A Program for Reliability Evaluation of Undirected Networks via Polygon-to-Chain Reductions

Mauricio G. C. Resende
University of California, Berkeley

Key Words—**Computer program, Data structure, Graph reduction, Network reliability, Series-parallel graph**

Reader Aids—
Purpose: To describe an existing program
Special math needed for explanations: Probability
Special math needed to use results: None
Results useful to: Reliability analysts and theoreticians

Abstract—**This paper discusses the design and implementation of *PolyChain*, a FORTRAN program for reliability evaluation of undirected networks of a special structure via polygon-to-chain reductions. Theoretical results presented by Satyanarayana & Wood are reviewed. The program's design and its implementation in FORTRAN are described. A small problem is tested illustrating the code's output. Several large problems are run to evaluate the code's performance capabilities.**

1. INTRODUCTION

The evaluation of network reliability [1] is important in engineering. Network reliability is an important parameter in both the design and operation of systems such as communication, power, and pipeline. Satyanarayana & Wood [8, 10] introduce a linear-time algorithm for computing the reliability of a network with an underlying series-parallel structure. This algorithm employs the new polygon-to-chain reductions. They extend the algorithm which enables the generation of a reduced network when the input network is not totally reducible.

This paper discusses the design and implementation of *PolyChain,* a portable FORTRAN program for K-terminal network reliability evaluation using polygon-to-chain reductions. Efficient data structures are implemented allowing fast reliability computation of large undirected networks. There are four output options which produce flexible, informative reports.

Section 2 briefly discusses some of the theoretical results of polygon-to-chain reductions. Section 3 describes the algorithm's implementation in FORTRAN. In section 4, *PolyChain* is applied to a small test problem illustrating the code's output. Several large networks are tested in section 5, showing the performance capabilities of the code. Conclusions and recommendations are made in section 6.

2. SERIES-PARALLEL GRAPHS AND POLYGON-TO-CHAIN REDUCTIONS

For a complete discussion of series-parallel graphs and polygon-to-chain reductions, see [8, 10]. This section defines the K-terminal network reliability problem and describes reductions that enable the evaluation of K-terminal network reliability.

Let $G \equiv (V, E)$ be a nonseparable graph with vertex set V and edge set E; vertices function perfectly and edges may fail independently of one another. Edge e_i has probability p_i of functioning and $q_i \equiv 1 - p_i$ of failing. $K \cup V$, $|K| \geq 2$, are distinguished vertices called the *K-vertices* of G. G_K is graph G with K specified. The *K-terminal reliability* of G_K, $R(G_K)$, is the probability that all K-vertices of G_K are connected by working edges.

The computation of the K-terminal reliability of a graph may require the application of the factoring algorithm [9]. Since the factoring algorithm has exponentially bounded time-complexity, it is desirable to reduce as much as possible the size of G_K prior to applying the factoring algorithm [11]. Reductions in the size of G_K can be obtained by applying *reliability-preserving* reductions, in which graph G_K is replaced by a reduced graph $G'_{K'}$, and $R(G_K) = \Omega R(G'_{K'})$, where Ω is a multiplicative factor that depends on the reduction applied. There are three well-known simple reliability-preserving reductions: parallel reduction, series reduction, and degree-2 reduction. In *parallel reduction*, parallel edges $e_a = uv$ and $e_b = uv$ are replaced by edge $e_c = uv$ with edge probability $p_c = 1 - q_a q_b$. In *series reduction,* edges $e_a = uv$ and $e_b = vw$, $u \neq w$, are replaced by edge $e_c = uw$ with probability $p_c = p_a p_b$. Let $u, v, w \in K$, $\deg(v) = 2$ and consider two edges $e_a = uv$ and $e_b = vw$, such that $u \neq w$. A *degree-2 reduction* substitutes edges e_a and e_b by edge $e_c = uw$ with $p_c = p_a p_b/(1 - q_a q_b)$ and $R(G_K) = (1 - q_a q_b)R(G_{K-v})$, where G_{K-v} is the new reduced graph.

The replacement of two series (parallel) edges by a single edge in a graph with no specified distinguished vertex set is called a *replacement*. A *series-parallel* graph is a graph that can be reduced to a single edge after successive series and parallel replacements. For a given series-parallel graph G, G_K need not be reduced to a single edge by successive simple reductions. G_K is said to be *series-parallel reducible* if it can be reduced to a single edge by successive simple reductions. It is *series-parallel irreducible* if it is not series-parallel reducible.

A *chain* χ is an alternating sequence of distinct vertices and edges, such that all vertices, except for the two endpoints, have degree 2. Let χ_1 and χ_2 be two distinct chains with common endpoints. $\triangle = \chi_1 \cup \chi_2$ constitutes a *polygon.*

The main result in [8, 10] is: Let G_K be a graph that admits no simple reduction. If G_K contains a polygon, it is one of seven types. A reliability preserving reduction permits the replacement of the polygons by chains with the

0018-9529/86/0400-0024$01.00

new edge reliability computed according to formulas given in appendix I. The above reductions are the *polygon-to-chain reductions.*

An $O(|E|)$ algorithm for reliability evaluation of a nonseparable series-parallel graph is presented in [8, 10]. The algorithm is described in appendix II. For a series-parallel graph the algorithm repeats two main steps until G_K is reduced to a single edge. The first step performs all simple reductions while the second finds a polygon and performs the corresponding polygon-to-chain reduction given in appendix I. An extension to the algorithm allowing all possible simple and polygon-to-chain reductions in a nonseries-parallel graph is also given in [8, 10]. This extension enables the algorithm to be used as a subroutine in a more general algorithm, such as the factoring algorithm, for computing network reliability. *PolyChain* is a direct implementation of that algorithm and the extension.

3. PROGRAM DESIGN AND IMPLEMENTATION

In this section, the code is briefly described. Programming is commented and data structures discussed. For a detailed description of program variables, *COMMON* blocks, and subroutines, see [6]. *PolyChain* is designed to be portable. All algorithmic routines are written in standard FORTRAN *77*. Only output-related code is system dependent. The program is modular, and has format free input and informative outputs.

The algorithm of Satyanarayana & Wood [8, 10] manipulates undirected networks. Undirected networks can have several representations in a computer code. Matrix representation is inefficient both with respect to core usage and execution times. These matrices are extremely sparse in practice, often having densities of less than 1%. For a static network, an efficient representation is a packed matrix. In this algorithm the network is dynamic and therefore linked lists are the most appropriate data structure. Helgason & Kennington [4], Thesen [7], Berztiss [2], among many, discuss efficient network representations using linked-list data-structures. Thesen [7] and Berztiss [2] discuss the implementation of linked lists in FORTRAN.

During the algorithm's reduction process, one or more edges or vertices are removed from the network. With respect to the data structure, this corresponds to removing elements from the linked lists. This process is repeated frequently in the algorithm. Doubly-linked lists [2, 5] use more core than lists with a single link, but are more efficient when many element deletions are required; they are implemented in *PolyChain*. Each vertex has a list of adjacent vertices, which besides indicating which vertices are adjacent to it, also provides information on whether the vertex and its adjacent vertices belong to set K. For every element of the list, there is a pointer indicating the address where information about the edge, corresponding to these two vertices, is stored. Figure 1 illustrates this multilist data structure for a small network.

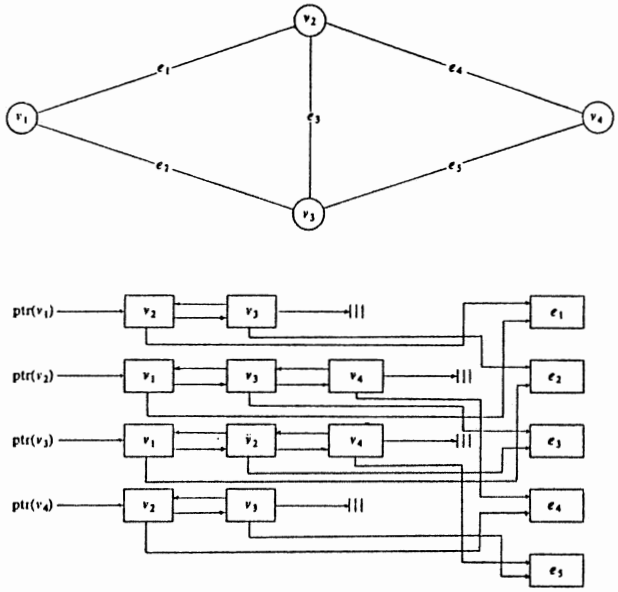
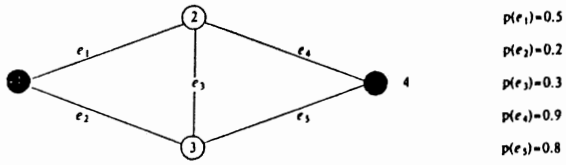


Fig. 1. Multilist data structure

Next, we describe the FORTRAN arrays used to implement the code's data structures. PTRADJ(i) points to the beginning of the list of vertices adjacent to vertex i. If PTRADJ(i) is positive, vertex i does not belong to set K. If PTRADJ(i) is negative, then vertex i is in set K. PTRADJ(i) may also be null. This indicates that vertex i has been removed from the network. Array ADJVRT(•) is the principle information element in the list structure. For each list it contains the vertices adjacent to the vertex that defines the list. If ADJVRT(•) is positive, that vertex is not in K, and if it is negative, the vertex is in the set K. LNKDWN(i) points to the next element downwards in the list of adjacent vertices. If LNKDWN(•) = 0, this element is the last element in the list. LNKUP(•) points to the element just above it in the adjacent vertices list. If LNKUP(•) = 0, then this element is the first in the list. EDGPRB(•) contains the edge reliability. Suppose vertex ADJVRT(j) is in the list pointed to by PTRADJ(i). Then LNKEDG(j) points to the position in array EDGPRB(•) corresponding to edge the links vertices i and ADJVRT(j). AVSADJ is a pointer to the beginning of the list of available space [2].

The algorithm also requires data structures for both the T list and the chain defined in [8, 10]. Since the order of the vertices in both the T list and the chain is irrelevant to the algorithm, linear stacks are used to represent both structures. TLIST(•) contains the stack of vertices that are elements of the T list. If TLIST(•) is positive, then that vertex is not in set K. If it is negative, the vertex is in set K. TTOP points to the top of the TLIST(•) stack. CHAIN(•) is a stack of vertices belonging to a given chain. TOP points to its top. As with TLIST(•), if CHAIN(•) is positive, then that vertex is not in set K. If it is negative, the vertex is in set K. ONLIST(i) = TRUE implies that vertex i is on the T list. DEG(i) indicates the degree of vertex i. Figure 2 illustrates the use of some of these arrays in network representation, where the darkened vertices are K-vertices.



i	1	2	3	4
ONLIST(i)	FALSE	TRUE	TRUE	FALSE
DEG(i)	2	3	3	2
PTRADJ(i)	-1	3	6	-9

i	ADJVRT(i)	LNKDWN(i)	LNKUP(i)	LNKEDG(i)
1	2	2	0	1
2	3	0	1	2
3	-1	4	0	1
4	3	5	3	3
5	-4	0	4	4
6	-1	7	0	2
7	2	8	6	3
8	-4	0	7	5
9	2	10	0	4
10	3	0	9	5

i	1	2	3	4	5
EDGPRB(i)	0.5	0.2	0.3	0.9	0.8

Fig. 2. FORTRAN Implementation of list structure

4. A TEST PROBLEM

PolyChain is applied to compute the reliability of a small network. For a detailed user-guide and an example of the application of the code to a series-parallel reducible network, see [6].

Consider a series-parallel irreducible network, the ARPA computer network, in figure 3. Vertices 1 and 21 are the network's *K*-vertices. Edge reliabilities are given alongside each edge.

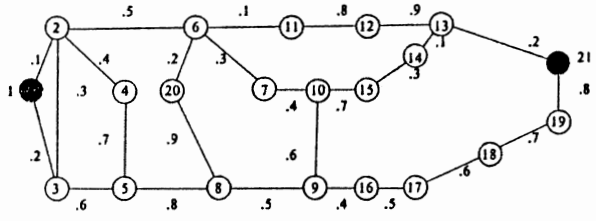


Fig. 3. ARPA Computer network

For series-parallel irreducible networks, *PolyChain* generates a 2-part report together with an output file containing the reduced network. The first section of the report, which is identical to the case of series-parallel reducible networks, contains the input network. The second part indicates that the network is series-parallel irreducible, contains the reduced network, the updated value of $M = \prod_j \Omega_j$, and summarizes the reductions performed, including percentage reductions in the network dimensions. CPU time, excluding I/O, is indicated. The 4-page report generated by the code for the ARPA computer network is given in appendix III. The reduced network obtained by *PolyChain* for the ARPA computer network is given in figure 4.

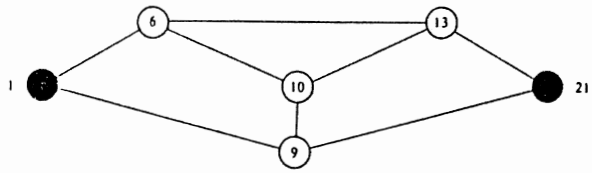


Fig. 4. Reduced network

For the series-parallel irreducible case, *PolyChain* generates a file with the updated value of M and the reduced network. This file can be used as input to a factoring algorithm program for reliability evaluation.

5. LARGE NETWORKS

Next, we test *PolyChain* with a number of large networks. These problems help understand *PolyChain's* behavior when treating real networks and enable the estimation of CPU times for large problems. A random generator was used to provide the series-parallel irreducible networks, while the series-parallel reducible networks were built aggregating several 150-vertex, 208-edge networks. Problems were run on the VAX 11/750 of the Etcheverry Hall VAX/UNIX CAE Laboratory, at Berkeley. The code was compiled on the UNIX f77 computer using the -O optimizing option. The code was run under Berkeley's BSD 4.2 UNIX operating system. CPU times were measured through the *dtime* system routine. Table 1 summarizes the networks tested.

TABLE I
Large-Scale Test Problems

Problem	Edges	Vertices	K-Vertices	Reducible?
1	208	150	63	Yes
2	418	300	126	Yes
3	838	600	252	Yes
4	1258	900	328	Yes
5	2518	1800	756	Yes
6	211	150	63	No
7	428	300	126	No
8	840	600	252	No
9	1272	900	328	No
10	2521	1800	756	No

The results obtained for these problems are contained table II.

Each test problem was run 10 times. The network size, $|E|$, processed per CPU second varied from 733 to 784 edges/sec on the series-parallel reducible networks tested and from 694 to 766 edges/sec on the series-parallel irreducible networks. The mean network size, $|E|$, processed per CPU second was, respectively, 775 and 725 edges/sec for the reducible and irreducible networks.

TABLE II
Test Results

Prob	Reduction Types										Mean CPU
	Ser	Deg2	I	II	III	IV	V	VI	VII	VIII	Time
1	87	56	8	24	3	0	5	0	1	17	0.27s
2	174	114	16	49	6	0	10	0	2	35	0.57s
3	348	230	32	99	12	0	20	0	4	71	1.09s
4	522	346	48	149	18	0	30	0	6	107	1.60s
5	1044	694	96	299	36	0	60	0	12	215	3.23s
6	52	9	0	1	0	1	0	0	0	3	0.30s
7	98	18	1	1	0	0	0	0	0	1	0.60s
8	216	50	0	2	0	0	0	0	0	4	1.14s
9	330	36	2	1	0	0	0	0	0	1	1.66s
10	583	101	0	2	1	0	0	0	0	1	3.59s

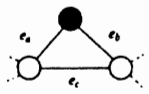
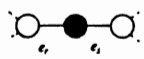
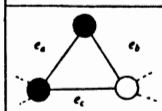
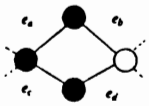
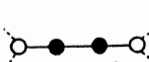
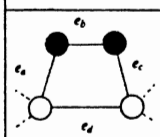
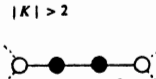
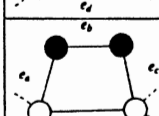
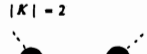
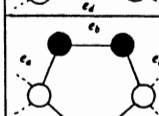
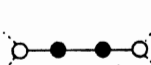
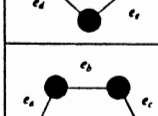
These first results show that *PolyChain* is feasible for computing K-terminal reliability of large series-parallel reducible networks. With 1 min. CPU on a VAX 11/750, one should be able to compute the K-terminal reliability of series-parallel reducible networks having upwards of 35000 edges.

6. CONCLUSION

The code's implementation facilitates further extensions and enhancements. It uses a multilist data structure representation of the network, which enables good core management and efficient network manipulation. Input is simple and format free. Outputs are detailed and a data file, containing the reduced network, is generated for use by another program, when the network is series-parallel irreducible. The code was tested on large networks and the results are encouraging.

The algorithm has a constraint on the topology of the input network. The network must be nonseparable. In the present version, *PolyChain* does not test for this requirement. Even [3] presents a depth-first-search based algorithm for determining all nonseparable components of a network. To insure the evaluation of the K-terminal network reliability for any network, this code should encorporate a factoring algorithm. Enhancements to the code, under development, include both nonseparability testing, and a factoring algorithm with optimal pivot selection [9].

APPENDIX I - POLYGON-TO-CHAIN REDUCTIONS

Polygon	Chain	New Edge Reliabilities	where
		$p_r = \delta/(\alpha+\delta)$ $p_t = \delta/(\beta+\delta)$ $\Omega = [(\alpha+\delta)(\beta+\delta)]/\delta$	$\alpha = q_a p_b q_c$ $\beta = p_a q_b q_c$ $\delta = p_a p_b p_c [1+(q_a/p_a)+(q_b/p_b)+(q_c/p_c)]$
		$p_r = \delta/(\alpha+\delta)$ $p_t = \delta/(\beta+\delta)$ $\Omega = [(\alpha+\delta)(\beta+\delta)]/\delta$	$\alpha = q_a p_b q_c$ $\beta = p_a q_b q_c$ $\delta = p_a p_b p_c [1+(q_a/p_a)+(q_b/p_b)+(q_c/p_c)]$
		$p_r = \delta/(\alpha+\delta)$ $p_t = \delta/(\beta+\delta)$ $\Omega = [(\alpha+\delta)(\beta+\delta)]/\delta$	$\alpha = p_a q_b q_c p_d + q_a p_b p_c q_d + q_a p_b q_c p_d$ $\beta = p_a q_b p_c q_d$ $\delta = p_a p_b p_c p_d [1+(q_a/p_a)+(q_b/p_b)+(q_c/p_c)+(q_d/p_d)]$
		$p_r = \gamma/(\alpha+\gamma)$ $p_t = \gamma/(\beta+\gamma)$ $p_l = \gamma/(\delta+\gamma)$ $\Omega = [(\alpha+\gamma)(\beta+\gamma)(\delta+\gamma)]/\gamma^2$	$\alpha = q_a p_b q_c p_d$ $\beta = p_a q_b q_c p_d + q_a p_b p_c q_d$ $\delta = p_a q_b p_c q_d$ $\gamma = p_a p_b p_c p_d [1+(q_a/p_a)+(q_b/p_b)+(q_c/p_c)+(q_d/p_d)]$
		$\lvert K\rvert > 2$ $p_r = \gamma/(\alpha+\gamma)$ $p_t = \gamma/(\beta+\gamma)$ $p_l = \gamma/(\delta+\gamma)$ $\Omega = [(\alpha+\gamma)(\beta+\gamma)(\delta+\gamma)]/\gamma^2$	$\alpha = q_a p_b q_c p_d$ $\beta = p_a q_b q_c p_d + q_a p_b p_c q_d$ $\delta = p_a q_b p_c q_d$ $\gamma = p_a p_b p_c p_d [1+(q_a/p_a)+(q_b/p_b)+(q_c/p_c)+(q_d/p_d)]$
		$\lvert K\rvert = 2$ $p_r = (p_b + p_a q_b p_c p_d)/(p_b + p_a q_b p_c)$ $p_t = \gamma/(\beta+\gamma)$ $p_l = \gamma/(\delta+\gamma)$ $\Omega = p_b + p_a q_b p_c$	$\alpha = q_a p_b q_c p_d$ $\beta = p_a q_b q_c p_d + q_a p_b p_c q_d$ $\delta = p_a q_b p_c q_d$ $\gamma = p_a p_b p_c p_d [1+(q_a/p_a)+(q_b/p_b)+(q_c/p_c)+(q_d/p_d)]$
		$p_r = \gamma/(\alpha+\gamma)$ $p_t = \gamma/(\beta+\gamma)$ $p_l = \gamma/(\delta+\gamma)$ $\Omega = [(\alpha+\gamma)(\beta+\gamma)(\delta+\gamma)]/\gamma^2$	$\alpha = q_a p_b p_c q_d p_e$ $\beta = p_a q_b p_c p_d q_e + p_a q_b p_c q_d p_e + q_a p_b p_c p_d q_e + p_a p_b q_c q_d p_e$ $\delta = p_a p_b q_c p_d q_e$ $\gamma = p_a p_b p_c p_d p_e [1+(q_a/p_a)+(q_b/p_b)+(q_c/p_c)+(q_d/p_d)+(q_e/p_e)]$
		$p_r = \gamma/(\alpha+\gamma)$ $p_t = \gamma/(\beta+\gamma)$ $p_l = \gamma/(\delta+\gamma)$ $\Omega = [(\alpha+\gamma)(\beta+\gamma)(\delta+\gamma)]/\gamma^2$	$\alpha = q_a p_b p_c q_d p_e p_f$ $\beta = p_a q_b p_c q_d e_e p_f + p_a q_b p_c p_d q_e p_f + p_a q_b p_c p_d p_e q_f + p_a p_b q_c p_d q_e p_f + p_a p_b q_c q_d p_e p_f + q_a p_b p_c p_d q_e p_f + q_a p_b p_c p_d p_e q_f$ $\delta = p_a p_b q_c p_d p_e q_f$ $\gamma = p_a p_b p_c p_d p_e p_f [1+(q_a/p_a)+(q_b/p_b)+(q_c/p_c)+(q_d/p_d)+(q_e/p_e)+(q_f/p_f)]$

Note: Dark vertices are K-vertices

APPENDIX II - DESCRIPTION OF ALGORITHM

Input: A nonseparable graph G with vertex set V, $|V| \geq 2$, edge set E, $|E| \geq 2$, and set $K \cup V$, $|K| \geq 2$. Edge reliability p_i for each edge $e_i \in E$.
Output: $R(G_K)$ if G is series-parallel or reduced K, G_K, and $M = \Omega_1 \times \Omega_2 \times \ldots \times \Omega_n$ if G is nonseries-parallel, where n is the number of reductions performed.

Begin
  $M \leftarrow 1$.
  Make all series reductions.
  Make all degree-2 reductions setting $M \leftarrow M \times \Omega$ for each reduction.
  Construct list $T \leftarrow \{v | v \in V$ and $deg(v) > 2\}$.
  Mark all $v \in T$ *onlist* and all $v \in V - T$ *offlist*.
  While $T \neq \emptyset$ and $|E| > 2$ **do**
  Begin
    Remove v from T.
    $i \leftarrow 1$ (Index of next chain out of v to be searched.)
    Until $i > 3$ or v is deleted or deg(v) = 2 **do**
    Begin
      $i \leftarrow i + 1$
      Search the $(i - 1)$-th chain out of v.
      If a polygon $\triangle(v, w)$ of type j is found **then do**
      Begin
        Apply the correct polygon-to-chain reduction to $\triangle(v, w)$ to obtain chain $\gamma(v, w)$.
        Let $M \leftarrow M \times \Omega_j$.
        $i \leftarrow i - 1$
        If deg(v) = 2 or deg(w) = 2 **then do**
        Begin
          Perform all possible series and degree-2 reductions on the chain (or cycle) containing subchain $\gamma(v, w)$ to obtain completely reduced chain $\gamma(x, y)$ or parallel edges e_{xy} and e_{xy}, setting $M \leftarrow M \times \Omega$ for each degree-2 reduction.
          If $y \neq v$ and y is *offlist* **then** mark y *onlist* and add y to T.
          If $x \neq v$ and x is *offlist* **then** mark x *onlist* and add x to T.
        End
      End
    End
  End
  If $|E| = 2$ **then** output $R(G_K) = M(1 - q_a q_b)$ where $e_a, e_b \in E$.
  Else do
  Begin
    For all vertices $v \in V$ such that deg(v) > 2 **do**
    Begin
      Perform polygon-to-chain reductions on all polygons $\triangle(v, w)$ found emanating from vertex v.
      Set $M \leftarrow M \times \Omega_j$ for each reduction performed, where j is the polygon type of (v, w).
    End
    Output reduced K, G_K, and M.
  End
End.

APPENDIX III - EXAMPLE OUTPUT

```
PolyChain - Version 83.1
Polygon to Chain Reductions
in Network Reliability
                                        Date : Fri Jul 26 1985
                                        Time : 12:58:26

  Input Network
```

Edge	Vertex	Type	Vertex	Type	Reliability
1	1	K	2	nK	1.00000000d+00
2	1	K	3	nK	1.20000000d+00
3	2	nK	3	nK	1.30000000d+00
4	2	nK	4	nK	1.40000000d+00
5	2	nK	6	nK	1.50000000d+00
6	3	nK	5	nK	1.60000000d+00
7	4	nK	5	nK	1.70000000d+00
8	5	nK	8	nK	1.80000000d+00
9	5	nK	20	nK	1.90000000d+00
10	6	nK	7	nK	1.10000000d+00
11	6	nK	11	nK	1.20000000d+00
12	6	nK	20	nK	1.30000000d+00
13	7	nK	10	nK	1.40000000d+00
14	8	nK	9	nK	1.50000000d+00
15	9	nK	10	nK	1.60000000d+00
16	9	nK	16	nK	1.70000000d+00
17	10	nK	15	nK	1.80000000d+00
18	11	nK	12	nK	1.90000000d+00
19	12	nK	13	nK	1.10000000d+00
20	13	nK	14	nK	1.20000000d+00

Page 1

```
PolyChain - Version 83.1
Polygon to Chain Reductions
in Network Reliability
                                        Date : Fri Jul 26 1985
                                        Time : 12:58:26

  Input Network
```

Edge	Vertex	Type	Vertex	Type	Reliability
21	13	nK	21	K	1.30000000d+00
22	14	nK	15	nK	1.40000000d+00
23	16	nK	17	nK	1.50000000d+00
24	17	nK	18	nK	1.60000000d+00
25	18	nK	19	nK	1.70000000d+00
26	19	nK	21	K	1.80000000d+00

```
Summary of Input Network Data

Number of Vertices..................   21
Number of Edges.....................   26
Number of K-Vertices................    2
Network Density..................... 0.124

Summary of Core Usage

        Variable          Current          Usage        %
          Name            Value

        MAXEDG             5000             26         0.5
        MAXVRT             2000             21         1.0
```

Page 2

```
PolyChain - Version 83.1
Polygon to Chain Reductions
in Network Reliability
                                    Date : Fri Jul 26 1985
                                    Time : 12:58:26

  Network Series-Parallel Irreducible
  Reduced Network
```

Edge	Vertex	Type	Vertex	Type	Reliability
14	1	K	9	nK	1.31422812d+00
5	1	K	6	nK	1.53817522d+00
19	6	nK	13	nK	1.18000000d-01
13	6	nK	10	nK	1.40000000d-01
26	9	nK	21	K	1.11760000d+00
15	9	nK	10	nK	1.60000000d+00
22	10	nK	13	nK	1.64000000d-01
21	13	nK	21	K	1.30000000d+00

Page 3

```
PolyChain - Version 83.1
Polygon to Chain Reductions
in Network Reliability
                                    Date : Fri Jul 26 1985
                                    Time : 12:58:26


Updated value of H =  0.21679720d+00


Reductions Performed

Series.....  15
Degree 2.....  0
Type 1.....  3
Type 2.....  0
Type 3.....  0
Type 4.....  0
Type 5.....  0
Type 6.....  0
Type 7.....  0
Type 8.....  0

                    Original      Reduced       % Reduction
                    Network       Network

Edges.....          26            8             69.2
Vertices.....       21            6             71.4
K-Vertices.....      2            2              0.

Solution Time =  0.02 Secs.
```

Page 4

7. ACKNOWLEDGMENT

This research has been partially supported by the US Army Research Office, under contract DAAG29-81-K-0160, and Conselho Nacional de Desenvolvimento Científico e Tecnológico-CNPq, Brazil.

8. REFERENCES

[1] A. Agrawal, R. E. Barlow, "A survey of network reliability and domination theory", *Operations Research,* vol 32, no. 3, 1984, pp 478-492.

[2] A. T. Berztiss, *Data Structures: Theory and Practice,* Academic Press, 1975.

[3] S. Even, *Graph Algorithms,* Computer Science Press, 1979, pp 57-62.

[4] R. V. Helgason, J. L. Kennington, *Algorithms for Network Programming,* John Wiley & Sons, 1980.

[5] D. E. Knuth, *The Art of Computer Programming, Volume One: Fundamental Algorithms,* Addison-Wesley, 1973.

[6] M. G. C. Resende, "A computer program for reliability evaluation of large-scale undirected networks via polygon-to-chain reductions", Report ORC-83-10, 1983. Available from: Operations Research Center; University of California; Berkeley, California 94720 USA.

[7] A. Thesen, *Computer Methods in Operations Research,* Academic Press, 1978.

[8] A. Satyanarayana, R. K. Wood, "Polygon-to-chain reductions and network reliability", Report ORC 82-4, 1982. See [6].

[9] A. Satyanarayana, M. K. Chang, "Network reliability and the factoring theorem", *Networks,* vol 13, 1983, pp 107-120.

[10] R. K. Wood, "Polygon-to-chain reductions and extensions for reliability evaluation of undirected networks", PhD thesis, Dept. of Industrial Engineering and Operations Research, University of California, Berkeley, 1982.

[11] R. K. Wood, "A factoring algorithm using polygon-to-chain reductions for computing *K*-terminal network reliability", *Networks,* vol 15, 1985, pp 173-190.

AUTHOR

Mauricio G. C. Resende; Operations Research Center; University of California; Berkeley, California 94720 USA.

Mauricio G. C. Resende was born in Maceio, Brazil, on 1955 July 27. He received the degree of Electrical Engineer, with concentration in systems, from the Catholic University of Rio de Janeiro in 1978, and an MS in Operations Research from the Georgia Institute of Technology in 1979. Mr. Resende was an operations research analyst with the Furnas Power Company in Rio de Janeiro from 1979 to 1982. He also held a teaching position at the Catholic University of Rio de Janeiro from 1981 to 1982. He is a PhD candidate in operations research at the University of California at Berkeley.

Manuscript TR83-144 received 1983 September 24; revised 1985 October 21.

★ ★ ★

LIST OF REFEREES

(continued from page 18)

Richard C. Terzian □ TRW Defense & Space-Systems Group; Redondo Beach
L. C. Thomas □ University of Manchester; Manchester
Marlin U. Thomas □ Cleveland State University; Cleveland
William E. Thompson □ Naval Research Laboratory; Washington, DC
K. Toguchi □ Mitsubishi Kakoki Kaisha, Ltd.; Kawasaki
Martin Trachtenberg □ RCA; Moorestown
Ashok K. Trivedi □ Northern Telecom; Richardson
Masaaki Tsujitani □ Kobe Women's College; Hyogo
Steven S. Tung □ Hughes Aircraft Company; Culver City
Lonnie C. Vance □ General Motors Research Laboratories; Warren
P. Venkatachalam □ Indian Institute of Technology; Bombay
John Verrall □ The City University; London
Julio Vilar □ Inst. de Investigaciones Electricas; Cuernavaca
Jim Von Bank □ Control Data Corporation; Minneapolis
Mirko Vujosevic □ Mihailo Pupin Institute; Belgrade
Haskell A. Walker □ Harris Corp.; Melbourne
Duan Wei □ Bureau of Statistics; Taipei
Mike West □ University of Warwick; Coventry
Alan Winterbottom □ The City University; London
N. Keith Womer □ Clemson University; Clemson
Kam Wong □ Kambea Industries; Manhattan Beach
Thomas G. Woo □ Consultant; Medfield
Alan P. Wood □ ESL Inc.; Sunnyvale
Wenxin Xu □ Fujian Provincial Computing Centre; Fuzhow City
Janan Xue □ Beijing Light Industry Institute; Beijing
Shigeru Yamada □ Okayama University of Science; Okayama
Soung R. Yee □ Samsung Semiconductor & Telecom. Co. Ltd.; Buchun
John Yuan □ National Tsing Hua University; Hsinchu
Antonio Zanini □ University of Firenze; Firenze