[12] D.D. Harms and C.J. Colbourn, "Renormalization of Two-Terminal Reliability", *Networks*, to appear.

[13] V.A. Kaustov, Y.I. Litvak and I.A. Ushakov, "The Computational Effectiveness of Reliability Estimates by the Method of Nonedge-Intersecting Chains and Cuts", *Soviet J. Comput. Systems Sci.* 24 (1986), 70-73.

[14] C.G. Lekkerkerker and J.C. Boland, "Representation of a Finite Graph by a Set of Intervals on the Real Line", *Fund. Math.* 51 (1962), 45-64.

[15] L.D. Nel and H.J. Strayer, "Two-Terminal Reliability Bounds based on Edge Packing by Cutsets", *J. Comb. Math. Comb. Comp.* (to appear)

[16] J.S. Provan, "The Complexity of Reliability Computations in Planar and Acyclic Graphs", *SIAM J. Computing 15* (1986), 694-702.

[17] J.G. Shanthikumar, "Reliability of Systems with Consecutive Minimal Cutsets", *IEEE Trans. on Reliab.* 36 (1987), 546-550.

[18] J.G. Shanthikumar, "Bounding Network Reliability Using Consecutive Minimal Cutsets", *IEEE Trans. on Reliab.* 37 (1988), 45-49.

[19] D.R. Shier, "Algebraic aspects of computing network reliability", in: *Applications of Discrete Mathematics* (R.D. Ringeisen, F.S. Roberts; eds.) SIAM, Philadelphia, 1988, pp. 135–147.

[20] D.R. Shier, *Algebraic Structures and Network Reliability*, Clarendon Press, Oxford, 1991.

[21] H.J. Strayer and C.J. Colbourn, "Bounding Two-terminal Network Reliability via Surface Duality", *Proc. CLAIO VI, Mexico City, 1992, to appear.*.

[22] H.J. Strayer and C.J. Colbourn, "Bounding Network Reliability via Surface Duality", preprint, 1992.

[23] C.-L. Yang and P. Kubat, "An algorithm for network reliability bounds", *ORSA J. Comp.* 2 (1990), 336–345.

by the c2C, spl, and c2P methods can yield an adverse effect on the convergence.

In summary, the consecutive path bound appears, on the basis of the results reported here, to be a valuable tool in connectivity analysis.

Acknowledgments

Thanks to Daryl Harms for useful comments (and code).

References

[1] H.M. AboElFotoh and C.J. Colbourn, "Series-Parallel Bounds for the Two-Terminal Reliability Problem", *ORSA J. Comp* 1 (1988), 209-222.

[2] H.M. AboElFotoh and C.J. Colbourn, "Efficient Algorithms for Computing the Reliability of Permutation and Interval Graphs", *Networks* 20 (1990), 883-898.

[3] M.O. Ball, J.S. Provan and D.R. Shier, "Reliability covering problems", *Networks* 21 (1991), 345–358.

[4] T.B. Brecht and C.J. Colbourn, "Lower Bounds on Two-Terminal Network Reliability", *Discrete Appl. Math* 21 (1988), 185-198.

[5] C.J. Colbourn, *The Combinatorics of Network Reliability*, Oxford University Press, Oxford, 1987.

[6] C.J. Colbourn, "Edge-Packings of Graphs and Network Reliability", *Discrete Math.* 72 (1988), 49-61.

[7] C.J. Colbourn and D.D. Harms, "Evaluating Performability: Most Probable States and Bounds", *Technical Report CMPT TR92-06*, Simon Fraser University, Burnaby B.C., Canada, 1992.

[8] C.J. Colbourn, L.D. Nel, T.B. Boffey and D.F. Yates, "Network Reliability and the Probabilistic Estimation of Damage from Fire Spread", *Annals Oper. Res.*, to appear.

[9] D.R. Fulkerson and O.A. Gross, "Incidence Matrices and Interval Graphs", *Pacific J. Math.* 15 (1965), 835-855.

[10] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, London, 1980.

[11] D.D. Harms, *A Symbolic Algebra Environment for Research in Network Reliability*, Ph.D. Thesis, Simon Fraser University, 1992.

Bound	Number of States
total	2361183241434822606848
edc	1212988900292060774400
c2C	765651602982360317952
c2P	20165568859730340864
spl	18437359511879548928
edp	14103587382087385088
path	1487160960
cut	2557

Table 6: Numbers of States Determined By Bounds

Bounds Used	0 states	500 states	1000 states
none	.0000000	.8691955	.9018692
	1.0000000	1.0000000	1.0000000
cut, path	.8713729	.8927863	.9171433
	1.0000000	1.0000000	1.0000000
cut, path, c2C	.8713729	.8817954	.9094521
	.9998408	.9998408	.9998408
cut, path, c2C, edp	.9944427	.9956679	.9961680
	.9998408	.9998366	.9998363
cut, path, spl	.9973942	.9980692	.9983065
	1.0000000	.9999800	.9999307
cut, path, spl, edc	.9973942	.9980692	.9983065
	.9998893	.9998819	.9998819
cut, path, c2P	.9985832	.9994056	.9994442
	1.0000000	.9999231	.9998756
cut, path, c2P, edc	.9985832	.9994275	.9994498
	.9998893	.9998556	.9998487

Table 7: Most Probable States with Bounds: SP Reduced Arpa

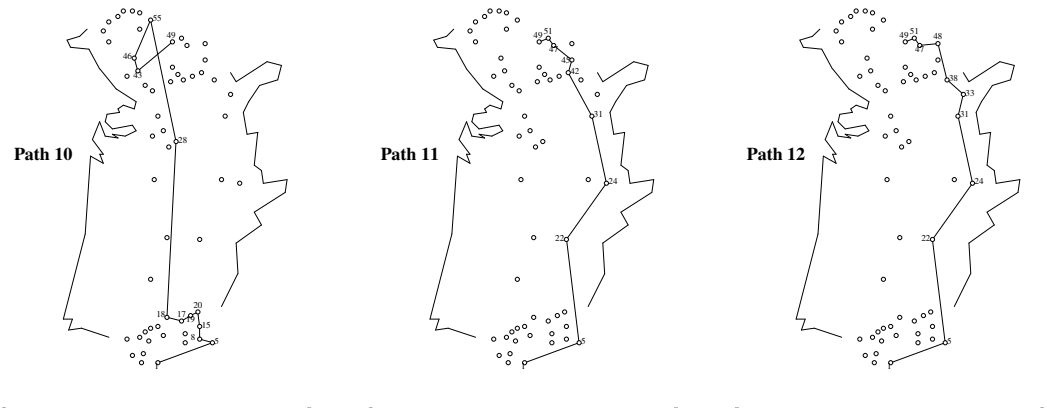

Figure 7: The 12 consecutive paths (continued)

Bounds Used	0 states	500 states	1000 states
none	.0000000	.6473949	.7041199
	1.0000000	1.0000000	1.0000000
cut, path	.8333055	.8350209	.8364256
	1.0000000	1.0000000	1.0000000
cut, path, c2C	.8333055	.8343601	.8352790
	.9998987	.9998987	.9998987
cut, path, c2C, edp	.9947361	.9948940	.9949963
	.9998987	.9998983	.9998977
cut, path, spl	.9975725	.9977014	.9977781
	1.0000000	.9999784	.9999784
cut, path, spl, edc	.9975725	.9977016	.9977783
	.9999701	.9999558	.9999556
cut, path, c2P	.9983317	.9989160	.9990610
	1.0000000	.9999257	.9999235
cut, path, c2P, edc	.9983317	.9989154	.9990360
	.9999701	.9999063	.9999050

Table 5: Most Probable States with Bounds: Arpa

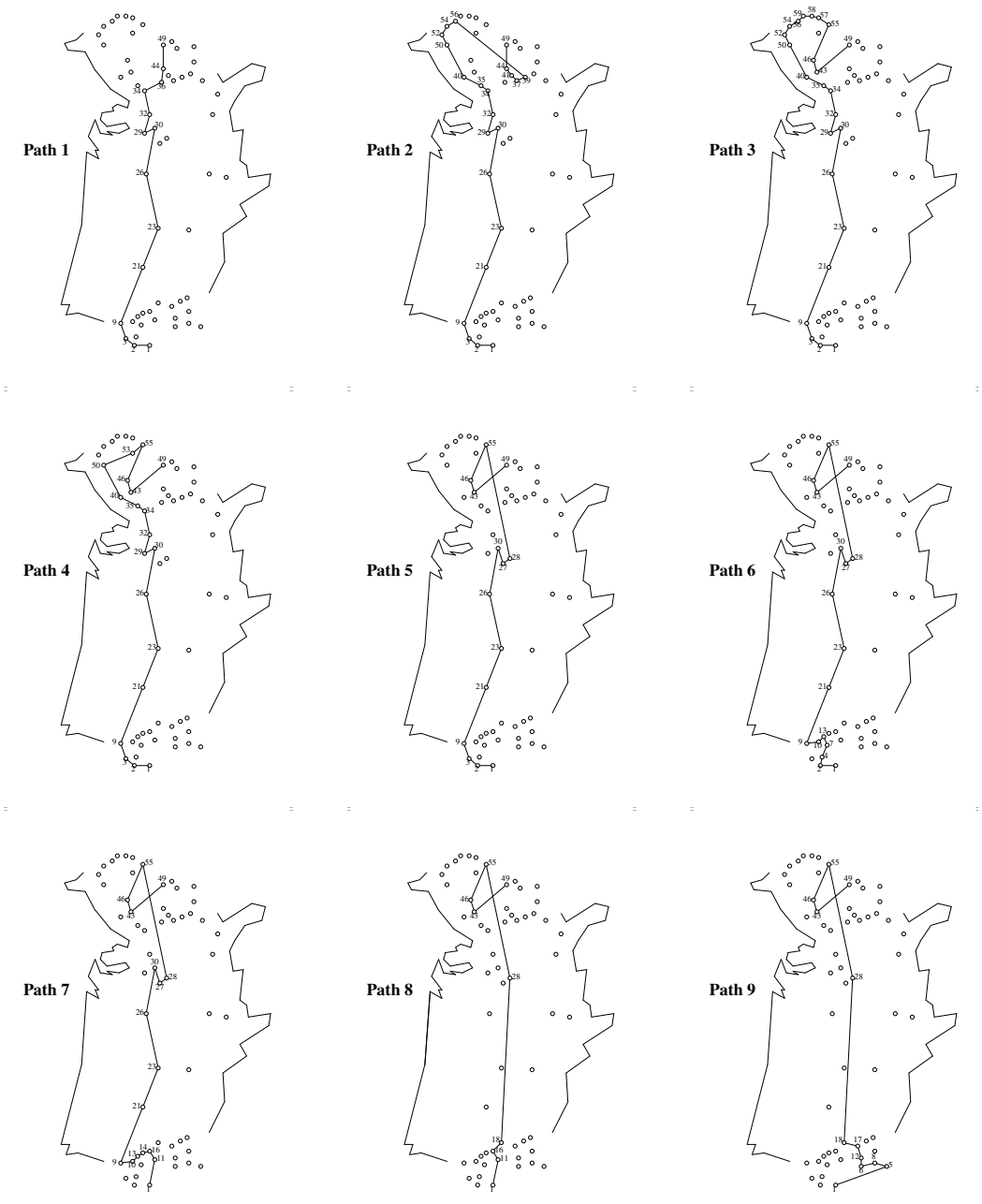

Figure 6: The 12 consecutive paths

In a recent paper, we establish that upper bounds can be used via surface duality to provide lower bounds on two–terminal reliability [22]. Thus we also include the dual versions of the more competitive static upper bounds, including renormalization [12] (dual ren), dual consecutive cuts [17, 18] (dual c2C), dual series–parallel [1] (dual spu) and dual non–crossing cuts [8] (dual nc$k$) bounds.

In our few test cases given here, the best results are always obtained by the consecutive paths bound or by dual renormalization. The series–parallel lower bound does, on occasion, deliver the tightest bound. However, in our test cases, the consecutive path bound appears to deliver a better bound than is typically available from any series-parallel approximation of the original network.

One of the more promising features of the consecutive cut and path bounds is their potential use in conjunction with most probable state techniques. Colbourn and Harms [7] have already reported on the merits of consecutive cut bounds in this context. We implemented the consecutive path bound in conjunction with the modified Yang–Kubat algorithm. In order to perform a fair test of the method, we employ the 1979 Arpanet with terminals 1 and 49. Rather than setting all edge probabilities equal, which is disadvantageous to most probable state techniques not incorporating bounds, we chose edge probabilities uniformly distributed between .975 and .985. We then examined the use of the basic Yang–Kubat algorithm, and the algorithm supplemented with various bounds. We employed the simple *path* bound based on the fact that the shortest path from 1 to 49 has eight edges, and the simple *cut* bound that the minimum cut size is three. We also employed the series–parallel lower bound (*spl*), the edge-disjoint path (*edp*) and edge–disjoint cut (*edc*) bounds, and the consecutive path (*c2P*) and consecutive cut (*c2C*) bounds. For the consecutive paths bound, we employed a set of twelve consecutive paths between nodes 1 and 49; the actual paths used are shown in Figures 6 and 7.

In our tests, the amount of time required to calculate one of these bounds never exceeded the time to generate twenty states by the Yang–Kubat algorithm. Moreover, the time to generate a state after the bounds are incorporated in the most probable states algorithm was increased only slightly (less than 1%) in our tests. Thus we simply report, for each test case, the lower and upper bound that the most probable state technique achieved after generating 0, 500, and 1000 states. Table 5 reports the results.

The critical ingredient in the successful application of a bound to improve the generation of most probable state generation appears to be not just that the bound determine the operational status of many network states, but also that those states include many of the most probable states. In Table 6, we show the number of network states whose state is determined by each of the bounds used, for the same example computation as in Table 5. The *total* entry is the total number of states, 2^{71}.

Finally, we repeated the generation of most probable states after undertaking all series and parallel reductions on the 1979 Arpanet. The resulting network has 20 nodes and 31 edges; it is nonplanar. We report results for the same node pair, $\{1, 49\}$, in Table 7. It is instructive to note that, in the application to the series-parallel reduced Arpa, the edge probabilities vary more widely as a result of the reductions; thus the edge orderings forced

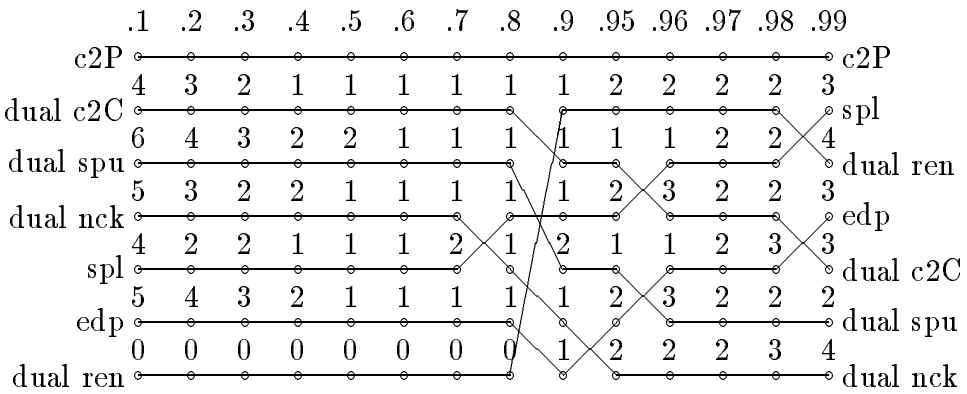

Figure 5: Graphical lower 2-terminal reliability bounds for the 1979 Arpanet, $s = 5$, $t = 56$

p	spl	edp	c2P	dual spu	dual c2C	dual nc$k$	dual ren
.3	.0007313	.0007307	**.0007392**	.0007367	.0007371	.0007349	.0000077
.4	.0041576	.0041379	**.0043069**	.0042454	.0042556	.0042037	.0001502
.5	.0163851	.0161131	**.0178184**	.0171320	.0172603	.0166603	.0015400
.6	.0524439	.0502754	**.0610365**	.0565011	.0574906	.0533322	.0105522
.7	.1482769	.1371079	**.1825986**	.1626568	.1680353	.1496750	.0555524
.8	.3770204	.3407131	**.4607487**	.4027063	.4225670	.3704681	.2643210
.9	.7811348	.7321002	**.8562712**	.7785587	.8112719	.7543486	.8201309
.95	.9536628	.9335328	**.9768067**	.9384753	.9563333	.9323257	.9729475
.96	.9737323	.9606726	**.9877774**	.9603024	.9734083	.9568249	.9848795
.97	.9877573	.9807804	**.9947502**	.9776152	.9860032	.9760643	.9926410
.98	.9960033	.9933876	**.9984372**	.9900911	.9942970	.9896622	.9971960
.99	.9994514	.9990380	**.9998068**	.9975510	.9987262	.9975331	.9994048

Table 4: Tabular lower 2-terminal reliability bounds for the 1979 Arpanet, $s = 5$, $t = 56$

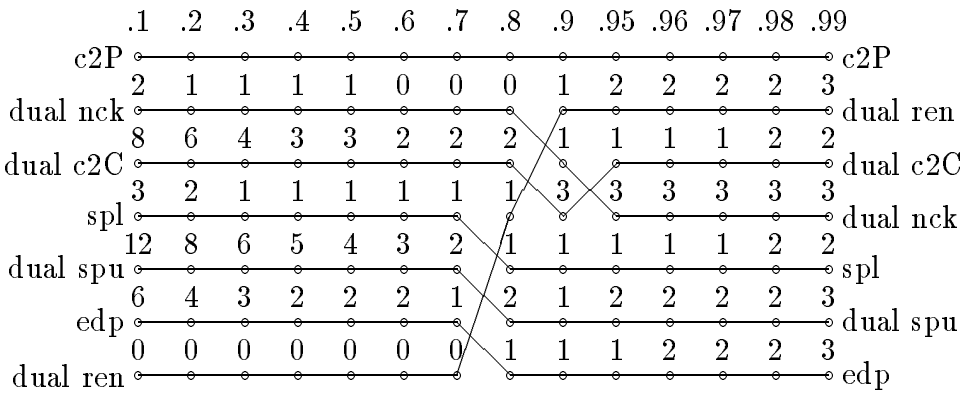

Figure 4: Graphical lower 2-terminal reliability bounds for the 1979 Arpanet, $s = 20$, $t = 41$

p	spl	edp	c2P	dual spu	dual c2C	dual nc$k$	dual ren
.3	.0000118	.0000118	**.0000123**	.0000118	.0000119	.0000119	.0000060
.4	.0002099	.0002097	**.0002291**	.0002099	.0002156	.0002156	.0001118
.5	.0019599	.0019521	**.0022537**	.0019597	.0020526	.0020537	.0011178
.6	.0121741	.0120566	**.0148704**	.0121688	.0130317	.0130520	.0078609
.7	.0567839	.0556971	**.0731916**	.0566659	.0618679	.0620687	.0450263
.8	.2096305	.2032191	**.2728641**	.2082441	.2288572	.2297486	.2130434
.9	.5946042	.5757802	**.7013583**	.5878493	.6287570	.6291905	.6685376
.95	.8545479	.8389879	**.9149396**	.8478182	.8752900	.8745993	.9070316
.96	.9002719	.8876628	**.9450644**	.8946397	.9159448	.9152270	.9401368
.97	.9400237	.9310539	**.9690358**	.9358925	.9503814	.9497637	.9663271
.98	.9715729	.9665376	**.9863150**	.9691856	.9769516	.9765483	.9851278
.99	.9924450	.9908572	**.9966255**	.9916713	.9940064	.9938626	.9963290

Table 3: Tabular lower 2-terminal reliability bounds for the 1979 Arpanet, $s = 20$, $t = 41$

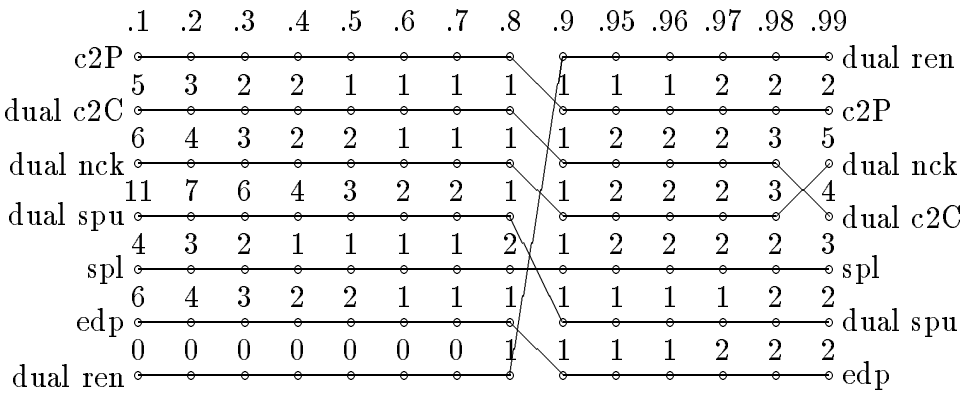

Figure 3: Graphical lower 2-terminal reliability bounds for the 1979 Arpanet, $s = 1$, $t = 59$

p	spl	edp	c2P	dual spu	dual c2C	dual nc$k$	dual ren
.3	.0001312	.0001312	**.0001320**	.0001318	.0001318	.0001318	.0000679
.4	.0013129	.0013102	**.0013382**	.0013296	.0013314	.0013296	.0007154
.5	.0078563	.0077972	**.0082176**	.0080594	.0080982	.0080613	.0046383
.6	.0340291	.0333102	**.0367110**	.0353242	.0358198	.0353850	.0222991
.7	.1174355	.1119727	**.1323873**	.1216841	.1259199	.1227107	.0870798
.8	.3332342	.3073968	**.3772532**	.3360230	.3586441	.3453954	.3015846
.9	.7322978	.6756324	.7789235	.7136485	.7635615	.7466581	**.8036517**
.95	.9255995	.8867141	.9422578	.9068495	.9381943	.9330353	**.9648106**
.96	.9526123	.9223762	.9633181	.9372406	.9609452	.9580246	**.9793603**
.97	.9738407	.9532330	.9796413	.9628520	.9785067	.9772697	**.9893877**
.98	.9887938	.9777283	.9911255	.9826330	.9907468	.9904898	**.9956929**
.99	.9973635	.9940316	.9978362	.9954343	.9977832	.9978283	**.9990163**

Table 2: Tabular lower 2-terminal reliability bounds for the 1979 Arpanet, $s = 1$, $t = 59$

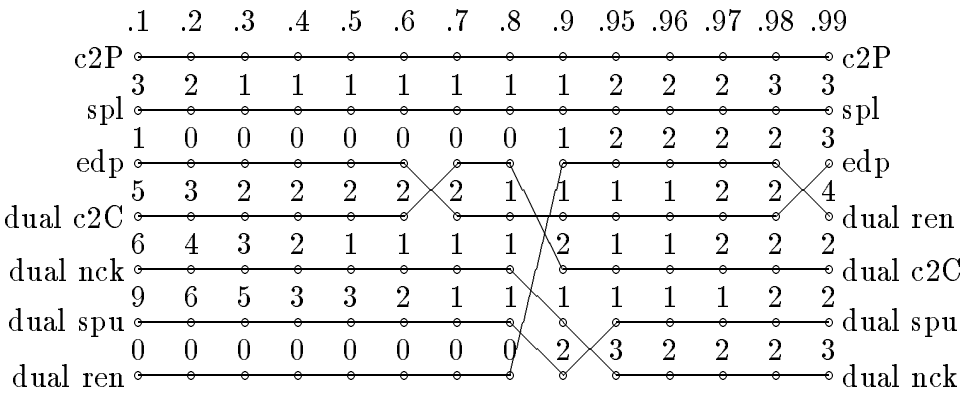

Figure 2: Graphical lower 2-terminal reliability bounds for the 1979 Arpanet, $s = 1$, $t = 49$

p	spl	edp	c2P	dual spu	dual c2C	dual nc$k$	dual ren
.3	.0001049	.0000854	**.0001066**	.0000852	.0000853	.0000852	.0000196
.4	.0011796	.0009240	**.0012186**	.0009173	.0009210	.0009174	.0002627
.5	.0078085	.0059731	**.0082284**	.0058522	.0059420	.0058577	.0019764
.6	.0367784	.0279757	**.0395020**	.0267265	.0278952	.0268463	.0105311
.7	.1348309	.1044372	**.1464656**	.0961654	.1053364	.0974188	.0462366
.8	.3906123	.3190833	**.4209485**	.2851360	.3253376	.2909274	.2036759
.9	.8124641	.7397975	**.8401440**	.6767808	.7383872	.6822952	.7635139
.95	.9652984	.9394351	**.9733213**	.8994119	.9269133	.8988280	.9615655
.96	.9810166	.9647239	**.9857761**	.9339633	.9527999	.9327338	.9782835
.97	.9914938	.9830459	**.9937995**	.9622760	.9733428	.9608924	.9893587
.98	.9973417	.9942692	**.9981184**	.9831884	.9881673	.9821449	.9959437
.99	.9996524	.9991816	**.9997615**	.9958547	.9970621	.9954482	.9991468

Table 1: Tabular lower 2-terminal reliability bounds for the 1979 Arpanet, $s = 1$, $t = 49$

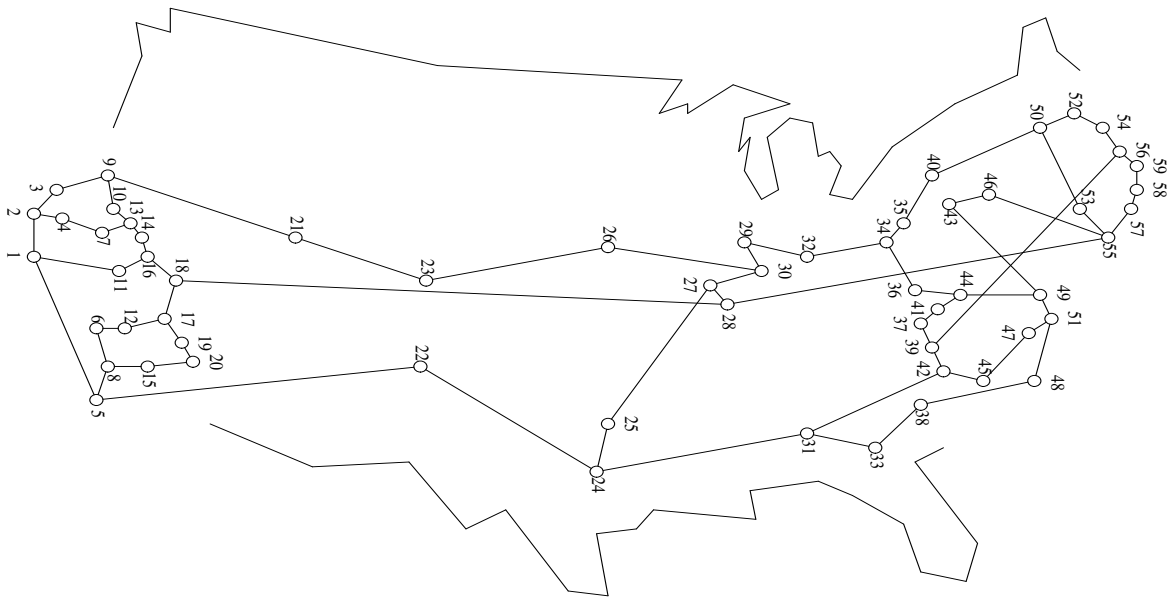

Figure 1: The 1979 Arpanet

7 Examples

We report only ck-paths lower bound results in this section as there exist many ck-cuts upper bound results already in the literature [11, 17, 18]. Algorithm B is used to generate the set of ck-paths. New frontier pathset size minimization is used as the mutation path selection heuristic.

We use the 1979 Arpanet with equal edge probabilities and a terminal set size of 2 because there are already numerous published results with which to compare [5, 6, 13, 22]. The terminal pairs we analyze are $(1, 49)$, $(1, 59)$, $(5, 56)$, and $(20, 41)$.

We give both a graphical and tabular presentation of the computational results for each of the four terminal pairs. The graph drawn shows the ordering of the bounds with respect to the accuracy for fourteen values of p, with the most accurate at the top. In order to reflect how close the bounds are, we report for each pair of bounds that are adjacent in the ordering an integer α, which is larger when the bounds are "close" than when they are "far apart". More specifically, if two bounds A, B with $A > B$ are compared, the number α is chosen so that $e^{-(\alpha-1)}A \leq A - B \leq e^{-\alpha}A$. Thus α indicates the order of magnitude of the relative error between two adjacent bounds.

To permit other comparisons, we also present the numerical results themselves to six digit accuracy, although all were computed to at least 16 digits of accuracy. We suppress the results for $p = .1$ and $p = .2$ in the tables.

Computational results have indicated that the best direct two–terminal lower bound techniques are series-parallel (spl) [1] and edge-disjoint paths (edp) [4, 13]. Hence these are the two static lower bounds that we include in our tables along with the c2P lower bound (c2P).

6 Use of ck-Bounds with the Yang-Kubat Method

Let $\kappa^r = \{C_1, C_2, ..., C_r\}$ be a consecutively ordered set of ck-cuts, and let G_{κ^r} be its associated interval graph. Number the edges so that all edges in $C_i \setminus \cup_{j=1}^{i-1} C_j$ have labels larger than any edge in $\cup_{j=1}^{i-1} C_j \setminus \cup_{j=i+1}^{r} C_j$. As mentioned earlier this defines an $s-t$ VPIN for G_{κ^r}. It also has the property that if C_y is the highest numbered ck-cut containing edge e_i, C_z the highest numbered ck-cut containing edge e_j, and $y < z$, then $i \leq j$.

We precompute and store a table of values $\gamma(i,j) = \vartheta(e_y)$ where y is the smallest index greater than j of an edge contained in ck-cut C_i. If no such edge exists then $\gamma(i,j) = \infty$. Also we predetermine vectors α and β where $\alpha(i)$ is the smallest index of a ck-cut in κ^r containing edge e_i, and $\beta(i)$ is the highest index of a ck-cut in κ^r containing e_i. All of these values can be obtained efficiently.

In the modified Yang-Kubat algorithm described in [7] a tree is constructed containing variable length binary strings $\sigma = (x_1 x_2 ... x_j)$. Each node's string represents the edges whose inclusion/exclusion status have been fixed. If edge e_i is included then $x_i = 1$ and if e_i is to be excluded then $x_i = 0$. Also associated with each node $\sigma = (x_1 x_2 ... x_j)$ are eight labels used to generate their off-spring. The ck–cuts can be used to provide upper bounds, and the ck–paths to provide lower bounds.

In order to incorporate lower bounds, we first choose a consecutive set $(P_1, \ldots, P_s)$ of paths, and an edge ordering $e_1, \ldots, e_m$. For each edge e_i, define $f_i = \min(j : e_i \in P_j)$, with $f_i = s+1$ if e_i appears on none of the paths. Then define $\ell_i = \max(j : e_i \in P_j)$, setting $\ell_i = s$ if e_i appears on none of the paths. Our edge ordering is chosen so that for all $1 \leq i < m$, either $f_i < f_{i+1}$, or $f_i = f_{i+1}$ and $\ell_i < \ell_{i+1}$.

To incorporate the consecutive path lower bound, we precompute a table $\Gamma_{i,j}$ of probabilities for $1 \leq i \leq s+1$ and $1 \leq j \leq m+1$. $\Gamma_{i,j}$ is the probability that all of paths $P_i, \ldots, P_s$ contain at least one failed edge from $\{e_j, \ldots, e_m\}$. A simple polynomial time algorithm for computing these measures is available, as follows.

1. Set $\Gamma_{s+1,m+1} = 1$, and $\Gamma_{i,m+1} = 0$ for $1 \leq i \leq s$.

2. for j from m to 1,

   (a) set $\Gamma_{i,j} = \Gamma_{i,j+1}$ $(1 \leq i \leq s+1)$.

   (b) set $\Gamma_{i,j} = \Gamma_{i,j} + (1-p_j)(\Gamma_{\ell_j+1,j+1} - \Gamma_{i,j+1})$ for $f_j \leq i \leq \ell_j$.

It is easily checked that $1 - \Gamma_{1,1}$ is precisely the ck–paths lower bound. Now employing this precomputed table for Γ in the modified Yang–Kubat method is straightforward. We need only maintain a single index λ. Initially, $\lambda = 0$. Edges are treated in order from e_1 to e_m. If the edge is contracted in generating the next state, λ is unchanged; if edge e_i is deleted, however, λ is replaced by $\max(\lambda, \ell_i)$. In this way, the lower bound as conditioned by the state of edges $e_1, \ldots, e_i$ is always available by a simple table lookup: it is $1 - \Gamma_{\lambda+1,i+1}$. Once the precomputation is done, the overhead is negligible, as only a single computation of a maximum is needed when each edge is processed. No additional storage beyond the $(s+1)(m+1)$ table entries in Γ, and the single integer λ, is involved.

Algorithm A

1. Set $x = 1$, $N\Gamma = \emptyset$
2. Set $y = 0$, $P_{x,y} = P_x$, $N\Gamma = N\Gamma \cup P_{x,y}$. $P_{x,y}$ is a *frontier pathset.*
3. Look for some path $P = v_i \rightarrow v_j$ in $E - N\Gamma$, where v_i and v_j are vertices in $P_{x,y}$. P is the *mutation path.* If no such path exists go to step 4. Otherwise set $y = y + 1$, and set the new frontier pathset $P_{x,y} = P_{x,y-1} \cup P$. Remove unnecessary edges from $P_{x,y} \backslash P$ by eliminating cycles and removing edges connecting non-terminal nodes to the subgraph $P_{x,y}$. Set $N\Gamma = N\Gamma \cup P_{x,y}$. Repeat this step.
4. If $x + 1 > a$ then stop. Otherwise look for some minimal *linking pathset* P which uses a portion of $P_{x,y}$, a portion of P_{x+1} and perhaps a subset of edges from $E - N\Gamma$. If such a P exists then set $y = y + 1$, $P_{x,y} = P$, $N\Gamma = N\Gamma \cup P_{x,y}$.
5. Set $x = x + 1$ and go to step 2.

In step 3 there are often several possible mutation paths so one can use some heuristic to choose amongst them. Such a heuristic might be to select the mutation path P which minimizes the size of the new frontier pathset $P_{x,y}$ being constructed. Of course any mutation path selection heuristic must be polynomial. Heuristics can also be used in determining linking pathsets.

The problem with Algorithm A is that it is highly dependent on the initial ordering selected for the original set of edge-disjoint k-paths. To minimize the effect due to ordering one can maintain a frontier pathset for each edge-disjoint k-path. The best mutation path for each frontier is found and the best of these is used to mutate its corresponding frontier pathset to form a new frontier pathset. The new frontier pathset is added to $N\Gamma$ and the process is repeated. When no more new frontier paths can be generated, linking pathsets are searched for between each current frontier pathset and its successor edge-disjoint k-paths. Denote this algorithm as Algorithm B. The initial ordering of the edge-disjoint k-paths still influences the linking pathsets generated.

It should be pointed out that there are other efficient heuristics for generate consecutive ck-cuts (ck-paths) which do not begin by finding a set of edge-disjoint (k-cuts) (k-paths) [8, 17, 18].

ck-path in ν^r has all of its edges operating at any instant of time. In other words, it is the probability that at no time does every ck-path contain a failed edge.

With each P_i we associate the open interval $[i-1, i]$. For each edge $e \in \cup_{i=1}^{r} E(P_i)$ we associate a segment which spans all intervals corresponding to ck-paths in ν^r which contain e. Since the ck-paths are in a consecutive ordering these intervals are contiguous. We close the endpoints of each segment so that they completely span their corresponding set of ck-path intervals. Thus if some edge e_1 is contained in ck-paths P_i through P_{i+j} and another edge e_2 is contained in ck-paths P_{i+j+1} through P_{i+j+k}, then the segments representing e_1 and e_2 overlap at $i+j$. Lastly, add an s^* segment spanning the interval $[-.5, .5]$ and a t^* segment spanning the interval $[r-.5, r+.5]$, each having an operational probability of 1. From this construct the corresponding $s-t$ VPIN $G_{\nu^r} = (V^*, E^*, \rho_{V^*}, \{s^*, t^*\})$ where $\rho_{V^*} = \{\overline{\rho} \cup \{p_{s^*} = 1, p_{t^*} = 1\}\}$ and $V^* = \{E \cup \{s^*, t^*\}\}$.

The probability PF that every ck-path has at least one failed edge in ν^r is just $Rel_2[G_{\nu^r}]$ where G_{ν^r} is the $s-t$ VPIN for the interval representation of ν^r. Then the probability ckP(G) that at least one path is operating is $1-PF$. Interval t^* is reachable from interval s^* if and only if at least one edge in every $P_i \in \nu^r$ fails. Again we can use the technique of AboElFotoh and Colbourn to efficiently compute $Rel_2[G_{\nu^r}]$ and thus ckP(G). Provided that the set of r consecutively ordered ck-paths can be generated in $O(S)$ time, we obtain an $O(S+m+r^2)$ time algorithm for computing ckP(G).

For networks with more than two terminals, bounds based on consecutive cuts may not perform as well as bounds based on noncrossing cuts; for a k–terminal network, a set of minimal consecutive cuts has at most $n-1$ cuts, while a set of noncrossing cuts may have $n-1+k-2$ [8]. When there are only two terminals, however, consecutive cut bounds are surely at least as accurate as noncrossing cut bounds, since all sets of noncrossing cuts are consecutive in this case. Similarly, using larger families of cuts or paths that meet the requirements of Shier's algebraic approach [19, 20] can lead to more accurate bounds than those obtained by consecutive sets alone. Nevertheless, as we see later, the consecutive bounds permit other improvements.

5 Implementation

What remains to be shown is that there is an efficient method of generating sets of ck-cuts (ck-paths). Any set of edge-disjoint k-cuts (k-paths) is consecutive and any ordering of such a set is a consecutive ordering. There are numerous efficient algorithms for generating edge-disjoint k-cuts (k-paths) [6, 13, 15]. By choosing a set of edge-disjoint k-cuts (k-paths) as part of a consecutive set, one is assured that the ck-bound is no worse that the edge-disjoint bound produced. Intermediate ck-cuts (ck-paths) can be efficiently added in a myriad of ways to further improve the ck-bounds. For example, let $\Gamma = \{P_i\}_{i=1}^{a}$, be a set of edge-disjoint minimal k-paths which has been arbitrarily ordered. A possible method of generated other consecutive ck-paths to add to $N\Gamma$ is the following:

4 Consecutive Bounds As $Rel_2[s-tVPIN]$ Problems

Let $\kappa^r = \{C_i\}_{i=1}^r$ be a set of ck-cuts of a network $G = (V, E, \rho, (s, T))$ which are ordered in some consecutive ordering. The ck-cuts upper bound (ckC) is the probability that no $C_i \in \kappa^r$ has all of its edges fail at any instant of time. In other words, it is the probability that at least one edge in every $C_i \in \kappa^r$ is operating at any instant of time.

With each $C_i \in \kappa^r$ we associate the open interval $[i-1, i]$. For each edge $e \in \cup_{i=1}^r E(C_i)$, we associate a segment which spans all intervals corresponding to ck-cuts in κ^r which contain e. Since the ck-cuts are in a consecutive ordering, these intervals are contiguous. We close the endpoints of each segment so that they completely span their corresponding set of ck-cuts intervals. Thus if some edge e_1 is contained in ck-cuts C_i through C_{i+j} and another edge e_2 is contained in ck-cuts C_{i+j+1} through C_{i+j+k}, then the segments representing e_1 and e_2 overlap at $i+j$. Lastly, we add an s^* segment spanning the interval $[-.5, .5]$ and a t^* segment spanning the interval $[r-.5, r+.5]$, each having an operational probability of 1. From this, construct the corresponding $s-t$ VPIN $G_{\kappa^r} = (V^*, E^*, \rho_{V^*}, \{s^*, t^*\})$, where $V^* = \{E \cup \{s^*, t^*\}\}$ and $\rho_{V^*} = \{\rho \cup \{p_{s^*} = 1, p_{t^*} = 1\}\}$.

The probability that no ck-cut completely fails in κ^r (i.e. ckC for G) is just $Rel_2[G_{\kappa^r}]$ where G_{κ^r} is the $s-t$ VPIN for the above described interval representation of κ^r. Interval t^* is reachable from interval s^* if and only if at least one edge in every $C_i \in \kappa^r$ operates. What remains to be shown is that an $s-t$ PVES can be constructed for G_{κ^r} with s^* as the first vertex and t^* as the last.

A well known property of interval graphs is that their maximal cliques can be ordered such that all cliques containing a vertex v_i occur consecutively [10, 9]. Such an ordering is a *consecutive clique ordering* (CCO). Let G be an $s-t$ VPIN and $\{K_i\}_{i=1}^r$ a CCO of its maximal cliques. If s is only in K_1 and t only in K_r then an $s-t$ PVES can be easily constructed for G by labeling the vertices consecutively from v_n to v_1 ensuring that all vertices in K_i are labeled before any vertices in $\cup_{j=1}^{i-1} K_j \setminus \cup_{j=i+1}^r K_j$. Unlabeled vertices within a K_i can be labeled in an arbitrary order since they are all adjacent to each other. Thus we may choose to label t v_n and s v_1 since t is contained in K_r and s is in K_1 only. The set of vertices corresponding to the edges in $C_i \cup C_{i-1}$ form a maximal clique K_i in G_{κ^r}, $2 \le i \le r$. $C_1 \cup \{s^*\}$ forms a maximal clique K_1 and $C_r \cup \{t^*\}$ forms a maximal clique K_{r+1}. Furthermore, $\{K_i\}_{i=1}^{r+1}$ is a CCO of G_{κ^r} with s^* occurring only in K_1 and t^* occurring in K_{r+1}. Hence an $s-t$ PVES exists for G_{κ^r} on which the AboElFotoh and Colbourn technique can be used to efficiently compute $Rel_2[G_{\kappa^r}] =$ ckC(G). Suppose it takes $O(S)$ time to generate a consecutively ordered set of r ck-cuts. The formulation of an $s-t$ PVES for G_{κ^r} can be accomplished in $O(m+r)$ time from G. Therefore the ck-cuts upper bound can be computed is in $O(S+m+r^2)$ time via interval graphs. This is essentially just a derivation of Shanthikumar's method that employs the interval graph representation. Perhaps what is most surprising is that this technique carries over quite directly to consecutive paths, as we see next.

The ck-paths lower bound can be defined in terms of interval graphs in a similar manner. Let $\nu^r = \{P_i\}_{i=1}^r$ be a set of ck-paths of a network $G = (V, E, \rho, (s, T))$ which are ordered in some consecutive ordering. The ck-paths lower bound ckP is the probability that some

$$Rel_k[G] = \sum_{P \in \nu} \prod_{e \in P} p_e \prod_{e \in E \setminus P} q_e \tag{1}$$

or k-cuts

$$Rel_k[G] = 1 - \sum_{C \in \kappa} \prod_{e \in C} q_e \prod_{e \in E \setminus C} p_e \tag{2}$$

An *interval graph* is a graph representing the non-empty intersections of a set of line segments which span intervals of the real line. The segments become the nodes of the interval graph. A pair of nodes are connected by an edge if their corresponding intervals overlap in at least one point [14]. A vertex v of an interval graph G is *simplicial* if it and all of its neighbours (all vertices adjacent to v) form a clique in G. A *perfect vertex elimination scheme* (PVES) for an interval graph G is an ordering of its vertices $(v_1, v_2, ..., v_n)$ such that v_i is a simplicial vertex of $G - \{v_1, v_2, ...v_{i-1}\}$ for all i. All interval graphs have many PVES since any vertex can be selected as v_n and any simplicial vertex can be selected at any stage.

3 Two-terminal Reliability of Interval Graphs

Let $G = (V, E, \rho_v, (s, t))$ be an interval graph with perfect edges and operation probabilities, $\rho_V = \{p_{v_i}, i = 1, \ldots, n\}$, assigned to its nodes. We assume that the vertex operation probabilities are independent. Let s be the source node in G and t the destination node. We assume that the operation probabilities of s and t are both 1. The set of vertex failure probabilities is $\overline{\rho_V} = \{1 - p_{v_i}, i = 1, n\}$. We call G an $s - t$ *vertex-probabilistic interval network* ($s - t$ VPIN).

Suppose there exists a PVES $(v_1, v_2, ..., v_n)$ for $s - t$ VPIN G such that $s = v_1$ and $t = v_n$ (we call this an $s - t$ PVES). Let N_i denote the neighbours of v_i in $G - \{v_1, v_2, ..., v_{i-1}\}$, and K_i denote the clique $v_i \cup N_i$. Define $\vartheta(v_i)$ to be the probability that node v_i is connected to t by a path containing only nodes in the set $\{v_{i+1}, v_{i+2}, ..., v_t\}$. Then the two–terminal reliability for G is $\vartheta(v_1 = s)$. AboElFotoh and Colbourn [2] give an iterative method for efficiently determining $\vartheta(s)$, maintaining a set of vertices $L = \{v_i\}$ for which $\vartheta(v_i)$ has been computed.

Initially $\vartheta(v_i) = 1$ for all $v_i \in K_m$ and $L = K_m$. Then for each j, $m - 1 \geq j \geq 2$ let $U_j = K_j - L$. Denote the size of U_j as u_j and the size of K_j as $\ell_j + u_j$. Thus $U_j = \{v_{j+1}, v_{j+2}, ..., v_{j+u_j}\}$. AboElFotoh and Colbourn show that for all v_i in U_j,

$$\vartheta(v_i) = \sum_{y=j+u_j+1}^{j+u_j+\ell_j} \Big(\prod_{x=j+u_j+1}^{y-1} q_{v_x} \Big) p_{v_y} \vartheta(v_y)$$

Once $\vartheta(v_i)$ is calculated v_i is added to L. For more details see [2]. Each $\vartheta(v_i)$ can be calculated in $O(n)$ time (by storing the $\prod_{x=a}^{b} q_{v_x}$ to be used in the calculation of $\prod_{x=a}^{b+1} q_{v_x}$). At most $n - 1$ such computations are required so this method computes the exact two–terminal reliability of G in $O(n^2)$ time.

both the ck-cuts and ck-paths bounds to which the technique of AboElFotoh and Colbourn can be applied.

2 Definitions

A *k-terminal network* $G = (V, E, \rho, (s, T = \{t_1, t_2, ...t_{k-1}\}))$ consists of a set of nodes V, an ordered set of edges E (if G is directed then E is a set of arcs), an ordered set of edge operation probabilities ρ, and an ordered set of terminals. The first terminal is designated as the source s and the others as destination terminals T. The value $p_{i,j} \in \rho$ represents the probability that edge (i, j) is operational at any instant of time. It is assumed that the edge operation probabilities are independent. Then $q_{i,j} = 1 - p_{i,j}$ represents the probability that edge (i, j) is non-operational at any instant of time, and $\overline{\rho} = \{q_{i,j} : (i, j) \in E\}$. In this paper the number of nodes in a network is denoted n and the number of edges as m. We assume a network is two-connected and simple since the k-terminal reliability of a one-connected multigraph can be efficiently reduced to the calculation of the k-terminal reliability of at most n simple two-connected subgraphs.

A *k-terminal pathset* (k-path) is a subgraph in which s is connected by paths to each of the destination terminals $t_i \in T$. A *minimal k-path* is a k-path which no longer remains a k-path if any edge in the k-path is removed. A *k-terminal cutset* (k-cut) is a subset of edges whose removal separates the graph into at least two connected components with s and some $t_i \in T$ in different components. A *minimal k-cut* is a k-cut which no longer remains a k-cut if any edge in the k-cut is removed. We denote the set of all k-paths as ν and the set of all k-cuts as κ. Also we use $E(C_i)$ ($E(P_i)$) to indicate the set of edges in cut C_i (path P_i).

Let (A_i, B_i) be a partition of the vertices of a graph $G - C_i$, C_i a k-cut of G, where no edges of $G - C_i$ have one end in A_i and one end in B_i (when C_i is minimal, these are the connected *shores* of the cut). Two cuts C_j and C_k are *noncrossing* if at least one of $A_j \cap A_k$, $B_j \cap B_k$, $A_j \cap B_k$ or $A_k \cap B_j$ is empty. A set $\{C_i\}_{i=1}^r$ of *consecutive k-cuts* (ck-cuts) is a set of k-cuts that can be ordered such that for all $i < j < k$, $E(C_i) \cap E(C_k) \subseteq E(C_j)$. Similarly a set $\{P_i\}_{i=1}^r$ of *consecutive k-paths* (ck-paths) is a set of k-paths that can be ordered such that for all $i < j < k$, $E(P_i) \cap E(P_k) \subseteq E(P_j)$. We call such an ordering of ck-cuts (ck-paths) a *consecutive ordering.*

When there are two terminals, any set of noncrossing cuts is consecutive; however, the converse is not true. When more than two terminals are present, there can be noncrossing cuts that are not consecutive, or consecutive cuts that are crossing. Colbourn, Nel, Boffey and Yates [8] obtain k–terminal upper bounds using a directed path graph representation for noncrossing cuts; here we pursue instead the generalization to consecutive k–terminal cuts. While we see here that the notion of consecutive cuts dualizes naturally to consecutive paths, we know of no path analogue for noncrossing cuts.

Given a k-terminal network $G = (V, E, \rho, (s, T))$, the *k-terminal reliability of G*, $Rel_k[G]$, is the probability that s is connected to all $t_i \in T$ at any instant of time. The k-terminal reliability can be expressed in terms of k-paths

1 Introduction

Our network model is a graph with perfect vertices and imperfect edges. The operation probabilities of the edges are assumed to be independent. A subset of k nodes is selected to be the *terminals* of the network; one is designated the *source* and the remainder are *destination* terminals. The k-terminal reliability of such a network is the probability that all destination nodes $T = \{t_1, t_2, ...t_{k-1}\}$ are reachable from the source node s at any instant of time. In undirected networks this is just the probability that all k terminals $T \cup \{s\}$ are connected by paths of operating edges. Computing k-terminal reliability is #P-complete even for planar graphs [16], even if there are just two terminals, and also if all vertices are terminals. Thus many polynomial time bounding techniques have been developed. These include both static and iterative techniques. Static techniques, such as Shanthikumar's [17, 18], produce a single bound with no obvious method of improvement. Iterative techniques, such as generating most probable states, produce bounds which can be improved at the expense of more computation time.

Colbourn and Harms [7] show that the Shanthikumar upper bound can be used to increase the efficiency of the Yang-Kubat most probable states method [23], hence providing a method to obtain iteratively improvable two–terminal upper bounds. The Shanthikumar upper bound generalizes to k-terminal reliability using consecutive k-terminal cutsets (ck-cuts). This bound can be used to produce iteratively improvable k-terminal upper bounds. Computational results suggest that the (ck-cuts) upper bound is quite competitive with other static upper bounds that can be efficiently computed. In addition, this bound is easy to implement and can be used with both directed and undirected networks having unequal edge operation probabilities. These attributes make the ck-cuts upper bound a very attractive bounding strategy.

Shier [19, 20] generalized Shanthikumar's upper bound. He establishes that for any family of cutsets or pathsets that form a "semilattice" structure, the probability that no cutset in the family is failed, or that no pathset in the family is operating, can be calculated efficiently (in time polynomial in the number of cuts or paths) using a dynamic programming technique. Unfortunately, in the context of most probable state techniques, the generalization does not appear to admit the constant–time update required when an edge is determined to be operating or failed (see [7] for details). Thus it remains of interest to examine restrictions of Shier's general techniques that lead to both accurate static methods, and to improvements in the Yang–Kubat method. For these reasons, we examine bounds based on consecutive pathsets. We devise a heuristic for choosing consecutive paths, and compare the accuracy of the resulting bound (the ck–paths bound) with other static lower bound techniques on a sample network, the 1979 Arpanet for terminal sets of size 2.

We employ a somewhat different formulation of the bound than that of Shier [19]. In [7], Colbourn and Harms allude to an interval graph representation of the Shanthikumar upper bound. AboElFotoh and Colbourn [2] demonstrate how two–terminal reliability of such interval graphs can be computed efficiently; Ball, Provan and Shier generalized this algorithm to directed path graphs [3]. We explicitly give the interval graph representation of

ABSTRACT

Shanthikumar developed an upper bound for two–terminal reliability based on consecutive $s - t$ cutsets. Subsequently, Shier generalized this strategy to obtain upper bounds from cutsets, and lower bounds from pathsets, when the cutsets or pathsets form a semilattice structure. We examine a restricted case of Shier's method that yields a k-terminal lower bound based on consecutive pathsets. Our approach employs a common reduction of consecutive cut and path bounds to the computation of the two–terminal reliability of an interval graph with imperfect vertices. Computational results are given to support the observation that the consecutive paths lower bound is competitive with the best efficiently computable bounds that are currently available. We then apply the consecutive path bound to reduce, in some cases dramatically, the number of states generated in a most probable state bounding method.

DIMACS Technical Report 93-23
April 1993

Consecutive Cuts and Paths, and Bounds on k-Terminal Reliability

by

Heidi J. Strayer
Computer Science
University of Waterloo
Waterloo, Ontario, CANADA
hjs@caddac.uwaterloo.ca

Charles J. Colbourn[1,2]
Combinatorics and Optimization
University of Waterloo
Waterloo, Ontario, CANADA
cjcolbou@math.uwaterloo.ca

[1]Special Year Visitor to DIMACS, whose support is gratefully acknowledged
[2]Research supported by NSERC Canada under grant number A0579.

DIMACS is a cooperative project of Rutgers University, Princeton University, AT&T Bell Laboratories and Bellcore.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.