Topological Optimization of a Communication Network Subject to a Reliability Constraint

Rong-Hong Jan, Member IEEE
National Chiao Tung University, Hsin-Chu
Fung-Jen Hwang
National Chiao Tung University, Hsin-Chu
Sheng-Tzong Cheng
University of Maryland, College Park

Key Words — **Network design, Network planning, Network reliability.**

Reader Aids —
Purpose: Widen state of the art
Special math needed for explanations: Probability, Branch & bound procedure
Special math needed to use results: None
Results useful to: Network designer, Reliability analysts & theoreticians

Summary & Conclusions — **This paper considers network topological optimization with a reliability constraint. The objective is to find the topological layout of links, at a minimal cost, under the constraint that the network reliability is not less than a given level of system reliability. A decomposition method, based on branch & bound, is used for solving the problem. In order to speed-up the procedure, an upper bound on system reliability in terms of node degrees is applied. A numerical example illustrates, and shows the effectiveness of the method.**

If a^*, an important parameter, is close to the minimal number of links in a network which satisfy the reliability constraint, then a better starting solution can be obtained, and many searching steps can be saved. In our method, the lower bound a^* is close to its actual value if the operational reliability of the link is close enough to 1. Also, if we can find the maximal increasing value of the reliability when a set of links is added to a specified topology, the efficiency of the branch & bound algorithm is improved.

1. INTRODUCTION

An important stage of network design is to find the best way to layout all the components to optimize a variable (usually refers to minimize cost) while, at the same time, meeting a performance criterion such as transmission delay, throughput, or reliability. This design stage is, ''network topological optimization''. Usually, a cost-effective large network has a multilevel, hierarchical structure consisting of a backbone network and several local access networks [4]. Therefore, designing the topology of a large network can be divided into two problems, the backbone network design and the local network design. This paper deals mainly with backbone network design.

For backbone network design, the deterministic connectivity measure usually adopted is reliability [6,11] because it is more easily computed than other probabilistic connectivity measures. However, probabilistic connectivity is a standard measure of network reliability. Many papers [1,2,5,10,13,14] consider topological optimization with a network-reliability criterion. For example, [1,2,5] consider topological optimization for maximizing network reliability subject to cost constraints; [14] considers minimizing the total link cost subject to reliability constraints. All of them find an approximate solution because as the number of links increases, the number of possible layouts of links grows faster than exponentially. However, the exact optimal solution can be important where the topology will be used for a long while. This paper presents a practical method for exactly solving the topological optimization problem.

The problem in this paper is minimization of the total link cost subject to the condition that the sys-reliability cannot be less than a given threshold. Sys-reliability is defined in *Nomenclature* in section 2 [7]. This paper presents a decomposition method for finding the exact, optimal solution. The decomposition method divides the problem into several subproblems by the link number of the network, and then these subproblems are solved by our branch & bound algorithm.

To speed up the solution procedure, a lower bound on the minimum number of links in a network, which may satisfy the reliability constraint, is applied. This reduces the number of subproblems.

Section 2 describes the problem formulation and system assumptions. Section 3 presents the solution method. Section 4 is an illustrative example with some experimental results.

2. STATEMENT OF THE PROBLEM

Nomenclature

- sys-reliability. Pr{the 2 nodes in each/every node-pair in the system can communicate with each other}
- degree sequence. Listing, by increasing i, of the degree of node i, for all nodes in the node set.
- ordered degree-sequence. A degree sequence wherein the degree of node i is non-decreasing.

Notation

N	node set, with $\|N\|$ nodes
L	link set, with $\|L\|$ links
(i,j)	a link between nodes i & j
p,q	link [reliability, unreliability] for all links; $q+p\equiv 1$
$G(N,L,p)$	graph (N,L), including p
$R(G)$	reliability of G
$x_{i,j}$	selection status of (i,j): $x_{i,j} = 1$ if (i,j) is selected, else $x_{i,j} = 0$

0018-9529/93/$3.00 ©1993 IEEE

x — $\{x_{1,2}, x_{1,3},\ldots,x_{1,n},x_{2,3}, x_{2,4},\ldots,x_{n-1,n}\}$: the set of all $x_{i,j}$
$f(x)$ — sys-reliability of the network implied by x
n — number of nodes, $|N|$
n^* — $\frac{1}{2}\cdot n\cdot(n-1)$: number of $x_{i,j}$
P_0 — reliability threshold
$c_{i,j}$, $c(e)$ — cost of $[(i,j)$, link $e]$
d_i — degree of (number of links incident on) node i
$\mathbf{d}$ — degree sequence
D — $\Sigma_{i=1}^{n} d_i$
$\mathbf{d}(\alpha,j)$ — a $\mathbf{d}$, such that: $d_i=\alpha$, for $i=1,\ldots,j$; $d_i=\alpha+1$, otherwise; for any $1 \le j \le n$
$\Sigma_{i,j}$ — $\Sigma_{i=1}^{n-1}\ \Sigma_{j=i+1}^{n}$
MP — main mathematical problem; see (1) & (2)
$P_n(l)$ — subproblem l, $l = n-1, \ldots, n^*$; see (3) & (4)
$z(l)$ — optimal solution of $P_n(l)$
z — optimal solution of MP
z^* — minimum z, so far
$\underline{z}(l)$ — minimum $z(l)$, so far
$RL(l)$ — sub-subproblem l; see (5) & (6)
$r(l)$ — solution to $RL(l)$, *viz*, maximum sys-reliability for l links
$\bar{r}(l)$ — an upper bound for $r(l)$

Other, standard notation is given in "Information for Readers & Authors" at the rear of each issue.

The problem is to find a network topology such that the total link cost is minimized and the sys-reliability $\ge P_0$.

Assumptions

1. The location of each network node is given
2. Each $c_{i,j}$ and the p are fixed and known
3. Each link is bidirectional, *ie*, a path can be traversed in either direction;
4. There is no redundant link in the network. □

The main problem can be stated mathematically:

Problem MP

$$z = \text{Minimize} \sum_{i,j} c_{i,j}\cdot x_{i,j} \tag{1}$$

subject to:

$$f(x) \ge P_0. \quad \square(2)$$

3. SOLUTION METHOD

MP can be decomposed by its link number into subproblems $P_n(l)$:

Problem $P_n(l)$

$$z(l) = \text{Minimize} \sum_{i,j} c_{i,j}\cdot x_{i,j} \tag{3}$$

subject to:

$$f(x) \ge P_0 \tag{4a}$$

$$\sum_{i,j} x_{i,j} = l. \quad \square(4b)$$

Because a connected graph has at least $n-1$ links and at most n^* links, the $P_n(l)$ are considered from $n-1$ to n^*. If $n^* - (n-1) + 1$ subproblems are solved then the optimal value of problem MP is $z=\min(z(l)\,|\,l=n-1,\ldots,n^*)$. This approach is not maximally efficient because all optimal solutions for subproblems must be found. Therefore, a more efficient solution is developed for solving MP:

Algorithm-1

1. Find a lower bound a^* of the minimal number of links such that

$f(x) \ge P_0$. Set $l = a^*$, and set current solution $z^* = \infty$.

2. While MP is not solved, perform the loop:

2.1 Solve $P_n(l)$ and obtain $z(l)$. If $z(l) < z^*$, then set $z^* = z(l)$.

2.2 $\underline{z}(l+1)$ is determined by summing the $l+1$ smallest $c_{i,j}$.

2.3 If $\underline{z}(l+1) < z^*$ then set $l = l + 1$ and go to step 2.1.

3. z^* is the optimal value for MP. STOP. □

In step 2.3, because the sequence of $\underline{z}(l)$, $l = n-1, \ldots, n^*$ is increasing, we can guarantee that if $\underline{z}(k) \ge z^*$, then —

$\underline{z}(i) \ge z^*$, for $i = k,k+1,\ldots,n^*$

z^* is the optimal value for MP.

Section 3.1 presents a method to determine an a^* for step 1 and a branch & bound algorithm is suggested to solve $P_n(l)$ for step 2.1.

3.1 *Lower Bound of Minimal Number of Links*

In order to determine a lower bound of the minimal number of links, it is necessary to know the maximum sys-reliability for a fixed $|L|$. Consider:

Problem $RL(l)$

$$r(l) = \text{Maximize } f(x) \tag{5}$$

subject to:

$$\sum_{i,j} x_{i,j} = l. \quad \square(6)$$

$RL(l)$ maximizes the sys-reliability such that $|L|=l$. If $r(l) < P_0$, then $P_n(l)$ does not have any feasible solutions. In addition to $r(n-1) < r(n) < r(n+1) < \ldots < r(n^*)$, we also determine the smallest k such that $r(k) \ge P_0$, and then set

$a^* = k$. However, it is difficult to evaluate $r(l)$. The expressions for $r(n-1)$, $r(n)$, $r(n+1)$ are in [9] and lemma 1. For simplicity, $n+1$ is assumed to be a multiple of 3 in lemma 1. For $l > n+1$, we find $\bar{r}(l)$ instead of $r(l)$. The $\bar{r}(l)$, $l = n+2, \ldots, n^*$ are in lemma 4.

Lemma 1. The $r(n-1)$, $r(n)$, $r(n+1)$ for problems (3) - (4) are:

$$r(n-1) = p^{n-1},$$

$$r(n) = p^n + n \cdot p^{n-1} \cdot q,$$

$$r(n+1) = p^{n+1} + (n+1) \cdot p^n \cdot q + ⅓ \cdot (n+1)^2 \cdot p^{n-1} \cdot q^2.$$

□

A result of lemma 1 is that the topology with maximal reliability —

- for $n-1$ links is the spanning tree
- for n links is the ring
- for $n+1$ links is a graph with 3 cycles; the cycle-lengths do not differ by more than one.

The topologies are shown in figure 1.

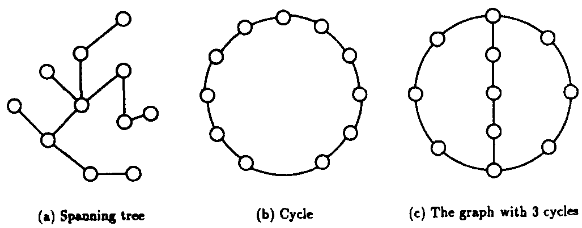


Figure 1. Optimal Topologies of $r(n-1)$, $r(n)$, $r(n+1)$

Before stating the expressions for $\bar{r}(l)$, $l = n+2, \ldots,$ n^*, we give (in lemma 2) a method to compute an upper bound of sys-reliability. Each network can be associated with a unique degree-sequence, and $D = 2 \cdot |L|$. For convenience, an ordered degree-sequence is assumed in this section. Lemmas 2 & 3 are proved in [9].

Lemma 2. Let G be a network with n nodes and degree sequence $\boldsymbol{d}$.

$$R(G) \le H(\boldsymbol{d})$$

$$H(\boldsymbol{d}) \equiv 1 -$$

$$\left\{ \sum_{i=1}^{n} q^{d_i} \cdot \prod_{k=1}^{m_i} (1-q^{d_k-1}) \cdot \prod_{k=m_i+1}^{i-1} (1-q^{d_k}) \right\}$$

$m_i \equiv \min(d_i, i-1)$, for all i. □

For example, let $\boldsymbol{d} = \{2,2,2,2\}$ and let $q = 0.1$.

$$R(G) \le 1 - [0.1^2 + 0.1^2 \cdot 0.9 + 0.1^2 \cdot 0.9 \cdot 0.9$$

$$+ 0.1^2 \cdot 0.9 \cdot 0.9 \cdot 0.99] = 0.9649.$$

$H(\boldsymbol{d})$ can be chosen as the upper bound of $R(G)$. Our goal is to find upper bounds on $\{r(l)\}_{l=n+2}^{n^*}$. This problem can be transformed to find a network such that $H(\boldsymbol{d})$ is maximal and $D = 2 \cdot |L|$.

Lemma 3. For any network $G(N,L,p)$ with ordered degree sequence $\boldsymbol{d}$,

if $d_s + 1 < d_t$ where $s < t$,

then there exists a node k with link $(k,t) \in L$ and link $(k,s) \notin L$. Remove link (k,t) from G and add link (k,s) to G. The resulting network G' has a greater upper reliability bound:

$$H(d_1, \ldots, d_s+1, \ldots, d_t-1, \ldots, d_n) \ge H(d_1, \ldots, d_s, \ldots, d_t, \ldots, d_n).$$ □

For example, $H(2,2,2,2) \ge H(1,2,2,3)$.

Lemma 4 follows from lemmas 2 & 3.

Lemma 4. Let network G have n nodes, $|L|$ links, and special degree-sequence $\boldsymbol{d}(\alpha,j)$ such that $D = 2 \cdot |L|$. Then $H(\boldsymbol{d}(\alpha,j))$ is a maximum over all possible $\boldsymbol{d}$. □

An interesting result from lemma 4 is that d_i in any graph cannot differ by more than 1 in order for the graph to have a maximum $H(\boldsymbol{d})$. For example, the network with $\boldsymbol{d}(2,2) = \{2,2,3,3,3,3\}$ ($|L| = 8$) has the maximum $H(\boldsymbol{d})$ among all networks with $n = 6$ and $|L| = 8$. Thus,

$$\bar{r}(l) = H(\boldsymbol{d}(\alpha,j)), \text{ for } l \ge n+2;\ \alpha = \text{gilb}(D/n),$$

$$\text{and } j = n \cdot (\alpha+1) - D.$$

By lemmas 1 & 4, we can determine,

$$r(n-1),\ r(n),\ r(n+1),\ \bar{r}(l) \text{ for } n+2 \le l \le n^*,$$

which is an ordered, non-decreasing sequence.

A binary search method [8] is suggested to find a^* such that:

$\bar{r}(a^*) \ge P_0$ and $\bar{r}(a^*-1) < P_0$ for $a^* > n+2$,

$\bar{r}(a^*) \ge P_0$ and $r(a^*-1) < P_0$ for $a^* = n+2$,

$r(a^*) \ge P_0$ and $r(a^*-1) < P_0$ otherwise.

3.2 *A Branch & Bound Algorithm for $P_n(l)$*

$P_n(l)$ is described in (3) & (4). The solution space consists of all $\binom{n^*}{l}$ combinations. In order to solve $P_n(l)$, we order the links according to $c_{i,j}$ in a non-decreasing sequence. The links are relabeled: the link with rank k becomes link e_k. We use a tree to represent all combinations. Figure 2a is an example fully connected network with 4 nodes. Figure 2b is the combinatorial tree for that network; the problem is then $P_4(4)$. The links of a combinatorial tree are labeled by possible choices of link e_k. The links from the root (level-0) node to level-1 nodes are specified by $\{e_i\}_{i=1}^{n^*-l+1}$. Links from the level-k node, pointed by the link with label e_i, to level-$(k+1)$ nodes are specified by $\{e_j\}_{j=i+1}^{n^*-l+(k+1)}$. For example, links from node 3 at level-2, pointed by the link with label e_2, to level-3 nodes are specified by e_3, e_4, e_5. The path from the root to the leaf defines a possible choice of l links. Thus, the solution space is defined by all paths from the root node to a leaf node. There are $\binom{6}{4} = 15$ leaf nodes in the tree of figure 2b.

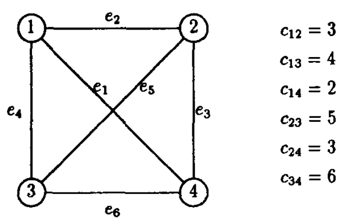


Figure 2a. A Fully Connected Network with 4 Nodes

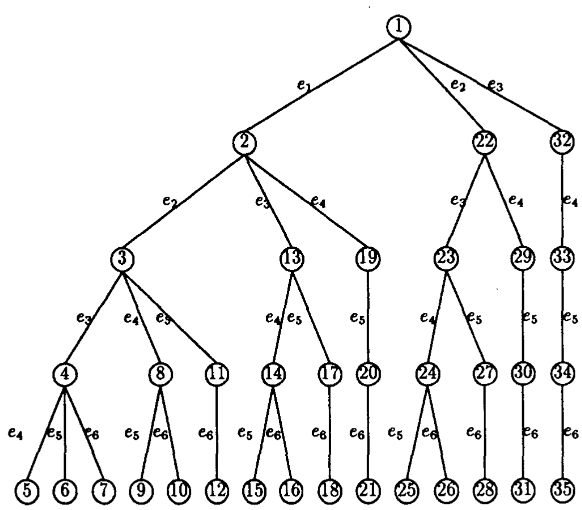


Figure 2b. The Combinatorial Tree for the Network in Figure 2a

To find an optimal solution, we do not consider all combinations since it is time-consuming. We apply a least-cost branch & bound algorithm to find the optimal solution by traversing only a small portion of the combinatorial tree. The branch & bound method has 3 decision rules that provide the method for —

1. Estimation of the lower bound of the objective function at every node of the combinatorial tree.
2. Feasibility testing at each leaf node.
3. Selecting the next live node for branching, and terminating the algorithm.

3.2.1 Estimation of the lower bound of the objective function at node v

Let:

- v be the current node in the combinatorial tree
- links defined by the path from the root to node v be $\{e_{i_j}\}_{j=1}^{k}$
- U be the set of $l-k$ links needed to be chosen from the remaining link set $\{e_j\}_{j=i_k+1}^{n^*-l+(k+1)}$
- $g(v)$ be the smallest cost that appears in the complete choices generated from node v.

$$g(v) \equiv \min\left\{ \sum_{j=1}^{k} c(e_{i_j}) + \sum_{e \in U} c(e)\text{: for all } U \right\}.$$

$$= \sum_{j=1}^{k} c(e_{i_j}) + \sum_{j=1}^{l-k} c(e_{i_k+j}). \qquad (7)$$

Eq (7) results from the ordering of the $\{c(e_i)\}$.

3.2.2 Feasibility testing at a leaf node

Whenever a leaf node is reached, the feasibility test is applied to it. A leaf node associated with path $\{e_{i_j}\}_{j=1}^{l}$ is *feasible* if $f(\boldsymbol{x}) \geq P_0$, where x is defined by $\{e_{i_j}\}_{j=1}^{l}$. The evaluation of $f(\boldsymbol{x})$ is time consuming. Fortunately, it is not necessary to evaluate $f(\boldsymbol{x})$ at every leaf node. Let network G with $\boldsymbol{d}$ be the network defined by $\{e_{i_j}\}_{j=1}^{l}$. The following test is used to determine the feasibility of a leaf node.

- If G is not connected, then $f(\boldsymbol{x}) = 0$ and the leaf node is infeasible.
- If $H(\boldsymbol{d}) < P_0$, then the leaf node is infeasible.
- Otherwise, compute $f(\boldsymbol{x})$ by the algorithm in [3]. Verify if $f(\boldsymbol{x}) \geq P_0$. □

3.2.3 Selection of a branching node and termination condition

To handle the generation of the combinatorial tree, a data structure (live-node list) records all live nodes that are waiting to be branched. The search strategy of the branch & bound algorithm is least cost. That is, the node, say v, selected for next branching is the live node whose $g(v)$ is the smallest among all the nodes in the live-node list. Two nodes, the first node on the next level and the next node on the same level, are generated from node v if these nodes exist, and added to live-node list. For example, see the combinatorial tree in figure 2b. If node-3 is selected for branching, then the 2 nodes, node-4 and node-13, are generated from node-3. If node-5 is selected from the live-node list for branching then there exists only the next node on same level, and only node-6 is generated. Traversal of the combinatorial tree starts at root node-0 and stops when the live-node list is empty. In addition, an upper bound cost

Algorithm 2. Branch & Bound Algorithm for Solving $P_n(l)$

```
Initialize the live-node list to be empty;
  /* live-node list is a priority queues storing live nodes */
Put root node v0 on the live-node list;
Set g(v0) := 0;
Set UC := ∞;
while live-node list is not empty do
begin
  choose node v with the minimum value of g(v) from the live-node list;
  Set S = Ø;
  if g(v) ≥ UC then
    remove node v from the live-node list;
  else begin
    Put the first child and next brother of node v into set S;
    for each node u in S do
      begin
        compute g(u) by (7);
        if node u is at level l then
          begin /* feasibility testing */
            if the network specified by the path from v0 to u is not connected or H(d) < P0
            then node u is infeasible
            else if f(x} ≥ P0 then node u is feasible and
              set min(UC, g(u))
          end
        else if g(u) < UC then
          insert node u into the live-node list
    end;
  remove node v from the live-node list
  end;
end;
output the answer: node w and the optimal value g(w) = UC.
```

(UC) is associated with the branch & bound algorithm. $UC=\infty$, initially, and is updated to be $\min(UC, g(u))$ whenever a feasible leaf node u is reached. If node v satisfies $g(v) \geq UC$, then it is bounded since further branching from v does not lead to a better solution. When the live-node list becomes empty, the optimal solution is defined by the path from the root to the leaf node u with $g(u) = UC$. Optimal cost UC is the output of algorithm 2.

4. NUMERICAL EXAMPLES & RESULTS

4.1 *Example 1*

A network has 5 nodes with $p = 0.8$; $P_0 = 0.90$. The link costs are:

e_k	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
(i,j)	(1,5)	(4,5)	(1,2)	(2,3)	(3,4)	(2,5)	(3,5)	(1,3)	(2,4)	(1,4)
$c_{i,j}$	25	29	32	34	36	45	52	54	58	62

The topological optimization can be formulated as the mathematical programming problem:

$$z = \text{Minimize} \sum_{i=1}^{4} \sum_{j=i+1}^{5} c_{i,j} \cdot x_{i,j}$$

subject to:

$$f(x) \geq 0.90.$$

The decision variables, $x = \{x_{1,2}, x_{1,3}, \ldots, x_{4,5}\}$, are the selecting status of the link (i,j). The details of the solution are:

1. By lemmas 1 & 4:

$$r(6) = 0.8520 \leq 0.90$$

$$\bar{r}(7) = H(2,3,3,3,3) = 0.9357 > 0.90.$$

Thus, set $a^* = 7$ and $z^* = \infty$.

2. Apply Algorithm-2 to solve $P_5(7)$.

2.1 The combinatorial tree for $P_5(7)$ is shown in figure 3. The nodes are numbered according to the sequence of Algorithm-2. The optimal solution is:

$$(x_{1,2}, x_{1,3}, x_{1,4}, x_{1,5}, x_{2,3}, x_{2,4}, x_{2,5}, x_{3,4}, x_{3,5}, x_{4,5})$$

$$= (1,1,0,1,1,0,1,1,0,1).$$

$z(7) = 255$. Since $z(7) < \infty$, then $z^* = z(7) = 255$.

2.2 $\underline{z}(8) = \sum_{i=1}^{8} c(e_i) = 307.$

2.3 $\underline{z}$ (8) $\geq z^*$, then $z^* = 255$ is the optimal MP; the procedure is complete.

This example takes 0.44 cpu sec. for the solution on SUN4 Sparc workstation.

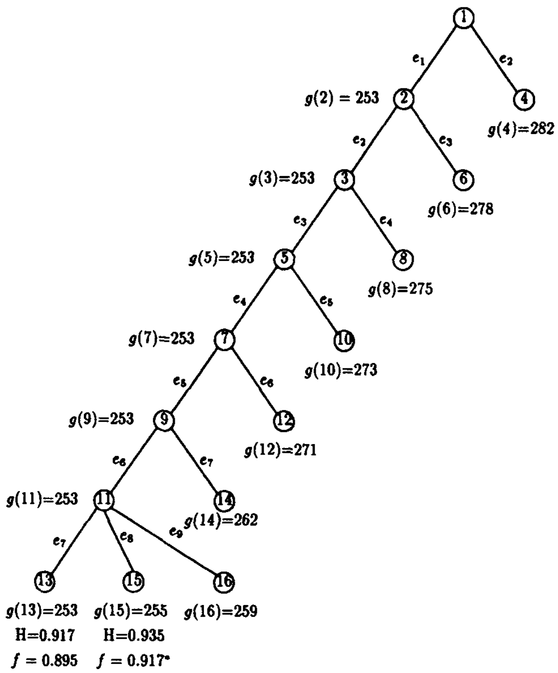


Figure 3. The Combinatorial Tree for $P_5(7)$

4.2 *Example 2.*

Find a minimum-cost topology for a non-fully connected network $G=(N,L,p)$ in figure 4a.

$|N|=20,\ |L|=30,\ p=0.95.$

$c_{i,j}$ is given in figure 4a.

$P_0 = 0.90.$

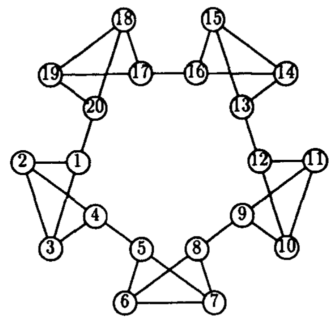
The problem is formulated as:

$$\text{Minimize} \sum_{(i,j)\in L} c_{i,j}\cdot x_{i,j}$$

subject to:

$$f(x) \geq 0.90.$$

Using Algorithm-2, the optimal topology is shown in figure 4b. $z^* = 596$. This example takes 518 cpu sec. for the solution on SUN4 Sparc workstation.



$c_{1,2} = 7$	$c_{10,12} = 9$
$c_{1,3} = 40$	$c_{11,12} = 76$
$c_{2,3} = 46$	$c_{12,13} = 19$
$c_{2,4} = 51$	$c_{13,14} = 21$
$c_{3,4} = 30$	$c_{13,15} = 11$
$c_{4,5} = 8$	$c_{14,15} = 100$
$c_{5,6} = 34$	$c_{14,16} = 9$
$c_{5,7} = 19$	$c_{15,16} = 22$
$c_{6,7} = 84$	$c_{16,17} = 13$
$c_{6,8} = 24$	$c_{17,18} = 33$
$c_{7,8} = 66$	$c_{17,19} = 65$
$c_{8,9} = 27$	$c_{18,19} = 4$
$c_{9,10} = 60$	$c_{18,20} = 59$
$c_{9,11} = 25$	$c_{19,20} = 44$
$c_{10,11} = 11$	$c_{1,20} = 10$

Figure 4a. A Non-Fully Connected Network with 20 Nodes and 30 Links

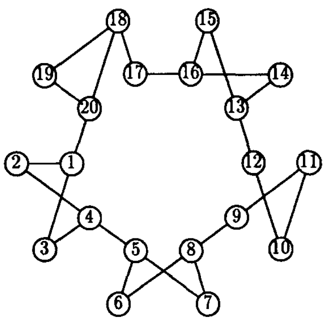


Figure 4b. The Optimal Topology of Example 2.

4.3 *Simulation Results for Algorithm-2*

The real execution times of the algorithm depend on n, m (number of available links), p, and P_0. Two types of the networks are considered:

1. All links between any two nodes are available: $m=n^*$. Max size = (12,66).
2. The available links are randomly generated: $m \leq 1.5\cdot n$. Max size = (20,30).

The link costs are randomly generated in [1,100]. For each (n,m,p,P_0), we randomly generate 20 set of data and solve them on a SUN4 Sparc workstation. The results are summarized in tables 1 & 2; they include:

- average number of leaf nodes generated,
- average number of computations of $f(x)$,
- average execution time (cpu sec).

Tables 1 & 2 show that Algorithm-2 appreciably reduces the number of unnecessary tries for infeasible leaf nodes. For example, if you apply an exhaustive search to a problem with size

TABLE 1
Computation Results for Fully Connected Graphs

n	m	p	P_0	Average Number of computations of $f(x)$	Average Number of leaf nodes generated	cpu sec
8	28	0.90	0.85	220	453	3.8
			0.90	155	279	5.1
			0.95	96	4301	12.5
		0.95	0.85	49	123	0.6
			0.90	241	623	3.5
			0.95	633	633	17.3
9	36	0.90	0.85	495	1199	14.8
			0.90	766	1500	46.9
			0.95	201	18349	58.4
		0.95	0.85	196	1421	5.5
			0.90	878	2902	15.8
			0.95	2940	8327	96.1
10	45	0.90	0.85	691	1558	65
			0.90	3254	5981	248
			0.95	435	23977	133
		0.95	0.85	1460	16588	78
			0.90	8668	136444	519
			0.95	14000	40677	649
11	55	0.90	0.85	7343	18164	1104
			0.90	15061	40388	1665
			0.95	1190	755436	3488
		0.95	0.85	2628	25260	161
			0.90	3747	19162	342
			0.95	7743	29069	1558
12	66	0.90	0.85	12597	38101	2351
			0.90	12280	33207	2418
			0.95	589	84543	1048
		0.95	0.85	14066	160032	1318
			0.90	3971	27320	415
			0.95	12162	44193	2444

TABLE 2
Computation Results for non-Fully Connected Graphs

n	m	p	P_0	Average Number of computations of $f(x)$	Average Number of leaf nodes generated	cpu sec
14	21	0.90	0.81	518	691	293
			0.84	617	712	547
			0.87	71	392	180
			0.90	220	920	748
		0.95	0.81	469	1452	50
			0.84	549	1709	117
			0.87	406	753	80
			0.90	878	1308	445
16	24	0.90	0.81	850	1229	1608
			0.84	2344	2592	9675
			0.87	941	1737	583
			0.90			†
		0.95	0.81	2811	9187	1212
			0.84	4621	13737	1719
			0.87	2012	3360	1585
			0.90	6244	11071	5821
18	27	0.90	0.81	1159	2374	2527
			0.84	3734	9627	1491
			0.87	1976	3518	1714
			0.90			†
		0.95	0.81	1349	8702	2307
			0.84	2849	11001	3590
			0.87	4456	12001	10220
			0.90	1480	2160	2432
20	30	0.90	0.81	171	360	3326
			0.84	395	1023	2273
			0.87	478	863	275
			0.90			†
		0.95	0.81	560	921	1365
			0.84	207	243	818
			0.87			†
			0.90			†

†Most of these problems cannot be solved by the algorithm in 20k cpu sec (5.5 cpu hours)

$(n,m) = (12,66)$, say $(n,m,p,P_0) = (12,66,0.90,0.95)$, then the total number of leaf nodes is:

$$\binom{66}{11} + \binom{66}{12} + \dots + \binom{66}{66} > 2^{65}.$$

However, the average number of leaf nodes generated by Algorithm-2 (for table 1) is:

$$84\ 543 < 2^{17} \text{ for } (n,m,p,P_0) = (12,66,0.90,0.95),$$

because we apply the $r(l)$ and branch & bound approach. Furthermore, the average number of computations of $f(x)$ is only 589. This means that the connectivity test and reliability bound test (line 18 of Algorithm-2) can be used to discard most infeasible leaf nodes before computing $f(x)$. In this case, we discarded 83 954 (84543-589) infeasible leaf nodes, on average. Thus, the connectivity test and reliability bound are very effective.

ACKNOWLEDGMENT

This work was supported by the National Science Council of the Republic of China under grant NSC 77-0404-E009-24. We are pleased to thank the editors & referees for their comments which helped to improve the presentation of the paper.

REFERENCES

[1] K. K. Aggarwal, Y. C. Chopra, J. S. Bajwa, "Topological layout of links for optimizing the s-t reliability in a computer communication system", *Microelectronics & Reliability*, vol 22, num 3, 1982, pp 341–345.

[2] K. K. Aggarwal, Y. C. Chopra, J. S. Bajwa, "Topological layout of links for optimizing the overall reliability in a computer communication system", *Microelectronics & Reliability*, vol 22, num 3, 1982, pp 347–351.

[3] M. Ball, R. M. Slyke, "Backtracking algorithms for network reliability analysis", *Ann. Discrete Mathematics*, vol 1, 1977, pp 49–64.
[4] R. R. Boorstyn, H. Frank, "Large-scale network topological optimization", *IEEE Trans. Communications*, vol COM-25, num 1, 1977, pp 29–47.
[5] Y. C. Chopra, B. S. Sohi, R. K. Tiwari, K. K. Aggarwal, "Network topology for maximizing the terminal reliability in a computer communication network", *Microelectronics & Reliability*, vol 24, num 5, 1984, pp 911–913.
[6] B. N. Clark, E. M. Neufeld, C. J. Colbourn, "Maximizing the mean number of communicating vertex pairs in series-parallel networks", *IEEE Trans. Reliability*, vol R-35, 1986 Aug, pp 247–251.
[7] C. J. Colbourn, *The Combinatorics of Network Reliability*, 1987; Oxford University Press.
[8] E. Horowitz, S. Sahni, *Fundamentals of Computer Algorithms*, 1984; Computer Science Press.
[9] R.-H. Jan, "Design of reliable networks", *Computers and Operations Research*, vol 20, 1993 Jan, pp 25–34.
[10] S. Kiu, D. F. McAllister, "Reliability optimization of computer-communication networks", *IEEE Trans. Reliability*, vol 37, 1988 Dec, pp 475–483.
[11] K. T. Newport, P. K. Varshney "Design of survivable communications networks under performance constraints", *IEEE Trans. Reliability*, vol 40, 1991 Oct, pp 433–440.
[12] S. Sahni, *Concepts in Discrete Mathematics*, 1985, pp 136–139; Prentice Hall.
[13] I. M. Soi, K. K. Aggarwal, "Reliability indices for topological design of computer communication networks", *IEEE Trans. Reliability*, vol R-30, 1981 Dec, pp 438–443.
[14] A. N. Venetsanopoulos, I. Singh, "Topological optimization of communication networks subject to reliability constraints", *Problem of Control and Information Theory*, vol 15, num 1, 1986, pp 63–78.

AUTHORS

Dr. Rong-Hong Jan; Dept. of Information and Computer Science; National Chiao Tung University; Hsinchu 30050, TAIWAN, Rep. of CHINA.

Rong-Hong Jan (M'87) received the BS & MS in Industrial Engineering, and the PhD in Computer Science from National Tsing Hua University, Taiwan, in 1979, 1983, and 1987. He joined the Dept. of Computer and Information Science, National Chiao Tung University, in 1987, where he is an Associate Professor. During 1991-1992, he was a Visiting Associate Professor in the Dept. of Computer Science, University of Maryland, College Park. His research interests include computer networks, distributed systems, network reliability, and operations research.

Fung-Jen Hwang; Dept. of Information and Computer Science; National Chiao Tung University; Hsinchu 30050, TAIWAN, Rep. of CHINA

Fung-Jen Hwang received the BS & MS in Computer Science from National Chiao Tung University, Taiwan, in 1986 & 1989. She joined the Veterans General Hospital as a Computer Engineer in 1989. Her research interests include computer networks and network reliability.

Dr. Sheng-Tzong Cheng; Dept. of Computer Science; University of Maryland; College Park, Maryland 20742 USA

Sheng-Tzong Cheng is working toward his PhD degree in the Dept. of Computer Science at University of Maryland, College Park. He received the BS & MS from National Taiwan University, Taiwan, in 1985 & 1987. His research interests include network flow control, network reliability, and real-time system design.

Manuscript TR91-304/92-076 received 1990 November 11; revised 1992 April 27.

IEEE Log Number 01691 ◄TR►