

## Chapter 4

# SURVIVABLE NETWORK DESIGN: ROUTING OF FLOWS AND SLACKS

Deepak Rajan

*Department of Industrial Engineering and Operations Research  
University of California, Berkeley 94720–1777  
deepak@ieor.berkeley.edu*

Alper Atamtürk\*

*Department of Industrial Engineering and Operations Research  
University of California, Berkeley 94720–1777  
atamturk@ieor.berkeley.edu*

**Abstract** We present a new mixed–integer programming model and a column generation method for the survivable design of telecommunication networks. In contrast to the failure scenario models, the new model has almost the same number of constraints as the regular network design problem, which makes it effective for large instances. Even though the complexity of pricing the exponentially many variables of the model is  $\mathcal{NP}$ –hard, in our computational experiments, we are able to produce capacity–efficient survivable networks with dense graphs up to 70 nodes.

**Keywords:** Column generation, p–cycles, pricing complexity, survivable network design.

## 1. Introduction

In this paper we consider the survivable design of telecommunication networks. Given an undirected graph  $G = (N, E)$ , where  $N$  is the set of nodes and  $E$  is the set of links, i.e., tuples of nodes, and demand of each node from every other node, the *telecommunication network design problem* is to install integer multiples of a capacity unit on the links and route the flow of demands so that

---

\*Supported, in part, by NSF grant DMII–0070127.

the total capacity installation cost is minimized. In many telecommunication networks (e.g. Asynchronous Transfer Mode networks), the capacity installed on a link between two nodes allows running of flow up to the capacity in both directions. Thus capacity is undirected even though flow is directed.

A network is said to be *survivable* if all of the demands can be met under the failure of any one of its links. In this paper, we define link failure as the event of decreasing the capacity of the link to zero. Since in telecommunication networks the probability of two components failing simultaneously is very small, designing a network protected against single component failures is considered satisfactory. Two edge-connectedness of the underlying graph  $G$  is a necessary condition for the survivability of the network, but is clearly not sufficient. In order to ensure that the flow on the network can be rerouted in case of a failure, sufficient spare (excess) capacity must be available on the working links of the network. However, over-provisioning of capacity is a major concern for telecommunication companies due to the high investment costs required in installing capacity. Therefore designing capacity-efficient survivable networks (i.e., networks with low capacity installation cost) is a critical problem.

Various heuristic and exact approaches have been developed for designing survivable networks. The reader is referred to [19] for an overview of survivable network design problems and a synthesis of related literature. One of the simplest techniques for protecting a network against failures is the so-called  $1 + 1$  *Diverse Protection* (DP) switching [7], which uses dedicated link-disjoint backup routes for each demand pair. Although rerouting disrupted flow to an alternative route can be done very fast in  $1 + 1$  DP networks and operating such networks is quite easy, allocating capacity for demand-dedicated link-disjoint routes results in a network that is highly capacity-inefficient.

With the advent of add/drop multiplexers, a new protection technique known as the *Self-Healing Rings* (SHR) [2, 11, 17, 19] has been introduced. The topology of SHR networks is a set of rings (undirected cycles) covering the nodes of the graph. Due to the ring topology, SHR networks are inherently survivable. If a link of a ring fails, flow on the link is sent along the ring in the reverse direction. SHR networks deliver very fast rerouting times in the event of a failure while achieving lower spare capacity requirements than  $1 + 1$  DP networks, since spare capacity on a ring is shared by all demand flows using that ring. Even though SHRs provide good survivability characteristics and extremely fast reconfiguration of flow, imposing a ring topology on the telecommunication network still leads to inefficient capacity utilization and therefore high cost.

A significant increase in the capacity efficiency of survivable networks can be achieved by allowing a general network topology and global rerouting of flows in the case of link failures. This requires provisioning link capacities that will allow rerouting under every link failure scenario [3]. Unfortunately,

the size of failure scenario based global link restoration models grows very rapidly with the size of the graph and render such models unfit for tackling practical problems. Furthermore, implementation of global rerouting of all flows – whether disrupted or not – in the event of a failure requires much more complex hardware and software packages and is inherently slower than the SHR and 1 + 1 DP networks.

Consequently, hierarchical restoration schemes are popular for designing survivable networks in practice. In the first stage, link capacities are determined for the no-failure scenario without survivability concerns. In the second stage, sufficient spare capacity is assigned to the links of the network so that the disrupted flow can be safely rerouted in the case of failures [4, 5, 12, 13, 14]. The reader may refer to [21] for a detailed comparison of various restoration strategies.

In this paper we introduce a new mixed-integer programming model for designing survivable networks. This model considers routing of no-failure flows and failure flows simultaneously by installing slacks on the directed cycles of the network so as to ensure survivability in the case of link failures. In a failure, only disrupted flow is rerouted. However, since failure and no-failure flows are considered simultaneously when determining link capacities, the model delivers survivable networks with capacity-efficiency very close to global link restoration. The model builds upon [12] and improves capacity efficiency by using *directed* p-cycles for routing disrupted flow. Furthermore, since only disrupted flow is rerouted, reconfiguration of the network can be done quickly. Of significant note, the number of the constraints of the formulation is almost the same as the regular network design problem, which makes the model effective for large instances. The number of the variables is exponential in the number of links of the graph; however, the variables are treated implicitly by a column generation approach.

**Outline.** In Section 2, in addition to the new model, we describe two others: global link restoration and spare capacity assignment using p-cycles, that are used for comparison. In order to tackle large instances, in Section 3 we develop a column generation approach and show that the pricing complexity of the variables is  $\mathcal{NP}$ -hard. In Section 4 we present results of computational experiments that compare the capacity-efficiency and ease of solvability of these models. Using a polynomial-time pricing heuristic, we solve the model by column generation and report successful computational experiments with dense graphs up to 70 nodes.

## 2. Link restoration

### 2.1 Global link restoration

Here we present a mathematical model of the minimum cost survivable network design problem. We refer to this model as the global link restoration, since it allows rerouting *all* flows, even those that are not disrupted under a link failure. In order to differentiate between undirected capacities and directed flow, we let  $[ij]$  denote the undirected link between nodes  $i$  and  $j$ , and  $(ij)$  and  $(ji)$  denote the two directed arcs corresponding to link  $[ij]$ . We define the arc set  $A$  of the network as the set of all arcs corresponding to  $E$ , that is,  $A = \{(ij), (ji) : [ij] \in E\}$ . Let  $S$  be the set of failure scenarios. For simplicity of notation, we represent the no-failure scenario with  $0 \in S$ ; thus  $S = E \cup \{0\}$  ( $0 \notin E$ ), and let  $A \setminus [ij] = A \setminus \{(ij), (ji)\}$ . We define commodities by aggregating all of the flow originating from a node. So, the demand associated with commodity  $k \in K \subseteq N$  is the sum of demands  $d_{ik}$  from node  $k$  to every other node  $i$ . Let  $b_i^k = \sum_{j \neq k} -d_{jk}$  if  $i$  is the origin node of commodity  $k$ ,  $b_i^k = d_{ik}$  if  $i$  is a destination node of commodity  $k$  and  $b_i^k = 0$  otherwise. Also let  $g_{ij}^k$  and  $h_e$  be the cost of routing commodity  $k$  on arc  $(ij)$  and installing a capacity unit on link  $e$ , respectively, and finally let  $c_e$  be the existing (previously installed) capacity on link  $e \in E$ . Then the *Global Link Restoration Model* is

$$\begin{aligned} \min \quad & \sum_{(ij) \in A} \sum_{k \in K} g_{ij}^k x_{ij}^{k0} + \sum_{e \in E} h_e y_e \\ & \sum_{j:(ji) \in A} x_{ji}^{ks} - \sum_{j:(ij) \in A} x_{ij}^{ks} = b_i^k \quad \forall i \in N, \forall k \in K, \forall s \in S \quad (4.1) \\ \text{(GLR)} \quad & \sum_{k \in K} x_{ij}^{ks} \leq c_e + y_e \quad \forall (ij) \in A \setminus \{s\}, \forall s \in S, e = [ij] \quad (4.2) \\ & y_e \in \mathbf{Z}_+ \quad \forall e \in E \\ & x_{ij}^{ks} \in \mathbf{R}_+ \quad \forall (ij) \in A, \forall k \in K, \forall s \in S. \end{aligned}$$

where  $x_{ij}^{ks}$  is the amount of commodity  $k$  routed through arc  $(ij)$  in failure scenario  $s$  and  $y_e$  is the capacity installed on link  $e$ . Constraints (4.1) guarantee that the all demands are satisfied in all failure scenarios. Constraints (4.2) ensure that the capacity installed on a link is large enough to accommodate flow routed through that link for all failure scenarios.

(GLR) imposes no restrictions on either the network structure or the routing of flow. Hence, it delivers the most capacity-efficient survivable network possible (by letting  $g_{ij}^k = 0$ ) if it can be solved to optimality. Note, however, that formulation (GLR) has  $|N||K||S| + |A||S|$  constraints and  $|A||K||S| + |E|$  variables. For a complete graph of  $|N|$  nodes with complete demand,  $|K| = |N|$ ,  $|A| = |N|(|N| - 1)$ , and  $|S| = |N|(|N| - 1)/2 + 1$ . So, for instance, when  $|N| = 20$ , (GLR) has 148,980 constraints and 1,451,790 variables. Even

when the network is not as dense, (GLR) can easily have tens of thousands of constraints and millions of variables for even medium-sized networks, making it virtually impossible to load into the computer memory, let alone solve it to optimality. Also note that the size of (GLR) is larger than the regular network design problem (NDP) by an order of magnitude  $|S|$  ((NDP) is the special case of (GLR) with the single no-failure scenario  $S = \{0\}$ ). Since solving (GLR) optimally is exceedingly difficult except for very small instances, researchers have adopted heuristic approaches to solve it and its variants [9, 16]. Polyhedral cutting planes are given in [3, 6] for (GLR).

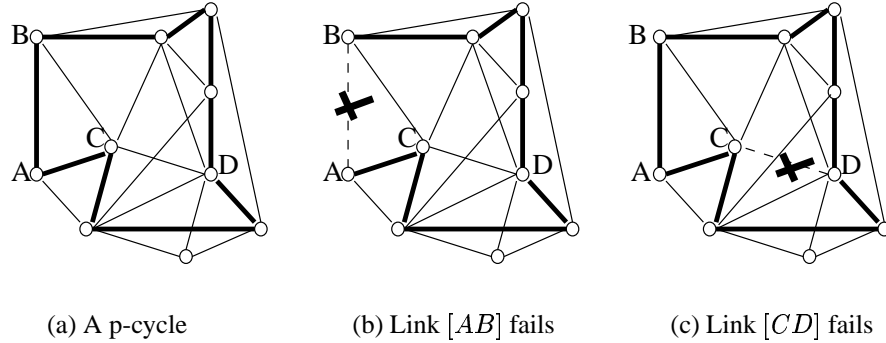
## 2.2 Spare Capacity Assignment

Even though the global link restoration scheme described in the previous section achieves the lowest capacity requirement, it carries several disadvantages. The first is the size of the formulation. A second disadvantage is that rerouting of disrupted as well as undisrupted flow in case of a failure is harder to implement than rerouting only disrupted flow. It requires sophisticated and expensive hardware and software and longer reconfiguration times. On the other hand, rerouting disrupted demand on rings in SHR networks requires much simpler equipment and control mechanisms at the nodes, and is much easier and faster to implement.

As a result, recently there has been an interest in developing hybrid networks which are capacity-efficient and at the same time are easy to restore under failure. This is accomplished in an hierarchical manner. In the first stage a capacity-efficient solution (flows and link capacities) is found for the network design problem without survivability concerns. In the second stage, given the working capacities, a minimum cost allocation of spare capacity on the links is determined so that flow on each arc can be routed, in case of a link failure. This second stage problem is referred to as the *Spare Capacity Assignment Problem* (SCA).

Grover and Stamatelakis [12] introduce the concept of utilizing (undirected) predefined cycles (*p-cycles*) of links for the configuration of spare capacity. In this scheme sufficient spare capacity is installed on the cycles of the graph so that the working capacity on any link is covered by the cycles that the link is either on or is a chord of. Figure 4.1 illustrates the way in which an individual *p*-cycle may be used for restoration. In (i), an example of a *p*-cycle is shown in bold links. Note that a *p*-cycle is just an undirected cycle which is used to cover working link capacities. In (ii), link  $[ab]$  on the *p*-cycle fails, and the remaining links of the cycle are used for rerouting the flow on link  $[ab]$ . In (iii), we see how the *p*-cycle can also be used for restoring the flow on a link that is a chord of the *p*-cycle. Here link  $[cd]$  fails, and the *p*-cycle provides two restoration paths between  $c$  and  $d$ . Thus installing half the working capacity

Figure 4.1. An undirected p-cycle



of  $[cd]$  on the  $p$ -cycle as spare capacity covers link  $[cd]$ . Further inspection of Figure 4.1 shows that this particular  $p$ -cycle provides restoration paths for nine *on-cycle* failures and for ten *chord* failures.

Let  $(\bar{x}, \bar{y})$  be a solution for the regular network design problem (NDP) and  $P$  be the set of simple undirected cycles of  $G$ . Grover and Stamatelakis [12] give the following integer set covering model for the capacity assignment problem with  $p$ -cycles:

$$\begin{aligned}
 \text{(SCA)} \quad \min \quad & \sum_{e \in E} h_e \sum_{p \in P} r_{pe} w_p \\
 & \sum_{p \in P} q_{pe} w_p \geq c_e + \bar{y}_e \quad \forall e \in E \quad (4.3) \\
 & w_p \in \mathbf{Z}_+ \quad \forall p \in P.
 \end{aligned}$$

The decision variable  $w_p$  denotes the number of spare capacity units assigned to  $p$ -cycle  $i \in P$ . Here  $r_{pe}$  is 1 if link  $e$  is on cycle  $p$  and 0 otherwise;  $q_{pe}$  is 2 if link  $e$  is a chord of cycle  $p$ , 1 if  $e$  is on cycle  $p$ , and 0 otherwise. Thus  $\sum_{p \in P} r_{pe} w_p$  is the spare capacity installed on link  $e$ .

Naturally, solving first a network design problem and then assigning spare capacity to cover the working links is less capacity-efficient than solving (GLR) directly. However, this hierarchical approach breaks the task of designing a survivable network into two problems that are much easier to tackle computationally than global link restoration (GLR) and therefore is often preferred in practice. The hierarchical spare capacity assignment (SCA) approach using  $p$ -cycles has been reported to achieve much better capacity-efficiency than ring architectures as well as very quick recovery times by several authors [12, 18, 20].

### 2.3 Routing of Flows and Slacks

In this section, we present a mixed–integer programming model for routing failure flows and no–failure flows simultaneously. Rather than using undirected p–cycles to cover working link capacities from (NDP), we utilize *directed* p–cycles of arcs to introduce sufficient slack on top of the no–failure flows, so that the flow on each arc can be rerouted along these slacks. We refer to this scheme as the routing of flows and slacks. Let  $x_{ij}^k$  be the amount of commodity  $k$  flowing through arc  $(ij) \in A$  in the no–failure scenario. Let  $C$  be the set of *directed* cycles of the network. Define a cycle–slack variable  $z_c$  to denote the amount of slack routed on cycle  $c \in C$ . For directed cycle  $c \in C$  and arc  $(ij) \in A$  let  $\alpha_{ij}^c$  be 1 if  $c$  includes  $(ij)$ , 0 otherwise, and let  $\rho_{ij}^c$  be 1 if  $(ij)$  is a chord to cycle  $c$ , 0 otherwise. Then the *Routing of Flows and Slacks* can be formulated as

$$\begin{aligned} \min \quad & \sum_{(ij) \in A} \sum_{k \in K} g_{ij}^k x_{ij}^k + \sum_{e \in E} h_e y_e \\ & \sum_{j: (ji) \in A} x_{ji}^k - \sum_{j: (ij) \in A} x_{ij}^k = b_i^k \quad \forall i \in N, \forall k \in K \quad (4.4) \end{aligned}$$

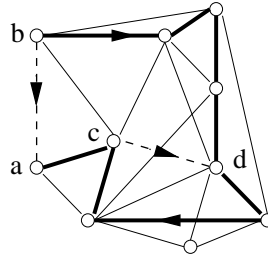
$$\text{(RFS)} \quad \sum_{k \in K} x_{ij}^k - \sum_{c \in C} \rho_{ij}^c z_c - \sum_{c \in C} \alpha_{ji}^c z_c \leq 0 \quad \forall (ij) \in A \quad (4.5)$$

$$\sum_{k \in K} x_{ij}^k + \sum_{c \in C} \alpha_{ij}^c z_c \leq c_e + y_e \quad \forall (ij) \in A, e = [ij] \quad (4.6)$$

$$\begin{aligned} y_e & \in \mathbf{Z}_+ & \forall e \in E \\ z_c & \in \mathbf{R}_+ & \forall c \in C \\ x_{ij}^k & \in \mathbf{R}_+ & \forall (ij) \in A, \forall k \in K. \end{aligned}$$

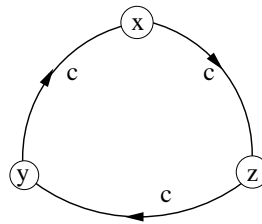
Constraints (4.5) ensure that for each arc  $(ij)$  the total slack installed on the directed cycles that  $(ij)$  is on or a chord is at least the total flow on  $(ij)$ . Observe that a directed cycle–slack provides coverage for flows in the reverse direction for the arc on the cycle; see arc  $(ba)$  in Figure 4.2. A directed cycle–slack provides only *one* recovery path for the flow on a chord arc; see arc  $(cd)$  in Figure 4.2. However, slack introduced for the directed cycle in the reverse direction also provides coverage for the flow on arc  $(cd)$  as well. Constraints (4.6) ensure that capacity installed on link  $[ij]$  is large enough to accommodate the flow routed on arc  $(ij)$  as well as the slack introduced on the arc. In order to emphasize the importance of routing slacks through directed p–cycles, rather than covering working link capacities with undirected p–cycles as in [12], we give the example in Figure 4.3. Here, capacity is available in units of  $c$  and the flow on each arc  $(x, z)$ ,  $(z, y)$ , and  $(y, x)$  equals  $c$ . Since installed capacity on a link allows flow in both directions up to capacity, the network in Figure 4.3 is survivable with a total of  $3c$  capacity. However, covering

Figure 4.2. A directed p-cycle



installed capacities with undirected p-cycles would require installing additional  $c$  units on each link and would double the installed capacity. However routing a cycle-slack of  $c$  units in the counter-clockwise direction (from  $x$  to  $y$  to  $z$ ) covers all of the flow on the network and requires no additional capacity. We note that routing slacks to cover failure-flows will lead to lower capacity than covering no-failure capacity even without the assumption that installed capacity on a link allows flow in both directions up to capacity. Consequently, (RFS)

Figure 4.3. A small survivable network



produces survivable networks that are more capacity-efficient than (SCA) for two reasons: (1) (RFS) routes slacks rather than using undirected p-cycles to cover working link capacities, and (2) (RFS) considers the routing of no-failure flows when determining excess capacity installation. Indeed our computational experiments indicate that the capacity-efficiency delivered by (RFS) is very close to (GLR) for small instances.

Interestingly, in contrast to (GLR), (RFS) requires only one additional constraint for each arc than the regular network design problem (NDP) without survivability requirement. However, the number of cycles in a graph, hence cycle-slack variables in the formulation, is exponential in the number of the arcs.

### 3. A column generation approach

Since (RFS) has exponentially many cycle–slack variables, all of the variables cannot be included in the model when solving large instances. Selecting a small subset of the variables a priori and solving the model with these variables can result in suboptimal solutions. Therefore, we develop a column generation [8] approach and introduce the cycle–slack variables into (RFS) with a restricted number of variables as they are needed. In order to solve larger instances efficiently, we also reformulate (RFS) using path–flow variables, rather than the arc–flow variables. This reduces the number of constraints, introduces an exponentially many path–flow variables, which can also be generated, as needed, via column generation.

Because the column generation algorithms for multicommodity flow problems converge to an optimal LP solution faster when commodities are disaggregated [15], we use pairwise demands as commodities in the formulation. Let  $K$  be the set of node pairs with positive demand and  $P_k$  denote the set of  $s_k$ – $t_k$  paths for commodity  $k \in K$ . For  $(ij) \in A$  and  $p \in P_k$ , let  $\delta_{ij}^p$  be 1 if path  $p$  includes arc  $(ij)$ , 0 otherwise. For  $p \in P_k$ , defining the path–flow variables  $x_p$  as the fraction of commodity  $k$  routed through path  $p$  under the no–failure scenario, we reformulate the problem of routing of flows and slacks as

$$\begin{aligned} \min \quad & \sum_{(ij) \in A} \sum_{k \in K, p \in P_k} d_k \delta_{ij}^p g_{ij}^k x_p + \sum_{e \in E} h_e y_e \\ (w_k) \quad & \sum_{p \in P_k} x_p = 1 \quad \forall k \in K \end{aligned} \quad (4.7)$$

$$(u_{ij}) \quad \sum_{k \in K, p \in P_k} d_k \delta_{ij}^p x_p - \sum_{c \in C} \rho_{ij}^c z_c - \sum_{c \in C} \alpha_{ji}^c z_c \leq 0 \quad \forall (ij) \in A \quad (4.8)$$

$$(v_{ij}) \quad \sum_{k \in K, p \in P_k} d_k \delta_{ij}^p x_p + \sum_{c \in C} \alpha_{ij}^c z_c \leq c_e + y_e \quad \forall (ij) \in A, e = [ij] \quad (4.9)$$

$$\begin{aligned} \text{(RFS–P)} \quad & x_p \in \mathbf{R}_+ \quad \forall p \in P_k, \forall k \in K \\ & y_e \in \mathbf{Z}_+ \quad \forall e \in E \\ & z_c \in \mathbf{R}_+ \quad \forall c \in C. \end{aligned}$$

where  $u, v, w$  are the corresponding dual variables of the LP relaxation of (RFS–P). Constraints (4.7) ensure the demand for each commodity is satisfied for the no–failure scenario. Constraints (4.8) ensure that sufficient slack is allocated to directed cycles to cover all no–failure flow on each arc. Constraints (4.9) ensure that for each edge, sufficient capacity is installed for routing the slack introduced and the no–failure flow on the arcs in either direction.

### 3.1 Pricing of cycle–slack variables

Given an LP–relaxation solution to (RSF–P) that has a restricted set of cycle–slack variables, we are interested in finding a cycle–slack variable  $z_c$  with negative reduced cost, if one exists. For a dual solution  $(u, v, w)$  the reduced cost of the cycle–slack variable  $z_c$  is

$$\sum_{ij \in A} ((u_{ji} - v_{ij})\alpha_{ij}^c + u_{ij}\rho_{ij}^c).$$

A negative reduced cost  $z_c$  can be identified by finding a negative weight directed cycle that has at least three arcs on  $G = (N, A)$ , where weight of an arc  $(ij)$  on the cycle is  $f_{ij}^a = u_{ji} - v_{ij}$  and the weight of an arc  $(ij)$  that is a chord of the cycle is  $f_{ij}^h = u_{ij}$ . Observe that  $f^a$  is unrestricted in sign, but  $f^h \leq 0$  since  $u \leq 0$ .

A negative weight directed cycle could be found in polynomial time, for instance, by the Bellman–Ford algorithm [1] if the cycles did not carry a weight for their chords. The presence of weights for the chords complicates the problem significantly.

**Pricing Problem of P–cycles.** Let us formally define the Pricing Problem of P–cycles (PPP) as: Given the arc weights  $f^a \in \mathbf{R}^A$  and chord weights  $f^h \in \mathbf{R}^A$ , either find a negative cost p–cycle or conclude that no such p–cycle exists.

**Theorem 1** *The Pricing Problem of P–cycles (PPP) is NP–hard.*

PROOF: We prove the theorem by reducing PPP to the decision version of TSP [10]: Given a complete directed graph  $G = (N, A)$ , weights  $d : A \mapsto \mathbf{Z}_+$ , and a positive integer  $k$ , does there exist a Hamiltonian cycle in  $G$  with total weight  $< k$ ? In order to answer TSP, we construct the following instance of PPP. Let  $n = |N|$ . For  $(ij) \in A$ , let  $f_{ij}^a = d_e + (M(n-3) - 2k)/(n(n-1))$  and  $f_{ij}^h = -(M+k)/(n(n-1))$ , where  $M > nk$ . Let  $h(c)$  denote the set of chords of p–cycle  $c$  in  $G$ . Any p–cycle on  $G$  with  $\ell$  arcs has exactly  $\ell^2 - 3\ell$  directed chords. Hence the weight of p–cycle  $c$  with  $\ell$  arcs is equal to

$$\begin{aligned} \sum_{(ij) \in c} f_{ij}^a + \sum_{(ij) \in h(c)} f_{ij}^h &= d(c) + \frac{\ell(M(n-3) - 2k)}{n(n-1)} - \frac{\ell(\ell-3)(M+k)}{n(n-1)} \\ &= d(c) + M \frac{\ell(n-3) - \ell(\ell-3)}{n(n-1)} - k \frac{2\ell + \ell(\ell-3)}{n(n-1)} \\ &= d(c) + M \frac{\ell(n-\ell)}{n(n-1)} - k \frac{\ell(\ell-1)}{n(n-1)} \end{aligned} \quad (4.10)$$

where  $d(c)$  is the weight of the cycle  $c$  for the TSP. When  $M$  is chosen as above, (4.10) is positive unless  $\ell = n$ . Hence, PPP has an affirmative answer only if

the  $p$ -cycle is Hamiltonian. However, since the weight of any Hamiltonian  $p$ -cycle  $c$  on  $G$  is  $d(c) - k$  (the second term in (4.10) vanishes when  $\ell = n$ ), PPP has an affirmative answer if there exists a Hamiltonian cycle  $c$  of weight  $d(c) < k$ . Hence, TSP has an affirmative answer if and only if PPP has an affirmative answer.  $\diamond$

**A polynomial-time heuristic.** When all chord weights  $f^h$  of  $p$ -cycles are zero, PPP reduces to finding a negative weight ( $f^a$ ) directed cycle with at least three arcs. This can be accomplished in  $O(|A||N|^2)$  with a simple modification to the Bellman-Ford label-correcting algorithm for finding shortest paths in a directed graph.

Since chord weights  $f^h$  are nonpositive, if we find a negative weight cycle  $c$  by assuming that  $f^h$  is zero, we also find a negative weight  $p$ -cycle. Therefore, in order to find negative reduced cost cycle-slack variables, we first find negative weight directed cycles using  $f^a$  and add the corresponding cycle-slack variables to the restricted formulation of (RFS-P). When we exhaust all such negative weight cycles, there can still be other negative reduced cost cycle-slack variables with  $f_{ij}^a < 0$ , that could not be found this way.

Note that the longer the  $p$ -cycle is, especially for dense graphs, the more the number of chords it has. Thus, longer  $p$ -cycles have a higher tendency of having a negative weight. Therefore, we could potentially get “good”  $p$ -cycles by changing the weights  $f^a$  in such a way that we get longer cycles when we solve the polynomial-time negative weight cycle problem. One possible way of accomplishing this is by reducing all arc weights  $f^a$  by a certain constant so that the longer cycles will be in favor. We incorporate this idea in the column generation algorithm for pricing cycle-slack variables that do not correspond to negative weight cycles.

### 3.2 Pricing of path-flow variables

The pricing problems of the path-flow variables are disjoint for each commodity  $k \in K$  and therefore can be solved separately. Given a dual solution  $(u, v, w)$  to the LP relaxation of the restricted (RFS-P), the reduced cost of a path-flow variable  $x_p$   $p \in P_k$  is

$$\sum_{ij \in A} (g_{ij}^k - u_{ij} - v_{ij}) d_k \delta_{ij}^p - w_k$$

Since  $u, v \leq 0$ ,  $\zeta = \min\{\sum_{ij \in A} (g_{ij}^k - u_{ij} - v_{ij}) d_k \delta_{ij}^p : p \in P_k\}$  is an  $s_k$ - $t_k$  shortest path problem with nonnegative weights, and can be solved efficiently using Dijkstra’s algorithm [1]. If  $\zeta < w_k$ , then the flow-path variable corresponding for an optimal  $s_k$ - $t_k$  path has a negative reduced cost, and is added to the restricted formulation.

#### 4. Computational Results

Here we present our computational experiments performed to compare the capacity–efficiency and ease of solvability of the models (NDP), (GLR), (SCA), (RFS), and (RFS–P). Our goal is to determine whether the new method of routing flow and slacks is a viable alternative for designing survivable telecommunication networks.

While (GLR) produces the most capacity–efficient survivable networks, the size of the formulation, which essentially carries a copy of the regular network design problem for each failure scenario, makes it unfit for tackling problems unless they are very small. Nevertheless, in our experiments we run the (GLR) model for small instances in order to find the lowest capacity requirement for the purpose of comparison. In order to have a fair comparison for the capacity–efficiency of (SCA) and (RFS), we solved these models with the same set of cycles that are selected a priori. For these experiments we created small instances of randomly generated graphs with 75% link density and 50% demand density. In Table 4.1 we present the number of constraints and variables in the four formulations for the graph with 14 nodes. (NDP) is the regular network design problem with no survivability constraints. In Table 4.1, we see that the formulations of (NDP) and (RFS) are about the same size. The formulation of (GLR) is orders of magnitude bigger than the rest of the models. For (RFS), the restricted set of  $p$ –cycles is chosen as follows: First, we calculate the minimum cost spanning tree ( $T$  - using edge weights proportional to  $h_e$ ). Then, for every edge  $e$  on the tree, we find the minimum cost cycle that uses edge  $e$  and exactly one edge  $\notin T$ . Cycle–slack variables (both directions) corresponding to these undirected cycles are added to the formulation. We also add these cycle–slack variables a priori to (RFS–P).

Table 4.1. Problem size

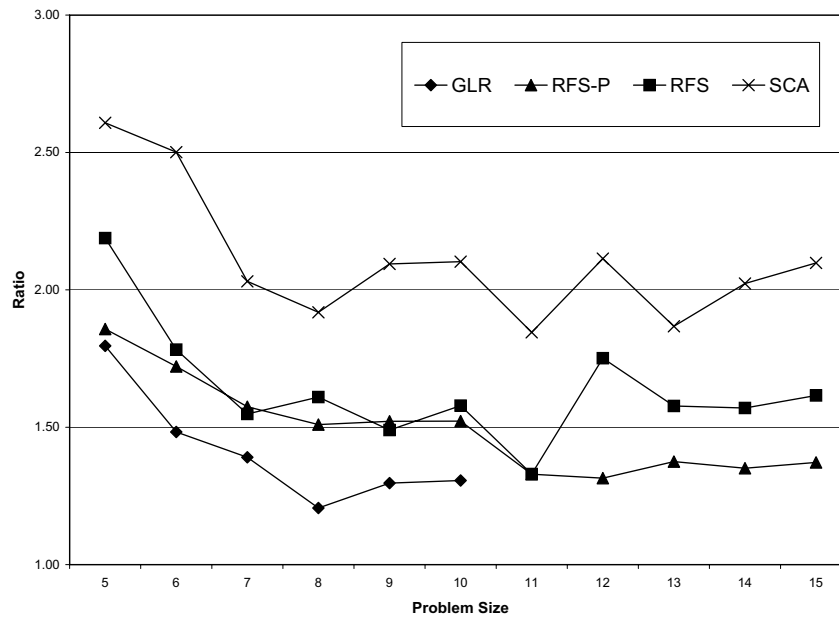
$n = 14$	(NDP)	(GLR)	(SCA)	(RFS)
Cons	342	27352	133	488
Vars	2117	151329	105	2173

All of the models, are solved with CPLEX7.5 MIP solver on an Intel Pentium4 2GHz Linux workstation with 1GB main memory with one hour CPU time limit. In Table 4.2 we report the solution times in CPU second (Time) and the best IP solutions found (IPsoln) for the four models. Optimality gap (End Gap) of the best IP solution is reported in place of time if the branch–and–bound computation is not finished in one hour time limit. In Figure 4.4 we show ratio of installed capacity for the solutions provided by (GLR), (SCA), and (RFS) to the capacity of the regular (nonsurvivable) network design model (NDP). Here

Table 4.2. Comparison of the models

Size <i>n</i>	Time/(End Gap)				IP soln			
	(NDP)	(GLR)	(SCA)	(RFS)	(NDP)	(GLR)	(SCA)	(RFS)
5	0.03	0.57	0.00	0.04	50.5	90.7	131.7	110.5
6	0.16	1.27	0.00	0.19	129.5	192.0	323.9	230.8
7	0.27	23.6	0.00	1.00	103.4	143.8	210.0	160.1
8	2.88	577.1	0.00	4.18	146.6	176.8	281.2	236.0
9	517.1	( 3.5 )	0.00	2.88	172.7	223.9	361.7	257.3
10	( 1.6 )	( 13 )	0.00	2.47	235.8	308.0	495.8	372.2
11	1216	( - )	0.00	42.5	289.3	( - )	533.9	384.7
12	( 3.0 )	( - )	0.00	61.2	326.0	( - )	689.0	570.9
13	( 1.6 )	( - )	0.00	2967	366.1	( - )	683.7	577.4
14	( 1.9 )	( - )	0.00	( 0.6 )	443.0	( - )	896.2	695.5
15	( 1.8 )	( - )	0.00	( 0.9 )	554.9	( - )	1164	896.4

Figure 4.4. Relative capacity–efficiency of the survivability models



we see that survivable networks produced by (SCA) has about 100% more capacity installed compared with the nonsurvivable networks (NDP), whereas (RFS) requires only about 50% increase in the capacity. Moreover, when compared with (GLR), which provisions the lowest possible capacity for survivable networks, we see that (RFS) requires only an additional 20 % capacity, whereas (SCA) provisions 59% excess capacity over (GLR).

When we compare the models in terms of ease of solvability, we see that the (SCA) model is the easiest to solve; it takes negligible time. We should recall, however, that (SCA) requires the solution of (NDP) as an input. So one should take into consideration of the solution time of (NDP) for designing survivable networks with (SCA). Surprisingly, we observe in Table 4.2 that (RFS) is solved more easily than (NDP) and we were able to obtain optimal solutions or feasible solutions within 1% of optimal (for the subset of cycle-slack variables used in the formulation) for all instances in Table 4.2. It was not possible to solve even the LP relaxations of (GLR) for instances with more than 14 nodes within an hour of CPU time. No feasible solution is found by CPLEX for instances with more than 10 nodes.

Table 4.3. The effect of column generation

Size <i>n</i>	(RFS)			(RFS-P)			Paths	Cycles
	(1)	(2)	(3)	(1)	(2)	(3)		
5	0.00	0.04	110.5	0.01	0.05	93.8	23	8
6	0.00	0.19	230.8	0.01	0.19	222.9	36	6
7	0.01	1.00	160.1	0.01	0.41	162.8	51	4
8	0.01	4.18	236	0.01	2.55	221.3	107	14
9	0.02	2.88	257.3	0.02	6.03	262.8	143	14
10	0.04	2.47	372.2	0.02	4.99	358.9	182	10
11	0.10	42.5	384.7	0.06	88.0	384.5	282	26
12	0.05	61.2	570.9	0.17	2337	428.6	499	38
13	0.08	2967	577.4	0.24	( 1.4 )	503.4	567	50
14	0.12	( 0.6 )	695.5	0.31	( 0.2 )	598.4	702	46
15	0.19	( 0.9 )	896.4	0.42	( 0.2 )	761.3	839	46

(1) LP Soln time, (2) Time (End Gap), (3) IP soln.

In Table 4.3 we compare (RFS-P) with (RFS) to see the effect of generating variables as needed rather than solving (RFS) on a subset of the variables selected a priori. A comparison of columns (3) indicate that the capacity-efficiency of the networks improve significantly by pricing the cycle-slack variables based on their LP reduced costs. The networks produced by (RFS-P) have about 15% more capacity than the ones from (GLR), whereas (RFS) has 20% excess and (SCA) has 59% excess capacity. The LP solution times for (RFS-P), which include the time for pricing variables, indicate that columns

can be generated very efficiently. Observe in the last two columns of Table 4.3 that only a small number of the variables are generated.

Finally in Table 4.4 we report the results of our experiments with the column generation approach for large instances. Table 4.4 demonstrates that relatively large instances could be effectively solved by the new survivability model (RFS-P) with a column generation approach. These computational experiments suggest that the method of routing flows and slacks is quite effective in designing survivable networks.

Table 4.4. Experiments with large instances

<i>Size</i>	<i>LP time</i>	<i>(EndGap)</i>	<i>IP Soln</i>	<i>Paths</i>	<i>Cycles</i>
20	3.97	( 0.7 )	1641	2171	100
30	27.8	( 1.4 )	3567	3549	190
40	123.4	( 2.8 )	6245	7262	262
50	186.3	( 1.1 )	12212	10643	352
60	2315	( 0.6 )	28754	14251	366
70	1333	( 1.4 )	24121	23490	390

## 5. Conclusions and research directions

In this paper, we have presented a new method (RFS) for designing capacity-efficient survivable telecommunication networks. This method differs from the hierarchical link capacity covering methods such as (SCA) [12], in that we route no-failure flows and slacks for disrupted flow through directed cycles of the network. Therefore, capacity-efficiency achieved by (RFS) is always at least as high as (SCA). Our computational experiments show that (RFS) delivers consistently about 30% more capacity-efficient networks than (SCA) does. In fact, (RFS) compares well with global link restoration (GLR), which gives theoretically the most capacity-efficient survivable networks possible (see Figure 4.4). In order to solve large instances we developed a column generation approach. We showed that the pricing problem for the cycle-slack variables problem is  $\mathcal{NP}$ -hard, and gave an efficient heuristic to price them effectively. Judicious selection of cycle-slack variables seems to be very important in increasing the capacity-efficiency of the networks. Pricing the variables based on their LP reduced costs, even with a heuristic method, improved capacity-efficiency of the networks over selecting them a priori significantly. Our computational experiments suggest that the method of routing flows and slacks is an effective way for designing survivable telecommunication networks.

There are several directions for further research. We are currently working on developing other efficient ways of pricing the cycle-slack variables. We

plan to study the weighted p-cycle problem in detail. In a subsequent paper, we will perform a polyhedral analysis of the new model, in order to develop strong cutting planes for the problem. A challenging issue is how to integrate column and cut generation schemes.

Finally the method of routing flows and slacks can be easily adapted to other failure scenarios, such as simultaneous failure of links or node failures and to particular technologies, such as WDM, VWP, WP networks.

## References

- [1] Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Englewood Cliffs, NJ.
- [2] Altinkemer, K. (1994). Topological design of ring networks. *Computers and Operations Research*, 21:421–431.
- [3] Alevras, D., Grötschel, M., and Wessälly R. (1998). Cost-efficient network synthesis from leased lines. *Annals of Operations Research*, 76:1–20.
- [4] Balakrishnan, A., Magnanti, T. L., Sokol, J. S., and Wang, Y. Modeling and solving the single facility line restoration problem. *Operations Research*, to appear.
- [5] Balakrishnan, A., Magnanti, T. L., Sokol, J. S., and Wang, Y. (2001). Telecommunication link restoration planning with multiple facility types. *Annals of Operations Research*, 106:127–154.
- [6] Bienstock, D. and Muratore, G. (2000). Strong inequalities for capacitated survivable network design problems. *Mathematical Programming*, 89:127–147.
- [7] Chung, S. H, King, H. G., Yoon Y. S., and Tcha, D. W. (1996). Cost-minimizing construction of a unidirectional SHR with diverse protection. *IEEE Transactions on Networking*, 4:921–928.
- [8] Chvátal, V. (1983). *Linear Programming*. W. H. Freeman and Company, New York.
- [9] Dahl, G. and Stoer, M. (1998). A cutting plane algorithm for multicommodity survivable network design problems. *INFORMS Journal on Computing*, 10:1–11.
- [10] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York.
- [11] Goldschmidt, O., Laugier, A., and Olinick, E. V. SONET/SDH ring assignment with capacity constraints. *Discrete Applied Mathematics*, to appear.
- [12] Grover, W. D. and Stamatelakis, D. (1998). Cycle-oriented distributed pre-configuration: ring-like speed with mesh-like capacity for self-planning network restoration. *Proceedings of IEEE International Conference on Communications 1998*, 537–543.
- [13] Herzberg, M., Bye, S. J., and Utano, A. (1995). The hop-limit approach for spare capacity assignment in survivable networks. *IEEE/ACM Transactions on Networking* 3:775–784.
- [14] Iraschko, R., MacGregor, M., and Grover, W. (1998). Optimal capacity placement for path restoration in STM or ATM mesh survivable networks. *IEEE/ACM Transactions on Networking* 6:325–336.
- [15] Jones, K. L., Lustig, I. J., Farvolden, J. M., and Powell, W. B. (1993). Multicommodity network fbws: the impact of formulation on decomposition. *Mathematical Programming* 62:95–117.

- [16] Lissner, A., Sarkissian, R., and Vial, J. P. (1995). Survivability in telecommunication networks. Technical Report 1995.3, Department of Management Studies, University of Geneva, Switzerland.
- [17] Luss, H., Rosenwein, M. B., and Wong, R. T. (1998). Topological network design for SONET ring architecture. *IEEE Transactions on Systems, Man and Cybernetics*, 28:780–790.
- [18] Schupke, D. A., Gruber, C. G., and Autenrieth, A. (2002). Optimal configuration of p-cycles in WDM networks. *IEEE International Conference on Communications 2002*.
- [19] Soriano, P., Wynants, C., Séguin, R., Labbé, M., Gendreau, M., and Fortz, B. (1998). Design and dimensioning of survivable SDH/SONET networks. In Sansò, B. and Soriano, P., editors, *Telecommunications Network Planning*, pages 147–168. Kluwer Academic Publishers, Netherlands.
- [20] Stamatelakis, D. and Grover, W. D. (2000). Theoretical underpinnings for the efficiency of restorable networks using pre-configured cycles (“p-cycles”). *IEEE Transactions on Communications*, 48:1262–1265.
- [21] Xiong, Y. and Mason, L. G. (1999). Restoration strategies and spare capacity requirements in self-healing ATM networks. *IEEE/ACM Transactions on Networking*, 7:98–110.