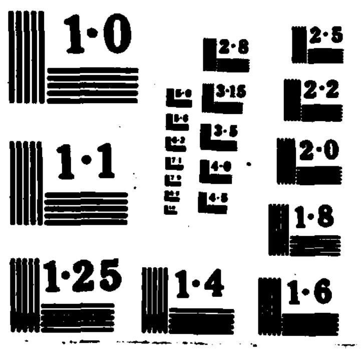
AD-A170 275   MARKOV ALGORITHMS FOR COMPUTING THE RELIABILITY OF   1/1
STAGED NETWORKS..(U) NORTH CAROLINA UNIV AT CHAPEL HILL
CURRICULUM IN OPERATIONS R.. R C CHENG APR 86
UNCLASSIFIED   UNC/ORSA/TR-86/8 AFOSR-TR-86-0436   F/G 12/1   NL


1·0
2·8
2·5
3·15
2·2
3·5
1·1
4·0
2·0
4·5
1·8
1·25
1·4
1·6

2

OPERATIONS RESEARCH AND SYSTEMS ANALYSIS

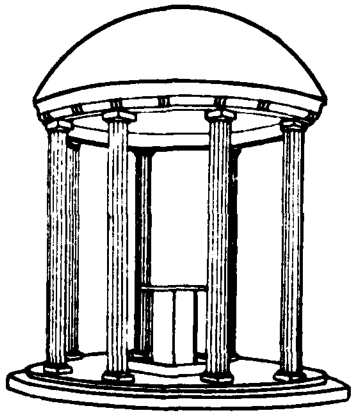
Markov Algorithms for Computing the Reliability of Staged Networks

R. C. H. Cheng

Technical Report No. UNC/ORSA/TR-86/8

April 1986

UNIVERSITY OF NORTH CAROLINA
AT CHAPEL HILL



Markov Algorithms for Computing the Reliability of Staged Networks

R. C. H. Cheng

Technical Report No. UNC/ORSA/TR-86/8

April 1986

Curriculum in Operations Research
and Systems Analysis
University of North Carolina at Chapel Hill

This research was supported by the Air Force Office of Scientific Research under grant AFOSR-84-0140. Reproduction in whole or part is permitted for any purpose of the United States government.

A -

ABSTRACT

Certain commonly occurring types of network, whether directed or undirected, exhibit a staged structure. Two algorithms, based on node partitioning, are presented which take advantage of such structure and which use a Markov transition-probability form of recursion. The algorithm for directed networks is related to the Markov chain formulation of Bailey and Kulkarni, but for undirected networks a more detailed form of state definition is used related to one suggested by Rosenthal.

The computational advantages of the algorithms are discussed and some numerical results presented.

Key Words: Network Reliability, Source-to-sink connectedness, Node Partition, Staged Network, Recursive Algorithm.

1. INTRODUCTION

Consider a network G with node set N and arc set A. The nodes are perfect but arcs fail randomly and independently with probability $1-p_a$, $a \epsilon A$. Let s and t be two specified nodes. The calculation of

$$p_{st} = \text{prob (there is a path from s to t)} \qquad (1.1)$$

is called the s-t connectedness problem.

This problem is NP hard [10]. However, many algorithms exist which solve it by enumerating key structures in the network. If the number of these is small then such an algorithm may be computationally efficient. Examples include spanning trees [2], acyclic subgraphs [8], K-graphs [7] and cutsets [4].

A number of methods use node partition [1,6,9]. Though none of these make any special assumption about the form of the network, node partition seems to be a particularly good approach for networks which have a staged structure. In this paper we consider two node partition algorithms which exploit such structure.

Our starting point is a decomposition used by Shogan [9] for directed networks where the nodes are divided into disjoint groups - which we call stages - and 'events' are then defined at each stage in a way which allows their probabilities of occurrence to be calculated from those of the previous stage. Shogan describes an algorithm based on path enumeration and mentions, but does not follow the possibility of one based on state enumeration. We suggest a modification of the definition of event given by Shogan which gives an algorithm of this latter type having a simpler form than that based on path enumeration.

Undirected networks can often also be decomposed into stages. Our main result will be to show how the algorithm can be adapted to deal with this case, too.

Both versions of the algorithm are related to ones already cited. The directed network version is related to that proposed by Bailey and Kulkarni [1], whilst the undirected network algorithm is a recursive version of the general, but not fully specified, framework described by Rosenthal [6].

As well as describing the algorithms, we assess their computational advantages - and drawbacks - including some numerical results.

For clarity, the bulk of our discussion assumes a form of stage decomposition that is simpler than necessary. In section 5 we show how the stage decomposition can be generalized to allow versions of the algorithms that are, in certain situations, much more efficient than the prototype versions.

2. DEFINITIONS AND NOTATION

2.1 Staged Networks

Let G be a network with node set N and arc set A. An arc connecting node i to node j is denoted by (i,j). We shall say that G is a <u>staged network</u> if N can be divided into a number, Q say, of disjoint sets

$$S_1 = \{s\},\ S_2, \ldots, S_Q = \{t\} \qquad (2.1)$$

such that

$\tilde{S}$: if $(i,j)\varepsilon A$ and $j\varepsilon S_q$ then $i\varepsilon S_{q-1}\cup S_q$. (2.2)

(Shogan gives a slightly more general version which allows $i\varepsilon S_1\cup S_{q-1}\cup S_q$ in (2.2). The generalized definition of Section 5 incorporates this case.)

Condition $\tilde{S}$ ensures that any path from s to t can only go from nodes of one stage to those of the next stage. Thus no 'backtracking' from a higher to lower stage is possible.

The node-stages induce a corresponding stage structure on the arcs A, dividing these into disjoint sets:

$$A_2,\ A_3,\dots,A_Q$$

where

$$A_q = \{(i,j)\ |\ i\varepsilon S_{q-1}\cup S_q \text{ and } j\varepsilon S_q\}. \qquad (2.3)$$

To note which arcs of A_q are up (operating) and which are down (failed), we define

X_i = 1 or 0 according as arc $i\varepsilon A_q$ is up or down, and

call

$$X = (X_1, X_2, \dots, X_{|A_q|}). \qquad (2.4)$$

a <u>failure pattern</u> of A_q. It will be clear, from the context, which subset A_q is being referred to, so the dependence of X on q will be suppressed.

Those arcs which, under X, are up, will be called the set of up-arcs of A_q. The probability of obtaining X is

$$p(X) = \prod_{a:X_a=1} p_a \prod_{a:X_a=0} (1-p_a). \qquad (2.5)$$

There are $2^{|A_q|}$ distinct failure patterns of A_q and we denote the set of all these as F_q.

Consider now an undirected network G. We shall say that G is <u>staged</u> if the nodes can be divided into disjoint sets

$$S_1 = \{s\},\ S_2,\ldots,S_Q = \{t\}$$

such that

$\tilde{T}$: if $(i,j) \in A$ and $i \in S_q$, $1<q<Q$ then $j \in S_{q-1} \cup S_q \cup S_{q+1}$. (2.6)

Condition $\tilde{T}$ requires that an arc can only join two nodes of the same or adjacent stages. Decomposition of the arc set into disjoint subsets A_q exactly as defined in (2.3) is possible, and the definition of failure pattern X given in (2.4) also still applies. An example of an undirected staged network is given in Fig. 2.

2.2 Recursive Markov Algorithms

The algorithms that we investigate operate in the following way:

(i) A set of events, E, has to be found where each event $I \in E$, is associated with some subset of nodes.

(ii) The events are Markov in the sense that their probabilities of occurrence, p_I, can be calculated from the standard Markov one-step transition formula

$$p_J = \sum_{I \in E} p_I q_{IJ} \qquad J \in E \tag{2.8}$$

where

$$q_{IJ} = \text{prob } (J \text{ occurs} \mid I \text{ occurs}) \tag{2.9}$$

is the transition probability.

(iii) If the events can be numbered so that transitions go only from a lower to higher numbered event, then $q_{IJ} = 0$ if $J<I$ and the p_J may be calculated recursively from (2.8) in increasing order of J. A special form of this is when the events can be grouped into disjoint sets $E_1, E_2, \ldots, E_Q$ with events of one group giving rise to events of the next higher group only. Then (2.8) reduces to

$$p_J = \sum_{I \varepsilon E_q} p_I q_{IJ}, \qquad J \varepsilon E_{q+1}, \qquad q=1,2,\ldots,Q. \qquad (2.10)$$

3. THE ALGORITHMS

3.1 A Markov Algorithm for a Directed Staged Network

Consider a directed staged network. The algorithm of Shogan focuses on events associated with a particular stage. Let V be a subset of S_q. Shogan defines the events

$$\left.\begin{aligned} U(V) &= \text{"there is a path from s to at least one node of V"} \\ I(V) &= \text{"there are paths from s to all nodes of V"} \end{aligned}\right\} \qquad (3.1)$$

and gives exclusion-inclusion formulas relating the probabilities of occurrence of such events to corresponding events of the previous stage.

It is perhaps simpler to use the following events. Let $V \subseteq S_{q-1}$ and $W \subseteq S_q$ and define

$$[V] = \text{"there is a path from } s \text{ to all nodes of } V, \text{ but none from } s \text{ to } S_{q-1}\backslash V\text{"} \quad (3.2)$$

$$[V,W] = \text{"Given } [V] \text{ has occurred, each node of } W \text{ is reachable from some node of } V \text{ but there is no path from } S_{q-1}\backslash V \text{ to } W, \text{ or from } V \text{ to } S_q\backslash W\text{"} \quad (3.3)$$

The event (3.2) can be viewed as the union of a number of events, of a more general type given by Bailey and Kulkarni [1], that utilizes the staged structure.

Let $p[V]$ and $p[V,W]$ denote the probabilities that $[V]$ and $[V,W]$ occur. Then

$$\begin{aligned} p[s] &= 1 \\ p[W] &= \sum_{V \subseteq S_{q-1}} p[V]p[V,W], \quad W \subseteq S_q, \; q=2,\ldots,Q. \end{aligned} \quad (3.4)$$

This has form (2.10): the events $\{[V], V \subseteq S_{q-1}\}$ form E_{q-1}, and the $p[V,W]$ are the q_{IJ}. The last stage calculation (when $q = Q$ and $W = \{t\}$) yields p_{st}, the probability of s-t connectedness.

What makes (3.4) simple is that the $p[V,W]$ are very easily calculated, because $[V,W]$ depends only on the states of the arcs of A_q. To see this, suppose that $v \in V$, $w \in S_q$ and there is a path of up-arcs from v to w. From the definition of staged network, this path may initially pass through nodes of S_{q-1}, however, once it reaches a node of S_q it can only continue through nodes of S_q until w is reached. Thus, given $[V]$ has occurred, all the nodes of S_{q-1} on the path must belong to V. If v' is the last node of V

on the path, there is, therefore, a path from $v' \epsilon V$ to w comprised entirely of up-arcs of A_q. Similarily a path from $u \epsilon S_{q-1} \backslash V$ to W would, given that $[V]$ has occurred, imply the existence of some $u' \epsilon S_{q-1} \backslash V$ and a path of up-arcs of A_q from u' to w. It follows, therefore, that $[V,W]$ occurs if and only if, using paths with arcs in A_q only, each node of W is reachable from some node of V, no node of $S_q \backslash W$ is reachable from V, no node of W is reachable from $S_{q-1} \backslash V$. This is the required result. It can be viewed in the following way:

Lemma 1 Let $V \subseteq S_{q-1}$ and $X \epsilon F_q$. Then the pair V,X induces a subset $W \subseteq S_q$ defined by

$$W(V,X) = \{w \mid w \epsilon S_q \text{ and there is a path, comprised only of up-arcs of } A_q \text{ (under } X\text{), from some node of } V \text{ to } w\} \tag{3.5}$$

$\square$

Figure 1 illustrates an example of such a $W(V,X)$.

Clearly

$$p[V,W] = \sum_{\substack{X \epsilon F_q \\ W(V,X) = W}} p(X) \tag{3.6}$$

and (3.4) reduces to $p[s] = 1$ and

$$p[W] = \sum_{\substack{V \subseteq S_{q-1}}} \sum_{\substack{X \epsilon F_q \\ W(V,X) = W}} p[V]p(X), \quad q=2,\dots,Q. \tag{3.7}$$

In numerical calculations (3.7) is easy to implement as there is no need to compute the $p[W]$ one at a time. Instead, note that in calculating all the p[W], each V,X combination is used once only. Thus (3.7) can be implemented as

Algorithm A

```
        $p[s] \leftarrow 1$
    for $q=2,..,Q$
        $p[W] \leftarrow 0$  all $W \subseteq S_q$
    for each $V \subseteq S_{q-1}$
    for each $X \epsilon F_q$                                    (3.8)
          $W \leftarrow W(V,X)$
        $p[W] \leftarrow p[W] + p[V]p(X)$
    next X
    next V
    next q
```

3.2 A Markov Algorithm for Undirected Staged Networks

For an undirected network the event definition (3.2) does not give a useful algorithm because a path may 'backtrack.' Instead, consider all the nodes of a stage S_q and think of them as partitioned into disjoint groups. Fig. 2 illustrates S_2 partitioned as (1)(2)(34). We denote a partition by π and the set of all partitions of S_q by T_q, and define the following event:

$$[I] = [\pi;\ \nu]$$

$$= \text{"}\pi \epsilon T_q \text{ and, using arcs of } \bigcup_{r=2}^{q} A_r \text{ only:}$$

(i) each group of nodes in π is connected

(ii) nodes between groups are unconnected

(iii) s is connected to group ν" (3.9)

The set of all such events associated with S_q will be denoted by E_q. Fig. 2 illustrates the event $[(1)(2)(34);3] \epsilon E_2$ and the event $[(5)(67);2] \epsilon E_3$.

The above definition can be viewed as a specialization of a general type of event considered by Rosenthal [6], who outlines an algorithm framework based on combining subnetworks two at a time. The framework does not make specific use of staged structure but clearly has such a form in mind. The algorithm below takes specific note of the Markov nature of the calculation, so that the computational form is somewhat different to that of Rosenthal's even though the event probabilities are essentially the same. We use an analogue of Lemma 1:

Lemma 2 *An event* $I = [\pi;\lambda] \epsilon E_{q-1}$ *and failure pattern* $X \epsilon F_q$ *induces a unique event* $J = [\rho;\nu] \epsilon E_q$, which will be denoted by $J(I,X)$.

Proof We form a subnetwork, H, as follows. The node set is $S_{q-1} \cup S_q$. The nodes of S_q are treated normally. However, those of S_{q-1} are divided by the partition π into disjoint groups and, in H, the nodes of each group are treated as being combined into a single node (there being a separate node for each set). The arc set of H is just the set of up-arcs of X.

Now in H the nodes of S_q can be naturally partitioned into disjoint sets; all the nodes within a set are connected (possibly via paths which pass through combined nodes of S_{q-1}), but nodes belonging to different sets are disconnected. If we denote this partition by ρ and take ν to be the set (of ρ) which is connected to λ, then this uniquely defines $J = [\rho;\nu]$. □

Fig. 2 gives an example of such an event $J(I,X)$.

The same recursive algorithm (3.7) still applies.

$$p[s;1] = 1$$

$$p[J] = \sum_{\substack{I \varepsilon E_{q-1} \\ J(I,X) = J}} \sum_{X \varepsilon F_q} p[I]p(X), \quad J \varepsilon E_q, \; q=2,\dots,Q \tag{3.10}$$

and again the modified form (3.8) can be used for numerical calculation:

Algorithm B

```
    $p[s;1] = 1$
  for $q=2,\dots,Q$
      $p[J] \leftarrow 0$  all $J \varepsilon E_q$
  for each $I \varepsilon E_{q-1}$                                  (3.11)
  for each $X \varepsilon F_q$
         $J \leftarrow J(I,X)$
      $p[W] = p[W] + p[I]p(X)$
  next X
  next I
  next q
```

4. COMPUTATIONAL ASPECTS AND COMPLEXITY

For Algorithm A, the calculation of $W(V,X)$ and $p[W]$ is done

$$2^{|S_{q-1}| + |A_q|} \tag{4.1}$$

times at stage q. For Algorithm B, finding the partitions ρ is the most time consuming calculation. From the definition of $J(I,X) = [\rho;\nu]$ where $I = [\pi;\lambda]$ we see that ρ depends on π and X only and not on ν. Thus in (3.11) ρ needs only be determined for each (π,X) and not each (π,X,ν), a total of

$$g(|S_{q-1}|) \times 2^{|A_q|} \tag{4.2}$$

times, where g(m) = number of possible partitions of m objects, is the sum of Stirling numbers of the second kind (see, eg. [5]). As g grows hyperexponentially, see Table 1 for selected values, this is the main factor limiting the size of networks that can be handled.

Table 1

g(m) = number of partitions of m objects

m	1	2	3	4	5	6	7	8
2m	2	4	8	16	32	64	128	256
g(m)	1	2	5	15	52	203	877	4140

Each calculation of W in Algorithm A, and ρ in Algorithm B, requires examination of the connectivity of the subgraph $G_q = \{S_{q-1} \cup S_q, A_q\}$. An algorithm like "breadth-first search" (see [5], for example) can be used for this, and will need $O(|S_{q-1}|+|S_q|+A_q)$ elementary operations.

Algorithm B has the additional requirement of a hash function or subroutine which (i) assigns a label to each partition (of a set of given cardinality) and, inversely, (ii) given a particular label, specifies the precise grouping of the nodes in that partition. Part (i) is needed to identify the component of the array p[J] to be incremented in (3.11) and (ii) is needed to allow partitions to be stepped through systematically in the outer loop of (3.11). The Appendix gives such a subroutine, whose complexity is O(m) where m is the cardinality of the set being partitioned.

To test its effectiveness, a Fortran version of Algorithm B was used, on an IBM Personal Computer AT, to calculated p_{st} for the dodecahedron network of Fig. 3 and the grid network of Fig. 4. The computation time was 4½ minutes and 52 minutes respectively. Though we have not made a direct comparison, this compares well with other algorithms. For example, for the test network used in [1], which is in effect a dodecahedron reduced by 3 nodes and 5 arcs, Bailey and Kulkarni report timings of 54 minutes, 8 minutes and 1 minute 26 seconds using respectively, the algorithms of Buzacott [3], Provan and Ball [4] and their own [1], on an IBM 4381-k which is approximately 10 times faster than the IBM PC-AT, (all algorithms computed the reliability as .99806 when $p_a = 0.9$, all a).

Both Algorithms A and B resemble dynamic programming recursions whose calculations of one stage depend only on the results of the immediately preceding stage. For networks like the grid of Fig. 4, this means that a substantial computational saving is possible by avoiding the repetition of identical calculations made at each stage. More precisely, suppose the subnetworks $G_q = \{S_{q-1} \cup S_q, A_q\}$ and $G_r = \{S_{r-1} \cup S_r, A_r\}$ (with $q<r$, say) have exactly the same structure. Then, during the calculations at stage q, the computed values of $W(V,X)$ (respectively $J(I,X)$) should be saved for all $V \subseteq S_{q-1}$ (respectively $I \varepsilon E_{q-1}$) and $X \varepsilon F_q$. This time consuming computation need not then be repeated during the rth stage calculation, as, with suitable labelling of the nodes and arcs of G_r, so that they correspond to those of G_q, the same values of $W(V,X)$ (respectively $J(I,X)$) can be used for $V \subseteq S_{r-1}$ (respectively $I \varepsilon E_{r-1}$) and $X \varepsilon F_r$.

For example in the grid network of Fig. 4, the CPU time of 52 minutes is mostly taken up by the calculations of stage 3 (12 min 5 secs) and of stages 4, 5 and 6 (13 min 10 secs each). As G_q $(q=3,4,5,6)$ are identical, we would estimate an algorithm that avoids the repeated calculation of the $J(I,X)$ would reduce the computation by more than half.

Finally we note that, for clarity of exposition, an unnecessarily rigid definition of staging has been given. In the next section we show how the definition can be generalized. It should be emphasized that though the timings above already compare favorably with published work, the use of generalized stages can lead to further substantial savings in computing time.

5. OVERLAPPING STAGES

The definitions of staging, $\tilde{S}$ and $\tilde{T}$, can be relaxed to give more flexibility to the two algorithms. The following generalization can be used for both directed and undirected networks.

$\tilde{U}$: There are three sequences of subsets of N:

$$\begin{aligned} &S_1 = \{s\},\ S_2,\ldots,S_Q = \{t\} \\ &M_1,\ M_2,\ldots,M_{Q-1} \\ &N_1 = \{s\},\ N_2,\ldots,N_Q = \{t\} \end{aligned} \qquad (5.1)$$

such that

$$\bigcup_{q-1}^{Q} S_q = N \qquad (5.2)$$

$$\left.\begin{aligned} S_q &= (S_{q-1} \backslash M_{q-1}) \cup N_q \\ M_{q-1} &\subseteq S_{q-1} \\ N_q \cap \Big(\bigcup_{r-1}^{q-1} S_r\Big) &= \phi. \end{aligned}\right\} \quad q=2,\ldots,Q \qquad (5.3)$$

And for any $(i,j)\varepsilon A$:

$$\text{if } i\varepsilon N_q,\ j\varepsilon N_r \text{ and } q>r \text{ then } j\varepsilon S_{q-1} \qquad (5.4)$$

□

The condition (5.3) allows the sets S_q to overlap. Conditions $\tilde{S}$ and $\tilde{T}$ comprise the special case $M_q = S_q$, all q, when $N_q = S_q$; the S_q are then disjoint and (5.4) reduces to (2.2) or (2.6).

It is readily verified that the N_q are disjoint and that

$$\bigcup_{q=1}^{Q} N_q = N. \qquad (5.5)$$

The node stages induce a staged structure on the arcs which still decompose into disjoint subsets

$$A_2, A_3, \ldots, A_Q \quad \text{with} \quad \bigcup_2^Q A_q = A \tag{5.6}$$

where

$$A_q = \{(i,j) | i \epsilon S_{q-1} \cup N_q \text{ and } j \epsilon N_q\}. \tag{5.7}$$

With this definition of S_q and A_q, Algorithms A and B still apply completely unaltered.

The stage q calculations are still of the order of (4.1) or (4.2) under condition $\tilde{U}$. The calculations of stage q will thus be efficient if S_{q-1} and A_q are such that the following two conditions are met.

(i) The cardinality of S_{q-1} is kept small to keep small the factor involving $|S_{q-1}|$ in (4.1) or (4.2). This requires making $|M_{q-1}|$ as large, and $|N_q|$ as small as possible.

(ii) The cardinality of A_q is kept small to keep small the factor $2^{|A_q|}$ in (4.1) or (4.2). This can be done, indirectly, by making $|N_q|$ as small as possible.

There is a trade-off here in that a small $|N_q|$ at each stage will make for a larger total number of stages Q. As computational complexity depends only linearly on Q, but exponentially on $|A_q|$, the trade in the direction of small $|N_q|$ will almost always be worth making.

As an example consider the grid of Fig. 4. If, instead of the staging given in the figure, we use

$$S_2 = (1,2,\ldots,6),$$

$$S_3 = (S_2\backslash(1))\cup 7 = (2,3,\ldots,7)$$

$$S_4 = (S_3\backslash(2))\cup 8 = (3,4,\ldots,8)$$

$$S_5 = (S_4\backslash(3))\cup 9 = (4,5,\ldots,9)$$

$$S_{30} = (25,26,\ldots,30)$$

$$S_{31} = (t)$$

then $|S_q| = 6$, $|M_{q-1}| = |N_q| = 1$ for $q = 2,3,\ldots.30$, and $|A_q| \leq 2$ for $q = 3,4,\ldots 30$. The four main stages of the original decomposition, each with $|A_q| = 11$, is thus replaced by 24 stages each with $|A_q| \leq 2$. The computational complexity is thus reduced by a factor $\sim (4\times2^{11})/(24\times2^2) = 85.3$. Because of incidental computational overheads which we have not fully accounted for (such as initialization of arrays), the actual improvement in speed will be by a factor rather less 85.3; however the improvement is still very substantial. A Fortran implementation of this version of staging reduced the computing time, from the 52 minutes quoted previously, to 1 minute 58 seconds.

A similar use of overlapping stages for the dodecahedron example of Fig. 3 reduced computation time from 4½ minutes to 34 seconds.

REFERENCES

[1] Bailey, M.P. and Kulkarni, V.G. A Recursive Algorithm for Computing Exact Reliability Measures, IEEE Trans. Reliability, vol 35, 1986, pp 36-40.

[2] Ball, M.O. and Nemhauser, G.L. Matroids and a Reliability Analysis Problem, Math. Oper. Res., vol 4, 1979, pp 132-143.

[3] Buzacott, J.A. A Recursive Algorithm for Finding the Proability that a Graph is Disconnected, Networks, vol 10, 1980, pp 311-328.

[4] Provan, J.S. and Ball M.O. Computing Network Reliability in Time Polynomial in the Number of Cuts., Oper. Res., vol 32, 1984,, pp 516-526.

[5] Roberts, F.S. Applied Combinatorics, 1984, Prentice-Hall, Englewood Cliffs.

[6] Rosenthal, A. Computing the Reliability of Complex Networks, SIAM J. Appl. Math., vol 32, 1977, pp 410-421.

[7] Satyanarayana, A. and Chang, M.K. Network Reliability and the Factoring Theorem, Networks, vol 13, 1983, pp 107-120.

[8] Satyanarayana, A. and Hagstrom, J.N. Combinatorial Properties of Directed Graphs Useful in Computing Network Reliability, Networks, vol 11, 1981, pp 357-366.

[9] Shogan, A.W. A Decomposition Algorithm for Network Reliability Analysis, Networks, vol 8, 1978, pp 231-251.

[10] Valiant, L.G. The Complexity of Enumeration and Reliability Problems, SIAM J. Comp., vol 8, 1979, pp 410-421.

APPENDIX

We describe here a method of labelling partitions and its inverse which enables a partition to be recovered from its label.

The partitions of M objects can be divided into M sets, the kth set comprising all those partitions where the objects are divided into k groups. For instance (13)(2)(4) represents the partition of 4 objects into 3 groups: (13), (2) and (4); this partition thus belongs to the 3rd set. Let

$$g(k,M) = \text{number of partitions (of } M \text{ objects)}$$
$$\text{in the } k\text{th set}$$

These are the Stirling numbers of the second kind (see, for example, [5]).

We shall label partitions in the order of these sets (i.e. those of the first set first, then those of the second set, and so on). The position of a partition is thus completely defined once the position within the set to which it belongs is fixed. Table 1A illustrates one method by which the partitions of m objects may be built up by systematically listing all the ways that the mth object can be adjointed to each of the partitions of the previous m-1 objects. The method used is as follows.

For a partition of m objects, let k_m be the set to which it belongs and j_m its position in the set. Now, a partition is defined by specifying, for each object m, the group n_m to which it belongs. In Table 1A, k_M and j_M are determined from n_m $(m=1,2,\dots,M)$ using:

$$\left.\begin{aligned}
&k_1 = 1,\ j_1 = 1,\\
&k_m = k_{m-1},\ j_m = (j_{m-1}-1)k_{m-1}+n_m \quad \text{if } n_m \le k_{m-1}\\
&k_m = n_m,\ j_m = j_{m-1}+g(k_m,m)-g(k_{m-1},m-1) \ \text{otherwise}
\end{aligned}\right\} m=2,\ldots,M$$

which recursively gives the position j_m, k_m of the partition of the first m objects, for $m=1,2,\ldots,M$.

Table 1A

Partition of m objects into k groups

m / k	1	2	3	4
1	(1)	(12)	(123)	(1234)
2		(1)(2)	(13)(2)	(134)(2)
				(13)(24)
			(1)(23)	(14)(23)
				(1)(234)
			(12)(3)	(124)(3)
				(12)(34)
				(123)(4)
3			(1)(2)(3)	(14)(2)(3)
				(1)(24)(3)
				(1)(2)(34)
				(13)(2)(4)
				(1)(23)(4)
				(12)(3)(4)
4				(1)(2)(3)(4)

The overall position of a partition is

$$r = \sum_{k=1}^{k_m - 1} g(k,m) + j_m .$$

The inverse calculation allows the group, n_m, to which each object m belongs, to be calculated from r. It is as follows.

Set m = M and determine k_m, j_m ($\equiv k_M, j_M$) from the conditions

$$r - \sum_{k=1}^{k_M - 1} g(k,m) = j_M > 0, \quad r - \sum_{k=1}^{k_M} g(k,M) \leq 0.$$

Moreover $g(k,m) = kg(k,m) + g(k-1,m-1)$, so that

A. Either $j_m - k_m g(k_m, m-1) = j > 0$, in which case the partition (of the first m objects) is the jth amongst the last $g(k_m - 1, m-1)$ partitions of the k_mth set. This means the object is in the last group (on its own!) and so

$$n_m \equiv k_m, \; k_{m-1} = k_m - 1, \; j_{m-1} = j .$$

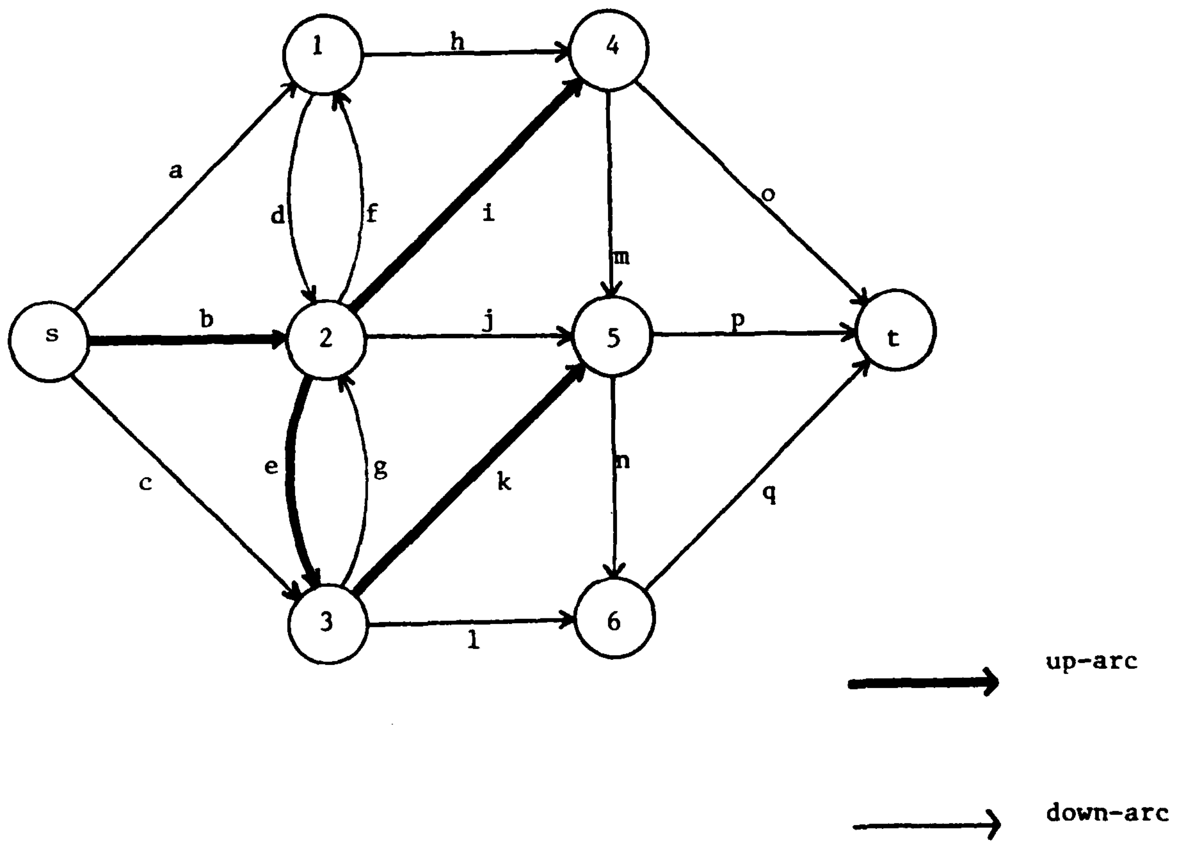
B. Otherwise we can set $k_{m-1} = k_m$ and find n_m, j_{m-1} from the conditions

$$n_m = j_m - (j_{m-1} - 1) k_m > 0, \; j_m - j_{m-1} k_m \leq 0.$$

This fixes object m as being in group n_m; and the partition of the first m-1 objects as being the j_{m-1}th in the k_{m-1}th set.

Thus setting $m \leftarrow m-1$, we can repeat steps A or B until m=1, when we set $n_1 = 1$; at which stage all the n_m (m=1,2,...,M) have been found.

Figure 1   A Directed Staged Network



$S_1 = (s)$ S_2 (1,2,3) $S_3 = (4,5,6)$ $S_4 = (t)$

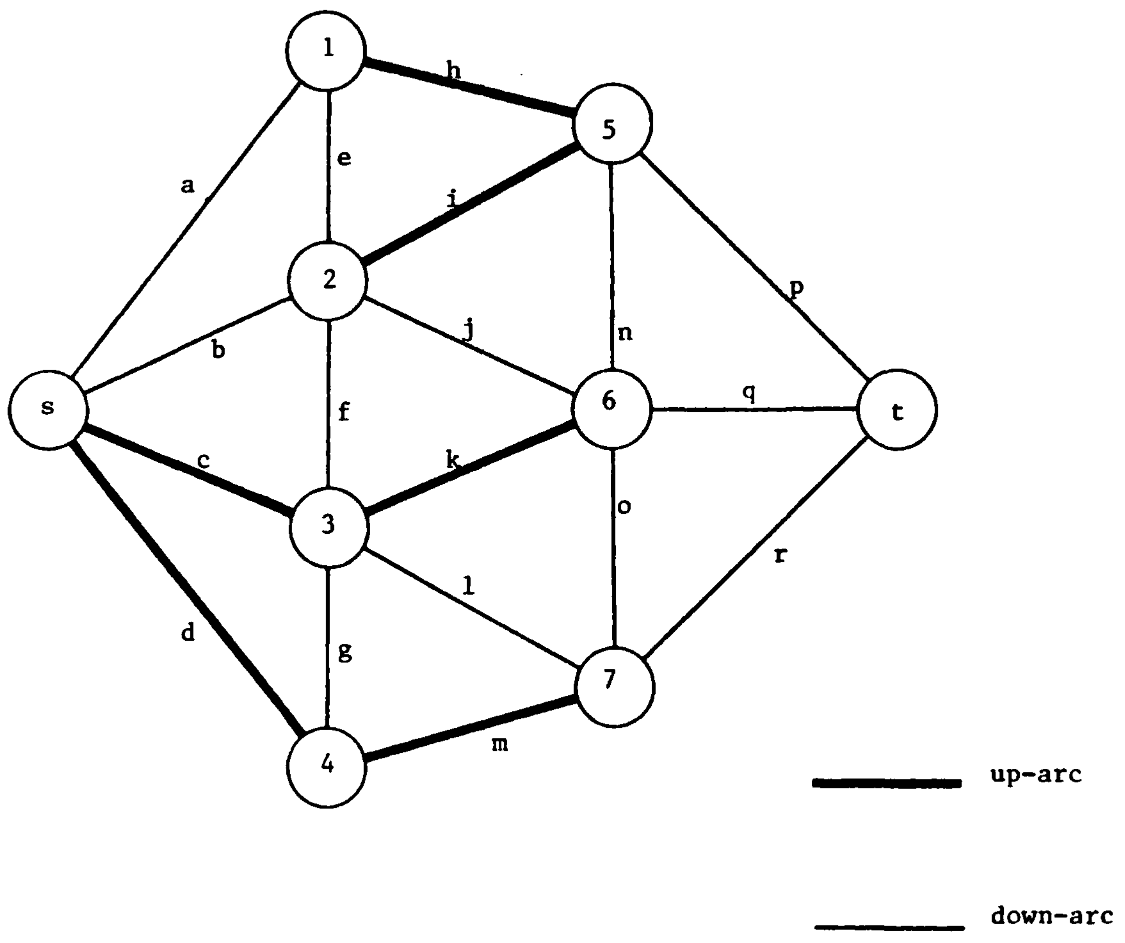
$A_2 = (a,b,\ldots,g)$ $A_3 = (h,i,\ldots,n)$ $A_4 = (o,p,q)$

Event [V] = [(2,3)] has occurred at stage 2.

A_2 has failure pattern X = (0101000).

W(V,X) = (4,5).

Figure 2   An Undirected Staged Network



S_1 = (s) S_2 = (1,2,3,4) S_3 = (5,6,7) S_4 = (t)

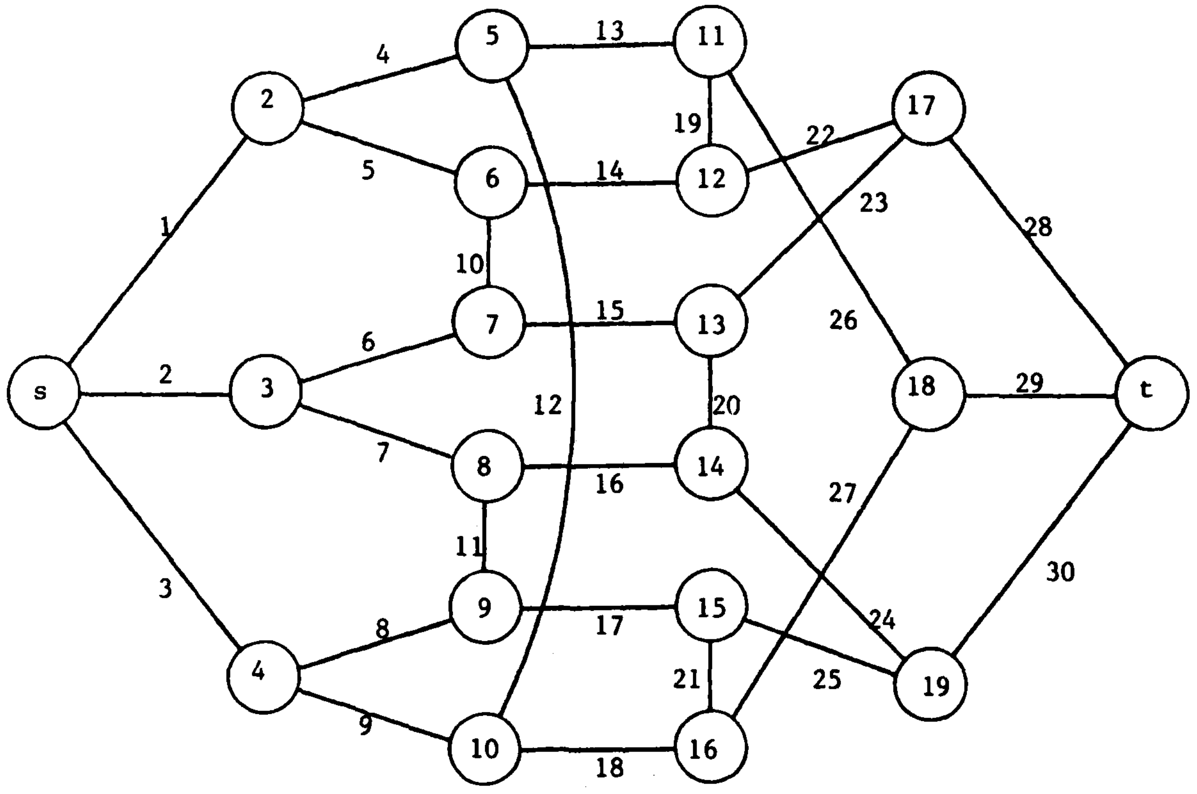
A_2 = (a,b,...,g) A_3 = (h,i,...,o) A_4 = (p,q,r)

Event I = [(1)(2)(34);3]   has occurred at stage 2.

A_2 has failure pattern X = (11010100)

J(I,X) = [(5)(67);2]

Figure 3 Dodecahedron



$S_1 = (s)$ $S_2 = (2,3,4)$ $S_3 = (5,6,\ldots,10)$ $S_4 = (11,12,\ldots,16)$

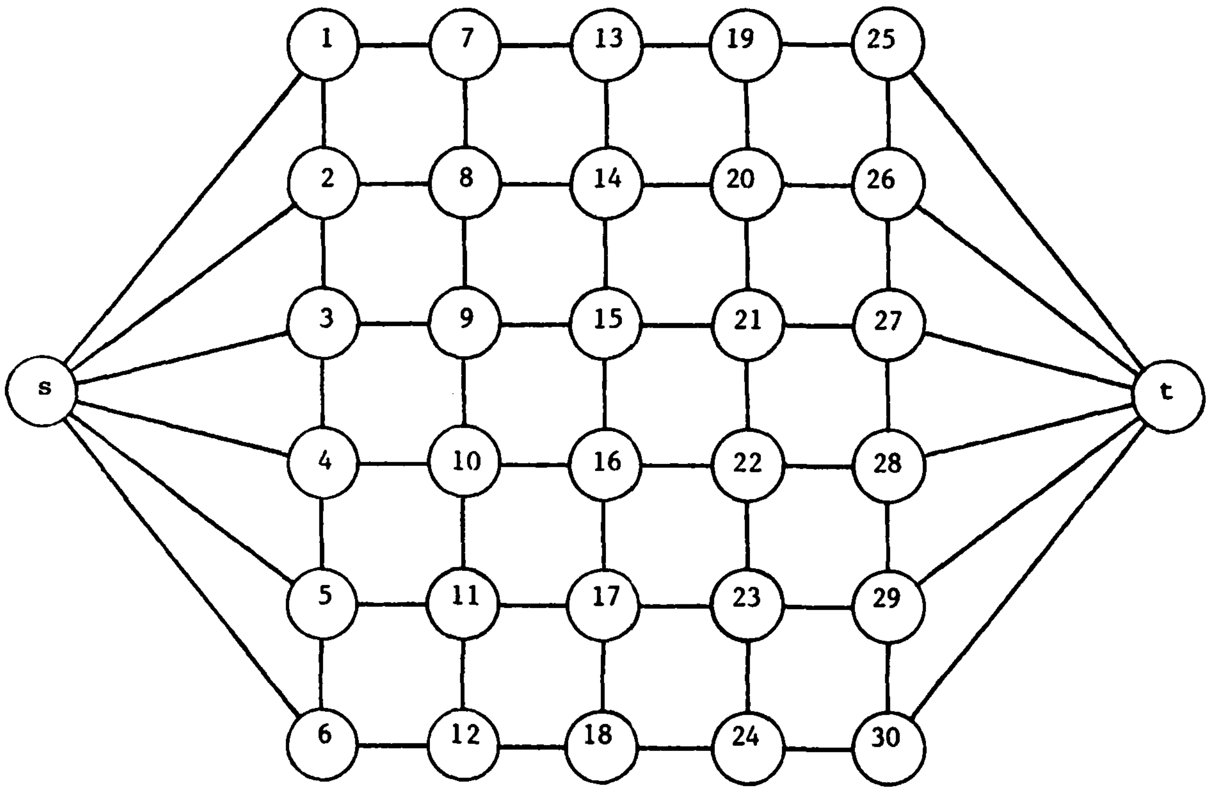
$S_5 = (17,18,19)$ $S_6 = (t)$

$A_2 = (1,2,3)$ $A_3 = (4,5,\ldots,12)$ $A_4 = (13,14,\ldots,21)$

$A_5 = (22,23,\ldots,27)$ $A_6 = (28,29,30)$

p(arc operational)	reliability
.5	.290155
.9	.997120
.95	.999705

Figure 4 A Grid Network



Reliability = 0.5 if p(arc operating) = 0.5

S_1 = (s) S_2 = (1,2,...,6) S_3 = (7,8,...,12)

S_4 = (13,14,...,18) S_5 = (19,20,...,24) S_6 = (25,26,...,30) S_7 = (t)

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

AD-A170 225

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY: NA	3. DISTRIBUTION/AVAILABILITY OF REPORT: Unlimited distribution
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE: NA	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S): AFOSR-TR- 86-0436
6a. NAME OF PERFORMING ORGANIZATION: University of North Carolina; 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION: AFOSR/NM
6c. ADDRESS (City, State and ZIP Code): Chapel Hill, NC 27514	7b. ADDRESS (City, State and ZIP Code): Bldg. 410, Bolling AFB, DC 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: AFOSR; 8b. OFFICE SYMBOL (If applicable): NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: AFOSR 84-0140
8c. ADDRESS (City, State and ZIP Code): Bldg. 410, Bolling AFB, DC	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
	2307	A5	

11. TITLE (Include Security Classification): Markov algorithms for computing the reliability of staged networks

12. PERSONAL AUTHOR(S)

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
technical	FROM ______ TO ______	April, 1986	25

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			
FIELD	GROUP	SUB. GR.	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number): Network reliability, source-to-sink connectedness, node partition, staged network, recursive algorithms

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Certain commonly occurring types of network, whether directed or undirected, exhibit a staged structure. Two algorithms, based on node partitioning, are presented which take advantage of such structure and which use a Markov transition probability form of recursion. The algorithm for directed networks is related to the Markov chain formulation of Bailey and Kulkarni, but for undirected networks a more detailed form of state definition is used related to one suggested by Rosenthal.

The computational advantages of the algorithms are discussed and some numerical results presented.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED [✓] SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION: Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL: R.C.H. Cheng	22b. TELEPHONE NUMBER (Include Area Code): 919-962-8401	22c. OFFICE SYMBOL: AFOSR/NM

DD FORM 1473, 83 APR   EDITION OF 1 JAN 73 IS OBSOLETE.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE