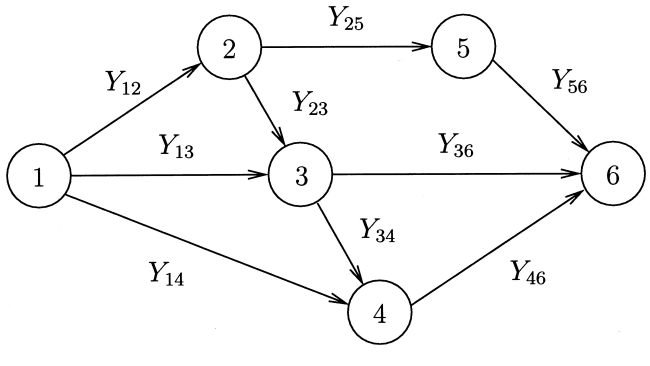

FIG. 1. A six-node, nine-arc stochastic activity network.

For each network, there will typically be several paths leading from node 1 to node n. Let M be the set of all paths, with the paths labeled $\pi_1, \pi_2, \ldots, \pi_r$, where $r = |M|$ is the number of paths. A path may be viewed as an ordered set of arcs leading in succession from node 1 to node n. The *length* of path π_k, denoted L_k, is the sum of all Y_{ij} corresponding to the arcs $a_{ij} \in \pi_k$.

For each realization of a stochastic network, the *critical path* π_c is the path with the greatest length, $L_c = \max\{L_1, L_2, \ldots, L_r\}$. The length of the critical path determines the time to complete the entire network. For a stochastic network, a path in M is the critical path with some probability $p(\pi_k) = \Pr(L_k = L_c)$, $k = 1, 2, \ldots, r$. Some arcs may be along more than one path. The probability that arc a_{ij} is along the critical path, also called the arc's *criticality*, denoted by ρ_{ij}, is the sum of all $p(\pi_k)$ where $a_{ij} \in \pi_k$. The criticality of arc a_{ij} can be calculated by

$$\rho_{ij} = \sum_{k=1}^{r} p(\pi_k)\delta_{ij}^k,$$

where δ_{ij}^k is 1 if $a_{ij} \in \pi_k$ and 0 otherwise.

2.1. Matrix Representation of the Network

The first step in building a model is defining a mathematical representation of a network. Matrices are well suited for this task because (a) each node and arc in the network can be designated by using the rows and columns of a matrix, and (b) matrices are easily instantiated in most computer languages. A node-arc incidence matrix was chosen due to its suitability for the algorithms developed here.

The node-arc incidence matrix associated with a network is an $n \times m$ matrix N, where each row represents a node and each column represents an arc. Let

$$N[i,k] = \begin{cases} 1 & \text{if arc } k \text{ leaves node } i \\ -1 & \text{if arc } k \text{ enters node } i \\ 0 & \text{otherwise.} \end{cases}$$

T1 Using the arc indexing given in Table 1 (which follows), the node-arc incidence matrix that describes the network in Figure 1 is

$$N = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & -1 \end{bmatrix}.$$

The only limitation of this representation is that it cannot show an arc that leaves and enters the same node. This ability will not be necessary, because feedback is not allowed by our definition of a stochastic activity network.

3. SIMULATION

We first consider the development of a simulation algorithm for estimating the distribution of the time to complete the network, the probability that a path is the critical path, and the criticality of an arc. Point and interval estimators for these measures of performance are discussed prior to presenting the algorithm. The simulation algorithm is presented to highlight the use of recursion and to check the subsequent analytic algorithms.

3.1. Point Estimators

If T_1 is assumed to be 0 (without loss of generality), then

$$T_j = \max_{i \in \mathcal{B}(j)} \{T_i + Y_{ij}\},$$

for $j = 2, 3, \ldots, n$, and T_n is the time to complete the entire network. The point estimator for $E[T_j]$ is the sample mean of the T_js. A T_j value is generated via the expression above in the Monte Carlo simulation algorithm to follow. The point estimator for the probability that path π_k is the critical path $p(\pi_k)$ is the fraction of the networks generated that have π_k as the critical path, $k = 1, 2, \ldots, r$. The criticality ρ_{ij} of some arc a_{ij} is estimated by

$$\hat{\rho}_{ij} = \sum_{k=1}^{r} \hat{p}(\pi_k)\delta_{ij}^k,$$

where δ_{ij}^k is 1 if $a_{ij} \in \pi_k$ and 0 otherwise.

TABLE 1. Arc-duration distributions for the network shown in Figure 1.

Arc index	Arc	Distribution of Y_{ij}
1	a_{12}	`Triangular`(1, 3, 5)
2	a_{13}	`Triangular`(3, 6, 9)
3	a_{14}	`Triangular`(10, 13, 19)
4	a_{25}	`Triangular`(3, 9, 12)
5	a_{23}	`Triangular`(1, 3, 8)
6	a_{36}	`Triangular`(8, 9, 16)
7	a_{34}	`Triangular`(4, 7, 13)
8	a_{56}	`Triangular`(3, 6, 9)
9	a_{46}	`Triangular`(1, 3, 8)

3.2. *Interval Estimators*

An approximate $(1-\alpha)\cdot 100\%$ confidence interval for $E[T_j]$ is

$$\bar{t}_j - t_{n^*-1,\alpha/2}\frac{s_j}{\sqrt{n^*}} < E[T_j] < \bar{t}_j + t_{n^*-1,\alpha/2}\frac{s_j}{\sqrt{n^*}},$$

for any node $j = 2, 3, \ldots, n$, where n^* is the number of replications of the simulation, $\bar{t}_j$ is the sample mean of the simulated T_js, s_j is the sample standard deviation of the simulated T_js, and $t_{n^*-1,\alpha/2}$ is the $1-\alpha/2$ fractile of a t distribution with n^*-1 degrees of freedom. It will often be the case that the distribution of T_j will be closer to Gaussian (via the Central Limit Theorem) as one moves from the source node to the terminal node in a stochastic activity network, resulting in improved actual coverage for this confidence interval. The approximately bell-shaped distribution of T_j associated with a node j that is close to the terminal node of the network will be more pronounced for (a) large n, (b) networks that have more series-type arrangements than parallel-type arrangements, so that sums dominate maximums (see Section 4.1), and (c) Y_{ij} with bell-shaped PDFs. In the extreme case of an entirely series network with independent and identically distributed (IID) stochastic activity durations, for example, the Central Limit Theorem can be applied directly to assure approximate normality because T_n is a sum of IID random variables. In the extreme case of an entirely parallel network with IID stochastic activity durations, a skewed distribution for the network completion time occurs because T_n is a maximum of IID random variables.

To determine an approximate $(1-\alpha)\cdot 100\%$ confidence interval for the probability estimates of $p(\pi_k)$ and ρ_{ij}, denoted generically below by p, we use [8]

$$\frac{1}{1+\frac{n^*-y+1}{yF_{2y,2(n^*-y+1),1-\alpha/2}}} < p < \frac{1}{1+\frac{n^*-y}{(y+1)F_{2(y+1),2(n^*-y),\alpha/2}}},$$

where y is the number of occurrences of some event out of n^* independent replications and $F_{a,b,c}$ is the $1-c$ fractile of an F distribution with a and b degrees of freedom.

3.3. *Algorithm*

The recursive algorithm below generates a single time to completion T_j for some node j given that the network is represented by the node-arc incidence matrix N and the stochastic activity durations Y_{ij} associated with arcs a_{ij} are generated *prior* to the call to procedure T. In most cases, this algorithm is called with argument n so that a realization of the time to complete the entire network T_n is generated. Loops and conditions are indicated by indentation. An implementation of this algorithm in C is available from the first author.

Parameters: One realization of m activity durations Y_{ij}, $n \times m$ node-arc incidence matrix N, n-array t_{long} (with entries initialized to -1), which will contain the T_j values.

Procedure name: T

Argument: node j

```
int i                                                  index for the rows of N
int k ← 1                                              index for the columns of N
int l ← 0                                              index for the predecessors to node j
float t                                                completion time of arc a_ij
float t_max ← 0.0                                      longest time of all possible paths to node j
while (l < |B(j)|)                                     loop through predecessor nodes to node j
    if (N[j,k] = −1)                                   if column k of N corresponds to an arc entering node j
        i ← 1                                          begin search for predecessor node
        while (N[i,k] ≠ 1)                             while i does not correspond to the predecessor index
            i ← i + 1                                  increment i
        if (t_long[i] ≠ −1)                            if t_long to predecessor node is already stored
            t ← t_long[i] + Y_ij                       add current arc duration to longest duration of predecessor
        else
            t ← T_i + Y_ij                             recursive call: t is the current completion time to node j
        if (t > t_max)                                 if t exceeds previous maximum completion time to node j
            t_max ← t                                  set t_max to the longest completion time
        l ← l + 1                                      increment predecessor index
    k ← k + 1                                          increment column index
t_long[j] ← t_max                                      store longest time to node j
return (t_long[j])                                     return completion time T_j
```

Because the use of straight recursion (similar to Hagstrom [6]) can cause redundant calls to nodes whose longest path has already been determined, our algorithm stores the longest time to each node once it has been calculated. After a predecessor node has been identified, if the longest time to that node has already been calculated and stored, it adds the

TABLE 2. Paths π_k for the network shown in Figure 1.

k	Node sequence	π_k
1	$1 \to 3 \to 6$	$\{a_{13}, a_{36}\}$
2	$1 \to 2 \to 3 \to 6$	$\{a_{12}, a_{23}, a_{36}\}$
3	$1 \to 2 \to 5 \to 6$	$\{a_{12}, a_{25}, a_{56}\}$
4	$1 \to 4 \to 6$	$\{a_{14}, a_{46}\}$
5	$1 \to 3 \to 4 \to 6$	$\{a_{13}, a_{34}, a_{46}\}$
6	$1 \to 2 \to 3 \to 4 \to 6$	$\{a_{12}, a_{23}, a_{34}, a_{46}\}$

current duration to the predecessor's time and calculates the maximum time of all paths entering the current node. The modified algorithm given above using dynamic programming performed roughly 11% faster than straight recursion for the simple network example that follows. For larger networks, the speed increase should be considerably greater.

3.4. Example

The simulation approach was tested on an activity network described by Pritsker ([11], pp. 216–221), shown in Figure 1. The distribution of the duration of each arc a_{ij} is given in Table 1. The three parameters of the triangular distribution are the minimum, mode, and maximum. The triangular distribution is often used for stochastic activity networks because the three parameters model the optimistic, most likely, and pessimistic times to complete an activity. An algorithm for parameter estimation using maximum likelihood when data is available can be found in van Dorp and Kotz [13]. The
T2 $r = 6$ paths through the network are given in Table 2.

The simulation was run for one million replications of the network using the multiplicative linear congruential generator $x_{i+1} = 48271x_i \bmod (2^{31} - 1)$ described in [10] with initial seed 8641. When the algorithm is called to generate one million T_6s, the order of the recursive calls associated with the node-arc incidence matrix given above, following the initial call to T_6, is T_3, T_1, T_2, T_5, T_4. For each replication, the time to completion T_j was calculated for each node in the network according to the algorithm in Section 3.3. Some sam-
T3 ple statistics for T_j are given in Table 3, where the columns show the the sample means of the times to completion, the sample standard deviations of the times to completion, and
T4 the 95% confidence interval half-widths. Table 4 shows point estimates and 95% confidence interval half-widths for $p(\pi_k)$. The point estimates total 1.001, rather than 1, due to roundoff.

TABLE 3. Sample statistics for T_j based on one million replications of the network shown in Figure 1 with arc durations from Table 1, where $\hat{h}$ is the estimated 95% confidence interval half-width.

j	$\widehat{E}[T_j]$	$\sqrt{\widehat{V}[T_j]}$	$\hat{h}$
1	0.000	—	—
2	3.001	0.817	0.002
3	7.419	1.388	0.003
4	16.037	1.971	0.004
5	10.997	2.041	0.004
6	20.754	2.087	0.004

TABLE 4. Estimated critical path probability $\hat{p}(\pi_k)$ for path π_k and estimated 95% confidence interval half-width $\hat{h}$ for one million replications for the network shown in Figure 1 with arc durations from Table 1.

k	$\hat{p}(\pi_k)$	$\hat{h}$
1	0.074	0.0005
2	0.170	0.0007
3	0.129	0.0007
4	0.198	0.0008
5	0.130	0.0007
6	0.300	0.0009

Table 5 shows point estimates and 95% confidence interval **T5**
half-widths for ρ_{ij}. Table 5 also shows the paths that contain each arc a_{ij}, using the path indices in Table 2.

Figure 2 shows the empirical CDF of the time to complete **F2**
the network for the one million replications. The random variable T_6 has support on $11 < t_6 < 34$, where the lower limit corresponds to minimum activity durations on the paths π_1 and π_4 and the upper limit corresponds to maximum activity durations on path π_6. Due to memory limitations, the empirical CDF was drawn by counting the number of network completion times that fell in 23,000 equal-width cells on $11 < t_6 < 34$, that is, (11.000, 11.001), [11.001, 11.002), . . . , [33.999, 34.000).

4. ANALYTICAL APPROACHES

Although straightforward to apply, the simulation approach has a distinct drawback. Each additional digit of accuracy in the estimate of $E[T_n]$, for example, requires approximately a 100-fold increase in replications due to the square root in the denominator of the confidence interval formulas for $E[T_n]$. The next two subsections outline efforts to derive the various performance measures analytically, eliminating the need for simulation.

4.1. Series-Parallel Networks

A series-parallel activity network is a special case of an activity network that can be simplified by a sequence of reductions to a simple network consisting of one arc and

TABLE 5. Estimated criticality $\hat{\rho}_{ij}$ for one million replications and 95% confidence interval half-width $\hat{h}$ for the network shown in Figure 1 with arc durations from Table 1.

Arc	Paths	$\hat{\rho}_{ij}$	$\hat{h}$
a_{12}	π_2, π_3, π_6	0.600	0.0010
a_{13}	π_1, π_5	0.203	0.0008
a_{14}	π_4	0.198	0.0008
a_{25}	π_3	0.129	0.0007
a_{23}	π_2, π_6	0.469	0.0010
a_{36}	π_1, π_2	0.244	0.0008
a_{34}	π_5, π_6	0.429	0.0010
a_{56}	π_3	0.129	0.0007
a_{46}	π_4, π_5, π_6	0.627	0.0010

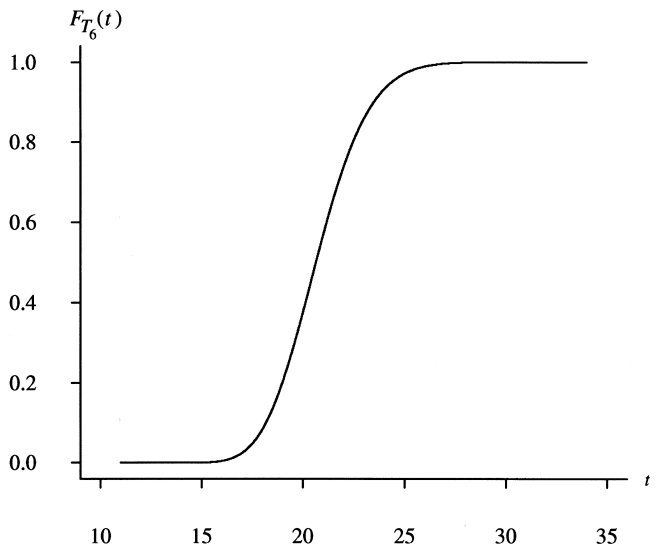

FIG. 2. Empirical CDF of T_6 for one million replications for the network shown in Figure 1 with arc durations from Table 1.

two nodes. Reductions consist of taking two arcs, either in series or in parallel, and replacing them with a single arc that has a random duration whose distribution is calculated in the algorithm. Once the network is completely reduced, the distribution of the single remaining arc is the distribution of the time T_n to complete the original network. The recursive algorithm then reconstructs the network to determine the critical path probabilities and arc criticalities. Parallel and series reductions and reconstructions are described below. Our algorithm will use these operations to simplify a series-parallel network, and then, by determining the distribution of T_n and reconstructing the network, determine the critical path probabilities and the criticalities. The earliest work on the exact analysis of series-parallel networks is Martin [9].

4.1.1. Parallel Reduction. The algorithm described in Section 4.1.5 is capable of processing two arcs in *parallel*, **F3** as illustrated in Figure 3. A parallel reduction is the process of combining these two arcs into a single arc. Let X_{ij} and Y_{ij} denote the random arc durations. Without loss of generality, if $T_i = 0$, then $T_j = \max\{X_{ij}, Y_{ij}\}$. So

$$F_{T_j}(t) = \Pr[T_j \le t] = \Pr[X_{ij} \le t \text{ and } Y_{ij} \le t] = F_{X_{ij}}(t)F_{Y_{ij}}(t)$$

on the support of T_j. The portion of the algorithm given in Section 4.1.5 prior to the second recursive call uses this formula to reduce two arcs in parallel to a single arc.

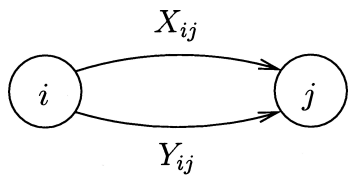

FIG. 3. Two arcs in parallel.

4.1.2. Parallel Reconstruction. Consider the single arc a_{ij} resulting from a parallel reduction. If the probability ρ_{ij} that this arc is on the critical path is known, then this probability can be allocated to the two original parallel arcs x and y based on each arc's activity duration CDF:

$$\rho_x = \Pr[X_{ij} > Y_{ij}] \cdot \rho_{ij} = \Pr[X_{ij} - Y_{ij} > 0] \cdot \rho_{ij}$$

and

$$\rho_y = \rho_{ij} - \rho_x.$$

This calculation involves determining the distribution of the difference between the two random variables. The portion of the algorithm given in Section 4.1.5 after the second recursive call calculates the distribution of $X_{ij} - Y_{ij}$ and the probability that this random variable is positive.

4.1.3. Series Reduction. The algorithm described in this subsection finds the CDF of the time to complete two arcs in *series*, as illustrated in Figure 4. Let Y_{ij} and Y_{jk} denote **F4** the durations of the two arcs with CDFs $F_{Y_{ij}}(t)$ and $F_{Y_{jk}}(t)$. Without loss of generality, if $T_i = 0$, then $T_k = Y_{ij} + Y_{jk}$. The CDF of T_k is ([2], p. 215)

$$F_{T_k}(t) = \Pr[T_k \le t] = \int_0^t F_{Y_{ij}}(t - y_{jk}) f_{Y_{jk}}(y_{jk})\, dy_{jk}.$$

4.1.4. Series Reconstruction. Consider the single arc a_{ik} resulting from a series reduction. For any two arcs a_{ij} and a_{jk} that have been reduced into a single arc a_{ik}, $\rho_{ij} = \rho_{jk} = \rho_{ik}$. If two arcs in series are along the critical path, then so is the respective arc resulting from the reduction.

4.1.5. Algorithm. Two arcs in series are detected by searching the rows of the node-arc incidence matrix N for the first row containing exactly two nonzero elements whose sum is 0. Such a row represents a single arc entering and a single arc exiting the given node. To reduce a series of two arcs to a single arc, the algorithm

(1) zeros out the row of N corresponding to the deleted middle node,
(2) links the first node of the series directly to the last node of the series, and
(3) computes the distribution of the sum of the durations of the two arcs.

Two arcs in parallel are identified by two identical nonzero columns of N, which represent two arcs linking the same two nodes. The algorithm iterates through each column and if parallel arcs are found, the algorithm

(1) computes the distribution of the maximum of the two arc durations, and
(2) zeros out the column of N corresponding to the eliminated arc.

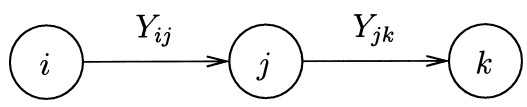

FIG. 4. Two arcs in series.

The following recursive algorithm reduces a series-parallel network, where N is the node-arc incidence matrix of the network. The PDFs for the activity durations of each arc are stored in a vector f (indexed in the same way that the arcs are indexed to correspond to the columns of the matrix N), and the criticalities of each arc are stored in a vector C. This algorithm recursively performs one arc reduction at a time, arbitrarily giving priority to series reductions over parallel reductions, until the network consists of just one arc. At this point, the value of C for that one arc is set to one and then the network is reconstructed on the return calls in reverse order, determining values of C for each pair of series or parallel arcs based on the value of C for the reduced arc. The algorithm returns the distribution of the time to complete the network and the final value of C, which contains ρ_{ij} for each arc a_{ij} in the network.

Parameters: $n \times m$ node-arc incidence matrix N, PDFs of m activity durations Y_{ij} stored in the vector f, number of nodes n, number of arcs m, number of arcs to be reduced *arcsremaining* (initialized to m)

Procedure name: `GetCriticalities`

```
local C, i, j, rowsum, absrowsum, negidx, posidx, c, d, l, k, empty, numdifference
if (arcsremaining = 1)                                   if network reduced to one arc
    for (i ← 1; i ≤ n; i ← i + 1)                        loop through rows of N
        for (k ← 1; k ≤ m; k ← k + 1)                    loop through columns of N
            if (N[i, k] ≠ 0)                             if true then arc k is the last existing arc
                C[k] ← 1                                 set criticality to 1
                print(CDF(f[k]))                         print CDF of project completion time
                return(C)                                return criticalities
for (i ← 1; i ≤ n; i ← i + 1)                            begin series reduction
    rowsum ← 0                                           initialize rowsum
    absrowsum ← 0                                        initialize sum of absolute values in a row
    for (k ← 1; k ≤ m; k ← k + 1)                        loop through columns of N
        rowsum ← rowsum + N[i, k]                        sum values in the ith row of N
        absrowsum ← absrowsum + abs(N[i, k])             sum absolute values of ith row of N
        if (N[i, k] = −1)                                if arc k enters node i
            negidx ← k                                   store the index of arc k
        if (N[i, k] = 1)                                 if arc k leaves from node i
            posidx ← k                                   store the index of arc k
    if (rowsum = 0 and absrowsum = 2)                    if only one arc enters and leaves ith node
        c ← f[negidx]                                    save PDF of first arc for reconstruction
        d ← f[posidx]                                    save PDF of second arc for reconstruction
        f[negidx] ← Convolution(c, d)                    combine the arcs into a single arc
        for (j ← 1; j ≤ n; j ← j + 1)                    loop through rows of N
            N[j, negidx] ← N[j, negidx] + N[j, posidx]   change node that arc enters
            N[j, posidx] ← 0                             delete middle node
        arcsremaining ← arcsremaining − 1                decrement number of arcs in network
        C ← GetCriticalities(N, f, n, m, arcsremaining)  recursive call with new network
        for (j ← 1; j ≤ n; j ← j + 1)                    begin series reconstruction
            if (N[j, negidx] = −1)                       if arc negidx enters node j
                N[j, negidx] ← 0                         delete arc made previously by reduction
                N[j, posidx] ← −1                        delete arc made previously by reduction
        N[i, negidx] ← −1                                reconstruct node deleted previously by reduction
        N[i, posidx] ← 1                                 reconstruct node deleted previously by reduction
        f[negidx] ← c                                    reconstruct arc duration
        f[posidx] ← d                                    reconstruct arc duration
        C[posidx] ← C[negidx]                            reconstruct arc criticality
        return(C)                                        return criticalities
for (l ← 1; l ≤ m; l ← l + 1)                            begin parallel reduction
    for (k ← l + 1; k ≤ m; k ← k + 1)                    loop through columns of N
        empty ← 0                                        initialize empty
        numdifference ← 0                                initialize numdifference
        for (i ← 1; i ≤ n; i ← i + 1)                    loop through rows of N
            if (N[i, l] ≠ N[i, k])                       if column difference found
                numdifference ← 1                        set boolean variable numdifference
```

```
    if (N[i, l] ≠ 0)                                          if a column is not empty
        empty ← 1                                             set boolean variable empty
if (numdifference = 0 and empty ≠ 0)                          if columns l and k are identical and nonzero
    c ← f[l]                                                  save PDF of arc l for reconstruction
    d ← f[k]                                                  save PDF of arc k for reconstruction
    f[l] ← Maximum(c, d)                                      store PDF of the maximum
    for (i ← 1; i ≤ n; i ← i + 1)                             loop through rows of N
        N[i, k] ← 0                                           zero out column of removed arc
    arcsremaining ← arcsremaining − 1                         decrement number of arcs in network
    C ← GetCriticalities(N, f, n, m, arcsremaining)           recursive call with new network
    for (i ← 1; i ≤ n; i ← i + 1)                             begin parallel reconstruction
        N[i, k] ← N[i, l]                                     reinsert arc k by copying column l
    f[l] ← c                                                  rebuild distribution
    f[k] ← d                                                  rebuild distribution
    C[k] ← CDF(Difference(f[l],f[k]), 0) · C[l]               calculate Pr[X_il − Y_il > 0] · ρ_il
    C[l] ← C[l] − C[k]                                        calculate ρ_y = ρ_il − ρ_x
    return(C)                                                 return criticalities
return(C)                                                     return criticalities
```

This algorithm requires symbolic processing capability to calculate the distribution of the maximum of two independent random variables for parallel reduction and the distribution of the sum of two independent random variables for series reduction. The Maple-based APPL language [5] has procedures `Maximum` and `Convolution` that can be used for these operations. The algorithm has been implemented in APPL and is available from the first author.

F5 **4.1.6. Example.** Figure 5 shows an example of a series-parallel network from Elmaghraby ([3], p. 261). The network can be reduced and reconstructed as illustrated in **F6** Figure 6.

According to the algorithm in Section 4.1.5, if the duration of each arc Y_{ij} is an exponential(b) random variable, where $1/b$ is the mean, then the CDF of the time to complete the network (T_5) is

$$F_{T_5}(t) = 1 - 3bte^{-bt} - \frac{b^2t^2}{2}e^{-bt} - 3e^{-2bt} + \frac{5b^2t^2}{2}e^{-2bt} + \frac{b^3t^3}{2}e^{-2bt} + 2e^{-3bt} + 3bte^{-3bt} + b^2t^2e^{-3bt}$$

F7 for $t > 0$. This CDF is plotted in Figure 7 for $b = 2$.

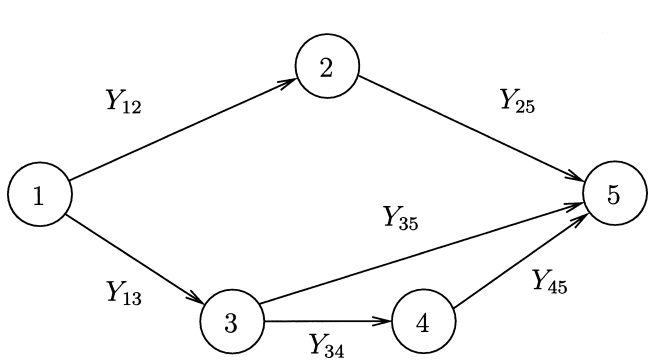

FIG. 5. Series-parallel network from Elmaghraby.

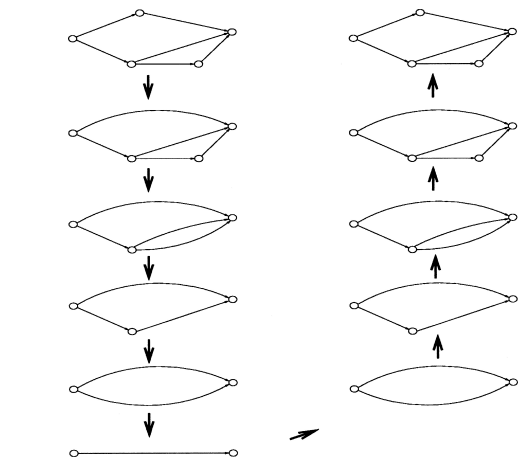

FIG. 6. Reduction and reconstruction of the series-parallel network shown in Figure 5.

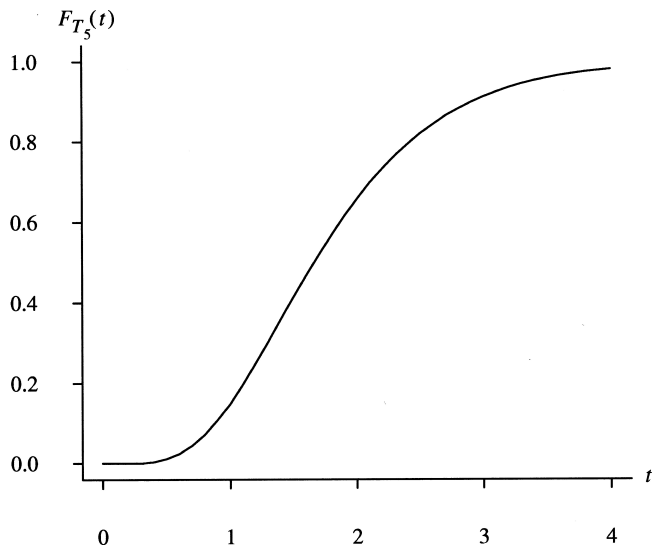

FIG. 7. CDF of T_5 for the series-parallel network shown in Figure 5 with $b = 2$.

The $r = 3$ paths in the network are listed in Table 6, along with the critical path probabilities. Table 7 contains the criticalities for the $m = 6$ arcs when $b = 2$. (Fisher, Saisi, and Goldstein [4] determine criticalities when activity durations are IID gamma random variables.)

This example has a small number of nodes and arcs along with simple distributions for the arc durations so as to make the final CDF for the project duration simple. If, however, the arc durations are each independent Triangular(0, 1, 2) random variables, then the problem of determining the distribution of T_5 becomes much more tedious. The arc criticalities for this example with Triangular(0, 1, 2) arc durations are shown in Table 8. The CDF of the project completion time is

$$F_{T_5}(t) = \begin{cases} 0 & t \le 0 \\ \frac{1}{64512}t^{12} & 0 < t \le 1 \\ -\frac{1}{21504}t^{12} + \frac{5}{4032}t^{11} - \frac{1357}{120960}t^{10} + \frac{1553}{30240}t^{9} - \frac{2431}{17280}t^{8} + \frac{15389}{60480}t^{7} & \\ \quad -\frac{5531}{17280}t^{6} + \frac{1237}{4320}t^{5} - \frac{44039}{241920}t^{4} + \frac{607}{7560}t^{3} - \frac{13}{560}t^{2} + \frac{59}{15120}t - \frac{17}{60480} & 1 < t \le 2 \\ -\frac{5}{21504}t^{12} + \frac{5}{672}t^{11} - \frac{1813}{17280}t^{10} + \frac{8693}{10080}t^{9} - \frac{556873}{120960}t^{8} + \frac{1018271}{60480}t^{7} & \\ \quad -\frac{749309}{17280}t^{6} + \frac{89059}{1120}t^{5} - \frac{8332907}{80640}t^{4} + \frac{156139}{1680}t^{3} - \frac{414913}{7560}t^{2} + \frac{57587}{3024}t - \frac{176249}{60480} & 2 < t \le 3 \\ -\frac{1}{21504}t^{12} + \frac{25}{12096}t^{11} - \frac{499}{12096}t^{10} + \frac{703}{1440}t^{9} - \frac{556873}{120960}t^{8} + \frac{1018271}{60480}t^{7} & \\ \quad -\frac{672253}{8640}t^{6} + \frac{699347}{3360}t^{5} - \frac{2587009}{6720}t^{4} + \frac{197879}{420}t^{3} - \frac{376261}{1080}t^{2} + \frac{94931}{756}t - \frac{68353}{7560} & 3 < t \le 4 \\ \frac{1}{144}t^{6} - \frac{1}{5}t^{5} + \frac{19}{8}t^{4} - \frac{89}{6}t^{3} + \frac{409}{8}t^{2} - \frac{1829}{20}t + \frac{7969}{120} & 4 < t \le 5 \\ -\frac{1}{720}t^{6} + \frac{1}{20}t^{5} - \frac{3}{4}t^{4} + 6\,t^{3} - 27\,t^{2} + \frac{324}{5}t - \frac{319}{5} & 5 < t \le 6 \\ 1 & t > 6. \end{cases}$$

4.2. *Nonseries-Parallel Networks*

Now consider the case of a nonseries-parallel network. Determining the distribution of the time to complete the network is complicated by the fact that the network cannot be reduced as in the series-parallel case. We begin with an example that illustrates the difficulty.

TABLE 6. Paths π_k and critical path probabilities $p(\pi_k)$ for the series-parallel network shown in Figure 5 with exponentially distributed arc durations with $b = 2$.

k	Node sequence	π_k	$p(\pi_k)$
1	$1 \to 2 \to 5$	$\{a_{12}, a_{25}\}$	$115/432 \cong 0.266$
2	$1 \to 3 \to 5$	$\{a_{13}, a_{35}\}$	$317/1728 \cong 0.183$
3	$1 \to 3 \to 4 \to 5$	$\{a_{13}, a_{34}, a_{45}\}$	$317/576 \cong 0.550$

TABLE 7. Criticalities ρ_{ij} for the series-parallel network shown in Figure 5 with exponentially distributed arc durations with $b = 2$.

Arc	Paths	ρ_{ij}
a_{12}	π_1	$115/432 \cong 0.266$
a_{13}	π_2, π_3	$317/432 \cong 0.734$
a_{25}	π_1	$115/432 \cong 0.266$
a_{35}	π_2	$317/1728 \cong 0.183$
a_{34}	π_3	$317/576 \cong 0.550$
a_{45}	π_3	$317/576 \cong 0.550$

4.2.1. Example 1: Bridge-Plus Network. Elmaghraby ([3], p. 305) considers the network shown in Figure 8. The activity durations are exponentially distributed with means

TABLE 8. Criticalities ρ_{ij} for the series-parallel network shown in Figure 5 with Triangular(0, 1, 2) arc durations.

Arc	Paths	ρ_{ij}
a_{12}	π_1	$30233/237600 \cong 0.127$
a_{13}	π_2, π_3	$207367/237600 \cong 0.873$
a_{25}	π_1	$30233/237600 \cong 0.127$
a_{35}	π_2	$6013643/85536000 \cong 0.070$
a_{34}	π_3	$68638477/85536000 \cong 0.802$
a_{45}	π_3	$68638477/85536000 \cong 0.802$

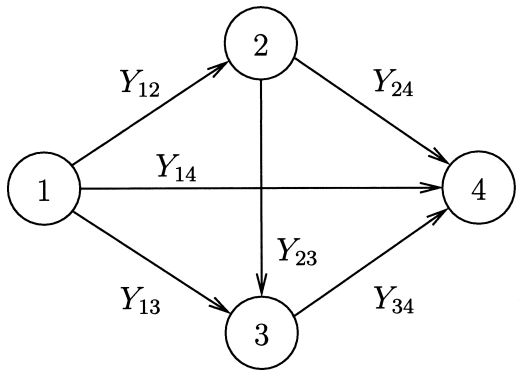

FIG. 8. A four-node, six-arc stochastic activity network.

of 5 (for Y_{12}, Y_{24}, Y_{34}) and 10 (for Y_{13}, Y_{14}, Y_{23}). There are four paths through the network, with random durations

$$L_1 = Y_{12} + Y_{24},$$
$$L_2 = Y_{12} + Y_{23} + Y_{34},$$
$$L_3 = Y_{13} + Y_{34},$$
$$L_4 = Y_{14}.$$

Because only Y_{12} and Y_{34} correspond to arcs that lie on more than one path, the conditional CDF of the time to complete the network T_4, given $Y_{12} = y_{12}$ and $Y_{34} = y_{34}$, is

$$F_{T_4}(t|y_{12}, y_{34})$$
$$= F_{L_1}(t|y_{12}, y_{34}) F_{L_2}(t|y_{12}, y_{34}) F_{L_3}(t|y_{12}, y_{34}) F_{L_4}(t|y_{12}, y_{34})$$

because, when $Y_{12} = y_{12}$ and $Y_{34} = y_{34}$ are fixed,

$$\begin{aligned}\Pr(T_4 \le t) &= \Pr(\max\{L_1, L_2, L_3, L_4\} \le t)\\ &= \Pr(L_1 \le t, L_2 \le t, L_3 \le t, L_4 \le t)\\ &= \Pr(L_1 \le t)\Pr(L_2 \le t)\Pr(L_3 \le t)\Pr(L_4 \le t).\end{aligned}$$

The CDFs for

$$L_1 = y_{12} + Y_{24},$$
$$L_2 = y_{12} + Y_{23} + y_{34},$$
$$L_3 = Y_{13} + y_{34},$$
$$L_4 = Y_{14},$$

conditioned on $Y_{12} = y_{12}$ and $Y_{34} = y_{34}$, are

$$F_{L_1}(t|y_{12}, y_{34}) = \begin{cases} 0 & t < y_{12} \\ 1 - e^{-(t-y_{12})/5} & t \ge y_{12}, \end{cases}$$

$$F_{L_2}(t|y_{12}, y_{34}) = \begin{cases} 0 & t < y_{12} + y_{34} \\ 1 - e^{-(t-y_{12}-y_{34})/10} & t \ge y_{12} + y_{34}, \end{cases}$$

$$F_{L_3}(t|y_{12}, y_{34}) = \begin{cases} 0 & t < y_{34} \\ 1 - e^{-(t-y_{34})/10} & t \ge y_{34}, \end{cases}$$

$$F_{L_4}(t|y_{12}, y_{34}) = \begin{cases} 0 & t < 0 \\ 1 - e^{-t/10} & t \ge 0. \end{cases}$$

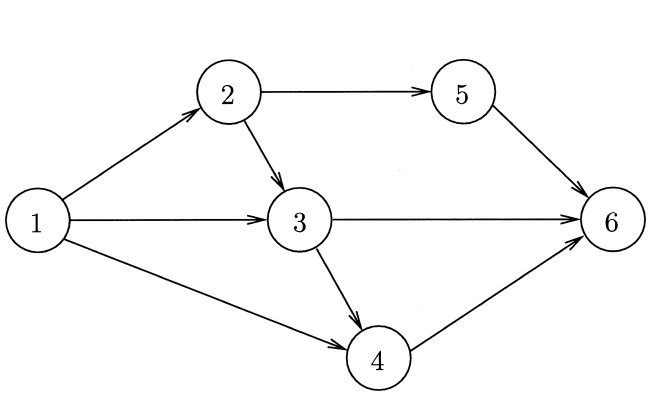

FIG. 9. Original activity network.

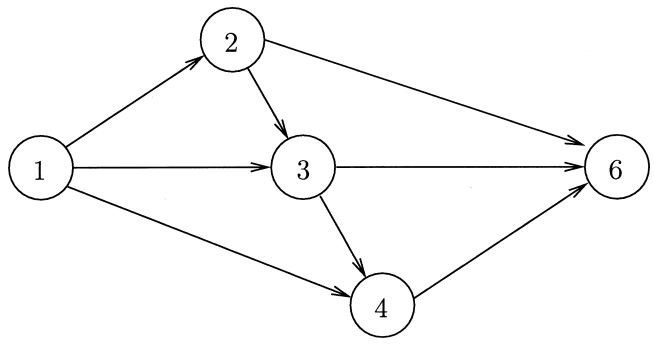

FIG. 10. Activity network from Figure 9 after reduction.

Thus, the unconditional CDF of T_4 is given by

$$F_{T_4}(t) = \int_0^t \int_0^{t-y_{12}} F_{T_4}(t|y_{12}, y_{34}) f_{Y_{12}}(y_{12}) f_{Y_{34}}(y_{34})\, dy_{34}\, dy_{12},$$

where the limits are chosen to satisfy $y_{34} \le t - y_{12}$ from the second portion of the support of L_2. This integral yields

$$F_{T_4}(t) = 1 - 7e^{-t/10} + 12e^{-t/5} + \frac{2t}{5}e^{-t/5} - 16e^{-3t/10} + 19e^{-2t/5} - 9e^{-t/2} - \frac{2t}{5}e^{-t/2}$$

for $t > 0$.

The Maple code for evaluating the double integral in this example is given in Appendix A. Although this example was particularly easy because all of the activity durations had support on $(0, \infty)$, it leads to the development of a more general algorithm for the nonseries-parallel case.

4.2.2. Algorithm. The difficulty with nonseries-parallel networks lies in the inability to reduce the network to a single arc. Conditioning must be used to create a general algorithm to solve for the distribution of the time to complete a nonseries-parallel stochastic activity network with continuous activity durations. Any such network should first be processed by the series-parallel algorithm, thereby either finding the distribution of the time to complete the network or leaving a nonseries-parallel network that would then go through the conditioning steps.

A conceptual presentation of the algorithm for the case of arc distributions with support $(0, \infty)$ is given by the five steps shown below. Each step is illustrated with the network from Pritsker (Fig. 1), which is given without the activity durations in Figure 9. **F9**

STEP 1: Series-parallel reduction. Perform all possible series-parallel reductions. One series reduction is possible, which eliminates node 5 by using the convolution formula described earlier, as illustrated in Figure 10. **F10**

STEP 2: Determine arc multiplicities. For each arc, the multiplicity is the number of paths that contain that arc, as illustrated in Figure 11. **F11**

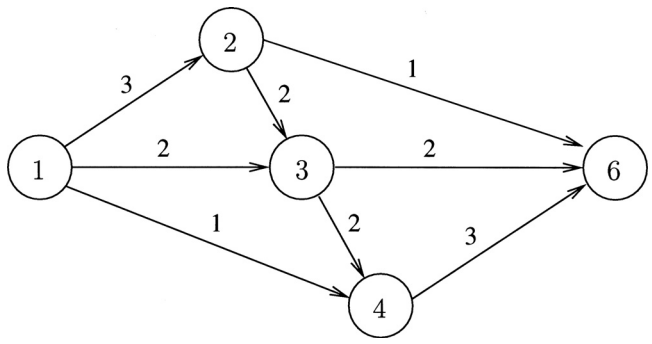

FIG. 11. Activity network from Figure 10 with arc multiplicities.

STEP 3: Condition on the arc durations associated with multiplicities greater than one. The random variables $L_1, L_2, \ldots, L_r$ denote the random path lengths associated with paths $\pi_1, \pi_2, \ldots, \pi_r$. The duration of an arc a_{ij} is random, and is denoted by the upper-case Y_{ij}, if the arc has a multiplicity of one. The arc duration is conditioned on, and is denoted by the lower-case y_{ij}, if the arc a_{ij} has a multiplicity greater than one. For the example,

$$L_1 = y_{12} + Y_{26}, \qquad L_2 = y_{12} + y_{23} + y_{36},$$
$$L_3 = y_{12} + y_{23} + y_{34} + y_{46}, \qquad L_4 = y_{13} + y_{36},$$
$$L_5 = y_{13} + y_{34} + y_{46}, \qquad L_6 = Y_{14} + y_{46}.$$

STEP 4: Determine integration region associated with the CDF of T_n. Let Q be the set of arcs with multiplicities greater than one and $q = |Q|$. Here, $q = 6$. The CDF of T_n is given by the q-fold integral

$$F_{T_n}(t) = \Pr(T_n \le t) = \int\int\cdots\int F_{T_n}(t|y_{ij} \in Q) \prod_{(i,j)|a_{ij}\in Q} f_{Y_{ij}}(y_{ij})dy_{ij}.$$

The integration region, given by the following constraints, depends on the conditional arc durations in the definitions of the random path lengths $L_1, L_2, \ldots, L_r$ defined in Step 3:

$$y_{12} \le t, \quad y_{12} + y_{23} + y_{36} \le t, \quad y_{12} + y_{23} + y_{34} + y_{46} \le t,$$
$$y_{13} + y_{36} \le t, \quad y_{13} + y_{34} + y_{46} \le t, \quad y_{46} \le t.$$

After eliminating redundant constraints, the integration region associated with $T_6 \le t$ is described by

$$y_{12} + y_{23} + y_{36} \le t, \qquad y_{12} + y_{23} + y_{34} + y_{46} \le t,$$
$$y_{13} + y_{36} \le t, \qquad y_{13} + y_{34} + y_{46} \le t.$$

STEP 5: Perform the integration. Because there is more than one constraint, determining the limits of integration can be complicated. The procedure `GetDistribution` returns the CDF of the network by conditioning the completion time for the network on the duration of arcs whose multiplicities exceed one. The algorithm examines all r paths, keeping track of the arcs on each path by using a matrix *Paths* passed to the algorithm. *Paths* is an $r \times m$ matrix with rows corresponding to paths and columns corresponding to the arcs along the respective paths. The first 0 entry in any row of *Paths* means that there are no more arcs along that path. For the network in Figure 10, where arcs correspond to columns 1 through 8 in the node-arc incidence matrix in the order a_{12}, a_{13}, a_{14}, a_{23}, a_{34}, a_{26}, a_{36}, a_{46}, the matrix *Paths* is

$$Paths = \begin{bmatrix} 1 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 7 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 5 & 8 & 0 & 0 & 0 & 0 \\ 2 & 7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 5 & 8 & 0 & 0 & 0 & 0 & 0 \\ 3 & 8 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Determining the integration limits is complicated because the geometry of the integration region is complex. The limits must be handled in a case-by-case manner. Each path may or may not have an arc with multiplicity one, although there is at most one string of consecutive arcs with multiplicity one along each path. The limits of the integral must be hard-coded into the program for each particular case. The development of an algorithm for automatic determination of the limits is addressed in Section 4.3. We assume that the network has been passed through series-parallel reduction (Step 1). In this algorithm, f stores the PDFs of the arc durations, F stores the CDFs of the arc durations, *frequency* stores the number of paths each arc is on, *count* stores the number of arcs on each path, *amtmultiple* stores the number of multiple-use arcs on each path, *multiple* stores the multiple-use arcs on each path, and *single* stores the single-use arcs, if any exist, on each path.

Parameters: The number of paths r through the reduced network, the number of arcs m, an $r \times m$ array *Paths* that holds the ordered arcs for each path, $f_{Y_{ij}}(y_{ij})$ and $F_{Y_{ij}}(y_{ij})$ for all arcs a_{ij}.

Procedure name: `GetDistribution`

local $k, l,$ *frequency*$[m]$, *count*$[r]$, *amtmultiple*$[r]$, *multiple*$[r, m]$, *single*$[r]$, q, g
for ($l \leftarrow 1; l \le r; l \leftarrow l + 1$) — loop through all paths of network
  count$[l] \leftarrow 0$ — initialize *count*
  single$[l] \leftarrow 0$ — initialize *single*
  amtmultiple$[l] \leftarrow 0$ — initialize *amtmultiple*
  for ($k \leftarrow 1; k \le m; k \leftarrow k + 1$) — loop through all arcs of network

```
        multiple[l, k] ← 0                                                  initialize multiple
for (k ← 1; k ≤ m; k ← k + 1)                                               loop through all arcs of network
    frequency[k] ← 0                                                        initialize frequency
for (l ← 1; l ≤ r; l ← l + 1)                                               loop through all paths of network
    k ← 1                                                                   begin Step 2
    while (Paths[l, k] ≠ 0)                                                 while more arcs on path
        frequency[Paths[l, k]] ← frequency[Paths[l, k]] + 1                 number of paths arc is on
        k ← k + 1                                                           move to next arc along path
    count[l] ← k − 1                                                        store number of arcs on path l
for (l ← 1; l ≤ r; l ← l + 1)                                               loop through all paths of network
    for (k ← 1; k ≤ count[l]; k ← k + 1)                                    loop through all arcs on path l
        if (frequency[Paths[l, k]] > 1)                                     if arc is on multiple paths
            amtmultiple[l] ← amtmultiple[l] + 1                             number of multiple-use arcs on path l
            multiple[l, amtmultiple[l]] ← Paths[l, k]                       store arcs on multiple paths
        if (frequency[Paths[l, k]] = 1)                                     if arc is on only one path
            single[l] ← Paths[l, k]                                         store arc on one path
q ← 0                                                                       initialize the number of arcs on multiple paths in whole network
g ← 1                                                                       initialize the integrand g
for (k ← 1; k ≤ m; k ← k + 1)                                               loop through all arcs
    if (frequency[k] > 1)                                                   if arc is on multiple paths
        q ← q + 1                                                           increment number arcs on multiple paths
        g ← g ∗ f[k](y_k)                                                   multiply PDFs into integrand
for (l ← 1; l ≤ r; l ← l + 1)                                               begin Step 3: loop through all paths of network
    if (single[l] ≠ 0)                                                      if an arc with multiplicity one is present on path
        g ← g ∗ F[single[l]](t − Σ_{multiple[l,k]>0} y_{multiple[l,k]})      multiply CDFs into integrand
return(∫ ∫ ... ∫ g Π_{k | frequency[k]>1} dy_k)                             begin Steps 4 and 5: evaluate q-fold integral
```

To execute `GetDistribution` for the network in Figure 11, it is called with $r = 6$, $m = 8$, *Paths* given previously, the PDF $f_{Y_{ij}}(t)$ of each arc, and the CDF $F_{Y_{ij}}(t)$ of each arc. The order of the eight arc durations in the data structure is $y_{12}, y_{13}, y_{14}, y_{23}, y_{34}, y_{26}, y_{36}, y_{46}$. The value of the integrand after the code is executed is

$$g = f_{Y_{12}}(y_{12}) f_{Y_{13}}(y_{13}) f_{Y_{23}}(y_{23}) f_{Y_{34}}(y_{34}) \times f_{Y_{36}}(y_{36}) f_{Y_{46}}(y_{46}) F_{Y_{26}}(t - y_{12}) F_{Y_{14}}(t - y_{46}).$$

The values in the data structures after the code is executed are

$$\begin{aligned} \mathit{frequency} &= (3, 2, 1, 2, 2, 1, 2, 3)', \\ \mathit{count} &= (2, 3, 4, 2, 3, 2)', \\ \mathit{amtmultiple} &= (1, 3, 4, 2, 3, 1)', \\ \mathit{single} &= (6, 0, 0, 0, 0, 3)', \\ \mathit{multiple} &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 7 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 5 & 8 & 0 & 0 & 0 & 0 \\ 2 & 7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 5 & 8 & 0 & 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \end{aligned}$$

The problem of determining the exact distribution of the completion time of a stochastic activity network has now been reduced to setting up the limits of integration associated with a set of linear constraints. Although algorithms exist to determine the extreme points of the integration region, we are not aware of any existing algorithm for converting the constraints to integration limits.

We conclude this subsection with an example of activity durations that have finite support, which illustrates the problems associated with setting the integration limits.

4.2.3. Example 2: Bridge Network. Consider the network in Figure 12, where all Y_{ij} are $U(0, 1)$ random variables. **F12**
As in Example 1, we again condition on the values of y_{12} and y_{34}, yielding the CDFs for

$$L_1 = y_{12} + Y_{24}, \quad L_2 = y_{12} + Y_{23} + y_{34}, \quad L_3 = Y_{13} + y_{34}$$

as

$$F_{L_1}(t|y_{12}, y_{34}) = \begin{cases} 0 & t < y_{12} \\ t - y_{12} & y_{12} \le t \le y_{12} + 1 \\ 1 & t > y_{12} + 1, \end{cases}$$

$$F_{L_2}(t|y_{12}, y_{34}) = \begin{cases} 0 & t < y_{12} + y_{34} \\ t - y_{12} - y_{34} & y_{12} + y_{34} \le t \le y_{12} + y_{34} + 1 \\ 1 & t > y_{12} + y_{34} + 1, \end{cases}$$

$$F_{L_3}(t|y_{12}, y_{34}) = \begin{cases} 0 & t < y_{34} \\ t - y_{34} & y_{34} \le t \le y_{34} + 1 \\ 1 & t > y_{34} + 1. \end{cases}$$

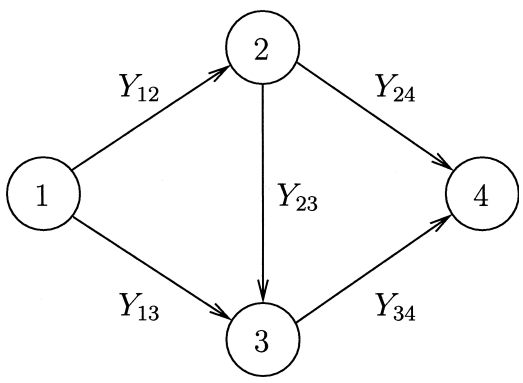

FIG. 12. The bridge network.

Procedure `GetDistribution` returns the CDF of T_4:

$$F_{T_4}(t) = \int\int F_{L_1}(t|y_{12}, y_{34}) F_{L_2}(t|y_{12}, y_{34}) \times F_{L_3}(t|y_{12}, y_{34}) f_{Y_{12}}(y_{12}) f_{Y_{34}}(y_{34})\, dy_{34}\, dy_{12}.$$

The support of T_4 is $0 < t_4 \leq 3$. The limits of integration are more complicated than in Example 1. For $0 < t \leq 1$,

$$F_{T_4}(t) = \int_0^t \int_0^{t-y_{12}} (t - y_{12}) \times (t - y_{12} - y_{34})(t - y_{34}) \cdot 1 \cdot 1\, dy_{34}\, dy_{12},$$

similar to Example 1. For $1 < t \leq 2$, five separate integrals
F13 are required. Figure 13 illustrates regions in the (y_{12}, y_{34}) coordinate system in which the form of the integrand remains fixed for $t = 1.8 \in (1, 2]$. Expanding the integral yields

$$\begin{aligned}
F_{T_4}(t) = &\int_0^{t-1} \int_0^{t-1-y_{12}} 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1\, dy_{34}\, dy_{12} && \text{I}\\
&+ \int_0^{t-1} \int_{t-1-y_{12}}^{t-1} 1 \cdot (t - y_{12} - y_{34}) \cdot 1 \cdot 1 \cdot 1\, dy_{34}\, dy_{12} && \text{II}\\
&+ \int_0^{t-1} \int_{t-1}^{1} 1 \cdot (t - y_{12} - y_{34})(t - y_{34}) \cdot 1 \cdot 1\, dy_{34}\, dy_{12} && \text{III}\\
&+ \int_{t-1}^{1} \int_0^{t-1} (t - y_{12})(t - y_{12} - y_{34}) \cdot 1 \cdot 1 \cdot 1\, dy_{34}\, dy_{12} && \text{IV}\\
&+ \int_{t-1}^{1} \int_{t-1}^{t-y_{12}} (t - y_{12})(t - y_{12} - y_{34})(t - y_{34}) \cdot 1 \cdot 1\, dy_{34}\, dy_{12} && \text{V}
\end{aligned}$$

for $1 < t \leq 2$. The Roman numerals at the right of each double integral denote the corresponding region in Figure 13. Finally, for $2 < t \leq 3$,

$$F_{T_4}(t) = 1 - (1 - t + 2)^2/2 + \int_{t-1}^{1} \int_{t-1-y_{12}}^{1} 1 \cdot (t - y_{12} - y_{34}) \cdot 1 \cdot 1 \cdot 1\, dy_{34}\, dy_{12},$$

which yields

$$F_{T_4}(t) = \begin{cases} 0 & t \leq 0 \\ \frac{11}{120} t^5 & 0 < t \leq 1 \\ -\frac{1}{120} t^5 - \frac{1}{6} t^4 + \frac{2}{3} t^3 - \frac{1}{3} t^2 - \frac{1}{6} t + \frac{1}{10} & 1 < t \leq 2 \\ \frac{1}{6} t^3 - \frac{3}{2} t^2 + \frac{9}{2} t - \frac{23}{6} & 2 < t \leq 3 \\ 1 & t > 3. \end{cases}$$

This result can be computed by integrating directly as in Example 1 (the code is given in Appendix B) or it can be computed by using Maple's `piecewise` capability (the code is given in Appendix C).

4.3. A General Algorithm

The remainder of this article is devoted to the development of a general algorithm, including integration limits, that can be used to find the distribution of the time to complete a stochastic activity network by conditioning on the duration of activities that appear on multiple paths. Because of the manipulations involved in solving for the distribution, it is necessary to place two additional restrictions on the networks being considered: first, all nodes, except the terminal node, have at most two incident arcs; second, each activity duration must be described by a PDF expressed as a single function defined on $(0, \infty)$, as opposed to a function that is defined in a piecewise manner. As long as these requirements are met by a network, the algorithm described here can be used to find the distribution of the time needed to complete that network. We begin with some additional notational conventions.

In the algorithm, arcs appearing on multiple paths (multiple-use arcs) are distinguished from arcs appearing on only one path (single-use arcs). Because the durations of activities appearing on multiple paths are conditioned upon, such durations must be denoted by variables. For convenience, these variables are used as the names of the arcs as

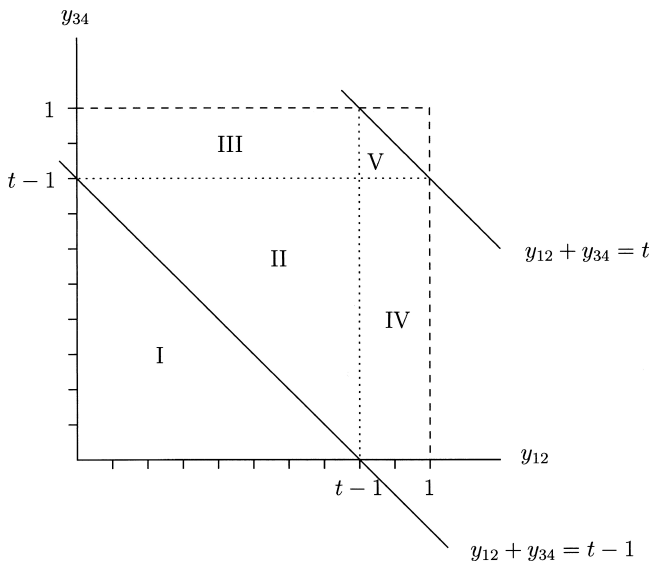

FIG. 13. Integration regions associated with $t = 1.8$.

well, namely $x_1, x_2, \ldots, x_k$, where k is the number of arcs appearing on multiple paths. This new notation for multiple-use arcs and their activity durations will be used instead of the previous designations a_{ij} and Y_{ij} to simplify expressions that follow and facilitate the use of certain ordering principles that will be required for the algorithm.

Three separate phases occur in the algorithm. The first phase sets up the limits of integration of several k-fold integrals. The second phase constructs the integrand that will be used in all of the k-fold integrals. The final phase evaluates the integrals. Because the limits of integration are determined before the integrand is constructed, when the k-fold integrals are shown, the symbol $I(\boldsymbol{x})$ will be used initially for the integrand and $d\boldsymbol{x}$ for the product of the differentials, e.g., $d\boldsymbol{x} = dx_1\, dx_2\, dx_3\, dx_4$.

4.3.1. Sample Networks. Before developing an algorithm to find the completion-time distribution for a stochastic activity network, the distributions for two specific networks will be derived below, based on the derivations in Section 4.2. After finding the conditional CDF of the completion time for each network, conditioned on the durations of its multiple-use arcs, the CDFs will be integrated over the appropriate range of values for each such arc.

4.3.2. Pritsker Network. The Pritsker network ([11], pp. 216–221) from Figure 1 is given in Figure 14 with the associated new notation. The arcs labeled x_1, x_2, x_3, x_4, x_5, and x_6, which denote the probabilistic activity durations, appear on multiple paths. The paths through this network are listed below, where a generic "1" represents an arc that appears on only one path:

F14

$$x_1 \to 1 \to 1, \quad x_1 \to x_3 \to x_5, \quad x_1 \to x_3 \to x_4 \to x_6,$$
$$x_2 \to x_5, \quad x_2 \to x_4 \to x_6, \quad 1 \to x_6.$$

If t is the completion time for the network, then the following inequalities state the relationships between the arcs being conditioned on and t:

$$x_1 + x_3 + x_5 \le t, \quad x_1 + x_3 + x_4 + x_6 \le t,$$
$$x_2 + x_5 \le t, \quad x_2 + x_4 + x_6 \le t.$$

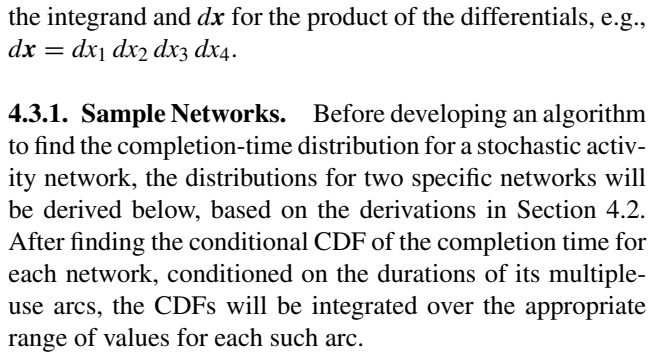

FIG. 14. Pritsker network with multiple-use arcs labeled.

The trivial inequalities, $x_1 \le t$ and $x_6 \le t$, which result from the first and last paths listed above, have been omitted because they are implied by the other inequalities.

The completion-time distribution for the network is found by integrating the product of the conditional CDFs of the single-use arcs and the PDFs of the multiple-use arcs over all possible durations the multiple-use arcs can attain. However, because of conditioning, the limits of integration must be set up carefully to avoid negative limits or cases where the lower limits exceed the upper limits. The algorithm determines appropriate limits as it progresses from the source node to the terminal node. The lower limit of integration for each variable is zero because the minimum possible duration for an arc is zero. The upper limit of integration is the highest possible duration for an arc. Because t is the maximum duration for the entire network, no arc may have a duration larger than t, but if other arcs are on a path that precedes a given arc, the maximum possible value of the later arc is t minus the sum of the durations of all preceding arcs. If multiple paths precede a given arc, the maximum duration of the arc is the difference between t and the largest sum of preceding arc durations. Using this approach, one possible order for the limits of integration is $x_1, x_2, x_3, x_4, x_5, x_6$. By combining this order with the inequalities found above, the following integral results:

$$F_{T_6}(t) = \int_{x_1=0}^{t} \int_{x_2=0}^{t} \int_{x_3=0}^{t-x_1} \int_{x_4=0}^{t-\max\{x_2, x_1+x_3\}} \times \int_{x_5=0}^{t-\max\{x_2, x_1+x_3\}} \int_{x_6=0}^{t-x_4-\max\{x_2, x_1+x_3\}} I(\boldsymbol{x})\, d\boldsymbol{x},$$

where $I(\boldsymbol{x})$ will be determined in the next step.

Because maximum expressions appear in the limits of integration, separate cases must be considered:

- Case 1: $x_1 < x_2$
  - — Subcase 1: $x_2 > x_1 + x_3$. This implies $0 < x_3 < x_2 - x_1$.
  - — Subcase 2: $x_2 < x_1 + x_3$. This implies $x_2 - x_1 < x_3 < t - x_1$.
- Case 2: $x_2 < x_1$. This implies $0 < x_3 < t - x_1$.

The CDF $F_{T_6}(t)$ must be expressed as the sum of several multiple integrals to account for these cases. The common integrand $I(\boldsymbol{x})$ for these integrals is formed by multiplying the product of the conditional CDFs of those paths through the system containing at least one single-use arc times the product of the PDFs of the (multiple-use) arcs being conditioned on. For simplicity it is assumed that the duration of each arc in the system has an exponential(1) distribution.

This network has two paths that contain single-use arcs. One path consists of x_1 followed by two single-use arcs, which can be replaced by one arc whose duration has an Erlang$(1, 2)$ distribution. The other path consists of x_6 and one single-use arc. The conditional CDFs corresponding to these paths are

$$F_{26}(t|x_1) = 1 - (t - x_1)e^{-(t-x_1)} - e^{-(t-x_1)} \qquad t > x_1,$$
$$F_{14}(t|x_6) = 1 - e^{-(t-x_6)} \qquad t > x_6.$$

Because we are assuming that the arcs have exponential(1) activity durations, the PDFs of the arcs on which we are conditioning are simply $e^{-x_1}, e^{-x_2}, e^{-x_3}, e^{-x_4}, e^{-x_5}$, and e^{-x_6}.

Combining, we find

$$F_{T_6}(t) = \int_{x_1=0}^{t}\int_{x_2=0}^{x_1}\int_{x_3=0}^{t-x_1}\int_{x_4=0}^{t-x_1-x_3} \times \int_{x_5=0}^{t-x_1-x_3}\int_{x_6=0}^{t-x_4-x_1-x_3} I(\boldsymbol{x})d\boldsymbol{x} + \int_{x_1=0}^{t}\int_{x_2=x_1}^{t}\int_{x_3=0}^{x_2-x_1}\int_{x_4=0}^{t-x_2}\int_{x_5=0}^{t-x_2}\int_{x_6=0}^{t-x_4-x_2} I(\boldsymbol{x})d\boldsymbol{x} + \int_{x_1=0}^{t}\int_{x_2=x_1}^{t}\int_{x_3=x_2-x_1}^{t-x_1}\int_{x_4=0}^{t-x_1-x_3} \times \int_{x_5=0}^{t-x_1-x_3}\int_{x_6=0}^{t-x_4-x_1-x_3} I(\boldsymbol{x})d\boldsymbol{x},$$

for $t > 0$, where

$$I(\boldsymbol{x}) = (1-(t-x_1)e^{-(t-x_1)} - e^{-(t-x_1)}) \times (1-e^{-(t-x_6)})e^{-x_1-x_2-x_3-x_4-x_5-x_6}.$$

Using Maple (the code is given at `www.math.wm.edu/~leemis/2006networks.prit.code`) to evaluate these integrals, the CDF of the network completion time is

$$F_{T_6}(t) = 1 + \frac{107}{4}e^{-2t} - \frac{71}{4}e^{-4t} - 8e^{-2t}t^2 - \frac{45}{2}e^{-2t}t - \frac{1}{6}e^{-2t}t^3 - \frac{1}{6}e^{-t}t^3 - 2e^{-t}t^2 - 2e^{-4t}t^2 - \frac{71}{2}e^{-3t}t + \frac{1}{8}e^{-2t}t^4 - \frac{1}{8}e^{-3t}t^4 - 9e^{-3t}t^2 + \frac{2}{3}e^{-3t}t^3 - 12e^{-4t}t - \frac{85}{4}e^{-3t} + \frac{45}{4}e^{-t},$$

for $t > 0$. A graph of this CDF is given in Figure 15. The mean network completion time is **F15**

$$E[T_6] = \int_0^\infty (1 - F_{T_6}(t))\,dt = \frac{4213}{864} \approx 4.8762,$$

which has been verified by a simulation using one million replications that yielded 4.8767.

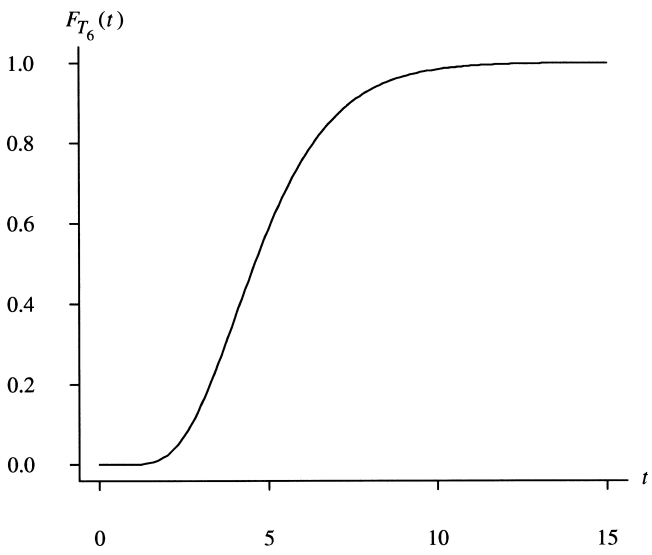

FIG. 15. Graph of $F_{T_6}(t)$ for the Pritsker network with exponential(1) activity durations.

4.3.3. Wheatstone Bridge Network. The network depicted in Figure 16 is the Wheatstone bridge network ([3], p. 114). **F16** The paths through this system are listed below, where a generic "1" represents a single-use arc:

$$x_1 \to 1 \to x_3, \qquad x_1 \to 1 \to x_4,$$
$$x_2 \to 1 \to x_3, \qquad x_2 \to 1 \to x_4.$$

Assuming that t is the network completion time, the following inequalities result:

$$x_1 + x_3 \le t, \quad x_1 + x_4 \le t, \quad x_2 + x_3 \le t, \quad x_2 + x_4 \le t.$$

Working from the source node to the terminal node yields an acceptable order of integration: x_1, x_2, x_3, x_4. With an order of integration established, it is now possible to determine the limits of integration. Both x_1 and x_2 can attain any value from 0 to t. The arcs x_3 and x_4 do not both appear on any path, but each appears after either x_1 or x_2 on every possible path through the network. The maximum duration on x_3 or x_4 is the difference between t and the larger of x_1 and x_2. As a result,

$$F_{T_6}(t) = \int_{x_1=0}^{t}\int_{x_2=0}^{t}\int_{x_3=0}^{t-\max\{x_1,x_2\}}\int_{x_4=0}^{t-\max\{x_1,x_2\}} I(\boldsymbol{x})d\boldsymbol{x},$$

where $I(\boldsymbol{x})$ will be determined shortly.

Considering the two cases $x_2 < x_1$ and $x_1 < x_2$ yields

$$F_{T_6}(t) = \int_{x_1=0}^{t}\int_{x_2=0}^{x_1}\int_{x_3=0}^{t-x_1}\int_{x_4=0}^{t-x_1} I(\boldsymbol{x})d\boldsymbol{x} + \int_{x_1=0}^{t}\int_{x_2=x_1}^{t}\int_{x_3=0}^{t-x_2}\int_{x_4=0}^{t-x_2} I(\boldsymbol{x})d\boldsymbol{x}$$

for $t > 0$. The integrand $I(\boldsymbol{x})$ is determined as in the previous example, again assuming that the duration of each activity is

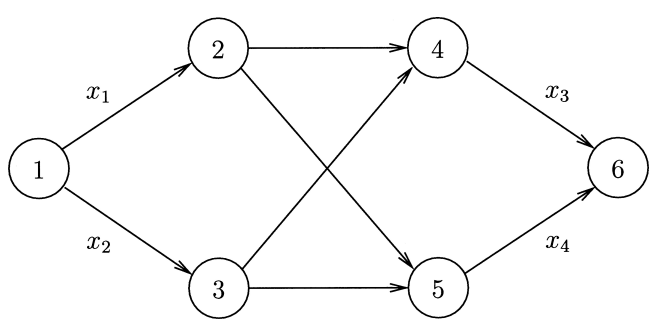

FIG. 16. Wheatstone bridge network with multiple-use arcs labeled.

an exponential(1) random variable. The conditional CDFs of the single-use arcs are given by

$$F_{24}(t|x_1,x_3) = 1 - e^{-(t-x_1-x_3)} \qquad t > x_1 + x_3,$$
$$F_{25}(t|x_1,x_4) = 1 - e^{-(t-x_1-x_4)} \qquad t > x_1 + x_4,$$
$$F_{34}(t|x_2,x_3) = 1 - e^{-(t-x_2-x_3)} \qquad t > x_2 + x_3,$$
$$F_{35}(t|x_2,x_4) = 1 - e^{-(t-x_2-x_4)} \qquad t > x_2 + x_4.$$

Combining the above facts yields

$$F_{T_6}(t) = \int_{x_1=0}^{t}\int_{x_2=0}^{x_1}\int_{x_3=0}^{t-x_1}\int_{x_4=0}^{t-x_1} I(\boldsymbol{x})d\boldsymbol{x} + \int_{x_1=0}^{t}\int_{x_2=x_1}^{t}\int_{x_3=0}^{t-x_2}\int_{x_4=0}^{t-x_2} I(\boldsymbol{x})d\boldsymbol{x}$$

for $t > 0$, where $I(\boldsymbol{x}) = (1-e^{-(t-x_1-x_3)})(1-e^{-(t-x_1-x_4)})(1-e^{-(t-x_2-x_3)})(1-e^{-(t-x_2-x_4)})$. Using Maple to compute these integrals,

$$F_{T_6}(t) = 1 + 20e^{-3t} + \frac{1}{3}e^{-2t}t^4 - 12e^{-2t}t^2 - 42e^{-2t} + 20e^{-t} + 4e^{-3t}t - 2e^{-3t}t^2 + e^{-4t} - 2e^{-t}t^2 - 4e^{-t}t$$

F17 for $t > 0$. This function is plotted in Figure 17. The code to evaluate this integral is given at `www.math.wm.edu/~leemis/2006networks.wheat.code`. The mean time to complete the network is

$$E[T_6] = \int_0^\infty (1 - F_{T_6}(t))\,dt = \frac{245}{54} \approx 4.5370,$$

which has been verified by a simulation using one million replications that yielded 4.5372.

The tedious nature of setting up the integrals for $F_{T_n}(t)$ and eliminating the maximums in their limits for processing by a computer algebra system (CAS) prompted us to develop an algorithm for the automated processing of a network.

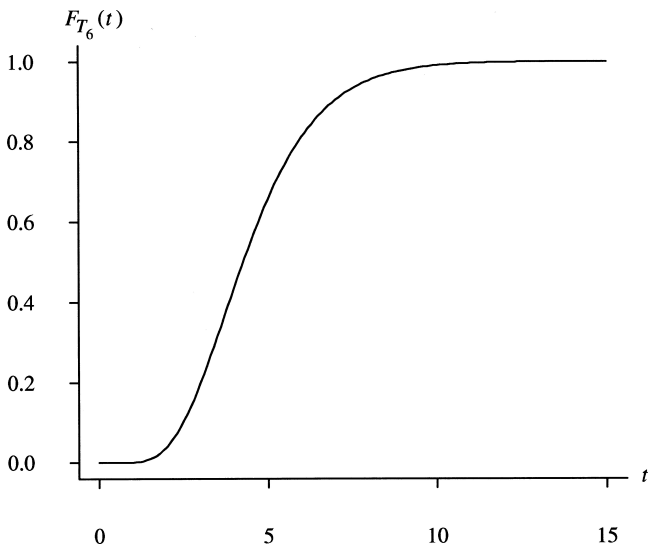

FIG. 17. Graph of $F_{T_6}(t)$ for the Wheatstone bridge network shown in Figure 16 with exponential(1) activity durations.

4.3.4. Algorithm Development. We now develop an algorithm for determining the distribution of the time to complete a stochastic activity network. The 11 steps listed below are incorporated in both the algorithm given at `www.math.wm.edu/~leemis/2006networks.algor.pdf` and the Maple implementation given at `www.math.wm.edu/~leemis/2006networks.code`.

STEP 1. List all paths through the system.

STEP 2. List the arcs that appear on only one path. If successive single-use arcs appear on one path, combine them into one arc whose PDF is the convolution of the original PDFs (i.e., a series reduction).

STEP 3. Denote by k the number of multiple-use arcs.

STEP 4. Label the arcs counted in Step 3 with variable names, as follows:

(a) First label each multiple-use arc that is the initial arc on some path, using x_1, x_2, x_3, etc., until all such initial arcs are labeled. The initial arcs may be labeled in any order, using the label x_1 first and then using consecutively subscripted labels.
(b) After eliminating all arcs previously labeled and each single-use arc that is now the first arc on some path, next label a multiple-use arc that is the first such arc appearing on one of the remaining paths. If the arc appears on more than one of the remaining paths, it must be the initial arc on each such remaining path to be chosen. If there are several arcs that meet this criterion simultaneously, any one may be chosen as the next arc to be labeled. This step is repeated until all k multiple-use arcs are labeled.

STEP 5. Choose x_1 as the outermost variable of integration, followed by x_2, then x_3 through x_k.

STEP 6. Set the limits of integration. All lower limits of integration will be zero. To set the upper limits of integration, use the following rules:

(a) For all variables of integration corresponding to initial arcs found in Step 4(a), the upper limit of integration is t.
(b) For all subsequent arcs, find all paths containing them. For each such arc, list the sequence(s) of multiple-use arcs that precede that arc. The upper limit of integration for the variable corresponding to that arc is then the difference of t and the maximum of the sums of the variables corresponding to each sequence of arcs in the list of preceding arcs.

STEP 7. To use a CAS to evaluate the integral, it is necessary to eliminate the maximums from the upper limits by considering many cases. But first, if the same arc appears in every summation within a particular maximum expression, bring that variable outside of the expression to simplify it.

STEP 8. Now the maximums are eliminated using the following rules:

(a) Starting with the left-most integral symbol and moving to the right, find the first limit of integration where a maximum occurs.
(b) Examine all variables in the maximum expression and select the variable with the highest subscript.
(c) Split the range of integration for the variable selected in Step 8(b) into two subintervals in such a way that on each subinterval the value of the maximum is evident. (Note: The breakpoint between the two subintervals may itself involve a maximum expression.) In this way split the original k-fold integral into the sum of two k-fold integrals.

STEP 9. For each k-fold integral produced by Step 8 that has a maximum in one or more of its limits of integration, repeat Step 8.

STEP 10. The common integrand for all of the k-fold integrals generated above is found by taking the product of PDFs and conditional CDFs:

(a) For each path through the system containing at least one single-use arc, find its CDF when conditioned on the duration of the arcs that appear on multiple paths. The product of all such CDFs is the first part of the integrand.
(b) Find the PDF for each arc that is conditioned on, that is, all arcs appearing on multiple paths. The product of all such PDFs is the second part of the integrand.
(c) Multiply the product from part (a) and the product from part (b) together to form the final integrand.

STEP 11. Evaluate all the k-fold integrals that have been generated and sum the resulting expressions to yield the CDF of the time to complete the network.

4.3.5. Applying the Algorithm. The algorithm developed in the previous section will now be applied to the eight-path network ([12], p. 53) shown in Figure 18, with its multiple-use arcs labeled. It is assumed that the distribution of the time to traverse each arc is an exponential(1) random variable. The 11 steps are listed below by number. **F18**

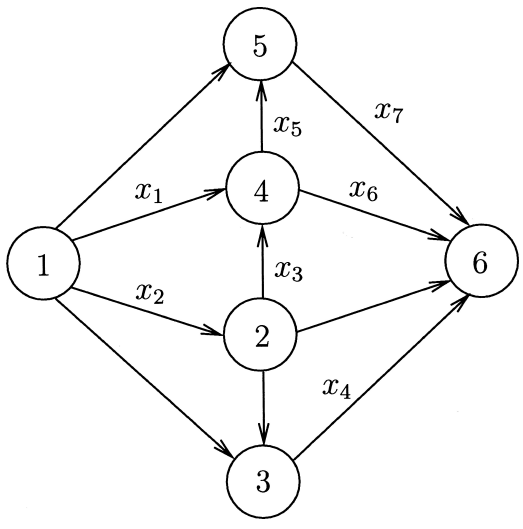

FIG. 18. The 8-path network with multiple-use arcs labeled.

STEP 1. The paths through the system are given in Table 9. **T9**

STEP 2. The arcs that are used on only one path are $a_{13}, a_{15}, a_{23}, a_{26}$.

STEP 3. Because there are seven multiple-use arcs, set $k = 7$.

STEP 4. Label the arcs.

(a) The initial arc in the paths through the system are $a_{12}, a_{13}, a_{14}, a_{15}$. However, only a_{12} and a_{14} are multiple-use arcs, so they are the only arcs labeled. Let $x_1 = a_{14}$ and $x_2 = a_{12}$.
(b) The arcs a_{24} and a_{36} are the next multiple-use arcs that appear first on the remaining paths after removing a_{12}, a_{14}, and the resulting initial single-use arcs a_{13}, a_{15}, and a_{36}. Let $x_3 = a_{24}$ and $x_4 = a_{36}$. After eliminating the four arcs already labeled and the three single-use arcs, the arcs that are now at the beginning of the paths through the network are a_{45} and a_{46}. Let $x_5 = a_{45}$ and $x_6 = a_{46}$. The only remaining multiple-use arc left in the network is a_{56}, so $x_7 = a_{56}$. Figure 18 shows the network with the appropriate labels added.

STEP 5. The order of integration is now set by arranging the variables to yield the integral

$$\int\int\int\int\int\int\int I(\boldsymbol{x})\, dx_7\, dx_6\, dx_5\, dx_4\, dx_3\, dx_2\, dx_1.$$

The integrand $I(\boldsymbol{x})$ will be determined later.

STEP 6. Set the limits of integration. All lower limits of integration are initially set to 0.

(a) The upper limits of integration for the arcs x_1 and x_2 found in Step 4(a) are set to t.
(b) To find the upper limit of integration for all other variables, the paths through the system must be examined. Table 10 shows the paths through the network with regard to multiple-use arcs. This table represents the same paths as Table 9 except expressed by the names of the multiple-use arcs. For convenience, these arc names are also used **T10**

TABLE 9. Paths for the network shown in Figure 18.

Node sequence	Path
$1 \to 2 \to 6$	$\{a_{12}, a_{26}\}$
$1 \to 2 \to 3 \to 6$	$\{a_{12}, a_{23}, a_{36}\}$
$1 \to 2 \to 4 \to 6$	$\{a_{12}, a_{24}, a_{46}\}$
$1 \to 2 \to 4 \to 5 \to 6$	$\{a_{12}, a_{24}, a_{45}, a_{56}\}$
$1 \to 3 \to 6$	$\{a_{13}, a_{36}\}$
$1 \to 4 \to 6$	$\{a_{14}, a_{46}\}$
$1 \to 4 \to 5 \to 6$	$\{a_{14}, a_{45}, a_{56}\}$
$1 \to 5 \to 6$	$\{a_{15}, a_{56}\}$

TABLE 10. Paths through the network shown in Figure 18.

Multiple-use arc sequences
$x_2 \rightarrow 1$
$x_2 \rightarrow 1 \rightarrow x_4$
$x_2 \rightarrow x_3 \rightarrow x_6$
$x_2 \rightarrow x_3 \rightarrow x_5 \rightarrow x_7$
$1 \rightarrow x_4$
$x_1 \rightarrow x_6$
$x_1 \rightarrow x_5 \rightarrow x_7$
$1 \rightarrow x_7$

for the duration associated with each arc. Recall that the single-use arcs are designated (or labeled) as "1." In all paths on which it appears, x_3 is always preceded by x_2, so the upper limit for x_3 is $t - x_2$. When x_4 is preceded by another multiple-use arc, it is also always x_2, so like x_3, the upper limit of integration for x_4 is $t - x_2$. Arc x_5 is preceded by x_1 on one path and by x_2 and x_3 on another path. Because x_2 and x_3 appear before x_5 on the same path, the upper limit of integration for x_5 is $t - \max\{x_1, x_2 + x_3\}$. The variable x_6 is similar to x_5 because it is preceded by x_1 or by x_2 and x_3. Therefore, its upper limit of integration is also $t - \max\{x_1, x_2 + x_3\}$. The final arc, x_7, is preceded either by the sequence of x_2, x_3, x_5 or by the sequence x_1, x_5. The upper limit of integration is formed in the same way as those of x_5 and x_6, so the upper limit of integration for x_7 is $t - \max\{x_1 + x_5, x_2 + x_3 + x_5\}$. The integral that represents the CDF of the network completion time, $F_{T_6}(t)$, can now be written as

$$F_{T_6}(t) = \int_{x_1=0}^{t} \int_{x_2=0}^{t} \int_{x_3=0}^{t-x_2} \int_{x_4=0}^{t-x_2} \int_{x_5=0}^{t-\max\{x_1,x_2+x_3\}} \times \int_{x_6=0}^{t-\max\{x_1,x_2+x_3\}} \int_{x_7=0}^{t-\max\{x_1+x_5,x_2+x_3+x_5\}} I(\boldsymbol{x})d\boldsymbol{x}$$

for $t > 0$.

STEP 7. Because the upper limit of integration for x_7 contains x_5 in both arguments of the maximum, the upper limit of integration can be simplified, giving the following integral for $F_{T_6}(t)$:

$$F_{T_6}(t) = \int_{x_1=0}^{t} \int_{x_2=0}^{t} \int_{x_3=0}^{t-x_2} \int_{x_4=0}^{t-x_2} \int_{x_5=0}^{t-\max\{x_1,x_2+x_3\}} \times \int_{x_6=0}^{t-\max\{x_1,x_2+x_3\}} \int_{x_7=0}^{t-x_5-\max\{x_1,x_2+x_3\}} I(\boldsymbol{x})d\boldsymbol{x}$$

for $t > 0$.

STEP 8. Eliminating the maximums.

(a) Beginning on the left with x_1, the first variable of integration encountered with a maximum in its limits of integration is x_5.
(b) The expression that appears in this limit of integration is $\max\{x_1, x_2 + x_3\}$. Of the variables in this expression, the one with the highest subscript is x_3, which can range in value from 0 to $t - x_2$.
(c) There are two cases to consider because of this maximum:

$$x_1 < x_2 + x_3 \qquad \text{and} \qquad x_2 + x_3 < x_1.$$

Because the range of x_3 is being split, it is necessary to isolate it in the inequalities above yielding

$$x_1 - x_2 < x_3 \qquad \text{and} \qquad x_3 < x_1 - x_2.$$

From these inequalities, it is easy to see that the point where the range of x_3 is divided is $x_1 - x_2$. However, if $x_1 - x_2$ is negative, then there is no split in the range of x_3, because x_3 can only assume nonnegative values. This possibility is addressed by rewriting the inequalities as:

$$\max\{0, x_1 - x_2\} < x_3 < t - x_2 \qquad \text{and} \qquad 0 < x_3 < \max\{0, x_1 - x_2\},$$

which takes into account the fact that the range of the variable x_3 is from 0 to $t - x_2$.

These two inequalities are used to set up two new sevenfold integrals, the sum of which is equal to the original sevenfold integral. It is important to note that the maximum that was eliminated for variable x_5 is the same one that appeared in the limits of x_6 and x_7, and thus these are also eliminated. The new integrals are

$$F_{T_6}(t) = \int_{x_1=0}^{t} \int_{x_2=0}^{t} \int_{x_3=0}^{\max\{0,x_1-x_2\}} \times \int_{x_4=0}^{t-x_2} \int_{x_5=0}^{t-x_1} \int_{x_6=0}^{t-x_1} \int_{x_7=0}^{t-x_5-x_1} I(\boldsymbol{x})d\boldsymbol{x} + \int_{x_1=0}^{t} \int_{x_2=0}^{t} \int_{x_3=\max\{0,x_1-x_2\}}^{t-x_2} \int_{x_4=0}^{t-x_2} \int_{x_5=0}^{t-x_2-x_3} \times \int_{x_6=0}^{t-x_2-x_3} \int_{x_7=0}^{t-x_5-x_2-x_3} I(\boldsymbol{x})d\boldsymbol{x}$$

for $t > 0$.

STEP 9. Repeat Step 8 until all maximums are eliminated.

(a) In the first seven-fold integral, a maximum occurs in the upper limit of x_3.
(b) In $\max\{0, x_1 - x_2\}$, x_2 is the variable with the highest subscript.
(c) There are two cases to consider:

$$0 < x_1 - x_2 \qquad \text{and} \qquad x_1 - x_2 < 0.$$

By isolating the variable x_2 in these inequalities and incorporating the range of x_2, they can be rewritten as:

$$0 < x_2 < x_1 \qquad \text{and} \qquad x_1 < x_2 < t.$$

It is clear that the point where the range of x_2 is to be split is x_1. This splits the first integral from Step 8 into two more seven-fold integrals; however, when $x_1 < x_2$, $\max\{0, x_1 - x_2\} = 0$, so both limits of integration for x_3 are 0, and thus the second seven-fold integral evaluates to 0. This means that only one integral is created, and

TABLE 11. Conditional CDFs of single-use arcs for the network in Figure 18 with exponential(1) arc durations.

Multiple-use arc sequence	Conditional CDF	Support
$x_2 \to 1$	$1 - e^{-(t-x_2)}$	$t > x_2$
$x_2 \to 1 \to x_4$	$1 - e^{-(t-x_2-x_4)}$	$t > x_2 + x_4$
$1 \to x_4$	$1 - e^{-(t-x_4)}$	$t > x_4$
$1 \to x_7$	$1 - e^{-(t-x_7)}$	$t > x_7$

thus we still have two seven-fold integrals:

$$F_{T_6}(t) = \int_{x_1=0}^{t}\int_{x_2=0}^{x_1}\int_{x_3=0}^{x_1-x_2}\int_{x_4=0}^{t-x_2}\int_{x_5=0}^{t-x_1} \times \int_{x_6=0}^{t-x_1}\int_{x_7=0}^{t-x_5-x_1} I(\boldsymbol{x})d\boldsymbol{x} + \int_{x_1=0}^{t}\int_{x_2=0}^{t}\int_{x_3=\max\{0,x_1-x_2\}}^{t-x_2}\int_{x_4=0}^{t-x_2}\int_{x_5=0}^{t-x_2-x_3} \times \int_{x_6=0}^{t-x_2-x_3}\int_{x_7=0}^{t-x_5-x_2-x_3} I(\boldsymbol{x})d\boldsymbol{x}$$

for $t > 0$.

(a) The second sevenfold integral still has a maximum in the lower limit of x_3.
(b) As in the previous example, the variable with the highest subscript is x_2.
(c) As before, there are two cases to consider:

$$0 < x_1 - x_2 \qquad \text{and} \qquad x_1 - x_2 < 0,$$

which lead to the following intervals for x_2:

$$0 < x_2 < x_1 \qquad \text{and} \qquad x_1 < x_2 < t.$$

Unlike the previous step, two new integrals are produced, and neither is equal to 0. The second seven-fold integral immediately below corresponds the case where $0 < x_1 - x_2$, and the third corresponds to the case where $x_1 - x_2 < 0$. The final expression is

$$F_{T_6}(t) = \int_{x_1=0}^{t}\int_{x_2=0}^{x_1}\int_{x_3=0}^{x_1-x_2}\int_{x_4=0}^{t-x_2}\int_{x_5=0}^{t-x_1} \times \int_{x_6=0}^{t-x_1}\int_{x_7=0}^{t-x_5-x_1} I(\boldsymbol{x})d\boldsymbol{x} + \int_{x_1=0}^{t}\int_{x_2=0}^{x_1}\int_{x_3=0}^{t-x_2}\int_{x_4=0}^{t-x_2}\int_{x_5=0}^{t-x_2-x_3} \times \int_{x_6=0}^{t-x_2-x_3}\int_{x_7=0}^{t-x_5-x_2-x_3} I(\boldsymbol{x})d\boldsymbol{x} + \int_{x_1=0}^{t}\int_{x_2=x_1}^{t}\int_{x_3=0}^{t-x_2}\int_{x_4=0}^{t-x_2}\int_{x_5=0}^{t-x_2-x_3} \times \int_{x_6=0}^{t-x_2-x_3}\int_{x_7=0}^{t-x_5-x_2-x_3} I(\boldsymbol{x})d\boldsymbol{x}$$

for $t > 0$.

STEP 10. Constructing the integrand.

(a) Four paths contain single-use arcs. The paths are listed in Table 11 with their conditional CDFs. As stated before, all arcs are assumed to have exponential(1) durations. **T11**
(b) There are seven multiple-use arcs. Because each arc duration is assumed to be an exponential(1) random variable, their PDFs are e^{-x_k} for $x_k > 0$, where $k = 1, 2, \ldots, 7$.
(c) Taking the product of the conditional CDFs from part (a) and the PDFs from part (b) gives

$$I(\boldsymbol{x}) = (1 - e^{-(t-x_2)})(1 - e^{-(t-x_2-x_4)})(1 - e^{-(t-x_4)}) \times (1 - e^{-(t-x_7)}) \cdot e^{-x_1-x_2-x_3-x_4-x_5-x_6-x_7}.$$

STEP 11. The integrals found by the preceding algorithm can be evaluated using a CAS such as Maple, yielding the following CDF:

$$\begin{aligned} F_{T_6}(t) = 1 &+ \frac{11}{4}e^{-3t}t^3 - \frac{1}{8}e^{-3t}t^4 + e^{-6t}t + e^{-2t}t^3 - \frac{147}{4}e^{-2t}t - 2e^{-t}t + \frac{63}{8}e^{-3t}t^2 \\ &- \frac{553}{48}e^{-3t} + \frac{443}{24}e^{-4t}t^2 - 10e^{-2t}t^2 + \frac{103}{6}e^{-5t}t - \frac{1}{6}e^{-t}t^3 - \frac{639}{8}e^{-3t}t \\ &+ \frac{11}{4}e^{-5t}t^2 - \frac{3}{4}e^{-4t}t^3 - \frac{677}{72}e^{-4t}t + \frac{1}{8}e^{-4t}t^4 + \frac{43}{12}e^{-6t} + \frac{53}{36}e^{-5t} \\ &- \frac{21217}{432}e^{-4t} + \frac{5431}{144}e^{-2t} + \frac{7285}{432}e^{-t} + \frac{1}{4}e^{-2t}t^4 - 2e^{-t}t^2 \end{aligned}$$

for $t > 0$. The graph of the function $F_{T_6}(t)$ is shown in **F19** Figure 19. The code to evaluate the above three integrals is given at `www.math.wm.edu/~leemis/2006networks.8path.code`. The mean time to complete the network is

$$E[T_6] = \int_0^\infty (1 - F_{T_6}(t))\,dt = \frac{51822023}{10368000} \approx 4.9983,$$

which has been verified by a simulation using one million replications that yielded 4.9985.

4.3.6. Implementation. A Maple procedure is used for the computer implementation of this algorithm because of the mathematical functions and data structures present within the Maple CAS. Pseudocode for the program is given

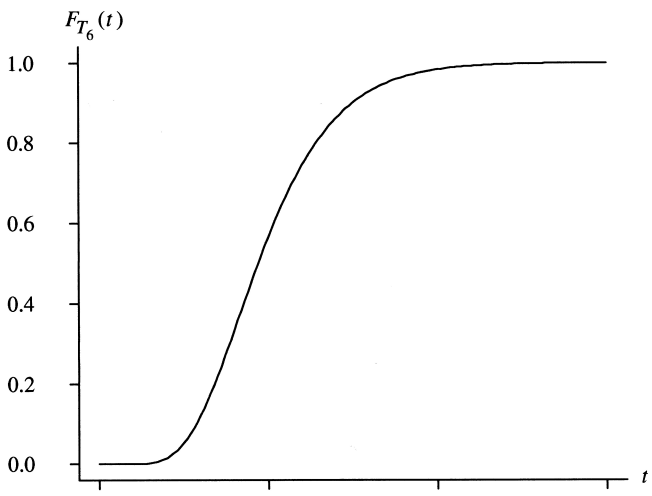

FIG. 19. Graph of $F_{T_6}(t)$ for the 8-path network in Figure 18 with exponential(1) arc durations.

at `www.math.wm.edu/~leemis/2006networks.algor.pdf` for readers wishing to implement the program in another language.

The primary data structures used for the implementation of this algorithm are matrices, arrays, and sets. Matrices are used to store the paths through the network, the upper and lower limits of integration, and the node-arc incidence matrix. Arrays are used to distinguish multiple-use arcs from single-use arcs, to store the order of integration and the durations of each activity. Sets are mainly used for data manipulation, such as combining different groups of data, or finding the overlap between two groups of numbers. In Maple, sets do not allow for repeated elements and can be manipulated using the set operations of intersection, union, and minus.

4.3.7. Preprocessing data. The implementation first reads in the network representation, that is, the node-arc incidence matrix N and the activity duration distributions. These can be stored in text files outside of the program and read into Maple using the `read` function. Using Maple, the number of nodes and arcs are determined by invoking the `rowdim(N)` and `coldim(N)` functions, respectively. It would also be possible to read in the values of n and m directly.

Once the data is read in, there is some preprocessing to be done. For the algorithm to process the matrix N correctly, the columns are sorted in such a way that they are ordered by the node from which their corresponding arcs exit. Then all arcs that exit the same node are ordered by the node they enter. In Maple, this ordering is done using two nested-for-loops and the `swapcol(N, col1, col2)` function that interchanges column `col1` and column `col2`. To find the paths through the system, it is necessary to determine how many arcs enter each node and the source node for each arc. This data is stored in two one-dimensional arrays. One array will have an entry for each node that stores the number of incoming arcs. The other array stores the source node for each arc. If the language being used supports columns and rows of index 0, then the data for incoming arcs can be stored in the 0*th* column and source node data can be stored in the 0*th* row.

STEP 1. The program finds all of the paths through the network. An upper bound on the number of paths is estimated, say at most `MaxPaths`. A `MaxPaths` $\times\, n$ matrix is set up to store the paths. Each row of this matrix will store one path through the network, with the row's ith entry being 1 if arc x_i lies on the path and 0 otherwise. The function used to find paths is a function called `GetPaths`. This function also counts the number of paths through the system. If the number of paths is less than `MaxPaths`, excess rows are deleted by the Maple function `delrows(Paths, numPaths + 1 .. MaxPaths)`. It is necessary to reverse the order of the nodes on each path because the function `GetPaths` reads in the paths by starting at the terminal node and working backward.

STEPS 2 and 3. The algorithm distinguishes multiple-use arcs from single-use arcs and counts the number of multiple-use arcs. This is done by following each path and incrementing a counter for each arc whenever that arc is encountered. At the end of the process, any counter exceeding 1 identifies a multiple-use arc.

STEPS 4 and 5. The order of integration is determined. Integration only occurs over multiple-use arcs, so they are the only ones the program examines. The first arc on each path is determined and placed into a set S_1; then all subsequent arcs are placed in a second set S_2. The arcs that are eligible to be first in the new array for the order of integration appear in the first set and not the second. All such arcs are added to the array. This procedure is repeated as the program traverses each path, ignoring the single-use arcs. These operations are performed by utilizing Maple's `intersect` and `minus` operations for sets.

STEPS 6 and 7. At this point, the most complex portion of the procedure begins, setting up the limits of integration. The upper and lower limits of integration are stored in separate matrices. The matrices for the upper limits, $upper_i$, have dimension $4 \times k$ and the matrices for lower limits, $lower_i$, have dimension $3 \times k$. Initially $i = 1$, because there is currently only one k-fold integral. Every entry in the $lower_1$ matrix is initialized to 0 as are all entries in $upper_1$, with the exception of the first row of $upper_1$ whose entries are initialized to t. The program then traverses the integration order matrix and adjusts the upper limit matrix. The jth column of $upper_1$ is filled in as follows, for $j = 1, 2, \ldots, k$: there are at most two paths leading to arc x_j. If there is exactly one path leading to arc x_j, then the variables for all arcs on that path preceding x_j are summed and placed in row 2 of the jth column of $upper_1$. If there are two such paths, the variables for one of the paths are summed and placed in row 3 of the jth column of $upper_1$ and the variables for the other path

are summed and placed in row 4. These entries in rows 3 and 4 correspond to the arguments in the maximum of the upper limit of integration for x_j. Any variables appearing in both paths are deleted from both row 3 and row 4, summed, and placed in row 2 of the jth column. Once the third and fourth rows are zeroed out, which is the terminal condition for Step 9 of the algorithm, the upper limits of integration are the differences between the elements in the first row and the corresponding elements in the second row. The initial matrix $upper_1$, corresponding to the eight-path network analyzed in Section 4.3.5, is given in Figure 20. **F20**

STEPS 8 and 9. After $upper_1$ and $lower_1$ have been initialized, the program enters a while-loop in which maximums are eliminated from all limits of integration according to the procedure outlined in the algorithm. The program first scans the upper limit matrices to see if there are any nonzero entries in the third or fourth rows. If there are, the current matrix is copied along with the corresponding lower limit matrix. These two matrices will correspond to the limit of integration being broken up; one will have the third row entry as the new upper limit and the other will have the fourth row entry. To find the point at which the range splits for the x_i being examined, the two arguments in the maximum expression must be set equal to each other. If the point splitting the range is a quantity that contains more than one variable, for example, $x_2 - x_1$, then when the limits of integration are changed to reflect this, the quantity must go into third row of one upper limit matrix and the second row of the other lower limit matrix; this is because the quantity results in another maximum that must be eliminated, because $x_2 - x_1$ can be negative, and that is not allowed. If only one variable is in the splitting point, then that variable goes into the first row of both matrices. It is possible to check the number of variables in a quantity by using the `nops(quantity)` function in Maple, which returns the number of operations in a set.

After the function makes a pass through the $upper_i$ matrices, it then passes through the $lower_i$ matrices because sometimes when a maximum in an upper limit of integration is eliminated, a new maximum is introduced in another upper limit of integration and in a lower limit of integration. Thus, the program must repeat the maximum elimination process on the lower limit matrices. Upon completing that, the program checks to see if all maximums have been eliminated, that is, the bottom two rows of all matrices have only zero entries. If all maximums are eliminated, the loop is broken; otherwise, the program starts the loop over by examining the upper limits of integration again.

STEP 10. The program constructs the integrand that will be used in all of the k-fold integrals. This entails finding each single-use arc and integrating its PDF from 0 up to the maximum possible arc duration, which is t minus the sum of all other arc variables on that path. This is the conditional CDF of the arc given the duration of all other arcs on the path. The integrand is then the product of the PDFs of all multiple-use arcs and the conditional CDFs of all the single-use arcs that were just found.

STEP 11. The final step in finding the completion-time distribution is to evaluate all of the integrals. This consists of an outer loop that processes each k-fold integral and an inner loop that evaluates each of the k integrations. The integrations are performed using Maple's `int(integrand, xi = lowerLimit .. upperLimit)` command. At the conclusion of these last two loops, the CDF of the time needed to complete the network is returned.

4.3.8. Examples. We will use our program to find the completion-time distributions for the four networks presented earlier.

4.3.9. Bridge-Plus Network. The bridge-plus network, shown in Figure 21 with its multiple-use arcs appropriately labeled, was originally addressed in Section 4.2.1 The activity durations are independent exponential random variables with PDFs **F21**

$$f_{12}(x) = f_{24}(x) = f_{34}(x) = \frac{1}{5}\, e^{-x/5} \qquad x > 0$$

and

$$f_{13}(x) = f_{14}(x) = f_{23}(x) = \frac{1}{10}\, e^{-x/10} \qquad x > 0.$$

	1	2	3	4	5	6	7
1	t	t	t	t	t	t	t
2	0	0	x_2	x_2	0	0	x_5
3	0	0	0	0	x_1	x_1	x_1
4	0	0	0	0	$x_2 + x_3$	$x_2 + x_3$	$x_2 + x_3$

FIG. 20. The initial matrix $upper_1$ for the network discussed in Section 4.3.5.

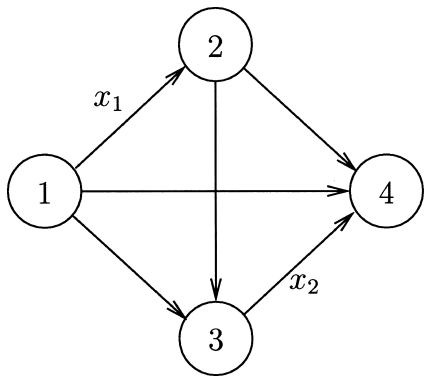

FIG. 21. Bridge-plus network with multiple-use arcs labeled.

With this information, the array of PDFs can be constructed:

$$dist = \left[\frac{1}{5}e^{-x/5}, \frac{1}{10}e^{-x/10}, \frac{1}{10}e^{-x/10}, \frac{1}{10}e^{-x/10}, \frac{1}{5}e^{-x/5}, \frac{1}{5}e^{-x/5}\right].$$

Using the array and matrix as input to the program, the cumulative distribution function returned by the program is

$$F_{T_4}(t) = 1 - 7e^{-t/10} + 12e^{-t/5} + \frac{2t}{5}e^{-t/5} - 16e^{-3t/10} + 19e^{-2t/5} - 9e^{-t/2} - \frac{2t}{5}e^{-t/2}$$

for $t > 0$. This is the same result as in Section 4.2.1.

4.3.10. Pritsker Network. The Pritsker network is shown in Figure 14 and discussed in Section 4.3.1. The original analysis of this network assumed that each activity had an exponential(1) duration; now we examine the more general case of exponential(λ) activity durations. As discussed earlier, because two single-use arcs appear on the same path, namely a_{25} and a_{56}, they are replaced by the single arc a_{26} whose activity duration PDF is Erlang(λ, 2) resulting from the convolution of the PDFs $f_{25}(x)$ and $f_{56}(x)$. By using the node-arc incidence matrix for the revised Pritsker network and the associated array of PDFs,

$$dist = \left[\lambda e^{-\lambda x}, \lambda e^{-\lambda x}, \lambda e^{-\lambda x}, \lambda e^{-\lambda x}, \lambda^2 x e^{-\lambda x}, \lambda e^{-\lambda x}, \lambda e^{-\lambda x}, \lambda e^{-\lambda x}\right],$$

the program returns the cumulative distribution function:

$$\begin{aligned} F_{T_6}(t) = 1 &+ \frac{107}{4}e^{-2\lambda t} - \frac{71}{4}e^{-4\lambda t} - 8e^{-2\lambda t}\lambda^2 t^2 \\ &- \frac{45}{2}e^{-2\lambda t}\lambda t - \frac{1}{6}e^{-2\lambda t}\lambda^3 t^3 - \frac{1}{6}e^{-\lambda t}\lambda^3 t^3 \\ &- 2e^{-\lambda t}\lambda^2 t^2 - 2e^{-4\lambda t}\lambda^2 t^2 - \frac{71}{2}e^{-3\lambda t}\lambda t \\ &+ \frac{1}{8}e^{-2\lambda t}\lambda^4 t^4 - \frac{1}{8}e^{-3\lambda t}\lambda^4 t^4 \\ &- 9e^{-3\lambda t}\lambda^2 t^2 + \frac{2}{3}e^{-3\lambda t}\lambda^3 t^3 - 12e^{-4\lambda t}\lambda t \\ &- \frac{85}{4}e^{-3\lambda t} + \frac{45}{4}e^{-\lambda t} \qquad t > 0. \end{aligned}$$

If λ is set to 1, this CDF equals the CDF found in Section 4.3.2, as it should.

4.3.11. Wheatstone Bridge Network. Now we consider the Wheatstone bridge network analyzed in Section 4.3.3 and shown in Figure 16, with activity durations that are exponential(1) random variables. By using the node-arc incidence matrix and the PDF array

$$dist = \left[e^{-x}, e^{-x}, e^{-x}, e^{-x}, e^{-x}, e^{-x}, e^{-x}, e^{-x}\right],$$

the program confirms that the expression for $F_{T_6}(t)$ found in Section 4.3.3 is correct.

4.3.12. The 8-Path Network. The final network we examine is the eight-path network shown in Figure 18. If we assume that the arcs exiting the source node have exponential(0.5) durations, that arcs entering the terminal node have exponential(0.25) durations and that all remaining arcs have exponential(1) durations, then we can use the node-arc incidence matrix and PDF array to find the distribution of time needed to complete the network. Our program returns the following completion-time CDF for this network:

$$\begin{aligned} F_{T_6}(t) = 1 &+ \frac{11884}{11025}e^{-11t/4} - \frac{4462}{315}e^{-5t/4} + \frac{2}{9}e^{-t}t^2 - \frac{152}{105}e^{-3t/2}t - \frac{355772}{23625}e^{-3t/4} + \frac{1756}{945}e^{-2t}t \\ &- \frac{9547}{23625}e^{-3t} + \frac{2}{45}e^{-5t/2}t + \frac{2}{175}e^{-13t/4}t - \frac{1}{75}e^{-3t}t - \frac{4}{105}e^{-11t/4}t + \frac{31646}{3675}e^{-9t/4} \\ &- \frac{36847}{14175}e^{-5t/2} + \frac{2432567}{99225}e^{-t} - \frac{36}{1225}e^{-15t/4} + \frac{257}{1575}e^{-7t/2} - \frac{52}{15}e^{-3t/4}t \\ &- \frac{863108}{70875}e^{-7t/4} + \frac{1093}{135}e^{-t}t - \frac{176}{675}e^{-7t/4}t - \frac{4}{21}e^{-9t/4}t - \frac{38}{135}e^{-5t/4}t + \frac{170}{27}e^{-t/2}t \\ &- \frac{778979}{165375}e^{-2t} - \frac{6842}{525}e^{-t/4} - \frac{1046}{6125}e^{-13t/4} + \frac{25797}{1225}e^{-3t/2} + \frac{979331}{165375}e^{-t/2} \qquad t > 0. \end{aligned}$$

5. CONCLUSIONS AND FURTHER WORK

Activity networks are useful when modeling a project having distinct activities that exhibit precedence relationships. When the durations of the activities are deterministic, it is easy to find the time needed to complete a project by using a simple longest path algorithm. However, when only random activity durations are given, the analysis is much more complicated. Monte Carlo simulation can provide an approximate solution in this case, but the simulation must be written specifically for the network and could require millions of replications for a reliable estimate. In some networks, series-parallel reduction can supply an exact solution analytically, but very few networks can be reduced in this manner.

In this article a more general approach to finding the distribution of the time needed to complete a stochastic activity network is described. The process is based on conditioning upon the durations of the multiple-use arcs to set up a function that can be integrated to solve for the CDF of the entire network. Our algorithm sets up appropriate k-fold integrals that must be evaluated, where k is the number of arcs appearing on multiple paths. A computer implementation completely automates the calculations to find the completion-time distribution; the only user input is the node-arc incidence matrix N and the PDFs of the individual activity durations.

Our algorithm is easy to use, but the restrictions placed on the network are rather severe. However, our algorithm provides a foundation for the development of a more general algorithm, allowing analysis of more realistic activity networks in the future.

The first restriction is that there can be at most two incident arcs to any nonterminal node. This limits the number of arguments in the maximum expressions for upper limits of integration to two, because each path through a node contributes an argument to a maximum expression. If we let a and b be the arguments in some maximum expression, that is, $\max\{a, b\}$, the only cases that must be considered are $a > b$ and $a < b$. Thus, each k-fold integral is split into at most two k-fold integrals when each maximum is eliminated, representing the cases $a > b$ and $a < b$. Allowing three incident arcs at a nonterminal node would yield three arguments in a maximum expression, with six possible orderings. Then splitting the original k-fold integral could result in as many as six k-fold integrals when a maximum is eliminated.

Dropping the two incoming arcs requirement would also create problems in the implementation. Currently each upper limit of integration is represented by at most four terms in a matrix. Allowing more arguments in the maximum expressions would require more rows in each matrix, but it would be impossible to know how many rows would be needed until the matrix was being constructed. Creation of the new limit-of-integration matrices would also be more difficult. The program currently functions by creating one new matrix for each maximum encountered. Under the more general case with $a > 2$ incoming arcs, there would be $a - 1$ new matrices created for each maximum encountered on the first iteration. Enumerating and editing these matrices would require added computation.

Another restriction placed on the networks is that the activity durations must have support on $(0, \infty)$. It is appropriate that activities have a positive duration, but not all will have an unbounded possible duration time. By requiring this restriction on the activity durations, many possible distributions for the duration are excluded, such as the uniform distribution, which has support on a finite interval. Relaxing this restriction would complicate the limits of integration for variables with finite support, resulting in more k-fold integrals.

The final restriction placed on the network is that the random activity durations have distributions that are a single function as opposed to a piecewise function, such as the triangular distribution. Removing this restriction causes the same problem as having finite support. When the k-fold integrals are set up, there would automatically be at least two integrals for each piecewise distribution, one for each distinct interval defined by the distribution function. Once these k-fold integrals are set up, the situation addressed above is encountered—there are variables that have lower limits of integration larger than 0 and upper limits of integration less than infinity. Therefore, the restriction to infinite support would have to be addressed before any work to relax this final restriction could proceed.

Another improvement would enable our algorithm to process networks having arcs in series. These arcs could be combined into a single arc whose duration distribution is the convolution of the constituent duration distributions. This was the case in the Pritsker network presented earlier, where two exponential(1) durations were combined into a single Erlang(1, 2) variable. In the future, the implementation of the algorithm could be revised to automatically find arcs in series and perform the convolution internally, perhaps using the Maple-based APPL language [5].

Acknowledgments

The first and third authors thank the College of William & Mary for providing a Faculty Research Assignment that was needed to complete this research. They also thank Professor Sid Lawrence for his help with the analytic approach in the nonseries-parallel case, and express their gratitude to the referees, Associate Editor, and Editor for their unusually thorough and helpful comments that improved the presentation in this article.

APPENDIX A. BRIDGE-PLUS NETWORK MAPLE CODE

```
FL1 := 1 - exp(-(t - y12) / 5);
FL2 := 1 - exp(-(t - y12 - y34) / 10);
```

```
FL3 := 1 - exp(-(t - y34) / 10);
FL4 := 1 - exp(-t / 10);
f12 := exp(-y12 / 5) / 5;
f34 := exp(-y34 / 5) / 5;
F := int(int(FL1 * FL2 * FL3 * FL4 * f12 * f34, y34 = 0 .. t - y12), y12 = 0 .. t);
```

APPENDIX B. BRIDGE NETWORK MAPLE CODE

```
FL1 := t - y12;
FL2 := t - y12 - y34;
FL3 := t - y34;
f12 := 1;
f34 := 1;
F1 := int(int(FL1 * FL2 * FL3 * f12 * f34, y34 = 0 .. t - y12), y12 = 0 .. t);
F2 := simplify(
      int(int(1 * 1 * 1 * f12 * f34, y34 = 0 .. t - 1 - y12), y12 = 0 .. t - 1) +
      int(int(1 * FL2 * 1 * f12 * f34, y34 = t - 1 - y12 .. t - 1), y12 = 0.. t - 1) +
      int(int(1 * FL2 * FL3 * f12 * f34, y34 = t - 1 .. 1), y12 = 0 .. t - 1) +
      int(int(FL1 * FL2 * 1 * f12 * f34, y34 = 0 .. t - 1), y12 = t - 1 .. 1) +
      int(int(FL1 * FL2 * FL3 * f12 * f34, y34 = t - 1 .. t - y12), y12 = t - 1 .. 1));
F3 := simplify(1 - ((1 - t + 2) ^ 2) / 2 +
      int(int(1 * FL2 * 1 * f12 * f34, y34 = t - 1 - y12 .. 1), y12 = t - 1 .. 1));
```

APPENDIX C. BRIDGE NETWORK MAPLE CODE WITH `PIECEWISE`

```
F401 := int(int((t - y12) * (t - y12 - y34) * (t - y34), y34 = 0 .. t - y12), y12=0 .. t);
F412 := int(int(1, y34 = 0 .. t - 1 - y12), y12 = 0 .. t - 1) +
        int(int(t - y12 - y34, y34 = t - 1 - y12 .. t - 1), y12 = 0 .. t - 1) +
        int(int((t - y12 - y34) * (t - y34), y34 = t - 1 .. 1), y12 = 0 .. t - 1) +
        int(int((t - y12) * (t - y12 - y34), y34 = 0 .. t - 1), y12 = t - 1 .. 1) +
        int(int((t - y12) * (t - y12 - y34) * (t - y34), y34 = t - 1 .. t - y12),y12 = t - 1 .. 1);
F423 := 1 - ((1 - t + 2) ^ 2) / 2 +
        int(int(t - y12 - y34, y34 = t - 1 - y12 .. 1), y12 = t - 2 .. 1);
F4   := simplify(piecewise(t <= 0, 0, t < 1, F401, t <= 2, F412, t < 3, F423, 1));
```

REFERENCES

[1] V.G. Adlakha and V.G. Kulkarni, A classified bibliography of research on stochastic PERT networks: 1966–1987, INFOR 27 (1989), 272–296.

[2] G.C. Casella and R. Berger, Statistical inference, 2nd ed., Wadsworth & Brooks/Cole, Pacific Grove, CA, 2002.

[3] S.E. Elmaghraby, Activity networks: Project planning and control by network models, John Wiley & Sons, New York, 1977.

[4] D.L. Fisher, D. Saisi, and W.M. Goldstein, Stochastic PERT networks: OP diagrams, critical paths and the project completion time, Comput Oper Res 12 (1985), 471–482.

[5] A.G. Glen, D.L. Evans, and L.M. Leemis, APPL: A probability programming language, Am Stat 55 (2001), 156–166.

[6] J.N. Hagstrom, Computing the probability distribution of project duration in a PERT network, Networks 20 (1990), 231–244.

[7] P.A. Laplante, Editor, Dictionary of computer science, engineering and technology, CRC Press, 2001.

[8] L. Leemis and K. Trivedi, A comparison of approximate interval estimators for the Bernoulli parameter, Am Stat 50 (1996), 63–68.

[9] J.J. Martin, Distribution of the time through a directed, acyclic network, Oper Res 13 (1965), 44–66.

[10] S.K. Park and K.W. Miller, Random number generators: Good ones are hard to find, Commun ACM 31 (1988), 1192–1201.

[11] A.A.B. Pritsker, Introduction to simulation and SLAM II, 4th ed., John Wiley & Sons, New York, 1995.

[12] D.R. Shier, Network reliability and algebraic structures, Oxford University Press, New York, 1991.

[13] J.R. van Dorp and S. Kotz, A novel extension of the triangular distribution and its parameter estimation, Statistician 51, Part 1 (2002), 63–79.