OPERATIONS RESEARCH CENTER

Working Paper

Modeling and Solving the Single Facility Line Restoration Problem

by
A. Balakrishnan, T.L. Magnanti,
J. Sokol and Y. Wang

OR 327-98 April 1998

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Modeling and Solving the Single Facility Line Restoration Problem

by

A. Balakrishnan, T.L. Magnanti,
J. Sokol and Y. Wang

OR 327-98 April 1998

Modeling and Solving the Single Facility Line Restoration Problem

Anant Balakrishnan, Thomas L. Magnanti, Joel Sokol, Yi Wang

April, 1998

Abstract

The network restoration problem is a specialized capacitated network design problem requiring the installation of spare capacity to fully restore disrupted network flows in the event of any edge failure. We present a new mixed integer programming formulation for a line restoration version of the problem using a single type of capacitated facility. We examine two different models reflecting the technologies used in both Synchronous Transfer Mode (STM) and Asynchronous Transfer Mode (ATM) networks. The problem is NP-complete in the strong sense. We study the problem's polyhedral structure by identifying strong valid inequalities of the underlying polyhedron. Our computational results on several real and randomly generated problems show that these inequalities considerably reduce the integrality gap, from an average of 10% to an average of under 1%. These results indicate that strong cutting planes combined with branch and bound can provide efficient algorithms for solving real-world problems in the telecommunication industry.

1. Motivation

As the telecommunications industry introduces novel sophisticated technologies, it faces new planning issues and current planning issues assume added importance. The introduction of new fiber transmission systems that can carry large amounts of data on a few strands of wire and the increasing strategic importance of corporate communications networks is a case in point. Because the fiber transmission systems have such large bandwidth, the failure of even a single transmission link in the network can create a severe service loss to customers. Therefore, a very high level of service reliability is becoming imperative for both system users and service providers.

According to a report by the Federal Aviation Administration ("Airport Phone Outage Not Isolated," *Washington Post*, September 21, 1991), between August 1990 and August 1991, 114 major telecommunications outages affected the air traffic control system. In particular, the FAA facility controlling traffic in the New York area experienced 10 major outages, and the Washington area experience 15. Cable cuts are common occurrences. For example, a beaver chewed a fiber cable in Kansas, creating a four-hour outage that caused the loss of multiple services including pilot-to-ground contact and controller-to-controller contact at one airport in the state. The number and frequency of outages underscores the fact that the telecommunications system used by the FAA has been vulnerable because it contained no redundancy. Magnanti and Wang [20] provide more examples of recent network disasters.

To prevent this kind of catastrophe, telecommunication companies such as AT&T have invested billions of dollars to develop the Fast Automated Restoration (FASTAR) system. This system uses the Digital Access and Cross-Connect System (DACS) and intelligent real-time routing to restore 67,200 voice circuits in minutes ("Network restoration," *Telephone Engineer & Management*, August, 1993). Dynamic restoration schemes use pre-assigned spare capacity in the network to accommodate the traffic when any equipment or link fails. Providing a high level of protection against fiber network failures can be very expensive because fiber transmission equipment is so costly. Therefore, reducing network protection costs while maintaining a desired level of survivability has become a key challenge for network planners and engineers. To address this situation, both telecommunication carriers and companies that maintain private networks would like to design cost-effective networks that can survive failures by rerouting traffic using pre-installed spare capacity. The objective of the *network restoration problem* (also known as the spare capacity assignment problem) is to determine where and how much spare capacity to install in a network in order to provide adequate protection at minimum cost. In many cases, planners have a choice of different facilities offering different levels of capacity at costs that exhibit economies of scale. The planners then need to determine the optimal combination of the facilities to install on the edges of the network.

This paper adds to the literature on the network restoration problems by (i) proposing a new mixed integer programming model for the line restoration problem using a single type of facility, (ii) studying the polyhedral structure of the problem by identifying strong valid inequalities, and (iii) including these inequalities in the formulation of the problem (either a priori or dynamically) to solve the problem. Adding valid inequalities tightens the formulation and results in a smaller integrality gap between the optimal objective value of the problem and the optimal objective value of its linear programming relaxation. We test the usefulness of these inequalities in computational experiments using real and randomly generated networks of practical size. Our results show that valid inequalities considerably reduce the integrality gap, from an average of 10% to an average of under 1%. These results indicate that strong cutting planes combined with branch and bound can provide efficient algorithms for solving real-world problems in the telecommunication industry.

This paper is organized as follows. In Section 2, we introduce our terminology, describe the problem and provide a literature review. In Section 3, we describe a mixed integer programming formulation of the problem and discuss the motivation for studying "distinct" and "integrated" versions of the problem. We show that the problem is NP-complete in the strong sense and compare our problem to some existing problems from the literature in Section 4. We discuss some model preprocessing techniques to reduce the problem size and obtain a stronger linear programming formulation in Section 5. In Section 6, we identify several classes of valid inequalities and show that some of them are facet-defining in the space of the capacity design variables under appropriate conditions. In Section 7 and 8, we report on our computational study. Section 9 presents a brief summary of our results.

2. Introduction and literature review

2.1. Terminology

To describe the problem, we first introduce some terminology. We express the traffic and demands in units of DS3 (Digital Signal Level 3, an industry standard) and the capacities in units of *OC-N* (Optical Carrier level N). The most commonly used OC-N levels are OC-1, OC-3, OC-12, OC-24 and OC-48. One unit of OC-N channel has a capacity of (N*51.84) Mbps and accommodates N units of DS3 signals. In a restoration network, some nodes (the so-called DCS nodes) are equipped with DCS facilities (a Digital Cross-connect Switch) that can connect the traffic passing through that node. It is able to reroute traffic by reconfiguring the DCS connections. A *span* is a collection of all the parallel point-to-point OC-N links, working and spare, between two DCS nodes. A *working link* is any link that is part of a path bearing live traffic. A *spare link* is an equipped-but-idle OC-N link that is required for dynamic network restoration when a span in the network fails. Every span consists of both working and spare links. When a span fails, the network loses both the working and spare links. In this paper, we use the terms span and edge interchangeably. For convenience, we refer to the network defined by the working links as the *base network*, and the network defined by the spare links as the *reserve network.*

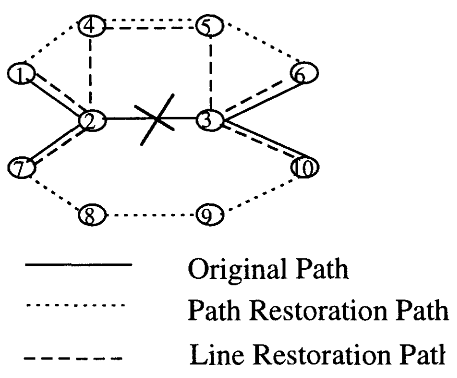


Figure 2.1: Line restoration and path restoration

Planners consider two categories of restoration schemes: *line restoration* (also known as local rerouting) and *path restoration* (also known as global rerouting). In line restoration, when an edge fails, the system redirects all the flow on that edge to an alternative route that connects the two ends of the failed edge (See Wu [25] and Wu [26]). In path restoration, the system separately restores each path passing through the faulty edge, redirecting the flows on each of these paths. Moreover, the system can use the capacity in the base network that is released by the interrupted flows. Figure 2.1 illustrates path restoration and line restoration when an edge (here edge {2,3}) fails. In the normal operating condition, one unit of demand flows from node 1 to node 6 via nodes 2 and 3, and one unit of demand flows from node 7 to node 10 via nodes 2 and 3. When edge {2,3} fails, the line restoration scheme reroutes the two units of working flow on edge {2,3} through the alternate route 2-4-5-3. The path restoration scheme redirects the one unit of demand from node 1 to 6 via the alternate route 1-4-5-6, and other unit of demand from node 7 to 10 via route 7-8-9-10.

In general, path restoration requires less spare capacity. Line restoration, however, provides other advantages: it requires less information storage and performs less complex rerouting operations at the time of a failure. For path restoration, the system must separately consider and restore each path passing through the failed edge. Consequently, path restoration requires more complex on-line algorithms and usually takes longer to perform than does line restoration.

2.2. Problem description

We wish to model and solve the following network restoration problem: what is the minimum cost assignment of spare capacity to each edge in the network so that if any edge fails, the network still has sufficient spare capacity to accommodate all of the traffic in the remaining operating network? We assume that on each edge e the network has a current flow of d_e which we refer to as the edge's working flow or demand. We assume that the operating system can repair any failure sufficiently fast that we need to consider only single edge failures. The network restores traffic using a line restoration scheme, that is, the interrupted traffic creates a local demand between the end points of the failed edge, and the system routes this demand from one end point to the other using spare capacity elsewhere in the network (not on the failed edge itself). We use one type of facility with fixed capacity b (i.e., OC-b) and allow bifurcation of the rerouted flows, i.e., the restoration flow may follow multiple routes.

Several variants of the network restoration problem are of practical interest. The network capacity might be of a single type (OC-1, OC-3 or OC-12) or of multiple types. We focus primarily on the single facility case (see Wang [24] for the two facility case). As we noted before, the system might execute path restoration or line restoration. The base network might be given, or we might simultaneously design the base and reserve networks. In practice, planners might wish to impose additional constraints on the rerouted flow. For instance, they might restrict the number of alternate paths used for rerouting any edge or path flow. They might also include "hop limits" that impose an upper bound on the number of edges on which rerouted traffic can travel. In this analysis, we do not consider these additional constraints on the rerouted flow.

2.3. Previous contributions

Several researchers have previously examined the network restoration problem, using different models and assumptions. Veerasamy, Venkatesan, and Shah [23] present a heuristic scheme for solving the path restoration problem. Their method considers faulty edges in the base network one at a time. For each path passing through that edge, they find an alternate minimum cost path and allocate sufficient additional spare capacity over previously installed spare capacity on the alternate path. The order for choosing the edges can influence the spare network design. They implement various versions of the method that differ in the order for choosing edges. They also discuss the advantages of path restoration compared to line restoration.

Sakauchi, Nishimura, and Hasegawa [21] propose a cutset formulation (see Section 6.2) for solving the line restoration problem. They solve the linear programming relaxation of the cutset formulation by iteratively generating constraints and adding them to the formulation. Finally, they round up the linear programming relaxation solutions to the nearest integers and execute the drop procedure. That is, they select edges sequentially and decrease their spare capacity by 1 unit. They either retain the new solution if it is feasible or restore the old solution if the new one is infeasible. The authors solve an example network (we refer to it as the SNH test problem in this paper) using this method. Our computational results show that their heuristic solutions are in fact optimal for this test problem. Herzberg [11] shows how to exploit the existence of two simple subgraphs, the triangle and the triangular pyramid subgraphs, to enhance the linear program solved by Sakauchi et al. [21].

Grover, Billodeau, and Venables [10] describe a heuristic to find a near optimal assignment of spare capacity in networks when the traffic on any failed edge can be rerouted on the k-shortest edge disjoint alternate paths (with respect to edge distances). The algorithm works well on the SNH test problem. Herzberg, Bye, and Utano [13], Chujo, Komine, Miyazaki, Ogura, and Soejima [6] include "hop limit" constraints that impose an upper bound on the number of edges on which rerouted traffic can travel.

Recently, Bienstock and Muratore [4] examined a model that arises in a different setting: if a vertex or edge fails, at least some prescribed fraction of each demand must be rerouted. They present several classes of facet-defining inequalities to strengthen cutset-type inequalities for the problem, some of which are similar to those presented in Magnanti and Wang [20] and in this paper.

Kennington and Whitler [14] consider the spare capacity allocation problem using only OC-1 facility in a mesh SONET telecommunications network. They present a node-arc formulation and develop a decomposition algorithm that iterates between a pure integer master problem and a set of minimum cost network flow subproblems.

Lisser, Sarkissian, and Vial [15] study the dimensioning of the spare network and the rerouting of traffic using a global rerouting scheme. They formulate the problem as a huge linear programming problem, and solve it using a cutting plane algorithm based on the concept of an analytic center. Lisser, Sarkissian, and Vial [16] consider an integrated problem of simultaneously designing the base and reserve networks. They assume local rerouting of traffic and propose a two-step procedure. The first step determines the base traffic and the spare capacity using a local rerouting strategy. The second step computes the necessary spare capacity to secure the base traffic using a global rerouting strategy.

Stoer and Dahl [22] study an integrated planning problem to decide what spans to install in the network and what capacities to install on these spans (including both the working and spare capacity) so that the network allows routing of point-to-point traffic when a single node or edge fails. This model can address more general failure situations as well, such as the simultaneous failure of two or more spans. One feature of their approach is that they split the integer variables into sums of 0-1 variables. As a result, the inequalities they obtain have a combinatorial flavor. They study the projection of the formulation onto the space of the design variables because the problems under consideration do not have flow costs.

3. Mathematical formulation

3.1. A mixed integer programming model

Let $G = (N, E)$ denote an underlying undirected graph with a set N of nodes and a set E of edges. Let $\bar{E}$ be the set of edges that require protection. For each failed edge $e \in \bar{E} \subseteq E$, d_e is the given working traffic on edge e. The failure of edge e interrupts its d_e units of working flow and creates a demand of d_e between the end points of the failed edge e. Therefore, we also refer to d_e as the demand on edge e. Since the direction of the working traffic on e is not important, we arbitrarily assign one end of edge e to be the origin $\mathrm{O}(e)$ and the other end to be the destination $\mathrm{D}(e)$. Let f_{ij}^e and f_{ji}^e be the amount of restoration flow for the failed edge e routed on edge $\{i, j\}$ from node i to node j, and from node j to node i. Let b be the capacity of each unit of transmission equipment: $b = N$ in the OC-N case. Let β_{ij} denote the idle capacity in the working links from node i to node j, i.e., the amount of capacity that is currently not used for flowing working traffic. For convenience, we assume that the network currently contains no spare links on any edge. We define a set of decision variables y_{ij} to be the number of transmission facilities assigned to edge $\{i, j\}$ for spare capacity. Using this notation, we have the following model.

Formulation (*SFLR*)

$$\text{Minimize} \sum_{\{i,j\} \in E} c_{ij} y_{ij} \tag{3.1}$$

subject to

$$\sum_{j \in N} f_{ij}^e - \sum_{j \in N} f_{ji}^e = \begin{cases} d_e & \text{if } i = O(e) \\ -d_e & \text{if } i = D(e) \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } i \in N, e \in \bar{E} \tag{3.2}$$

$$f_{ij}^e + f_{ji}^e \leq by_{ij} + \beta_{ij} \text{ for all } \{i,j\} \in E, e \in \bar{E} \tag{3.3}$$

$$f_{ij}^e, f_{ji}^e = 0 \text{ if } \{i,j\} = e \text{ for all } \{i,j\} \in E, e \in \bar{E} \tag{3.4}$$

$$f_{ij}^e, f_{ji}^e \geq 0 \text{ if } \{i,j\} \neq e \text{ for all } \{i,j\} \in E, e \in \bar{E} \tag{3.5}$$

$$y_{ij} \geq 0, y_{ij} \text{ integers for all } \{i,j\} \in E.$$

The objective coefficient c_{ij} in (3.1) which we assume is nonnegative, is the cost for each unit of spare transmission equipment on the edge $\{i,j\}$. We wish to minimize the total cost of all the equipment installed in the network. We assume that once the facilities are installed, the system incurs no additional routing cost. However, we could easily incorporate routing cost into the model, for example to prevent long routes. The constraints (3.2) are flow balance constraints that ensure the rerouting of the interrupted flow and the conservation of flow at each node when edge e fails. We assume that the demands d_e are integers. The constraints (3.3) impose capacity constraints on the edges, stating that the total restoration flow for the failed edge e on edge $\{i,j\}$, in both directions, cannot exceed the total spare capacity of the transmission equipment installed on $\{i,j\}$, plus the existing excess capacity β_{ij} if any. Constraints (3.4) restrict the flow variables to be zero on the failed edge.

We restrict the capacity design variables y to be nonnegative integers and flow variables f to be nonnegative. Given a feasible integer solution y to the problem, we can always find a feasible flow solution that is integral since the problem of finding a feasible flow given capacity values y in the network is a network flow problem which always has an integral solution whenever the demands and capacities are integral (see Ahuja, Magnanti, and Orlin [1]).

3.2. Distinct and integrated spare capacity systems

Recall that β_{ij} denotes the excess capacity in the working links from node i to node j. Suppose d_{ij} is not a multiple of b but we install the working links in increments of b. Edge $\{i,j\}$ will have a residual capacity of $b - d_{ij} \bmod b$ (we set $d_{ij} \bmod b = b$ if d_{ij} is a multiple of b). In our analysis, we typically assume that β_{ij} either equals the residual capacity for each edge $\{i,j\}$ (so that $\beta_{ij} < b$), or equals 0 for each edge $\{i,j\}$. In the first case, we can use the residual capacity to restore interrupted flow (a situation with an *integrated spare capacity system*). In the second case, we cannot use the residual capacity to restore interrupted flow (a situation with a *distinct spare capacity system*). The models used by Sakauchi et al. [21] and Grover et al. [10] assume an integrated spare capacity system. In this paper, we consider both systems because they model situations with ATM

(Asynchronous Transfer Mode) and STM (Synchronous Transfer Mode) technologies.

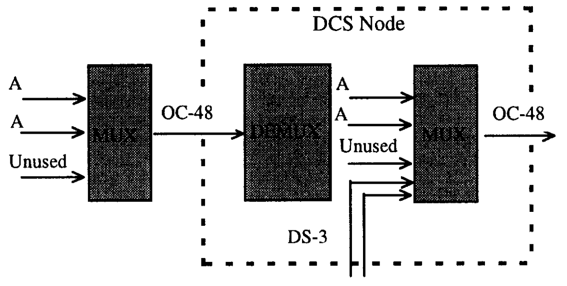


Figure 3.1: An OC-48 link demultiplexed and multiplexed to add DS3 traffic

For STM networks, the system prefers to maintain the working links and spare links separately because the multiplexers (MUX) and demultiplexers (DEMUX) are the most expensive network components. Demultiplexing and multiplexing DS3s into high levels of the digital hierarchy (for example OC-48) is expensive. At a certain DCS node of the network, suppose an incoming transmission link at the OC-48 level arrives with unused capacity. To add local DS3 signals to the link, the system must demultiplex and multiplex traffic as shown in Figure 3.1. If this link carries just pass through traffic (without the need for dropping any local traffic), the system prefers multiplexing the local add signals onto a brand new OC-48 transmission link, rather than demultiplexing the pass-through link and multiplexing the signals onto any unused capacity there. This practice saves on the number of multiplexers and demultiplexers, but results in lower "fill factor" for the STM transmission links. Generally, STM telecommunication systems utilize only about 80 percent of the DS3 capacity when using OC-12 transmissions ("Broadband Network Restoration," *IEEE Communications Magazine*, July 1996). We use the distinct system to model this practice and set $\beta_{ij} = 0$ for all edge $\{i, j\}$.

For the purposes of cell switching, the ATM networks demultiplex all the incoming links at a node even if it is just passing through. Therefore the network can multiplex the local add signals onto the unused capacity on any transmission link. This practice results in the effective use of capacity on the transmission links for restoration. We use the integrated system to model this practice and set the residual capacity $\beta_{ij} = b - d_{ij} \bmod b$ for all edges $\{i, j\}$. Note that if $b = 1$, then $\beta_{ij} = 0$ for all $\{i, j\}$ and the two systems are the same.

4. Problem complexity

4.1. *SFLR* is NP-complete in the strong sense

We prove that the *SFLR* problem is NP-complete by transforming any instance of the Hamilton Circuit (*HC*) problem to an instance of the *SFLR* problem. The strong NP-completeness result follows because the instances of the *SFLR* problem created by the transformation satisfy the similarity assumption, that is, the data is polynomially bounded in the number of nodes. Therefore, no pseudo-polynomial time algorithm will solve the problem unless P = NP. In fact, *SFLR* is NP-complete even when the costs are all the same and $b = 1$.

Proposition 4.1. *SFLR is NP-complete in the strong sense, even for the class of problems with equal costs on the edges and* $b = 1$.

Proof. See Appendix A.

4.2. Comparison to some existing problems

In the 1960's, Gomory and Hu [7] studied the minimum network synthesis problem with single commodity non-simultaneous flow requirements. For this problem, we are given a set of flow requirements, each specified by an origin and destination pair $\{k, l\}$ and a corresponding requested flow value r^{kl}. The problem is to assign sufficient capacity on each edge of the network to accommodate the flow requirements one at a time, at the minimum possible cost. If the costs on the edges are all the same, the problem is to build a network using the least possible total capacity. Gomory and Hu [8] showed how to obtain an optimal solution to the special case with uniform costs in polynomial time by decomposing the requirements into subtrees of uniform requirements. Our model also has non-simultaneous flow requirements, but it has a set of complicating constraints (3.4). The NP-completeness proof shows that *SFLR* is NP-complete even in the unit cost case. The introduction of a single variation to the underlying problem, namely, the complicating constraints (3.4), causes the problem to become much harder to solve.

The network loading problem (*NLP*) has attracted much research attention in recent years (see Bienstock and Günlük [3], Brockmüller, Günlük, and Wolsey [5], Magnanti, Mirchandani, and Vachani [18], Magnanti and Mirchandani [17] and Magnanti, Mirchandani, and Vachani [19]). NLP is a specialized network design problem which seeks to meet prescribed point-to-point demand between pairs of nodes of a network by installing capacitated facilities on the edges. Because the demands must be met simultaneously, this problem has simultaneous flow requirements. For situations with a single facility type $b = 1$, NLP is solvable by solving shortest path problems for each demand separately. Even with extra constraints stating that the flow on the OD (origin-destination) edge itself must be zero, the NLP problem can still be decomposed into shortest path problems for each OD edge (while setting the cost of the OD edge to be infinity). In this sense, non-simultaneous flow requirements are more difficult than the simultaneous flow requirements since the restoration problem is NP-complete even when $b = 1$.

5. Model preprocessing

In this section, we discuss some model preprocessing techniques to reduce the problem size and obtain a stronger linear programming formulation for the distinct system.

5.1. Decomposition method

A *bottleneck node* is a node whose removal would disconnect the network. Whenever the network contains a bottleneck node, we can duplicate the node and decompose the network into two or more connected subnetworks. First, we remove the bottleneck node and its incident edges, and decompose the network into connected components. Then we re-introduce copies of the bottleneck node in each component, and add the removed edges that are incident to each component. If an edge fails in one subnetwork, the rerouted traffic would never flow through any other subnetwork. If it did, we could delete the flows that crossed the bottleneck node and obtain a valid re-routing using less spare capacity. Therefore, if the network decomposes into r connected components, the restoration for each component would only use edges in that component. As a result, we can solve the restoration problems for these r components separately, and combine the solutions.

5.2. Removal of triangles

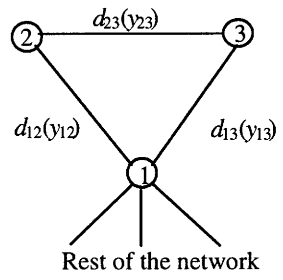


Figure 5.1: Removal of the triangle subnetwork

As a special case of the bottleneck situation, if the network contains a triangle subnetwork as shown in Figure 5.1, we can decompose the problem into two components: a triangle and the rest of the network, with node 1 duplicated in both the triangle and the rest of the network. For the triangle subnetwork shown in Figure 5.1, we have the following relations:

$$\begin{aligned} by_{13} + \beta_{13} &\geq Max(d_{12}, d_{23}), \\ by_{23} + \beta_{23} &\geq Max(d_{12}, d_{13}), \\ by_{12} + \beta_{12} &\geq Max(d_{13}, d_{23}). \end{aligned}$$

We can solve the triangle exactly by setting $y_{13} = \max\{\lceil(\max(d_{12}, d_{23}) - \beta_{13})/b\rceil, 0\}$, $y_{23} = \max\{\lceil(\max(d_{12}, d_{13}) - \beta_{23})/b\rceil, 0\}$ and $y_{12} = \max\{\lceil(\max(d_{13}, d_{23}) - \beta_{12})/b\rceil, 0\}$. Practical problems appear to often have triangle subnetworks. By removing the triangles, we have been able to reduce the size of one of our test problems by more than one third (see Section 8).

5.3. Round-up-and-scale-down method for the distinct system

Consider situations when $b > 1$. In a distinct spare capacity system, we cannot use any of the residual capacity in the base network, that is $\beta_{ij} = 0$ for all $\{i, j\}$. In these situations, we can first round the working traffic d_e on any edge e up to the nearest integer multiple of b. We claim that there is an one-to-one correspondence between the solutions to our original problem and to the rounded problem. Suppose $(\bar{y}, \bar{f})$ is a feasible solution to the original problem, and $(\hat{y}, \hat{f})$ is a feasible solution to the rounded problem. In the rounded problem, the working flow on any edge is at least the amount in the original network. Therefore, $\hat{y}$

provides sufficient spare capacity for the original problem. Now given $(\bar{y}, \bar{f})$, we want to show that $\bar{y}$ provides sufficient spare capacity for the rounded problem. Consider any edge e in the network defined by $\bar{y}$. Suppose that we can send a maximum flow of g units from O(e) to D(e) (not on edge e itself). Obviously, g is a multiple of b since the spare capacity on each edge is a multiple of b. Since g must be at least d_e, and consequently at least the nearest multiple of d_e, $\bar{y}$ provides sufficient spare capacity in the rounded problem.

Since the rounded working traffic on all edges is a multiple of b and our model has no flow cost, we can scale down the rounded working traffic by b and set $b = 1$ (that is, we now measure all flows and demands in units of b). We refer to this transformation as the *round-up-and-scale-down* method. The revised model has a stronger linear programming relaxation than the original model.

6. Polyhedral results

In this section, we discuss polyhedral results, derive six families of valid inequalities, and show that some inequalities are facet defining in the subspace of capacity design variables y.

6.1. Arc residual capacity inequality

If we relax the flow balance constraints by associating a Lagrangian multiplier v_i^e with the mass balance constraint (3.2) for node i when restoring the traffic on edge e, we have the following Lagrangian relaxation.

Formulation (LAG)

Minimize $\sum_{\{i,j\}\in E} \{c_{ij}y_{ij} + \sum_{e\in\bar{E}} (f_{ji}^e - f_{ij}^e)(v_i^e - v_j^e)\} + \sum_{e\in\bar{E}} v_{D(e)}^e d_e$

subject to

$$f_{ij}^e + f_{ji}^e \leq by_{ij} + \beta_{ij} \text{ for all } \{i,j\} \in E, e \in \bar{E} \tag{6.1}$$

$$f_{ij}^e + f_{ji}^e \leq d_e \text{ for all } \{i,j\} \in E, e \in \bar{E} \tag{6.2}$$

$$f_{ij}^e, f_{ji}^e \begin{cases} \geq 0 \text{ if } \{i,j\} \neq e \\ = 0 \text{ if } \{i,j\} = e \end{cases} \text{ for all } \{i,j\} \in E, e \in \bar{E}$$

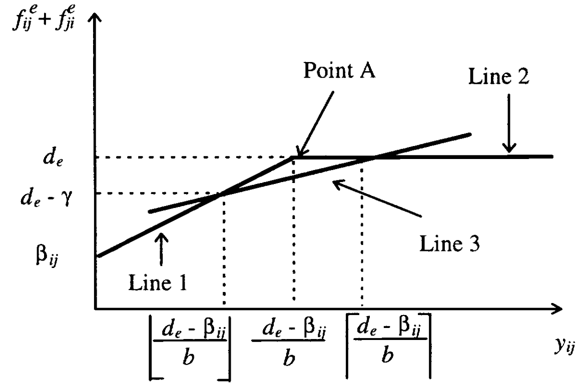
$$y_{ij} \geq 0, \text{and integers for all } \{i,j\} \in E.$$

Before forming the Lagrangian relaxation, we have added the redundant inequalities (6.2) to the formulation. For any given set of Lagrangian multipliers, problem LAG decomposes into separate subproblems, one for each pair of $\{i,j\}$ and e. The subproblem does not satisfy the integrality property (that is, its linear programming relaxation does not necessarily have an integral solution). To tighten this model, we introduce the *arc residual capacity inequality*

$$f_{ij}^e + f_{ji}^e - \gamma y_{ij} \leq d_e - \mu\gamma. \tag{6.3}$$

In this expression, $\gamma = (d_e - \beta_{ij}) \bmod b$, and $\mu = \lceil (d_e - \beta_{ij})/b \rceil$.

Figure 6.1 illustrates the effect of the arc residual capacity inequality. In this figure, line 1 corresponds to the inequality (6.1), line 2 corresponds to the inequality (6.2), and line 3 corresponds to the arc residual capacity inequality (6.3). The arc residual capacity inequality is designed to cut away fractional point A with $f_{ij}^e + f_{ji}^e = d_e$ and $y_{ij} = (d_e - \beta_{ij})/b$.



$f_{ij}^e + f_{ji}^e$
Point A
Line 2
d_e
$d_e - \gamma$
β_{ij}
Line 3
Line 1
$\left\lfloor \frac{d_e - \beta_{ij}}{b} \right\rfloor$
$\frac{d_e - \beta_{ij}}{b}$
$\left\lceil \frac{d_e - \beta_{ij}}{b} \right\rceil$
y_{ij}

Figure 6.1: Arc residual capacity inequality

Remark. Inequality (6.3) is a single commodity specialization of the valid inequalities discussed in Magnanti et al. [18]. While their Network Loading Problem has exponential number of arc residual capacity constraints, our model has $|E| * |\bar{E}|$ such constraints. As they showed, by adding the arc residual capacity inequalities, the Lagrangian subproblem satisfies the integrality property. Therefore, the strengthened linear programming formulation always approximates the value of the mixed integer program at least as well as the Lagrangian relaxation bound.

6.2. Strengthened cutset capacity constraints

Sakauchi et al. [21] use a cutset formulation consisting of only the *cutset capacity constraints*

$$\sum_{u \in W \setminus \{e\}} (by_u + \beta_u) \geq d_e \text{ for all } e \in W \text{ and all cutset } W. \tag{6.4}$$

A simple application of the max-flow-min-cut theorem shows that the cutset formulation and the flow formulation (either with or without integrality conditions imposed upon the y variables) are equivalent (in the space of the design variable y). The cutset formulation contains an exponential number of constraints while the flow formulation has a polynomial number of constraints (as a function of the number of edges and nodes). Therefore, the flow formulation is a compact formulation that is equivalent to the exponentially-sized cutset formulation.

Letting $y_{W\setminus\{e\}} = \sum_{u \in W\setminus\{e\}} y_u$ and $\beta_{W\setminus\{e\}} = \sum_{u \in W\setminus\{e\}} \beta_u$, we can rewrite the cutset capacity constraint (6.4) as

$$by_{W\setminus\{e\}} + \beta_{W\setminus\{e\}} \geq d_e,$$

or

$$y_{W\setminus\{e\}} \geq (d_e - \beta_{W\setminus\{e\}})/b.$$

Since y are integers, we can round up the righthand side and obtain the following *strengthened cutset capacity constraints*

$$y_{W\setminus\{e\}} \geq \max\{\lceil (d_e - \beta_{W\setminus\{e\}})/b \rceil, 0\} \equiv \text{Req}(W, e) \text{ for each edge } e \in W. \tag{6.5}$$

The term $\text{Req}(W, e)$ denotes the requirements in cutset W with respect to edge e.

We say a subnetwork G' is *restorable* if for any failure edge in G', the subnetwork G' contains a path from one end of the edge to the other (not the failed edge itself) on which we can install spare capacity.

Theorem 6.1. *Given a cutset W containing edge e, the strengthened cutset capacity inequality (6.5) is facet defining in the space of the y variables if the two subnetworks formed by removing the edge set W are each restorable.*

Proof. See Appendix A.

Remark. For a cardinality-2 cut W, say $W = \{1,2\}$, the strengthened cutset capacity inequalities are the same as the arc residual capacity inequalities. The strengthened cutset capacity inequality for edge 2 is given by

$$y_1 \geq \lceil (d_2 - \beta_1)/b \rceil .$$

The arc residual capacity inequality is given by

$$f_1^2 - \gamma y_1 \leq d_2 - \mu\gamma$$

with $\gamma = (d_2 - \beta_1) \bmod b$, and $\mu = \lceil (d_2 - \beta_1)/b \rceil$. Since $f_1^2 = d_2$, we have $\gamma y_1 \geq \mu\gamma$, and consequently $y_1 \geq \mu = \lceil (d_2 - \beta_1)/b \rceil$.

6.3. Cardinality-k cutset inequality

We define a *cardinality-k cut* to be any cut K in the graph G consisting of k edges. Let $K = \{1, 2, ..., k\}$ be the index set for the edges in the cut. Adding the k strengthened cutset capacity constraints for any edge $e \in K$

$$y_{K \setminus \{e\}} \geq \text{Req}(K, e),$$

we obtain

$$(k-1)y_K \geq \sum_{e \in K} \text{Req}(K, e).$$

In this expression, $y_K = \sum_{e \in K} y_e$. We let $\text{Req}(K) = \sum_{e \in K} \text{Req}(K, e)$ denote the total requirement in the cutset K. We then divide both sides of the inequality by $(k-1)$ and round up the righthand side to obtain the *cardinality-k cutset inequality*

$$y_K \geq \lceil \text{Req}(K)/(k-1) \rceil . \tag{6.6}$$

Theorem 6.2. *The cardinality-k cutset inequality (6.6) for $k \geq 3$ defines a facet of the convex hull of the feasible solutions to SFLR in the subspace of y variables, if*

(i) the two subnetworks defined by removing the edges in the cutset K are each restorable;

(ii) $Req(K)$ is not a multiple of $(k-1)$; and

(iii) $Req(K,e) < \lceil Req(K)/(k-1) \rceil$ for all $e \in K$.

Proof. See Appendix A.

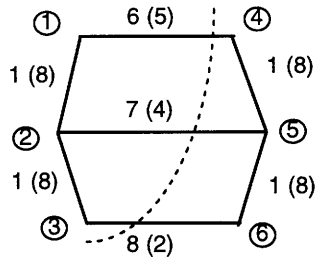


Figure 6.2: Example 1

We note that the condition (i) in Theorem 6.2 is sufficient, but not necessary. Consider the cardinality-3 cutset K consisting of edges $\{1,4\}$, $\{2,5\}$ and $\{3,6\}$ in Figure 6.2 and $b = 1$. The numbers next to the edges are demands and the numbers in parentheses specify a feasible solution. If we remove the edges in the cutset K, the two subnetworks are not restorable. For example, we cannot restore edge $\{1,2\}$ since the subnetwork contains no path from node 1 to node 2 (except the edge $\{1,2\}$). The total demand in cutset K is 21, so $\lceil \text{Req}(K)/(k-1) \rceil = 11$. As required by the cardinality-k cutset inequalities, we allocate a total of 11 units of capacity in the cutset K with 5 units on edge $\{1,4\}$, 4 units on edge $\{2,5\}$ and 2 units on edge $\{3,6\}$. We then assign a capacity of 8 units (equal to the largest demand in the network) to all the other edges in the network. Since the edges in the subnetworks have very small demands compared to the spare capacity allocated in the cutset K, we can restore these edges by sending the rerouted flow across the cutset K and back without the need to allocate any additional capacity in K. For example, we restore edge $\{1,2\}$ by routing the 1 unit of flow on edges $\{1,4\}$, $\{4,5\}$ and $\{5,2\}$. In this case, the cardinality-3 inequality is facet-defining although the problem data does not satisfy condition (i).

6.4. p-partition inequality

One way to view the cardinality-k inequalities is in terms of network aggregation. We aggregate the network into two aggregate nodes S and T and examine the valid inequalities for the edge set joining S and T. Suppose we aggregate the network into p aggregate nodes S_1, S_2..., S_p. For any edge $\{i,j\}$ in the original network, with i, j contained in the different node sets, e.g., $i \in S_1$ and $j \in S_2$, we include an edge between nodes $S_1$and S_2. Let K_1, K_2..., K_p denote the edge sets incident to any node in the aggregate network, and let k_1, k_2 and k_3 be their cardinality. Let y_{K_i} denote the total capacity in set K_i, and let $\mathrm{Req}(K_i)$ denote the total requirement in cut K_i. The cardinality-k inequality for cutset K_i is given by

$$y_{K_i} \geq \lceil \mathrm{Req}(K_i)/(k_i - 1) \rceil .$$

Adding these inequalities and dividing both sides by two, we obtain the *p-partition inequality*

$$1/2 \sum_{1 \leq i \leq p} y_{K_i} \geq \left\lceil 1/2 \sum_{1 \leq i \leq p} \lceil \mathrm{Req}(K_i)/(k_i - 1) \rceil \right\rceil . \tag{6.7}$$

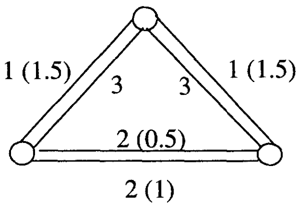


Figure 6.3: Example 2, a 3-partition

We use the example in Figure 6.3 to illustrate the p-partition inequality for $p = 3$. Consider a three node aggregate network with $k_1 = k_2 = k_3 = 4$ and $b = 1$. The numbers next to the edges are the demands and the numbers in parentheses specify a linear programming relaxation solution. We calculate the cardinality-k inequalities as follows:

$$\begin{aligned} y_{K_1} &\geq \lceil 8/3 \rceil = 3 \text{ for cutset } K_1; \\ y_{K_2} &\geq \lceil 8/3 \rceil = 3 \text{ for cutset } K_2; \\ y_{K_3} &\geq \lceil 8/3 \rceil = 3 \text{ for cutset } K_3. \end{aligned}$$

The linear programming relaxation solution shown in Figure 6.3 is fractional with a total capacity 4.5. It satisfies the previous three inequalities as an equality. We can cut away this fractional solution by adding the 3-partition inequality

$$1/2(y_{K_1} + y_{K_2} + y_{K_3}) \geq \lceil (\lceil 8/3 \rceil + \lceil 8/3 \rceil + \lceil 8/3 \rceil)/2 \rceil = 5.$$

6.5. $K - C$ cutset inequality

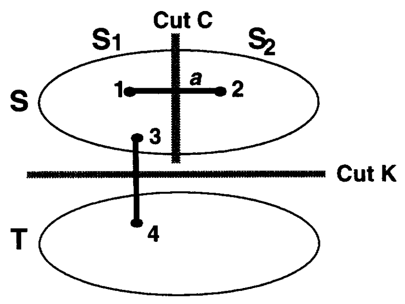


Figure 6.4: $K - C$ cutset configuration

In Figure 6.4, a cutset K of cardinality k separates node sets S and T. S contains two designated nodes 1 and 2. Let $S_1$and S_2 define a partition of node set S that separates nodes 1 and 2. That is, $S = S_1 \cup S_2$, $1 \in S_1$, $2 \in S_2$ and $S_1 \cap S_2 = \emptyset$. Let C denote the set of edges across the $S_1 - S_2$ cut. Let a be the working flow on edge $\{1, 2\}$. Let $\text{Req}(K, *) = \max_{e \in K} \text{Req}(K, e)$ and let edge $\{3, 4\}$ denote the largest requirement edge (with requirement $\text{Req}(K, *)$) in cutset K. Let $\theta_K = \max\{\lceil \text{Req}(K)/(k-1) \rceil, \text{Req}(K, *)\}$.

The following $K - C$ *inequality* is valid

$$y_{C \setminus \{1,2\}} + y_K \geq \theta_K + \left\lceil \frac{a - \lfloor (b\theta_K + \beta_K)/2 \rfloor - \beta_{C \setminus \{1,2\}}}{b} \right\rceil. \tag{6.8}$$

In this inequality, $y_{C \setminus \{1,2\}}$ is the total spare capacity installed in the edge set C excluding edge $\{1, 2\}$ and y_K is the total spare capacity installed in the cutset K.

To derive inequality (6.8), first we have

$$y_K \geq \theta_K.$$

Suppose that edge {1,2} fails, and that a_1 units out of the a units of the edge $\{1,2\}$ flow cross the cutset K, and that the remaining $(a - a_1)$ units are routed within the set S. Since a_1 units must cross the cutset K at least twice, by aggregating the capacity constraints, we obtain the following inequality

$$by_K + \beta_K \geq 2a_1. \tag{6.9}$$

Since $(a - a_1)$ units must cross the edge set C,

$$by_{C \setminus \{1,2\}} + \beta_{C \setminus \{1,2\}} \geq a - a_1. \tag{6.10}$$

Multiplying inequality (6.10) by 2 and adding it to inequality (6.9) gives us

$$2by_{C\setminus\{1,2\}} + by_K + \beta_K + 2\beta_{C\setminus\{1,2\}} \geq 2a.$$

We add $by_K + \beta_K \geq b\theta_K + \beta_K$ to this inequality and obtain

$$2by_{C\setminus\{1,2\}} + 2by_K + 2\beta_K + 2\beta_{C\setminus\{1,2\}} \geq 2a + b\theta_K + \beta_K.$$

By dividing both sides by 2 and rounding up the righthand side, we obtain

$$b(y_{C\setminus\{1,2\}} + y_K) \geq a + \lceil (b\theta_K + \beta_K)/2 \rceil - \beta_K - \beta_{C\setminus\{1,2\}}.$$

After dividing both sides by b and round up the right hand side, we obtain

$$y_{C\setminus\{1,2\}} + y_K \geq \left\lceil \frac{a + \lceil (b\theta_K + \beta_K)/2 \rceil - \beta_K - \beta_{C\setminus\{1,2\}}}{b} \right\rceil .$$

Equivalently,

$$y_{C\setminus\{1,2\}} + y_K \geq \theta_K + \left\lceil \frac{a - \lfloor (b\theta_K + \beta_K)/2 \rfloor - \beta_{C\setminus\{1,2\}}}{b} \right\rceil .$$

Theorem 6.3. *The $K - C$ inequality (6.8) defines a facet of the convex hull of feasible solutions to the SFLR in the subspace of y variables if*

(i) $a \geq \lfloor (b\theta_K + b + \beta_K)/2 \rfloor + \beta_{C\setminus\{1,2\}}$*;*

(ii) Req(K) is not a multiple of $(k-1)$ if $\theta_K = \lceil Req(K)/(k-1) \rceil$;

(iii) θ_K is odd if $b = 1$;

(iv) the subgraphs induced by T, S_1 and S_2 are each restorable; and

(v) the edge set C contains at least two edges (including edge {1,2}); the cutset K contains at least one edge connecting S_1 and T (if $\theta_K = Req(K,*)$, then it contains an edge besides {3,4}), and at least one edge connecting S_2 and T.

Proof. See Wang [24].

When $b = 1$, then $\beta_{ij} = 0$ for any edge $\{i,j\}$ and the $K - C$ inequality (6.8) becomes

$$y_{C\setminus\{1,2\}} + y_K \geq a + \lceil \theta_K/2 \rceil . \qquad (6.11)$$

Notice that if the total spare capacity in the cutset K attains its lower bound θ_K and θ_K is odd, then at least one unit of that spare capacity cannot be used to restore edge $\{1,2\}$ since the restoration flow for edge $\{1,2\}$ must cross the cutset K even number of times. The solution to the linear programming relaxation might, however, send the restoration flows in fractional units and fully utilize the spare capacity in the cutset K. The inequality (6.11) cuts away such fractional solutions. An alternate way to eliminate these fractional solutions is to generate the following Gomory cut in the full f-y space.

$$(y_K - f_K^{1,2}) + (y_K - \theta_K) \geq 1. \tag{6.12}$$

In this expression, $f_K^{1,2}$ denotes the total restoration flow for failed edge $\{1,2\}$ in the cutset K. Therefore

$$y_K - f_K^{1,2} \geq 0. \tag{6.13}$$

We also have the lower bound inequality

$$y_K - \theta_K \geq 0. \tag{6.14}$$

Adding (6.13) and (6.14) gives

$$(y_K - f_K^{1,2}) + (y_K - \theta_K) \geq 0.$$

However, (6.13) and (6.14) cannot be both zero, since θ_K is odd by assumption and $f_K^{1,2}$ is even. Thus, we have

$$(y_K - f_K^{1,2}) + (y_K - \theta_K) \geq 1.$$

We now show how to transform the Gomory cut (6.12) to $K-C$ inequality (6.11). Moving $f_K^{1,2}$ and θ_K to the righthand side of (6.12) gives

$$2y_K \geq \theta_K + 1 + f_K^{1,2}.$$

Multiplying inequality $y_{C\setminus\{1,2\}} \geq a - a_1$ by 2 and adding it to this inequality gives

$$2y_K + 2y_{C\setminus\{1,2\}} \geq \theta_K + 1 + f_K^{1,2} + 2(a - a_1).$$

Since $f_K^{1,2} \geq 2a_1$, we have

$$2y_K + 2y_{C\setminus\{1,2\}} \geq \theta_K + 1 + 2a.$$

Dividing both sides by 2 gives (6.11).

Note that the Gomory cut (6.12) involves only cutset K. The advantage of this form is that it does not require the identification of edge set C, only the identification of edge $\{1,2\}$. Therefore inequality (6.12) is easier to work with in the separation algorithm (see Section 7).

6.6. Subset-Q inequalities

Given any cardinality-k cutset K and any subset $Q \subseteq K$, we define $\text{Req}(Q)$ to be the total requirement in subset Q and $\text{Req}(Q) = \sum_{e \in Q \subseteq K} \text{Req}(K, e)$. Let $r_Q = \text{Req}(Q) \bmod (q-1)$. Magnanti and Wang [20] derive the following *subset-Q inequalities*

$$r_Q y_Q + (r_Q + 1) y_{\bar{Q}} \geq r_Q \left\lceil \text{Req}(Q)/(q-1) \right\rceil. \tag{6.15}$$

Let $A = \text{Req}(K)/(k-1)$ denote a modified average of the total requirement on the edges of K.

Theorem 6.4. *When Q is a proper subset of the cutset K, the subset-Q inequality (6.15) defines a facet for the network restoration problem in the space of the capacity variables if*

(i) The two subnetworks formed by removing the cutset K are each restorable;

(ii) $Req(Q)$ is not a multiple of $(q-1)$;

(iii) $Req(K, e) \leq \lfloor Req(Q)/(q-1) \rfloor$ for all edges $e \in K$; and

(iv) $A < \lfloor Req(Q)/(q-1) \rfloor$.

Proof. See Wang [24]. Magnanti and Wang [20] prove this theorem for $b = 1$.

Given any cardinality-k cutset K, the following theorem characterizes the convex hull of set $Y = \{y \in Z_k^+ \mid \sum_{u \in K \setminus \{e\}} (by_u + \beta_u) \geq d_e \text{ for all edge } e \in K\}$.

Theorem 6.5. *Given any cardinality-k cutset K, the following constraints completely describe the convex hull of Y:*

(i) The strengthened cutset capacity constraints $\sum_{j \in K \setminus \{e\}} y_j \geq Req(K, e)$ for all edges $e \in K$;

(ii) The subset-Q inequalities $r_Q y_Q + (r_Q + 1) y_{\bar{Q}} \geq r_Q \lceil Req(Q)/(q-1) \rceil$ for all subsets Q of K (including K); and

(iii) The nonnegativity constraints.

Proof. See Wang [24]. Magnanti and Wang [20] prove this theorem for $b = 1$.

Bienstock et al. [4] have independently studied a similar polyhedron. The primary difference is that they consider only the $b = 1$ case and that their righthand sides for the cutset capacity constraints in (i) of Theorem 6.5 are some positive integer D for all $e \in K$, while we allow the righthand sides to be different for each cutset capacity constraint. Therefore, their polyhedron is a special case of ours and the inequalities they identified for this polyhedron are a subset of inequalities (6.15).

7. Solution procedures

To test the effectiveness of some of the valid inequalities identified in the previous section, we used these inequalities in a cutting plane procedure. We start with the linear programming relaxation, and given a fractional solution for the current linear programming formulation, identify a valid inequality that this solution violates and append it to the current formulation. We then solve the augmented formulation and continue this procedure until we either find an integer solution or cannot identify any inequalities violated by the current fractional solution. After the separation algorithm terminates, we use the CPLEX 5.0 branch-and-bound solver to solve the integer programming problem to optimality.

Since the separation problems for the strengthened cutset capacity, cardinality-k, subset-Q, and $K - C$ cutset inequalities seem to be difficult, we use heuristics to identify violated inequalities. Given any fractional feasible solution (y, f), our heuristics examine the cutsets that have a high probability of violating the valid inequalities. We found that two types of cutsets usually have tight spare capacity. The first type are the *singleton cutsets* incident to single nodes. The second type are the *GH cutsets* obtained by solving an all-pairs minimum cut problem in the network with edge capacities y by solving $(n - 1)$ max-flow problems, and constructing a spanning tree using Gomory and Hu's algorithm (see Gomory and Hu [9]). Since the removal of any edge in the Gomory-Hu tree partitions the nodes into two sets and corresponds to a cutset in the original network, we examine $(n - 1)$ such cutsets identified by the Gomory-Hu tree.

7.1. Distinct system

We first transformed the problems using the round-up-and-scale-down method to the $b = 1$ case. We then solved the linear programming relaxation obtaining a solution (y, f). The cutting plane algorithm contains several modules. Module D1 and Module D2 examine singleton cutsets. Module D1 searches for violated cardinality-k or subset-Q inequalities. If D1 finds a violated inequality, the algorithm adds it to the linear programming formulation. The algorithm exits Module D1 if it exhausts all singleton cutsets and finds no violated cardinality-k

or subset-Q inequalities. Module D2 searches for violated $K - C$ cutset inequalities and uses the same control scheme as Module D1. We use the alternative form (6.12) which does not involves edge set C. Module D3 and Module D4 solve the all-pairs minimum cut problem first, and examine the $(n - 1)$ GH cutsets identified by the Gomory-Hu tree. If a GH cutset is also a singleton cutset, the algorithm skips it and continues with the next cutset. Module D3 searches for violated cardinality-k or subset-Q inequalities and uses the same control scheme as Module D1. Module D4 searches for violated $K - C$ cutset inequalities and uses the same control scheme as Module D1.

Algorithm Summary

Transform the problems using the round-up-and-scale-down method so that $b = 1$; Solve the linear programming relaxation obtaining a solution (y, f); if y is integral, then the solution is the integer programming optimal solution; if y is fractional, continue;

Module D1:

1. Starting with the cutset K incident to node 1, do
   (a) Does y violates the cardinality-k inequality? If so, go to step 3;
   (b) Does cutset K contain zero valued edges (edge e with $y_e = 0$)? If not, go to step 2;
   (c) Store the zero valued edges in any order in a list L;
   (d) Let Q be K excluding the edges in the list L. If y violates subset-Q inequality, go to step 3;
   (e) Delete the first edge from the list L. If L is nonempty, go back to step (d);
2. Continue with the cutset incident to the next node, returning to step (a). If we have exhausted all the nodes, exit and go to the next module;
3. Add the violated cardinality-k or subset-Q inequality, solve the resulting linear programming relaxation obtaining a new solution (y, f), and return to step 1.

Module D2:

1. Solve the all-pairs minimum cut problem, obtaining the Gomory-Hu tree;

2. For a cutset defined by the tree with both partitions containing more than one node, do

   Repeat steps (a) to (e) from Module D1;

3. Continue with the next cutset, returning to step (a). If we have exhausted all the cutsets, exit and go to the next module;

4. Add the violated cardinality-k or subset-Q inequality, solve the resulting linear programming relaxation obtaining a new solution (y, f), and return to step 1.

Module D3:

1. Starting with the cutset K incident to node 1, do

   (a) Calculate $\theta_K = \max\{\lceil \text{Req}(K)/(k-1) \rceil, \text{Req}(K, *)\}$ for the cutset K;

   (b) If the total spare capacity in cutset K equals θ_K and θ_K is odd, then continue to step (b); otherwise, go to step 2;

   (c) For all edges $e \in \bar{E}$ not contained in K, calculate the total restoration flow f_K^e for edge e in cutset K. If $f_K^e > (\theta_K - 1)$ for some e, go to step 3;

2. Continue with the cutset incident to the next node, returning to step (a). If we have exhausted all the nodes, exit and go to the next module;

3. Add the violated $K - C$ cutset inequality, solve the resulting linear programming relaxation obtaining a new solution (y, f), and return to step 1.

Module D4:

1. Solve the all-pairs minimum cut problem, obtaining the Gomory-Hu tree;

2. For a cutset defined by the tree with both partitions containing more than one node, do

   Repeat steps (a) to (c) from Module D3;

3. Continue with the next cutset, return to step (a). If we have exhausted all the cutsets, terminate;

4. Add the violated $K - C$ cutset inequality, solve the resulting linear programming relaxation obtaining a new solution (y, f), and return to step 1.

After the cutting plane algorithm terminates, we used the CPLEX 5.0 branch-and-bound solver to solve the integer programming problem to optimality. The algorithm is coded in C programming language and linked to the CPLEX callable library. It is implemented on a Pentium Pro personal computer with the Linux environment, a 200 Mhz processor and 128 Mb RAM.

7.2. Integrated system

We added all the arc residual inequalities to the formulation after solving the initial linear programming relaxation. We used the same heuristics as in the distinct system to separate cardinality-k and subset-Q inequalities in Modules I1 and I3. Modules I2 and I4 search for violated strengthened cutset capacity constraints by examining singleton cutsets and GH cutsets and use the same control scheme as Module I1.

Algorithm Summary

Solve the linear programming relaxation obtaining a solution (y, f);
Add the arc residual capacity inequalities; solve the linear programming relaxation again, obtaining a solution (y, f);

Module I1. Same as Module D1.
Module I2.

1. Starting with the cutset K incident to node 1, do
   (a) List the edges of the cutset in any order in a list L;
   (b) Starting with the first edge in L, if the solution y violates the strengthened cutset capacity constraint with respect to this edge, go to step 3;
   (c) Remove the first edge from the list L. If L is nonempty, go back to step (b);
2. Continue with the cutset incident to the next node, returning to step (a). If we have exhausted all the nodes, exit and go to the next module;

3. Add the violated strengthened cutset capacity constraint, solve the resulting linear programming relaxation obtaining a new solution (y, f), and return to step 1.

Module I3. Same as Module D2.
Module I4.

1. Solve the all-pairs minimum cut problem, obtaining the Gomory-Hu tree;
2. For a cutset defined by the tree with both partitions containing more than one node, do

   Repeat steps (a) to (c) from Module I2;
3. Continue with the next cutset, returning to step (a). If we have exhausted all the cutsets, terminate;
4. Add the violated strengthened cutset capacity constraint, solve the resulting linear programming relaxation obtaining a new solution (y, f), and return to step 1.

After the cutting plane algorithm terminates, we used the CPLEX 5.0 branch-and-bound solver to solve the integer programming problem.

8. Computational results

8.1. Edge cost structure

Recall that the objective coefficient c_{ij} is the cost for each unit of transmission equipment on edge $\{i, j\}$. Two kinds of costs contribute to c_{ij}: the transmission system cost at the two ends of edge $\{i, j\}$ and the fiber cost which may be proportional to the length of edge $\{i, j\}$. In some cases, the fiber cables are already in place, and we need to consider only the transmission system cost at the ends of the edge. Therefore, c_{ij} has the same value on each edge and is set to be 1. We refer to this case as the *EIC* (edge independent cost) case. In other cases, we need to consider both the fiber cable and transmission system costs. We refer to this case as the *EDC* (edge dependent cost) case. We tested our algorithm for both cost structures.

8.2. Test problems

We obtained three real world problem data sets. Sakauchi et al. [21] first analyzed the SNH test problem. The other two problems (called BMSW1 and BMSW2) were provided by a telecommunications company. Table 8.1 specifies the dimension of the these problems and Appendix B shows the network topologies. For BMSW2, we could exploit the special structure of the network and remove six triangles. The remaining network has 41 nodes and 61 edges. The resulting formulation has 7,503 variables and 6,283 constraints, so the procedure reduced the problem size by about one third.

	SNH Problem	BMSW1 Problem	BMSW2 Problem
$\lvert N\rvert$	11	10	53
$\lvert E\rvert$	23	14	79
$\lvert\bar{E}\rvert$	23	14	74
Number of variables	1,081	406	11,771
Number of constraints	807	350	9,842
Average node degree	4.18	2.8	2.98

Table 8.1: Real world problem dimensions

We then used a random graph generator (see Balakrishnan et al. [2]) to obtain more test problems. The random graph generator creates a doubly-connected random graph for any specified number of nodes and edges. We created four sets of problems, R20, R30, R40, and R50, each set consisting of ten instances. The problems range from 20 to 50 nodes, 40 to 100 edges and have an average node degree of 4. The demands on each edge were integers uniformly distributed between 1 and 100. The costs for the EDC case were integers randomly assigned between 1 and 100. These random problems are representative of the size, costs, and demand structures arising in real problems. Table 8.2 summarizes the random problem data and dimensions.

	R20	R30	R40	R50
$\lvert N\rvert$	20	30	40	50
$\lvert E\rvert$	40	60	80	100
$\lvert \bar{E}\rvert$	40	60	80	100
Number of variables	3,240	7,260	12,880	20,100
Number of constraints	2,400	5,400	9,600	15,000
Average node degree	4	4	4	4
Number of instances	10	10	10	10

Table 8.2: Random problem dimensions

8.3. Distinct system

We first tested the algorithm on three real world problem data sets. Sakauchi et al. [21] obtained solutions for the SNH problem (which turned out to be optimal) by solving a linear programming formulation and rounding the linear programming solutions to integers. This example also served as a test problem for the heuristic methods in Grover et al. [10] and Herzberg [12]. The SNH problem does not contain information on the edge length. All three papers solved it using unit costs on all edges. Therefore, we tested the OC-1, OC-3 and OC-12 cases for this problem assuming edge independent costs (EIC). In the OC-3 and OC-12 cases, we used the round-up-scale-down method to transform the problem into the $b = 1$ case. The LP (linear programming) optimal values equaled the IP (integer programming) optimal values for all three cases. We identified a violated cardinality 3 inequality around node 6 for the OC-3 and OC-12 cases and a violated cardinality 4 inequality around node 4 for the OC-12 case. Adding the violated inequalities resulted in integral LP solutions, while the objective values remained the same.

For both BMSW1 and BMSW2 problems, we assumed $b = 1$ and tested both EDC and EIC cases. BMSW1 problem had integral LP solutions for both cases. BMSW2 problem had an integral LP solution for the EDC case. For the EIC case, BMSW2 had a fractional LP solution with its objective value equal the value of the IP solution. We did not find any violated inequalities. Branch and bound explored ten nodes and solved the problem to optimality in about one minute.

LP:	initial LP value
LPX:	LP value after adding all the violated cuts identified by the heuristics
IP:	optimal IP solution determined by branch-and-bound
LP gap %:	100*(IP-$\lceil$LP$\rceil$)/IP
LPX gap %:	100*(IP-$\lceil$LPX$\rceil$)/IP
# cuts:	total number of violated cuts identified by the heuristics
# Bnodes:	total number of tree nodes explored by branch-and-bound
CPU_sec:	total CPU time in seconds required to solve the problem to optimality

Instance	LP gap %	LPX gap %	# cuts	# Bnodes	CPU_sec
SNH (OC-1)	0.0	0.0	0	2	4
SNH (OC-3)	0.0	0.0	1	0	3
SNH (OC-12)	0.0	0.0	2	0	3
BMSW1 (EIC)	int	-	-	-	1
BMSW1 (EDC)	int	-	-	-	1
BMSW2 (EIC)	0.0	0.0	0	10	62
BMSW2 (EDC)	int	-	-	-	41

Table 8.3: Real world problem solution summary, distinct system

We further tested the algorithm on the four sets of random problems. We tested these problems using both the EIC and the EDC cost structures and assuming $b = 1$. Table 8.4 and Table 8.5 summarize the results for the random problems. The linear relaxation bounds are very tight. For many problem instances, the lower bounds are equal to the integer optimal values. The B&B (branch and bound) solved quickly to optimality because of the tight lower bounds. We found a small number of cuts, but the gap reduction was not dramatic. However, the cuts helped in proving optimality. For example, an EIC instance with 30 nodes had an LP value of 1850.21. Our algorithm found two cuts and raised the LP lower bound to 1851.05, rounding up to 1852. The branch and bound quickly found an integer solution with cost 1852 and proved optimality after exploring 72 tree nodes and using 35 seconds of CPU time. Without the cuts, B&B required 20,235 nodes and 266 seconds of CPU time to prove optimality, although it identified the IP solution after 13 nodes.

Batch	LP gap %	LPX gap %	# cuts	# Bnodes	CPU_sec
R20	0.0	0.0	0.3	12.9	3.6
R30	0.04	0.02	0.9	30.3	26.6
R40	0.03	0.01	0.8	57.8	87.9
R50	0.04	0.02	0.8	59.2	183.3

Table 8.4: Random instances solution summary, EIC, distinct system

Batch	LP gap %	LPX gap %	# cuts	# Bnodes	CPU_sec
R20	0.04	0.04	0.1	3.9	2.2
R30	0.08	0.07	0.5	8.2	15.7
R40	0.1	0.06	1.0	22.4	66.7
R50	0.1	0.07	1.0	29.1	141.1

Table 8.5: Random instances solution summary, EDC, distinct system

The CPU times reported in this computational study serve as a general indication of how long the problems took to reach optimality. The CPU times varied by as much as 70 percent if we ran the same problem multiple times on the same machine. Therefore, these figures provide only general information concerning how the running time changes with problem sizes, and the level of difficulty of these problems.

8.4. Integrated system

For the integrated system, we tested the three real world problems and four sets of random problems assuming $b = 12$. We solved the SNH problem using EIC cost structure, and the BMSW1 and BMSW2 problems using both EIC and EDC cost structures. Table 8.6 summarizes the results for the real world problems.

Instance	LP gap%	LPA gap%	LPX gap%	#KQcuts	#CAPcuts	#Bnodes	CPU_sec
SNH (EIC)	11.4	6.0	0.7	2	8	24	2
BMSW1 (EIC)	3.5	1.8	1.0	0	5	5	1
BMSW1 (EDC)	3.8	0.8	0.3	0	5	2	1
BMSW2 (EIC)	12.4	7.7	1.1	2	25	38	140
BMSW2 (EDC)	8.6	5.3	0.8	3	22	47	188

Table 8.6: Real world problems solution summary, OC-12, integrated system

LP:	initial LP value
LPA:	LP value after adding the arc residual capacity constraints
LPX:	LP value after adding arc residual capacity constraints and all the violated cuts identified by the heuristics
LPV:	LP value after adding all the violated cuts identified by the heuristics
IP:	optimal IP solution solved by branch-and-bound
LP gap %:	$100*(\text{IP}-\lceil\text{LP}\rceil)/\text{IP}$
LPA gap %:	$100*(\text{IP}-\lceil\text{LPA}\rceil)/\text{IP}$
LPX gap %:	$100*(\text{IP}-\lceil\text{LPX}\rceil)/\text{IP}$
LPV gap %:	$100*(\text{IP}-\lceil\text{LPV}\rceil)/\text{IP}$
# KQcuts:	total number of violated cardinality-k and subset-Q inequalities identified by the heuristics
# CAPcuts:	total number of violated strengthened cutset capacity inequalities identified by the heuristics
# Bnodes:	total number of tree nodes explored by branch-and-bound
CPU_sec:	total CPU time in seconds taken to solve the problem to optimality

We used the EDC cost structure for the random instances, obtaining the results reported in Table 8.6. The results in Table 8.6 and 8.7 show that the valid inequalities were effective in reducing the integrality gap. For example, a random instance with 50 nodes and 100 edges originally had an LP gap of 12%. Our cuts reduced the gap to 0.3%. B&B found the best integer solution and proved optimality after exploring 45 nodes. The entire process used less than four minutes of CPU time. Without the cuts, B&B took about six hours of CPU time to find an integer solution that was still 0.4% away from the optimal IP value, and managed to reduce the gap to around 10%. The B&B tree size grew to 200 mega bytes and terminated prematurely.

Batch	LP gap%	LPA gap%	LPX gap%	# KQcuts	# CAPcuts	# Bnodes	CPU_sec
R20	9.52	4.42	0.26	3.0	17.0	4.4	6.2
R30	9.02	4.82	0.48	4.4	27.6	32.2	55.8
R40	8.44	4.46	0.13	7.6	30.6	49.0	198.6
R50	8.88	5.02	0.56	9.2	37.6	98.3	295.2

Table 8.7: Random instances solution summary, OC-12, integrated system

The computational results demonstrate that for practical purposes it is adequate to examine two classes of cutsets (singleton and GH cutsets) for violated inequalities. The fact that we were able to reduce the integrality gaps to under 1% for almost all problems indicates that the benefit of examining other cutsets can only be marginal and does not justify the extra computation time.

Our computational study was also designed to identify the best way to improve the formulation. We performed the following experiment and collected information on the reduction of the gap for each class of inequalities. Recall that the algorithm added arc residual capacity constraints before switching to Module I1 to I4. For our problem instances, by adding these constraints, we increased the number of constraints by 67 percent. A larger sized LP takes longer to solve. Since the algorithm spends most of the time solving LPs, we solved the random problems again using only Model I1 to I4 (without adding the arc residual capacity constraints). Table 8.8 shows that we achieved almost exactly the same performance in total gap reduction (with negligible differences in just a few cases) compared to Table 8.7. Although we are adding more cuts, and thus solving more LPs, the savings in total CPU times are significant. As reported in Table 8.9. we then determined the total number of KQ cuts (cardinality-k and subset-Q inequalities) and CAP cuts (strengthened cutset capacity inequalities) identified by the algorithm. The algorithm found more CAP cuts than KQ cuts. The KQ cuts achieved a little less than fifty percent of the total gap reduction, while the CAP cuts achieved between 53 and 60 percent of the total gap reduction. As reported in Table 8.10, we also determined the effectiveness of examining the singleton cutsets and GH cutsets. The singleton cuts achieved almost ninety percent of the total gap reduction, while the GH cuts achieved between 8 and 11 percent of the total gap reduction. However, since we examined the GH cutsets after the singleton cutsets, the "tailing effect" might have undermined the impact of GH cutsets.

Batch	LP gap%	LPV gap%	# Bnodes	CPU_sec
R20	9.52	0.26	5.0	4.6
R30	9.02	0.48	68.2	23.4
R40	8.44	0.13	92.7	95.4
R50	8.88	0.56	106.5	126.3

Table 8.8: Random instances, OC-12, without arc residual capacity constraints

Why does the cutting plane algorithm run almost as well without arc residual capacity inequalities? Recall that the arc residual capacity constraint (6.3) is designed to cut off the fractional solutions of the form

$$\begin{cases} f_{ij}^e + f_{ji}^e = d_e \\ y_{ij} = (d_e - \beta_{ij})/b \end{cases} \text{for some } \{i,j\} \in E, e \in \bar{E}.$$

When would $f_{ij}^e + f_{ji}^e = d_e$ be most likely satisfied? For a cardinality-2 cutset consisting of $\{i,j\}$ and e, this equation certainly holds. However, recall that for a cardinality-2 cut W, the strengthened cutset capacity inequalities are the same as the arc residual capacity inequalities. For a cardinality-k cutset with $k > 2$ containing $\{i,j\}$ and e, the restoration flow for edge e is more likely shared by edge $\{i,j\}$ and other edges in the cutset. If this were the case, i.e., $f^e_{ij} + f^e_{ji} < d_e$, then the current solution would be likely to satisfy the arc residual capacity constraint.

Batch	Number of cuts		Gap reduction %	
	KQ cuts	CAP cuts	KQ cuts	CAP cuts
R20	6.2	21.8	42.7	57.3
R30	9.6	33.8	41.9	58.1
R40	15.5	41.0	46.6	53.4
R50	18.8	48.4	39.7	60.3

Table 8.9: Gap reduction by inequality type

Batch	Number of cuts		Gap reduction %	
	Singleton cutsets	GH cutsets	Singleton cutsets	GH cutsets
R20	22.6	5.4	89.0	11.0
R30	35.8	7.6	90.5	9.5
R40	48.2	8.3	91.3	8.7
R50	57.6	9.6	90.3	9.7

Table 8.10: Gap reduction by cutset type

9. Conclusions

In this paper, we studied single facility line restoration problem (*SFLR*) with both distinct and integrated spare capacity system requirements. We examined the underlying polyhedron structure and developed several classes of strong valid inequalities, many of which are facet-defining in the subspace of capacity variables under appropriate conditions. We presented computational results to demonstrate that the linear programming formulation is very strong in the distinct case, and that branch-and-bound can solve the problems to optimality within a reasonable amount of time. In the integrated case, our computational results showed that including these valid inequalities considerably improves the integrality gap, from an average of 10% to an average of under 1%. These results indicate that strong cutting planes combined with branch and bound can provide efficient algorithms for solving real world problems in the telecommunication industry.

A. Proofs

Proof of Proposition 4.1.

The Hamilton Circuit (*HC*) problem: Does a given undirected graph $G = (N, E)$ contain a cycle that visits each node exactly once?

Given any instance of the *HC* problem, we shall construct an *SFLR* instance as follows.

Define a connected and undirected graph $G = (N, E)$ corresponding to the base network. Assume that $c_{ij} = 1$ for all $\{i, j\}$ and $b = 1$. Assume that we need to restore every edge in E, and that the working flow on each edge is 1, that is, $d_{ij} = 1$ for all $\{i, j\}$.

Decision problem:

Is min $\sum_{\{i,j\}\in E} c_{ij}y_{ij} \leq |N|$?

Note the following properties of any optimal solution to the *SFLR*.

(i) The value of each component of y is either 0 or 1. If $y_{ij} = 1$, we install one unit of spare capacity on edge $\{i, j\}$. Let F denote the set of edges we choose to install spare links. We refer to $H = (N,\ F)$ as the reserve network.

(ii) $H = (N, F)$ must be connected. Suppose, to the contrary, that no spare links connect two node sets N_1 and N_2. Since the base network G is connected, it contains an edge $\{i, j\}$ in G with $i \in N_1$ and $j \in N_2$. When edge $\{i, j\}$ fails, the reserve network cannot send any flow from N_1 to N_2.

(iii) Each edge in H must lie on a cycle. If any edge $\{i, j\}$ not contained in any cycle fails, the reserve network would contain no other path between node i and j.

(iv) $|F| \geq |N|$. As a result of (ii) and (iii), H must be connected and contain at least one cycle.

We claim that we would have a Yes instance of *HC* if and only if we have a Yes instance of *SFLR*. If we have a Yes instance of *HC*, we install the spare links on the edges of the Hamilton circuit. The circuit contains two paths between any node pair $\{i, j\}$ in H. If any edge $\{i, j\}$ in G fails, we would be able to reroute the flow on at least one other path. Therefore, $|F| = |N|$ and the solution is feasible and so it is optimal.

Now assume that we have an optimal solution $H = (N,\ F)$ to *SFLR* with $|F| = |N|$, and we want to show that H is a Hamilton circuit. By (iii), each edge must lie on a cycle in H, and the number of edges is $|N|$. By (ii), H is connected.

Therefore, the solution is a Hamilton circuit, and we have a Yes instance of the *HC* problem.

Notice that the instances of *SFLR* created by our transformation all have unit costs and unit demands. Therefore, this subset of problems satisfy the similarity assumption. This transformation actually shows that the subset of the *SFLR* instances satisfying the similarity assumption is itself NP-complete. Therefore, *SFLR* is NP-complete in the strong sense. ■

Proof of Theorem 6.1.

Let

$$\sum_{i \in E} \alpha_i y_i = \beta \tag{*}$$

represent an arbitrary equation that is satisfied by every feasible solution y that satisfies the strengthened cutset capacity inequality (6.5) as an equality. Using a common proof technique in polyhedral combinatorics, to show that the inequality is a facet, we will show that the coefficients in equation (*) are a multiple of those in inequality (6.5). We use interchange arguments to derive a relationship between the coefficients appearing in (*).

We first construct a feasible solution y satisfying the strengthened cutset capacity inequality (6.5) as an equality as follows. Since the two subnetworks formed by removing the cutset W are restorable, we can install sufficient capacity to restore the edges in the subnetworks. We assign a sufficiently large amount of spare capacity on edge e so that it can accommodate the interrupted flow if any other edge in W fails. We select an arbitrary edge $i \in W \backslash \{e\}$ and assign $\text{Req}(W, e)$ units of spare capacity on it. Therefore edge e can be restored and the solution is feasible.

(1) Claim: $\alpha_j = 0$ for all $j \notin W \backslash \{e\}$.

Given any feasible solution satisfying (6.5) as an equality, we can always increase y_j by 1 for $j \notin W \backslash \{e\}$ while keeping other variables unchanged. The new solution y' is feasible and satisfies (6.5) as an equality. Substituting the y and y' values into (*) and subtracting shows that $\alpha_j = 0$. Thus, the coefficients of every edge not in $W \backslash \{e\}$ are zero in the equation (*).

(2) Claim: $\alpha_i = \alpha_j$ if $i, j \in W \backslash \{e\}$.

Since we select edge $i \in W \backslash \{e\}$ arbitrarily, we can obtain a new solution y'' by selecting edge $j \in W \backslash \{e\}$ instead of i. The new solution y'' is feasible and satisfies (6.5) as an equality. Substituting the y and y'' values into (*) and subtracting shows that $\alpha_i = \alpha_j$ if $i, j \in W \backslash \{e\}$.

Therefore, the coefficients in equation (*) are a multiple of those in the inequality (6.5) and so in the space of y variables, (6.5) is a facet. ■

Proof of Theorem 6.2.

Let

$$\sum_{i \in E} \alpha_i y_i = \beta \qquad (**)$$

represent an arbitrary equation that is satisfied by every feasible solution y that satisfies the cardinality-k inequality (6.6) as an equality. Again we use interchange arguments to derive a relationship between the coefficients appearing in (**).

We construct a feasible solution y satisfying (6.6) as an equality as follows. We allocate a sufficiently large amount of spare capacity on all the edges in the two subnetworks. For example, we can set y_i to be the largest demand for all edges i. Since the two subnetworks formed by removing the cutset K are restorable, we have enough capacity to restore the edges in the subnetworks. To restore the edges in the cutset K, we then install a total of $\lceil \mathrm{Req}(K)/(k-1) \rceil$ units of spare capacity in K as follows. We set $y_1 = \lceil \mathrm{Req}(K)/(k-1) \rceil - \mathrm{Req}(K,1)$, $y_2 = \lceil \mathrm{Req}(K)/(k-1) \rceil - \mathrm{Req}(K,2)$, and so on, until the total $\lceil \mathrm{Req}(K)/(k-1) \rceil$ units of spare capacity are all allocated. Some of the edges in cutset K might have zero capacity. It is easy to verify that the solution is feasible.

(1) Claim: $\alpha_j = 0$ for all $j \notin K$.

Given any feasible solution satisfying (6.6) as an equality, we can always increase y_j by 1 while keeping other variables unchanged. The new solution y' is feasible, and satisfies (6.6) as an equality. Substituting the y and y' values into (**) and subtracting shows that $\alpha_j = 0$. Thus, the coefficients of every edge not in K are zero in the equation (**).

(2) Claim: $\alpha_i = \alpha_j$ if $i, j \in K$.

First we show that at least one of the edges in K, let it be edge k, has capacity less than $\lceil \mathrm{Req}(K)/(k-1) \rceil - \mathrm{Req}(K,k)$. Suppose not, then we have $y_i = \lceil \mathrm{Req}(K)/(k-1) \rceil - \mathrm{Req}(K,i)$ for all $i \in K$. Then the total capacity in K is

$$\begin{aligned} k \lceil \mathrm{Req}(K)/(k-1) \rceil - \mathrm{Req}(K) &= k(\frac{\mathrm{Req}(K) - r}{k-1} + 1) - \mathrm{Req}(K) \\ &= (\mathrm{Req}(K) - r)/(k-1) + (k - r) \\ &= \lceil \mathrm{Req}(K)/(k-1) \rceil + (k - r). \end{aligned}$$

In this expression, $r = \mathrm{Req}(K) \bmod (k-1)$. Since $r < k$, the total capacity in K is strictly greater than $\lceil \mathrm{Req}(K)/(k-1) \rceil$. This result contradicts the fact that the total capacity in K is $\lceil \mathrm{Req}(K)/(k-1) \rceil$.

We then construct alternate solutions by adding one unit to y_k and subtracting one unit from some y_i, given edge i has positive capacity. The new solution contains sufficient capacity for restoring edges in K, and satisfies inequality (6.6) as an equality. The interchange arguments show that $\alpha_i = \alpha_k$ for all edges i with positive capacity.

At least one of the edges, let it be edge 1, has a strictly positive capacity. For the edges j with zero capacity, we construct alternate solutions by subtracting one unit from y_1 and adding one unit to y_j. Since $\text{Req}(K, j) < \lceil \text{Req}(K)/(k-1) \rceil$, $\text{Req}(K, j) \leq \lceil \text{Req}(K)/(k-1) \rceil - 1$, and the new solution contains sufficient capacity for restoring edges in K. It also satisfies inequality (6.6) as an equality. The interchange arguments show that $\alpha_i = \alpha_1$ for all edges i with zero capacity.

Therefore, the coefficients in equation (**) are a multiple of those in the inequality (6.6) and so in the space of y variables, (6.6) is a facet. ■

B. Real-world problem instances

Figure B.1, B.2 and B.3 show the network topologies for the three real-world problems that we used in our computational experiments.

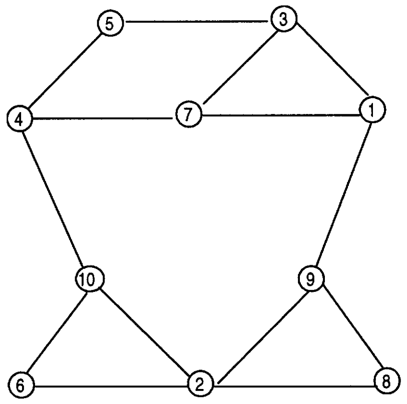


Figure B.1: BMSW1 problem

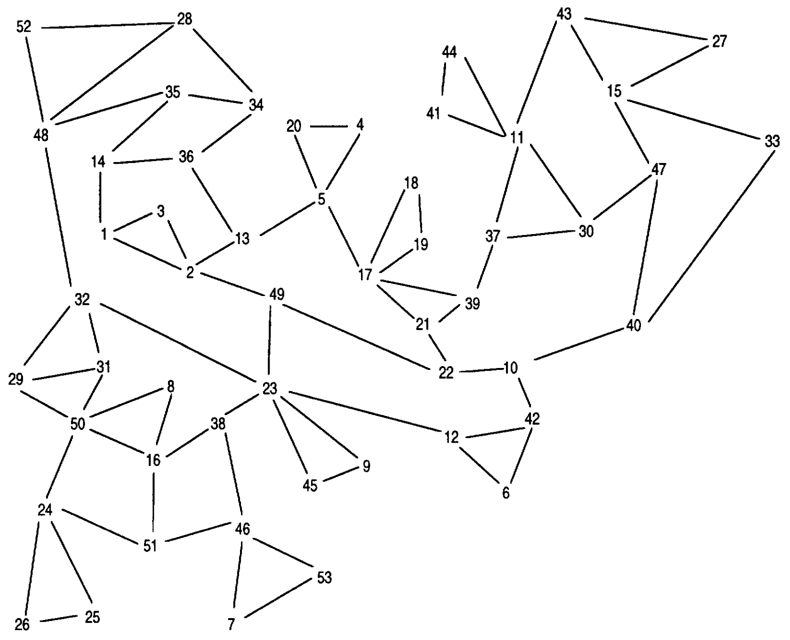


Figure B.2: BMSW2 problem

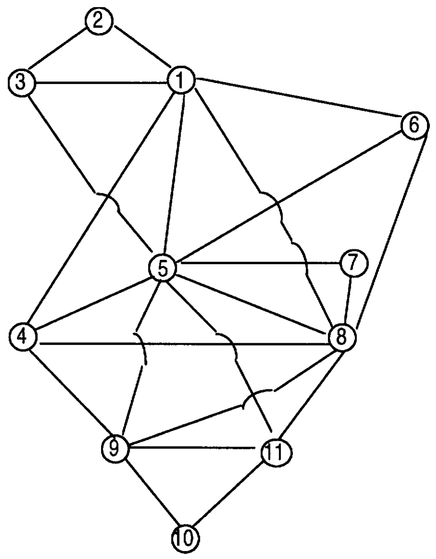


Figure B.3: SHN problem

References

[1] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice-Hall, Englewood Cliffs, N.J., 1993.

[2] A. Balakrishnan, T. Magnanti, J. Sokol, and Y. Wang. Algorithms for link restoration in telecommunication networks. Working paper, Operations Research Center, MIT, Cambridge, MA, 1996.

[3] D. Bienstock and O. Günlük. Capacitated network design - polyhedral structure and computation. *INFORMS Journal on Computing*, 8(3):243–59, 1996.

[4] D. Bienstock and G. Muratore. Strong inequalities for capacitated survivable network design problems. Working paper, Columbia University, 1997.

[5] B. Brockmüller, O. Günlük, and L. Wolsey. Designing private line networks - polydedral analysis and computations. Discussion Paper 9647, Center for Operations Research and Econometrics, Université Catholique de Louvain, Belgium, 1996.

[6] 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(7):1–9.

[7] R. E. Gomory and T. C. Hu. An application of generalized linear programming to network flows. *SIAM Jour. Appl. Math.*, 10:207–83, 1962.

[8] R. E. Gomory and T. C. Hu. Synthesis of a communication network. *SIAM Jour. Appl. Math.*, 12:348–69, 1964.

[9] R. Gormory and T. Hu. Multi-terminal network flows. *J. Soc. Indust. Appl. Math.*, 9(4):551–70, 1961.

[10] W. Grover, T. Billodeau, and B. Venables. Near optimal spare capacity planning in a mesh restorable network. *Proc. IEEE Globecom'91*, 2007–12, 1991.

[11] M. Herzberg. A decomposition approach to assign spare channels in self-healing networks. *IEEE Global Telecommunications Conference*, 1601–5, 1993.

[12] M. Herzberg. A decomposition approach to assign spare channels in self-healing networks. *Globecom '93*, 1601–5, 1993.

[13] 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):775–84.

[14] J. Kennington and J. E. Whitler. An efficient decomposition algorithm to optimize spare capacity in a telecommunications network. Technical Report 97-CSE-5, Southern Methodist University, Dallas, TX, 1998.

[15] A. Lisser, R. Sarkissian, and J. Vial. Survivability in transmission telecommunication networks. Note Technique, CNET, Paris, France, 1995.

[16] A. Lisser, R. Sarkissian, and J. Vial. Optimal joint syntheses of base and spare telecommunication networks. International Symposium on Mathematical Programming, 1997.

[17] T. Magnanti and P. Mirchandani. Shortest paths, single origin-destination network design and associated polyhedra. *Networks*, 23:103–121, 1993.

[18] T. Magnanti, P. Mirchandani, and R. Vachani. The convex hull of two core capacitated network design problems. *Mathematical Programming*, 60:233–250, 1993.

[19] T. Magnanti, P. Mirchandani, and R. Vachani. Modeling and solving the two facility capacitated network loading problem. *Operations Research*, 43:142–157, 1995.

[20] T. Magnanti and Y. Wang. Polyhedral properties of the network restoration problem – with the convex hull of a special case. Working Paper OR 323-97, Operations Research Center, MIT, 1997.

[21] H. Sakauchi, Y. Nishimura, and S. Hasegawa. A self-healing network with an economical spare-channel assignment. *Globecom'90*, 438–43, 1990.

[22] M. Stoer and G. Dahl. A polyhedral approach to multicommodity survivable network design. *Numerische Mathematik*, 68(1):149–67, 1994.

[23] J. Veerasamy, S. Venkatesan, and J. Shah. Spare capacity assignment in telecom networks using path restoration. *Mascots'95*, 370–4, 1995.

[24] Y. Wang. *Modeling and Solving Single and Multiple Facility Network Restoration Problems.* PhD thesis, Operations Research Center, MIT, Cambridge, MA, in progress.

[25] T. Wu. *Fiber Network Service Survivability.* Artech House, London, 1992.

[26] T. Wu. Emerging technologies for fiber network survivability. *IEEE Communications Magazine*, 33(2):58–74, 1995.