Technical Report 01-EMIS-04

An Incremental Procedure for Improving Path Assignment in a Telecommunications Network

by

D. Allen

I. Ismail

J. Kennington

and

E. Olinick

Department of Engineering Management, Information, and Systems

School of Engineering

Southern Methodist University

Dallas, TX 75275-0123

Revised September 5, 2003

Abstract

A fundamental problem in the design and management of a telecommunications network is that of determining an optimal routing pattern for a given set of origin-destination demand pairs. In addition, reliability considerations may require provisioning a set of backup paths to protect the working traffic against network failures. In the literature, the problem of finding an optimal routing for a network with fixed link capacities and a list of point-to-point demands (origin-destination pairs), each with a set of candidate routing paths has been referred to as the *path-assignment problem*. There are three versions of this problem that correspond to the type of network protection required (no protection, dedicated protection, and shared protection). The solution to those models can be used to determine an initial design for a new network. Over time, however, changes in the demand pattern and/or upgrades to the network equipment may create a situation in which the working and/or backup paths are sub-optimal.

For network managers who are reluctant to make wholesale changes to an established and reliable routing assignment, a complete modification to obtain an optimal assignment that uses fewer network resources is viewed as highly risky. This investigation presents a new procedure to take a given feasible, but sub-optimal design and improve it by making a series of incremental improvements each of which only changes a small number of path assignments. Network managers view this strategy as much less risky since only a few customers are affected by any one change. Test cases that require no protection, dedicated protection, and shared protection were examined in an empirical analysis. For all cases, near-optimal solutions were achieved irrespective of the quality of the given sub-optimal starting solutions.

key words: path-assignment problem, spare-capacity planning, greedy heuristic, dedicated protection

Acknowledgment

This work was partially supported by the Office of Naval Research under Award No. N00014-96-1-0315 and by the Linda and Mitch Hart eCenter at SMU http://www2.smu.edu/ecenter/.

1 Introduction

This investigation addresses an important, and difficult, combinatorial optimization problem in the design and management of telecommunication systems. In the *path-assignment problem*, one is given a network with fixed link capacities and a list of point-to-point demands (origin-destination pairs), each with a set of candidate routing paths. A path assignment is said to be *feasible* if it meets the following two criteria. First, all traffic for a given origin-destination pair must be routed along (assigned to) exactly one of the paths available for that pair. Second, the total volume of traffic routed over any particular link must be within the given capacity limit. There is a cost associated with using each path and the problem is to find an optimal (i.e., minimum-cost) path assignment. The path-assignment problem without protection is $\mathcal{NP}$-hard. There are versions of this problem that require the determination of both the working paths and backup paths. One version assumes dedicated backup paths for each working path and another allows for multiple backup paths to share spare capacity.

The existing literature [1, 3, 13, 16] treats this problem from the perspective of determining the initial routing assignments for a new network. The implementation of a routing plan may occur over a period of several months. As a result, the actual demands for services may differ from those used in the design of the plan. Also, during implementation, situations may arise that result in circuit routings that differ from those specified in the plan. For example, services may be required prior to provisioning of the planned network expansion. Furthermore, after restoration due to a failure in the network, the carrier may not revert to the original optimal routing, thus, leaving the network routing in a sub-optimal state. Over time, changes in the network may result in an inefficient use of the current resources. The optimization techniques developed to find the initial routing assignment and backup paths could also be used to find an optimal design for the new network conditions. However, the optimal assignment by itself does not provide the network manager with enough information to re-optimize the system in a way that minimizes the impact on the customers.

In addition to cost considerations, network managers are also concerned with the quality of service provided by their networks. Understandably, they are reluctant to make major changes to an operating network that may inadvertently result in service disruptions for numerous

clients simultaneously. This investigation presents a strategy and solution procedures for the various path-assignment problems in this context.

The objective of this investigation is to develop and test solution procedures that produce a series of route-improving reassignments of working and/or backup paths for the point-to-point demands. This allows the network manager to implement the improved routing in stages with minimal changes to the overall routing plan between any two consecutive stages. Although the procedure may only reroute a few demands at each stage, the routing after the last stage has been implemented should be as close as possible to an optimal routing.

We claim three contributions from this investigation. First, we formally define a problem of interest to telecommunication network managers that has not, to our knowledge, been addressed in the literature. This problem may be defined as follows: *Given a feasible solution to the path-assignment problem, determine a sequence of incremental assignment modifications that lead to an optimal assignment.* There are various versions of the path-assignment problem based upon the type of protection required. A solution to this problem provides a network manager with a low-risk strategy for improving network utilization. Second, we implemented our algorithms in software and in an empirical analysis demonstrated the efficacy of our procedures for test cases involving various levels of protection. Finally, we make our software available on the World Wide Web for downloading free of charge for immediate use by network management groups who wish to experiment with our procedures.

2 Routing on Mesh Networks

In this section we present models and algorithms for the incremental path assignment problem for mesh networks. We consider three different versions of the problem depending on the *protection scheme* used to protect the network against service disruptions caused by link failures. In Section 3 we extend the first model and algorithm to systems of SONET rings.

2.1 Without Protection

We now present an integer programming formulation for the simplest version of the path-assignment problem. For this model, there are no backup paths to protect the working paths. Let E and N denote the set of links and nodes in the network and D be the set of all point-

to-point demands. Since our model does not allow for *demand splitting*, all of the traffic for a particular demand $d \in D$ must be routed over exactly one of a set of available routing paths, J_d. However, demand splitting can be accommodated by replacing a demand for r circuits by r individual demands. Let $P = \cup_{d \in D} J_d$ be the set of all available paths and $P_e \subseteq P$ denote the set of paths traversing link $e \in E$. Let r_d denote the number of demand units required, e.g. OC-48's, for demand $d \in D$. The cost of using a path $p \in P$ and the capacity of link $e \in E$ are denoted as a_p and c_e, respectively. Let the binary decision variable $x_p = 1$ if path p is used to satisfy some demand; and 0, otherwise.

Using the above notation, the path-assignment problem without protection is formulated as the following integer linear program (ILP):

$$\min \quad \sum_{p \in P} a_p x_p \tag{1}$$

$$\text{s.t.} \quad \sum_{p \in J_d} x_p = 1, \qquad \forall d \in D \tag{2}$$

$$\sum_{d \in D} \sum_{p \in J_d \cap P_e} r_d x_p \leq c_e, \qquad \forall e \in E \tag{3}$$

$$x_p \in \{0, 1\}, \qquad \forall p \in P \tag{4}$$

Thus, the path-assignment problem without protection is to assign each demand d to exactly one of its candidate routing paths in J_d in accordance with the link capacities so as to minimize the total routing cost. Constraint sets (2) and (4) ensure that each demand is assigned to exactly one of its candidate paths and the link capacities are enforced by constraint set (3).

Given an instance of the path-assignment problem, a $|P|$-component vector $\hat{x}^0$ satisfying constraint sets (2)-(4), and an integer k, we define an *incremental path-assignment sequence* as a sequence of feasible solutions $\hat{X} = \{\hat{x}^1, \hat{x}^2, \ldots, \hat{x}^\ell\}$ such that

$$\sum_{p} |\hat{x}_p^i - \hat{x}_p^{i-1}| \leq 2k, \qquad \forall i = 1, \ldots \ell, \text{ and} \tag{5}$$

$$\sum_{p \in P} a_p \hat{x}_p^i < \sum_{p \in P} a_p \hat{x}_p^{i-1}, \qquad \forall i = 1, \ldots, \ell \tag{6}$$

The parameter k in (5) indicates the number of paths that may be rerouted at any time. Observe that rerouting a single demand d from path $p \in J_d$ to path $q \in J_d$ corresponds to

changing two entries in the solution vector: $\hat{x}_p \leftarrow 0$ and $\hat{x}_q \leftarrow 1$. The inequality (6) ensures a cost improvement in each successive solution. This procedure for finding $\hat{X}$ is similar to the *k-interchange* heuristic for the traveling salesman problem [15].

Given a feasible solution $\hat{x}^i$ to the path-assignment problem, the basic step of our procedure (using $k = 2$) is to find a least-cost way of routing a particular pair of demands d_1 and d_2 while keeping the other demands routed according to the current solution (i.e., $\hat{x}^i$). Let p_1 and p_2 be the paths assigned to demands d_1 and d_2 by $\hat{x}^i$, respectively. That is, $\hat{x}^i_{p_1} = \hat{x}^i_{p_2} = 1$, $p_1 \in J_{d_1}$ and $p_2 \in J_{d_2}$. A *trial* consists of choosing a pair of paths $q_1 \in J_{d_1}$ and $q_2 \in J_{d_2}$ such that $a_{q_1} + a_{q_2} < a_{p_1} + a_{p_2}$ and constructing the *candidate solution* $\tilde{x}$ from $\hat{x}^i$ by letting $\tilde{x}_j = \hat{x}^i_j$ $\forall j \in P \setminus \{p_1, p_2, q_1, q_2\}$, $\tilde{x}_{p_1} = \tilde{x}_{p_2} = 0$, and $\tilde{x}_{q_1} = \tilde{x}_{q_2} = 1$. If $\tilde{x}$ is feasible, then the trial is said to be *successful* and the procedure reports the reassignment of demand d_1 to path q_1 and demand d_2 to path q_2 as the next incremental demand rerouting and adds the next solution in the incremental sequence, $\hat{x}^{i+1} \leftarrow \tilde{x}^i$.

In the worst case, this process requires $|J_{d_1}| \times |J_{d_2}| - 1$ trials. In an attempt to minimize the number of trials required per pair, the procedure inspects pairs q_1 and q_2 in non-decreasing order of total cost $(a_{q_1} + a_{q_2})$ until a successful trial is made or until the list of candidate solutions is exhausted. As shown in Figure 1, this process is repeated for each pair of demands $d_1, d_2 \in D$ until no more successful trials are possible.

Figure 1 About Here.

2.2 With Protection

Network reliability is a critical issue and working paths are frequently paired with a backup path that is used when failure occurs. While failure may occur in links, nodes, or individual channels in the case of WDM networks, our investigation only considers the most common type, link failures. Furthermore, we assume that the probability of multiple link failures is very small and provide spare capacity only for single link failures. Excellent discussions of network protection can be found in [4, 5, 17].

Protection schemes come in two varieties, *dedicated protection* and *shared protection*. In a shared-protection scheme, the spare capacity allocated to backup paths can be shared. If two

backup paths share a common link e, then there may be an opportunity to share the spare capacity allocated to link e. If the backup paths require u_1 and u_2 units of spare capacity, then a dedicated scheme provisions $u_1 + u_2$ units of spare capacity while a shared scheme may only require $\max(u_1, u_2)$. If two backup paths are never needed at the same time, then $\max(u_1, u_2)$ will suffice. Investigations giving models and algorithms for designing fault-tolerant networks can be found in [2, 6, 7, 8, 9, 10, 11, 12, 14, 18, 19]

2.2.1 Dedicated Protection

For the case of dedicated protection, every working path is paired with a link-disjoint backup path. These two paths taken together form a cycle and the objective is to select a set of least-cost cycles that satisfies the link-capacity constraints. Hence, the dedicated protection model is (1)-(4) with cycles (a working path plus a backup path) used in place of working paths. With this modification, the incremental path assignment algorithm (Figure 1) applies to this case. In the literature dedicated protection is referred to as 1+1 protection (see [17]) and we use these terms interchangeably.

2.2.2 Shared Protection

The difference between dedicated protection and shared protection is illustrated in the example network given in Figure 2. The working and backup paths for the two demands are illustrated in Figures 2b and 2c. Note that the backup paths both use links $(4, 5)$ and $(5, 6)$. However, as seen from Table 1, it can be observed that the backup paths are never in use at the same time provided there is only a single link failure. Hence, the capacity required is only $\max(4, 6) = 6$ rather than the $4 + 6 = 10$ required for dedicated protection.

Figure 2 and Table 1 About Here.

2.2.3 ILP for Path Assignment with Shared Protection

We now extend the ILP model from Section 2.1 to include shared protection. To help clarify the new model, we will refer to the network topology, demand pairs, and paths from Figure 2 as an example throughout this discussion. Let the working and backup paths in Figure 2b be

paths 1 and 2, respectively. Likewise let the working and backup paths in Figure 2c be paths 3 and 4.

In the model for shared protection, $x_p = 1$ if path p is used to carry working traffic; and zero, otherwise. Let w_e be the volume of working traffic routed on link e. Thus, in Figure 2 we have $x_1 = x_3 = 1$, $x_2 = x_4 = 0$, $w_{(1,2)} = w_{(2,6)} = 4$, $w_{(2,4)} = w_{(2,3)} = 6$, and $w_e = 0$ for all other links. Let $y_p = 1$ if path p is used as a backup path; and 0, otherwise. For example $y_1 = y_3 = 0$ and $y_2 = y_4 = 1$ in Figure 2. Let H be the set of all pairs of paths $\{p, q\}$ such that p and q share at least one link in common. If $p \in J_d$ is the working path for some demand $d \in D$, then $q \in J_d$ may be selected as the backup path for d only if $\{p, q\} \notin H$.

We refer to the scenario of link e failing as *scenario* e. The binary variable z_{pe} indicates whether or not path p is used as a backup path in scenario e. For any given demand d exactly one path $p \in J_d$ will be used in scenario e. Suppose that $x_p = y_q = 1$ for $p, q \in J_d$; that is, p is the working path and q is the backup path. If $p \in P_e$, then backup path q must be used when link e fails which means that $z_{qe} = 1$ and $z_{ie} = 0$ for all $i \in J_d \setminus \{q\}$. However, if $p \notin P_e$, then the working path p can (and will) be used when link e fails. In this case, $z_{ie} = 0$ for all $i \in J_d$. As an example, consider the scenario where link $(1, 2)$ fails in Figure 2. From Table 1, we see that in scenario $(1, 2)$ traffic between nodes 1 and 6 is switched from path 1 to path 2. Thus $z_{1,(1,2)} = 0$ and $z_{2,(1,2)} = 1$. In scenario $(2, 3)$, however, the working path for 1-6 does not contain the failed link. Thus, no backup path is used for this demand in this scenario; and so, $z_{1,(2,3)} = z_{2,(2,3)} = 0$.

Using the notation above, our ILP model for the path-assignment problem with shared protection is stated as follows. First, the objective is to minimize the total cost of the working and backup paths. Thus, the objective function is given by

$$\sum_{p \in P} a_p(x_p + y_p) \tag{7}$$

For each demand, we must select a pair of link-disjoint working and backup paths. Thus, we have the following set of constraints:

$$\sum_{p \in J_d} x_p = 1, \qquad \forall d \in D \tag{8}$$

$$\sum_{p \in J_d} y_p = 1, \qquad \forall d \in D \tag{9}$$

$$x_p + y_q \leq 1, \qquad \forall d \in D, p \in J_d, q \in J_d, \{p, q\} \in H \tag{10}$$

The following set of constraints accumulate the working traffic on the links:

$$\sum_{d \in D} \sum_{p \in J_d \cap P_e} r_d x_p = w_e, \qquad \forall e \in E \tag{11}$$

We must ensure that each link $e \in E$ has sufficient capacity to carry its working traffic plus the additional traffic that would be diverted to backup paths that use e if any link $f \in E \setminus \{e\}$ fails. Thus, we impose the following set of constraints:

$$\sum_{d \in D} \sum_{p \in J_d \cap P_e} r_d z_{pf} + w_e \leq c_e, \qquad \forall e \in E, f \in E \setminus \{e\} \tag{12}$$

For a given failure scenario f, path p cannot be used as a backup path if it contains link f. So, we can use the following constraints to set some of the z variables to zero:

$$z_{pf} = 0, \qquad \forall f \in E, p \in P_f \tag{13}$$

The following constraints ensure a consistent relationship between the y and z variables:

$$z_{pf} \leq y_p, \qquad \forall f \in E, p \in P \setminus P_f \tag{14}$$

The following set of constraints ensure the correct relationship between the x and z variables. Suppose that link $f \in E$ fails and consider a demand $d \in D$; either the working path for d is used (if it does not contain f) or else an appropriate backup path is used. This requirement is expressed mathematically as

$$\sum_{p \in J_d \setminus P_f} (x_p + z_{pf}) = 1, \qquad \forall f \in E, d \in D \tag{15}$$

Finally, we impose integrality and non-negativity constraints on the decision variables as follows:

$$x_p, y_p \in \{0, 1\}, \qquad \forall p \in P, \tag{16}$$

$$z_{pe} \in \{0, 1\}, \qquad \forall p \in P, e \in E, \tag{17}$$

$$w_e \geq 0, \qquad \forall e \in E \tag{18}$$

Thus, the ILP for the model with shared protection is to minimize (7), subject to (8)-(18).

2.2.4 Incremental Path-Assignment with Shared Protection when $k = 2$

The procedure shown in Figure 1 can be modified to work with the shared-protection model. Given a feasible solution $[\hat{x}^i, \hat{y}^i, \hat{z}^i]$ to the path-assignment problem with shared protection, the basic step of our modified procedure is to find a least-cost way of routing and protecting a particular pair of demands d_1 and d_2 while keeping the rest of the demands routed according to the current solution.

Let p_1 and p_2 be the working paths assigned to demands d_1 and d_2 by $\hat{x}^i$. Let q_1 and q_2 be the corresponding protection paths. That is, $\hat{x}^i_{p_1} = \hat{x}^i_{p_2} = 1$, $p_1 \in J_{d_1}$, $p_2 \in J_{d_2}$, and $\hat{y}^i_{q_1} = \hat{y}^i_{q_2} = 1$, $q_1 \in J_{d_1}$ and $q_2 \in J_{d_2}$. With shared protection, a trial consists of choosing a pair of working paths $p_3 \in J_{d_1}$ and $p_4 \in J_{d_2}$, and a pair of protection paths $q_3 \in J_{d_1}$ and $q_4 \in J_{d_2}$ such that $a_{p_3} + a_{q_3} + a_{p_4} + a_{q_4} < a_{p_1} + a_{q_1} + a_{p_2} + a_{q_2}$. We then construct the candidate solution $[\tilde{x}, \tilde{y}]$ where $\tilde{x}_j = \hat{x}^i_j \; \forall j \in P \setminus \{p_1, p_2, p_3, p_4\}$, $\tilde{x}_{p_1} = \tilde{x}_{p_2} = 0$, $\tilde{x}_{p_3} = \tilde{x}_{p_4} = 1$, and $\tilde{y}_j = \hat{y}^i_j$ $\forall j \in P \setminus \{q_1, q_2, q_3, q_4\}$, $\tilde{y}_{q_1} = \tilde{y}_{q_2} = 0$, and $\tilde{y}_{q_3} = \tilde{y}_{q_4} = 1$. Given $\tilde{x}$ and $\tilde{y}$, the $\tilde{z}$ vector can be derived in a straight-forward manner. If $[\tilde{x}, \tilde{y}, \tilde{z}]$ is feasible, then the trial is successful and the procedure reports the reassignment of the working and protection paths for demands d_1 and d_2 as the next incremental demand rerouting. It then inserts this solution as the next solution in the incremental sequence, $\hat{x}^{i+1} \leftarrow \tilde{x}^i$, $\hat{y}^{i+1} \leftarrow \tilde{y}^i$, and $\hat{z}^{i+1} \leftarrow \tilde{z}^i$.

We now introduce some additional notation so that we may describe the procedure in more detail. Let E_p denote the set of links in path p. For each $d \in D$ and $p \in J_d$, let $rp_p = r_d$. For a given routing assignment $[x, y, z]$, let b_{ef} denote the additional units of demand that must be routed over link e in the event that link f fails. In the example illustrated by Figure 2 and Table 1, $b_{(4,5),(1,2)} = 4$ and $b_{(4,5),(2,3)} = 6$. Observe that under a shared protection scheme the spare capacity requirement for a given link e is the maximum value of b_{ef} over all failure scenarios f. Given a set of working and backup paths, we define the *residual capacity* of link e as $rc_e = c_e - w_e - \max_{f \in E} b_{ef}$. In the following subsection, we describe how our procedure uses information about the residual capacities for the current routing assignment to reduce the number of trials it must consider for a given pair of demands and to determine if a given solution is feasible.

Figure 3 gives pseudo code for the main routine of our procedure for finding an incremental path-assignment sequence with shared protection. The procedure begins by calling the subroutine shown in Figure 4 to calculate E_p for all paths p and to calculate the initial values of the b matrix and rc vector. Thus, it determines the residual capacity for each edge given the initial set of working and backup paths.

Figures 3 and 4 About Here.

The first step in generating a set of trials for a given pair of demands d_1 and d_2 is to *unroute* the demands with the subroutine in Figure 5. Unrouting a pair of demands consist of reducing the volume of working traffic on each link in p_1 (p_2) by r_{d_1} (r_{d_2}). That is, we temporarily remove the traffic for d_1 and d_2. This reduces the working traffic on the edges in p_1 and p_2, and also reduces the spare capacity requirements on the links in the backup paths q_1 and q_2 under certain failure scenarios. Specifically, we can reduce b_{ef} for each $e \in q_1$ (q_2) and $f \in p_1$ (p_2) by rp_{p_1} (rp_{p_2}). Finally, we recalculate the residual capacities of the links in p_1, q_1, p_2, and q_2.

Figure 5 About Here.

As shown in Figure 6, the procedure uses the updated residual capacities to select certain combinations of new working and backup paths for rerouting d_1 and d_2. Consider link-disjoint paths p_3 and q_3 (p_4 and q_4) in J_{d_1} (J_{d_2}). Specifically, there must be at least r_{d_1} (r_{d_2}) units of residual capacity on every link in the cycle formed by p_3 and q_3 (p_4 and q_4) in order to use them as the new working and backup paths for d_1 (d_2). This criterion along with the total cost of the paths is used to determine the set of candidate trials for d_1 and d_2.

Figure 6 About Here.

Given a set of working and backup paths that meet the above criteria, our procedure constructs a candidate path assignment by rerouting demands d_1 and d_2. This is done with the subroutine shown in Figure 7 which also updates the residual capacity for the links in the proposed new working and backup paths p_3, p_4, q_3, and q_4. If the residual capacities on these links are all non-negative, then the candidate path assignment is feasible and the procedure

updates the x, y, and z variables. Otherwise, the procedure unroutes d_1 and d_2 and tries the next candidate trial.

Figure 7 About Here.

3 Routing on Ring Networks

In this section we describe how the incremental path-assignment algorithm described in Section 2.1 and Figure 1 can be adapted to a system of SONET rings. There are variety of ring-based protection schemes for SONET systems all of which are based on the strategy of placing node equipment on a loop (ring) of fiber-optic cable. The ring topology provides two link-disjoint paths between each pair of nodes it connects. Thus, if a particular fiber link between two nodes on the ring is broken, service can be restored quickly and automatically by rerouting traffic in other direction around the ring. Each node on a SONET ring uses a device called an *add-drop multiplexer* to send and receive traffic to and from other nodes. A node may be placed on multiple rings, but requires one ADM for each ring on which it is placed (see [17] and [21] for more information at SONET rings). Our approach to the incremental path-assignment problem for ring-based systems is to model them as mesh networks and then apply the results of Section 2.1. For this investigation we only considered 4-fiber, *bi-directional line-switched rings (BLSR)* each having a capacity of OC-192, but our approach can easily be modified for other types of rings.

When designing SONET ring networks, one approach is (a) to determine a set of candidate rings (cycles) that provide connectivity between all nodes, then (b) to decide which of these candidate rings should be equipped with one or more SONET ADM ring systems. Figure 8 shows an example network with four candidate rings. Based on the offered traffic demands, ten OC-192 SONET ring systems are required to carry the traffic, as shown in Figure 9 . Notice that multiple SONET ring systems are "stacked" on each candidate ring. The notation $n_i r_j$ refers to an ADM in SONET ring system j located at node i of the network in Figure 8a. A SONET ring system that is based on a particular candidate ring does not have to have ADMs at every node in the candidate ring. For example, observe that SONET ring systems 5 and 6 in Figure 9 are based on candidate ring 2 in Figure 8b, but have *glassthroughs* at node 3.

Figures 8 and 9 About Here.

Figure 10 depicts a mesh representation of a network consisting of SONET ring systems 3, 4, 5, and 6. In this figure, nodes n_1, n_2, n_3, and n_4, represent digital cross connects (DCS) placed at nodes $1, 2, 3$, and 4 of the network in Figure 8a. A DCS at node i is used to switch traffic from one SONET ring system to an other. For example, the DCS at node 1 in Figure 10 is used to switch traffic between the ADMs installed at node 1 in SONET ring systems 3, 4, 5 and 6. In this investigation, we use OC-48 as the base demand unit. Since we are working with OC-192 rings, each ADM has the capacity to add/drop a total of 8 OC-48s: four from/to the "East" direction and four from/to the "West". Thus, the link from node n_i to node $n_i r_j$ in our mesh representation has a capacity of eight units and the links between nodes on the same modular ring have a capacity of four units. Our procedure for converting a network of SONET ring systems to a mesh representation is described as follows:

1. If node i of the underlying network has ADMs connecting it to SONET ring systems $j_1, j_2, \ldots, j_k$, then the node set in the mesh representation will contain nodes $n_i, n_i r_{j_1}, n_i r_{j_2}$, $\ldots, n_i r_{j_k}$, and links $(n_i, n_i r_{j_1}), (n_i, n_i r_{j_2}), \ldots, (n_i, n_i r_{j_k})$. These links have a capacity of eight units.

2. If SONET ring system j links nodes $i_1, i_2, \ldots, i_k$ in clockwise order, then the edge set E in the mesh representation contains links $(n_{i_1} r_j, n_{i_2} r_j), (n_{i_2} r_j, n_{i_3} r_j), \ldots, (n_{i_k} r_j, n_{i_1} r_j)$. These links have a capacity of four units.

3. A point-to-point demand for the *o-d* pair *i-j* may be routed over any path in the mesh that starts at node n_i and ends node n_j provided that there is sufficient capacity on each of the links in the path. To determine the cost of a path in the mesh representation we assign a cost of two to each link connecting an ADM node to a DCS node (i.e., each eight-unit link) and cost of one unit to each intra-ring link (i.e., each four-unit link). This metric favors paths with fewer hops and fewer ring-to-ring transitions.

4. Since SONET rings automatically provide a form of dedicated protection, we don't need to find protection paths for the demands. Thus, the incremental path assignment problem for SONET rings maps to the the no-protection model in Section 2.1.

Figure 11 illustrates the node architecture and traffic for node 4 for the complete network consisting of all SONET ring systems depicted in Figure 9.

Figures 10 and 11 About Here.

4 Empirical Analysis

In this section, we present the results of our computational experiments with the procedures described in Figures 1 and 3. The data files and codes are available on the World Wide Web at `http://www.engr.smu.edu/~olinick/papers/da/da.html`. All computational experiments reported in this paper were made on a Compaq DS20E AlphaServer System with dual EV6.7 processors and 4GB RAM.

Table 2 gives the problem characteristics for the eight mesh problems in our test suite. We constructed three basic models for each of the eight problems: one with no protection using the ten shortest, loopless paths as J_d for each $d \in D$, one using up to ten cycles per demand for a solution with 1+1 protection, and one using the shared protection model. Table 3 gives the total number of paths used for each problem instance for each of the three models.

Tables 2 and 3 About Here.

4.1 No Protection Model

We implemented model (1)-(4) with the AMPL modeling language and solved the problem instances with the CPLEX mixed integer programming solver. The ILP characteristics and results for the problems without protection may be found in Table 4.

Table 4 About Here.

To test the procedure for finding an incremental path-assignment sequence, we used two starting solutions for each of the eight problem instances. The first solution was generated by changing the objective function in the path-assignment ILP from minimize to maximize - that is, we used CPLEX to find the worst possible solution. The second starting solution was obtained by restoring the original objective function and adding a constraint that the solution cost at least 4% more than the optimal solution.

Tables 5 and 6 About Here.

Table 5 gives a summary of the results of using the incremental path-assignment procedure with the worst possible starting solution for the no-protection model. On average, the starting solutions ($\hat{x}^0$) were 55.3% more expensive than optimal. In all eight cases, the sequence of incremental solutions terminated in an optimal solution. As shown in Table 6, the procedure also found optimal solutions for all problems except DA50 when the initial path assignment was already fairly close to optimal (i.e., within about 4% of optimality). The average solution time reported in Tables 5 and 6 is 3.8 seconds.

Based on the above results, it is clear that this simple heuristic can be applied to improve a sub-optimal routing. Ideally, a network manager would first obtain an optimal solution and compare this objective value with that of the current solution. If the current solution is close to optimal, then no changes should be planned. However, if changes appear to be needed, the incremental path-assignment procedure can be run to determine the sequences of modifications required. One disadvantage of our procedure is that it may require many minor modifications to the initial routing. As illustrated in Table 5, DA200 with 200 demands required 199 successful trials. Since each successful trial reroutes at least one demand, this means that on average, every demand routing must be modified at least once and some require multiple changes. At ten changes each day, it could take a month to modify the routings. Some of these changes result in only minor improvements and should be ignored. We modified the heuristic to ignore any new assignment that failed to yield at least a 10% cost reduction for the current pair. Tables 7 and 8 summarize the results from these runs. The final solutions were not quite as good, but the reduction in the number of changes required was substantial. For DA200 the modifications were reduced from 199 and 31 to 134 and 11. Based on these results, we believe that most network managers will prefer sequences requiring the 10% minimum improvement. Therefore, we imposed this requirement on the computational runs for the models providing network protection.

Tables 7 and 8 About Here.

4.2 Dedicated Protection Model

Optimal solutions for the test cases using dedicated protection may be found in Table 9. The results from running the incremental path-assignment procedure on the 1+1 protection problems are given in Tables 10 and 11. On average, the worst possible starting solutions were 51.9% above optimal and the procedure found incremental sequences of solutions leading to path assignments that were within 1.7% of optimality. Starting with solutions that were approximately 4% more expensive than optimal, the procedure found solutions that were, on average, only 0.8% more expensive than the optimal path assignments. The worst-case time for the runs reported in Tables 10 and 11 was 62 seconds.

Tables 9, 10, and 11 About Here.

4.3 Shared Protection Model

As shown in Table 12, the MIP's for the shared protection model are significantly larger and more time consuming to solve than those arising from the models without protection or with 1+1 protection. The results from running the incremental path-assignment procedure outlined in Figure 3 on the shared-protection model are given in Table 13. On average the starting solutions listed in Table 13 were approximately 33% above optimal and the procedure found incremental sequences of solutions leading to path assignments that were within 6% of optimality in all eight cases. The average solution time for these runs was about 34 minutes. The largest problem required 95 routing modifications.

Tables 12 and 13 About Here.

4.4 SONET Ring Problems

We generated six problems for a SONET ring architecture using a commercial SONET planning software system. The base topology for these problems comes from a network that has 35 links connecting 18 European cities and is described in [20]. We input the topology of this network and 330 point-to-point demands for 1 to 4 OC-48s each into the planning software which then proposed a system of 211 modular rings (OC-192 4-fiber BLSRs). Our six problem instances all use the ring system proposed by the planning software and a randomly selected subset of

the demands. We converted these problems to a mesh representation as described in Section 3 using the routes proposed by the planning software and the 10 shortest paths for each demand as the candidate paths. The characteristics for the mesh representations of these problems are given in Tables 14 and 15.

Tables 14 and 15 About Here.

Table 16 gives a summary of the results of using the incremental path-assignment procedure with the worst possible starting solution for the SONET ring problems. On average, the starting solutions ($\hat{x}^0$) were 48.11% more expensive than optimal. In all six cases, the sequence of incremental solutions terminated in a near-optimal solution with an average final deviation from optimal of 1.57%. With the exception of DA200, the heuristic required less than a minute of CPU time for these problems and the average solution time reported in Table 16 is less than five minutes.

Table 16 About Here.

5 Summary and Conclusions

In this investigation, we present procedures to produce a sequence of path-assignment modifications that can be used to modify a poor set of assignments to one that is near optimal. The potential for major service disruptions is reduced since changes can be made incrementally to achieve a near optimal solution. The procedures apply to both mesh and ring architectures including applications that require backup path protection as well as those for which protection is not required.

Our procedures are for the specific case of $k = 2$ (i.e. modify at most two working paths and two backup paths at each iteration); however, the basic idea could be extended in a fairly straight-forward way to work with larger values of k. Although we cannot guarantee that our procedures will always find optimal working and backup paths, in our experiments they produced near-optimal solutions for large, realistic problem instances with 68 nodes, 107 links, up to 200 origin-destination pairs with 10 routing paths each. The solutions produced were

good regardless of the quality of the initial starting solution. Based on our computational results, this appears to be an effective heuristic for solving real-world instances of this problem.

An interesting direction for further study of this problem would be to consider alternative problem statements. For example, what is the minimum number of incremental changes required to move from the initial path assignment to one that is within a given percentage of optimality? Or, what is the minimum number of moves required to improve the solution by a given amount?

References

[1] C. Anderson, K. Fraughnaugh, M. Parker, and J. Ryan. Path assignment for call routing: An application of tabu search. *Annals of Operations Research*, 41:301–312, 1993.

[2] T. Chujo, H. Komine, K. Miyazaki, T. Ogura, and T. Soejima. Distributed self-healing network and its optimum spare-capacity assignment algorithm. *Electronics and Communication in Japan*, 74:1–9, 1991.

[3] L. Cox, L. Davis, and Y. Qiu. Dynamic anticipatory routing in circuit-switched telecommunications networks. In L. Davis, editor, *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[4] O. Gertsel and R. Ramaswami. Optical layer survivability: A services perspective. *IEEE Communications Magazine*, pages 104–113, March 2000.

[5] O. Gertsel and R. Ramaswami. Optical layer survivability: an implementation perspective. *IEEE Journal on Selected Areas in Communications*, 18:1885–1899, 2000.

[6] W. Grover, T. Bilodeau, and B. Venables. Near optimal synthesis of a mesh restoreable network. In *GLOBECOM '91*, pages 2007–2012, 1991.

[7] W. Grover and D. Stamatelakis. Cycle-oriented distributed preconfiguration: Ring-like speed with mesh-like capacity for survivable networks with hop limits. In *Proceedings ICC '98*, volume 1, pages 537–543, 1998.

[8] M. Herzberg. A decomposition approach to assign spare capacity channels in self-healing networks. In *GLOBECOM '93*, pages 1601–1605, 1993.

[9] M. Herzberg and S. Bye. An optimal spare capacity assignment model for survivable networks with hop limits. In *GLOBECOM '94*, volume 3, pages 1601–1606, 1994.

[10] R. Iraschko, M. MacGregor, and W. Grover. Optimal capacity placement for path restoration in STM or ATM mesh survivable networks. *IEEE/ACM Transactions on Networking*, 6(3):325–336, 1998.

[11] J. Kennington and M. Lewis. The path restoration version of the spare capacity allocation problem with modularity restrictions: Models, algorithms, and an empirical analysis. *INFORMS Journal on Computing*, 13:181–190, 2001.

[12] J. Kennington and J. Whitler. An efficient decomposition algorithm to optimize spare capacity in a telecommunications network. *INFORMS Journal on Computing*, 11(2):149–160, 1999.

[13] M. Laguna and F. Glover. Bandwidth packing: A tabu search approach. *Management Science*, 39:492–500, 1993.

[14] K. Murakami and H. Kim. Optimal capacity and flow assignment for self-healing ATM networks based on line and end-to-end restoration. *IEEE/ACM Transaction on Networking*, 6:207–221, 1998.

[15] G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization*. John Wiley and Sons, New York, NY, 1988.

[16] M. Parker and J. Ryan. A column generation algorithm for bandwidth packing. *Telecommunications Systems*, 2:185–195, 1994.

[17] R. Ramaswami and K. Sivarajan. *Optical Networks: A Practical Perspective*. Morgan Kaufman Publishers, Inc., San Francisco, CA, second edition, 2002.

[18] D. Stamatelakis and W. Grover. IP layer restoration and network planning based on virtual protection cycles. *IEEE Journal on Selected Areas in Communications*, 18:1938–1949, 2000.

[19] D. Stamatelakis and W. Grover. Theoretical underpinnings for the efficiency of restorable networks using preconfigured cycles ("p-cycles"). *IEEE Transactions on Communications*, 48:1262–1265, 2000.

[20] B. Van Caenegem, W. Van Parys, F. De Turck, and P. Demesster. Dimensioning of survivable WDM networks. *IEEE Journal on Selected Areas in Communications*, 16(7):1146–1157, 1998.

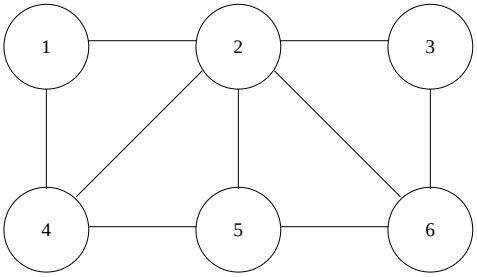
[21] T-H. Wu. *Fiber Network Service Survivability*. Artech House, Inc., 1992.

Procedure: Find Incremental Path-Assignment Sequence for $k = 2$
Input: A feasible path assignment $\hat{x}^0$, a_p for all $p \in P$, c_e for all $e \in E$, and r_d for all $d \in D$
Output: An incremental sequence of feasible path assignments $\hat{X} = \{\hat{x}^1, \hat{x}^2, \ldots, \hat{x}^\ell\}$
{Iinitialization: let c_path[d] be the current path assigned to $d \in D$}
For $d \in D$, $p \in J_d$
  If $\hat{x}^0_p = 1$ **Then** c_path[d] $\leftarrow p$

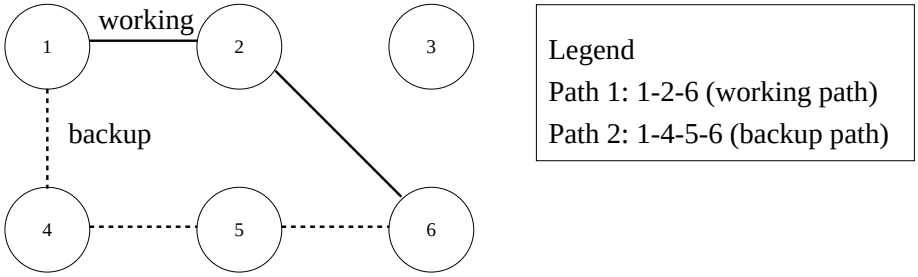
done $\leftarrow$ "false", $i \leftarrow 0$
Repeat
  done $\leftarrow$ "true"
  For Each $d_1, d_2 \in D$ **Do**
    $p_1 \leftarrow$ c_path[d_1], $p_2 \leftarrow$ c_path[d_2]
    $C \leftarrow \{(q_1 \in J_{d_1}, q_2 \in J_{d_2}) : a_{q_1} + a_{q_2} < a_{p_1} + a_{p_2}\}$
    better_routing_found $\leftarrow$ "false"
    Repeat
      For Each $(q_1, q_2) \in C$ such that $a_{q_1} + a_{q_2} = \min_{(p,q)\in C}(a_p + a_q)$ **Do**
        feasible $\leftarrow$ "true", $C \leftarrow C \setminus \{q_1, q_2\}$
        For $\{e \in E : q_1 \in P_e$ or $q_2 \in P_e\}$ **Do**
          {find the total flow on link e in the proposed rerouting}
          $f \leftarrow 0$
          For $d \in D \setminus \{d_1, d_2\}$
            If c_path[d] $\in P_e$ **Then** $f \leftarrow f + r_d$
          If $q_1 \in P_e$ **Then** $f \leftarrow f + r_{d_1}$
          If $q_2 \in P_e$ **Then** $f \leftarrow f + r_{d_2}$
          If $f > c_e$ **Then**
            feasible $\leftarrow$ "false" **And Break**

        If feasible **Then**
          Begin
          $i \leftarrow i + 1$, $\hat{x}^i \leftarrow \hat{x}^{i-1}$
          $\hat{x}^i_{p_1} \leftarrow 0$, $\hat{x}^i_{q_1} \leftarrow 1$, c_path[$d_1$] $\leftarrow q_1$
          $\hat{x}^i_{p_2} \leftarrow 0$, $\hat{x}^i_{q_2} \leftarrow 1$, c_path[$d_2$] $\leftarrow q_2$
          $\hat{X} \leftarrow \hat{X} \cup \{\hat{x}^i\}$
          done $\leftarrow$ "false", better_routing_found $\leftarrow$ "true"
          Break
          End
      End
    Until better_routing_found = "true" **Or** $|C| = 0$
  End
Until done = "true"
Return $\hat{X}$

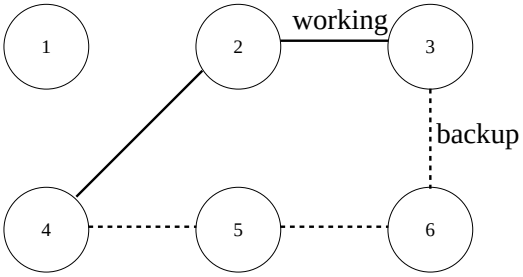
Figure 1: Pseudocode for Incremental Path-Assignment



a. Network Topology



b. Paths Used for Demand 1-6 of 4 OC48s



c. Paths Used for Demand 4-3 of 6 OC48s

Figure 2: Example Network With Two Demands

Procedure: Incremental Path-Assignment Sequence with Shared Protection for $k = 2$
Input: A feasible path assignment with shared protection $[\hat{x}^0, \hat{y}^0, \hat{z}^0]$,
a_p for all $p \in P$, c_e for all $e \in E$, and r_d for all $d \in D$
Output: A sequence of incremental path assignments $[\hat{X}, \hat{Y}, \hat{Z}]$
where $\hat{X} = \{\hat{x}^1, \hat{x}^2, \ldots, \hat{x}^\ell\}$, $\hat{Y} = \{\hat{y}^1, \hat{y}^2, \ldots, \hat{y}^\ell\}$, and $\hat{Z} = \{\hat{z}^1, \hat{z}^2, \ldots, \hat{z}^\ell\}$,

(working_path, backup_path, w, b, rc) $\leftarrow$ Initialize($\hat{x}^0, \hat{y}^0, \hat{z}^0$)
Repeat
  done $\leftarrow$ "true"
  For Each $d_1, d_2 \in D$ **Do**
    $p_1 \leftarrow$ working_path[d_1], $p_2 \leftarrow$ working_path[d_2]
    $q_1 \leftarrow$ backup_path[d_1], $q_2 \leftarrow$ backup_path[d_2]
    $(rc, b) \leftarrow$ Unroute($d_1, d_2, p_1, q_1, p_2, q_2, b$)
    $C \leftarrow$ Generate_Trials($d_1, d_2, p_1, q_1, p_2, q_2, rc$)
    better_routing_found $\leftarrow$ "false"
    Repeat
      $C' \leftarrow \{(p_3, q_3, p_4, q_4) \in C :$
      $a_{p_3} + a_{q_3} + a_{p_4} + a_{q_4} = \min_{(p_5,q_5,p_6,q_6)\in C}(a_{p_5} + a_{q_6} + a_{p_6} + a_{q_6})\}$
      For $(p_3, q_3, p_4, q_4) \in C'$ **Do**
        $(rc, b) \leftarrow$ Reroute($d_1, d_2, p_3, q_3, p_4, q_4, b$)
        feasible $\leftarrow$ "true"
        For $e \in E_{p_3} \cup E_{p_4} \cup E_{q_3} \cup E_{q_4}$ **Do**
          If $rc_e < 0$ **Then**
            feasible $\leftarrow$ "false", $(rc, b) \leftarrow$ Unroute($d_1, d_2, p_3, q_3, p_4, q_4, b$)
            Break
        If feasible **Then**
          $x \leftarrow \hat{x}^i$, $y \leftarrow \hat{y}^i$, $z \leftarrow \hat{z}^i$
          For $p \in J_{d_1} \cup J_{d_2}$ **Do** $x_p \leftarrow 0$, $y_p \leftarrow 0$
          For $p \in J_{d_1} \cup J_{d_2}$, $e \in E$ **Do** $z_{pe} \leftarrow 0$
          $x_{p_3} \leftarrow 1$, $x_{p_4} \leftarrow 1$, $y_{q_3} \leftarrow 1$, $y_{q_4} \leftarrow 1$
          For $\{e \in E : p_3 \in P_e\}$ **Do** $z_{q_3 e} \leftarrow 1$
          For $\{e \in E : p_4 \in P_e\}$ **Do** $z_{q_4 e} \leftarrow 1$
          $i \leftarrow i + 1$, $\hat{x}^i \leftarrow x$, $\hat{y}^i \leftarrow y$, $\hat{z}^i \leftarrow z$
          $\hat{X} \leftarrow \hat{X} \cup \{\hat{x}^i\}$, $\hat{Y} \leftarrow \hat{Y} \cup \{\hat{y}^i\}$, $\hat{Z} \leftarrow \hat{Z} \cup \{\hat{z}^i\}$
          working_path[d_1] $\leftarrow p_1$, working_path[d_2] $\leftarrow p_2$
          backup_path[d_1] $\leftarrow q_1$, backup_path[d_2] $\leftarrow q_2$
          done $\leftarrow$ "false", better_routing_found $\leftarrow$ "true"
          Break
    Until better_routing_found ="true" **Or** $|C| = 0$
    If better_routing_found = "false" **Then** $(rc, b) \leftarrow$ Reroute($d_1, d_2, p_1, q_1, p_2, q_2, b$)
Until done = "true"
Return $[\hat{X}, \hat{Y}, \hat{Z}]$

Figure 3: Pseudocode for Incremental Path-Assignment with Shared Protection

Procedure: Initialize
Input: A feasible path assignment with shared protection $[\hat{x}^0, \hat{y}^0, \hat{z}^0]$
Output: Vectors working_path, backup_path, w, b and rc

done $\leftarrow$ "false", $i \leftarrow 0$
For $e \in E$ **Do** $w_e \leftarrow 0$
For $d \in D$, $p \in J_d$ **Do**
    If $\hat{x}_p^0 = 1$ **Then**
        Begin
        working_path$[d] \leftarrow p$
        For $e \in E_p$ **Do** $w_e \leftarrow w_e + r_d$
        End
    Else If $\hat{y}_p^0 = 1$ **Then** backup_path$[d] \leftarrow p$

For $\{p \in P,\ f \in E : z_{pf} = 1\}$ **Do**
    For $e \in E_p$ **Do** $b_{ef} \leftarrow b_{ef} + rp_p$

For $e \in E$ **Do** $rc_e \leftarrow c_e - w_e - \max_{f \in E} b_{ef}$

Return working_path, backup_path, w, b, rc

Figure 4: Pseudocode for Initializing_Data_Structures

Procedure: Unroute
Input: A pair of demands d_1 and d_2, with working and backup paths p_1, q_1, p_2, and q_2; and b_{ef} for all $e, f \in E$
Output: Residual and backup capacity vectors rc and b resulting from unrouting d_1 and d_2

For $f \in E_{p_1}$ **Do**
    $w_f \leftarrow w_f - r_{d_1}$
    For $e \in E_{q_1}$ **Do** $b_{ef} \leftarrow b_{ef} - r_{d_1}$

For $f \in E_{p_2}$ **Do**
    $w_f \leftarrow w_f - r_{d_2}$
    For $e \in E_{q_2}$ **Do** $b_{ef} \leftarrow b_{ef} - r_{d_2}$

For $e \in E_{p_1} \cup E_{q_1} \cup E_{p_2} \cup E_{q_2}$ **Do** $rc_e \leftarrow c_e - w_e - \max_{f \in E} b_{ef}$

Return rc, b

Figure 5: Pseudocode for Procedure Unroute

Procedure: Generate_Trials
Input: A pair of demands d_1 and d_2, with working and backup paths p_1, q_1, p_2, and q_2; and rc_e for all $e \in E$
Output: A set of trial incremental routings C for d_1 and d_2

$C \leftarrow \emptyset$
$C_1 \leftarrow \{p_3 \in J_{d_1}, q_3 \in J_{d_1} : (p_3, q_3) \notin H\}$
$C_2 \leftarrow \{(p_3, q_3) \in C_1 : \min_{e \in E_{p_3} \cup E_{q_3}} rc_e \geq r_{d_1}\}$
$C_3 \leftarrow \{p_4 \in J_{d_2}, q_4 \in J_{d_2} : (p_4, q_4) \notin H\}$
$C_4 \leftarrow \{(p_4, q_4) \in C_3 : \min_{e \in E_{p_4} \cup E_{q_4}} rc_e \geq r_{d_2}\}$
$C_5 \leftarrow C_2 \times C_4$
For $(p_3, q_3, p_4, q_4) \in C_5$ **Do**
    If $a_{p_3} + a_{q_3} + a_{p_4} + a_{q_4} \leq 0.9(a_{p_1} + a_{q_1} + a_{p_2} + a_{q_2})$ **Then**
        $C \leftarrow C \cup \{(p_3, q_3, p_4, q_4)\}$

Return C

Figure 6: Pseudocode for Generating Trials for Shared Protection for $k = 2$

Procedure: Reroute
Input: A pair of demands d_1 and d_2, with working and backup paths p_1, q_1, p_2, and q_2; and b_{ef} for all $e, f \in E$
Output: Residual and backup capacity vectors rc and b resulting from rerouting d_1 and d_2

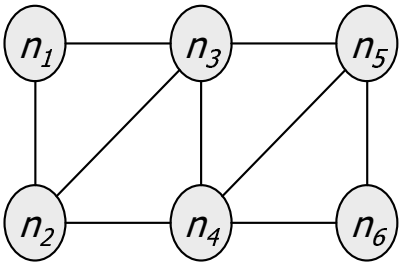
For $f \in E_{p_1}$ **Do**
    $w_f \leftarrow w_f + r_{d_1}$
    For $e \in E_{q_1}$ **Do** $b_{ef} \leftarrow b_{ef} + r_{d_1}$

For $f \in E_{p_2}$ **Do**
    $w_f \leftarrow w_f + r_{d_2}$
    For $e \in E_{q_2}$ **Do** $b_{ef} \leftarrow b_{ef} + r_{d_2}$

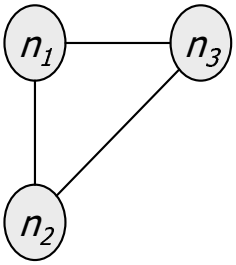
For $e \in E_{p_2} \cup E_{q_2} \cup E_{p_3} \cup E_{q_3}$ **Do** $rc_e \leftarrow c_e - w_e - \max_{f \in E} b_{ef}$

Return rc, b

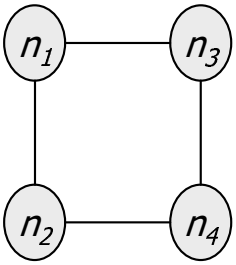
Figure 7: Pseudocode for Procedure Reroute



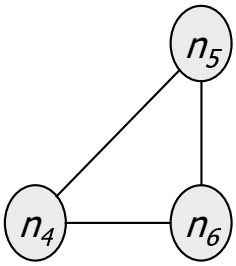
a. Example Network



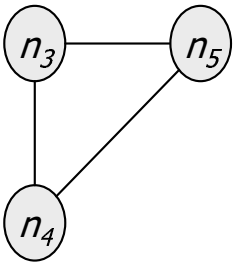
Ring 1



Ring 2



Ring 3



Ring 4

b. Candidate Rings

Figure 8: Example Network With Four SONET Rings

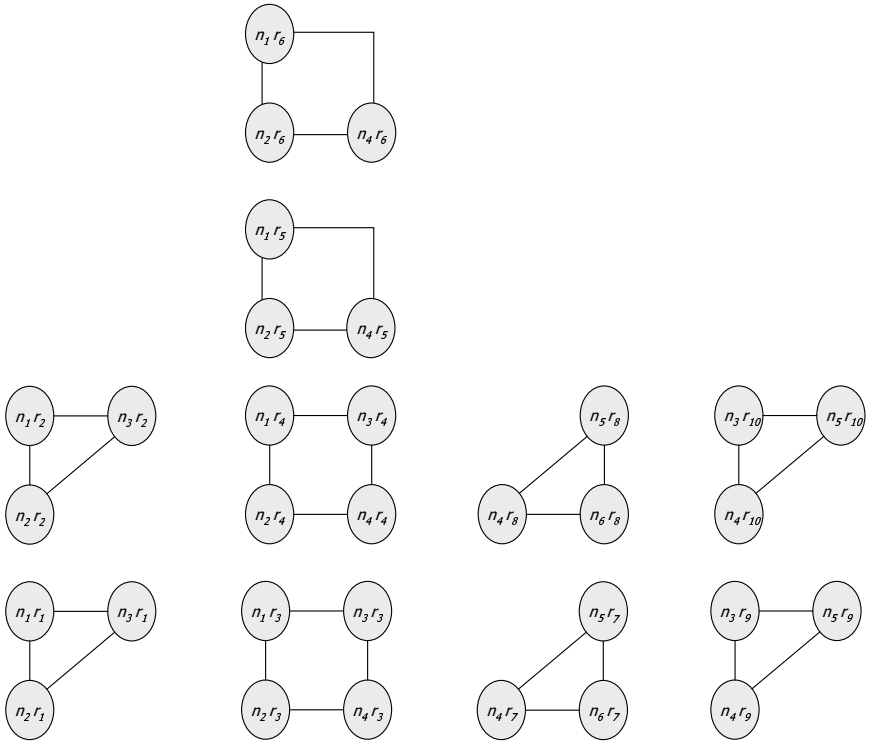


Figure 9: SONET Ring Systems

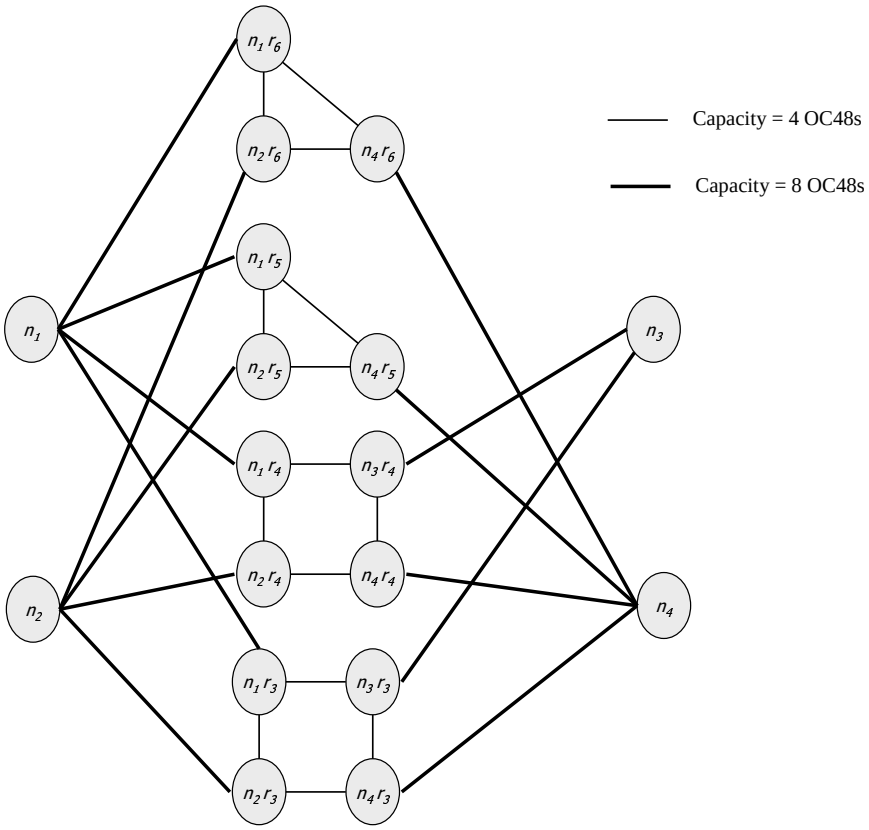


Figure 10: Mesh Representation of stacked SONET Rings

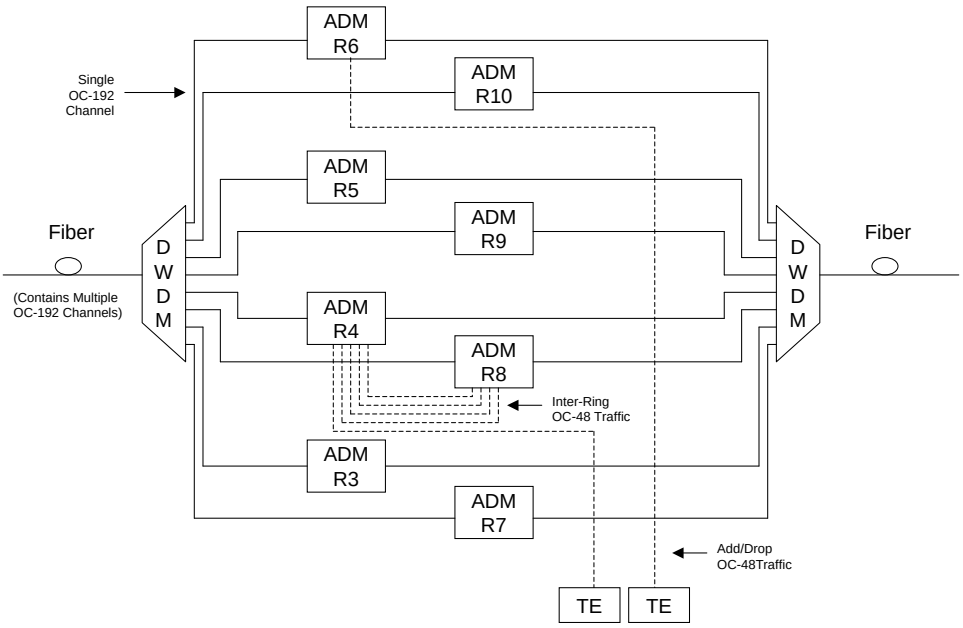


Figure 11: Node 4 Architecture and Traffic Flow

Table 1: Spare Capacity Used For Various Link Failures

Failed Link	Path Used		Spare Capacity Used			
	Demand 1-6	Demand 4-3	(1,4)	(3,6)	(4,5)	(5,6)
(1,2)	backup	working	4	0	4	4
(2,6)	backup	working	4	0	4	4
(2,3)	working	backup	0	6	6	6
(2,4)	working	backup	0	6	6	6
Spare Capacity Required Under Shared Protection			4	6	6	6
Spare Capacity Required Under Dedicated Protection			4	6	10	10

Table 2: Problem Characteristics

Problem Name	Number of		
	Nodes	Links	Demands
ATT55	11	23	55
KL100	18	33	100
DA50	68	107	50
DA75	68	107	75
DA100	68	107	100
DA125	68	107	125
DA175	68	107	175
DA200	68	107	200

Table 3: Number of Paths used in each Model

Problem Name	Protection Model		
	None	Dedicated	Shared
ATT55	550	332	518
KL100	1000	770	904
DA50	500	500	500
DA75	750	741	742
DA100	1000	984	992
DA125	1250	1216	1234
DA175	1750	1698	1734
DA200	2000	1903	1952

Table 4: ILP Characteristics and Results for Model with No Protection

Problem Name	Total Constraints	Binary Variables	CPU Seconds	Optimal Solution
ATT55	78	550	1	93
KL100	133	1,000	1	220
DA50	157	500	1	85,589
DA75	182	750	1	121,139
DA100	207	1,000	1	161,488
DA125	232	1,250	1	205,156
DA175	282	1,750	1	297,563
DA200	307	2,000	1	333,547

Table 5: Results for Model with No Protection with Worst-Possible Starting Solution

Problem Name	Starting Solution	Starting Deviation	Best Solution	Final Deviation	CPU Seconds	Number of Trials	Number Successful
ATT55	205	120.4%	93	0.0%	0.6	62	55
KL100	472	114.5%	220	0.0%	1.8	102	100
DA50	111,394	30.1%	85,589	0.0%	3.2	2,309	51
DA75	166,077	37.1%	121,139	0.0%	1.7	349	78
DA100	220,401	36.5%	161,488	0.0%	4.9	1,251	102
DA125	278,976	36.0%	205,156	0.0%	5.8	846	127
DA175	394,769	32.7%	297,563	0.0%	7.1	174	174
DA200	451,248	35.3%	333,547	0.0%	6.4	199	131
Ave.	—	55.3%	—	0.0%	—	—	—

Table 6: Results for Model with No Protection Starting 4% Above Optimal

Problem Name	Starting Solution	Starting Deviation	Best Solution	Final Deviation	CPU Seconds	Number of Trials	Number Successful
ATT55	97	4.3%	93	0.0%	0.3	3	2
KL100	229	4.0%	220	0.0%	1.5	2	2
DA50	89,013	4.0%	85,623†	0.0%	3.3	1,531	15
DA75	125,985	4.0%	121,139	0.0%	1.9	429	17
DA100	167,948	4.0%	161,488	0.0%	3.3	738	19
DA125	213,363	4.0%	205,156	0.0%	8.4	1,418	10
DA175	309,466	4.0%	297,563	0.0%	5.6	37	37
DA200	346,889	4.0%	333,547	0.0%	4.8	31	31
Ave.	—	4.0%	—	0.0%	—	—	—

†not optimal.

Table 7: Results for Model with No Protection, a Worst-Possible Start, and a 10% Minimum Incremental Improvement

Problem Name	Starting Solution	Starting Deviation	Best Solution	Final Deviation	CPU Seconds	Number of Trials	Number Successful
ATT55	205	120.4%	93	0.0%	0.6	62	55
KL100	472	114.5%	220	0.0%	1.8	102	100
DA50	111,394	30.1%	87,078	1.7%	0.5	188	34
DA75	166,077	37.1%	123,870	2.3%	1.1	63	54
DA100	220,401	36.5%	164,297	1.7%	1.9	73	72
DA125	278,976	36.0%	210,081	2.4%	2.8	91	88
DA175	394,769	32.7%	305,362	2.6%	5.6	115	115
DA200	451,248	35.3%	341,806	2.5%	7.2	134	134
Ave.	—	55.3%	—	1.65%	—	—	—

Table 8: Results for Model with No Protection Starting 4% Above Optimal, and a 10% Minimum Incremental Improvement

Problem Name	Starting Solution	Starting Deviation	Best Solution	Final Deviation	CPU Seconds	Number of Trials	Number Successful
ATT55	97	4.3%	93	0.0%	0.3	3	2
KL100	229	4.0%	220	0.0%	1.5	2	2
DA50	89,013	4.0%	86,130	0.6%	0.5	50	4
DA75	125,985	4.0%	121,557	0.3%	0.6	6	6
DA100	167,948	4.0%	162,200	0.4%	1.5	5	5
DA125	213,363	4.0%	205,679	0.3%	1.6	5	5
DA175	309,466	4.0%	299,133	0.5%	4.5	8	8
DA200	346,889	4.0%	334,395	0.3%	4.1	11	11
Ave.	—	4.0%	—	0.3%	—	—	—

Table 9: MIP Characteristics and Results for 1+1 Protection

Problem Name	Total Constraints	Binary Variables	CPU Seconds	Optimal Solution
ATT55	71	325	0.1	233
KL100	118	755	0.1	558
DA50	157	500	0.3	201,051
DA75	182	741	0.1	283,988
DA100	207	984	1.1	379,286
DA125	232	1,216	3.9	486,925
DA175	282	1,698	1.9	706,502
DA200	307	1,903	1.7	798,693

Table 10: Results for Model with 1+1 Protection, a Worst-Possible Start, and a 10% Minimum Incremental Improvement

Problem Name	Starting Solution	Starting Deviation	Best Solution	Final Deviation	CPU Seconds	Number of Trials	Number Successful
ATT55	421	80.7%	236	1.3%	2.0	1,461	60
KL100	995	78.3%	560	1.8%	4.6	944	92
DA50	284,011	41.3%	204,026	1.5%	7.8	3,510	42
DA75	429,114	51.1%	290,005	2.1%	13.7	8,942	64
DA100	554,334	46.2%	387,576	2.2%	25.7	6,949	86
DA125	692,977	42.3%	494,038	1.5%	61.8	14,465	108
DA175	980,192	38.7%	718,014	1.6%	24	1,732	128
DA200	1,090,111	36.5%	812,739	1.8%	42.2	3,167	143
Ave.	—	51.9%	—	1.7%	—	—	—

Table 11: Results for 1+1 Protection Model Starting 4% Above Optimal, and a 10% Minimum Incremental Improvement

Problem Name	Starting Solution	Starting Deviation	Best Solution	Final Deviation	CPU Seconds	Number of Trials	Number Successful
ATT55	243	4.3%	234	0.4%	2.7	1,300	5
KL100	581	4.1%	560	0.4%	3.0	531	3
DA50	209,094	4.0%	202,847	0.9%	1.3	859	5
DA75	295,348	4.0%	286,460	0.9%	0.9	479	5
DA100	394,458	4.0%	383,870	1.2%	8.3	1,696	3
DA125	506,402	4.0%	494,116	1.5%	31.4	7,880	12
DA175	734,763	4.0%	710,933	0.6%	13.5	931	11
DA200	830,641	4.0%	807,283	1.1%	35.9	3,379	12
Ave.	—	4.0%	—	0.8%	—	—	—

Table 12: MIP Characteristics and Results for Shared Protection

Problem Name	Total Constraints	Binary Variables	CPU Time	Optimal Solution
ATT55	14,500	11,371	00:00:03	225
KL100	35,430	28,190	00:00:10	548
DA50	73,198	54,500	00:05:19	197,689
DA75	103,149	80,878	00:07:10	280,808
DA100	134,172	108,128	00:19:45	373,347
DA125	164,335	134,506	01:02:34	479,326
DA175	226,565	189,006	01:26:22	698,796
DA200	253,904	212,768	01:09:17	789,394

Table 13: Results for Model with Shared Protection and a 10% Minimum Incremental Improvement

Problem Name	Starting Solution	Starting Deviation	Best Solution	Final Deviation	CPU Time	Number of Trials	Number Successful
ATT55	397	76.4%	229	1.8%	00:00:45	96	55
KL100	861	57.1%	548	0.0%	00:05:28	88	88
DA50	243,691	23.3%	208,312	5.4%	00:05:58	59	19
DA75	354,944	23.2%	297,123	5.8%	00:13:51	242	34
DA100	473,174	26.7%	385,127	3.2%	00:25:38	68	52
DA125	559,822	16.8%	508,305	6.0%	00:35:59	129	41
DA175	809,592	15.9%	722,315	3.4%	01:18:52	202	70
DA200	968,395	22.7%	815,149	3.3%	01:46:25	203	95
Ave.	—	32.8%	—	3.6%	—	—	—

Table 14: Characteristics for SONET Problems††

Problem Name	Number of			
	Nodes	Links	Demands	Paths
SONET50	605	1114	50	720
SONET75	605	1114	75	1064
SONET100	605	1114	100	1426
SONET125	605	1114	125	2018
SONET175	605	1114	175	2486
SONET200	605	1114	200	2845

††Data refer to the mesh represenation. The SONET system has 211 modular rings built on a network of 18 nodes and 35 links.

Table 15: MIP Characteristics for Mesh Represenation of SONET Ring Networks

Problem Name	Total Constraints	Binary Variables	CPU Seconds	Optimal Solution
SONET50	627	719	0.03	274
SONET75	837	1,063	0.06	438
SONET100	954	1,425	0.07	599
SONET125	1,034	2,018	0.12	757
SONET175	1,163	2,485	0.22	1,061
SONET200	1,212	2,844	0.43	1,234

Table 16: Results for SONET Ring Network with Worst Possible Starting Solution and a 10% Minimum Incremental Improvement

Problem Name	Starting Solution	Starting Deviation	Best Solution	Final Deviation	CPU Time	Number of Trials	Number Successful
SONET50	442	61.31%	283	3.28%	00:00:01	59	37
SONET75	677	54.57%	443	1.14%	00:00:04	1,668	66
SONET100	889	48.41%	605	1.00%	00:00:16	4,645	85
SONET125	1111	46.76%	758	0.13%	00:00:38	10,138	106
SONET175	1500	41.38%	1072	1.04%	00:00:20	78,824	128
SONET200	1681	36.22%	1269	2.84%	00:23:27	398,926	123
Ave.	—	48.11%	—	1.57%	—	—	—