Distribution of the Longest Path of a Stochastic Activity Network with Continuous Activity Durations

Lawrence Leemis · Jeffrey Mallozzi · Kerry Connell *
The College of William & Mary
Williamsburg, VA

May 6, 2002

Abstract

The probability distribution of the longest path of a stochastic activity network with continuous activity durations is calculated for arbitrary networks. The three techniques used here are a recursive Monte Carlo simulation algorithm, series-parallel decomposition, and conditioning. Examples illustrate the use of the three techniques.

1 Introduction

Activity networks are used to plan projects by showing precedence relationships between the various activities that constitute a project. An activity network is a special case of a directed graph in which the nodes or vertices represent events in time and the arcs or edges represent activities with time values representing 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 arc ij. Activity start times are constrained in that no activity emanating from a given node can start until all activities which enter that node have been completed.

*The authors gratefully acknowledge support from the National Science Foundation for providing funding for an Educational Innovation Grant CDA9712718 "Undergraduate Modeling and Simulation Analysis." They also thank Professor Sid Lawrence for his help with the analytic approach in the non-series-parallel case.

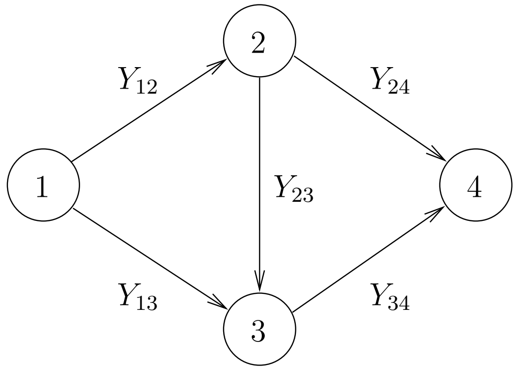

Figure 1: A four-node, five-arc stochastic activity network.

2 Notation

Activity networks have one source node labeled node 1 and one terminal node labeled node n, where n is the number of nodes in the network. Node 1 has one or more arcs leaving it and node n only has arcs entering it. All other nodes must have at least one arc entering and at least one arc leaving. Arc ij is denoted by a_{ij}, where i is the arc's source node and j is its terminal node. There is no possibility for feedback or looping within the network. The number of arcs in the network is denoted by m.

For each arc a_{ij}, there is a random activity duration Y_{ij} with positive support. The distribution of Y_{ij} is determined by its cumulative distribution function $F_{Y_{ij}}(t)$. For each node j there is a random time value T_j which is the time of completion of all activities entering node j. T_n is therefore the time of completion of the entire network.

When an arc leaves node i and enters node j, as in Figure 2, then it is said that i is adjacent *to* j and that j is adjacent *from* i. The "before" set $\mathcal{B}(i)$ is the set of all nodes adjacent to or immediately *before* i, and similarly, the "after" set $\mathcal{A}(i)$ is the set of all nodes adjacent from or immediately *after* i.

For each network, there will be a number of paths leading from node 1 to node n. Let M be the set of all paths, with each path labeled $\pi_1, \pi_2, ..., \pi_r$, where $r = |M|$ is the number of paths. A path may be viewed as a set of arcs leading in succession from node 1 to node n.

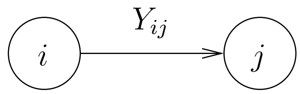

Figure 2: Arc a_{ij}.

Thus arc $a_{ij} \in \pi_k$ means that arc a_{ij} is along the path π_k. The *length* of path π_k, denoted L_k, is the convolution of all Y_{ij} corresponding to the arcs $a_{ij} \in \pi_k$.

For each realization of a stochastic network, there is a *critical path* π_c which is the path with the longest length, $L_c = \max\{L_1, L_2, ..., L_r\}$. 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 $p(\pi_k) = \Pr(\pi_k \equiv \pi_c)$, $k = 1, 2, \ldots, r$. 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 $a_{ij} \in \pi_k$.

2.1 Matrix representation of the network

The first step in building a model is defining a mathematical representation of any network. A matrix is well suited for this task because two subscripts can be used to define arc a_{ij} and the ease of computer implementation. There are two ways in which a matrix can be used to represent an activity network: an adjacency matrix and a node-arc incidence matrix.

2.1.1 Adjacency matrix

The adjacency matrix is an $n \times n$ matrix where a 1 in the i, j^{th} position represents an arc leaving node i and entering node j, and all other positions are marked with a 0. Since all arcs enter a higher numbered node than that which it leaves, the result is that the adjacency matrix is an upper-diagonal matrix with zeros on the diagonal. The matrix

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ & 0 & 1 & 1 \\ & & 0 & 1 \\ & & & 0 \end{bmatrix}$$

is the adjacency matrix of the network in Figure 1. The disadvantage of this network representation, however, is that it is incapable of representing more than one arc between the same two nodes and going in the same direction. Even if it is defined that activity networks can not have more than one arc between two nodes it will become necessary for the parallel decomposition operation presented in Section 4.1.1 for our representation to have the ability to do so in order to decompose a network. Thus a slightly more flexible representation is required to represent networks for the algorithms presented here.

2.1.2 Node-arc incidence matrix

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

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

The matrix

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

for example, is a node-arc incidence matrix of the network in Figure 1. The only limitation of this representation is that it can not show an arc which leaves and enters the same node. This ability will not be necessary since 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 a 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 for checking analytic algorithms presented subsequently.

3.1 Point estimators

If T_1 is assumed to be 0.0 (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 $E[T_j]$, for example, is the sample mean of the T_j's generated using the algorithm to follow (which is based on the expression above). The point estimator for the probability that path π_k is a critical path, for example, 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 the probability that arc a_{ij} is along the critical path or

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

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

3.2 Interval estimators

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

$$\bar{x} - t_{n-1,\alpha/2}\frac{s}{\sqrt{n-1}} < E[T_j] < \bar{x} + t_{n-1,\alpha/2}\frac{s}{\sqrt{n-1}},$$

for any node j, where n is the number of replications of the simulation, $\bar{x}$ is the sample mean of the simulated T_j's, s is the sample standard deviation of the simulated T_j's, 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 left to right in the stochastic activity network, resulting in improved actual coverage for this confidence interval.

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

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

where y is the number of occurrences of some event out of n independent replications, F denotes the F distribution, and the third subscript on F denotes the right-hand tail probability (Leemis and Trivedi, 1996).

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 each arc a_{ij} are generated *prior* to the call to T. Loops and conditions are indicated by indentation.

Global parameters: node-arc incidence matrix N, one realization of activity durations Y_{ij}

Procedure name: T

Argument: node j

int i — loop index for rows of N
int $k \leftarrow 1$ — index for the columns of N
int $l \leftarrow 0$ — index for the predecessors to node j
float t — completion time of arc a_{ij}
float $t_{\max} \leftarrow 0.0$ — longest time of all possible paths to node j
while $(l < |\mathcal{B}(j)|)$ — loop through predecessor nodes to node j
    if $(N[j, k] = -1)$ — if column k of N corresponds to an arc entering node j
        $i \leftarrow 1$ — begin search for predecessor node
        while$(N[i, k] \neq 1)$ — while i does not correspond to the predecessor index
            $i \leftarrow i + 1$ — increment i
        $t \leftarrow T_i + Y_{ij}$ — recursive call: t is the completion time of a_{ij}
        if $(t \geq t_{\max})$ $t_{\max} \leftarrow t$; — choose largest completion time
        $l \leftarrow l + 1$ — increment predecessor index
    $k \leftarrow k + 1$ — increment column index
return $(t_{\max})$ — return completion time T_j

One advantage to this recursive implementation is that it avoids the use of an event calendar, which is typically the case in the standard discrete-event simulation approach. 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. This algorithm has been implemented in C and is available from the first author.

3.4 Example

The simulation approach was tested on an activity network described by Pritsker (1995, pages 216–221) shown in Figure 3 with $n = 6$ nodes and $m = 9$ activities. One node-arc

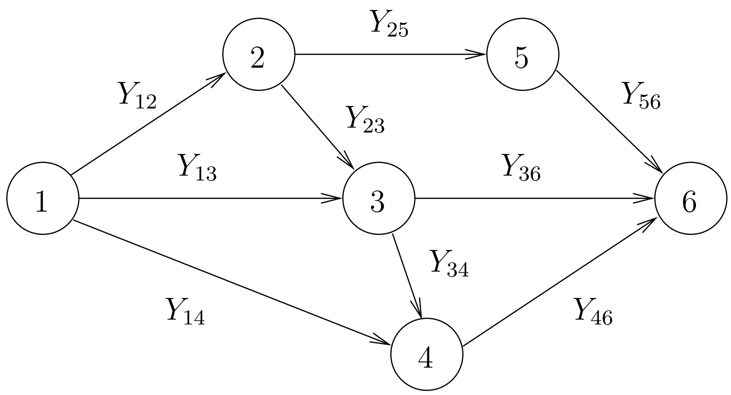

Figure 3: Network from Pritsker (1995, pages 216–221).

incidence matrix that describes the network is

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

The distribution of the duration of each arc a_{ij} is given in Table 1. The three parameters on the triangular distribution are the minimum, mode, and maximum. The triangular distribution is often used for stochastic activity networks using the three parameters as the optimistic, most likely, and pessimistic times to complete an activity. An algorithm for parameter estimation using maximum likelihood is given in van Dorp and Kotz (2002).

The $r = 6$ paths in the network are given in Table 6. The simulation was run for one million realizations of the network using the multiplicative linear congruential generator $x_{i+1} = 7^5 x_i \bmod (2^{31} - 1)$ (Park and Miller, 1988) and an initial seed of 8641. When the algorithm is called to generate one million T_6's, the order of the recursive calls associated with the node-arc incidence matrix given above is T_6, T_3, T_1, T_2, T_1, etc. For each realization, the

Arc Index	Arc	Distribution of Y_{ij}
1	$a_{1,2}$	`Triangular`(1, 3, 5)
2	$a_{1,3}$	`Triangular`(3, 6, 9)
3	$a_{1,4}$	`Triangular`(12, 13, 19)
4	$a_{2,5}$	`Triangular`(3, 9, 12)
5	$a_{2,3}$	`Triangular`(1, 3, 8)
6	$a_{3,6}$	`Triangular`(8, 9, 16)
7	$a_{3,4}$	`Triangular`(4, 7, 13)
8	$a_{5,6}$	`Triangular`(3, 6, 9)
9	$a_{4,6}$	`Triangular`(1, 3, 8)

Table 1: Distributions for arc durations.

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	$1 \to 2 \to 3 \to 4 \to 6$	$\{a_{12}, a_{23}, a_{34}, a_{46}\}$

Table 2: Paths π_k.

time to completion T_j was calculated for each node in the network according to the algorithm in Section 3.3. Some sample 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 time to completion, and the 95% confidence interval half-widths. Table 4 shows point estimates and 95% confidence interval interval halfwidths for $p(\pi_k)$. The point estimates total 1.001, rather than 1, due to roundoff. Table 5 shows point estimates and 95% confidence interval halfwidths for ρ_{ij}. Table 5 also shows which paths each arc a_{ij} are in according to the path indexes in Table 6. Figure 4 shows the empirical cdf of the time to complete the network for the one million realizations. 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 one million realizations of the number of network

j	$\hat{E}[T_j]$	$\sqrt{\hat{V}[T_j]}$	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 3: Estimated expected time to completion T_j for one million simulation replications and 95% confidence interval halfwidth h.

k	$\hat{p}(\pi_k)$	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 4: Estimated critical path probability $\hat{p}(\pi_k)$ for one million simulation replications and 95% confidence interval halfwidth h.

Arc	Paths	$\hat{\rho}_{ij}$	h
$a_{1,2}$	π_2, π_3, π_6	0.600	0.0010
$a_{1,3}$	π_1, π_5	0.203	0.0008
$a_{1,4}$	π_4	0.198	0.0008
$a_{2,5}$	π_3	0.129	0.0007
$a_{2,3}$	π_2, π_6	0.469	0.0010
$a_{3,6}$	π_1, π_2	0.244	0.0008
$a_{3,4}$	π_5, π_6	0.429	0.0010
$a_{5,6}$	π_3	0.129	0.0007
$a_{4,6}$	π_4, π_5, π_6	0.627	0.0010

Table 5: Estimated criticality $\hat{\rho}_{ij}$ for one million simulation replications and 95% confidence interval halfwidth h.

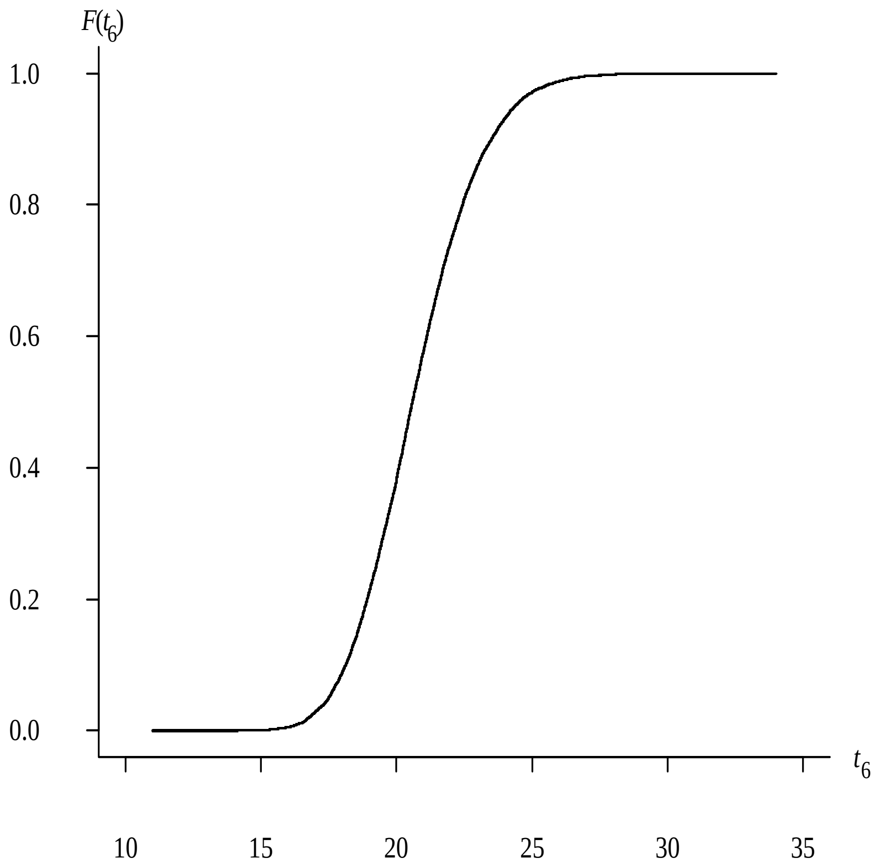

Figure 4: Empirical cdf of T_6 for 1,000,000 replications.

completion times that fell in 23,000 equal-width cells on $11 < t_6 < 34$, e.g., $[11.000, 11.001)$, $[11.001, 11.002)$, $\ldots$, $[33.999, 34.000)$.

4 Analytical approaches

While general, the simulation approach has a distinct drawback. Each additional digit of accuracy of 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 sub-sections outline efforts to arrive at 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 reduced by a sequence of decompositions to a simple network consisting of one arc and two nodes. Decompositions 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 decomposed, the distribution of the remaining arc is the distribution of the time to complete the original network T_n. The recursive algorithm then reconstructs the network to determine the critical path probabilities and criticalities. Parallel and series decompositions and reconstructions are described below. These represent the only types of operations that the algorithm described here will encounter to reduce a series-parallel network to a single arc to determine the distribution of T_n and then reconstruct the network to determine the critical path probabilities and the criticalities.

4.1.1 Parallel decomposition

The algorithm described in Section 4.1.5 will encounter two two arcs in *parallel*, as illustrated in Figure 5. A parallel decomposition is the process of reducing these two arcs to a single arc. Let X_{ij} denote the duration of one arc, and let Y_{ij} denote the duration of the other.

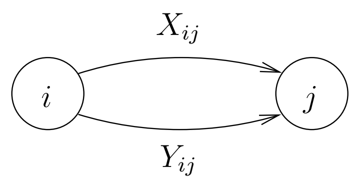

Figure 5: Two arcs in parallel.

Without loss of generality, if $T_i = 0.0$ then $T_j = \max\{X_{ij}, Y_{ij}\}$. So

$$\begin{aligned} F_{T_j}(t) &= \Pr[T_j \le t] \\ &= \Pr[\max\{X_{ij}, Y_{ij}\} \le t] \\ &= \Pr[X_{ij} \le t \text{ and } Y_{ij} \le t] \\ &= \Pr[X_{ij} \le t] \cdot \Pr[Y_{ij} \le t] \\ &= F_{X_{ij}}(t) F_{Y_{ij}}(t) \end{aligned}$$

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

4.1.2 Parallel reconstruction

Consider the single arc a_{ij} after a parallel decomposition. If the probability that this arc is on the critical path, ρ_{ij}, 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 recursive algorithm given in Section 4.1.5 after the recursive call calculates the distribution of $X_{ij} - Y_{ij}$ and calculates the probability that this random variable is positive.

4.1.3 Series decomposition

The algorithm described in this subsection will encounter two arcs in *series*, as illustrated in Figure 6. Let Y_{ij} and Y_{jk} denote the durations of the two arcs with CDFs $F_{Y_{ij}}(t)$ and

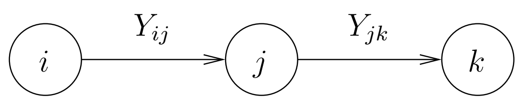

Figure 6: Two arcs in series.

$F_{Y_{jk}}(t)$. Without loss of generality, if $T_i = 0$ then $T_k = Y_{ij} + Y_{jk}$. The CDF of T_k is (Casella and Berger, 2002, page 215):

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

4.1.4 Series reconstruction

Consider the single arc a_{ik} after a series decomposition. For any two arcs a_{ij} and a_{jk} in series which have been decomposed into a single arc a_{ik}, $\rho_{ij} = \rho_{jk} = \rho_{ik}$. If two arcs in series are along the critical path then so must their decomposed arc.

4.1.5 Algorithm

The following recursive algorithm decomposes 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 Y (indexed by the index of each arc in the matrix N) and the criticalities of each arc are stored in a vector C (indexed by the index of each arc in N). This algorithm recursively decomposes the network one decomposition at a time until the network consists of just one arc. At this time, the value of C for that one arc is set to one and then the network is recomposed on the return calls in reverse order, determining values of C for each arc based on the value of C of the decomposed arc. The algorithm returns the distribution of the time to complete the network and C contains ρ_{ij} for all arcs a_{ij} in the network.

Arguments: node-arc incidence matrix N, CDFs of activity durations Y_{ij}

Procedure name: *GetCriticalities*

Returned value: Criticalities $C[m]$

```
GetCriticalities := proc(NET, YA, ma)
  local C, N, Y, g, i, j, time, m, rowsum, absrowsum, negidx;   local posidx, c, d, l, k,
empty, same, changed, x3, x4;
   g ← [[x → −x], [−∞, ∞]];
   m ← ma;
   N ← NET;
   Y ← YA;
   if m = 1 then
      C[1] ← 1;
      time ← Y[1];
      print(NET);
      print(C);
      print(time);
      RETURN(C)
   end if;
   for i to maxnodes do
      rowsum ← 0;
      absrowsum ← 0;
      for j to maxedges do
         rowsum ← rowsum + N[i, j];
         absrowsum ← absrowsum + abs(N[i, j]);
         if N[i, j] = −1 then negidx ← j end if;
         if N[i, j] = 1 then posidx ← j end if
      end do;
      if rowsum = 0 and absrowsum = 2 then
         c ← Y[negidx];
         d ← Y[posidx];
         Y[negidx] ← Convolution(c, d);
         for j to maxnodes do
            N[j, negidx] ← N[j, negidx] + N[j, posidx];
            N[j, posidx] ← 0
```

```
        end do;
        m ← m − 1;
        C ← GetCriticalities(N, Y, m);
        for l to maxnodes do
            if N[l, negidx] = −1 then
                N[l, negidx] ← 0; N[l, posidx] ← −1
            end if
        end do;
        N[i, negidx] ← −1;
        N[i, posidx] ← 1;
        Y[negidx] ← c;
        Y[posidx] ← d;
        C[posidx] ← C[negidx];
        print(N);
        print(C);
        RETURN(C)
    end if
end do;
for j to maxedges do for k from j + 1 to maxedges do
    empty ← 0;
    same ← 0;
    for i to maxnodes do
        if N[i, j] ≠ N[i, k] then same ← same + 1
    end if;
    if N[i, j] ≠ 0 then empty ← empty + 1
    end if
end do;
if same = 0 and empty ≠ 0 then
    c ← Y[j];
    d ← Y[k];
    Y[j] ← Maximum(c, d);
    for i to maxnodes do N[i, k] ← 0 end do;
    m ← m − 1;
    changed ← 1;
    C ← GetCriticalities(N, Y, m);
    for i to maxnodes do N[i, k] ← N[i, j]
    end do;
    Y[j] ← c;
    Y[k] ← d;
    x3 ← Transform(YA[k], g);
    x4 ← Convolution(YA[j], x3);
    C[k] ← CDF(x4, 0)*C[j];
    C[j] ← C[j] − C[k];
    print(N);
```

```
        print(C);
        RETURN(C)
    end if
    end do
    end do;
    RETURN(C)
end proc
```

This algorithm requires symbolic processing capability in order to calculate the distribution of the maximum of two independent random variables (for parallel decomposition) and the distribution of the convolution of two independent random variables (for series decomposition). The Maple-based APPL language (Glen, Leemis, and Evans, 2001) has procedures `Maximum` and `Convolution` that can be used for these operations.

4.1.6 Example

Figure 7 is an example of a series-parallel network from Elmaghraby (1978, page 261) which can be described by the node-arc incidence matrix:

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

The network can be decomposed and recomposed as illustrated in Figure 8.

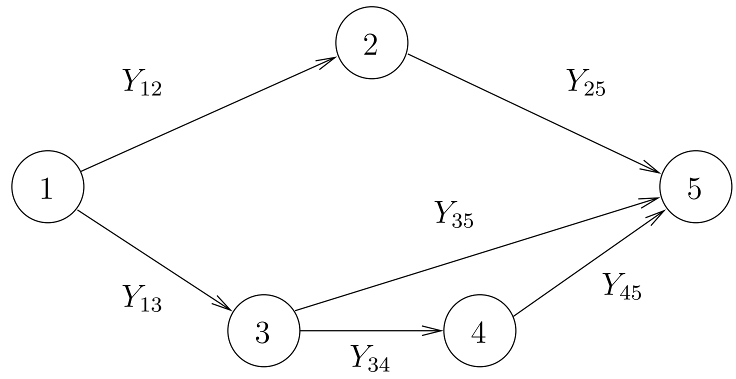

Figure 7: Series-Parallel network from Elmaghraby (1978, page 261).

If the duration of each arc Y_{ij} is an exponential(b) random variable (where b is the failure rate), the CDF of the time to complete the network (T_5) according to the algorithm in Section 4.1.5 is:

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

for $t > 0$. This CDF is plotted in Figure 9 for $b = 1/2$.

The $r = 3$ paths in the network are described in Table 6. Table 6 also shows which paths each arc a_{ij} are in according to the path indexes.

k	Node sequence	π_k	$p(\pi_k)$
1	$1 \to 2 \to 5$	$\{a_{12}, a_{25}\}$	$115/432 = 0.266$
2	$1 \to 3 \to 5 \to 6$	$\{a_{13}, a_{35}\}$	$317/1728 \cong 0.183$
3	$1 \to 3 \to 4 \to 5$	$\{a_{13}, a_{34}, a_{45}\}$	$317/576 \cong 0.550$

Table 6: Paths π_k and estimated critical path probabilities $p(\pi_k)$ when $b = 0.5$.

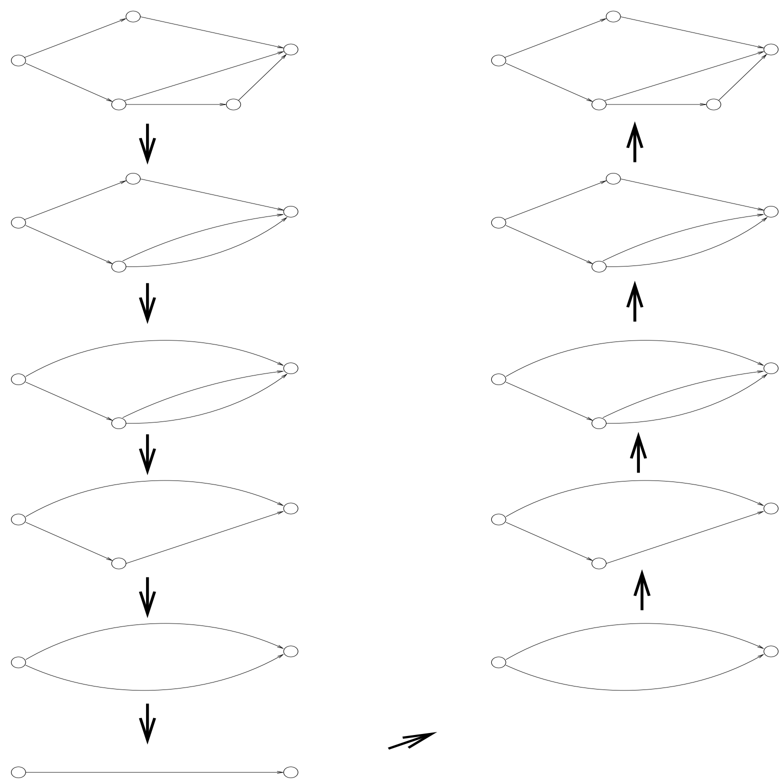

Figure 8: Decomposition and recomposition of a series-parallel network.

Arc	Paths	ρ_{ij}
$a_{1,2}$	π_1	$115/432 \cong 0.266$
$a_{1,3}$	π_2, π_3	$317/432 \cong 0.734$
$a_{2,5}$	π_1	$115/432 \cong 0.266$
$a_{3,5}$	π_2	$317/1728 \cong 0.183$
$a_{3,4}$	π_3	$317/576 \cong 0.550$
$a_{4,5}$	π_3	$317/576 \cong 0.550$

Table 7: Criticalities ρ_{ij} when $b = 0.5$.

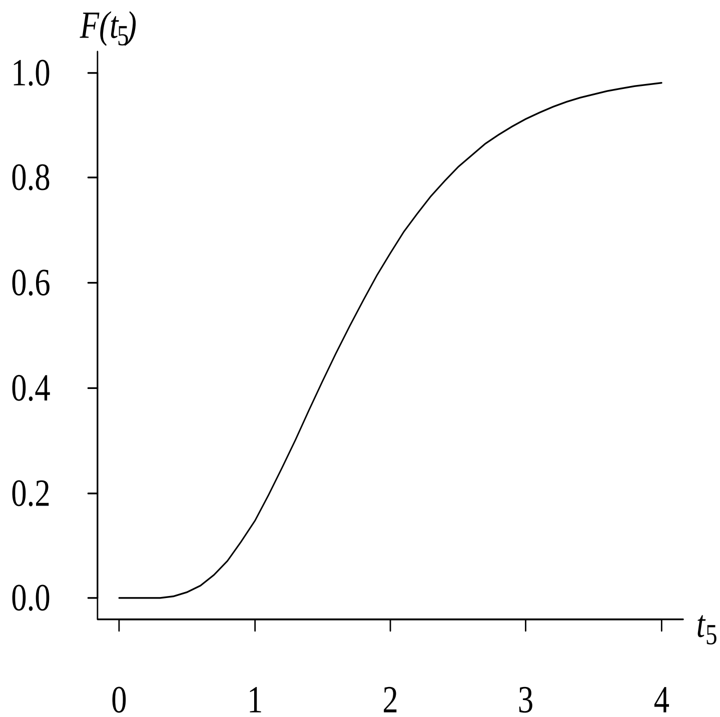

Figure 9: CDF of T_5 for $b = 1/2$.

4.2 Non Series-Parallel Networks

Now consider the case of a non-series-parallel network. Determining the distribution of the time to complete the network is complicated by the fact that the network cannot be decomposed as in the series-parallel case. We begin with two examples that illustrate the difficulty.

Example 1: Elmaghraby (1978, page 305) considers the network shown in Figure 10. The

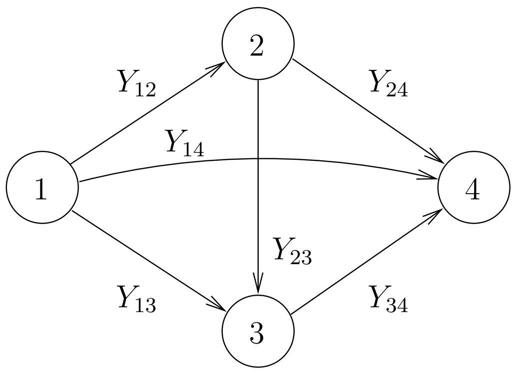

Figure 10: A four-node, six-arc stochastic activity network.

activity durations are exponentially distributed with means 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

$$\begin{aligned} W_1 &= Y_{12} + Y_{24} \\ W_2 &= Y_{12} + Y_{23} + Y_{34} \\ W_3 &= Y_{13} + Y_{34} \\ W_4 &= Y_{14}. \end{aligned}$$

Since Y_{12} and Y_{34} 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

$$F_{T_4}(t|y_{12}, y_{34}) = F_{W_1}(t|y_{12}, y_{34})F_{W_2}(t|y_{12}, y_{34})F_{W_3}(t|y_{12}, y_{34})F_{W_4}(t|y_{12}, y_{34})$$

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

$$\begin{aligned}
\Pr(T_4 \le t) &= \Pr(\max\{W_1, W_2, W_3, W_4\} \le t) \\
&= \Pr(W_1 \le t, W_2 \le t, W_3 \le t, W_4 \le t) \\
&= \Pr(W_1 \le t)\Pr(W_2 \le t)\Pr(W_3 \le t)\Pr(W_4 \le t).
\end{aligned}$$

The CDFs for

$$\begin{aligned}
W_1 &= y_{12} + Y_{24} \\
W_2 &= y_{12} + Y_{23} + y_{34} \\
W_3 &= Y_{13} + y_{34} \\
W_4 &= Y_{14}
\end{aligned}$$

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

$$\begin{aligned}
F_{W_1}(t|y_{12}, y_{34}) &= \begin{cases} 0 & t < y_{12} \\ 1 - e^{-(t-y_{12})/5} & t \ge y_{12}, \end{cases} \\
F_{W_2}(t|y_{12}, y_{34}) &= \begin{cases} 0 & t < y_{12} + y_{34} \\ 1 - e^{-(t-y_{12}-y_{34})/10} & t \ge y_{12} + y_{34}, \end{cases} \\
F_{W_3}(t|y_{12}, y_{34}) &= \begin{cases} 0 & t < y_{34} \\ 1 - e^{-(t-y_{34})/10} & t \ge y_{34}, \end{cases} \\
F_{W_4}(t|y_{12}, y_{34}) &= \begin{cases} 0 & t < 0 \\ 1 - e^{-t/10} & t \ge 0. \end{cases}
\end{aligned}$$

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 $y_{34} \leq t - y_{12}$ from the support of W_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 $t > 0$. There are two ways to evaluate this integral using a symbolic language. The first is to use the limits as indicated in the example. The Maple code for this example is given in Appendix A. The second way to evaluate the integral is to run both integration limits from 0 to ∞ but use Maple's `piecewise` function to assure that the proper limits of integration are appropriately executed. This example was particularly easy because all of the activity durations had support on $(0, \infty)$. This second computational approach is important because, as will be seen in the next example, the limits of integration can become unwieldy for more complicated distributions or complicated networks.

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

$$\begin{aligned} W_1 &= y_{12} + Y_{24} \\ W_2 &= y_{12} + Y_{23} + y_{34} \\ W_3 &= Y_{13} + y_{34} \end{aligned}$$

as

$$F_{W_1}(t|y_{12}, y_{34}) = \begin{cases} 0 & t < y_{12} \\ t - y_{12} & y_{12} \leq t \leq y_{12} + 1 \\ 1 & t > y_{12} + 1 \end{cases}$$

$$F_{W_2}(t|y_{12}, y_{34}) = \begin{cases} 0 & t < y_{12} + y_{34} \\ t - y_{12} - y_{34} & y_{12} + y_{34} \le t \le y_{12} + y_{34} + 1 \\ 1 & t > y_{12} + y_{34} + 1 \end{cases}$$

$$F_{W_3}(t|y_{12}, y_{34}) = \begin{cases} 0 & t < y_{34} \\ t - y_{34} & y_{34} \le t \le y_{34} + 1 \\ 1 & t > y_{34} + 1. \end{cases}$$

As in the previous example, the unconditional CDF of the network completion time T_4 is given by

$$F_{T_4}(t) = \int \int F_{W_1}(t|y_{12}, y_{34}) F_{W_2}(t|y_{12}, y_{34}) F_{W_3}(t|y_{12}, y_{34}) f_{Y_{12}}(y_{12}) f_{Y_{34}}(y_{34})\, dy_{34}\, dy_{12}.$$

The support of T_4 is $0 < t_4 \le 3$. The limits of integration are more complicated than in the previous example. For $0 < t \le 1$

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

as in the previous case. For $1 < t \le 2$, Figure 11 illustrates, for $t = 1.8$, the integrand over various regions in the y_{12}, y_{34} coordinate system. So the integral is

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

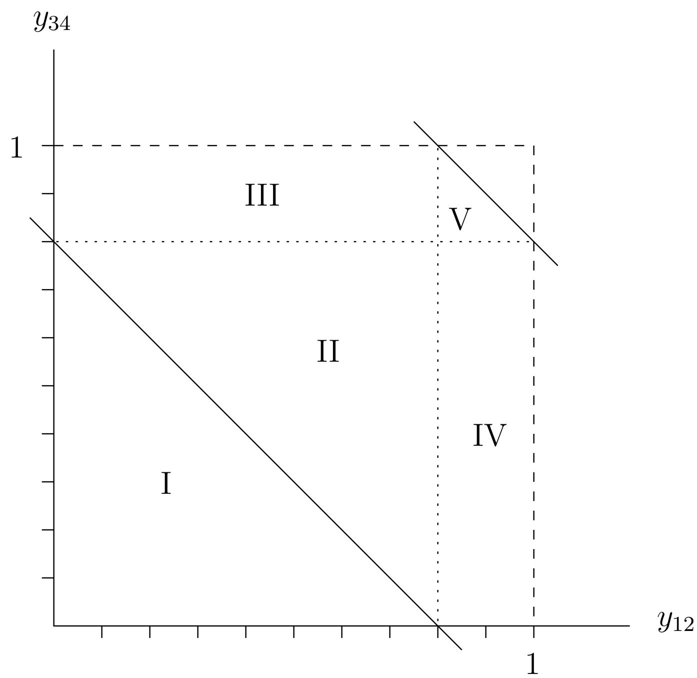

Figure 11: Integration regions associated with $t = 1.8$.

for $1 < t \leq 2$. The roman numerals at the right of each double integral denote the region in Figure 11. Finally, for $2 < t \leq 3$,

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

which yields:

$$F_{T_4}(t) = \begin{cases} 0 & t \leq 0 \\ \frac{11}{120} t^5 & 0 < t \leq 1 \\ -\frac{1}{6} t^4 - \frac{1}{3} t^2 - \frac{1}{120} t^5 + \frac{2}{3} t^3 + \frac{1}{10} - \frac{1}{6} t & 1 < t \leq 2 \\ -\frac{7}{2} + \frac{9}{2} t - \frac{3}{2} t^2 + \frac{1}{6} t^3 & 2 < t \leq 3 \\ 1 & t > 3. \end{cases}$$

As before this result can be computed solving the integrals directly or by using Maple's `piecewise` capability (the code is given in Appendix B).

Appendix A

```
restart;
Fw1 := 1 - exp(-(t - y12) / 5);
Fw2 := 1 - exp(-(t - y12 - y34) / 10);
Fw3 := 1 - exp(-(t - y34) / 10);
Fw4 := 1 - exp(-t / 10);
f12 := exp(-y12 / 5) / 5;
f34 := exp(-y34 / 5) / 5;
F := int(int(Fw1 * Fw2 * Fw3 * Fw4 * f12 * f34, y34=0 .. t - y12), y12 = 0 .. t);
```

Appendix B

```
restart;
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] Casella, G.C., and Berger, R. (2002), *Statistical Inference*, Second Edition, Wadsworth & Brooks/Cole.

[2] Elmaghraby, Salah E., *Activity Networks: Project Planning and Control by Network Models*, John Wiley & Sons, 1977.

[3] Glen, A., Leemis, L., Evans, D.L. (2001), APPL: A Probability Programming Language, *The American Statistician*, Volume 55, Number 2, 156–166.

[4] Hagstrom, Jane N., Computing the Probability Distribution of Project Duration in a PERT Network, *Networks*, Vol. 20, 1990, pp 231–244.

[5] Leemis, L. and Trivedi, K. (1996), "A Comparison of Approximate Interval Estimators for the Bernoulli Parameter", *The American Statistician*, Vol. 50, No. 1, pp 63–68.

[6] Park, S.K. and Miller, K.W. (1988), "Random Number Generators: Good Ones are Hard to Find", *Communications of the ACM*, Vol. 31, pp 1192–1201.

[7] Pritsker, A. Alan B., *Introduction to Simulation and SLAM II*, Third Edition, John Wiley & Sons, 1986.

[8] Shier, Douglas R., *Network Reliability and Algebraic Structures*, Oxford University Press, 1991.

[9] van Dorp, J.R. and Kotz, S. (2002), "A Novel Extension of the Triangular Distribution and its Parameter Estimation", *The Statistician*, Volume 51, Part 1, pp 63–79.