

# Exact and Approximate Distributions of Stochastic PERT Networks

Derek O'Connor  
University College, Dublin

December 13, 2007

## Abstract

This paper presents a definition and a characterization of *Conditioning Activities* which, along with a simple theorem, leads to an algorithm that can be used to calculate exact or approximate (Monte Carlo) distributions for the completion time of Stochastic PERT Networks.

The time complexity, for networks of  $n$  activities, is exponential for exact distributions and  $O(n^2N)$  for Monte Carlo approximations with sample size  $N$ . Computational tests are reported for 4 networks varying in size from 10 to 40 activities.

**Keywords & Phrases :** Stochastic PERT, Enumeration, Conditioning Activities, Minimum Conditioning Sets, Conditional Monte Carlo, Variance Reduction.

This is a revised version of a paper written in the early-1980s. It is based on the author's doctoral dissertation, *Stochastic Network Distributions*, which was supervised by John F. Muth (1930–2005) at Indiana University, 1977. See [www.derekroconnor.net](http://www.derekroconnor.net)

## 1 INTRODUCTION

The computation of the completion time distribution of a stochastic PERT network — the Stochastic PERT Problem — is a difficult, essentially intractable, problem. Hagstrom [Hag 88] has shown that this problem is in the class of *#P-Complete* problems ( see [G&J 78] ) which means that it is equivalent to a counting problem and that exact solutions can be obtained only by exhaustive enumeration. Hence exact solutions can be obtained for small or special networks only.

Since PERT and CPM networks were introduced in the 1950s many methods for computing their completion time distributions have been devised. These may be classified as approximate, bounding, or exact. The methods of Martin [Mar 65], and Charnes, Cooper and Thompson [CCT 64] are exact and were applied to small, special networks. The methods of Kleindorfer [Kle 71], Shogan [Sho 77], Dodin [Dod 85], and Robillard and Trahan [R&T 77] are bounding methods, each giving bounds on the exact distribution. The first three of these bounding methods have been applied to large, complex networks with varying degrees of success. The simulation methods of Van Slyke [Van 63] and Hartley & Wortham [H&W 66] are approximate. The accuracy of the approximations depend on the sample sizes, and for large networks computation costs can be prohibitive if high accuracy is required. The conditional Monte Carlo methods of Burt and Garman [B&G 71] & [Gar 72], and Sigal *et al.* [SPS 79] & [SPS 80] are approximate. These methods are, in general, superior to simple (unconditional) Monte Carlo because they have smaller sampling variances.

The new algorithm to be described here can be used to calculate exact or approximate distributions, depending on the amount of computation time one is willing to spend. It is somewhat similar to the methods of Garman [Gar 72] and Sigal *et al.* [SPS 79], in that certain activities of the network are singled out for special attention.

An outline of the paper is as follows:

*Section 1* contains the definitions and assumptions used throughout the paper. The standard algorithm for deterministic PERT networks is defined. This is used as the basis of discussion in Sections 2 and 3. *Section 2* discusses the cause of the difficulty in solving stochastic PERT networks. The special activities that cause this difficulty are defined and this definition is used in a theorem which is the basis of the new algorithm. In *Section 3* the new algorithm is defined and a small, hand-calculated example is given. *Section 4* shows how the exact algorithm of Section 3 can be used as a Monte Carlo approximation algorithm. The approximation algorithms published to date are discussed and compared with the exact algorithm. An analysis and summary of the time complexities of these algorithms is given. *Section 5* discusses the implementation and testing of the new algorithm. Tests on 4 networks ranging in size from 10 to 40 activities are reported and some conclusions are drawn.

## 2 ASSUMPTIONS, DEFINITIONS, AND NOTATION.

A *PERT Network* is a directed acyclic graph  $D = (N, A)$ , where  $N$  is a set of *nodes* and  $A$  is a set of *arcs*. Let the nodes be labelled  $i = 1, 2, \dots, n$ , where  $n = |N|$ . Let  $(i, j)$  denote the arc directed from node  $i$  to node  $j$ . Let  $Pred(i)$  denote the set of *immediate predecessors* of node  $i$ , i.e.,  $Pred(i) = \{j | (j, i) \in A\}$ . Let  $Succ(i)$  denote the set of *immediate successors* of node  $i$ , i.e.,  $Succ(i) = \{j | (i, j) \in A\}$ . Node  $j$  is a *predecessor (successor)* of node  $i$  if there exists a directed path from  $j$  to  $i$  ( $i$  to  $j$ ). Assume that there is one node with no predecessors and call this the *source* node. Likewise, assume that there is one node with no successors and call this the *sink* node. Assume that the nodes are topologically labelled. This

means that the source is labelled 1, the sink labelled  $n$ , and for every arc  $(i, j)$  in the network,  $i < j$ .

Adopting the “activity-on-the-node” convention, let node  $i$  of the network represent the  $i^{\text{th}}$  activity of the project, and let arc  $(i, j)$  represent the precedence relation “activity  $i$  must be completed before activity  $j$  can start”. The activity-on-the-node convention is used because it simplifies the subsequent discussion and derivations.

For each node  $i = 1, 2, \dots, n$  define the following non-negative discrete random variables :

1.  $X_i$  = Start Time of Activity  $i$ , taking on all values  $x_i$  in the range  $[x'_i, x''_i]$ ,
2.  $A_i$  = Activity Time (Duration) of Activity  $i$ , taking on all values  $a_i$  in the range  $[a'_i, a''_i]$ ,
3.  $Y_i$  = Finish Time of Activity  $i$ , taking on all values  $y_i$  in the range  $[y'_i, y''_i]$ .
4.  $Y_n$  is called the *completion time* of the project.

Assume the activity times  $\{A_i\}$  are independent.

Because the network is topologically labelled, the project starts at time  $X_1$  and is completed at time  $Y_n$ .

**Definition 1.** (Stochastic PERT Problem) : Given a PERT network of  $n$  activities and the distributions of the independent random activity times  $A_1, A_2, \dots, A_n$ , find the completion time  $Y_n$ , and its distribution function  $\Pr(Y_n \leq t)$ .

## 2.1 Calculating the Completion Time $Y_n$

The completion time of the project,  $Y_n$ , can be calculated as follows, assuming that the project starts at time 0 and no activity can start until all its immediate predecessors have been completed :

$$X_1 = 0, \quad Y_1 = A_1, \quad (1)$$

$$X_i = \max_j \{Y_j\}, \quad j \in \text{Pred}(i), \quad i = 2, 3, \dots, n \quad (2)$$

$$Y_i = X_i + A_i, \quad i = 2, 3, \dots, n \quad (3)$$

$$\text{and } Y_n = \text{Project Completion Time.} \quad (4)$$

The set of values for each random variable are easily calculated because the random variables take on all values in their range. Hence, for any random variable all we need to calculate are the lower and upper values in the range, as shown below :

$$x'_i = \max_j \{y'_j\}, \quad x''_i = \max_j \{y''_j\}, \quad j \in \text{Pred}(i), \quad (5)$$

$$y'_i = x'_i + a'_i, \quad y''_i = x''_i + a''_i, \quad (6)$$

## 2.2 Calculating the Completion Time Distribution $F(Y_n; t)$

We now define the distribution functions of the random variables  $X_i, A_i,$  and  $Y_i$ , using the following notation to avoid multi-level subscripting :

For each node  $i$  let

$$F(X_i; t) = \Pr(X_i \leq t), \quad F(Y_i; t) = \Pr(Y_i \leq t), \quad f(A_i; t) = \Pr(A_i = t), \quad F(A_i; t) = \Pr(A_i \leq t). \quad (7)$$

The distribution functions for the random variables defined in (1), (2), and (3) are calculated as follows :

$$F(X_1; t) = 1, \quad F(Y_1; t) = F(A_1; t), \quad \text{for all } t \geq 0. \quad (8)$$

for  $i = 2, 3, \dots, n$

$$F(X_i; t) = \Pr \left[ \max_{j \in \text{Pred}(i)} \{Y_j\} \leq t \right] = \Pr \left[ \bigcap_{j \in \text{Pred}(i)} \{Y_j \leq t\} \right], \quad x'_i \leq t \leq x''_i \quad (9)$$

$$F(Y_i; t) = \Pr [X_i + A_i \leq t] = \sum_{a_i=a'_i}^{a''_i} \Pr \left[ (X_i \leq t - a_i) \text{ and } (A_i = a_i) \right], \quad y'_i \leq t \leq y''_i \quad (10)$$

Let us simplify expressions in (9) and (10) by assuming, without loss of generality, that node  $i$  has 2 immediate predecessors, node  $j$  and node  $k$ . The expressions for node  $i$  are now :

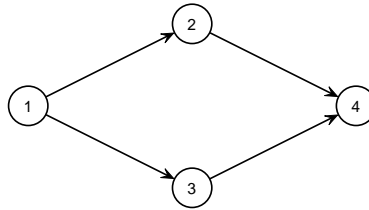
$$F(X_i; t) = \Pr[\max\{Y_j, Y_k\} \leq t] = \Pr[(Y_j \leq t) \text{ and } (Y_k \leq t)], \quad x'_i \leq t \leq x''_i. \quad (11)$$

Now, by definition,  $X_i$  and  $A_i$  are independent. Hence, from (10) we have,

$$F(Y_i; t) = \sum_{a_i=a'_i}^{a''_i} \Pr[X_i \leq t - a_i] \times \Pr[A_i = a_i] = \sum_{a_i=a'_i}^{a''_i} F(X_i; t - a_i) f(A_i; a_i), \quad y'_i \leq t \leq y''_i. \quad (12)$$

### 2.3 The Main Difficulty

It is obvious that (12) can be calculated easily once (11) has been calculated. Unfortunately,  $F(X_i; t)$  in (11) cannot be calculated easily because, in general,  $Y_j$  and  $Y_k$  are dependent random variables. To see how this dependence occurs, consider the small network shown in Figure 1.



**Figure 1** : Dependence in a Small Network.

The finish times of activities 2 and 3 are  $Y_2 = A_1 + A_2$  and  $Y_3 = A_1 + A_3$ . The start time of activity 4 is  $X_4 = \max\{Y_2, Y_3\}$ .  $Y_2$  and  $Y_3$  are not independent because  $A_1$  is common to both. Hence, to calculate the distribution of  $X_4$  we would need to calculate the joint probability mass function  $\Pr[Y_2 = a_1 + a_2, Y_3 = a_1 + a_3]$  for all possible triples  $(a_1, a_2, a_3)$ . This is not practicable, even for small networks.

Thus we see that the central difficulty in the Stochastic PERT Problem is the calculation of  $F(X_i; t)$ , the start time distribution of each activity (the reason for using the “activity-on-node” convention is to isolate this calculation).  $F(X_i; t)$  is the distribution of the maximum of a set of dependent random variables and its calculation is a special case of the difficult problem of evaluating a multiple sum or integral over a complicated domain.

The calculation of  $F(X_i;t)$  can be avoided if we are willing to settle for a bound or an approximation of  $F(X_i;t)$  and hence  $F(Y_n;t)$ . The methods used for calculating approximations and bounds can be broadly classified as follows :

1. *Conditioning* — on a certain set of activity times to eliminate dependence between the finish times ( $Y$ 's) of the immediate predecessors of any activity. Finding  $F(Y_n;t)$  by calculating the probabilities of all possible  $n$ -tuples  $(a_1, a_2, \dots, a_n)$  of the activity times is an extreme case of conditioning. Usually, conditioning is done randomly by Monte Carlo sampling and this gives a statistical approximation to  $F(Y_n;t)$ . The methods of [Van 63], [H&W 66], [B&G 71], [Gar 72], and [SPP 79] fall into this class.
2. *Probability Inequalities* —  $F(X_i;t)$  is replaced by a lower or upper bound which is easier to calculate ( the trivial bounds 0 and 1 require no calculation ). The methods of [Dod 85], [Kle 71], [R&T 77], and [Sho 77] are examples in this class.

### 3 CONDITIONING SETS AND ELIMINATING DEPENDENCE.

We have seen that for each node  $i$  in the network we must calculate  $F(X_i;t)$  and  $F(Y_i;t)$ , defined in (11) and (12). The calculation of (12) is the distribution operation of *convolution* and corresponds to the addition of the two independent random variables  $X_i$  and  $A_i$ . Because  $A_i$  is independent of  $X_i$ , the calculation is simple. The calculation of (11) is the distribution operation which corresponds to the *maximum* of the two random variables  $Y_j$  and  $Y_k$ . Because  $Y_j$  and  $Y_k$  are *not* independent, the calculation of (11) is difficult.

The calculation of (11) would be simple if  $Y_j$  and  $Y_k$  were independent. We would then have

$$\begin{aligned}
 F(X_i;t) &= \Pr [\max\{Y_j, Y_k\} \leq t] \\
 &= \Pr [(Y_j \leq t) \text{ and } (Y_k \leq t)] \\
 &= \Pr [Y_j \leq t] \times \Pr [Y_k \leq t] \\
 &= F(Y_j;t) \times F(Y_k;t), \quad x'_i \leq t \leq x''_i.
 \end{aligned} \tag{13}$$

The calculation of (13) is the distribution operation of *multiplication* and corresponds to the maximum of two *independent* random variables. To summarize :

RANDOM VARIABLE OPERATOR	CORRESPONDING DISTRIBUTION OPERATOR
Addition : $Y_i = X_i + A_i$	Convolution : $F(Y_i;t) = \sum_{a_i=a'_i}^{a''_i} F(X_i;t - a_i) f(A_i;a_i), \quad y'_i \leq t \leq y''_i$
Maximum : $X_i = \max\{Y_j, Y_k\}$	Multiplication : $F(X_i;t) = F(Y_j;t) \times F(Y_k;t), \quad x'_i \leq t \leq x''_i$

We now develop a method of conditioning on a subset  $C$  of activities that eliminates the dependence between the  $Y$ 's of the immediate predecessors of any node. This will allow us to calculate a conditional completion time distribution for  $Y_n$ , using the simple distribution operators (13) and (12). If we calculate a conditional completion time distribution for each possible realization of the set of random activity times of  $C$ , we can find the unconditional completion time distribution  $F(Y_n;t)$  using the 'law of total probability'.

**Definition 2.** (Conditioning Set) : A conditioning set of a stochastic PERT network is a subset  $C$  of the nodes  $N$  such that when the random activity times of  $C$  are replaced by constants, the finish times of the immediate predecessors of any node in  $N$  are independent.

This definition does not define a unique conditioning set, as we show below. Also, the definition is not constructive because it gives no method for finding such a subset, except by trial and error. It is obvious however, that  $N$  is a conditioning set of itself. In fact, using this conditioning set gives the following crude method of calculating the completion time distribution,  $F(Y_n; t)$ .

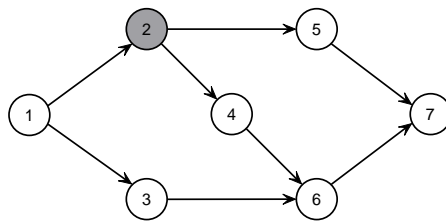
### 3.1 Complete Enumeration

Given a stochastic PERT network with  $n$  activities, each of which can take on one of  $v$  different values with known probabilities : (i) Generate all possible  $n$ -tuples of activity-time values and calculate the corresponding probability of each  $n$ -tuple, and (ii) for each  $n$ -tuple find the completion time of the network using equations (1), (2), and (3). The completion times and their probabilities are summed using the law of ‘total probability’ to obtain the completion time distribution. Complete enumeration requires an amount of computation that is  $O(n^2 v^n)$  and it is obvious that this method is impracticable even for small networks.

If the size of the conditioning set can be reduced from  $n$  to  $m < n$ , then we may substantially reduce the amount of computation required for complete enumeration. This is the motivation behind the *unique arc* method of Burt and Garman [B&G 71], [Gar 72], and the *uniformly directed cutset* method of Sigal, *et al.* [SPP 79], [SPP 80]. These methods identify conditioning sets whose size  $m$  is less than  $n$ . We now propose a very simple method for identifying a conditioning set in any PERT network.

### 3.2 Conditioning Sets and C-Nodes

We wish to identify nodes whose random activity times, when replaced by constants, give a network in which the finish times of the immediate predecessors of any node are statistically independent.



**Figure 2 :** A Simple Network with One Conditioning Activity

Consider the network shown in Figure 2. There are three paths through this network. The lengths of the paths are the random variables

$$\begin{aligned}
 L_1 &= A_1 + A_2 + A_5 + A_7, \\
 L_2 &= A_1 + A_2 + A_4 + A_6 + A_7, \\
 L_3 &= A_1 + A_3 + A_6 + A_7.
 \end{aligned}$$

The completion time of the network is the random variable

$$Y_7 = \max\{L_1, L_2, L_3\} = \max\{(A_1 + A_2 + A_5 + A_7), (A_1 + A_2 + A_4 + A_6 + A_7), (A_1 + A_3 + A_6 + A_7)\} \quad (i)$$

The random variables  $L_1, L_2, L_3$  are dependent because  $A_1, A_2, A_6,$  and  $A_7$  are common to one or more pairs of  $Ls$ . To eliminate statistical dependence in (i) we would need to condition on  $A_1, A_2, A_6,$  and  $A_7$ . Thus, activities 1, 2, 6, and 7 are *conditioning activities* and  $C = \{1, 2, 6, 7\}$  is a conditioning set.

We can reduce the size of the conditioning set by rearranging (i) as follows:

$$Y_7 = \max\{(A_2 + A_5), (A_2 + A_4 + A_6), (A_3 + A_6)\} + A_1 + A_7. \quad (ii)$$

We note that this rearrangement is always possible for any PERT network because the source and sink activities are common to all paths through the network.

We eliminate dependence in (ii) by conditioning on  $A_2$  and  $A_6$ . Thus, activities 2 and 6 are conditioning activities and  $C = \{2, 6\}$  is a conditioning set.

This conditioning set is half the size of (i). We can reduce the size of this conditioning set by rearranging (ii) as follows :

$$Y_7 = \max\{(A_2 + A_5), \max\{(A_2 + A_4), A_3\} + A_6\} + A_1 + A_7. \quad (iii)$$

The conditioning set for (iii) is  $C = \{2\}$  because if we set  $A_2 = a_2$ , a constant, we get

$$Y_7 = \max\{(a_2 + A_5), \max\{(a_2 + A_4), A_3\} + A_6\} + A_1 + A_7,$$

and all random variables involving the max –operator are independent.

We note that conditioning on  $A_6$  is avoided by exploiting the directedness of the network : although node 6 is common to the directed paths  $\{1, 2, 4, 6, 7\}$  and  $\{1, 3, 6, 7\}$ , it affects only the *single* directed sub-path  $\{6, 7\}$ . This idea is the basis for the following definition.

**Definition 3.** (*C–Node*) : Let  $D = (N, A)$  be a PERT network of  $n$  activities. Any node  $i$  is a *C–Node* (*Conditioning Node*) if

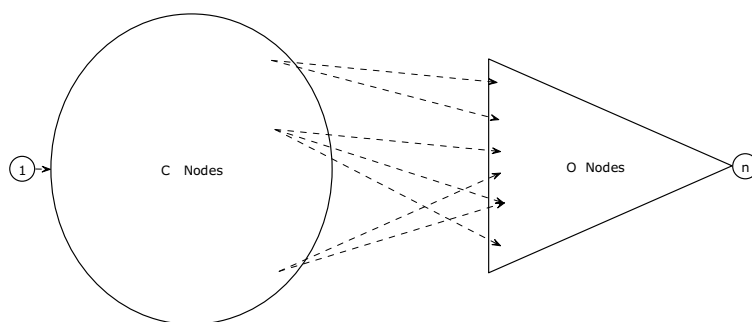
1. Node  $i$  has two or more immediate successors, or
2. Any successor of node  $i$  is a *C–Node*.

If node  $i$  is not a *C–Node* then it is an *O–Node* (*Ordinary Node*). □

This recursive definition partitions  $N$  into two sets  $N_c$  and  $N_o$  with cardinalities  $|N_c| = m$  and  $|N_o| = n - m$  (see Figure 3). We show below that  $N_c$  is a conditioning set. Note that this definition depends only on the topology of the network. This is because the random activity times  $\{A_i\}$  are assumed to be independent. We now give two lemmas and a theorem to prove that the set of *C–nodes*,  $N_c$ , is a conditioning set.<sup>1</sup>

---

<sup>1</sup>The definition above will classify node 1 as a *C–node* in all cases except the simple series network, even though it can always be removed from the conditioning set as shown in equation (ii) above. To avoid the complication of this special case we will always classify node 1 as a *C–node*.



**Figure 3 :** Partition of a Directed Acyclic Graph into  $C$ -nodes and  $O$ -nodes

**Lemma 1.** *A PERT network whose  $n$  nodes are partitioned into  $m$   $C$ -nodes and  $n - m$   $O$ -nodes has the following properties :*

- P1. The sink node and its immediate predecessors are  $O$ -nodes. Hence  $m < n$ .
- P2. All nodes on all paths from the source node to any  $C$ -node are  $C$ -nodes.
- P3. A topological labelling of the network can be found such that the  $C$ -nodes are labelled  $1, \dots, m$ .

**Proof :** An easy application of the  $C$ -node definition. □

**Lemma 2.** *(Tree Property of  $O$ -nodes) : In a PERT network the  $O$ -nodes and their associated arcs form a directed tree  $T$  whose root is the sink node of the network.*

**Proof :** (By contradiction) If  $T$  is not a directed tree then this implies that there exist two or more paths from some node  $k$  in  $T$  to the sink. This implies that node  $k$ , or some successor of node  $k$ , has two or more immediate successors. This implies that some node in  $T$  is a  $C$ -node, which contradicts the hypothesis.

If the sink node is not the root then this implies it has a successor. Again, a contradiction. □

**Theorem 1.** *(Elimination of Dependence) : If the random activity times of the  $C$ -nodes of a stochastic PERT network are replaced by constants, then the random finish times of the immediate predecessors of any node  $i$  are statistically independent, i.e., the set of  $C$ -nodes  $N_c$  is a conditioning set.*

**Proof**

1. By definition and property P2 of Lemma 1, the finish times of all  $C$ -nodes are constants because they are functions of the activity times of  $C$ -nodes only. Hence the finish times of the immediate predecessors of any  $C$ -node are constants and, therefore, statistically independent.
2. The  $O$ -nodes form a directed tree  $T$  by Lemma 2. Consider any  $O$ -node  $i$  in  $T$  and assume, without loss of generality, that it has two immediate predecessors, nodes  $j$  and  $k$ , with random finish times  $Y_j$  and  $Y_k$ . These finish times are functions of their respective sets of predecessor activity times. The only nodes common to these predecessor sets are  $C$ -nodes because of the tree property of  $O$ -nodes. Thus  $Y_j$  and  $Y_k$  are functions of constants and mutually exclusive sets of independent random activity times. Hence  $Y_j$  and  $Y_k$  are statistically independent. □

### 3.3 An Algorithm for Identifying C-nodes

We now give a simple algorithm for identifying  $C$ -nodes. The algorithm applies Definition 3 to each node of the network  $D$ . Its complexity is  $O(n)$  for all PERT networks. In the algorithm a boolean array  $Cnode[1..n]$  is initialized to `False`. At the end of the algorithm if node  $i$  is a  $C$ -node then  $Cnode[i] = \text{True}$ . The algorithm starts at the sink node  $n$  which is an  $O$ -node by definition.

<pre> <b>algorithm</b>   Cnodes (<math>D</math>)   assumes <math>D</math> is topologically labelled  <b>for</b> <math>i := 1</math> <b>to</b> <math>n</math> <b>do</b> <math>Cnode[i] := \text{False}</math> <b>endfor</b> <b>for</b> <math>i := n - 1</math> <b>downto</b> <math>1</math> <b>do</b>   <b>if</b> <math> Succ(i)  &gt; 1</math> <b>then</b>     <math>Cnode[i] := \text{True}</math>   <b>else</b>     <math>j := Succ(i)</math> only 1 successor     <math>Cnode[i] := Cnode[j]</math>   <b>endif</b> <b>endfor</b> <b>endalg</b> Cnodes </pre>
---

Figure 4 shows a network and its  $C$ -nodes (shaded). The properties of Lemmas 1 and 2 obviously hold and it is not too difficult to verify the theorem for this network. This theorem is the basis of the algorithm given in Section 3. The algorithm generates all possible sets of values for the  $C$ -nodes and then calculates, for each set, a conditional completion time distribution using the simple distribution operators (10) and (11). The conditional distributions are then combined, using the ‘law of total probability’, to give the unconditional completion time distribution.

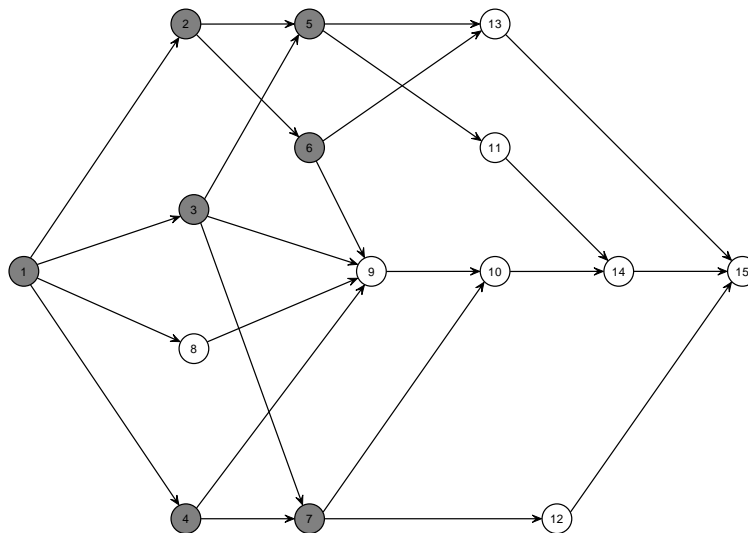


Figure 4 : A 17-Node Network with 8  $C$ -nodes.

Definition 2 and Algorithm *Cnodes* are easily modified to handle ‘activity-on-the-arc’ networks by changing ‘node’ to ‘arc’. For example, the conditioning set of arcs given by the algorithm for the network

in Figure 5 below is  $\{1, 2, 4\}$ . The method of Burt & Garman (1971) gives the set  $\{1, 2, 4, 5, 6, 8\}$ , while the method of Sigal *et al.* (1979) gives the set  $\{1, 5, 6, 8\}$ . This example is taken from Sigal *et al.* (1979), Figure 2.

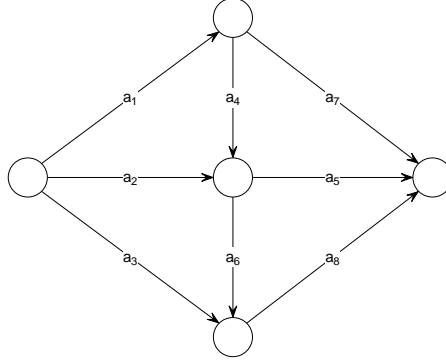


Figure 5 : An “activity-on-the arc” Network. Arcs  $a_1, a_2, a_4$  are  $C$ -arcs.

## 4 AN EXACT STOCHASTIC PERT NETWORK ALGORITHM.

We now define an exact algorithm to calculate the completion time distribution of a stochastic PERT network. Assume the network has  $n$  nodes(activities) and  $m$   $C$ -nodes, and that each  $C$ -node can have  $v$  distinct values for its activity time<sup>2</sup>. Hence there are  $v^m$  different  $m$ -tuples of  $C$ -node values. Let  $C_k$  denote the set of  $C$ -node values for the  $k$ th  $m$ -tuple, i.e.,

$$C_k = \{a_{1k}, a_{2k}, \dots, a_{mk}\}, \quad k = 1, 2, \dots, v^m,$$

where  $a_{ik}$  is the activity time of the  $i^{th}$   $C$ -node in the  $k^{th}$   $m$ -tuple. We have

$$Pr [C_k] = Pr [A_1 = a_{1k}, A_2 = a_{2k}, \dots, A_m = a_{mk}] = \prod_{i=1}^m Pr [A_i = a_{ik}] = \prod_{i=1}^m f(A_i; a_{ik}). \quad (13)$$

Let  $F(X_i; t|C_k)$ ,  $x'_i \leq t \leq x''_i$ , and  $F(Y_i; t|C_k)$ ,  $y'_i \leq t \leq y''_i$ , denote the start and finish time distributions of nodes  $i = 1, \dots, n$ , conditional on the set of  $C$ -node values  $C_k$ .

Let  $F(Y_n; t, k)$ , denote the *partial* unconditional completion-time distribution, defined recursively as follows, where  $y'_n \leq t \leq y''_n$  :

$$\begin{aligned} F(Y_n; t, 0) &\triangleq 0, \\ F(Y_n; t, k) &= F(Y_n; t, k-1) + F(Y_n; t|C_k) Pr[C_k], \quad k = 1, \dots, v^m, \\ F(Y_n; t, v^m) &\triangleq F(Y_n; t). \end{aligned} \quad (14)$$

We now give an informal statement of the algorithm and a small example.

<sup>2</sup>This assumption simplifies the description of the following algorithm and its analysis. If we assume that each activity  $i$  takes on  $v_i$  different values then we would have  $\prod_{i=1}^m v_i$  different  $m$ -tuples.

**algorithm** Informal Exact Completion-Time Distribution

---

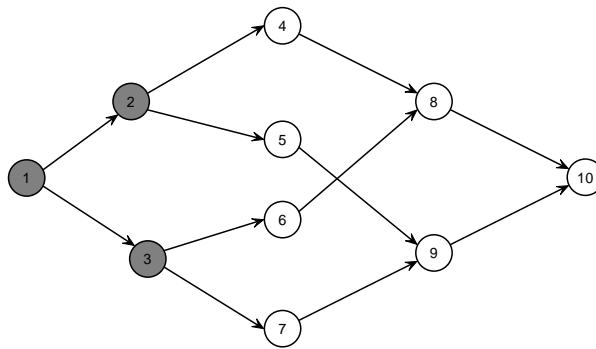
For each  $m$ -tuple of  $C$ -node values  $C_k$ ,  $k = 1, 2, \dots, v^m$ , do steps 1, 2, and 3

1. Set the activity times of the  $C$ -nodes to the values of  $C_k$  and calculate  $\Pr[C_k]$ .
2. Pass through the network and calculate the conditional distribution,  $F(Y_n; t|C_k)$ , using equations (10) and (11).
3. Calculate the partial unconditional completion-time distribution,

$$F(Y_n; t, k) = F(Y_n; t, k - 1) + F(Y_n; t|C_k) \Pr[C_k].$$

**endalg** Informal

**Example 1.** The network in Figure 6 has ten activities, with  $A_1 = 0$ ,  $A_{10} = 0$ , and  $A_i = \{ 1 \text{ or } 2 \text{ with prob. } 0.5 \}$ ,  $i = 2, 3, \dots, 9$ .



**Figure 6 :** A 10-activity Network with 2  $C$ -nodes

This network has 2  $C$ -nodes (2 and 3) each with two values (we ignore node 1). Hence the parameters for this problem are:  $n = 10, m = 2, v = 2, v^m = 4$ . The four enumerations are shown in Table 1.

Table 1: Enumerations.

Enumeration	$A_2$	$A_3$	$\Pr[A_2 \text{ and } A_3]$
$C_1$	1	1	$0.5 \times 0.5 = 0.25$
$C_2$	1	2	$0.5 \times 0.5 = 0.25$
$C_3$	2	1	$0.5 \times 0.5 = 0.25$
$C_4$	2	2	$0.5 \times 0.5 = 0.25$

The conditional completion time distributions  $F(Y_{10}; t|C_k)$  are calculated using the multiplication and convolution operations (10) and (11) at each node. This gives Table 2.

Each of these distributions is multiplied by the probability of its corresponding enumeration and accumulated as shown in the algorithm. This gives Table 3, where  $F(Y_{10}; t, 4) \equiv F(Y_{10}; t)$ , the completion time distribution.

Table 2: Conditional Distributions.

$t$	3	4	5	6
$F(Y_{10};t C_1)$	0.015625	0.390625	1.000000	1.000000
$F(Y_{10};t C_2)$	0.000000	0.062500	0.562500	1.000000
$F(Y_{10};t C_3)$	0.000000	0.062500	0.562500	1.000000
$F(Y_{10};t C_4)$	0.000000	0.015625	0.390625	1.000000

Table 3: Partial Distributions.

$t$	3	4	5	6
$F(Y_{10};t, 1)$	0.003906	0.097656	0.250000	0.250000
$F(Y_{10};t, 2)$	0.003906	0.113281	0.390625	0.500000
$F(Y_{10};t, 3)$	0.003906	0.128906	0.531250	0.750000
$F(Y_{10};t, 4)$	0.003906	0.132812	0.628906	1.000000

The formal algorithm is shown on the next page.

#### 4.1 Analysis of the Exact Algorithm

The algorithm has three main loops indexed by the variables  $k$  (enumerations),  $i$  (nodes) and  $j$  (predecessors). The upper bounds on these are  $v^m$ ,  $n$ , and  $i$ , respectively. Hence, for each enumeration  $k$ , the algorithm performs  $O(n^2)$  multiplication operations and  $O(n)$  convolution operations. The enumeration loop is performed  $v^m$  times. Hence the time complexity of the algorithm is  $O(n^2v^m)$ , assuming, for convenience, that the multiplication and convolution operations can be performed in a constant time. Recalling that  $m$  is the number of  $C$ -nodes and that the time complexity of complete enumeration is  $O(n^2v^n)$ , we see that the new algorithm always does better (asymptotically) than complete enumeration because  $m < n$  (by P1, Lemma 1). Nevertheless the new algorithm's time is not bounded by a polynomial in  $n$  and  $v$  and, therefore, does not solve the problem in polynomial time. This is to be expected because the Stochastic Pert Problem is  $\#P$ -Complete (see Hag88).

We note before passing on, that we may assume any polynomial-time complexity for the distribution operations : the complexity of this algorithm is still exponential because the outer ( $k$ ) loop is  $O(v^m)$ . This is, of course, in keeping with the fact that the algorithm is solving a  $\#P$ -Complete problem. The assumption of  $O(1)$  for the distribution operations simplifies the analysis and emphasizes the source of the exponential complexity. Ironically, the most tedious parts of the *implementation* of the algorithm are the distribution operations.

<b>algorithm</b> Exact Completion-Time Distribution	
Initialize	
Set $F(Y_n; t, 0) := 0$ , for all $t \geq 0$	
Set $y'_1 := a'_1$ , $y''_1 := a''_1$	
Set $F(Y_1; t) := F(A_1; t)$ , for all $t \in [y'_1, y''_1]$ .	
Main Loop	
<b>for</b> $k := 1, 2, \dots, v^m$ <b>do</b>	
Generate the $k^{\text{th}}$ $m$ -tuple $C_k$	
Set the activity times of nodes $1, 2, \dots, m$ , to the values of $C_k$	
Calculate $\Pr[C_k]$	
<b>for</b> $i := 2, 3, \dots, n$ <b>do</b> Calculate $F(Y_n; t C_k)$	
Set $x'_i = 0$ ; $x''_i = 0$	
Set $F(X_i; t C_k) = 1$ , for all $t \in [x'_i, x''_i]$	
<b>for</b> each $j \in \text{Pred}(i)$ <b>do</b>	
$x'_i := \max\{x'_i, y'_j\}$	
$x''_i := \max\{x''_i, y''_j\}$	
<b>for</b> each $t \in [x'_i, x''_i]$ <b>do</b>	
$F(X_i; t C_k) := F(X_i; t C_k) \times F(Y_j; t C_k)$	Multiplication
<b>endfor</b> $t$ -loop	
<b>endfor</b> $j$ -loop	
$y'_i := x'_i + a'_i$	
$y''_i := x''_i + a''_i$	
<b>for</b> each $t \in [y'_i, y''_i]$ <b>do</b>	
$F(Y_i; t C_k) := \sum_{a_i=a'_i}^{a''_i} [F(X_i; t - a_i C_k) f(A_i; a_i)]$	Convolution
<b>endfor</b> $t$ -loop	
<b>endfor</b> $i$ -loop	
<b>for</b> each $t \in [y'_n, y''_n]$ <b>do</b>	
$F(Y_n; t, k) := F(Y_n; t, k-1) + F(Y_n; t C_k) \Pr[C_k]$	Accumulation
<b>endfor</b> $t$ -loop	
<b>endfor</b> $k$ -loop	
<b>endalg</b> Exact	

We note that in the implementation of this algorithm if node  $i$  is a  $C$ -node then the multiplication and convolution operations are not necessary because the finish times of the immediate predecessors of node  $i$  are constants.

## 5 MONTE CARLO APPROXIMATION ALGORITHMS

The time complexity of the exact algorithm is exponential and so it is useful only for small or special networks. The algorithm is exponential because the outer loop is performed  $v^m$  times, where  $m$  is the number of conditioning nodes. If  $m$  were bounded above by a fixed number for all networks then we would have a polynomial algorithm. This is not the case and it is obvious that, in general, the number of conditioning nodes,  $m$ , increases with  $n$ , the total number of nodes in the network.

The exponential complexity of the exact algorithm forces us to consider approximation algorithms which require less computation. We now show that with simple modifications, the exact algorithm can be used to calculate Monte Carlo approximations for the completion time distribution.

### 5.1 Simple Monte Carlo Approximations

Monte Carlo sampling has been used with some success to find approximate solutions to the stochastic PERT problem. Good discussions of the method can be found in Burt & Garman [B&G 71b] and Elmaghraby, Chapter 6 [Elm 77].

The *Simple Monte Carlo Method* of calculating an approximation or estimate of the completion time distribution is as follows :

1. Set the activity time of each node in the network to a value chosen at random from the node's activity time distribution.
2. Calculate the network completion time  $Y_n$  for this set of  $n$  activity times.
3. Repeat  $N$  times, steps 1 and 2 , counting the number of times,  $N_t$ , that  $Y_n \leq t$ .

Then the simple Monte Carlo approximation for the completion time distribution is  $\hat{F}_S(Y_n; t) = N_t/N$ .

This method (called *Crude Monte Carlo*, see Hammersley and Handscomb, 1964, pages 51–55) is more efficient than the *Hit-or-Miss Monte Carlo* used by some researchers.

We can obtain the simple Monte Carlo method from the exact algorithm by the modifications shown in the algorithm below. The random  $n$ -tuple  $C_k = \{a_{1k}, a_{2k}, \dots, a_{nk}\}$  is generated by sampling independently from each of the  $n$  activity time distributions. No calculation of  $\Pr[C_k]$  is necessary.

We note that because  $m = n$  all nodes are  $C$ - nodes and hence no distribution calculations are really necessary. In practice a much simpler algorithm would be used instead the modified exact algorithm. Nonetheless, the analysis of Algorithm 2 given below holds for any Simple Monte Carlo algorithm.

The analysis of the Simple Monte Carlo algorithm is the same as the Exact algorithm except that the number of iterations in the outer ( $k$ ) loop is reduced from  $v^m$  to  $N$ , the sample size. Hence the time complexity of the Simple Monte Carlo algorithm is  $O(n^2N)$ .

This reduction in complexity comes at the cost of uncertainty in the calculated completion time distribution. This uncertainty is due to the sampling error or variance of the estimate  $\hat{F}_S(Y_n; t)$  and can be reduced by increasing the sample size  $N$ . Unfortunately the sample size  $N$  may need to be very large to obtain good accuracy and so a great deal of effort has been spent on finding *variance reduction* techniques that reduce the sampling error without increasing the sample size.

```

algorithm Simple Monte Carlo


---


Set  $m := n$  and regard all nodes in the network as  $C$ -nodes
Initialize --
    Same as Exact Algorithm
for  $k := 1$  to  $N$  do
    Generate a random  $m$ -tuple  $C_k$ 
    Same as Exact Algorithm
     $F(Y_n; t, k) := F(Y_n; t, k - 1) + F(Y_n; t | C_k)$  Accumulation
endfor  $k$ -loop
for each  $t \in [y'_n, y''_n]$  do
     $\hat{F}_S(Y_n; t) := F(Y_n; t, N) / N$  Averaging
endfor
endalg Simple Monte-Carlo

```

## 5.2 Conditional Monte Carlo Approximations

The Conditional Monte Carlo method is a generalization of the Simple Monte Carlo method and is based on the following principle :

A General Principle of Monte Carlo : if, at any point of a Monte Carlo calculation, we can replace an estimate by an exact value, we shall reduce the sampling error of the final result. (Hammersley & Handscomb, 1964, page 54)

We apply this principle as follows : instead of randomly sampling from each of the  $n$  activity-time distributions we sample from a subset of size  $m < n$ . The subset is chosen so that for each sample a conditional completion time distribution can be calculated exactly, either analytically or numerically, and hence without sampling error. These conditional distributions are then averaged over a sample of size  $N$  to give an estimate of the completion time distribution. The sampling error or variance is reduced because less sampling is done and more of the network is calculated exactly. Indeed, if it were possible to find a sampling subset of size 0 then we would obtain 100% accuracy with no sampling.

The crucial part of the Conditional Monte Carlo method for the Stochastic PERT problem is the identification of the subset of activities whose distributions will be sampled. Burt [Bur 72] gives a heuristic algorithm for identifying *unique* and *non-unique* activities. The non-unique activities are used for sampling. Sigal *et al.*, [SPP 79] use their *uniformly directed cutset* method to identify the sampling activities. They claim that their method gives the smallest possible set of sampling activities. This claim is false (Figure 5 above is a counter-example).

The set of sampling activities we use for Conditional Monte Carlo is the conditioning set identified by the simple  $C$ -node definition given in Section 2 above. The sets of sampling activities identified by this method are smaller than those given in [Bur 72] and [SPP 79] but there is no guarantee that it will give

the smallest sampling (conditioning) set. This was shown by a counter-example given by Elmaghraby [Elm 81].

We obtain the Conditional Monte Carlo algorithm from the exact algorithm by using the same modification that was used for the Simple Monte Carlo algorithm, but *without* setting  $m := n$ . In other words, the Conditional Monte Carlo algorithm is obtained from the exact algorithm by taking a random sample of all possible  $m$ -tuples of  $C$ -node values, rather than exhaustively generating all  $m$ -tuples. The time complexity of Conditional Monte Carlo is the same as Simple Monte Carlo, i.e.,  $O(n^2N)$ , assuming the complexity of each distribution operation is  $O(1)$ .

### 5.3 The Kleindorfer Bounds

In the testing of the exact and Monte Carlo algorithms below it is useful to have an easily-calculated check on the results. This is provided by the Kleindorfer bounds [Kle 71]. Kleindorfer showed, among other things, that the start time distribution of any activity (see equation (8) above) could be bounded as follows :

Let the Kleindorfer lower and upper bounds on  $F(\cdot; t)$  be  $F_K^l(\cdot; t)$  and  $F_K^u(\cdot; t)$  respectively. Then

$$F_K^l(X_i; t) = F_K^l(Y_j; t) \times F_K^l(Y_k; t), \quad x_i' \leq t \leq x_i'' \tag{15}$$

$$F_K^u(X_i; t) = \min[F_K^u(Y_j; t), F_K^u(Y_k; t)], \quad x_i' \leq t \leq x_i'' \tag{16}$$

$$F_K^l(Y_i; t) = \sum_{a_i=a_i'}^{a_i''} F_K^l(X_i; t - a_i) f(A_i; a_i), \quad y_i' \leq t \leq y_i'' \tag{17}$$

$$F_K^u(Y_i; t) = \sum_{a_i=a_i'}^{a_i''} F_K^u(X_i; t - a_i) f(A_i; a_i), \quad y_i' \leq t \leq y_i'' \tag{18}$$

It should be noted that neither bound requires  $Y_j$  or  $Y_k$  to be independent. If they are independent then the lower bound is exact.

It can be seen that the lower bounds in (15) and (17) are the same as the expressions for the multiplication and convolution operations in (11) and (12) above . Hence, we can get Kleindorfer's lower bound from the exact algorithm by setting the number of  $C$ -nodes equal to zero, i.e.,  $m := 0$ . We get the upper bounds by adding (16) and (18) to the exact algorithm.

The analysis of Kleindorfer's algorithm is obtained from the exact algorithm by setting  $m := 0$ . This gives a time complexity of  $O(n^2v^m) = O(n^2)$ , assuming the complexity of each distribution operation is  $O(1)$ . Thus the calculation of the Kleindorfer bounds is a trivial part of the main calculations in the exact or Monte Carlo algorithms.

We complete this section with a summary of the time complexities of the algorithms discussed above.

Table 4: Complexities of Stochastic PERT Algorithms.

ALGORITHM	TYPE OF DISTRIBUTION	COMPLEXITY	COMMENTS
Complete Enumeration	Exact	$O(n^2v^n)$	$m = n$
Exact Algorithm	Exact	$O(n^2v^m)$	$m < n$
Kleindorfer	Lower & Upper Bounds	$O(n^2)$	Poor Upper Bound
Simple Monte Carlo	Approximate	$O(n^2N)$	Large Variance
Conditional Monte Carlo	Approximate	$O(n^2N)$	Reduced Variance

## 6 IMPLEMENTATION, TESTING, AND RESULTS

### 6.1 Implementation

The exact algorithm was implemented in standard FORTRAN 77 and consists of three parts : (i) Network construction, initialization, flagging of  $C$ -nodes, and topological labelling; (ii) Distribution calculations; (iii) Statistical calculations and summary results. The same program, with appropriate modifications, was used to calculate the various Monte Carlo approximations and the Kleindorfer bounds. The flagging (identification) of  $C$ -nodes is straightforward and required no special programming.

Because the main loop of the exact algorithm is exponential some care was taken to reduce unnecessary calculations. A substantial reduction in calculations can be achieved by generating the  $m$ -tuples  $C_k$  (line 6, Algorithm 1) so that only one value in  $C_k = \{a_{1k}, a_{2k}, \dots, a_{mk}\}$  changes in each iteration of the outer loop. This means that only those nodes reachable from the  $C$ -node with the changed value are involved in the calculations of the inner loops (lines 9 and 12, Algorithm 1). This modification reduced the running time for the 24-node network (Appendix A) by a factor of 10. The problem of  $m$ -tuple generation is part of the general problem of minimum change algorithms for generating combinatorial objects (see [RND 77] and [OCo 82]).

Implementing the Simple and Conditional Monte Carlo methods, and the Kleindorfer bounds require relatively minor modifications to the exact algorithm as we have shown in Sections 5.1, 5.2, and 5.3.

### 6.2 Testing

Four networks were used to test the algorithm. There are complete descriptions of these networks in Appendix 1. Table 5 gives summary details of these networks.

Table 5: Test Networks.

NETWORK	NODES	$C$ -NODES	SOURCE
NET10	10	3	[Kle 71]
NET16	16	9	[O'Co 81]
NET24	24	10	[O'Co 81]
NET40	40	22	[Kle 71]

The exact algorithm was used to calculate exact distributions for the first three networks. The exact distribution of NET40 could not be calculated in a reasonable amount of time. The exact algorithm with simple modifications was used to calculate unconditional (simple) and conditional Monte Carlo approximations for all networks. Samples of size  $N = 25, 100$  were taken for each network. These samplings were repeated 5 times with different random number generator seeds and the results averaged over the 5 repetitions. The random number generator used was URAND, given in Forsythe, *et al.* [FMM 77] and the 5 seeds used were 12345, 180445, 101010, 654321, 544081. For comparison the Kleindorfer bounds were calculated for each network, using the exact algorithm with  $m$  set to 0.

The execution times for the Exact and Conditional Monte Carlo algorithms are shown in Table 6. These times were obtained on a Dell Workstation 400 with a Pentium II 300 MHz processor and the compiler was Digital Visual Fortran V5.

Table 6: Execution Times (secs) Pentium II 300MHz.

Network	Enumerations	Exact	Conditional Monte Carlo	
			$N = 25$	$N = 100$
NET10	75	0.90	0.43	1.04
NET16	18,777	9.61	0.60	1.61
NET24	66,348	28.67	0.77	2.12
NET40	$10^{12}$	—	1.74	5.14

The exact distribution for Kleindorfer's 40-node network was not attempted because it has 22  $C$ -nodes and would require approximately  $10^{12}$  enumerations.

### 6.3 Results

The complete results of all experiments are given in Appendix 2. The Exact algorithm was tested on NET10, 16, & 24 only, because of its exponential complexity. Nonetheless the results provide a useful set of benchmarks for other methods.

It can be seen from the results that the Kleindorfer lower bound is quite good but the upper bound is poor. As Kleindorfer pointed out in [Kle 71], the upper bound rarely equals the exact distribution whereas the lower bound is exact for certain types of networks.

The results of the Monte Carlo experiments are best summarized by their *variance reduction ratios*, *i.e.*, the ratio of the sample variances of the Unconditional Monte Carlo (UMC) to the Conditional Monte Carlo method (CMC), or any other variance reduction method. Table 7 gives a summary of the average variance reduction ratios for the modified exact algorithm. From this table we draw the following *tentative* conclusions for Stochastic PERT networks :

- Conditional Monte Carlo can give large variance reduction ratios.
- Sample size has no effect on the variance reduction ratios.
- Network size has no effect on the variance reduction ratios.
- The percentage of  $C$ -nodes has no effect on the variance reduction ratios.

Table 7: Variance Reduction Ratios (UMC/CMC).

Network	Sample Size	
	25	100
NET10	5.25	5.25
NET16	8.50	8.50
NET24	4.76	4.38
NET40	7.19	6.58

## 7 DISCUSSION

The main contribution of his paper has been a new definition of *Conditioning Activities* in PERT networks, which were then used in a simple enumerative algorithm. This algorithm can be used to calculate exact, approximate, or bounding distributions (discussed below) for the completion time of PERT networks. The simplicity of the algorithm is obvious from this skeleton version :

<b>algorithm</b> Exact Completion-Time Distribution	
Initialize. $m = \text{No. } C\text{-nodes}$	
<b>for</b> $k := 1, 2, \dots, v^m$ <b>do</b> Enumeration	
Generate the $k^{\text{th}}$ $m$ -tuple $C_k$	
<b>for</b> $i := 2, 3, \dots, n$ <b>do</b> Calculate $F(Y_n; t   C_k)$	
<b>for</b> each $j \in \text{Pred}(i)$ <b>do</b>	
Multiplication	$F(X_i; t   C_k) := F(X_i; t   C_k) \times F(Y_j; t   C_k)$
<b>endfor</b> $j$ -loop	
Convolution	$F(Y_i; t   C_k) := \sum_{a_i=a_i'}^{a_i''} [F(X_i; t - a_i   C_k) f(A_i; a_i)]$
<b>endfor</b> $i$ -loop	
Accumulation	$F(Y_n; t, k) := F(Y_n; t, k - 1) + F(Y_n; t   C_k) \text{Pr}[C_k]$
<b>endfor</b> $k$ -loop	
<b>endalg</b> Exact	

Although this algorithm is easy to state, it is tedious to implement because of the crucial need to avoid unnecessary computations. Once implemented however, it is easily modified to calculate bounding and approximate (Monte Carlo) distributions. The implementation is based on this modular template below, where all messy details have been abstracted away.

<b>algorithm</b> General Modular Version	
<b>for</b> each $C_k \in C$ <b>do</b>	Enumeration
$F(Y_n; t   C_k) := \text{ConDist}(C_k)$	Conditional Distribution
$F(Y_n; t, k) := F(Y_n; t, k - 1) + F(Y_n; t   C_k) \text{Pr}[C_k]$	Accumulation
<b>endfor</b> $k$ -loop	
<b>endalg</b> General	

This modular version allows us to see that the only difference between all the methods discussed above is the interpretation of the main loop statement ‘**for** each  $C_k \in C$  **do**’. The statements within the body of this loop are essentially the same for all methods. This means that we can derive any method by an appropriate change to the main enumeration loop which generates each  $C_k \in C$ . These are shown in Table 8.

Table 8: Conditioning Set  $C$  for each Algorithm

Method	Conditioning Set $C$	Size of $C$	Comment
Complete Enumeration	All $n$ -tuples	$v^n$	Exact
$C$ -node Enumeration	All $m$ -tuples	$v^m$	Exact
Simple Monte Carlo	Random set of $n$ -tuples	$N_{\text{samp}}$	Approximate
Conditional Monte Carlo	Random set of $m$ -tuples	$N_{\text{samp}}$	Approximate
Kleindorfer	Empty	0	Bounding

## 7.1 Exact Distributions

The Stochastic PERT Problem is  $\#P$ -Complete and this suggests that enumeration is the only way to solve it exactly. Complete enumeration has always been available as the method of last resort. The partitioning of the PERT graph into  $C$ -nodes and  $O$ -nodes reduces the burden of complete enumeration because only  $C$ -nodes need to be enumerated (conditioned on). However, as the network size increases, so does the number of  $C$ -nodes and we still have exponential growth of computing time.

## 7.2 Approximate Distributions

Conditioning activities have been used by other authors for the same purpose, *i.e.*, to reduce computation time and variance. The identification of conditioning activities using *Definition 3* is simple whereas the definitions given by other authors ([B&G 71b], [SPS 79]) require substantial algorithms. Also it seems that the size of the sets of conditioning activities identified by *Definition 3* are usually smaller than those identified by other methods published to date. There is no guarantee that it gives a minimum conditioning set. The problem of identifying a minimum conditioning set of activities has neither been solved nor adequately characterized.

Variance reduction by Conditional Monte Carlo seems to give good results for Stochastic PERT networks. However, the small number and size of problems published here and elsewhere do not allow really firm

conclusions to be reached. Thorough empirical testing of the method on a large set of standard problems is necessary. A good problem generator would be very useful in this regard.

Combining conditioning with other variance reduction methods, such as antithetic variables, control variables, importance sampling, etc., may be useful but it is difficult to see how these methods could be ‘tailored’ to the PERT problem. For example, what part of the finish time distribution should importance sampling concentrate on? It is not difficult to see how activity node conditioning reduces both variance and computation because it depends on the structure of the network only and not on interactions between random variables. Perhaps this is why Hammersley and Handscomb devote an entire chapter to Conditional Monte Carlo, yet devote just one page to each of the other methods.

### 7.3 Lower Bounding Distributions

We have seen that the Kleindorfer lower bounds in (15) and (17) in Section 5.3 are the same as the expressions for the multiplication and convolution operations in (11) and (12) above. Hence, we can get Kleindorfer’s lower bound  $F'(Y_n; t)$  from the exact algorithm by setting the number of  $C$ -nodes equal to zero, i.e.,  $m := 0$ .

This suggests that the algorithm can be used to calculate lower bounds on the exact distribution by setting  $m$  to be less than the actual number of conditioning activities  $m_e$  ( $m = 0$  gives the Kleindorfer lower bounds;  $m = m_e$  gives the exact distribution). Preliminary testing of this idea on the networks above shows that it does give lower bounds for the exact distribution. That is, the sequence of conditioning set sizes  $m = 0 < m_1 < m_2 < \dots < m_e$ , gives a sequence of lower bounds

$$F'(Y_n; t) = F^{(0)}(Y_n; t), F^{(m_1)}(Y_n; t), F^{(m_2)}(Y_n; t), \dots, F^{(m_e)}(Y_n; t) = F(Y_n; t), \quad (19)$$

where each  $F^{(m)}(Y_n; t) \leq F(Y_n; t)$

Unfortunately, the sequence of lower bounding distributions is not monotone increasing and so we may have  $F^{(m)}(Y_n; t) > F^{(m+1)}(Y_n; t)$ . This means we have done extra work and obtained a poorer bound.

A slight variation of this idea is to choose increasing-size subsets of the  $C$ -nodes rather than the first 2, 3, etc. This raises the possibility that some subsets give better bounds than others of the same size, and that there is a sequence of increasing-size subsets that gives a sequence of bounding distributions that is monotone increasing. This would allow us to stop the calculations once a bounding distribution has ceased to increase by a pre-set tolerance.

### 7.4 Other Topics

Many papers on the Stochastic PERT Problem discuss (1) reducing networks by series-parallel transformations, and (2) continuous activity-time distributions. We have not discussed network reduction because ultimately, we must deal with irreducible networks.

We have not considered continuous distributions because it is easier to see the combinatorial nature of the problem when using discrete random variables. Using the exact algorithm with continuous distributions poses the obvious question : how do we represent a continuous distribution?

## 8 REFERENCES

1. [B&G 71] Burt, J.M., and Garman, M.B. : “Conditional Monte Carlo: A Simulation Technique for Stochastic Network Analysis”, *Management Science* **18**, No. 3, 207–217, (1971).
2. [B&G 71b] Burt, J.M., and Garman, M.B. : “ Monte Carlo Techniques for Stochastic PERT Network Analysis”, *INFOR* **9**, No.3, 248–262, (1971).
3. [CCT 64] Charnes A., Cooper, W.W., and Thompson,G.L. : “Critical Path Analysis via Chance Constrained Stochastic Programming”, *Operations Research* **12**, 460–470, (1964).
4. [Dod 85] Dodin, B. : “Bounding the Project Completion Time Distribution in PERT Networks”, *Operations Research*, **23**, No. 4, 862–881, (1985).
5. [Elm 77] Elmaghraby, S.E. : *Activity Networks : Project Planning and Control by Network Models*, Wiley, (1977).
6. [Elm 81] Elmaghraby, S.E. : Private Communication, July, 1981.
7. [FMM 77] Forsythe, G.E., Malcolm, M.A., and Moler, C.B. : *Computer Methods for Mathematical Computation* , Prentice-Hall, (1977).
8. [Gar 72] Garman, M.B. : “More on Conditioned Sampling in the Simulation of Stochastic Networks”, *Management Science* **19**, No.1, 90–95, (1972).
9. [Hag 88] Hagstrom, J.N. : “Computational Complexity of PERT Problems”, *Networks*, **18**, 139-147, (1988).
10. [H&H 64] Hammersley, J.M., and Handscomb, D.C. : *Monte Carlo Methods*, Methuen, (1964).
11. [H&W 66] Hartley, H.O., and Wortham, A.. : “A Statistical Theory for PERT Critical Path Analysis”, *Management Science* **12**, 469–481, (1966).
12. [Kle 71] Kleindorfer, G.B. : “Bounding Distributions for a Stochastic Acyclic Networks”, *Operations Research* **19**, 1586–1601, (1971).
13. [Mar 65] Martin, J.J. : “Distribution of Time Through a Directed Acyclic Network”, *Operations Research* **13**, 46–66, (1965).
14. [OCo 81] O’Connor, D.R. : “Exact Distributions of Stochastic PERT Networks”, *Tech. Rep. MIS-81-1*, University College, Dublin, (1981).
15. [OCo 82] O’Connor, D.R. : “A Minimum-Change Algorithm for Generating Lattice Points”, *Tech. Rep. MIS-82-1*, University College, Dublin, (1982).  
(See [www.derekroconnor.net/PERT/Alglat2006.pdf](http://www.derekroconnor.net/PERT/Alglat2006.pdf) for revised version.

16. [RND 77] Reingold, E.M., Nievergelt, J., and Deo, N. : *Combinatorial Algorithms : Theory & Practice*, Prentice-Hall, (1977).
17. [R&T 77] Robillard, P. and Trahan, M. : "The Completion Time of PERT Networks", *Operations Research* **25**, 15–29, (1977).
18. [Sho 77] Shogan, A.W. : "Bounding Distributions for a Stochastic PERT Network", *Networks* **7**, 359–381, (1977).
19. [SPS 79] Sigal, C.E, Pritsker, A.A.B., and Solberg, J.J. : "The Use of Cutset in the Monte Carlo Analysis of Stochastic Networks", *Mathematics and Computers in Simulation* **XXI**, 376–384, (1979).
20. [SPS 80] Sigal, C.E, Pritsker, A.A.B., and Solberg, J.J. : "The Stochastic Shortest Route Problem", *Operations Research* **30**, 1122–1129, (1980).
21. [Van 63] Van Slyke, R.M. : "Monte Carlo Methods and the PERT Problem", *Operations Research* **11**, 839–860, (1963).

## 9 APPENDIX A — Test Networks

This appendix contains the six networks used in testing the algorithms. The activity-time distributions are all discrete and non-negative and are either rectangular or triangular with parameters (L, U) and (L, M, U) respectively.

**TABLE A.1**—10 NODE NETWORK WITH 2 C-NODES

node	dist	L	M	U	succs	
1	Rect	0	-	0	2	3
2	Rect	1	-	5	4	5
3	Rect	1	-	5	6	7
4	Rect	1	-	5	8	
5	Rect	1	-	5	9	
6	Rect	1	-	5	8	
7	Rect	1	-	5	9	
8	Rect	1	-	5	10	
9	Rect	1	-	5	10	
10	Rect	1	-	1	-	

**TABLE A.2**—16 NODE NETWORK WITH 9 C-NODES

node	dist	L	M	U	succs	
1	Rect	1	-	1	2	3
2	Rect	2	-	5	4	10
3	Rect	3	-	6	6	7 10
4	Rect	1	-	2	5	6
5	Rect	3	-	5	9	12
6	Rect	4	-	8	8	
7	Rect	3	-	5	8	11
8	Rect	1	-	2	9	14
9	Rect	3	-	4	13	15
10	Rect	8	-	12	11	
11	Rect	2	-	7	15	
12	Rect	4	-	7	13	
13	Rect	5	-	6	16	
14	Rect	10	-	11	16	
15	Rect	8	-	10	16	
16	Rect	1	-	1	-	

TABLE A.3—24 NODE NETWORK WITH 10 C-NODES

node	dist	L	M	U	succs	node	dist	L	M	U	succs
1	Rect	0	-	0	2 3 4	13	Rect	10	-	15	19
2	Rect	2	-	5	5 11	14	Rect	9	-	13	16
3	Rect	6	-	9	6 7 14	15	Rect	12	-	15	16
4	Tria	1	2	4	8 15	16	Rect	4	-	6	21
5	Rect	10	-	12	9 12 13	17	Rect	15	-	18	23
6	Rect	3	-	6	9	18	Tria	13	17	20	23
7	Rect	10	-	11	10	19	Rect	3	-	7	24
8	Tria	5	8	9	10 17 18	20	Tria	5	7	11	24
9	Rect	11	-	12	19 20 21	21	Rect	3	-	6	24
10	Rect	5	-	7	21 22 23	22	Rect	7	-	10	24
11	Rect	10	-	15	16	23	Rect	6	-	10	24
12	Rect	9	-	13	19	24	Rect	1	-	1	-

TABLE A.4—40 NODE NETWORK WITH 22 C-NODES

node	dist	L	M	U	succs	node	dist	L	M	U	succs
1	Rect	1	-	1	2 3	21	Rect	14	-	15	35 38
2	Tria	9	10	11	4 5 6	22	Rect	1	-	2	35 38
3	Tria	1	2	4	10 11	23	Tria	1	2	4	26
4	Rect	2	-	6	12	24	Tria	2	3	4	26
5	Rect	12	-	13	18 21 24	25	Tria	1	2	3	26
6	Rect	1	-	2	7 8 9 33	26	Tria	1	3	4	32
7	Tria	1	3	4	12	27	Tria	7	19	31	36
8	Tria	3	4	5	13 14 28	28	Tria	4	5	7	32
9	Rect	1	-	4	15 23 30	29	Tria	1	8	15	37
10	Tria	15	17	18	15 23 30	30	Tria	1	5	6	32
11	Rect	2	-	24	25 27	31	Rect	1	-	3	36
12	Rect	1	-	3	16 17	32	Rect	2	-	5	37
13	Rect	1	-	3	18 21 24	33	Tria	8	10	12	37
14	Tria	4	5	7	19 29 31	34	Rect	3	-	5	39
15	Tria	5	11	17	19 29 31	35	Rect	1	-	2	39
16	Rect	4	-	7	20	36	Rect	3	-	11	39
17	Rect	3	-	5	19 29 31	37	Rect	5	-	21	40
18	Rect	2	-	4	20	38	Rect	13	-	14	40
19	Rect	3	-	5	22 34	39	Rect	1	-	19	40
20	Tria	1	2	4	22 34	40	Rect	1	-	1	-

## 10 APPENDIX B— Summary Results

This appendix contains the results of all tests performed on the four networks. The column headings in the tables are :

1. klower, kupper : Kleindorfer lower and upper bound distributions.
2. umc, cmc : Unconditional (Simple) and Conditional Monte Carlo approximate distributions.
3. vumc, vcmc : Variances of Unconditional (Simple) and Conditional Monte Carlo approximate distributions.
4. N : Sample size.

**TABLE B.1**—10 NODE NETWORK WITH 3 C-NODES

time	exact	klower	kupper	N = 25				N = 100			
				umc	cmc	vumc	vcmc	umc	cmc	vumc	vcmc
4	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
5	0.00009	0.00001	0.03200	0.00000	0.00000	0.00000	0.00000	0.00000	0.00010	0.00000	0.00000
6	0.00096	0.00022	0.08000	0.00000	0.00102	0.00000	0.00000	0.00400	0.00107	0.00004	0.00000
7	0.00602	0.00218	0.16000	0.00800	0.00611	0.00032	0.00001	0.00600	0.00647	0.00006	0.00000
8	0.02654	0.01409	0.28000	0.03200	0.02636	0.00128	0.00009	0.02200	0.02785	0.00022	0.00002
9	0.08017	0.05472	0.42400	0.09600	0.07821	0.00352	0.00044	0.06600	0.08214	0.00062	0.00011
10	0.18496	0.14893	0.57600	0.18400	0.17873	0.00587	0.00121	0.16800	0.18670	0.00140	0.00031
11	0.34822	0.31217	0.72000	0.32000	0.33913	0.00797	0.00205	0.36200	0.34962	0.00231	0.00052
12	0.55249	0.52812	0.84000	0.55200	0.54206	0.00955	0.00213	0.55600	0.55316	0.00247	0.00054
13	0.74235	0.73055	0.92000	0.71200	0.73295	0.00848	0.00133	0.73800	0.74209	0.00194	0.00033
14	0.88571	0.88210	0.96800	0.92000	0.88051	0.00299	0.00048	0.88800	0.88562	0.00098	0.00012
15	0.96936	0.96889	0.99200	0.97600	0.96922	0.00096	0.00007	0.97600	0.96991	0.00024	0.00002
16	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000	0.00000	1.00000	1.00000	0.00000	0.00000
mean	12.20310	12.35800	10.00000	12.20000	12.24558	0.00315	0.00060	12.21400	12.19526	0.00079	0.00015

**TABLE B.2**—16 NODE NETWORK WITH 9 C-NODES

time	exact	klower	kupper	N = 25				N = 100			
				umc	cmc	vumc	vcmc	umc	cmc	vumc	vcmc
22	0.00028	0.00000	0.00278	0.00000	0.00036	0.00000	0.00000	0.00000	0.00032	0.00000	0.00000
23	0.00257	0.00014	0.01389	0.00000	0.00436	0.00000	0.00001	0.00000	0.00289	0.00000	0.00000
24	0.01210	0.00265	0.04167	0.00800	0.01609	0.00032	0.00004	0.00800	0.01215	0.00008	0.00001
25	0.03959	0.01890	0.09444	0.05600	0.04804	0.00213	0.00019	0.03400	0.04059	0.00033	0.00004
26	0.09960	0.07199	0.17778	0.14400	0.11040	0.00499	0.00053	0.10200	0.09958	0.00092	0.00011
27	0.20307	0.18041	0.29167	0.23200	0.21760	0.00723	0.00098	0.18600	0.20193	0.00152	0.00023
28	0.34769	0.33538	0.42778	0.30400	0.36027	0.00843	0.00128	0.34000	0.34102	0.00225	0.00033
29	0.51326	0.50799	0.57222	0.49600	0.51929	0.01003	0.00134	0.51200	0.50333	0.00249	0.00033
30	0.67109	0.66962	0.70833	0.70400	0.67218	0.00845	0.00093	0.67600	0.66104	0.00218	0.00022
31	0.80009	0.79996	0.82222	0.84000	0.79902	0.00525	0.00050	0.80800	0.79338	0.00155	0.00012
32	0.89306	0.89306	0.90556	0.92000	0.89449	0.00293	0.00021	0.90800	0.88956	0.00084	0.00005
33	0.95278	0.95278	0.95833	0.95200	0.95316	0.00184	0.00007	0.95400	0.95051	0.00044	0.00002
34	0.98472	0.98472	0.98611	0.98400	0.98462	0.00064	0.00001	0.98600	0.98362	0.00014	0.00000
35	0.99722	0.99722	0.99722	0.99200	0.99716	0.00032	0.00000	0.99800	0.99693	0.00002	0.00000
36	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000	0.00000	1.00000	1.00000	0.00000	0.00000
MEAN	29.48290	29.58520	29.00000	29.36800	29.42298	0.00350	0.00041	29.48800	29.52314	0.00085	0.00010

TABLE B.3—24 NODE NETWORK WITH 10 C-NODES

time	exact	klower	kupper	umc	N = 25			umc	cmc	vumc	vcmc
					umc	cmc	vumc				
29	0.00000	0.00000	0.00595	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
30	0.00003	0.00000	0.02976	0.00000	0.00001	0.00000	0.00000	0.00000	0.00003	0.00000	0.00000
31	0.00070	0.00003	0.08333	0.00000	0.00069	0.00000	0.00000	0.00000	0.00083	0.00000	0.00000
32	0.00712	0.00133	0.17262	0.00800	0.00690	0.00032	0.00001	0.00400	0.00769	0.00004	0.00000
33	0.03843	0.01630	0.29167	0.04800	0.03740	0.00181	0.00016	0.02400	0.04062	0.00023	0.00004
34	0.12987	0.08674	0.42857	0.12000	0.12643	0.00421	0.00080	0.10000	0.13324	0.00090	0.00020
35	0.30522	0.25750	0.57143	0.34400	0.29450	0.00920	0.00177	0.29400	0.30773	0.00204	0.00048
36	0.53611	0.50555	0.70833	0.60800	0.52344	0.00984	0.00228	0.52000	0.54029	0.00251	0.00063
37	0.75335	0.74180	0.82738	0.78400	0.74568	0.00683	0.00149	0.72400	0.75387	0.00201	0.00045
38	0.89906	0.89610	0.91667	0.92000	0.89183	0.00304	0.00073	0.89400	0.89873	0.00095	0.00019
39	0.96806	0.96754	0.97024	0.98400	0.97048	0.00064	0.00023	0.96400	0.97011	0.00035	0.00005
40	0.99405	0.99405	0.99405	1.00000	0.99429	0.00000	0.00003	1.00000	0.99486	0.00000	0.00001
41	1.00000	1.00000	1.00000	1.00000	1.00000	0.00000	0.00000	1.00000	1.00000	0.00000	0.00000
MEAN	36.36800	36.53310	35.00000	36.18400	36.40834	0.00276	0.00058	36.47600	36.35200	0.00070	0.00016

TABLE B.4—40 NODE NETWORK WITH 22 C-NODES

time	klower	kupper	umc	N = 25			umc	cmc	vumc	vcmc
				umc	cmc	vumc				
49	0.00119	0.03125	0.00800	0.00202	0.00032	0.00002	0.00200	0.00232	0.00002	0.00000
50	0.01163	0.18750	0.01600	0.01767	0.00064	0.00014	0.02400	0.01781	0.00024	0.00003
51	0.04748	0.38914	0.07200	0.06848	0.00269	0.00042	0.07600	0.06932	0.00071	0.00010
52	0.11176	0.43317	0.13600	0.14966	0.00485	0.00085	0.17600	0.15044	0.00146	0.00021
53	0.18382	0.47862	0.19200	0.23366	0.00632	0.00107	0.23800	0.22790	0.00183	0.00026
54	0.25004	0.52532	0.24800	0.30738	0.00757	0.00116	0.30000	0.29528	0.00212	0.00030
55	0.31559	0.57315	0.30400	0.37300	0.00864	0.00128	0.33800	0.35913	0.00226	0.00033
56	0.38366	0.62195	0.37600	0.43912	0.00944	0.00140	0.39600	0.42447	0.00240	0.00037
57	0.45234	0.66995	0.44000	0.50606	0.00973	0.00150	0.46800	0.49039	0.00249	0.00040
58	0.52056	0.71042	0.51200	0.57322	0.00973	0.00155	0.54600	0.55626	0.00248	0.00042
59	0.58711	0.74312	0.57600	0.63733	0.00997	0.00154	0.61800	0.61979	0.00238	0.00042
60	0.64997	0.77334	0.62400	0.69564	0.00944	0.00142	0.67600	0.67806	0.00220	0.00039
61	0.70738	0.80107	0.70400	0.74754	0.00837	0.00124	0.72400	0.73019	0.00200	0.00034
62	0.75849	0.82649	0.76800	0.79289	0.00723	0.00104	0.77000	0.77650	0.00178	0.00029
63	0.80276	0.84988	0.79200	0.83169	0.00656	0.00085	0.79600	0.81685	0.00162	0.00024
64	0.84009	0.87126	0.84000	0.86441	0.00541	0.00067	0.83600	0.85112	0.00138	0.00019
65	0.87096	0.89064	0.87200	0.89143	0.00448	0.00052	0.87400	0.87975	0.00110	0.00015
66	0.89622	0.90803	0.89600	0.91347	0.00379	0.00039	0.89000	0.90336	0.00098	0.00012
67	0.91679	0.92348	0.92000	0.93129	0.00293	0.00029	0.91600	0.92264	0.00077	0.00009
68	0.93352	0.93706	0.94400	0.94562	0.00208	0.00021	0.92600	0.93830	0.00068	0.00007
69	0.94713	0.94886	0.96000	0.95714	0.00144	0.00015	0.95200	0.95100	0.00045	0.00005
70	0.95823	0.95899	0.96800	0.96644	0.00120	0.00011	0.95800	0.96132	0.00040	0.00003
71	0.96729	0.96759	0.96800	0.97398	0.00120	0.00007	0.97200	0.96973	0.00027	0.00002
72	0.97468	0.97478	0.96800	0.98007	0.00120	0.00005	0.97400	0.97658	0.00025	0.00002
73	0.98068	0.98070	0.97600	0.98496	0.00093	0.00003	0.97600	0.98216	0.00024	0.00001
74	0.98550	0.98551	0.97600	0.98885	0.00093	0.00002	0.97600	0.98664	0.00024	0.00001
75	0.98934	0.98934	0.97600	0.99191	0.00093	0.00001	0.98000	0.99019	0.00020	0.00000
76	0.99233	0.99233	0.97600	0.99425	0.00093	0.00001	0.98400	0.99296	0.00016	0.00000
77	0.99462	0.99462	0.99200	0.99602	0.00032	0.00000	0.98800	0.99507	0.00012	0.00000
78	0.99633	0.99633	0.99200	0.99733	0.00032	0.00000	0.99400	0.99664	0.00006	0.00000
79	0.99758	0.99758	0.99200	0.99826	0.00032	0.00000	0.99600	0.99779	0.00004	0.00000
80	0.99847	0.99847	1.00000	0.99891	0.00000	0.00000	1.00000	0.99860	0.00000	0.00000
81	0.99907	0.99907	1.00000	0.99935	0.00000	0.00000	1.00000	0.99915	0.00000	0.00000
82	0.99947	0.99947	1.00000	0.99963	0.00000	0.00000	1.00000	0.99951	0.00000	0.00000
83	0.99971	0.99971	1.00000	0.99981	0.00000	0.00000	1.00000	0.99974	0.00000	0.00000
84	0.99986	0.99986	1.00000	0.99991	0.00000	0.00000	1.00000	0.99987	0.00000	0.00000
85	0.99994	0.99994	1.00000	0.99996	0.00000	0.00000	1.00000	0.99995	0.00000	0.00000
86	0.99998	0.99998	1.00000	0.99999	0.00000	0.00000	1.00000	0.99998	0.00000	0.00000
87-90	1.00000	1.00000	1.00000	1.00000	0.00000	0.00000	1.00000	1.00000	0.00000	0.00000
MEAN	58.97850	56.07200	59.01600	58.25164	0.00309	0.00043	58.66000	58.49324	0.00079	0.00012