Discrete Applied Mathematics 21 (1988) 185–198
North-Holland

LOWER BOUNDS ON TWO-TERMINAL NETWORK RELIABILITY

Timothy B. BRECHT and Charles J. COLBOURN

Computer Communications Networks Group, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

Received 4 March 1986
Revised 10 June 1987

Computing the probability that two nodes in a probabilistic network are connected is a well-known computationally difficult problem. Two strategies are devised for obtaining lower bounds on the connection probability for two terminals. The first improves on the Kruskal–Katona bound by using efficient computations of small pathsets. The second strategy employs efficient algorithms for finding edge-disjoint paths. The resulting bounds are compared; while the edge-disjoint path bounds typically outperform the Kruskal–Katona bounds, they do not always do so. Finally, a method is outlined for developing a uniform bound which combines both strategies.

Keywords. Network reliability, efficient algorithm, reliability bound, two-terminal reliability, topological design.

1. Introduction

The topology of a computer network is often represented simply as a *graph*. Investigations of computer network reliability have therefore adopted many graph-theoretic techniques. Studies of reliability partition into two classes, depending on the cause of component failures. When failures are caused by an "enemy" with knowledge of the topology, measures of reliability are called *deterministic*. Such measures are typically the minimum number of component failures required to render the network non-operational. However, when random failures occur, we are not interested in the worst case, but rather in the average case. *Probabilistic* measures consider the probability that the network is non-operational.

In this paper, we concern ourselves with probabilistic measures. We adopt a standard network model, the *probabilistic graph*. This is a graph with node set V and edge set E; each edge $e \in E$ has an associated *success probability* p_e. Nodes are assumed to be operational at all times, and edge failures are assumed to be statistically independent. Using this model, the *two-terminal connection probability* for two nodes s and t is the probability that the operational edges include an s,t-path (equivalently, the failed edges do not include an s,t-cut). This model has been widely

used, and was developed significantly in the pioneering work of Moore and Shannon [16]. A large body of research is devoted to computing two-terminal connection probabilities [5, 19]; however, known exact algorithms all require exponential time in the worst case. This is not surprising when one notes that Valiant [21] showed that the problem is #P-complete, and hence almost certainly intractable.

Naturally, the intractability of the problem does not reduce its practical importance at all. Nevertheless, we are forced to consider approximation strategies. Two different avenues have been pursued in the literature. The first employs Monte Carlo techniques, which typically yield good estimates but no absolute guarantee of success. The second is to develop absolute upper and lower bounds. We follow the second avenue here. It is of prime importance to remember that our motivation for calculating bounds is to avoid the exponential exact computation; hence we will only consider bounds which can be computed efficiently, i.e., in polynomial time.

Bounds in the literature have typically relied on one or more of *state enumeration*, *path enumeration*, or *cut enumeration*. With state enumeration, one examines all 2^e distinct states of the network; hence, exponential time is invested. Enumeration of s,t-paths and enumeration of minimal s,t-cuts are both known to be #P-complete [18, 21]. In fact, the Esary–Proschan bounds [8] and related bounds [3, 20] rely on the computation of all minimal s,t-paths. One must therefore enumerate all of the Hamiltonian paths (among others). Even enumerating Hamiltonian paths is #P-complete [21]. Hence our restriction to efficiently computable bounds leaves us with few published results. Zemel [24] and Assous [1] developed bounds which do not assume statistical independence, and thus are quite poor bounds on the case when statistical independence holds. The one set of efficiently computable bounds of interest in the literature is the Kruskal–Katona bounds [22]. It is important to note that the Ball–Provan bounds [2], which improve on the Kruskal–Katona bounds in the all-terminal case, do not apply to the two-terminal case.

In this paper, we first introduce the Kruskal–Katona bounds and describe a powerful, but efficient, technique for improving them. We then devise a new class of bounds which exploit maximal sets of edge-disjoint s,t-paths. Empirical evidence is given to show that these new bounds typical outperform even the improved Kruskal–Katona bounds. Finally, in Section 5 we employ techniques developed in [7] to obtain a uniform bound which improves on both the Kruska–Katona and the edge-disjoint path bounds.

2. Subgraph counting: The Kruskal–Katona bounds

To obtain a bound on the connection probability, we first assume that each edge e has equal success probability p. We use the notation $q = 1 - p$. Consider a network with e edges at some instant of time. Each edge is either operational or failed; suppose i edges are failed. Then the probability of this specific state of the network is

$q^i p^{e-i}$. Denoting by F_i the number of sets of i edges which do *not* contain an s,t-cut, $F_i q^i p^{e-i}$ is the probability that exactly i edges fail and the network remains operational. Then the connection probability CP(p) satisfies

$$\mathrm{CP}(p) = \sum_{i=0}^{e} F_i q^i p^{e-i},$$

CP(p) is often called the *reliability polynomial* and was introduced by Moore and Shannon [16]. The importance of the reliability polynomial is that it transforms the computation into one of combinatorial enumeration: one need only compute each F_i, which is a count of operational subgraphs on $e-i$ edges. Some of these coefficients are easily computed. We let l denote the length of a shortest s,t-path. Then if more than $e-l$ edges fail, there can be no s,t-path. Thus $F_i=0$ for $i>e-l$. Similarly, let c be the cardinality of a minimum s,t-cut; when fewer than c edges fail, there must be an s,t-path. Hence $F=\binom{e}{i}$ for $i<c$. In addition, Ball and Provan [2] give an algorithm for finding N_l, the number of paths of length l, and hence $F_{e-l}=N_l$. Unfortunately, not all coefficients can be computed so easily. In fact, computing F_c is #P-complete [18]. Nevertheless, the existence of an s,t-cut of cardinality c ensures that $F_c \le \binom{e}{c} - 1$.

In view of the complexity, bounds are computed on the unknown F_i. To do this, we employ a combinatorial result due to Kruskal [13] and Katona [12] (a recent simple proof is given by Frankl [10]. The *k-canonical representation* or *k-cascade* of a number m is the unique sequence $a_k > a_{k-1} > \cdots > a_s \ge 1$ for which

$$m = \binom{a_k}{k} + \binom{a_{k-1}}{k-1} + \cdots + \binom{a_s}{s}.$$

Define the *(i, k)th-lower pseudopower* of m to be

$$m^{i/k} = \binom{a_k}{i} + \binom{a_{k-1}}{i-1} + \cdots + \binom{a_s}{i-k+s}.$$

The Kruskal–Katona theorem states that when $F_k = m$ we have

$$F_i \ge m^{i/k} \quad \text{when } i \le k,$$

$$F_i \le m^{i/k} \quad \text{when } i \ge k.$$

Using these inequalities, we obtain the *Kruskal–Katona bounds*:

$$\mathrm{CP}(p) \ge \sum_{i=0}^{c-1} \binom{e}{i} p^{e-i} q^i + \sum_{i=c}^{d} F_d^{i/d} p^{e-i} q^i,$$

$$\mathrm{CP}(p) \le \sum_{i=0}^{c-1} \binom{e}{i} p^{e-1} q^i + \sum_{i=c}^{d-1} \left(\binom{e}{c} - 1 \right)^{i/c} p^{e-i} q^i + F_d p^{e-d} q^d,$$

where $d = e - l$.

The accuracy of the Kruskal–Katona bounds depends on the accuracy of the estimates of the "unknown" coefficients F_i. Therefore, one method for improving

these bounds is to devise efficient strategies for computing additional coefficients. We adopt this approach, by giving efficient algorithms for computing $F_{e-(l+1)}$ and $F_{e-(l+2)}$. These are, respectively, the number of subgraphs with $l+1$ and $l+2$ edges containing an s,t-path; an i-edge subgraph containing an s,t-path is called an *i-pathset*. Ball and Provan [2] give a polynomial-time algorithm for counting l-pathsets, which are simply paths of length l. We first examine a similar algorithm (which is less efficient but more easily generalized).

For each node v and each i, define $\#\text{walks}(v,i)$ to be the number of walks (paths with repeated edges allowed) from s to v of length i. For $i=0$, we define

$$\#\text{walks}(s,0) = 1,$$

$$\#\text{walks}(v,0) = 0 \quad \text{for all } v \neq s.$$

Then for $i \geq 1$, we compute

$$\#\text{walks}(v,i) = \sum_{x \in N(v)} \#\text{walks}(x, i-1),$$

where $N(v)$ contains those nodes adjacent to v. Since the length of the shortest s,t-path is l, any walk of length l from s to t is necessarily a path, and hence $F_{e-l} = \#\text{walks}(t,l)$.

Consider $F_{e-(l+1)}$. This enumerates $(l+1)$-pathsets, which can be formed in two ways: s,t-paths of length $l+1$, and s,t-paths of length l together with any other single edge. The proper paths are enumerated by $\#\text{walks}(t,l+1)$, since again every s,t-walk of length $l+1$ is a path. Hence

$$F_{e-(l+1)} = \#\text{walks}(t,l+1) + (e-l) \times \text{walks}(t,l).$$

The computation of $F_{e-(l+2)}$ is not so straightforward. First of all, $\#\text{walks}(t,l+2)$ enumerates more than s,t-paths of length $l+2$; overcounting results precisely from repetition of exactly one edge incident to a path of length l. However, this overcounting can be easily corrected by counting the number of edges incident with each s,t-path of length l. Let $\#\text{paths}(l+2)$ be the number of s,t-paths so computed. Then consider the number

$$\text{OC} = \#\text{paths}(l+2) + (e-(l+1)) \times \#\text{walks}(t,l+1) + \binom{e-l}{2} \times \#\text{walks}(t,l),$$

OC overcounts $(l+2)$-pathsets. Exactly two configurations result in overcounting. The first occurs when a path of length l intersects a path of length $l+1$ as in Fig. 1. The second overcounting occurs when two paths of length l intersect as shown in Fig. 2. The number of each type of these configurations can be counted easily, and subtracted from OC to obtain $F_{e-(l+2)}$ in polynomial time. It should be noted here that the computation of $F_{e-(l+k)}$ could be carried out in polynomial time for any fixed k by enumerating all possible types of overcounting; for practical purposes, we have restricted ourselves to $k=2$ here.

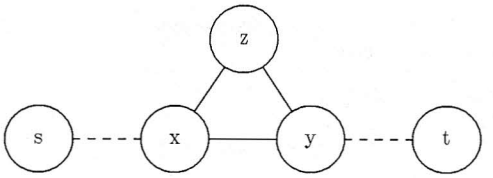

Fig. 1.

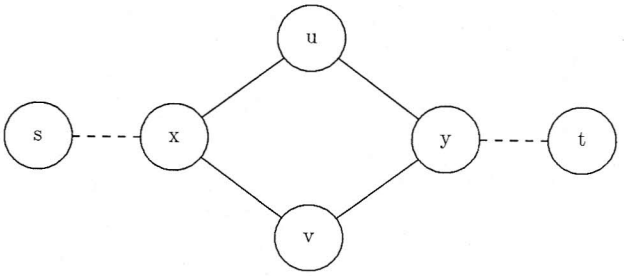

Fig. 2.

Knowledge of two additional coefficients allows us to refine the Kruskal–Katona bounds (KK bounds) to obtain a new set of bounds, which we call the KK2 bounds:

$$\mathrm{CP}(p) \leq \sum_{i=0}^{c-1} \binom{e}{i} p^{e-i} q^i + \sum_{i=c}^{d-3} \left(\binom{e}{c} - 1 \right)^{i/c} p^{e-i} q^i + \sum_{i=d-2}^{d} F_i p^{e-i} q^i,$$

$$\mathrm{CP}(p) \geq \sum_{i=0}^{c-1} \binom{e}{i} p^{e-i} q^i + \sum_{i=c}^{d-3} F_{d-2}^{i/d-2} p^{e-i} q^i + \sum_{i=d-2}^{d} F_i p^{e-i} q^i.$$

The KK2 bounds are at least as tight as the KK bounds, and often improve on them quite substantially, especially when p is small. To account for this improvement, in Table 1 we give the bounds on the F_i computed for the KK and KK2 lower and upper bounds for the ten-node "ladder" depicted in Fig. 3. Exact values in this table were found using the algorithm from [23].

In Section 4, we report computational results which compare the KK and KK2 bounds, among others.

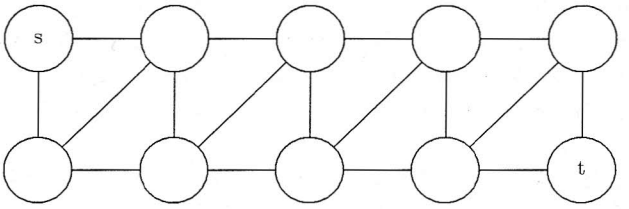

Fig. 3.

Table 1
Coefficients in the reliability polynomial

Coefficient	KK lower	KK2 lower	Exact	KK2 upper	KK upper
F_0	1	1	1	1	
F_1	17	17	17	17	17
F_2	77	90	134	135	135
F_3	285	362	643	665	665
F_4	714	989	2073	2275	2275
F_5	1286	1945	4671	5733	5733
F_6	1708	2829	7403	11011	11011
F_7	1688	3073	8078	16445	16445
F_8	1231	2484	5756	19305	19305
F_9	645	1469	2458	17875	17875
F_{10}	230	613	613	613	13013
F_{11}	50	84	84	84	7371
F_{12}	5	5	5	5	5

3. Edge-disjoint paths

In this section, we develop bounds employing a significantly different technique. The underlying observation here is that a lower bound on the reliability is given by the probability that at least one of a set of edge-disjoint subgraphs is operational. This observation has been used in the context of all-terminal reliability [17], and in the computation of expected flows in networks [6]. For two-terminal reliability, operational subgraphs contain s,t-paths; hence, we examine edge-disjoint paths. Let $P_1, \ldots, P_m$ be a collection of edge-disjoint s,t-paths. Then

$$\mathrm{CP}(p) \geq 1 - \prod_{i=1}^{m} L_i,$$

where L_i is the *failure* probability of path P_i. Now $L_i = 1 - \prod_{j=1}^{l_i} p_j$, where p_j is the probability that the jth edge of P_i is operational, and l_i is the length (in edges) of P_i. With the assumption of equal edge failure probabilities, $L_i = 1 - p^{l_i}$.

Any collection of edge-disjoint s,t-paths yield a lower bound on $\mathrm{CP}(p)$. In general, one obtains a better bound by examining sets containing the maximum number of edge-disjoint s,t-paths. However, the lengths of the paths chosen also affects the bound; shorter paths yield a more accurate bound. The particular number k of paths and selection of path lengths $\{l_1, \ldots, l_k\}$ which produce the best bound depends on p. Moreover, the determination of the best number of paths and corresponding path lengths appears difficult in view of the following result:

Theorem 3.1 [11]. *Given an n-node graph, a length $l \geq 5$, and a number m, determining whether the graph has at least m edge-disjoint s,t-paths of length at most l is NP-complete.*

We therefore resort to heuristic techniques for producing edge-disjoint paths. Observe that a trivial bound is obtained by taking a single shortest path (of length l), yielding $\mathrm{CP}(p) \geq p^l$. This is easily improved upon. If c is the cardinality of a minimum s,t-cut, there exist c edge-disjoint paths from s to t [15]. A set of c such paths can be found efficiently using the maxflow-mincut theorem [9]. Two potential problems arise with this method. The first is that, although it is *typically* better to have more paths, it is not always better. For example, consider the network depicted in Fig. 4. In this case, one has a choice of one path of length 3, or two paths each of length 6. Hence, $\mathrm{CP}(p) \geq p^3$, and also $\mathrm{CP}(p) \geq 2p^6 - p^{12}$. When p is small, the first bound is better, whereas high values of p make the second bound better. For this reason, one would like to take the best bound obtained over all possible numbers of paths. Using flow techniques, this is easily done, as the set of edge-disjoint paths is constructed one path at a time. Taking the best bound over all sets of edge-disjoint paths constructed yields the *maximum flow* or *maxflow* bound.

The second drawback of using maximum flow techniques is that no effort is made to ensure that path lengths remain short. In view of the Itai–Perl–Shiloach theorem, one might suspect that length information about the paths cannot be efficiently incorporated. However, there is an efficient algorithm [14] for *minimum cost network flows* in which the collection of edge-disjoint paths found have minimum total length (that is, the sum of path lengths is minimized). This is simply a variant of the maxflow algorithm in which shortest paths are selected during augmentation. Once again, selecting the best bound over all numbers of paths yields a lower bound which we call the *mincost* bound. Typically, one expects the mincost bound to improve on maxflow, since mincost employs the minimum possible number of edges. However, occasionally maxflow outperforms mincost since mincost does not guarantee to pick the minimum length for any individual path, but rather chooses the shortest total length. To illustrate this, consider the graph in Fig. 5.

Mincost finds, in turn, edge-disjoint path collections with path lengths $\{3\}$, $\{3, 6\}$, $\{3, 8, 8\}$, and finally $\{7, 7, 9, 9\}$; the last yields 32 edges in total. On the other hand, maxflow finds paths of length $\{3\}$, $\{3, 6\}$, $\{4, 8, 8\}$, and finally $\{4, 8, 8, 15\}$; the last yields 35 edges in total (and hence is *not* minimum cost). Nevertheless, Table 2 shows that maxflow often improves on mincost. The reason

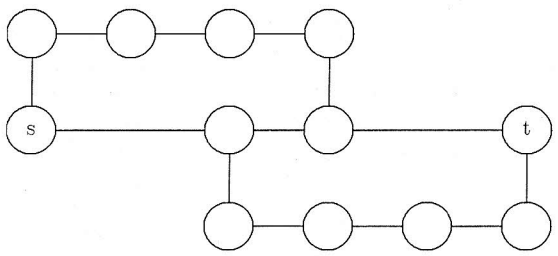

Fig. 4.

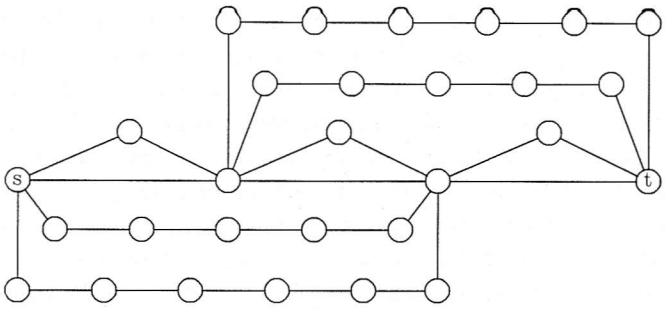

Fig. 5.

is perhaps clear: mincost achieves minimum total path length at the expense of eliminating the (desirable) path of length 4. It is perhaps worth noting that neither mincost nor maxflow finds the set of four paths with lengths {3, 8, 8, 16}, which improves on both for large p.

Other techniques for producing sets of edge-disjoint paths could also be employed to produce bounds, with the understanding that two goals are to produce many paths, and to produce short paths. The mincost strategy provides an effective balancing of these two goals, but of course any strategy which realizes both goals more effectively would improve on mincost. Results comparing mincost, maxflow, and the subgraph counting bounds discussed earlier are given in the next section.

Table 2
Results for Fig. 5 (* denotes the best of the two methods for that graph; if there is no * in either column the results are equal; # denotes the number of paths used to obtain the bound)

p	#	mincost	maxflow	#
0.10	2	0.001001	0.001001	2
0.20	2	0.008063	0.008063	2
0.30	2	0.027709	0.027709	2
0.40	2	0.067834	0.067834	2
0.50	2	0.138672	0.138672	2
0.60	2	0.252578	0.252578	2
0.70	2	0.420295	0.420295	2
0.80	3	0.662010 *	0.639926	2
0.90	3	0.912096 *	0.911417	4
0.91	3	0.930844	* 0.933242	4
0.92	3	0.947559	* 0.952038	4
0.93	4	0.963511	* 0.967585	4
0.94	4	0.977469	* 0.979790	4
0.95	4	0.987559	* 0.988722	4
0.96	4	0.994160	* 0.994645	4
0.97	4	0.997880	* 0.998033	4
0.98	4	0.999519	* 0.999548	4
0.99	4	0.999965	* 0.999967	4

4. Computational results

In Sections 2 and 3, we developed four bounds on CP(p), namely the KK bound, the KK2 bound, the maxflow bound, and the mincost bound. In this section, we describe our computational experience with these bounds. Each bound can be computed in polynomial time. The KK bounds require the value of c, the value of l, and F_{e-l}. Each can be efficiently computed: c using network flows [9], l using shortest path algorithms [14], and F_{e-l} using Ball and Provan's algorithm [2]. The KK2 bounds require in addition the number of $(l+1)$-pathsets and $(l+2)$-pathsets; the algorithm given in Section 2 finds these is polynomial time. In practical terms, however, this is by far the most time-consuming part of the computation. Finally, both the maxflow and mincost bounds are obtained by efficient network flow techniques [14].

The most important comparison of the bounds involves their *accuracy*. We have extensively tested each bound, and have found that the KK2 bounds outperform the KK bounds significantly, especially when p is near zero. In addition, the mincost bound almost always improves on the maxflow bound. Perhaps of more interest is the comparison of the subgraph counting and the edge-disjoint path bounds. It is of interest to note here that in the all-terminal reliability problem, subgraph counting bounds are typically much better that edge-disjoint subgraph bounds [7]. For two-terminal bounds, however, we have found the exact opposite situation: mincost almost always improves on KK2.

We give a small collection of results which illustrate the behaviour of the bounds; a more comprehensive comparison can be found in [4]. We include results using the

Table 3
The ten-node ladder (Fig. 3)

p	KK (0)	KK2 (2)	Mincost (m)	Maxflow (M)	Ranking
0.10	0.0000409510	0.0000687793	0.0000199999	0.0000199999	2,0,mM
0.20	0.0010757116	0.0022165203	0.0006398976	0.0006398976	2,0,mM
0.30	0.0067385833	0.0141760289	0.0048540951	0.0048540951	2,0,mM
0.40	0.0236085387	0.0462748820	0.0203751424	0.0203751424	2,0,mM
0.50	0.0605545044	0.1065139771	0.0615234375	0.0615234375	2,mM,0
0.60	0.1285654555	0.2007626807	0.1494733824	0.1494733824	2,mM,0
0.70	0.2428156278	0.3353783955	0.3078924751	0.3078924751	2,mM,0
0.80	0.4301397466	0.5227844138	0.5479858176	0.5479858176	mM,2,0
0.90	0.7279011487	0.7815387689	0.8323015599	0.8323015599	mM,2,0
0.91	0.7631727458	0.8105402996	0.8586481721	0.8586481721	mM,2,0
0.92	0.7986710793	0.8394946226	0.8837745922	0.8837745922	mM,2,0
0.93	0.8339442950	0.8680534036	0.9073944314	0.9073944314	mM,2,0
0.94	0.8684065014	0.8957672849	0.9291929307	0.9291929307	mM,2,0
0.95	0.9013089334	0.9220639998	0.9488249358	0.9488249358	mM,2,0
0.96	0.9317059579	0.9462225923	0.9659127592	0.9659127592	mM,2,0
0.97	0.9584151291	0.9673430499	0.9800439245	0.9800439245	mM,2,0
0.98	0.9799703949	0.9843107484	0.9907687867	0.9907687867	mM,2,0
0.99	0.9945674374	0.9957548793	0.9975980248	0.9975980248	mM,2,0

four bounds on five networks: the ten-node ladder from Fig. 3 (Table 3), the network from Fig. 5 (Table 4), the 1979 Arpanet (Table 5) from [2] (using $s = \text{ISI22}$ and

Table 4
The graph of Fig. 5

p	KK (0)	KK2 (2)	Mincost (m)	Maxflow (M)	Ranking
0.10	0.0010000000	0.0012746422	0.0010009990	0.0010009990	2, mM, 0
0.20	0.0080000000	0.0116067613	0.0080634880	0.0080634880	2, mM, 0
0.30	0.0270000000	0.0411333050	0.0277093170	0.0277093170	2, mM, 0
0.40	0.0640000001	0.0972685666	0.0678338560	0.0678338560	2, mM, 0
0.50	0.1250000491	0.1835327493	0.1386718750	0.1386718750	2, mM, 0
0.60	0.2160088640	0.3001808084	0.2525783040	0.2525783040	2, mM, 0
0.70	0.3435463935	0.4454494096	0.4202953930	0.4202953930	2, mM, 0
0.80	0.5248603153	0.6232596781	0.6620096493	0.6399262720	m, M, 2, 0
0.90	0.8223618631	0.8669223894	0.9120963807	0.9114171421	m, M, 2, 0
0.91	0.8566941176	0.8931114770	0.9308440421	0.9332417719	M, m, 2, 0
0.92	0.8895211886	0.9179325107	0.9476688235	0.9520375072	M, m, 2, 0
0.93	0.9197074617	0.9405818560	0.9635114176	0.9675846417	M, m, 2, 0
0.94	0.9460669318	0.9602281853	0.9774694213	0.9797895379	M, m, 2, 0
0.95	0.9675241929	0.9761286109	0.9875588367	0.9887217044	M, m, 2, 0
0.96	0.9833401889	0.9877903989	0.9941597619	0.9946450265	M, m, 2, 0
0.97	0.9933802170	0.9951616289	0.9978803414	0.9980325747	M, m, 2, 0
0.98	0.9983539372	0.9987998663	0.9995193196	0.9995479883	M, m, 2, 0
0.99	0.9998702229	0.9999055926	0.9999654823	0.9999670888	M, m, 2, 0

Table 5
The 1979 Arpanet

p	KK (0)	KK2 (2)	Mincost (m)	Maxflow (M)	Ranking
0.10	0.0000010000	0.0000010002	0.0010010001	0.0000010000	m, M, 20
0.20	0.0000640000	0.0000640000	0.0000641024	0.0000640205	m, M, 20
0.30	0.0007290000	0.0007290000	0.0007349006	0.0007307702	m, M, 20
0.40	0.0040960000	0.0040960000	0.0042004281	0.0041377822	m, M, 20
0.50	0.0156250000	0.0156250000	0.0165863037	0.0161065902	m, M, 20
0.60	0.0466560000	0.0466560004	0.0524205066	0.0501494333	m, M, 20
0.70	0.1176490004	0.1176490004	0.1425732318	0.1357860888	m, M, 20
0.80	0.2621463221	0.2621463221	0.3413706847	0.3333015111	m, M, 20
0.90	0.5340970673	0.5340970673	0.7321002905	0.7175691319	m, M, 20
0.91	0.5726505830	0.5726505830	0.7771468606	0.7633113480	m, M, 20
0.92	0.6147062610	0.6147062610	0.8209381816	0.8082641025	m, M, 20
0.93	0.6610698405	0.6610698405	0.8624110028	0.8513513335	m, M, 20
0.94	0.7126103333	0.7126103333	0.9003670347	0.8913068568	m, M, 20
0.95	0.7698783722	0.7698783722	0.9335328446	0.9267209039	m, M, 20
0.96	0.8322276165	0.8322276165	0.9606726836	0.9561471457	m, M, 20
0.97	0.8962243822	0.8962243822	0.9807804630	0.9783062185	m, M, 20
0.98	0.9536035618	0.9536035618	0.9933876415	0.9924387347	m, M, 20
0.99	0.9910080560	0.9910080560	0.9990380601	0.9988847183	m, M, 20

t = CCA), a complete seven-node network (Table 6), and a 7×7 grid network, depicted in Fig. 6 (Table 7).

Table 6
The complete seven-node graph

p	KK (0)	KK2 (2)	Mincost (m)	Maxflow (M)	Ranking
0.10	0.1000000000	0.1460143815	0.1441089551	0.1441089551	2, mM, 0
0.20	0.2000000110	0.3298044649	0.3477018419	0.3477018419	mM, 2, 0
0.30	0.3000038852	0.4968170179	0.5631774984	0.5631774984	mM, 2, 0
0.40	0.4001902187	0.6357753009	0.7490728346	0.7490728346	mM, 2, 0
0.50	0.5029544830	0.7495417595	0.8813476563	0.8813476563	mM, 2, 0
0.60	0.6203807813	0.8435083297	0.9570503270	0.9570503270	mM, 2, 0
0.70	0.7712523337	0.9219654930	0.9896492425	0.9896492425	mM, 2, 0
0.80	0.9259296528	0.9782001374	0.9987906765	0.9987906765	mM, 2, 0
0.90	0.9956825505	0.9988497919	0.9999752390	0.9999752390	mM, 2, 0
0.91	0.9973866946	0.9993091455	0.9999864910	0.9999864910	mM, 2, 0
0.92	0.9985324481	0.9996148871	0.9999931602	0.9999931602	mM, 2, 0
0.93	0.9992505237	0.9998047033	0.9999968495	0.9999968495	mM, 2, 0
0.94	0.9996619515	0.9999125024	0.9999987179	0.9999987179	mM, 2, 0
0.95	0.9998713030	0.9999669023	0.9999995595	0.9999995595	mM, 2, 0
0.96	0.9999616690	0.9999902025	0.9999998815	0.9999998815	mM, 2, 0
0.97	0.9999922533	0.9999980315	0.9999999784	0.9999999784	mM, 2, 0
0.98	0.9999992282	0.9999998050	0.9999999981	0.9999999981	mM, 2, 0
0.99	0.9999999863	0.9999999966	1.0000000000	1.0000000000	mM, 2, 0

Table 7
The 7×7 grid graph (Fig. 6)

p	KK (0)	KK2 (2)	Mincost (m)	Maxflow (M)	Ranking
0.10	0.0000000001	0.0000000002	0.0000000000	0.0000000000	2, 0, mM
0.20	0.0000000979	0.0000002145	0.0000000082	0.0000000082	2, 0, mM
0.30	0.0000058649	0.0000119043	0.0000010629	0.0000010629	2, 0, mM
0.40	0.0001047754	0.0001972150	0.0000335542	0.0000335542	2, 0, mM
0.50	0.0009765029	0.0016782284	0.0004882216	0.0004882216	2, 0, mM
0.60	0.0060466014	0.0093706539	0.0043488263	0.0043488263	2, 0, mM
0.70	0.0282475235	0.0391953831	0.0274909932	0.0274909932	2, 0, mM
0.80	0.1073741999	0.1331096404	0.1327165870	0.1327165870	2, mM, 0
0.90	0.3488357172	0.3871709320	0.4850926299	0.4850926299	mM, 2, 0
0.91	0.3897708376	0.4279343938	0.5409605348	0.5409605348	mM, 2, 0
0.92	0.4351706648	0.4726214307	0.6001542027	0.6001542027	mM, 2, 0
0.93	0.4856635613	0.5217435175	0.6619697347	0.6619697347	mM, 2, 0
0.94	0.5421209522	0.5760234641	0.7253404836	0.7253404836	mM, 2, 0
0.95	0.6057775408	0.6365030496	0.7887311510	0.7887311510	mM, 2, 0
0.96	0.6782813259	0.7045875537	0.8500062679	0.8500062679	mM, 2, 0
0.97	0.7612905280	0.7816826342	0.9062675001	0.9062675001	mM, 2, 0
0.98	0.8543882458	0.8673171943	0.9536531110	0.9536531110	mM, 2, 0
0.99	0.9477608984	0.9525528440	0.9870916026	0.9870916026	mM, 2, 0

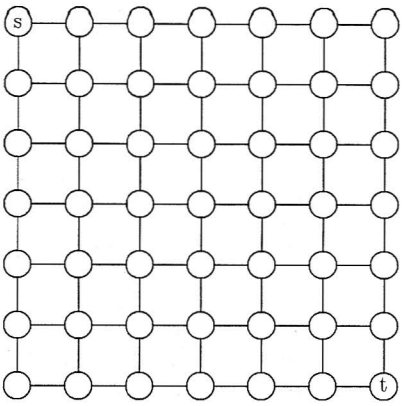

Fig. 6.

5. Combining the bounds

One immediate conclusion from Section 4 is that, while mincost may typically outperform the other methods, it does not always do so. One would prefer a bounding technique which always produces the best available bound. One approach to this problem is to try to combine all of the bounds described thus far. A similar problem exists in the case of all-terminal network reliability, and a strategy for producing a combined bound has been developed by Colbourn and Harms [7]. We apply a similar technique here; we review the method, but refer to [7] for implementation details.

It is convenient to view the production of bounds as the development of constraints on the F_i coefficients of the reliability polynomial. Subgraph counting bounds in general, and the KK and KK2 bounds in particular, produce constraints on each individual F_i. Other bounds, such as mincost and maxflow, do not bound individual coefficients. Nevertheless, they do bound CP(p), the reliability polynomial. If we evaluate CP(p) at a particular value of p, then the reliability polynomial becomes a linear function of the F_i. This produces a linear constraint on the F_i values. In fact, selecting any bound and any value for p, we obtain a linear constraint.

To produce a lower bound on the reliability polynomial at a particular value of p, we evaluate CP(p) again to obtain a linear function. This is a linear objective function which must satisfy all of the linear constraints produced. Minimizing this objective function produces a lower bound; one can see that it combines constraints from all bounds available, and hence can be no worse than any of them. In fact, it improves on all of them on occasion. We should also note that this combined bound can be efficiently computed using efficient techniques for linear programming. This thumbnail sketch is not intended to develop the combined bounds thoroughly, since the details parallel [7] quite closely. It is, however, of interest to

Table 8
The effect of combining the bounds

p	KK2 lower	Maxflow	Mincost	Combined	KK2 upper
0.1	0.00127464	0.001001	0.001001	0.00127464	0.0773211
0.2	0.0116068	0.00806349	0.00806349	0.0116068	0.479599
0.3	0.0411333	0.0277093	0.0277093	0.0411333	0.748084
0.4	0.0972686	0.0678339	0.0678339	0.0972686	0.869969
0.5	0.183533	0.138672	0.138672	0.183535	0.937495
0.6	0.300181	0.252578	0.252578	0.300336	0.9744
0.7	0.445448	0.420295	0.420295	0.449742	0.9919
0.75	0.529037	0.524769	0.531836	0.544089	0.996094
0.8	0.62326	0.639926	0.66201	0.662167	0.9984
0.85	0.735398	0.769112	0.795768	0.795768	0.999494
0.9	0.866922	0.911417	0.912096	0.912883	0.9999
0.91	0.893111	0.933242	0.930844	0.933411	0.999934
0.92	0.917933	0.952038	0.947559	0.952038	0.999959
0.93	0.940582	0.967585	0.963511	0.967585	0.999976
0.94	0.960228	0.97979	0.977469	0.97979	0.999987
0.95	0.976129	0.988722	0.987559	0.988722	0.999994
0.96	0.98779	0.994645	0.99416	0.994645	0.999997
0.97	0.995162	0.998033	0.99788	0.998033	0.999999
0.98	0.9988	0.999548	0.999519	0.999548	1
0.99	0.999906	0.999967	0.999965	0.999967	1

consider the effect of combining the bounds. Table 8 gives results using the combined bound for the network of Fig. 5.

6. Summary

Two strategies for computing bounds on two-terminal connection probabilities have been introduced, and efficient techniques for their implementation have been outlined. The most promising resulting bound appears to be the mincost bound; despite this, both maxflow and KK2 are of interest since each occasionally outperforms mincost. The improvement of the edge-disjoint path bounds seems difficult, in view of the apparent intractability of maintaining paths of short length. Nevertheless, heuristic approaches for finding better sets of edge-disjoint paths definitely merit further study. The subgraph counting bounds could be improved by computing $(l+k)$-pathsets for fixed k; however, in practical terms, this computation is quite complex, although still polynomial-time. Clever enumeration techniques which avoid the overcounting would be valuable here.

Finally, the combined bounds produce improvements over all of the individual methods. The improvement of the combined bounds requires either the improvement of one of the basic bounds, or the development of a new strategy for obtaining bounds.

Acknowledgment

We would like to thank Mike Ball, Ehab El Mallah, Daryl Harms, André Paradis, and Aparna Ramesh for comments on this work. Research of the second author is supported by NSERC Canada under grant A0579.

References

[1] J.Y. Assous, First and second order bounds for terminal reliability, Networks 16 (1986) 319–329.
[2] M.O. Ball and J.S. Provan, Calculating bounds on reachability and connectedness in stochastic networks, Networks 13 (1983) 253–278.
[3] Z.W. Birnbaum, J.D. Esary and S.C. Saunders, Multicomponent systems and structures and their reliability, Technometrics 3 (1961) 55–77.
[4] T.B. Brecht, Lower bounds for two-terminal network reliability, M.Math. Thesis, Department of Computer Science, University of Waterloo, Waterloo, Ont. (1985).
[5] J.A. Buzacott, A recursive algorithm for finding reliability measures related to the connection of nodes in a graph, Networks 10 (1980) 311–327.
[6] M. Carey and C. Hendrickson, Bounds on expected performance of networks with links subject to failure, Networks 14 (1984) 439–456.
[7] C.J. Colbourn and D.D. Harms, Bounding all-terminal reliability in computer networks, Networks (to appear).
[8] J.D. Esary and F. Proschan, Coherent structures of non-identical components, Technometrics 5 (1963) 191–209.
[9] L. Ford and D. Fulkerson, Flows in Networks (Princeton University Press, Princeton, NJ, 1962).
[10] P. Frankl, A new short proof for the Kruskal–Katona theorem, Discrete Math. 48 (1984) 327–329.
[11] A. Itai, Y. Perl, and Y. Shiloach, The complexity of finding maximum disjoint paths with length constraint, Networks 12 (1982) 277–286.
[12] G. Katona, A theorem on finite sets, in: Theory of Graphs (Academia Kiado, Budapest, 1968) 187–207.
[13] J.B. Kruskal, The number of simplices in a complex, in: Mathematical Optimization Techniques (University of California Press, Berkeley, CA, 1963) 251–278.
[14] E. Lawler, Combinatorial Optimization: Networks and Matroids (Holt, Rinehart and Winston, New York, 1976).
[15] K. Menger, Zur allgemeine Kurventheorie, Fund. Math. 10 (1927) 96–115.
[16] E.F. Moore and C.E. Shannon, Reliable circuits using less reliable relays, J. Franklin Inst. 262 (1956) 191–208, 281–297.
[17] V.P. Polesskii, A lower boundary for the reliability of information networks, Problems Inform. Transmission 7 (1971) 165–171.
[18] J.S. Provan and M.O. Ball, The complexity of counting cuts and of computing the probability that a graph is connected, SIAM J. Comput. 12 (1983) 777–788.
[19] A. Rosenthal, Computing the reliability of complex networks, SIAM J. Appl. Math. 32 (1977) 384–393.
[20] A.W. Shogan, Sequential bounding of the reliability of a stochastic network, Oper. Res. 34 (1976) 1027–1044.
[21] L.G. Valiant, The complexity of enumeration and reliability problems, SIAM J. Comput. 8 (1979) 410–421.
[22] R.M. van Slyke and H. Frank, Network reliability analysis: Part I, Networks 1 (1972) 279–290.
[23] J.A. Wald and C.J. Colbourn, Steiner trees in probabilistic networks, Microelectron. Reliab. 23 (1983) 837–840.
[24] E. Zemel, Polynomial algorithms for estimating network reliability, Networks 12 (1982) 439–452.