

A new model for designing survivable networks

Alper Atamtürk¹ and Deepak Rajan

Department of Industrial Engineering and Operations Research

University of California, Berkeley 94720-1777

email: {atamturk}{deepak}@ieor.berkeley.edu

Abstract

We present a new approach for designing survivable networks. In this scheme, we explicitly introduce slack on directed cycles of the network, which are used to reroute disrupted flow under failure. We model this scheme using both hierarchical and integrated framework. We give strong valid inequalities that explicitly uses survivability requirements. We present a computational study on solving the integrated model using a column generation method for pricing the cycle variables and the valid inequalities as cutting planes.

1 Introduction

A network is said to be survivable if all of the demands on the nodes can be met under the failure of any one of its links. In order to ensure that the flow on the network can be rerouted in the case of a failure, sufficient spare (excess) capacity must be available on the working links of the network.

Since over-provisioning of capacity is a major concern due to the high investment costs required for installing capacity, designing capacity-efficient survivable networks is a critical problem in the telecommunication industry. A number of strategies have been developed for designing survivable networks; see for instance, [10, 3, 11, 2, 20]. The reader is referred to [18] for an overview of survivable network design problems and a synthesis of the related literature.

Recently there has been an increased effort in designing *hybrid* networks that would achieve reconfiguration times down to the regime of rings, but without giving up the desired capacity-efficiency of a mesh/general network. There are two limited senses in which hybrids are already being implemented in telecommunication networks. One is the principle of “*ring access and mesh transport*.” In this widely used form of hybrid, restoration is applied for backbone networks and SHRs are used for Local Area Networks (LANs). A second practice that could be called a type of hybrid is “*meshed-rings*”, which is really ring-based networking with inter-ring transitions being managed by restoration. The third type of hybrid networks allows for the allocation of capacity on various cycles to act as rings. Any fraction of a commodity may now be routed on these rings, and is inherently survivable (see [12] for details). The rest of the commodity is routed as in a mesh-network.

Here we focus on a fourth approach for designing hybrid networks, which involves the use of failure-flow patterns for restoration of disrupted flow. It uses a mesh network for no-failure routing, but utilizes specified patterns for routing flow in case of disruption. We build on [13], in which the authors first determine link capacities and routing of commodities for the no-failure scenario, and then add undirected rings (p-cycles) on the network in order to cover working link capacities. Our approach differs from [13] in three aspects. Firstly, in the hierarchical scheme, we explicitly introduce slack on directed cycles of the network in such a way that flow on each disrupted arc can be rerouted through the slack; thus existing slacks in the network are utilized as well. Secondly, with an integrated model that incorporates survivability into routing decisions, we determine working capacities as well as excess capacities simultaneously. Thirdly, as a first step of a detailed study, here we do not allow covering of flow on chords of a cycle. This simplifies column generation and polyhedral analysis significantly.

The proposed models are presented in Section 2. In Section 3, we give polyhedral valid inequalities that explicitly consider the survivability requirements. Section 4, we present computational results that compares the installed capacity for the hierarchical model and the integrated model. The computational results also show that the cutting planes developed here improve the performance of the solution process significantly. Finally, in Section 5, we conclude by discussing directions for future research.

¹Supported, in part, by NSF grants DMI-0070127 and DMI-0218265 and a grant from ILOG, Inc.

2 Designing survivable networks using cycles

In this section we present two mixed-integer programming models for designing survivable networks using directed cycles. We utilize directed cycles to introduce sufficient slack on top of the no-failure flows, so that the flow on each arc can be rerouted along these cycles. Observe that excess capacity (slack) on a directed cycle provides coverage for flows in the reverse direction to the arc on the cycle. In Figure 2, the directed cycle (in bold) covers disrupted flow on arc (1, 2) in the reverse direction. If arc link [1, 2] fails, the flow along arc (1, 2) can be sent from node 1 to node 2 along the cycle.

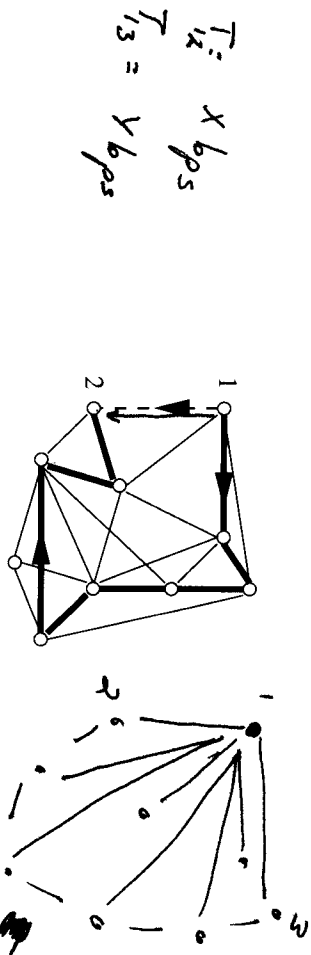


Figure 1: A directed cycle protecting flow on (1,2)

2.1 Hierarchical approach

First, we present a hierarchical scheme for designing survivable networks using cycles. In this scheme, we first solve the network design model without survivability requirements (NDP). Then, we use the optimal solution (no-failure flows and working capacity) from (NDP) as input to the spare capacity assignment model, where we explicitly introduce slack into the directed cycles so that the no-failure flow on each arc can be safely rerouted along these cycles.

Let f_{ij} be the amount of flow through arc $(ij) \in A$, and w_e be the working capacity in the no-failure scenario (obtained as the solution to (NDP)). Let C be the set of directed cycles of the network. Define a

cycle variable z_c to denote the amount of slack introduced for cycle $c \in C$. For directed cycle $c \in C$ and arc $(ij) \in A$ let α_{ij}^c be 1 if c includes (ij) , 0 otherwise. Define capacity variable $y_{[ij]}$ as the amount of capacity installed on edge $[ij] \in E$, and let h_e be the cost of installing unit of capacity on edge e . Then the Spare Capacity Assignment (SCA) can be formulated as

$$\min \sum_{e \in E} h_e y_e \quad (1)$$

$$\text{s.t.:} \quad \sum_{c \in C} \alpha_{ij}^c z_c \geq f_{ij} \quad \forall (ij) \in A \quad (1)$$

$$\text{(SCA)} \quad f_{ij} + \sum_{c \in C} \alpha_{ij}^c z_c \leq w_e + y_e \quad \forall (ij) \in A, e = [ij] \quad (2)$$

$$y_e \in \mathbb{Z}_+ \quad \forall e \in E, \quad z_c \in \mathbb{R}_+ \quad \forall c \in C$$

Constraints (1) ensure that for each arc (ij) the total slack introduced on the directed cycles with the reverse arc (ji) is at least the total flow on (ij) . Constraints (2) 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.

2.2 Integrated approach

Next, we present a model for simultaneous routing of flows and cycles. Here, all decisions (no-failure routing, allocation of cycles and installation of capacity) are made at the same time. Therefore, the integrated model provides a network with a cost that is at most the optimal cost of the hierarchical model.

Let z, y, h and α be as defined in Section 2.1. Define variable x_{ij}^k as the amount of commodity k flowing through arc $(ij) \in A$ in the no-failure scenario, and let g_{ij}^k be the cost associated with routing each unit of the commodity. Let b_i^k be the supply of commodity k at node i . (b_i^k is negative if i is a demand node.) Then the Simultaneous Routing of Flows and Cycles

(RFC) can be formulated as

$$\begin{aligned} \min \quad & \sum_{(ij) \in A} \sum_{k \in K} g_{ij}^k x_{ij}^k + \sum_{e \in E} h_e y_e \\ \text{s.t.} \quad & \sum_{j:(j,i) \in A} x_{ji}^k - \sum_{j:(i,j) \in A} x_{ij}^k = b_i^k, & \forall i \in N, \forall k \in K & (3) \end{aligned}$$

$$\begin{aligned} \text{(RFC)} \quad & \sum_{k \in K} x_{ij}^k - \sum_{c \in C} \alpha_{ij}^c z_c \leq 0 & \forall (ij) \in A & (4) \\ & \sum_{k \in K} x_{ij}^k + \sum_{c \in C} \alpha_{ij}^c z_c \leq y_e & \forall (ij) \in A, e = [ij] & (5) \end{aligned}$$

$$y_e \in \mathbb{Z}_+, \quad \forall e \in E, z_c \in \mathbb{R}_+, \quad \forall c \in C, x_{ij}^k \in \mathbb{R}_+, \quad \forall (ij) \in A, \forall k \in K.$$

Constraints (4) ensure that for each arc (ij) the total slack installed on the directed cycles with the reverse arc (ji) is at least the total flow on (ij) . Constraints (5) 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.

Since we route flows and cycles simultaneously, (RFC) is an improvement on a hierarchical framework using cycles for ensuring survivability. Computational experiments in Section 4 show that significant savings are achieved over (SCA).

2.3 Pricing cycle variables

Interestingly, (RFC) requires only one additional constraint for each arc than the regular network design problem. However, the number of cycle variables is exponential in the number of arcs, and all of the variables cannot be included in the model when solving large instances. Therefore, we develop a column generation scheme to include the cycle variables into the model as they are needed.

Given an LP-relaxation solution $(\bar{x}, \bar{y}, \bar{z})$ to (RFC) (see Section 2.2), we now search for a cycle c with negative reduced cost, if one exists. Let (w, u, v) be the dual variables corresponding to constraints (3), (4) and (5) respectively. Note that w is unrestricted in sign and $u, v \leq 0$. For each arc (ij) on cycle c , the variable z_c appears twice in the formulation: 1. in constraint (4) for arc (ji) with coefficient $-\alpha_{ij}^c$; and 2. in constraint (5)

with coefficient α_{ij}^c . Then, given a dual solution (u, v, w) , the reduced cost of cycle c is: $\sum_{ij \in A} (u_{ji} - v_{ij}) \alpha_{ij}^c$. We define the Cycle Pricing Problem (CPP): either find a negative reduced-cost cycle with at least three arcs, or conclude that no such cycle exists. CPP is solvable in $O(mn^2)$ with a variant of the Bellman-Ford algorithm [1]. A delayed column generation approach that detects negative cycles in the network is developed and used to price the cycle variables.

3 Strong valid inequalities

Cutting plane methods have been successfully used in solving network designing problems (see [4], [5], [6], [7], [8], [9], [16], [17]). In particular, cut-set and partition inequalities have been shown to be very effective in practice. In this section, we describe strong polyhedral inequalities that explicitly utilize the cycle variables for ensuring survivability of the network.

3.1 2-partition inequalities

First, we derive valid inequalities for (RFC) based on its single-commodity cut-set (2-partition) relaxation. Let (U, V) represent a non-empty partitioning of the network. Let A^+ be the arcs directed from U to V , A^- be the arcs directed from V to U , and d denote the supply of a commodity in U for V .

Consider the following *cut-set* relaxation of (RFC)

$$x(A^+) - x(A^-) = d \tag{6}$$

$$x_a + z_a \leq y_a \quad \forall a \in A \tag{7}$$

$$0 \leq x_a \leq z_{\bar{a}} \quad \forall a \in A \tag{8}$$

$$z(A^+) = z(A^-) \tag{9}$$

where x_a denotes the flow on arc a , z_a the capacity reserved for cycle variables on arc a and y_a the units of capacity installed in arc a . \bar{a} is used to indicate the arc in the reverse direction of arc a .

Constraints (6) are obtained by aggregating the flow balance constraints across the cut-set in the single-commodity formulation of (RFC). Constraints (7) and (8) are the capacity and survivability (flow restoration) constraints for the arcs in the cut-set. Constraint (9) ensures that the net capacity reserved for cycles across the cut-set is the same in either direction, since any directed cycle which goes across the cut-set (using arcs in A^+) also has to come back across the cut-set (using arcs in A^-); see Figure 2. We denote the convex hull of all feasible points that satisfy (6), (7), (8) and (9) by \mathcal{F}_C .

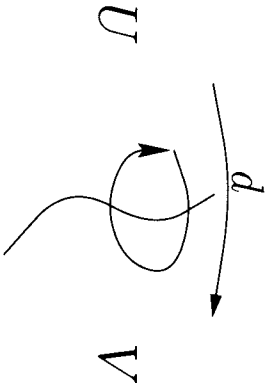


Figure 2: Cut-set

Before presenting a class of cut-set inequalities for \mathcal{F}_C , we motivate and explain the simplest variant of the inequalities. The net flow across any cut-set must be at least d . Furthermore, since the network has to be survivable, the net slack allocated for cycles across the cut-set must be sufficient to cover this flow. Consequently, the net capacity across this cut-set must be at least the sum of these two values, each of which must be at least d . Finally, since the capacity variables are integral, we can round up $2d$ to the smallest greater integer, giving us the inequality

$$y(A^+) \geq \lceil 2d \rceil \quad (10)$$

Defining $\eta = \lceil 2d \rceil$ and $\tau = 2d - \lceil 2d \rceil$, we can generalize this inequality to include flow and cycle variables, and also variables in the reverse direction of the cut-set.

Theorem 3.1. *For $S^+ \subseteq A^+$ and $S^- \subseteq \overline{S^+}$, the cut-set inequality*

$$\begin{aligned} z(A^+ \setminus S^+) + \tau y(S^+) + x(A^+ \setminus S^+) \\ + (1-\tau)y(S^-) - x(S^-) - x(\overline{A^+ \setminus S^+}) \geq \tau\eta \end{aligned}$$

is valid for \mathcal{F}_C . (11)

Inequality (11) is not always facet-defining for \mathcal{F}_C ; however, it often defines a high dimensional face. Next, we present necessary and sufficient facet-defining conditions for (11).

Theorem 3.2. *Let $S^- = \emptyset$. Then inequality (11) is facet-defining for \mathcal{F}_C if and only if $\tau > 0$ and $S^+ \neq \emptyset$.*

$$\tau y(S^+) + x(A^+ \setminus S^+) + z(A^+ \setminus S^+) - x(\overline{A^+ \setminus S^+}) \geq \tau\eta \quad (12)$$

For a fixed cut-set A , the separation problem of (11) is trivial: given $(\bar{y}, \bar{x}, \bar{z})$, if $(1-r)\bar{y}_a > \bar{x}_a$ for $a \in A^-$, then we include \bar{a} in S^+ if $\tau\bar{y}_{\bar{a}} < \bar{x}_{\bar{a}} + \bar{z}_{\bar{a}} - \bar{x}_a$. On the other hand, if $(1-r)\bar{y}_{\bar{a}} \leq \bar{x}_{\bar{a}}$ for $a \in A^+$, then we include a in S^+ and \bar{a} in S^- if $\tau\bar{y}_a + (1-r)\bar{y}_{\bar{a}} < \bar{x}_a + \bar{z}_a$. However, finding the best cut-set A is not easy.

3.2 3-partition inequalities

Consider a non-empty partition (A, B, C) of the network. Let AB be the arcs directed from A to B , and similarly define the other arc-sets. For each proper subset (U) of $\{A, B, C\}$, we define E_U as the arcs from U to $N \setminus U$, and d_U the net supply of U . Let $\eta_U = \lceil 2d_U \rceil$ and $\tau_U = 2d_U - \lceil 2d_U \rceil$. We further divide each set of arcs into 2 groups $(AB$ into AB_1 and $AB_2)$.

Now, we can write cut-set inequalities for all the possible partitions $(U, N \setminus U)$. Exactly half of these inequalities are always dominated, since η_U or $\eta_{N \setminus U}$ is 0. There are six such inequalities; here we present two of them.

$$\begin{aligned} \tau_A y(AB_1 \cup AC_1) + x(AB_2 \cup AC_2) \\ + z(AB_2 \cup AC_2) - x(BA_2 \cup CA_2) \geq \tau_A \eta_A \end{aligned} \quad (13)$$

$$\begin{aligned} \tau_B y(AB_1 \cup BC_1) + x(BA_2 \cup BC_2) \\ + z(BA_2 \cup BC_2) - x(AB_2 \cup CB_2) \geq \tau_B \eta_B \end{aligned} \quad (14)$$

Adding (13) and (14) we get

$$\begin{aligned} (\tau_A + \tau_B)y(AB_1) + \tau_A y(AC_1) + \tau_B y(BC_1) + x(AC_2 \cup BC_2) \\ - x(CA_2 \cup CB_2) + z(AB_2 \cup BA_2 \cup AC_2 \cup BC_2) \geq \tau_A \eta_A + \tau_B \eta_B \end{aligned}$$

Dividing this inequality by $(r_A + r_B)$, and applying MIR, we get the 3-partition inequality

$$\begin{aligned} & r(r_A + r_B)y(AB_1) + r_A y(AC_1) + r_B y(BC_1) + x(AC_2 \cup BC_2) \\ & - x(CA_2 \cup CB_2) + z(AB_2 \cup BA_2 \cup AC_2 \cup BC_2) \geq r\eta(r_A + r_B) \end{aligned} \quad (15)$$

where $\eta = \lfloor \frac{\tau_A \bar{y}_A + \tau_B \bar{y}_B}{\tau_A + \tau_B} \rfloor$ and $r = \frac{\tau_A \bar{y}_A + \tau_B \bar{y}_B}{\tau_A + \tau_B} - \lfloor \frac{\tau_A \bar{y}_A + \tau_B \bar{y}_B}{\tau_A + \tau_B} \rfloor$

For a fixed non-empty partition (A, B, C) , the separation problem of (15) is simple. Given $(\bar{y}, \bar{x}, \bar{z})$ for $a \in AB$, we include a in AB_1 if $r(\tau_A + \tau_B)\bar{y}_a < \bar{z}_a + \bar{x}_a$. For $a \in BC$, we include a in BC_1 if $r_B \bar{y}_a + < \bar{x}_a + \bar{z}_a - \bar{x}_a$. For $a \in AC$, we include a in AC_1 if $r_A \bar{y}_a + < \bar{x}_a + \bar{z}_a - \bar{x}_a$.

4 Computational Results

In this section, we present the results of a computational study undertaken to study the capacity requirements of simultaneous routing of flows and cycles (RFC) vis-a-vis a hierarchical approach (SCA), and regular network design without survivability requirements (NDP). We also test the effectiveness of the cuts presented in Section 3. We solve all instances using CPLEX8.1 MIP solver on an Intel Pentium4 2GHz Linux workstation with 1GB memory. We terminate each run after one hour of CPU time if the optimal solution has not been obtained yet, and recover the best feasible solution.

We randomly generated one instance each of network size (n) from 5 to 12, with 75% link density and 50% demand density. In order to reduce the problem size, we aggregate demands by node at the source, thus using demand trees instead of demand pairs. We price cycles to optimality at the root node of the branch-and-bound tree. We noticed that using cycles (RFC) to design survivable networks allows us to work with formulations of roughly the same size as (NDP). (SCA) has roughly the same number of constraints as (RFC), but much fewer variables since the flow is pre-terminated – however, one must first solve (NDP) before solving (SCA).

First, we compare the time taken to solve the various models, and the minimum amount of capacity required. In Table 1, if a problem is solved to optimality in 1 hour, we report the optimal objective value and the elapsed CPU time in seconds, otherwise we report the percentage gap

between the objective value of the best known feasible solution and the best lower bound at termination, and the objective value of that feasible solution.

Size n	Time/(End Gap)			Best IP soln		
	NDP	SCA	RFC	NDP	SCA	RFC
5	0.03	0.01	0.06	50.5	110.3	103.5
6	0.16	0.02	0.23	129.5	251.7	235.6
7	0.27	0.03	2.29	103.4	222.3	189.4
8	2.88	0.04	14.0	146.6	286.2	259.6
9	517.1	0.15	20.9	172.7	364.9	311.6
10	(1.6)	0.21	(0.3)	235.8	457.6	432.4
11	1216	0.16	(0.8)	289.3	562.7	527.1
12	(3.0)	0.30	(2.7)	326.0	641.2	592.1

Table 1: Comparing the models

Comparing the capacity requirements of the models, we see that (SCA) needs about 100% more than (NDP), whereas (RFC) on the average requires 80% more, a saving of 10%. Comparing the models in terms of ease of solvability, we see that (SCA) takes the least amount of time. However, one must understand that we first need to solve (NDP) before using this solution as an input to (SCA). Interestingly, (RFC) is not any harder to solve than (NDP).

Next, we investigate the effectiveness of polyhedral survivability inequalities presented in Section 3 when used as cutting planes to improve the linear programming relaxations. We report the improvement of the integrality gap at the root node and its effect on the number of branch-and-bound nodes and the solution times, with and without the cuts. The default CPLEX cuts are added in both runs. In Table 2, we summarize the computations on dense graphs (75% link density). We see that we obtain more than twice as much integrality gap improvement at the root when survivability cuts are added. This leads to significant reductions in the number of nodes and solution time. In particular, we were able to solve problems 10 and 11 to optimality within an hour, which was not possible without the survivability.

We measured the effect of the cuts for sparse graphs (with 25% link density) as well; see Table 3. Again, in all the instances, the new cuts

Size n	Root impr.		B&B nodes		Time/(End Gap)	
	Def	Surv cuts	Def	Surv cuts	Def	Surv cuts
5	76.5	77.1	47	70	0.06	0.07
6	30.6	100	279	0	0.23	0.01
7	24.0	72.0	2298	172	2.29	0.32
8	34.6	71.6	7060	855	14.0	2.13
9	38.1	78.4	7122	369	20.9	2.12
10	31.4	74.6	1476571	60559	(0.3)	164
11	41.2	66.5	558102	159407	(0.8)	1331
12	22.3	49.6	264203	200180	(2.7)	(1.3)

Table 2: Dense graphs

result in a large reduction of the integrality gap at the root node. In fact, two instances are solved without branching when cuts are added. Based on these computations, we can state that the survivability cuts developed in the previous section improve the performance of the solution process significantly.

Size n	Root impr.		B&B nodes		Time/(End Gap)	
	Def	Surv cuts	Def	Surv cuts	Def	Surv cuts
5	83.6	100	6	0	0.02	0.01
6	23.2	95.2	45	10	0.06	0.04
7	36.0	85.6	88	51	0.08	0.07
8	43.5	94.1	50	0	0.06	0.03
9	33.0	77.3	8672	477	13.2	1.12
10	26.3	55.1	58973	27752	73.6	39.6
11	31.6	59.8	296569	31545	784	93.9
12	12.4	44.1	803131	781598	(1.3)	(0.7)

Table 3: Sparse graphs

5 Conclusions

We introduced a new methodology for designing survivable networks that explicitly introduces slack on directed cycles of a network. We first pre-

sent a hierarchical framework, and then extended it in a model that makes all routing and capacity decisions simultaneously. Even though one class of variables was exponential in size, we priced them exactly using a column generation scheme. We also studied the polyhedral structure of the mixed integer programming model and developed strong cutting planes that were quite effective in practice. Finally, we confirmed the advantages of the new model and the effectiveness of the cuts by implementing them in a branch-and-cut-and-price algorithm. The integrated approach provided savings of about 10% over the hierarchical scheme. The survivability cuts improved the computations significantly.

Our experiments suggest that simultaneous routing of flow and cycles is an computationally effective way for designing survivable networks. At the same time, we can possibly reduce the capacity requirements further by considering other failure-flow patterns, for instance, by using chords of the directed cycles as well. We will explore the pricing complexity and polyhedral structure of this more complicated failure pattern in a subsequent paper.

References

- [1] Ahuja, R. K., Magnanti, T. L., and J. B. Orlin, **Network Flows: Theory, Algorithms, and Applications**, Prentice-Hall, Englewood Cliffs, NJ, (1993).
- [2] Alevras, D., Grötschel, M., and R. Wesälly, “Cost-efficient network synthesis from leased lines”, *Annals of Operations Research*, vol. 76, 1–20 (1998).
- [3] Altnkemmer, K., “Topological design of ring networks”, *Computers and Operations Research*, vol. 21, 421–431 (1994).
- [4] Atamtürk, A., “On capacitated network design cut-set polyhedra”, *Mathematical Programming*, vol. 92, 425–437 (2000).
- [5] Atamtürk, A. and D. Rajan, “On splittable and unsplittable flow capacitated network design arc-set polyhedra”, *Mathematical Programming*, vol. 92, 315–333 (2001).

- [6] Barahona, F., "Network design using cut inequalities", *SIAM Journal on Optimization*, vol. 6, 823–837 (1996).
- [7] Bienstock, D. and O. Günlük, "Capacitated network design - Polyhedral structure and computation", *INFORMS Journal on Computing*, vol. 8, 243–259 (1996).
- [8] Brockmüller, B., Günlük, O., and L. A. Wolsey, "Designing private line networks - Polyhedral analysis and computation", CORE Discussion Paper 9647, Université Catholique de Louvain, (1996).
- [9] Bienstock, D. and G. Muratore, "Strong inequalities for capacitated survivable network design problems", *Mathematical Programming*, vol. 89, 127–147 (2000).
- [10] Chung, S. H., King, H. G., Yoon Y. S., and D. W. Tcha, "Cost-minimizing construction of a unidirectional SHR with diverse protection", *IEEE Transactions on Networking*, vol. 4, 921–928 (1996).
- [11] Goldschmidt, O., Laugier, A., and E. V. Olinick, "SONET/SDH ring assignment with capacity constraints", *Discrete Applied Mathematics*, to appear.
- [12] Grover, W. D and R. G. Martens, "Optimized design of ring-mesh hybrid networks", *Proceedings of IEEE/VDE Design of Reliable Communication Networks 2000*, 291–297 (2000).
- [13] Grover, W. D. and D. Stamatakis, "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 (1998).
- [14] Iri, M., "On an extension of the max-flow min-cut theorem to multi-commodity flows", *Journal of the Operations Research Society of Japan*, vol. 13, 129–135 (1971).
- [15] Luss, H., Rosenwein, M. B., and R. T. Wong, "Topological network design for SONET ring architecture", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 28, 780–790 (1998).
- [16] Magnanti, T. L., Mirchandani, P., and R. Vachani, "The convex hull of two core capacitated network design problems", *Mathematical Programming*, vol. 60, 233–250 (1993).
- [17] Magnanti, T. L., Mirchandani, P., and R. Vachani, "Modeling and solving the two-facility capacitated network loading problem", *Operations Research*, vol. 43, 142–157 (1993).
- [18] Soriano, P., Wynants, C., Séguin, R., Labbé, M., Gendreau, M., and B. Fortz, "Design and dimensioning of survivable SDH/SONET networks", in: **Telecommunications Network Planning**, eds. Sansó, B. and P. Soriano, Kluwer Academic Publishers, Netherlands, 147–168 (1998).
- [19] Wolsey, L. A, **Integer Programming**, John Wiley and Sons, New York, NY, (1998).
- [20] Xiong, Y. and L. G. Mason, "Restoration strategies and spare capacity requirements in self-healing ATM networks", *IEEE/ACM Transactions on Networking*, vol 7, 98–110 (1999).

Alper Atamtürk is an Assistant Professor of Industrial Engineering and Operations Research at the University of California, Berkeley. He holds B.S. and M.S. degrees in Industrial Engineering from Bilkent University, Turkey, and a Ph.D. from the Georgia Institute of Technology. His research interests include computational optimization, integer programming, logistics of production, distribution, transportation, telecommunication systems.

Deepak Rajan received the B.Tech degree from Indian Institute of Technology at Madras, India in 1999, and the M.S degree from University of California at Berkeley, in 2001. Since 2001, he has been a Ph.D Candidate at the Department of Industrial Engineering and Operations Research at the University of California at Berkeley. His research interests include integer programming, computational optimization, network design problems and stochastic combinatorial optimization.