Technical Report 05-EMIS-02

Basic Mathematical Programming Models for Capacity Allocation in Mesh-based Survivable Networks

Jeffery L. Kennington, Eli V. Olinick *,

EMIS Department
School of Engineering
Southern Methodist University
Dallas, TX 75275
fax: 214.768.1112

and

Gheorghe Spiride

Nortel Networks
Richardson, TX 75082
fax: 972.684.3713

Abstract

Designing a low cost, survivable, telecommunications network is an extremely complicated process. Most commercial products available to help with this process are based on simulation and/or proprietary heuristics. However, there is a growing consensus that mathematical programming belongs in the designer's "toolkit." Easy-to-use modeling languages coupled with powerful optimization solvers have greatly reduced the burden of implementation of mathematical programming theory into the practice of commercial network design. This manuscript presents an introduction to the basic mathematical programming models for capacity allocation that have been proposed for mesh-based survivable networks.

Key words: Capacity analysis, Integer programming, LP, Mathematical programming, Optimization, Telecommunications

1 Introduction

Reliability and capacity requirements remain major invariants in the evolution of network and node architecture for optical-fiber-based transport networks over the past two decades. Drawing upon stringent historical voice traffic requirements, availability targets are specified in terms of the maximum expected period of downtime per year that a network, point-to-point link, or piece of equipment may experience. Consequently, equipment designed to meet the strict "carrier-grade" availability requirements is expected to offer at least 99.999% availability, or no more than 5 minutes of downtime per year.

Wavelength Division Multiplexing (WDM), Dense WDM (DWDM) technology, and optical amplifier technologies are responsible for meeting capacity and scalability requirements in a cost effective manner. Individual wavelengths may carry signals at data rates exceeding 10Gbps, and available commercial systems multiplex up to 164 wavelengths inside the same fiber strand. Such large capacities increase manifold the magnitude of the disruption caused by physical link failures in a transport network. Thus, the failure of a single fiber strand could potentially affect millions of users leading to large revenue losses.

Network *survivability* is the ability of a telecommunications network to continue to provide service in the event of failures, and comprises both planning and operations aspects. The planning aspects involve different *protection schemes* for allocating spare capacity to the network, which is to be used upon the occurrence of a failure event. From a network operations standpoint, protection schemes are implemented via *restoration mechanisms* which are activated to restore service to affected customers when failure events happen. This manuscript introduces the reader to the basic optimization models developed for allocating spare capacity in a telecommunications network in order to ensure its survivability. Our focus is on the problem of optimally allocating spare capacity to implement a given protection scheme and restoration mechanism in an existing network. The models presented in this paper assume that decisions about the topology of the network have already been made. These models are based on variations of the classical multi-commodity flow problem and before we present them, we briefly discuss the complexity of representing an actual telecommunications network with an abstract network flow model.

The capacity planning models presented in this manuscript can be adapted to support node architecture configurations in which nodes provide cross-connection at the electrical level (in which case the basic unit of capacity

* Corresponding Author
Email addresses: jlk@engr.smu.edu (Jeffery L. Kennington), olinick@engr.smu.edu (Eli V. Olinick), gspiride@nortelnetworks.com (Gheorghe Spiride).

is the SONET STS-1 51.84Mbps signal), or at the optical level (when the basic unit of capacity is an individual opaque wavelength). The term opaque refers to the fact that the nature of the signal carried by that wavelength is not relevant from the point of view of the capacity models presented here. In both cases, either additional modularity constraints need to be considered for the capacity models, or the solution needs to be rounded up to valid permissible values.

Figure 1 illustrates the multi-layered node architecture of a fiber-optics based telecommunications networks. Different layers of equipment are required to handle electrical and optical domain multiplexing, and the conversion between the electrical and optical domains. Clients' traffic demand requirements are expressed in different units, depending upon the type of network considered. Several multiplexing hierarchies may be involved: SONET/SDH legacy services along with DWDM multiplexing. The fiber link between two cross-connects (OXC) comprises multiple elements such as multiplexers (MUXs), amplifiers, and regenerators. The complexity of the underlying node architecture influences the set of failure scenarios which need to be considered.

Figure 1 About Here

This figure illustrates how a physical fiber cut may affect different equipment at the same time and may trigger near-simultaneous restoration efforts at multiple layers. The loss of signal triggered by the physical fiber cut is detected at the individual OXC layer, as well as the SONET and IP router layers, since the signals carried will no longer be present. Typically, recovery efforts initiated at various layers in response to a physical failure take place on different time scales. Recovery at the optical/SONET level happens faster than the reaction time required by routers at the IP level. While certain failure patterns may be "masked" to the upper layers by the quick intervention of optical or SONET-based recovery mechanisms, there may be instances where that is not the case. Complex coordination between layers is required, made more difficult by the lack of standard interfaces (both signaling and data carrying) between equipment made by different vendors. A different, yet similar problem is that of failure patterns which are not detectable at the optical layer directly: a transponder laser failure is a typical example of this at the interface between IP and optical or SONET equipment. Patching errors may actually trigger failure detection at the OXC layer, but in a pattern which affects only a subset of the wavelengths carried on the fiber.

Notwithstanding these complexities, optical-fiber-based transport networks can be abstractly modeled as undirected graphs in which nodes correspond to locations where active equipment is placed, and edges represent optical fiber links. The remainder of this paper discusses optimization models based on the abstract graph representation of the network. However, the reader is advised

that any practical application of the methods surveyed will need to carefully consider node architecture, failure scenarios, and the extent to which restoration at various layers inter-operate. The algorithms, techniques and solution approaches surveyed in this paper were originally designed with particular choices about the node architecture, failure scenarios, etc in mind, but may also apply to other practical scenarios, provided an accurate "translation" is achieved.

The *degree* of a node is the number of links incident to the node. For this survey, a connected graph in which every node has degree 2 is called a *ring*. A graph is said to be *two-connected* if there are at least two node-disjoint paths between every pair of nodes. Every survivable network must be 2-connected. The problem of selecting a minimum cost set of edges to ensure that a network is two-connected is a fundamental problem in network design and has been studied extensively. Khuller [44] presents approximation results this and a series of related NP-hard network design problems. For surveys of the integer programming literature related to this problem see Grötschel et al. [24], Soni et al. [66], and Fortz et al. [21].

For this survey, a two-connected graph with at least one node with degree greater than 2 is called a *mesh*. Hence, all two-connected networks can be classified as having either a ring or mesh architecture. A ring is the simplest network topology that is 2-connected and it also makes service restoration relatively easy to implement: if a direct link between two nodes in a ring fails, one can simply send the traffic the link was carrying in the other direction around the ring. Hence, rings have been used extensively in the design of survivable networks. In a ring-based architecture, a large network is composed of a collection of smaller ring networks and this architecture is best suited for situations where the network can be constructed in such a way that most of the traffic it carries is between pairs of nodes on the same ring (i.e., there is relatively little intra-ring traffic). However, most backbone networks are more densely connected and use mesh architectures, with point-to-point traffic distributed over many pairs of nodes. Ramaswami and Sivarajan [60] note that typical North American backbone networks have approximately 50 nodes with an average node degree between 3 and 4, and a few of the nodes will have degree in the range 5 to 10. For networks of this type, mesh protection architectures usually require substantially less spare capacity than ring architectures.

Grover [25] presents an example where the total capacity of a mesh design was less by a factor of 2.75 (400 versus 160) than the total capacity needed for a ring design for the same traffic demand. The RingBuilder algorithm (Slevinsky et al. [64], Grover et al. [27]), the SONET Toolkit (Attanasio and Hoffman [3]), and the heurisitics in Gardner et al. [22], and Wuttisittikulkij and O'Mahony [72], do much better than the procedure described in [25], but still incur a large penalty to require building blocks with a ring structure. Due

to fast 60ms restoration offered by rings, they were used extensively during the 1990s. However, since the 60ms restoration time is not a hard number and the processing power available to implement mesh protection has dramatically increased, the research community is seeing a resurrection of mesh protection schemes. While ring design techniques are important for any design tool, this manuscript presents the basic mathematical programming models for mesh-based survivable networks. See Ramaswami and Sivarajan [60], and Wu [71] for detailed discussions of ring-based protection schemes, Chow and Lin [13] and Smith et al. [65] for a recent survey of mathematical programming models for ring-network designs.

2 Working Capacity Allocation Models

The simplest telecommunications network design problem is to determine the least capacity needed to satisfy a set of given point-to-point demands. This is called the *working capacity allocation problem* and can be modeled as an integer linear program (ILP) using either a node-arc or an arc-path formulation. Both models will be given in this section. For this presentation, a *link* denotes the bi-directional connection between a pair of nodes. For a modern DWDM telecommunications network, a link connecting nodes i and j consists of many pairs of fiber optic cable co-located in a single fiber optic duct. One member of the pair is for traffic from i to j while the other is for traffic in the opposite direction. In practice most large carriers use a separate cable for each direction of transmission.

2.1 The Node-Arc Model

Let $[N, L]$ be a graph where $N = \{1, \ldots, n\}$ denotes the set of nodes and L denotes unordered pairs of nodes corresponding to links. Let $E = \{(i, j), (j, i) : \{i, j\} \in L\}$ be a set of ordered pairs called *arcs* corresponding to the links. Flow on arc (i, j) implies that flow is from i to j. Flow in the opposite direction must be on arc (j, i). The directed graph (network) is given by $G = [N, E]$. Let d_{ij} denote the demand for traffic with origin node i and destination node j. Traffic demand need not be symmetrical (i.e., it's possible that $d_{ij} \neq d_{ji}$), and it's assumed that $d_{ii} = 0$ for $i = 1, \ldots, n$. The corresponding matrix is called the *demand* or *traffic matrix*. Since all the traffic prescribed by the demand matrix must share the same network represented by $G = [N, E]$, the problem is a member of the class of multicommodity network flow problems. An individual commodity can be expressed as either an (i, j) pair for each (i, j) such that $d_{ij} > 0$ or an origin (destination) node for each $i \in N$. Usually smaller models result from the second strategy which is adopted here. Hence,

there will be $|N| = n$ commodities, each having up to $n-1$ destinations, corresponding to each of the other nodes in N rather than (as many as) $n^2 - n$ commodities corresponding to each demand pair in the traffic matrix. Note that this approach sets up a multi-commodity flow problem, where each commodity has a single "source" and multiple "sinks" corresponding to the nodes of a given commodity's demand matrix.

Let the n-component requirement vector e^k for commodity $k \in N$ be given by

$$e_i^k = \begin{cases} \sum_{j \in N} d_{kj}, \text{ if } i = k \\ -d_{ki}, \qquad \text{otherwise.} \end{cases}$$

For a given commodity, a node requirement greater than zero corresponds to a supply node, a node requirement less than zero corresponds to a demand node, and a node requirement of zero corresponds to a transshipment node. For each arc (i, j), let the variable g_{ij}^k denote the flow of commodity k on arc (i, j). Let the variable c_{ij} denote the capacity of link $\{i, j\}$. The *node-arc formulation of the working capacity allocation model* is stated mathematically as follows:

minimize Total Working Capacity:

$$\sum_{\{i,j\} \in L} c_{ij}$$

subject to Flow Conservation:

$$\sum_{(i,j) \in E} g_{ij}^k - \sum_{(j,i) \in E} g_{ji}^k = e_i^k, \forall i \in N, \forall k \in N$$

subject to Capacity in Normal Direction:

$$\sum_{k \in N} g_{ij}^k \leq c_{ij}, \forall \{i, j\} \in L$$

subject to Capacity in Reverse Direction:

$$\sum_{k \in N} g_{ji}^k \leq c_{ij}, \forall \{i, j\} \in L$$

subject to Nonnegativity and Integrality:

$$c_{ij} \geq 0, \forall \{i, j\} \in L$$
$$g_{ij}^k \geq 0 \text{ and integer}, \forall k \in N, \forall (i, j) \in E$$

This model assumes symmetrical capacity, so that the capacity of c_{ij} on link $\{i, j\}$ can accommodate a simultaneous flow of c_{ij} in both directions. In the classical multicommodity network flow problem as defined in Kennington and Helgason [37], the capacities are constant and need not be identical in both directions.

A software implementation of all the models presented in this manuscript and the data file for the test case illustrated in Figure 2 may be found in [36]. This code was used to solve the example problem illustrated in Figure 2 with following demand matrix:

	1	2	3	4	5	6
1	-	10	0	10	10	0
2	0	-	10	10	10	10
3	0	0	-	0	10	10
4	0	0	0	-	10	0
5	0	0	0	0	-	10
6	0	0	0	0	0	-

The solution is illustrated in Figure 3a.

The advantage of this model is that it requires very little input data and implicitly considers all possible paths for every demand pair. One slight disadvantage is that some additional analysis or post processing of the LP solution is required to find the paths and flow for a given demand pair. This can be easily accomplished with a procedure similar to that suggested by Dijkstra [16] for finding shortest paths. Also some of the paths in the optimal solution may use a large number of arcs. The number of arcs in a path is known as the *hop count* and this can not be restricted in the node-arc model.

Figures 2 and 3 About Here

2.2 The Arc-Path Model

A *directed path* from node s to node t in the network $G = [N, E]$ is a sequence of nodes and arcs $p = \{i_i, (i_1, i_2), i_2, (i_2, i_3), i_3, \ldots, i_\ell, (i_\ell, i_{\ell+1}), i_{\ell+1}\}$, where $i_1 = s$, $i_{\ell+1} = t$, and each arc and node are distinct. Let D denote the set of demand pairs. That is, $(i, j) \in D$ implies that $d_{ij} > 0$. Let Q_{ij} denote the set of directed paths from i to j in $G = [N, E]$ for all $(i, j) \in D$, and let $T = \cup_{(i,j)\in D} Q_{ij}$. Let A_{ij} denote the set of paths that contain arc $(i, j) \in E$, and let y_p denote the

flow on path p. The *arc-path formulation of the working capacity allocation model* is stated mathematically as follows:

minimize Total Working Capacity:

$$\sum_{\{i,j\}\in L} c_{ij}$$

subject to Demand:

$$\sum_{p\in Q_{ij}} y_p = d_{ij}, \forall (i,j) \in D \tag{1}$$

subject to Capacity in Normal Direction:

$$\sum_{p\in A_{ij}} y_p \leq c_{ij}, \forall \{i,j\} \in L \tag{2}$$

subject to Capacity in Reverse Direction:

$$\sum_{p\in A_{ji}} y_p \leq c_{ij}, \forall \{i,j\} \in L \tag{3}$$

subject to Nonnegativity and Integrality:

$$c_{ij} \geq 0, \forall \{i,j\} \in L \tag{4}$$
$$y_p \geq 0 \text{ and integer}, \forall p \in T \tag{5}$$

One advantage of this model is that the hop count for all paths can be restricted. Paths that exceed the hop count do not appear in the sets Q_{ij}. A disadvantage is that the cardinality of the sets Q_{ij} can be very large. For most applications Q_{ij} is replaced with $\bar{Q}_{ij} \subset Q_{ij}$ where only a few of the shortest paths from i to j appear in $\bar{Q}_{ij}$. When this substitution is made, however, there is no guarantee that the arc-path model will give as good a solution as the node-arc model. The solution for the example problem is illustrated in Figure 3b. Note that the total working capacity is identical for both the node-arc and arc-path formulations, but the individual link capacities are different.

3 Spare Capacity Allocation Models

This section presents optimization models for the spare capacity allocation problem in a mesh telecommunications network. Since the users of these networks require high reliability, these networks are designed to continue to operate even when a single link failure occurs. It is generally assumed that the probability of multiple link failures during the time required to repair a failure is so small that network designers plan restoration strategies based on single link failures. Designers have identified two basic strategies to protect a network against single link failures: *dedicated protection* and *shared protection*. A discussion of each strategy follows.

The simplest idea for protecting the links in a path is to provision a node-disjoint backup path. This is also called *1+1* protection since each working path (the path(s) a demand normally takes when all links are functioning) has a backup path in reserve that will be used whenever, and only when, a link in the working path fails (see [60]). While this strategy may be required for some applications, it is generally the most expensive of the various protection strategies. In shared protection schemes, the spare capacity on a link is not dedicated to any given demand pair and may be used in the restoration of various demand pairs. Shared protection schemes come in two varieties: *link restoration* and *path restoration*. Models for each follow.

3.1 Link Restoration

In link restoration, it is assumed that each node has the capability of detecting link failures and implementing a rerouting algorithm around the defective link. If link $\{s, t\}$ fails, then restoration requires that all working traffic that uses link $\{s, t\}$ be rerouted on the reduced graph $[N, L \setminus \{s, t\}]$. For the network illustrated in Figure 3b, 10 units of spare capacity on links in one or more paths from node 1 to node 2 must be available to protect link $\{1, 2\}$. Likewise, 20 units of spare capacity on links in one or more paths from node 1 to node 4 must be available to protect link $\{1, 4\}$. Of course, the failure of link $\{1, 4\}$ implies the loss of both the working capacity and spare capacity on that link. That is, the working capacity and the spare capacity are both assigned to fibers that are carried in the same duct. A catastrophic failure usually refers to a cut that destroys all fiber in a duct. However, the spare capacity used to restore $\{1, 2\}$ is also available for the restoration of $\{1, 4\}$. Link restoration models come in node-arc, arc-path, and *p-cycle* varieties.

3.1.1 The Node-Arc Model for Link Restoration

Let c_{ij} for all $\{i,j\} \in L$ denote the known volume of working traffic on link $\{i,j\}$. Suppose link $\{s,t\}$ fails. Then c_{st} units of flow must be rerouted from node s to node t and vice versa. In the node-arc model for link restoration, the requirement at node i is given by

$$r_i^{st} = \begin{cases} c_{st}, & \text{if } i = s \\ -c_{st}, & \text{if } i = t \\ 0, & \text{otherwise.} \end{cases}$$

Let the variable h_{ij} denote the spare capacity assigned to link $\{i,j\}$ and the variable f_{ij}^{st} denote the restoration flow on arc (i,j) when $\{s,t\}$ fails. The *node-arc formulation of the link restoration version of the spare capacity allocation model* is stated mathematically as follows:

minimize Total Working Plus Spare Capacity:

$$\sum_{\{i,j\} \in L} (c_{ij} + h_{ij})$$

subject to Flow Conservation:

$$\sum_{(k,j) \in E} f_{kj}^{st} - \sum_{(i,k) \in E} f_{ik}^{st} = r_i^{st}, \forall k \in N, \forall \{s,t\} \in L$$

subject to Capacity in Normal Direction:

$$f_{ij}^{st} \leq h_{ij}, \forall \{i,j\} \in L, \forall \{s,t\} \in L$$

subject to Capacity in Reverse Direction:

$$f_{ji}^{st} \leq h_{ij}, \forall \{i,j\} \in L, \forall \{s,t\} \in L$$

subject to Link Failures:

$$f_{st}^{st} + f_{ts}^{st} = 0, \forall \{s,t\} \in L$$

subject to Nonnegativity:

$$h_{ij} \geq 0, \forall \{i,j\} \in L$$

$$f_{ij}^{st} \geq 0 \text{ and integer}, \forall \{s,t\} \in L, \forall (i,j) \in E$$

where c_{ij} for all $\{i,j\} \in L$ are fixed (usually to values determined by solving a working capacity allocation model).

This model assumes symmetrical working and spare capacity. That is, the capacity is the same in both directions on all links. This model also assumes that the working traffic was routed separately prior to determining the routing for restoration. In Section 4 we discuss models where the working and restored traffic are routed jointly. The solution of the problem instance given in Figure 3b is illustrated in Figure 4a. Note that 100 units of spare capacity are required to protect 110 units of working capacity. This is typical for this type of problem. The first investigations of the node-arc formulation for this problem may be found in Whitler [70] and Kennington and Whitler [42].

Figure 4 about here.

A clever, but different linear programming model for this problem can be found in Sakauchi et al. [62]. Their model is based on the concept of a cut and we call it the *spare capacity cutset model*. Let s and t be distinct nodes in N, and let $\bar{S}$ and $\bar{T}$ be a partition of the nodes such that $s \in \bar{S}$, $t \in \bar{T}$, $\bar{S} \cap \bar{T} = \emptyset$, and $\bar{S} \cup \bar{T} = N$. The cutset $C(s,t) = \{\{i,j\} \in L : i \in S, j \in T, \text{or } i \in T, j \in S\}$. By numbering the links in L, $1, \ldots, m = |L|$, every cutset $C(s,t)$ can be represented by an m-component vector q where $q_\ell = 1$ if link ℓ is in $C(s,t)$ and 0; otherwise. Suppose link $\{s,t\}$ fails and define B^{st} as the binary matrix whose rows are cutset vectors corresponding to all s-t cuts in $[N, L \setminus \{\{s,t\}\}]$. Then the spare capacity vector h must satisfy $B^{st}h \geq c_{st}\underline{1}$, where $\underline{1}$ is a vector of all 1s. Since this must hold for each link in L, the *spare capacity cutset model* is minimize $\{\underline{1}'h : B^{st}h \geq c_{st}\underline{1}, \forall\{s,t\} \in L, h \geq 0\}$.

Of course, the disadvantage of this formulation is that it is very difficult to generate all cutsets for a given link. However, such a model is amenable to a decomposition scheme, and one such algorithm can be found in Herzberg [30]. Herzberg also presents an extension of the basic spare capacity cutset model along with pre-processing rules that can speed convergence. A distributed real-time algorithm for finding a backup path for link restoration has been presented by Chow et al. [12]. It is a real-time version of the efficient two-tree procedure examined in Helgason et al. [29]. The problem of finding k successively shortest link-disjoint paths in the context of service restoration in telecommunications networks has been investigated by MacGregor and Grover [48]. Their algorithm will quickly find restoration paths for a failed link in a centralized (as opposed to distributed) control environment.

3.1.2 The Arc-Path Model for Link Restoration

The arc-path model for link restoration uses the set Z^{st} for all $\{s,t\} \in L$ to denote the set of directed paths from node s to node t excluding the direct

arc (s,t). Therefore, the set of all potential backup paths is given by $T = \cup_{\{s,t\}\in L} Z^{st}$. The variable h_{ij} for all $\{i,j\} \in L$ denotes the spare capacity on link $\{i,j\}$ and the variable w_p^{st} denotes the restoration flow on path p when link $\{s,t\}$ fails. The *arc-path formulation of the link restoration version of the spare capacity allocation model* may be stated as follows:

minimize Total Working Plus Spare Capacity:

$$\sum_{\{i,j\}\in L} (c_{ij} + h_{ij})$$

subject to Lost Working Capacity:

$$\sum_{p\in Z^{st}} w_p^{st} = c_{st}, \forall \{s,t\} \in L$$

subject to Link Capacities Normal Direction:

$$\sum_{p\in A_{ij}} w_p^{st} \leq h_{ij}, \forall \{s,t\} \in L, \forall \{i,j\} \in L \setminus \{\{s,t\}\}$$

subject to Link Capacities Reverse Direction:

$$\sum_{p\in A_{ji}} w_p^{st} \leq h_{ij}, \forall \{s,t\} \in L, \forall \{i,j\} \in L \setminus \{\{s,t\}\}$$

subject to Nonnegativity:

$$h_{ij} \geq 0, \forall \{i,j\} \in L$$
$$w_p^{st} \geq 0, \forall \{s,t\} \in L, \forall p \in T$$

were c_{ij} for all $\{i,j\} \in L$ are fixed. The solution for the test problem is illustrated in Figure 4b. Note that the total spare capacity is 110 as opposed to 100 in Figure 4a. This is due to the fact that not all paths are included in Z^{st}.

Using a similar model that included a hop limit to reduce the size of each Z^{st}, Herzberg and Bye [31] proposed a heuristic procedure to find the optimal integral spare capacity. A generalization of this model can be found in Herzberg et al. [32]. In a distributed, real-time algorithm used to implement link restoration, the solution can be obtained using the *max flow criteria*. However, this criteria is difficult to implement in a distributed environment and the simpler *k-successively shortest link-disjoint path criteria* has been adopted. In an

empirical investigation of these two procedures, Dunn et al. [20] found that the penalty for using the simpler procedure was only about 1%. Hence, they recommend using k-shortest paths for real-time, distributed restoration.

3.1.3 The P-Cycle Model for Link Restoration

The objective of the p-cycle concept is to obtain the restoration speed of self-healing rings (which use 1+1 protection) while simultaneously retaining the total capacity reduction offered by the shared protection paradigm. This novel idea is developed in the following series of manuscripts: Grover and Stamatelakis [28], Stamatelakis and Grover [67,68], and Grover [26].

A directed cycle containing arc (i, j) is a directed path from j to i followed by the arc (i, j). In this application, the set of links that correspond to any directed cycle is called a *p-cycle* which is equivalent to a ring. Since a p-cycle is composed of bi-directional links, restoration flow in any p-cycle can be in either a clockwise or a counter-clockwise direction. A link $\{i, j\}$ in L is said to be a *chord-link* of a given p-cycle if the p-cycle contains nodes i and j, but does not contain arc (i, j) or arc (j, i). The P-cycle model requires that every link that carries working traffic be protected by a p-cycle. That is, there must be a p-cycle containing both of the nodes incident to the link. The link $\{2, 6\}$ in Figure 5a is a chord-link and is protected by two backup paths, $\{2, (2, 1), 1, (1, 5), 5, (5, 6), 6\}$ and $\{2, (2, 3), 3, (3, 6), 6\}$, each with a capacity of 5. The link $\{1, 2\}$ in Figure 5b is a member of the p-cycle and is protected by the single backup path $\{1, (1, 5), 5, (5, 6), 6, (6, 3), 3, (3, 2), 2\}$. Note that a solution using p-cycles typically consists of multiple cycles. For example if all the links in Figure 2 carry working traffic, then a possible solution would be to use two p-cycles: $\{1, (1, 2), 2, (2, 5), 5, (5, 4), 4, (4, 1), 1\}$ protecting links $\{1, 2\}$, $\{1, 4\}$, $\{1, 5\}$, $\{2, 5\}$, and $\{4, 5\}$, and $\{2, (2, 3), 3, (3, 6), 6, (6, 5), 5, (5, 2), 2\}$ protecting the remaining links.

Figure 5 about here

Let R^{st} for all $\{s, t\} \in L$ denote the set of potential p-cycles (rings) that contain (s, t) as a link and S^{st} for all $\{s, t\} \in L$ denote the set of potential p-cycles that contain $\{s, t\}$ as a chord-link. The set of potential p-cycles that can protect $\{s, t\}$ is $F^{st} = R^{st} \cup S^{st}$ and the set of all potential p-cycles is given by $U = \cup_{\{s,t\} \in L} F^{st}$. Let C_p for all $p \in U$ denote the set of links in p-cycle p. The variable v_p for all $p \in U$ denotes the capacity of p-cycle p, and h_{ij} denotes the spare capacity of link $\{i, j\}$. Let x_p^{st} denote the restoration flow on p-cycle p when link $\{s, t\}$ fails, and let z_p^{st} be a binary variable that is one if $x_p^{st} > 0$ and is zero, otherwise. The *p-cycle spare capacity allocation model* may be stated mathematically as follows:

minimize Total Working Plus Spare Capacity:

$$\sum_{\{i,j\}\in L} (c_{ij} + h_{ij})$$

subject to Restoration Flow:

$$\sum_{p\in R^{st}} x_p^{st} + 2.0 \sum_{p\in S^{st}} x_p^{st} = c_{st}, \forall\{s,t\} \in L$$

subject to the Relationship Between x and z:

$$x_p^{st} \leq c_{st} z_p^{st}, \forall\{s,t\} \in L, \forall p \in F^{st}$$

subject to One Cycle for Restoration:

$$\sum_{p\in F^{st}} z_p^{st} = 1, \forall\{s,t\} \in L$$

subject to Cycle Capacity Number 1:

$$x_p^{st} \leq v_p, \forall\{s,t\} \in L, \forall p \in R^{st}$$

subject to Cycle Capacity Number 2:

$$0.5 x_p^{st} \leq v_p, \forall\{s,t\} \in L, \forall p \in S^{st}$$

subject to Link Capacity:

$$v_p \leq h_{ij}, \forall p \in U, \{i,j\} \in C_p$$

subject to Nonnegativity and Integrality:

$$z_p^{st} \in \{0,1\}, \forall\{s,t\} \in L, \forall p \in F^{st}$$
$$h_{ij} \geq 0, \forall\{i,j\} \in L$$
$$x_p^{st} \geq 0, \forall\{s,t\} \in L, \forall p \in F^{st}$$
$$v_p \geq 0, \forall p \in U$$

The nine potential p-cycles given in Table 1 along with the given working capacities were used to solve the problem illustrated in Figure 3b. An optimal

solution using p-cycles 2, 3, 6, 8, and 9 is illustrated in Figure 6. The total spare capacity is 140 compared to the 110 units required for the arc-path link restoration model.

Figures 5 and 6 and Table 1 about here

3.2 Path Restoration

Failure of a link may affect one or more paths that are used to carry working traffic. Therefore, restoration requires allocation of spare capacity to a set of paths that do not use the failed link. The distinction between path restoration and link restoration is that path restoration uses alternative routes from the origins to the destinations of the demand pairs affected by the failed link rather than simply taking a "detour" around it. Thus it is a distinction between "global" and "local" rerouting. Notice that p-cycles reroute all of the traffic affected by a link failure on one or two paths around the link whereas a path restoration scheme may distribute the rerouted traffic over a larger number of backup paths. Thus, path restoration will generally require less total spare capacity than either link or p-cycle restoration at the expense of a more sophisticated restoration scheme. The set V_{ij}^{st} is the set of paths available for restoration from i to j when link $\{s,t\}$ fails. That is, V_{ij}^{st} is the set of directed paths from i to j that do not use link $\{s,t\}$. Let $\bar{V}_{ij}^{st}$ be the set of directed paths from i to j that are *not* available for working traffic when link $\{s,t\}$ fails. That is, $\bar{V}_{ij}^{st} = \{p \in Q_{ij} : (s,t) \in P_p \text{ or } (t,s) \in P_p\}$. The *arc-path formulation of the path restoration version of the spare capacity allocation model* may be stated mathematically as follows:

minimize Total Working Plus Spare Capacity:

$$\sum_{\{i,j\} \in L} (c_{ij} + h_{ij}) \tag{6}$$

subject to Spare Demand 1:

$$\sum_{p \in V_{ij}^{st}} w_p^{st} = \sum_{p \in \bar{V}_{ij}^{st}} y_p, \qquad \forall \{s,t\} \in L, \forall (i,j) \in D \tag{7}$$

subject to Spare Demand 2:

$$\sum_{p \in V_{ji}^{st}} w_p^{st} = \sum_{p \in \bar{V}_{ji}^{st}} y_p, \qquad \forall \{s,t\} \in L, \forall (i,j) \in D \tag{8}$$

subject to Spare Capacity Normal Direction:

$$\sum_{p \in A_{ij}} w_p^{st} \leq h_{ij}, \forall \{s,t\} \in L, \forall \{i,j\} \in L \setminus \{\{s,t\}\} \tag{9}$$

subject to Spare Capacity Reverse Direction:

$$\sum_{p \in A_{ji}} w_p^{st} \leq h_{ij}, \forall \{s,t\} \in L, \forall \{i,j\} \in L \setminus \{\{s,t\}\} \tag{10}$$

subject to Nonnegativity:

$$h_{ij} \geq 0, \forall \{i,j\} \in L \tag{11}$$
$$w_p^{st} \geq 0, \forall \{s,t\} \in L, \forall p \in T \tag{12}$$

where c_{ij} and y_p are constants determined by solving the arc-path formulation of the working capacity allocation model.

When this model is applied to the example problem, the total spare capacity needed was only 95 compared to 140 and 110 for p-cycle and link restoration, respectively. The solution is illustrated in Figure 7. Of course, a more sophisticated restoration procedure is needed to achieve these savings.

Chujo et al. [14] present a path-based heuristic for spare capacity assignment along with a distributed algorithm for real-time restoration. They begin with an initial assignment using shortest-time routes. Alternative paths are examined in an attempt to reduce total spare capacity. This continues until a pre specified objective is reached. Doverspike and Wilson [19] compared link and path restoration in the presence of node as well as link failures. While path restoration was superior, the difference was small for networks with low levels of congestion. The difference increased as a function of traffic congestion.

In path restoration, it is possible to release the surviving parts of a working path and use them for restoration. This option is known as *stub release*. Iraschko et al. [33,34] found that path restoration with stub release yielded a 19% capacity reduction over link restoration. In a similar investigation, Murakami and Kim [56] found the savings to range from a low of 3% to a high of 55%. Xiong and Mason [73] also compared link and path protection with and without stub release. They found that stub release and path protection could be quite beneficial for large sparse networks (i.e., networks with small average node degree).

4 Joint Capacity Planning Models

A joint model is one in which both working and spare capacity can be determined in a single model. In the previous sections, working capacity was determined with one model and then spare capacity was determined to protect the optimal working capacity against single link failures. Of course, the amount of working capacity on a link determines the amount of spare capacity needed elsewhere to provide for restoration, so joint optimization should require less total capacity.

One of the first investigations using a joint model was by Murakami and Kim [53]. A column-generation technique was used to obtain new variables as needed. They experimented with a pair of problems from the literature achieving a 10% cost savings for their joint model. A full report of their investigation can be found at Murakami and Kim [55]. Joint capacity models have also been investigated by Saito et al. [61] and Iraschko et al. [34].

The joint model is a combination of the *arc-path formulation of the working capacity allocation model* and the *arc-path formulation of the path restoration version of the spare capacity allocation model.* The model is stated mathematically as minimize (6) subject to (1)-(5) and (7)-(12). When applied to the example problem, the total capacity needed was 176 compared to 205 for the two-phase approach. This solution is illustrated in Figure 8. The routing for the traffic demands in the absence of failure is split across multiple paths, which results in a smaller overall spare capacity requirement than was the case with the two-stage approaches considered earlier. The reduction in spare capacity when using a joint model is significant, but comes at the expense of splitting the working traffic routing and, possibly, an increase in the overall working traffic requirements (see Table 3 for a larger example problem which illustrates this point). Table 2 gives a summary of the results of running the code in [36] on this model, as well as those presented in the preceding sections, on the test case illustrated in Figure 2.

5 Limitations and Enhancements

The optimization models presented in this survey are abstractions of real networks. The value of any model-based design is dependent on how well the optimization model represents the actual design environment. The more accurately the model represents the designers' problem the better the resulting design. However, more accurate modelling often comes at the expense of increased computational difficulty and for large networks even the basic models can prove challenging to solve with straight-forward applications of off-the-

shelf ILP solvers. This manuscript focuses on the modelling part of network design, as opposed to specialized algorithms required to solve such models. To help demonstrate the computational tractability, the models have been solved for the European network defined in Figure 4 of [38]. This network has 18 nodes and 35 links, and the capacity allocation problem has 72 demand pairs, 610 paths, and 25 candidate p-cycles. The computational results are given in Table 3. The run was made using AMPL and CPLEX 8.00 on a Compaq DS10 running at 667 MHZ with 1280 MB of RAM. The p-cycle model was the most difficult to solve taking approximately 10 seconds of CPU time. The number of variables and constraints are those reported after preprocessing which can reduce the number of both. In addition, some integer variables were converted to binary variables. Limitations inherent in the presented formulations along with proposed modelling and algorithmic enhancements complete this manuscript.

5.1 Modelling Enhancements

Several of the models presented here allow for *demand splitting*,but in practice some clients insist that their traffic be routed on a single path. Disallowing working traffic demand splitting may come at the expense of the potential capacity savings that using a joint model may yield and must be modelled at the expense of additional integer variables — as the number of integer variables becomes large, computational efficiency becomes an issue. The case for using integer programming to handle a variety of design issues is made by Birkan et al. [9]. The models presented in this exposition provided for restoration at the lowest level (the physical layer). Anderson et al. [1] explored various techniques and capabilities for providing fast restoration at the ATM layer. In certain test cases, they found that ATM layer protection resulted in significant improvement over physical layer protection. The advantages of ATM layer protection are also touted by Kawamura et al. [35] and Murakami and Kim [54].

The problem of simultaneously selecting link capacities and point-to-point routes so that the queuing delays are acceptable and the design cost is minimal has been addressed by Gavish and Neuman [23]. Queuing delays are converted to dollars and balanced with additional link capacity. Their model is a nonlinear binary program that is quite challenging to solve.

Network topology is also important in determining the type of restoration strategy that is best. Doverspike et al. [18] show that the spare capacity needed in a mesh design is highly sensitive to the average node degree. Sparse networks require significantly more capacity than dense networks. A heuristic procedure that addresses the issue of network topology has been developed by

Kershenbaum et al. [43]. MacGregor and Grover [49] address the issue of node failures.

The models presented assume that all point-to-point demands are known with certainty at the design stage. Sen et al. [63] present a two-stage stochastic linear program for a problem with point-to-point demands represented by a multi-dimensional random variable. A special algorithm and specialized software was constructed to obtain solutions. Andrade et al. [2] present a two-stage mixed integer stochastic programming model for the node-arc version of the working capacity allocation problem when the number of demand pairs in each scenario has a Poisson distribution and the capacity required for each demand follow normal or log-normal distributions. The robust optimization methodology of Mulvey et al. [52] has been used to model uncertain demands by Laguna [46] and Kennington et al. [38]. Reliability issues have been addressed by Chamberland and Sanso [11].

In the models presented in this manuscript, it was assumed that the traffic could be physically routed along the paths specified. With wavelength division multiplexing (WDM) an added complication arises. A given node may transmit optical signals on different wavelengths that are multiplexed onto a single fiber. A *lightpath* is a logical connection between a pair of nodes in the network and is composed of a sequence of links. The assignment of wavelengths along the lightpaths used to satisfy the point-to-point demands must be distinct. The basic design problem posed by this complication is known as the routing and wavelength-assignment problem (RWA), (see Zang et al. [75] and Ramaswami and Sivarajan [60]), which may be stated as follows: given a network topology and a set of point-to-point demands, determine a lightpath and wavelength assignment for each demand using the minimum number of wavelengths. The difficulty involves the restriction that two lightpaths must not be assigned the same wavelength on the same link. In addition, RWA comes in two versions, with and without wavelength conversion. Investigations of this problem can be found in Zhang et al. [76], Kennington et al. [41], and Kennington and Olinick [40]. A comprehensive model that involves dimensioning of the network components, routing of the ligthpaths, and wavelength assignment can be found in Zymolka et al. [77]. A model for joint routing and wavelength assignment can be found in Chamberland et al. [10].

Many of the models presented permit arbitrary capacity values on the links and they implicitly assume that capacity has a linear cost function (i.e., they minimize total capacity rather than total cost). However, in today's networks capacity is typically available only in discrete, modular values. For example the industry-standard values for optical networks are OC-3, OC-12, OC-48, OC-96, and OC-192. An OC-12 link has four times the capacity of and OC-3 link, and an OC-48 link has four times the capacity of an OC-12 link, and so forth. The per-unit cost of capacity decreases from one OC level to the

next level, but not necessarily in a linear fashion (i.e., there is an economy of scale). This complication can be modelled by using additional integer variables and has been addressed by Balakrishnan et al. [4], Bienstock and Günlük [8], Doucette et al. [17], Kennington and Lewis [39], and Lewis [47]. Other examples of recent of models that take the cost of hardware configuration into account are Kröeller and Wessäely [45] and Melián et al. [50,51].

5.2 Algorithmic Enhancements

The modelling enhancements described in the preceding subsection result in models that are more difficult to solve and call for novel solution techniques. To illustrate the computational difficulty of solving the enhanced models described above and to give a sense of the size of problems that can be reasonably solved, we now briefly discuss some algorithmic enhancements developed to solve three of the models presented earlier when modular capacities are considered.

Bienstock and Günlük [8] developed cutting plane algorithms for the node-arc version of the working capacity allocation problem when capacity is installed in two modular sizes and reported results for two problem sets: one derived from a network with 15 nodes and 22 links, and the other from a network with 16 nodes and 49 links. In both cases the demand matrix is fully dense with demand between all pairs of nodes in the network. Using CPLEX 2.1 on a SPARC 10-51 to solve the basic ILP formulation the solution times ranged from 12 seconds to several hours for problems in the first data set. However, by using the cutting plane algorithms to add inequalities violated by the LP relaxation of the basic model, and then solving the resulting ILP with CPLEX, the *total* solution time dropped to under two minutes in all cases. Their cutting plane procedure took considerably longer to solve the problems in the second data set, but CPLEX was unable to solve these problems when given the basic ILP without any the cuts.

Balakrishnan et al. [4] present an extensive computational study of LP-based heuristics for the node-arc formulation of the link restoration version of the spare capacity allocation model using multiple modular capacity sizes. Using their heuristics they find good solutions (within 10% of the LP bound) to problems with up to 50 nodes, 150 links, and three modular capacities. CPLEX 3.0, running on a Sun Sparc workstation, was unable to solve many of these problems. However in the cases where CPLEX solutions were found, the heuristic solutions are often within 0.5% of optimality. Kennington and Lewis [39] propose a novel, modular branch-and-bound scheme for the node-arc formulation of the path restoration version of the spare capacity allocation model. A software implementation of this scheme was found to be substan-

tially faster than CPLEX 6.5.3. on a test suite of related problems, having up to 50 nodes, 87 links, 200 demand pairs, and three modular capacities. Their branch-and-bound scheme obtained solutions guaranteed to be within 4% of optimality in five minutes of CPU time on a DEC AlphaStation.

Survivable network design has received considerable attention in the optimization literature and due to the steady adoption of new technology in the telecommunications industry, we expect it to continue to be an active area of research. A wide array of advanced integer programming techniques have been proposed to tackle these challenging problems. For example, Dahl and Stoer [15] employ row and column generation techniques in a cutting plane procedure for minimizing the cost of adding capacity to a network to meet survivability requirements for a given demand matrix. In addition to the work of Murakami and Kim [53,55] discussed in Section 4, some other examples of applying column generation to solve arc-path formulations include Poppe and Demeester [58], and Orlowski and Wessäly [57] who use it in conjunction with a branch-and-cut algorithm to study the effect of using hop limits to constrain the number of available paths for each demand. Stidsen and Thomadsen [69] present an algorithm using column generation rather than a fixed, predetermined set of the candidate p-cycles as required by the model presented here. Belotti and Malucelli [6] present decomposition approaches based on Lagrangian relaxation for shared protection models with path restoration. This is only a small sampling of the literature on survivable network design. Recent surveys may be found in Ben-Ameur and L. Gouveia [7], Rajan and Atamtürk [59], Belotti [5], and Yuan [74].

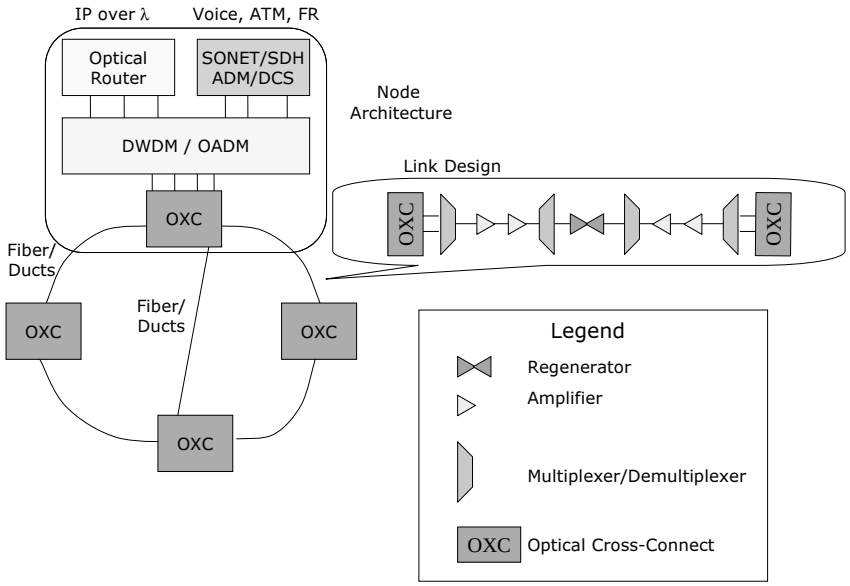


Fig. 1. Stacked Node

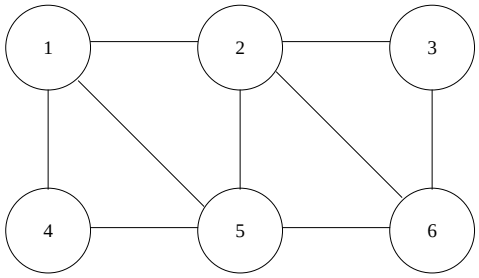
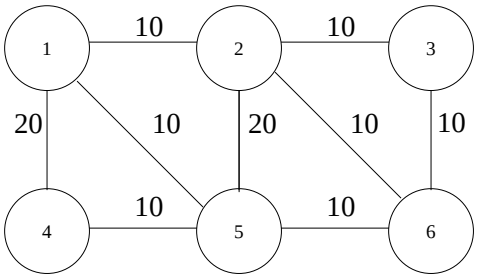


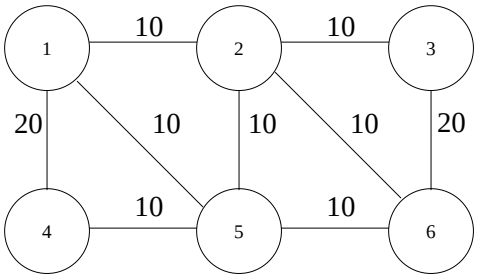
Fig. 2. Example Problem

Table 1
P-cycles for Example Problem

Links	Cycles									Working
	1	2	3	4	5	6	7	8	9	
(1,2)	x				x	x		x	x	10
(1,4)		x			x				x	20
(1,5)	x	x				x		x		10
(2,3)			x				x	x		10
(2,5)	x			x	x		x			10
(2,6)			x	x		x			x	10
(3,6)			x				x	x		20
(4,5)		x			x				x	10
(5,6)				x		x	x	x	x	10

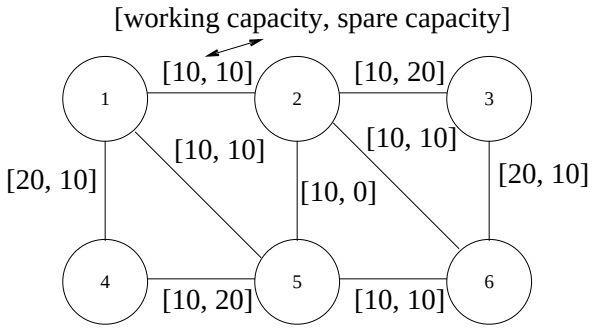


a. Solution to Node−Arc Formulation of the Working Capacity Allocation Model (Total Working Capacity is 110)

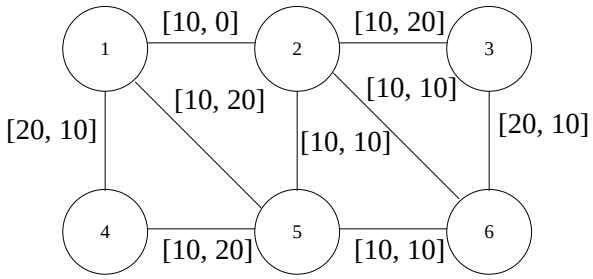


b. Solution to Arc−Path Formulation of the Working Capacity Allocation Model (Total Working Capacity is 110)

Fig. 3. Solution to Working Capacity Allocation Models

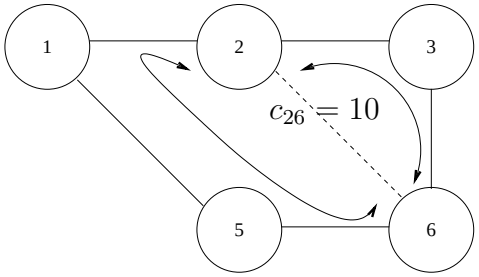


4a. Solution to the Node−Arc Formulation of the Link Restoration Version of the Spare Capacity Allocation Model

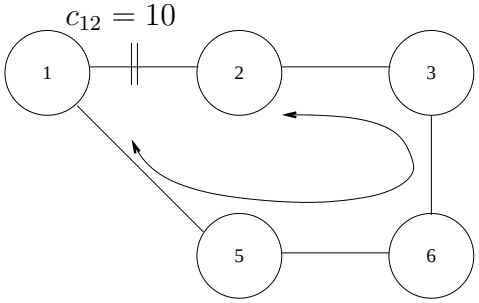


4b. Solution to the Arc−Path Formulation of the Link Restoration Version of the Spare Capacity Allocation Model

Fig. 4. Solution to Link Restoration Models



a. Link (2, 6) Protected By a P−cycle with Capacity of 5



b. Link (1, 2) Protected by a P−Cycle with Capacity of 10

Fig. 5. Example of P-cycle Protection

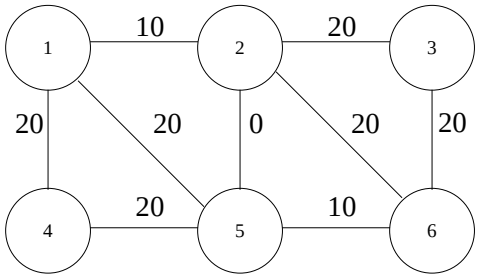


Fig. 6. Optimal Solution for P-Cycle Protection (Total Spare Capacity = 140)

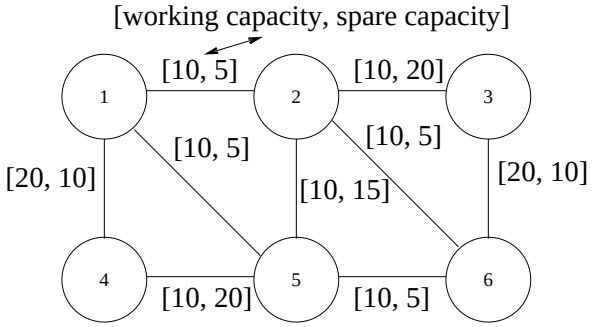


Fig. 7. Solution to the Arc-Path Formulation of the Path Restoration Version of the Spare Capacity Allocation Model (Spare Capacity = 95)

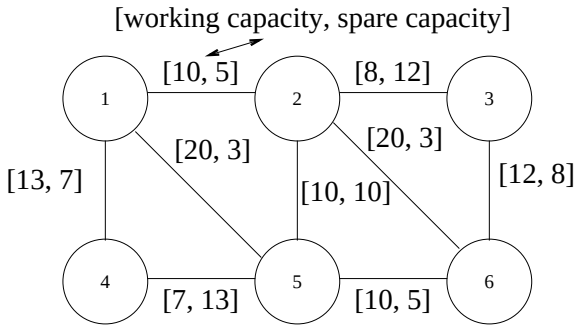


Fig. 8. Solution to the Joint Model (Working Capacity = 110 and Spare Capacity = 66)

Table 2
Summary of Results Example Problem: $|N| = 6, |L| = 9, |D| = 11, |T| = 110, |E| = 18, |U| = 9$

Model	Working Capacity	Spare Capacity	Total Capacity
Node-Arc Formulation of the Working Capacity Allocation Model	110	—	—
Arc-Path Formulation of the Working Capacity Allocation Model	110	—	—
Node-Arc Formulation of the Link Restoration Version of the Spare Capacity Allocation Model	110	100	210
Arc-Path Formulation of the Link Restoration Version of the Spare Capacity Allocation Model	110	110	220
P-Cycle Spare Capacity Allocation Model	110	140	250
Arc-Path Formulation of the Path Restoration Version of the Spare Capacity Allocation Model	110	95	205
Joint Working and Spare Capacity Model	110	66	176

Table 3
Test Run With The European Problem: $|N| = 18, |L| = 35, |D| = 72, |T| = 610, |E| = 70, |U| = 25$

Model	Variables			Const.	CPU	Capacity		
	Integer	Binary	Continuous		Seconds	Working	Spare	Total
Node-Arc Formulation of the Working Capacity Allocation Model	1,260	0	35	394	1	271	—	—
Arc-Path Formulation of the Working Capacity Allocation Model	425	5	35	106	1	271	—	—
Node-Arc Formulation of the Link Restoration Version of the Spare Capacity Allocation Model	2,380	0	35	3,010	7	271	206	477
Arc-Path Formulation of the Link Restoration Version of the Spare Capacity Allocation Model	0	0	21,303	2,412	2	271	251	522
P-Cycle Spare Capacity Allocation Model	0	165	225	549	10	271	335.75	606.75
Arc-Path Formulation of the Path Restoration Version of the Spare Capacity Allocation Model	0	0	10,096	2,470	2	271	233	504
Joint Working and Spare Capacity Model	425	5	11,147	2,857	3	289	168	457

A The AMPL Models

```
# All Models - J Kennington Jun 2005

set N;                          # N denotes the set of nodes
set L within {N,N};             # L denotes the set of links
param NP;                       # NP denotes the number of paths (must be even)
param NPD2;                     # NPD2 = NP/2
set T := 1..NP;                 # T denotes the set all all paths
set P{T} within {N,N};          # P[p] denotes the arcs in path p
                                # That is, paths are directed.
set Q{N,N} within T;            # Q[a,b] denotes the paths that can be used
                                # to satisfy demand pair (a,b)
param d{N,N} default 0.0;       # d[a,b] denotes the demand with origin node a
                                # and destination node b
param NC;                       # denotes the number of cycles
set U := 1..NC;                 # U denotes the set of cycles
set C{U} within L;              # C[k] denotes the links in cycle k

data data.txt;
display N;
display L;
let NPD2 := NP/2;
display NP, NPD2;
display T;
display P;
for {p in 1..NPD2} {
  let P[p+NPD2] := {};
  for {(i,j) in P[p]} let P[p+NPD2] := P[p+NPD2] union {(j,i)};
}
display P;
display Q;
display U;
display d;
display NC;
display C;

for{i in N}
  for {j in N: i < j} {
    let Q[j,i] := {};
    for {p in Q[i,j]}
      let Q[j,i] := Q[j,i] union {p+NPD2};
  } # j
display Q;

set E within {N,N};  # E denotes the set of arcs
```

```
                    # if (i,j) in L, then both (i,j) and (j,i) are in E
let E := L;
for {(i,j) in L} let E := E union {(j,i)};

set A {E} within {1..NP}; # A[i,j]  denotes the set of paths that use arc (i,j)
for {(i,j) in E} {
  let A[i,j]  := {};
  for {p in {1..NPD2}} {
    if (i,j) in P[p] then {
      let A[i,j]  := A[i,j]  union {p};
    }
    if (j,i) in P[p] then {
      let A[i,j]  := A[i,j]  union {p+NPD2};
    }
  }  # p
}    # (i,j)
display A;

set D within {N,N}; # D denotes the set of demand pairs
let D := {};
for {a in N}
  for {b in N: a<>b}
    if d[a,b] > 0.0 then let D := D union {(a,b),(b,a)};
display D;

set V{E,D} within T;     # V[s,t,a,b] denotes the paths from a to b
                         # that do not contain either arc (s,t) or arc (t,s)
for {(s,t) in E}
  for {(a,b) in D}
    let V[s,t,a,b] := Q[a,b] diff (A[s,t] union A[t,s]);
display V;

set W within {N};   # nodes in a given cycle
set R{L}  within U; # R[i,j] denotes the cycles that contain link (i,j)
set S{L} within U;  # S[i,j] denotes the cycles that contain link (i,j)
                    # as a chord link
set F{L} within U;  # F[i,j] = R[i,j] union S[i,j]

for {(i,j) in L} {
  let R [i,j] := {};
  let S[i,j] := {};
  for {p in U}
    if (i,j) in C[p] then let R[i,j] := R[i,j] union {p};
    else {
      let W := {};
      for {(ii,jj) in C[p]} let W := W union {ii,jj};
      if i in W and j in W then let S[i,j] := S[i,j] union {p};
```

```
    }
  let F[i,j] := R[i,j] union S[i,j];
}
display R; display S; display F;
param M default 1000000.0;
                        # denotes the maximum working capacity;

param r {L,N} default 0.0;

param e {N,N} default 0.0;
                        # e[k,i] denotes the requirement at node i
                        # for commodity k

set Z{L} within T;      # Z[s,t] denotes the set of paths from s to t
                        # excluding the direct path

var g{N,E}       >= 0 integer;
                        # g[k,i,j] denotes the flow in arc (i,j) for
                        # commodity k
var y{T}         >= 0 integer;
                        # y[p] denotes the flow on path p
var c{L}         >= 0;  # c[i,j] denotes the capacity of link (i,j)
var w {T,L}      >= 0;  # w[p,s,t] denotes the restoration flow on path p when
                        # (s,t) fails
var h{L}         >= 0;  # h[i,j] denotes the spare capacity on link (i,j)

var x {(i,j) in L, F[i,j]} >= 0;
                        # x[i,j,p] denotes the flow on cycle p required to
                        # protect the failure of link (i,j)
var z {(i,j) in L, F[i,j]} binary;
                        # z[i,j,p] = 1, if cycle p is used to protect link (i,j)
                        # z[i,j,p] = 0, otherwise
var v{U} >= 0;          # v[p] denotes the capacity of cycle p

var f{L,E} >= 0.0 integer;
                        # f[s,t,i,j] denotes the restoration flow on arc (i,j)
                        # when link (s,t) fails

minimize WorkingCapacity: sum {(i,j) in L}  c[i,j];
minimize   TotalCapacity: sum {(i,j) in L} (c[i,j] + h[i,j]);
minimize     BackupPaths: sum {p in T, (i,j) in L} w[p,i,j];

subject to FlowCons {i in N, k in N}:
  sum {(i,j) in E} g[k,i,j] - sum {(j,i) in E} g[k,j,i] = e[k,i];

subject to CapacityND {(i,j) in L}:
  sum {k in N} g[k,i,j] <= c[i,j];
```

```
subject to CapacityRD {(i,j) in L}:
  sum {k in N} g[k,j,i] <= c[i,j];

subject to WorkingDemand {(i,j) in D}:
  sum {p in Q[i,j]} y[p] = d[i,j];

subject to WorkingCapacityNormalDirection  {(i,j) in L}:
  sum {p in A[i,j]} y[p] <= c[i,j];

subject to WorkingCapacityReverseDirection {(i,j) in L}:
  sum {p in A[j,i]} y[p] <= c[i,j];

subject to SpareDemand1 {(s,t) in L, (i,j) in D: i < j}:
  sum {p in V[s,t,i,j]} w[p,s,t] =
  sum{p in Q[i,j]: (s,t) in P[p] or (t,s) in P[p]} y[p];

subject to SpareDemand2 {(s,t) in L, (i,j) in D: i < j}:
  sum {p in V[s,t,j,i]} w[p,s,t] =
  sum{p in Q[j,i]: (s,t) in P[p] or (t,s) in P[p]} y[p];

subject to SpareDemand3 {(s,t) in L}:
  sum {p in Z[s,t]} w[p,s,t] = c[s,t];

subject to SpareCapacityNormalDirection  {(s,t) in L, (i,j) in L diff {(s,t)}}:
  sum {p in A[i,j]} w[p,s,t] <= h[i,j];

subject to SpareCapacityReverseDirection  {(s,t) in L, (i,j) in L diff {(s,t)}}:
  sum {p in A[j,i]} w[p,s,t] <= h[i,j];

subject to RestorationFlow {(s,t) in L}:
      sum {p in R [s,t]} x[s,t,p] +
  2.0*sum {p in S[s,t]} x[s,t,p]   = c[s,t];

subject to Relationship_x_z {(s,t) in L, p in F[s,t]}:
  x[s,t,p] <= c[s,t]*z[s,t,p];

subject to OneCycleRestoration {(i,j) in L}:
  sum {p in F[i,j]} z[i,j,p] = 1;

subject to CycleCapacity1 {(s,t) in L, p in R[s,t]}:
  x[s,t,p] <= v[p];

subject to CycleCapacity2 {(s,t) in L, p in S[s,t]}:
  x[s,t,p]/2.0 <= v[p];

subject to LinkCapacity {p in U, (i,j) in C[p]}:
```

```
  v[p] <= h[i,j];

subject to FlowConservation {(s,t) in L, k in N}:
  sum{(k,j) in E} f[s,t,k,j] -
  sum{(i,k) in E} f[s,t,i,k] = r[s,t,k];

subject to CapNormalDirection {(s,t) in L, (i,j) in L}:
  f[s,t,i,j] <= h[i,j];

subject to CapReverseDirection {(s,t) in L, (i,j) in L}:
  f[s,t,j,i] <= h[i,j];

subject to LinkFailures {(s,t) in L}:
  f[s,t,s,t] + f[s,t,t,s] = 0;

problem NodeArcWorkingCapacity:
  g, c,
  WorkingCapacity,
  FlowCons,
  CapacityND, CapacityRD
;

problem ArcPathWorkingCapacity:
  y, c,
  WorkingCapacity,
  WorkingDemand, WorkingCapacityNormalDirection,
  WorkingCapacityReverseDirection
;

problem NodeArcSpareCapacityLinkRestoration:
  f, h,
  TotalCapacity,
  FlowConservation, CapNormalDirection, CapReverseDirection,
  LinkFailures
;

problem ArcPathSpareCapacityLinkRestoration:
  w, h, c,
  TotalCapacity,
  SpareDemand3,
  SpareCapacityNormalDirection, SpareCapacityReverseDirection
;

problem Pcycle:
  x, z, v, c, h,
  TotalCapacity,
  RestorationFlow, Relationship_x_z, OneCycleRestoration,
```

```
  CycleCapacity1, CycleCapacity2, LinkCapacity
;

problem ArcPathSpareCapacityPathRestoration1:
  c, y, w, h,
  TotalCapacity,
  SpareDemand1, SpareDemand2,
  SpareCapacityNormalDirection, SpareCapacityReverseDirection
;

problem ArcPathSpareCapacityPathRestoration2:
  c, w, h,
  BackupPaths,
  SpareDemand1, SpareDemand2,
  SpareCapacityNormalDirection, SpareCapacityReverseDirection
;

problem Joint:
  y, c, w, h,
  TotalCapacity,
  WorkingDemand,
  WorkingCapacityNormalDirection, WorkingCapacityReverseDirection,
  SpareDemand1, SpareDemand2,
  SpareCapacityNormalDirection, SpareCapacityReverseDirection
;

printf"\n\nBegin Node-Arc Working Capacity Model\n\n";
for {k in N}
  for {i in N}
    if i == k then let e[k,i] := sum{j in N} d[k,j];
    else           let e[k,i] := -d[k,i];
problem NodeArcWorkingCapacity;
solve   NodeArcWorkingCapacity;
display c;
for {k in N}
  for {(i,j) in E}
    if g[k,i,j] > 0.0 then
      printf"g[%2d,%2d,%2d] = %8.2lf\n",k,i,j,g[k,i,j];
param NodeArcWorkingCapacityOBJ;
let   NodeArcWorkingCapacityOBJ := WorkingCapacity;

printf"\n\nBegin  Arc-Path Working Capacity Model\n\n";
problem ArcPathWorkingCapacity;
solve   ArcPathWorkingCapacity;
for {p in T} {
  if y[p] > 0.0 then printf"y[%3d] = %8.2lf\n",p, y[p];
}
```

```
display c;
param ArcPathWorkingCapacityOBJ;
let   ArcPathWorkingCapacityOBJ := WorkingCapacity;

printf"\n\nBegin Spare Capacity: Link Model Node-Arc Version\n\n";
for {(s,t) in L} {
  let r[s,t,s] :=  c[s,t];
  let r[s,t,t] := -c[s,t];
}
display r;

problem NodeArcSpareCapacityLinkRestoration;
solve   NodeArcSpareCapacityLinkRestoration;
display h;

for {(s,t) in L, (i,j) in E}
  if f[s,t,i,j] > 0.0 then
    printf"x[%2d,%2d,%2d,%2d] = %8.2lf\n",s,t,i,j,f[s,t,i,j];
param NodeArcSpareCapacityLinkRestorationOBJ;
let   NodeArcSpareCapacityLinkRestorationOBJ := TotalCapacity;

printf"\n\nBegin Spare Capacity: Link Model Arc-Path Version\n\n";
for {(s,t) in L} {
  let Z[s,t] := {};
  for {p in Q[s,t]}
  if (s,t) not in P[p] then let Z[s,t] := Z[s,t] union {p};
}
problem ArcPathSpareCapacityLinkRestoration;
fix c;
solve   ArcPathSpareCapacityLinkRestoration;
for {p in T}
  if y[p] > 0.0 then
    printf"y[%3d] = %8.2lf\n",p,y[p];
display h;
for {p in 1..NP, (i,j) in L}
  if w[p,i,j] > 0.0 then
    printf"w[%3d,%2d,%2d] = %8.2lf\n",p,i,j,w[p,i,j];
param ArcPathSpareCapacityLinkRestorationOBJ;
let   ArcPathSpareCapacityLinkRestorationOBJ := TotalCapacity;


printf"\n\nBegin Spare Capacity: P-Cycle Model\n\n";
let M := 0.0;
for {(i,j) in L} if c[i,j] > M then let M := c[i,j];
display M;
problem Pcycle;
fix c;
```

```
solve   Pcycle;
display c;
display h;
for {(i,j) in L, p in F[i,j]}
  if x[i,j,p] > 0.0 then
    printf"x[%2d,%2d,%2d] = %8.2lf\n",i, j, p, x[i,j,p];

for {(i,j) in L, p in F[i,j]}
  if z[i,j,p] > 0.0 then
    printf"z[%2d,%2d,%2d] = %8.2lf\n",i, j, p, z[i,j,p];

display v;
param PcycleOBJ;
let   PcycleOBJ := TotalCapacity;

printf"\n\nBegin Spare Capacity: Path Model\n\n";
problem ArcPathSpareCapacityPathRestoration1;
fix c;
fix y;
solve   ArcPathSpareCapacityPathRestoration1;
display c;
for {p in T}
  if y[p] > 0.0 then
    printf"y[%3d] = %8.2lf\n",p,y[p];
display h;
for {p in 1..NP, (i,j) in L}
  if w[p,i,j] > 0.0 then
    printf"w[%3d,%2d,%2d] = %8.2lf\n",p,i,j,w[p,i,j];
param ArcPathSpareCapacityPathRestoration1OBJ;
let   ArcPathSpareCapacityPathRestoration1OBJ := TotalCapacity;

printf"\n\nBegin Spare Capacity: Path Model With Only Paths Needed\n\n";
problem ArcPathSpareCapacityPathRestoration2;
fix h;
solve   ArcPathSpareCapacityPathRestoration2;
display c;
for {p in 1..NP}
  if y[p] > 0.0 then
    printf"y[%3d] = %8.2lf\n",p,y[p];
display h;
for {p in 1..NP, (i,j) in L}
  if w[p,i,j] > 0.0 then
    printf"w[%3d,%2d,%2d] = %8.2lf\n",p,i,j,w[p,i,j];

printf"\n\nBegin Spare Capacity: Joint Model\n\n";
problem Joint;
```

```
unfix c;
solve   Joint;
display c;
for {p in 1..NP}
  if y[p] > 0.0 then
    printf"y[%3d] = %8.2lf\n",p,y[p];
display h;
for {p in 1..NP, (i,j) in L}
  if w[p,i,j] > 0.0 then
    printf"w[%3d,%2d,%2d] = %8.2lf\n",p,i,j,w[p,i,j];
param JointOBJ;
let   JointOBJ := TotalCapacity;

printf"\n\n";
display NodeArcWorkingCapacityOBJ;
display ArcPathWorkingCapacityOBJ;
display NodeArcSpareCapacityLinkRestorationOBJ;
display ArcPathSpareCapacityLinkRestorationOBJ;
display PcycleOBJ;
display ArcPathSpareCapacityPathRestoration1OBJ;
display JointOBJ;
```

B Data File

```
# Appendix B. Data File

param d :=
2 4 10
3 5 10
1 2 10
1 4 10
1 5 10
2 3 10
2 5 10
2 6 10
3 6 10
4 5 10
5 6 10
;

set N := 1 2 3 4 5 6;
set L := (1,2) (1,4) (1,5) (2,3) (2,5) (2,6) (3,6) (4,5) (5,6);
param NP := 100;
set P[1] := (1,2);
set P[2] := (1,5) (5,2);
set P[3] := (1,2) (2,3);
set P[4] := (1,2) (2,6) (6,3);
set P[5] := (1,5) (5,6) (6,3);
set P[6] := (1,4);
set P[7] := (1,5) (5,4);
set P[8] := (1,5);
set P[9] := (1,2) (2,5);

set P[10] := (1,4) (4,5);
set P[11] := (1,2) (2,6);
set P[12] := (1,5) (5,6);
set P[13] := (1,2) (2,3) (3,6);
set P[14] := (1,2) (2,5) (5,6);
set P[15] := (1,4) (4,5) (5,6);

set P[16] := (2,3);
set P[17] := (2,6) (6,3);
set P[18] := (2,1) (1,4);
set P[19] := (2,5) (5,4);
set P[20] := (2,5);
set P[21] := (2,6) (6,5);
set P[22] := (2,1) (1,5);
set P[23] := (2,6);
```

```
set P[24] := (2,3) (3,6);
set P[25] := (2,5) (5,6);
set P[26] := (3,2) (2,1) (1,4);
set P[27] := (3,2) (2,5) (5,4);
set P[28] := (3,6) (6,5) (5,4);
set P[29] := (3,2) (2,5);
set P[30] := (3,6) (6,5);
set P[31] := (3,6);
set P[32] := (3,2) (2,6);
set P[33] := (4,5);
set P[34] := (4,1) (1,5);
set P[35] := (4,5) (5,6);
set P[36] := (4,1) (1,5) (5,6);
set P[37] := (4,1) (1,2) (2,6);
set P[38] := (5,6);
set P[39] := (5,2) (2,6);
set P[40] := (5,2) (2,3) (3,6);

set P[41] := (1,4) (4,5) (5,2);
set P[42] := (1,2) (2,5) (5,4);
set P[43] := (2,5) (5,6) (6,3);
set P[44] := (2,3) (3,6) (6,5);
set P[45] := (2,1) (1,4) (4,5);
set P[46] := (3,2) (2,5) (5,6);
set P[47] := (4,1) (1,2) (2,5);
set P[48] := (4,1) (1,2) (2,5) (5,6);

set P[49] := (4,1) (1,2) (2,3) (3,6);

set P[50] := (4,1) (1,5) (5,2) (2,6);

set Q[1,2] := 1 2 41;
set Q[1,3] := 3 4 5;
set Q[1,4] := 6 7 42;
set Q[1,5] := 8 9 10;
set Q[1,6] := 11 12 13 14 15;
set Q[2,3] := 16 17 43;
set Q[2,4] := 18 19;
set Q[2,5] := 20 21 22 44 45;
set Q[2,6] := 23 24 25;
set Q[3,4] := 26 27 28 ;
set Q[3,5] := 29 30;
set Q[3,6] := 31 32 46;
set Q[4,5] := 33 34 47;
set Q[4,6] := 35 36 37 48 49 50;
set Q[5,6] := 38 39 40;
```

```
param NC := 9;
set C[1]  := (1,2) (1,5) (2,5);
set C[2]  := (1,4) (1,5) (4,5);
set C[3]  := (2,3) (2,6) (3,6);
set C[4]  := (2,5) (2,6) (5,6);
set C[5]  := (1,2) (1,4) (2,5) (4,5);
set C[6]  := (1,2) (1,5) (2,6) (5,6);
set C[7]  := (2,3) (2,5) (3,6) (5,6);
set C[8]  := (1,2) (1,5) (2,3) (3,6) (5,6);
set C[9]  := (1,2) (1,4) (2,6) (4,5) (5,6);
```

Acknowledgements

This work was partially supported by the Office of Naval Research under contract N00014-03-1-0053.

References

[1] J. Anderson, B. Doshi, S. Dravida, and P. Harshavardhana. Fast restoration of ATM networks. *IEEE Journal on Selected Areas in Communications*, 12(1):128–138, 1994.

[2] R. Andrade, A. Lisser, N. Maculan, and G. Plateau. Telecommunication network capacity design for uncertain demand. *Computational Optimization and Applications*, 29:127–146, 2004.

[3] R. Attanasio and K. Hoffman. The right tool kit is all a network planner needs. *Telephony*, 224(15):22–26, 1993.

[4] A. Balakrishnan, T. Magnanti, J. Sokol, and Y. Wang. Telecommunication link restoration planning with multiple facility types. *Annals of Operations Research*, 106:127–154, 2001.

[5] P. Belotti. *Multicommodity network design with survivability constraints: Some models and algorithms*. PhD thesis, Politecnico di Milano Dipartimento di Elettronica ed Informazione, Milan, Italy, 2003. Available on line at http://www.elet.polimi.it/upload/belotti/papers/thesis.pdf.

[6] P. Belotti and F. Malucelli. Shared protection network design: Valid inequalities and a decomposition approach. Technical report, Operations Research Group DEI, Politecnico di Milano, P.za L. da Vinci 32, Milano - Italy, 2003. Available on line at http://www.elet.polimi.it/upload/belotti/papers/belotti_shared_protection.pdf.

[7] W. Ben-Ameur and L. Gouveia. Some recent contributions to network optimization. *Networks*, 44:27–30, 2004.

[8] D. Bienstock and O. Günlük. Capacitated network design — polyhedral structure and computation. *INFORMS Journal on Computing*, 8(3):243–259, 1996.

[9] G. Birkan, J. Kennington, E. Olinick, A. Ortynski, and G. Spiride. Making a case for using integer programming to design DWDM networks. *Optical Networks Magazine*, 4(6):107–120, 2003.

[10] S. Chamberland, D. Khyda, and S. Pierre. Joint routing and wavelength assignment in wavelength division mulitplexing networks for permanent and reliable paths. *Computers and Operations Research*, 32(5):1073–1087, 2005.

[11] S. Chamberland and B. Sanso. Sequential and parallel approaches to incorporate reliability in the synthesis of computer networks. *Journal of Network and Systems Management*, 5(2):131–157, 1997.

[12] C. Chow, J. Bicknell, S. McCaughen, and S. Syed. A fast distributed network restoration algorithm. In *Proceedings International Phoenix Conference on Computer and Communications*, pages 1–7, 1993.

[13] T. Chow and P. Lin. The ring grooming problem. *Networks*, 44(3):194–202, 2004.

[14] T. Chujo, H. Komine, K. Miyazaki, T. Ogura, and T. Soejima. Distributed self-healing network and its optimum spare-capacity assignment algorithm. *Electronics and Communications in Japan*, 74(1):479–486, 1991.

[15] G. Dahl and M. Stoer. A cutting plane algorithm for multicommodity survivable network design problems. *INFORMS Journal on Computing*, 18(1):1–11, 1998.

[16] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[17] J. Doucette, W. Grover, and R. Martens. Modularity and economy-of-scale effects in optical networks. In *IEEE Canadian Conference Electrical and Computer Engineering*, volume 1, pages 266–231, 1999.

[18] R. Doverspike, J. Morgan, and W. Leland. Network design sensitivity studies for use of digital cross-connect systems in survivable network architectures. *IEEE Journal on Selected Areas in Communications*, 12(1):69–78, 1994.

[19] R. Doverspike and B. Wilson. Comparison of capacity efficiency of DCS network restoration routing techniques. *Journal of Network and Systems Management*, 2(2):95–123, 1994.

[20] A. Dunn, W. Grover, and M. MacGregor. Comparison of k-shortest paths and maximum flow routing for network facility restoration. *IEEE Journal on Selected Areas in Communications*, 12(1):88–99, 1994.

[21] B. Fortz, M Labbé, and F. Maffioli. Solving the two-connected network with bounded meshes problem. *Operations Research*, 48:866–877, 2000.

[22] L. Gardner, M. Heydari, J. Shah, I. Sudborough, I. Tollis, and C. Xia. Techniques for finding ring covers in survivable networks. *Proceedings of IEEE GLOBECOM '94*, 3:1862–1866, 1994.

[23] B. Gavish and I. Neuman. A system for routing and capacity assignment in computer communication networks. *IEEE Transactions on Communications*, 37(4):360–366, 1989.

[24] M. Grötschel, C. Monma, and M. Stoer. Polyhedral and computational investigations for designing communication networks with high survivability requirements. *Operations Research*, 43(6):1012–1024, 1995.

[25] W. Grover. Case studies of survivable ring, mesh and mesh-arc-hybrid networks. *Proceedings of IEEE GLOBECOM '92*, 1:633–638, 1992.

[26] W. Grover. *Mesh-based Survivable Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networks*. Prentice Hall PTR, Upper Saddle River, New Jersey, 2004.

[27] W. Grover, J. Slevinsky, and M. MacGregror. Optimized design of ring-based survivable networks. *Canadian Journal of Electrical and Computer Engineering*, 20(3):139–149, 1995.

[28] W. Grover and D. Stamatelakis. Cycle-oriented distributed pre-configuration: Ring-like speed with mesh-like capacity for self-planning network restoration. In *Proceedings IEEE International Conf. Commun. '98*, pages 537–543, 1998.

[29] R. Helgason, J. Kennington, and B. Stewart. Computational comparison of sequential and parallel algorithms for the one-to-one shortest path problem. *Computational Optimization and Applications*, 1:47–75, 1993.

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

[31] M. Herzberg and S. Bye. An optimal spare capacity assignment model for survivable networks with hop limits. *Proceedings of IEEE GLOBECOM '94*, 3:1601–1606, 1994.

[32] M. Herzberg, S. Bye, and A. Utano. The hop-limit approach for spare-capacity assignment in survivable networks. *IEEE/ACM Transactions on Networking*, 3(6):775–784, 1995.

[33] R. Irashcko, M. MacGregor, and W. Grover. Optimal capacity placement for path restoration mesh survivable networks. In *IEEE International Communications and Conference (ICC'96)*, volume 3, pages 1568–1574, 1996.

[34] R. Irashcko, 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.

[35] R. Kawamura, K. Sato, and I. Tokizawa. Self-healing ATM networks based on virtual path concept. *IEEE Journal on Selected Areas in Communications*, 12(1):120–127, 1994.

[36] J. Kenningtion, E. Olinick, and G. Spiride. Basic mathematical programming models for mesh-based survivable networks. Technical Report 05-EMIS-02, Southern Methodist University, Dallas, TX 75275, 2005. Available on line at http://www.engr.smu.edu/∼olinick/webarchive/survey.pdf.

[37] J. Kennington and R. Helgason. *Algorithms for Network Programming*. John Wiley & Sons, New York, NY, 1980.

[38] J. Kennington, K. Lewis, E. Olinick, A. Ortynski, and G. Spiride. Robust solutions for the WDM routing and provisioning problem: Models and algorithms. *Optical Networks Magazine*, 4(2):74–84, 2003.

[39] 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(3):181–190, 2001.

[40] J. Kennington and E. Olinick. Wavelength translation in WDM networks: Optimization models and solution procedures. *INFORMS Journal on Computing*, 16(2):174–187, 2004.

[41] J. Kennington, E. Olinick, A. Ortynski, and G. Spiride. Wavelength routing and assignment in a survivable WDM mesh network. *Operations Research*, 51(1):67–79, 2003.

[42] 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.

[43] A. Kershenbaum, P. Kermani, and G. Grover. MENTOR: An algorithm for mesh network topological optimization and routing. *IEEE Transactions on Communications*, 39(4):503–513, 1991.

[44] S. Khuller. Approximation algorithms for finding highly connected subgraphs. In D. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, pages 236–265. PWS Publishing Company, Boston, MA, 1997.

[45] A. Kröller and R. Wessäly. Integrated optimization of hardware configuration and capacity dimensioning in SDH and opaque WDM networks. In *Proc. of the 1st International Network Optimization Conference (INOC 2003)*, October 2003.

[46] M. Laguna. Applying robust optimization to capacity expansion of one location in telecommunications with demand uncertainty. *Management Science*, 44(11):5101–5110, November 1998.

[47] M. Lewis. *Spare Capacity Planning for Telecommunication Networks: Models and Algorithms*. PhD thesis, School of Engineering and Applied Science, Southern Methodist University, Dallas, TX, 2000.

[48] M. MacGregor and W. Grover. Optimized k-shortest-paths algorithm for facility restoration. *Software-Practice and Experience*, 24(9):823–834, 1994.

[49] M. MacGregor and W. Grover. Investigation of a cut-tree approach to network restoration from node loss. In *IEEE International Conference on Communications*, volume 3, pages 1530–1535, 1995.

[50] B. Melián, M. Laguna, and J. Moreno-Pérez. Minimizing the cost of placing and sizing wavelength division multiplexing and optical cross-connect equipment in a telecommunications network. *submitted to NETWORKS*, 2002.

[51] B. Melián, M. Laguna, and J. Moreno-Pérez. Capacity expansion of fiber optic networks with WDM systems: Problem formulation and comparative analysis. *Computers and Operations Research*, 31(3):461–472, 2004.

[52] J. Mulvey, R. Vanderbei, and S. Zenios. Robust optimization of large-scale systems. *Operations Research*, 43(2):264–281, 1995.

[53] K. Murakami and H. Kim. Joint optimization of capacity and flow assignment for self-healing ATM networks. In *IEEE International Conference on Communications*, pages 216–220, 1995.

[54] K. Murakami and H. Kim. Virtual path routing for survivable ATM networks. *IEEE/ACM Transactions on Networking*, 4(1):22–39, 1996.

[55] 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(2):207–221, 1998.

[56] M. Murakami and H. Kim. Comparative study on restoration schemes of survivable ATM networks. In *Proceedings IEEE INFOCOM '97*, volume 1, pages 345–352, 1997.

[57] S. Orlowski and R. Wessäly. The effect of hop limits on optimal cost in survivable network design. Technical Report ZIB-Report 4-23, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustrasse 7, D-14195 Berlin-Dahlem, Germant, 2004.

[58] F. Poppe and P. Demeester. Economic allocation of spare capacity in mesh-restorable networks: Models and algorithms. In *Proceedings of the 6th International Conference on Telecommunciations Systems (ICTSM 1998), Modeling and Analysis*, pages 77–86, 1998.

[59] D. Rajan and A. Atamtürk. A directed cycle-based column-and-cut generation method for capacitated survivable network design. *Networks*, 43(4):201–211, 2004.

[60] R. Ramaswami and K. Sivarajan. *Optical Networks: A Practical Perspective*. Morgan Kaufman Publishers, Inc., New York, NY, 2002.

[61] H. Saito, Y. Miyao, T. Komine, and F. Kubota. Joint capacity assignment to state-independent working and spare paths for enhanced network survivability. In *Proceedings of IEEE International Conference on Communications*, volume 3, pages 1743–1748, 1998.

[62] H. Sakauchi, Y. Nishimura, and S. Hasegawa. A self-healing network with an economical spare-channel assignment. *Proceedings of the IEEE GLOBECOM '90*, 1:438–443, 1990.

[63] S. Sen, R. Doverspike, and S. Cosares. Network planning with random demand. *Telecommunication Systems*, 3:11–30, 1994.

[64] J. Slevinsky, W. Grover, and M. MacGregor. An algorithm for survivable network design employing multiple self-healing rings. *Proceedings of IEEE GLOBECOM '93*, pages 1568–1573, 1993.

[65] J. Smith, A. Schaefer, and J. Yen. A stochastic integer programming approach to solve a synchronous optical network ring design problem. *Networks*, 44(1):12–26, 2004.

[66] S. Soni, R. Gupta, and H. Pirkul. Survivable network design: The state of the art. *Information Systems Frontiers*, 1(3):303–314, 1999.

[67] D. Stamatelakis and W. Grover. IP layer restoration and network planning based on virtual protection cycles. *IEEE JSCA Special Issue on Protocols and Architectures for Next Generation Optical WDM Networks*, 18(10):1938–1949, 2000.

[68] D. Stamatelakis and W. Grover. Theoretical underpinnings for the efficiency of restorable networks using pre-configured cycles ("p-cycles"). *IEEE Transactions on Communications*, 48(8):1262–1265, 2000.

[69] T. Stidsen and T. Thomadsen. Joint optimization of working and p-cycle protection capacity. Technical Report IMM-2004-8, DK-280 Kongens Lyngby,Denmark, Technical University of Denmark, 2004. Available on line at http://netsys.edm.trlabs.ca/p-cycles/techreports.htm.

[70] J. Whitler. *Telecommunications Network Design: Optimization Models and Efficient Algorithms*. PhD thesis, School of Engineering and Applied Science, Southern Methodist University, Dallas, TX, 1997.

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

[72] L. Wuttisittikulkij and M. O'Mahony. Design of a WDM network using a multiple ring approach. *Proceedings of GLOBECOM '97*, pages 551–555, 1997.

[73] Y. Xiong and L. Mason. Restoration strategies and spare capacity requirements in self-healing ATM networks. *IEEE/ACM Transactions on Networking*, 7(1):98–110, 1999.

[74] D. Yuan. *Optimization models and methods for communication network design and routing*. PhD thesis, Linköping University, Linköping, Sweden, 2001.

[75] H. Zang, J. Jue, and B. Mukherjee. A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks. *Optical Networks Magazine*, 1(1):47–60, 2000.

[76] J. Zang, K. Zhu, L. Sahasrabuddhe, S. Yoo, and B. Mukherjee. On the study of routing and wavelength assignment approaches for survivable wavelength-routed WDM mesh networks. *Optical Networks Magazine*, 4(6):16–28, 2003.

[77] A. Zymolka, A. Koster, and R. Wessäly. Transparent optical network design with sparse wavelength conversion. Technical Report ZIB-Report 02-34, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustrasse 7, D-14195 Berlin-Dahlem, Germant, 2002.