# AN ALGORITHM FOR DESIGNING SURVIVABLE NETWORKS

**Yogesh K. Agarwal**

***Yogesh K. Agarwal*** *is a member of technical staff in the Network Design and Optimization Department of AT&T Bell Laboratories in Holmdel, New Jersey. Mr. Agarwal joined the company in 1985 and is responsible for developing algorithms for high-capacity customer-network design tools. He has an M.S. in industrial engineering from the National Institute for Training in Industrial Engineering, Bombay, India, and a Ph.D. in operations research from Case Western Reserve University.*

In this paper, we consider the design of telecommunications networks using high-capacity transport facilities that can survive under a single-link failure scenario. The model considers three priority levels for circuits: high, normal, and low. Each high-priority circuit is assigned a primary path and a link-disjoint alternate path. The model ensures that all high-priority circuits can be rerouted in case of a link failure by preempting low-priority circuits if necessary. The problem is formulated as an integer program solved by a heuristic Lagrangean-relaxation technique and a partial branch-and-bound approach.



## Introduction

We consider the problem of designing telecommunications networks using high-capacity facilities. The input to the problem includes the node locations and the circuit requirements among those locations. A circuit can either be connected directly on a low-capacity facility or multiplexed with other circuits using a high-capacity facility. The cost of a high-capacity facility is usually significantly less than the total cost of low-capacity facilities needed for all the circuits that can be multiplexed on a single high-capacity facility. The objective of the network design problem is to build a network of high-capacity facilities and route individual circuits in this network to satisfy all circuit requirements at a minimal cost.

We also address the problem of designing *survivable* facility networks. Survivability, in this context, is the ability to reroute critical circuits through alternate paths in the network in case of a link failure. To define survivability more rigorously, we assume that each circuit is assigned one of three possible priority levels: high, normal, and low, represented by numbers 2, 1, and 0, respectively. It is assumed that, in the event of a link failure, high-priority circuits can preempt low-priority circuits to find an alternate route. Normal-priority circuits can not preempt, nor are they preempted by, other priority levels. A network is considered survivable under a particular failure scenario if *all* high-priority circuits can be rerouted by either using spare capacity or

preempting the low-priority circuits if necessary.

The formulation presented in this paper addresses the design of networks that are survivable under single-link failures. By single-link failure, we mean the failure of *all* facilities on a link. This is a more realistic assumption than assuming only single-facility failures because multiple facilities between a pair of nodes are most likely to be provisioned over the same physical path and, therefore, will most probably fail together.

The solution to the network design problem specifies the topology of the network and the route taken by each circuit. The survivable network design problem also has to specify the alternate paths taken by the high-priority circuits in case of a link failure. Conceivably, for a given circuit, a different alternate path could be specified for each failure scenario. This would lead to a very complex problem formulation because of many different failure scenarios. Besides, the flexibility of choosing a different path based on a failure scenario is not realistic for operational reasons.

We assume that each high-priority circuit is assigned a primary path and a link-disjoint alternate path. Because the two paths are link-disjoint, one will always be available in any single-link failure scenario. Initially, all circuits are routed along their primary paths. If the failed link is on the primary path of a circuit, the circuit is switched to its alternate path; otherwise, the circuit stays on its primary path. It is also assumed that all circuits revert to their primary paths as soon as the failed link is back in service. The objective is to design a minimum-cost network that satisfies all circuit demands and is survivable according to these criteria.

#### Direct-Route Analysis

Obviously, for a given set of circuit requirements, a network consisting of only high-capacity facilities would be more expensive than one in which low-capacity facilities are also used. For operational reasons, it is desirable for a circuit to either ride on high-capacity facilities end-to-end or use a low-capacity facility on a direct link without any multiplexing. (The latter option is termed a *direct route.*) In other words, switching from a high-capacity facility to a low-capacity facility, or vice versa, is not permitted. The algorithm presented in this paper is designed to perform the direct-route analysis. It creates a backbone network of high-capacity facilities and routes each circuit either on the backbone network or on a direct route, to minimize the total network cost, including the cost of direct routes.

Clearly, even if direct-route analysis is not desired, the same algorithm can be applied simply by using an arbitrarily high cost for the direct-route facilities. Similarly, if certain types of circuits (e.g., those of high priority) must ride on the backbone network, this can be done with a high direct-route cost for these circuits.

#### Literature Review

The problem of routing traffic over a given network topology, which is equivalent to the minimum-cost, multicommodity flow problem, is treated by many authors in different contexts.[1–3] A topology design problem, similar to the one considered here, also occurs in transportation networks and has been treated in that context by Dantzig et al.,[4] Boffey and Hinxman,[5] Boyce et al.,[6] and others.

The problem of survivable network design with a different survivability criterion than ours is addressed by Steiglitz et al.[7] While Steiglitz et al. address the problem solely from the topology viewpoint, Pirkul and Narasimhan[8] consider the problem of finding primary and secondary traffic routes for a given topology in the context of packet-switched data networks. The survivability issue is closely related to the two-connectedness of graphs; the characteristics and design of such graphs have been studied by Kajitani and Ueno.[9]

An important subproblem related to the survivability issue is that of finding the shortest link-disjoint path-pair in a network. This problem was first treated by Suurballe[10] (see also Suurballe and Tarjan[11]), who presented an efficient algorithm that finds the optimum

**Panel 1. Basic Notation**

| | | |
|---|---|---|
| $i,j$ | = | link indexes. |
| $A$ | = | set of all links. |
| $l$ | = | circuit index. |
| $k$ | = | index for routing alternatives. |
| $p$ | = | priority level (0, 1, or 2). |
| $C_p$ | = | set of circuits with priority level $p$. |
| $C$ | = | $C_0 \cup C_1 \cup C_2$, set of all circuits. |
| $S_l$ | = | index set of routing alternatives for circuit $l$. |
| $\mathbf{a}_l^k$ | = | binary column vector representing primary route for circuit $l$ associated with routing alternative $k \in S_l$. |
| $\mathbf{b}_l^k$ | = | binary column vector representing alternate route for high-priority circuit $l$ associated with routing alternative $k \in S_l$. (Note that for given circuit $l$ and routing alternative $k$, $\mathbf{a}_l^k \cdot \mathbf{b}_l^k = 0$, because the primary and alternate paths are link-disjoint.) |
| $x_l^k$ | = | routing variables; binary decision variables where $x_l^k = 1$ if routing alternative $k$ is selected for circuit $l$, and $x_l^k = 0$ otherwise. |
| $c_l$ | = | cost of satisfying circuit $l$ by direct route. |
| $c_l^k$ | = | cost of routing alternative $k$ for circuit $l$; $c_l^k = 0$ for a backbone route, and $c_l^k = c_l$ for direct route. |
| $K_j$ | = | cost of a high-capacity facility for link $j$. |
| $y_j$ | = | topology variables; integer decision variables representing the number of high-capacity facilities on link $j$. |
| $T$ | = | capacity (in number of circuits) of a high-capacity facility. |

path-pair in two applications of any standard shortest-path algorithm. However, this algorithm is valid only when the same distance metric is applicable for both the primary and alternate paths. Li et al.[12] have considered several variations of this problem and have shown that all of them are strongly NP-complete.

### Mathematical Formulation

In this section, we present an integer programming formulation of the problem. In this formulation, each circuit route is represented by a binary column vector whose $i$th element is 1 if link $i$ is used on the route, and 0 otherwise. A direct route can be represented by a column of all zeroes because it does not use any high-capacity links.

Panel 1 shows our basic notation; other terms will also be developed for the formulation. Note that the set $S_l$ of routing alternatives for circuit $l$ consists of columns $\mathbf{a}_l^k$ for low- and normal-priority circuits. However, for high-priority circuits, it consists of path-pairs $(\mathbf{a}_l^k, \mathbf{b}_l^k)$ of link-disjoint paths. Although a particular path may appear in several path-pairs, either as a primary or as an alternate, each pair in a set is distinct from all others.

Even for a moderate-sized problem, there can be many possible paths and/or path-pairs between any two nodes in the network. Because it is not practical to enumerate them all in advance, we use a column-generation scheme for the paths and path-pairs by solving a shortest-path or path-pair problem, as described later.

**Additional Notation.** Using the notation shown in Panel 1, the flow of priority $p$ circuits on any link $j$ can be expressed as a function of routing variables $x$ as follows:

$$F_j^p(\mathbf{x}) = \sum_{l \in C_p} \sum_{k \in S_l} a_{jl}^k \, x_l^k$$

Consider two links, $i$ and $j$. If link $i$ fails, the flow of normal- and high-priority circuits on link $j$ can be affected in two ways:

- Flow on link $j$ will increase because of those high-priority circuits that use $i$ on the primary path and $j$ on the alternate path. The number of such circuits is denoted by $g_{ij}(\mathbf{x})$ expressed as a function of $x$ as follows:

$$g_{ij}(\mathbf{x}) = \sum_{l \in C_2} \sum_{k \in S_l} a_{il}^k \, b_{jl}^k \, x_l^k$$

- On the other hand, if a normal- or high-priority circuit uses both links $i$ and $j$ on its primary path, the flow on link $j$ will decrease because the circuit will no longer be using its primary path. The number of such circuits is denoted by $h_{ij}(\mathbf{x})$ expressed as a function of $x$ as

follows:

$$h_{ij}(\mathbf{x}) = \sum_{l \in C_1 \cup C_2} \sum_{k \in S_l} a_{il}^k \, a_{jl}^k \, x_l^k$$

Note that low-priority circuits are not included in this term because they can be preempted in a failure scenario and the entire capacity that they use can be assumed to be available in a link failure. We are interested only in the flow of normal- and high-priority circuits in a link failure.

The net increase in the flow of normal- and high-priority circuits on link $j$ as a result of a failure of link $i$ would be $g_{ij}(\mathbf{x}) - h_{ij}(\mathbf{x})$. Let $G_j(\mathbf{x})$ denote the maximum net increase in flow of link $j$ as a result of *any* single-link failure. Thus:

$$G_j(\mathbf{x}) = \underset{i}{\text{Maximum}} \, [\, g_{ij}(\mathbf{x}) - h_{ij}(\mathbf{x})]$$

The quantities $F_j^p(\mathbf{x})$ and $G_j(\mathbf{x})$ are referred to simply as $F_j^p$ and $G_j$ in subsequent sections.

**Integer Programming Formulation.** Using this notation, the network design problem can be written as the following integer program (P).

$$\text{Minimize} \qquad Z_P = \sum_{l \in C} \sum_{k \in S_l} c_l^k \, x_l^k + \sum_{j \in A} K_j \, y_j \qquad \text{(P)}$$

$$\text{subject to:} \quad F_j^0(\mathbf{x}) + F_j^1(\mathbf{x}) + F_j^2(\mathbf{x}) \le T y_j \quad \forall \; j \in A \quad \text{(P1)}$$

$$F_j^1(\mathbf{x}) + F_j^2(\mathbf{x}) + G_j(\mathbf{x}) \le T y_j \quad \forall \; j \in A \quad \text{(P2)}$$

$$\sum_{k \in S_l} x_l^k = 1 \quad \forall \; l \in C \qquad \text{(P3)}$$

$$x_l^k = 0 \text{ or } 1 \quad \forall \; l, k \qquad \text{(P4)}$$

$$y_j \ge 0 \text{ and integer} \qquad \text{(P5)}$$

The objective function $Z_P$ is the sum of the cost of direct routes and the cost of the high-capacity backbone network. Constraint set (P1) ensures that the number of circuits routed through any link does not exceed the capacity of the facilities on that link. Constraint set (P2) ensures that, in any single-link failure, when the circuits on the failed link are switched to their respective alternate paths, there is enough capacity in the network to accommodate such rerouting. Notice that the term $F_j^0$ appears in constraint set (P1), but not in (P2), because it is assumed that low-priority circuits can be preempted to make room for high-priority circuits. Constraint sets (P4) and (P5) express the fact that variables $x_l^k$ and $y_j$ are binary and integer, respectively. Note that:

$$\max \, [\, (F_j^0 + F_j^1 + F_j^2), \, (F_j^1 + F_j^2 + G_j) \,]$$

is the maximum flow that will occur on link $j$. This quantity is also referred to simply as "flow" in subsequent sections.



**Problem Decomposition and Routing Subproblem**

If the topology variables $y$ are assigned specific values, then the problem reduces to one of finding the least-cost routing of circuits over a given high-capacity backbone. The backbone cost is already determined by assignment of values to variables $y_j$. Because the only remaining cost element is the cost of direct routes, the objective of this problem is to route as many circuits over the backbone as possible and assign the rest to direct routes to minimize the total cost of direct routes. We call this problem the *routing subproblem*. In the next section, we present a heuristic algorithm, based on Lagrangean relaxation, to solve this problem.

Given an algorithm for the routing subproblem, a natural approach to solve the original problem is to perform a search over the domain of variables $y_j$ to find a solution with the least overall cost. This problem is solved by a branch-and-bound approach in which the lower bound on the master problem is obtained by a

modified version of the routing subproblem algorithm. The details of this procedure are described in the master problem section of this paper.

**Routing Subproblem Algorithm**

If variables $y_j$ are assigned values $Y_j$, then the routing subproblem (S) is as follows:

$$\text{Minimize} \quad \sum_{l \in C} \sum_{k \in S_l} c_l^k \, x_l^k + \sum_{j \in A} K_j \, Y_j \qquad \text{(S)}$$

$$\text{subject to:} \quad F_j^0(\mathbf{x}) + F_j^1(\mathbf{x}) + F_j^2(\mathbf{x}) \le T\,Y_j \quad \forall\ j \in A \quad \text{(S1)}$$

$$F_j^1(\mathbf{x}) + F_j^2(\mathbf{x}) + G_j(\mathbf{x}) \le T\,Y_j \quad \forall\ j \in A \quad \text{(S2)}$$

$$\sum_{k \in S_l} x_l^k = 1 \quad \forall\ l \in C \qquad \text{(S3)}$$

$$x_l^k = 0 \text{ or } 1 \quad \forall\ l, k \qquad \text{(S4)}$$

Our approach for solving this problem is motivated by the theory of Lagrangean relaxation and Everett's theorem.[13] We introduce the Lagrangean dual multipliers $u_j$ and $v_j$ to dualize constraint sets (S1) and (S2), respectively. A slight rearrangement of the terms in the objective function yields the following Lagrangean problem (SL):

$$\max_{u,v} \left[ \min_{x} L(\mathbf{x},\mathbf{u},\mathbf{v}) = \left\{ \sum_{l \in C} \sum_{k \in S_l} c_l^k \, x_l^k + \sum_{j \in A} [u_j F_j^0 + (u_j + v_j)(F_j^1 + F_j^2) + v_j G_j] - \sum_j [K_j - (u_j + v_j)T] Y_j \right\} \right] \qquad \text{(SL)}$$

subject to:

$$\sum_{k \in S_l} x_l^k = 1 \quad \forall\ l \in C$$

$$x_l^k = 0 \text{ or } 1 \quad \forall\ l, k$$

$$u_j,\ v_j \ge 0 \quad \forall\ j$$

The last term in the objective function is not a function of $\mathbf{x}$ and can be ignored as far as the inner minimization is concerned. By substituting for $F_j^p$ and $G_j$, as per the definitions given in the section on additional notation, the cost of the second term can be apportioned among individual circuits and $L_{u,v}(\mathbf{x})$ [i.e., $L(\mathbf{x},\mathbf{u},\mathbf{v})$ for given values of $\mathbf{u}$ and $\mathbf{v}$] can be rewritten as follows:

$$L_{u,v}(\mathbf{x}) = \sum_{l \in C} \sum_{k \in S_l} c_l^k \, x_l^k + \sum_{l \in C_0} \sum_{k \in S_l} \left[ \sum_{j \in A} u_j \, a_{jl}^k \right] x_l^k$$

$$+ \sum_{l \in C_1} \sum_{k \in S_l} \left[ \sum_{j \in A} (u_j + v_j) a_{jl}^k - \sum_{j \in A} \left[ \sum_{\substack{i \in A \\ i \ne j}} \delta_{ij} \, a_{il}^k \right] a_{jl}^k \, v_j \right] x_l^k$$

$$+ \sum_{l \in C_2} \sum_{k \in S_l} \left[ \sum_{j \in A} (u_j + v_j) a_{jl}^k - \sum_{j \in A} \left[ \sum_{\substack{i \in A \\ i \ne j}} \delta_{ij} \, a_{il}^k \right] a_{jl}^k \, v_j \right.$$

$$\left. + \sum_{j \in A} \left[ \sum_{\substack{i \in A \\ i \ne j}} \delta_{ij} \, a_{il}^k \right] b_{jl}^k \, v_j \right] x_l^k$$

where:

$$\delta_{ij} = \begin{cases} 1 \text{ if } g_{ij}(\mathbf{x}) - h_{ij}(\mathbf{x}) = G_j \\ 0 \text{ otherwise} \end{cases}$$

If $\delta_{ij} = 1$, we say that link $i$ is critical for link $j$ (or link $j$ is critical to link $i$). This means that failure of link $i$ causes

**Algorithm 1. The Shortest-Path or Path-Pair Problem**

Step 0: Let $Z = \infty$, and define the metric $\alpha$ for the primary path as $\alpha_j = (u_j + v_j)$.

Step 1: Determine the primary path by solving the shortest-path problem using metric $\alpha$, and let the cost of the primary path be $\theta_1$. If $\theta_1 \geq Z$, STOP.

Step 2: Define the metric $\beta$ for the alternate path as follows:

$$\beta_j = \begin{cases} \infty & \text{if } j \text{ is on the primary path} \\ 0 & \text{if } \delta_{ij} = 0 \text{ for all } i \text{ on the primary path} \\ v_j & \text{if } \delta_{ij} = 1 \text{ for any } i \text{ on the primary path} \end{cases}$$

Determine the alternate path by solving the shortest-path problem using metric $\beta$, and let the cost of the alternate path be $\theta_2$. If $\theta_2 = 0$, STOP. If $\theta_1 + \theta_2 < Z$, update $Z$ and save the incumbent.

Step 3: Let $j^*$ be the link on the alternate path with maximum $\beta_j$. Find the corresponding link $i^*$ on the primary path such that $\delta_{i^*j^*} = 1$. Let $\alpha_{i^*} = \infty$ and go to Step 1.

the maximum amount of flow on link $j$. A close examination of the above expression reveals the following points:

- A low-priority circuit contributes a cost $u_j$ to the objective function if it uses link $j$ on its path.
- A normal-priority circuit contributes $u_j + v_j$ for using link $j$ if $j$ is not critical to any other link on the path. On the other hand, if $j$ is critical to some link on the path of the circuit, the contribution of link $j$ is only $u_j$.
- The contribution of high-priority circuits with respect to links on the primary path is the same as for normal-priority circuits. Link $j$ on the alternate path makes an additional contribution of $v_j$ if any link on the primary path is critical for link $j$ (i.e., $\delta_{ij} = 1$ for some $i$ with $a_{il}^k = 1$).

It is possible that, in a degenerate case, for a given link $j$, there may be more than one link $i$ with $\delta_{ij} = 1$. In that case, the contribution $v_j$ needs to be added or subtracted only once. The expression of $L_{u,v}(\mathbf{x})$ above has been written assuming the nondegenerate case.

**Minimization of $L_{u,v}(\mathbf{x})$.** Note that minimization of $L_{u,v}(\mathbf{x})$ is equivalent to routing each circuit along a path to minimize its cost contribution to the objective function. For low-priority circuits, this amounts to solving the shortest-path problem over the metric $\mathbf{u}$. However, for normal- and high-priority circuits, the problem is not as straightforward. The cost of using link $j$ depends on which other links are used on the path and on the values of $\delta_{ij}$. However, $\delta_{ij}$ can be determined only if the routing of all the circuits is known. Also, for high-priority circuits, we must determine the least-cost pair of link-disjoint paths. The cost of using a link on the alternate path depends on which links are used on the primary path, which further complicates the solution of the shortest-path problem.

While minimization of $L_{u,v}(\mathbf{x})$ is almost trivial if there are no normal- or high-priority circuits, it is quite difficult in the presence of normal- and high-priority circuits. We use a two-pass heuristic procedure. In the first pass, circuits are routed one-by-one along the least-cost path (or path-pair) with respect to the flows prevailing at the time. At the end of the first pass, the routes for some of the circuits may no longer be optimal because the flows, and hence the $\delta_{ij}$ values, may have changed. In the second pass, the shortest path for each circuit is recomputed and the circuit is assigned to the new path. It is clear that, even by using the second pass several times,

we can hope to reach, at best, a local minimum.

**The Shortest Path or Path-Pair Problem.** As mentioned earlier, the optimum path for low-priority circuits can be determined by solving the standard shortest-path problem on metric $\mathbf{u}$ using any of the well-known algorithms, such as Dijkstra's.[14] For normal-priority circuits, we use Dijkstra's algorithm with the metric $\mathbf{u} + \mathbf{v}$. This amounts to ignoring the term $h_{ij}(\mathbf{x})$ in the formulation, thus avoiding the complication that arises from the cost of a link being dependent on other links used on the path. This leads to some suboptimality; however, we feel that the loss of optimality is not significant. The primary path for the high-priority circuits is treated in the same way.

The link-disjoint path-pair for high-priority circuits is determined by using Algorithm 1, shown in the panel. In Step 1, we determine the primary path; in Step 2, we determine the least-cost alternate path for the given primary path. If the cost of the alternate path on metric $\boldsymbol{\beta}$ turns out to be zero, then this is obviously the least-cost path-pair. However, if the cost of the alternate path is nonzero, we identify the link on the primary path whose inclusion on the primary path contributes the most to the cost of the alternate path. This link is made unavailable for the primary path in the next iteration, which may result in a better path-pair. The algorithm will terminate either in Step 1 or in Step 2, when it becomes obvious that making any more links unavailable for the primary path does not lead to a better solution. The scope of this algorithm can be expanded to perform a wider search, but the running time may become excessive.

**Solving the Lagrangean Problem (SL).** The Lagrangean problem (SL) is usually solved by a subgradient optimization technique, yielding a lower bound on the objective function of the original problem (S). However, a solution to problem (SL) will almost certainly be infeasible to problem (S) because (SL) is a relaxation of (S). A common approach found in the literature is to use heuristics that modify this solution to obtain feasibility.

Because solving (SL) to near-optimality using subgradient optimization can be quite expensive and also does not usually lead to a feasible solution of (S), we attempt to solve (S) directly. We develop a heuristic algorithm based on a multiplier-adjustment procedure. For given $\mathbf{u}$ and $\mathbf{v}$, let $\mathbf{x}_0$ be the solution to the inner minimization in (SL). If this $\mathbf{x}_0$ is feasible for (S) and, for every multiplier with nonzero value, the corresponding constraint in (S) is tight, then $\mathbf{x}_0$ is optimal for (S). However, such a solution may not exist because of the duality gap. If we relax the latter condition (i.e., not require that the corresponding constraint be tight for each multiplier with nonzero value), then this solution need not be optimal. However, it can be considered a "good" solution if the slack values for the constraints with nonzero multipliers are "small." In fact, according to Everett's theorem,[13] this solution is optimal for a similar problem in which the right-hand side of the constraints with nonzero multipliers is decreased by the amount of slack on those constraints. If $Z_0$ is the value of this solution, then it can be shown that $Z_0 - \mathbf{us}^1 - \mathbf{vs}^2$ is a lower bound on the optimum solution of (SL). Here $\mathbf{s}^1$ and $\mathbf{s}^2$ are the vectors of slacks for constraint sets (S1) and (S2), respectively.

We want to find a set of multipliers $\mathbf{u}, \mathbf{v}$ and the corresponding solution $\mathbf{x}_0$ to $L_{u,v}(\mathbf{x})$, so that $\mathbf{x}_0$ is feasible to (S), and the quantity $\mathbf{us}^1 + \mathbf{vs}^2$ is small. Algorithm 2 is a heuristic approach that finds such a solution.

The algorithm starts with $u_j, v_j = 0$ for all $j$, and a corresponding routing solution $\mathbf{x}_0$ is found. This solution, in all likelihood, violates the capacity constraint for some links. In Step 2, we identify the link $j^*$ with maximum violation; in Step 3, we increase the $u_j$ or $v_j$ for this link by some amount $\Delta$. In the next iteration, when we solve the routing problem again, the increased value of the multiplier will make link $j^*$ unattractive for some of the circuits, leading to a decrease in flow on this link. If the backbone topology has enough capacity to accommodate all circuits, then, eventually, such a solution may be found. Otherwise, some circuits will be assigned to direct

**Algorithm 2. Routing Subproblem**

Step 0: Set $u_j, v_j = 0$ for all $j$.

Step 1: Minimize $L_{u,v}(\mathbf{x})$ and let $\mathbf{x}_0$ be the solution. Determine the quantities $F_j^p$ and $G_j$ for this solution and, thus, the slacks $s_j^1$ and $s_j^2$ for constraint sets (S1) and (S2), respectively.

Step 2: If all slacks are nonnegative, STOP. Otherwise, determine the constraint $t$ that has the most negative slack and the corresponding link $j^*$.

Step 3: If $t$ is in constraint set (S1), let $u_{j^*} \leftarrow u_{j^*} + \Delta$; and if $t$ is in constraint set (S2), let $v_{j^*} \leftarrow v_{j^*} + \Delta$. (Here $\Delta$, the increase in the value of the multiplier, is determined based on a parameter using certain heuristic rules.) Go to Step 1.

routes. Note that the direct route is also an option that competes with the backbone routes while solving the shortest-path problem. As the multipliers increase, the cost of backbone routes gradually increases and may eventually exceed the cost of direct routes for some circuits. These circuits will be assigned to direct routes in the final solution.

Notice that we increase only one multiplier at a time, rather than several of them. This is because, if only one multiplier is increased at a time, it is much easier to ensure that the slack for constraints with positive multipliers is small in the final solution. This condition is important to ensure a good solution. In addition, if only one multiplier is changed at a time, reoptimization of $L_{u,v}(\mathbf{x})$ is much easier. The quantity $\Delta$, by which the multiplier is increased, is determined dynamically to reduce the infeasibility of the constraint in question by a certain amount. For example, $\Delta$ can be chosen to reduce the infeasibility on the link by at least 50 percent. Selection of a small $\Delta$ will produce a better solution; however, the convergence will be slow. On the other hand, a large $\Delta$ will produce faster convergence, but the solution quality may suffer.

### The Master Problem

The master problem (P) is solved by a partial branch-and-bound approach. A lower bound on the master problem is computed by a modified version of the subproblem algorithm described in the next section. Recall that the master problem is defined over the topology variables $y_j$. Initially, the integrality condition on the topology variables is dropped, and a lower bound is computed at the root node. Then, at every stage of branching, constraints of the form $y_j \geq Y_j$ or $y_j = Y_j$ (where $Y_j$ is an integer) are gradually added to the master problem. Let us call the former the type-1 constraint and the latter the type-2. The constraints are added so that the lower bound increases minimally with each added constraint. A feasible solution is reached when all variables have integer values. Partial backtracking can also be performed, subject to the computer time limit.

**A Lower Bound on the Master Problem.** In this section, we develop a lower bound on the master problem (P). This lower bound is developed for an arbitrary node of the branch-and-bound tree (i.e., we assume that some of the variables $y_j$ have a type-1 constraint imposed on them and the others have a type-2 constraint). We show that the problem of computing the lower bound reduces to the routing subproblem (S) with some special conditions.

To develop this lower bound, let us examine the Lagrangean relaxation of the master problem at an arbitrary node of the branch-and-bound tree. Let $B_1$ and $B_2$ be the index sets of variables with type-1 and type-2 constraints, respectively. Note that every link $j$ is either in $B_1$ or in $B_2$ (i.e., $B_1 \cup B_2 = A$). Because the integrality

**Algorithm 3. Lower Bound**

Step 0: Set $u_j, v_j = 0$ for all $j$, and let $\bar{B}_1 = B_1$.

Step 1: Solve the inner minimization, Minimize $L_{u,v}(\mathbf{x}, \mathbf{y})$, and let $\mathbf{x}_0$ be the solution. Determine the quantities $F_j^p$ and $G_j$ for this solution and, hence, the slacks $\mathbf{s}_j^1$ and $\mathbf{s}_j^2$ for constraint sets (S1) and (S2), respectively.

Step 2: If all slacks for $j \in \bar{B}_1 \cup B_2$ are nonnegative, STOP. Otherwise, determine the constraint $t$ that has the most negative slack and the corresponding link $j^* \in \bar{B}_1 \cup B_2$.

Step 3: If $t$ is in constraint set (S1), let $u_{j^*} \leftarrow u_{j^*} + \Delta$; and if $t$ is in constraint set (S2), let $v_{j^*} \leftarrow v_{j^*} + \Delta$. If $j^* \in \bar{B}_1$, make sure that $\Delta \leq (K_j/T) - (u_{j^*} + v_{j^*})$. Remove $j^*$ from $\bar{B}_1$ if $\Delta = K_j/T - (u_{j^*} + v_{j^*})$. Go to Step 1.

on $y$ variables has been dropped, it leads to the following problem:

$$\underset{u,v}{\text{Maximize}} \quad \left[\underset{x,y}{\text{Minimize}}\ L(\mathbf{x}, \mathbf{y}, \mathbf{u}, \mathbf{v})\right] \qquad \text{(PL)}$$

subject to:

$$\sum_{k \in S_l} x_l^k = 1 \quad \forall\ l \in C$$

$$x_l^k = 0 \text{ or } 1 \quad \forall\ l, k$$

$$y_j \geq Y_j \qquad j \in B_1$$

$$y_j = Y_j \qquad j \in B_2$$

Here, $L(\mathbf{x},\mathbf{y},\mathbf{u},\mathbf{v})$ is the same as $L(\mathbf{x},\mathbf{u},\mathbf{v})$ of problem (SL), except that $Y_j$ has been replaced with $y_j$ for all $j$. Note that, if all $j$ are in $B_2$, this problem reduces to problem (SL), the relaxation of the routing subproblem, because the entire backbone topology is specified. The type-1 constraint ($y_j \geq Y_j$) can be interpreted to mean that link $j$ should have at least $Y_j$ high-capacity facilities. In other words, the flow may exceed the capacity $TY_j$ for links in set $B_1$ but not for links in set $B_2$. Notice that, at the root node of the branch-and-bound tree, all links belong to set $B_1$ with $Y_j = 0$ for all $j$.

Note that the coefficient of $y_j$ in $L_{u,v}(\mathbf{x},\mathbf{y})$ is $[K_j - (u_j + v_j)\ T]$. If $(u_j + v_j) < K_j/T$, then $y_j = Y_j$ minimizes $L_{u,v}(\mathbf{x},\mathbf{y})$. If $(u_j + v_j) = K_j/T$, then the coefficient of $y_j$ becomes zero and any value of $y_j$ will be optimum and can be chosen so as to make the routing solution feasible. However, if $(u_j + v_j) > K_j/T$, then $L_{u,v}(\mathbf{x},\mathbf{y})$ decreases unboundedly by increasing $y_j$. Because we wish to maximize $L(\mathbf{x},\mathbf{y},\mathbf{u},\mathbf{v})$ with respect to $\mathbf{u}$ and $\mathbf{v}$, we can conclude that the optimal solution of (PL) must satisfy $(u_j + v_j) \leq K_j/T$ for $j \in B_1$. Note that $K_j/T$ is the prorated cost per circuit for a high-capacity facility. Therefore, this condition makes sense intuitively because additional flow pays for the capacity used in a prorated fashion. Thus, a lower bound can be obtained by solving a modified version of the routing subproblem in which the flow is permitted to exceed $TY_j$ on links $j \in B_1$ if $(u_j + v_j) = K_j/T$. On such links, if the flow exceeds $TY_j$, then additional units of flow incur a cost $K_j/T$, which implies a lower bound because it accounts for the prorated cost of the used capacity of a high-capacity facility.

The routing subproblem algorithm presented earlier can be easily adapted for this change as described below. Let us define another set:

$$\bar{B}_1 = \left[j : j \in B_1,\ (u_j + v_j) < \frac{K_j}{T}\right]$$

Then, the subproblem algorithm can be adapted as

**Algorithm 4. The Master Problem**

Step 0: Let $Y_j = 0$ and $j \in B_1$ for all $j$; $B_2 = \phi$; and $Z = \infty$.

Step 1: Solve the routing subproblem using Algorithm 3, and determine the lower bound $L$, upper bound $U$, and values of variables $y_j$. If $U < Z$, then let $Z = U$ and save the incumbent solution.

Step 2: If $y_j \leq Y_j$ for all $j$, STOP (or backtrack to an unfathomed previous node, subject to the computer time limit); otherwise, for all $j \in B_1$, if $\lfloor y_j \rfloor > Y_j$, let $Y_j = \lfloor y_j \rfloor$.

Step 3: Select a branch variable $j^*$ and branch direction according to the criteria described in the branching strategy section. In the case of an up-branch, let $Y_{j^*} \leftarrow Y_{j^*} + 1$; in the case of a down-branch, move $j^*$ from set $B_1$ to set $B_2$. Go to Step 1.

shown in Algorithm 3 to produce a lower bound. In Step 3, we make sure that $u_j + v_j$ does not exceed $K_j/T$ for links in set $B_1$. Also note that in Step 2 the nonnegativity of slacks is checked only for variables in the set $\bar{B}_1 \cup B_2$. Because the procedure for minimizing $L_{u,v}(\mathbf{x},\mathbf{y})$ is a heuristic, the solution value obtained may not be a true lower bound. Nevertheless, if the solution is reasonably close to optimal, it can still be used effectively for selection of the branch variables and branching direction.

**Branching Strategy.** As indicated earlier, imposing the type-1 or type-2 constraints on the variables $y_j$ constitutes the branching process in our approach. At the root node, each $y_j$ has a type-1 constraint $y_j \geq 0$ imposed on it (i.e., $j \in B_1$, and $Y_j = 0$), which really amounts to no constraint at all. Solving the subproblem will yield a certain value for $y_j$, which, in all likelihood, will be fractional. Then the up-branch corresponds to setting $Y_j = \lceil y_j \rceil$ (i.e., $y_j$ rounded up to the nearest integer value) and $j$ stays in set $B_1$. On the other hand, a down-branch will correspond to setting $Y_j = \lfloor y_j \rfloor$ (i.e., $y_j$ rounded down to the nearest integer value) and moving $j$ from $B_1$ to $B_2$.

In branch-and-bound algorithms, a commonly used criterion for selection of the branch variable is to pick the variable whose value is closest to an integer value. We use this criterion for the selection of the branch variable and first examine the obvious branch direction (i.e., the up-branch if the fractional value is closer to one, and the down-branch if it is closer to zero). If the resulting increase in the lower bound is modest, we ignore the other branch and continue. However, if the increase is significant, the other branch is also examined and the branch with the smaller value of lower bound is selected. If the difference between the lower bounds for the two branches is very small, then the information corresponding to one branch is saved so that it may be used as a starting point for possible future backtracking.

At each node of the branch-and-bound tree, a feasible solution and, hence, an upper bound for the master problem is immediately available by upward rounding of variables $y$.

**Algorithm for the Master Problem.** Based on these observations, Algorithm 4 is the algorithm for the master problem.

### A More Effective Lower Bound at the Root Node

The solution of the routing subproblem at the root node of the branch-and-bound tree yields a lower bound on the master problem. However, this lower bound is not very tight, particularly when the direct-route analysis is not desired, because this bound completely ignores the integrality constraint on variables $y_j$. In this section, we develop a more effective lower bound, which is partly able to take into account the integrality of $y_j$. This bound is applicable for the case when direct-route analysis is not desired.

Consider a cut (indexed by $t$) of the network that partitions the node set into two subsets $N_t$ and $\bar{N}_t$. Let $A_t$ be the set of links on the cut. All circuits that have one

**Table I. Seven-Node Problems**

| Problem number | Circuits/node-pair | | Normalized solution | Percent above lower bound |
|---|---|---|---|---|
| | Low priority | High priority | | |
| 1 | 3 | 0 | 100 | 27 |
| 2 | 2 | 1 | 109 | 1 |
| 3 | 1 | 2 | 109 | 1 |
| 4 | 0 | 3 | 142 | 14 |
| 5 | 4 | 0 | 100 | 30 |
| 6 | 3 | 1 | 109 | 1 |
| 7 | 2 | 2 | 109 | 1 |
| 8 | 1 | 3 | 142 | 14 |
| 9 | 0 | 4 | 145 | 19 |
| 10 | 5 | 0 | 100 | 17 |
| 11 | 4 | 1 | 100 | 17 |
| 12 | 3 | 2 | 100 | 17 |
| 13 | 2 | 3 | 100 | 17 |
| 14 | 1 | 4 | 104 | 19 |
| 15 | 0 | 5 | 138 | 9 |

end-point in set $N_t$ and the other in $\bar{N}_t$ must flow across this cut. Obviously, the total capacity available on the links in this cut must exceed this flow. If the total flow required across the cut is $f_t$, there must be at least $\lceil f_t/T \rceil$ high-capacity facilities on this cut. The number of facilities on the cut can be expressed as a function of variables $y_j$, and is given by the expression $\sum_{j \in A_t} y_j$. Therefore, any feasible solution to the network design problem must satisfy the following inequality across every cut $t$:

$$\sum_{j \in A_t} y_j \geq \left\lceil \frac{f_t}{T} \right\rceil$$

Let us decompose $f_t$, the flow across cut $t$, into three components corresponding to the three priority levels. Let these components be $f_t^0$, $f_t^1$, and $f_t^2$, respectively. Now consider the case of a single-link failure across the cut. All high-priority circuits must still be able to get across the cut. In addition, any normal-priority circuits that were not using the failed link will continue to use the capacity of the cut, because they cannot be preempted. Because the capacity of a facility is $T$, the number of such unaffected normal-priority circuits must be at least $\max(f_t^1 - T, 0)$. This implies the following constraint:

$$\sum_{j \in A_t} y_j - 1 \geq \left\lceil \frac{f_t^2 + \max(f_t^1 - T, 0)}{T} \right\rceil$$

Because each feasible integer solution must satisfy these two constraints for each cut, it is easy to see that the solution of the following linear program is a lower bound on the cost of the optimal network.

$$\text{Minimize} \quad \sum_{j \in A} K_j y_j$$

$$\text{subject to:} \quad \sum_{j \in A_t} y_j \geq \left\lceil \frac{f_t}{T} \right\rceil \quad \forall\ t$$

$$\sum_{j \in A_t} y_j - 1 \geq \left\lceil \frac{f_t^2 + \max(f_t^1 - T, 0)}{T} \right\rceil \quad \forall\ t$$

$$y_j \geq 0$$

Although there are many possible cuts, it is not necessary to include them all in this problem. Our experience shows that even a small subset of appropriately selected cuts is enough to produce a good lower bound.

Note that the problem solved to obtain this bound does not involve the flow variables. As a result, even if an integrality condition is imposed on the variables $y_j$, the optimal solution to this problem may not be feasible for the original problem. We use this bound only to assess the quality of a final solution, and not for any branching decisions. The effectiveness of this lower bound will be apparent from the computational results given in the next section.

**Table II. Ten-Node Problems**

| Problem number | Circuits/node-pair | | Normalized solution | Percent above lower bound |
|---|---|---|---|---|
| | Low priority | High priority | | |
| 1 | 2 | 0 | 100 | 19 |
| 2 | 1 | 1 | 114 | 11 |
| 3 | 0 | 2 | 149 | 15 |
| 4 | 3 | 0 | 100 | 13 |
| 5 | 2 | 1 | 102 | 16 |
| 6 | 1 | 2 | 113 | 15 |
| 7 | 0 | 3 | 151 | 21 |
| 8 | 4 | 0 | 100 | 16 |
| 9 | 3 | 1 | 100 | 16 |
| 10 | 2 | 2 | 100 | 16 |
| 11 | 1 | 3 | 117 | 17 |
| 12 | 0 | 4 | 131 | 14 |
| 13 | 5 | 0 | 100 | 16 |
| 14 | 4 | 1 | 100 | 16 |
| 15 | 3 | 2 | 100 | 16 |
| 16 | 2 | 3 | 106 | 12 |
| 17 | 1 | 4 | 108 | 14 |
| 18 | 0 | 5 | 127 | 17 |

**Computational Results**

The algorithm was tested on two sets of problems, one with seven nodes and the other with ten nodes. These are the typical sizes of high-capacity backbones for most private corporate networks. Within each set, several problem instances were created by varying the number of circuits between each node-pair and the mix of high- and low-priority circuits. Within each instance, the number of circuits and the mix of high- and low-priority circuits is the same for each node-pair. Thus, for a given a node set, a problem instance is defined by two numbers: the number of high-priority circuits for each node-pair, and the number of low-priority circuits.

The algorithm was used to design networks without direct routes. The capacity of the high-capacity facilities was assumed to be 24 circuits. The results of computational testing for the seven-node problems are listed in Table I and for the ten-node problems in Table II. In both tables, columns 2 and 3 list the number of low- and high-priority circuits for each node-pair in the problem. Column 4 shows the normalized cost of the solution obtained, where, within each problem set, the cost for the network with all low-priority circuits is normalized to 100. Column 5 shows the percent difference between the solution and the lower bound described in the preceding section.

Note that the normalized solution value also reflects the premium for survivability. The premium for survivability is defined as the percentage by which the cost of a network exceeds the cost of the equivalent base network in which all circuits are of low priority. As expected, the premium increases as the ratio of high-priority circuits is increased in each subset of problems. It is interesting to note that the premium for full survivability (i.e., when all circuits are of high priority) varies from 27 percent to 51 percent. This variability is to be expected because of the high capacity of the facilities. In some cases, the base network (i.e., when all circuits are of low priority) may contain a large amount of spare capacity and, in some cases, very little. It is intuitively obvious that the premium for survivability will be low in the former case and high in the latter case.

For the seven-node problems, the gap between the lower bound and the solution varies from 1 percent to 30 percent, with an average gap of 12 percent. For the ten-node problems, the gap varies from 11 percent to 21 percent with an average of 15 percent. The computer time taken to solve the problems varied from 2 seconds to 27 seconds for the first set, and from 15 to 300 seconds for the second set. The computer time increases rapidly with the number of high-priority circuits and the number of nodes in the problem.

The computational results should be viewed in light of the complexity of this problem. In a problem with $n$ nodes, the number of topology variables is $n(n-1)/2$. The number of routing variables for every node-pair is equal to the total number of paths between the two nodes, which grows as a factorial function of the number of nodes in the problem. In a ten-node problem, the number of paths with five or fewer links between any pair of nodes could be as high as 162, thus leading to a total of 7290 routing variables if there was only one circuit per node-pair. The consideration of link-disjoint path-pairs for high-priority circuits further adds to the complexity of the problem. That is why the computation time rises steeply as the number of high-priority circuits in the problem is increased.

**Conclusion and Future Work**

We have presented an algorithmic framework for designing high-capacity networks and addressing the survivability issue. Although the results appear to be satisfactory, it should be possible to improve the running time as well as the solution quality of the algorithm. The shortest-path or path-pair problem for high-priority circuits is a major computational bottleneck and deserves further improvement. The LP-based lower bound was used only to assess the quality of the solution. It may be possible to use the topology information contained in its solution to further improve the performance of the algorithm.

The algorithms described in this paper are implemented in AT&T's E-INOS (Enhanced Interactive Network Optimization System). AT&T uses E-INOS to design voice, data, and integrated networks for AT&T customers.

**References**


1. M. Minoux, "Multiflots de Cout Minimal Avec Fonctions de Cout Concaves," *Annales des Telecommunication*, Vol. 31, Nos. 3-4, March/April 1976, pp. 77-92.
2. S. Narasimhan, H. Pirkul, and P. De, "Route Selection in Backbone Data Communications Networks," *Computer Networks and ISDN Systems*, Vol. 15, No. 2, 1988, pp. 121-133.
3. B. Gavish and S. L. Hantler, "An Algorithm for Optimal Route Selection in SNA Networks," *IEEE Transactions on Communications*, Vol. 31, No. 10, October 1983, pp. 1154-1161.
4. G. B. Dantzig et al., "Formulating and Solving the Network Design Problem by Decomposition," *Transportation Research*, Vol. 13-B, 1979, pp. 5-17.
5. T. B. Boffey and A.I. Hinxman, "Solving the Optimal Network Problem," *European Journal of Operational Research*, Vol. 3, No. 5, September 1979, pp. 386-393.
6. D. E. Boyce, A. Farhi, and R. Weischdel, "Optimal Network Problem: A Branch-and-Bound Algorithm," *Environment and Planning*, Vol. 5, 1973, pp. 519-533.
7. K. Steiglitz, P. Weiner, and D. J. Kleitman, "The Design of Minimum-Cost Survivable Networks," *IEEE Transaction on Circuit Theory*, Vol. 16, No. 4, November 1969, pp. 455-460.
8. H. Pirkul and S. Narasimhan, "Selecting Primary and Secondary Routes in Computer Networks," presented at ORSA/TIMS, Operations Research Society of America/The Institute of Management Science, New Orleans, 1987.
9. Y. Kajitani and S. Ueno, "The Minimum Augmentation of a Directed Tree to a $k$-Edge-Connected Directed Graph," *Networks*, Vol. 16, No. 2, Summer 1986, pp. 181-197.
10. J. W. Suurballe, "Disjoint Paths in a Network," *Networks*, Vol. 4, No. 2, 1974, pp. 125-145.
11. J. W. Suurballe and R. E. Tarjan, "A Quick Method for Finding Shortest Pairs of Disjoint Paths," *Networks*, Vol. 14, No. 2, Summer 1984, pp. 325-336.
12. C.-L. Li, S.T. McCormick, and D. Simchi-Levi, "The Complexity of Finding Two Disjoint Paths with Min-Max Objective Function," presented at ORSA/TIMS, Operations Research Society of America/The Institute of Management Science, Washington, D.C., 1988.
13. L. S. Lasdon, *Optimization Theory for Large Systems*, Macmillan Publishing Co., New York, 1970, Chapter 8, p. 402.
14. E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, Vol. 1, No. 3, September 1959, pp. 269-271.


*(Manuscript received January 11, 1989)*