Building Edge-Failure Resilient Networks

C. Chekuri[1] and A. Gupta[1] and A. Kumar[2] and J. Naor[3] and D. Raz[3]

[1] Lucent Bell Labs, 600 Mountain Ave., Murray Hill, NJ 07974. {chekuri,anupamg}@research.bell-labs.com.
[2] Computer Science Dept., Cornell University, Ithaca, NY 14853. amitk@cs.cornell.edu.
[3] Computer Science Department, Technion, Israel Institute of Technology, Haifa 32000, Israel. {naor,danny}@cs.technion.ac.il.

Part of this work was done while the last three authors were visiting Lucent Bell Labs.

Abstract. We consider the design of resilient networks that are fault tolerant against single link failures. Resilience against single link failures can be built into the network by providing backup paths, which are used when an edge failure occurs on a primary path. We consider several network design problems in this context: the goal is to provision primary and backup bandwidth while minimizing cost. Our models are motivated by current high speed optical networks and we provide approximation algorithms for the problems below.

The main problem we consider is that of *backup allocation*. In this problem, we are given an already provisioned (primary) network, and we want to reserve backup capacity on the edges of the underlying network so that all the demand can be routed even in the case of a single edge failure. We also consider a variant where the primary network has a tree topology and it is required that the restored network retains the tree topology. We then address the problem of simultaneous *primary and backup allocation*, in which we are given specifications of the traffic to be handled, and we want to both provision the primary as well as the backup network. We also investigate the online model where the primary network is not known in advance. Demands between source-sink pairs arrive online and the goal is to provide a primary path and set of backup edges that can be used for restoring a failure of any of the primary edges.

1 Introduction

Fault tolerance in networks is an important and well studied topic with many applications. Telephone networks and other proprietary networks adopt a variety of techniques to provide reliability and resilience to network failures and have been in use for many years now. On the other hand data networks such as the Internet have very little centralized fault tolerance. Instead, the network relies on the routing protocols that adapt to failures by sending traffic on alternate paths. This has been acceptable since there are no guarantees on the quality of service on the Internet. With maturity of the Internet, many applications now require quality of service guarantees. The emergence of very high capacity optical networks has enabled the move towards providing users with their own virtual private networks (VPNs). Several networks are accommodated on the underlying high capacity optical network by splitting the available bandwidth among them. Although this approach helps in providing QoS to applications and users, fault tolerance becomes a very critical issue: failure of a single high capacity link can disrupt many VPNs that use the link.

One way to provide VPN over optical networks is using MPLS [9]. Survivability issues of IP over optical networks are discussed in [24] and [12], and restoration in MPLS tunnels are discussed in [19] and [20]. In many cases the speed and the capacity of the links do not allow, unlike the Internet, to rely on the routing protocol to successfully reroute traffic on alternate routes *after* the failure. Thus, one needs to provision the network in advance to handle failures. This places two constraints on these networks: 1) resources for re-routing traffic should be reserved at the same time the sub-networks are provisioned, and 2) the routing protocol should be simple both for the regular routing and when a fault occurs.

For the reasons mentioned above, there has been much recent interest in obtaining algorithmic solutions for problems of guaranteeing resilience against failures. A variety of failure and recovery models have been proposed. It is not feasible to give even an overview of all the models and their intricacies, hence we mention only a few high level assumptions that we make in this paper. The precise model is given later in this section. One central assumption we make is the *single* link failure assumption, that is we assume that at most one link can fail at any particular time. This is a common assumption that seems to work reasonably well in practice. Further, the resulting optimization problems are already hard and it is useful to obtain insight into this case.

Clearly, resilience against single edge failures can be built into the network by providing backup paths, which are used when an edge failure occurs on the primary path. However, note that these backup paths could intersect each other and *share* the same amount of bandwidth, provided they are used for the failures of *different* edges in the network. This multiplexing is one of the factors that makes this problem especially difficult; we shall spell out some of the others as we explain the models and our results.

Finally, most of our results are for an *uncapacitated* network. In other words, we assume that capacities of the underlying network are unlimited and there is a cost on each edge per unit of bandwidth. Although this assumption is not true for any practical network, we make it for two reasons. First, we believe it is a reasonable approximation since the capacities of the underlying network are usually much larger than the capacity of any *single* VPN. Second, the capacitated versions are much harder in theory and we believe that the domain in which they are hard does not apply to real settings. For example, the disjoint paths problem is notoriously hard for small capacities, but it is much easier if the capacities of the edges are sufficiently large compared to the individual demands. See [6, 11, 17] for similar assumptions.

We consider several network design problems with the above assumptions. Though our problems have similarities to traditional network design problems, they also differ in some respects. Our contributions include providing models and building upon existing techniques to give algorithms for these new problems. We hope that our techniques and ideas will be useful in related contexts.

The first problem we consider is that of **Backup Allocation**. In this problem, we are given an already *provisioned (primary) network*, and we want to reserve *backup* capacity on the edges of the underlying network so that all the demand can be routed even in the case of an edge failure. At this point, we point a requirement of the network: the restoration has to be handled *locally*; i.e., if edge $e = (i, j)$ carrying $u(e)$ bandwidth fails, there must be a *single* path $P(e)$ in the backup network between i and j with capacity at least $u(e)$, which stands in for the edge e.

Local restoration is important for timing guarantees, otherwise it could take too much time before other portions of the network are aware of a failure at e; it is also useful since it does not require any of the other paths being currently used in the network to be changed or rerouted. It is imperative that there is a *single* path between u and v that routes all of $u(e)$; having a backup network that is able to push the right amount of "flow" would not suffice. This is necessary in optical networks, where splitting the traffic is not feasible. This is also the reason that the traffic between two hosts in the originally provisioned network is routed on a single path. (As an aside, the reader curious about the actual mechanisms of efficient local restoration of paths is pointed to the literature on MPLS [9].)

The second problem we consider is that of **Primary and Backup Allocation**. In this paper we consider two cases of the problem. In the first case, we are given specifications of the traffic to be handled, and we want to provision both the primary as well as the backup network. The second case is related to an online version of the problem, where demands arrive one by one. Here, we must find both a primary path and a backup path between a pair of terminals $\{s, t\}$, but where some of the edges may have already been chosen as part of a backup path for previous demands. Since we are allowed to share those edges between backups of different pairs of terminals, we model this by allowing different costs for an edge depending on whether it is a part of a primary or a backup path.

1.1 Models and results

We now give detailed and precise formulations of the problems studied and results obtained in this paper.

Backup Allocation: In this paper, we look at (undirected) *base networks* $G = (V, E)$ with edge costs c_e. In backup allocation, we are given a provisioned network $G^p = (V^p, E^p)$, with each edge $e \in E^p$ having provisioned capacity $u^p(e)$. The objective is to find an edge set $E^b \subseteq E$ (which could intersect with E^p) and backup capacities u^b for these edges, so that for each $e = (u, v) \in E^p$, there is a path $P(e) \in E^b \setminus \{e\}$ between its endpoints u and v on edges of capacity at least $u^p(e)$. This path $P(e)$ can be used to locally restore e lest it fail.

In Section 3, we describe an $O(1)$ approximation algorithm for this problem. We first examine the uniform capacity case, that is when $u^p(e) = 1$ for all $e \in E^p$. This special case is similar to the Steiner network problem [23, 15, 18], in that it prescribes connectivity requirements for vertex pairs, except that now there is a forbidden edge for each pair. We give an algorithm to handle this in Section 2, and then use scaling to handle the general case with non-uniform primary capacities.

Primary and Backup Allocation: The most common model for specifying traffic characteristics is the *point-to-point demand model*, where a *demand* matrix $D = (d_{ij})$ gives demands between each pair of terminals, and the objective is to find the cheapest network capable of sustaining traffic specified by D. In the uncapacitated case which is of interest here, allocating the primary can be done trivially by routing flow on shortest-paths between the terminals.

Considering that good estimates are often not known for the pairwise demands in real networks, Duffield et al. [10] proposed an alternate way to specify traffic patterns, the so-called *VPN hose model*. In its simplest form, each terminal i is given a *threshold* $b(i)$, and a symmetric demand matrix $D = (d_{ij})$ is called *valid* if it respects all thresholds, i.e., if $\sum_j d_{ij} \leq b(i)$ for all i. The primary network is specified by a vector u^p indicating the bandwidth allocated on the various edges, and also paths P_{ij} on which all the flow between terminals (i, j) takes place unsplittably. Feasibility of a solution implies that for each valid demand matrix (d_{ij}),

$$\sum_{i<j} d_{ij}\ \chi(P_{ij}) \leq u^p,$$

where $\chi(P)$ is the characteristic vector of P, and the sum is a vector sum. Provisioning the primary network in the hose model was studied in [16], where among other results, an optimal algorithm was given when the provisioned network was required to be a tree; it was also shown that this tree is a 2-approximation for general networks (without the tree restriction).

These are just some ways to specify the traffic requirements; given a primary network, the backup network is defined just as before. In this paper, we show that if there is a α-approximation algorithm for allocating the primary, there is an $O(\alpha \log n)$ approximation for both primary and backup allocation. The simple two-stage algorithm that achieves this first allocates the primary primary network G^p using the α-approximation algorithm, and then uses the algorithm of Section 3 to find a near-optimal backup network for G^p.

Tree networks: Simplicity, along with good routing schemes and error recovery, make trees particularly attractive. This prompted [16] to give an algorithm for primary allocation in the VPN hose model which outputs the optimal tree (which is within a factor 2 of the optimal network). However, when some edge e in this tree fails and is locally restored by $P(e)$, the new network may no longer be a tree. For some applications, and also for simplicity of routing schemes, it is convenient that the network remains a tree at all times, even after restoration. In Section 5, we study the problem of allocating backup while retaining the tree topology of a given primary network. We show that this problem is hard to approximate within $\Omega(\log n)$. We also give a backup allocation algorithm whose cost is $O(\log^2 n)$ times the optimal cost of primary and backup allocation.

The Online Problem: In practical applications, the demands often appear in an online manner, i.e., new demands for both primary and backup paths between pairs of nodes arrive one by one. Here we need to solve the primary and backup allocation problem in the point-to-point demand case, i.e., when there is a single source-sink pair in the network with a given demand. Concretely, the goal is to construct both a primary path and a set of backup edges that can be used for restoring a failure of any of the primary edge. As explained before, a backup edge can be used to back up more than one primary edge, and hence some edges may have already been paid for by being on a previous backup path. We model this by allowing different *primary costs* and *backup costs* for an edge, depending on the purpose for which we will use this edge. Clearly, the primary cost of an edge should be at least as high as the backup cost. We present a simple 2-approximation algorithm for this case. We then present two natural linear programming formulations of the problem and show that one formulation dominates the other for all instances. Note, that we are considering the local optimization problem that needs to be solved each time a new demand arrives, and do not aim at performing the usual competitive analysis where the online algorithm is compared to the best offline solution.

Related Work: There have been several papers on (splittable) flow networks resilient to edge-failures; see, e.g., [6, 7, 11]. The papers [19, 20] formulate the online restoration problem as an integer program, and give some empirical evidence in favor of their methods. The paper of [17] considers backup allocation in the VPN hose model and gives a constant-factor approximation when accounting *only* for the cost of edges not used in the primary network. The paper [1] looks at the problem of limited-delay restoration; however, it does not consider the question of bandwidth allocation.

The problem of survivable network design has also been investigated extensively (see, e.g.,[2] and the references therein). Most of this work has been focused on obtaining strong relaxations to be used in cutting plane methods. In fact, the linear programs we use have been studied in these contexts, and have been found to give good empirical performance. For more details on these, and on polyhedral results related to them, see [3–5, 8]. In contrast to most of these papers, we focus on worst-case approximation guarantees, and our results perhaps explain the good empirical performance of relaxations considered in the literature. Our models and assumptions also differ in some ways from those in the literature. We are interested in local restoration, and not necessarily in end-to-end restoration. This allows our results to be applicable to the VPN hose model as well, in contrast to the earlier literature, which is concerned with the point-to-point model. We also focus on path restoration as opposed to flow restoration. On the other hand, we do consider a simpler model and limit ourselves only to the case of uncapacitated networks.

2 Constrained Steiner Network Problem

Recall that our model assumes the following: when link $e = (u, v)$ fails, the backup path for (u, v) *locally* replaces e, i.e, any path that used e now uses the backup path in place of e without altering the rest of the path. Given provisioned network G^p, the Backup problem seeks to find a set of edges such that for each $(u, v) \in G^p$ there is a backup path that does not use (u, v) itself. Note that we can share edges in the backup paths for different edges. If all the capacities are the same, this is similar in spirit to traditional network design problems. Motivated by this we study a variant of the Steiner network problem that is described below.

In the Steiner network problem we are given an undirected graph $G = (V, E)$ and cost function $c : E \to \mathbb{R}^+$. We are given a *requirement* $r_{ij} \in \mathbb{Z}^+$ for pairs of vertices $(i, j) \in V$. (We can assume that $r_{ij} = 0$ for pairs (i, j) for which there is no requirement.) The goal is to select a minimum cost set of edges $E' \subseteq E$ such that there are r_{ij} edge-disjoint paths between i and j in E'. In a seminal paper, Jain [18] gave a 2-approximation for this problem, improving upon the earlier $2H_{r_{max}}$ approximation [23] where r_{max} is the largest requirement.

For our application, we add the constraint that for pairs of vertices (i, j), $E' - \{(i, j)\}$ must support r_{ij} edge-disjoint paths between i and j. Note that though we are not allowing the edge (i, j) to be used in connecting i and j, (i, j) could be used in E' to connect some other pair (i', j'). We will refer to the Steiner network problem as the SN problem, and our modified problem as the CSN (constrained SN) problem.

We show that any α-approximation algorithm for SN can be used to solve the CSN problem with an additional loss of a factor of 2 in the approximation ratio. The algorithm is simple and is given below.

- Let I_1 be the instance of SN with requirement r on G. Solve I_1 approximately, and let E' be the set of edges chosen.
- Define a new requirement function r' as follows. For $(i, j) \in E'$ such that $r_{ij} > 0$, set $r'_{ij} = r_{ij} + 1$, else set $r'_{ij} = r_{ij}$.
- Let I_2 be the instance of SN on G with requirement function r' and with the cost of edges in E' reduced to zero. Let E'' be an approximate solution to I_2. Output $E'' \cup E'$.

It is easy to see that the above algorithm produces a feasible solution. Indeed, if $(i, j) \notin E'$ then $E' - \{(i, j)\}$ contains r_{ij} edge-disjoint paths between i and j. If $(i, j) \in E'$ then E'' contains $r_{ij} + 1$ edge-disjoint paths between (i, j), and hence $E'' - \{(i, j)\}$ contains r_{ij} edge-disjoint paths.

Lemma 1. *The cost of the solution produced is at most* $2\alpha\,\text{OPT}$ *where* α *is the approximation ratio of the algorithm used to solve SN.*

Proof. It is easy to see that $\text{OPT}(I_1) \leq \text{OPT}$, and hence $c(E') \leq \alpha\,\text{OPT}$. We claim that $\text{OPT}(I_2) \leq \text{OPT}$. Indeed, if $A \subseteq E$ is an optimal solution to I, then $A \cup E'$ is feasible for requirements r'. Therefore, $c(E'' - E') \leq \alpha\,\text{OPT}(I_2) \leq \alpha\,\text{OPT}$, and $c(E'' \cup E') \leq 2\alpha\,\text{OPT}$.

2.1 Integrality Gap of LP Relaxation for CSN

We used the algorithm for the Steiner network problem as a black box in obtaining the above approximation ratios. Consider the following integer linear programming formulation for CSN, where x_e is the indicator variable for picking edge e in the solution. For compactness we use the following notation to describe the constraints. We say that a function $\bar{x}$ on the edges *supports* a flow of f between s and t if the maximum flow between s and t in the graph with capacities on the edges given by $\bar{x}$ is at least f.

$$\begin{aligned} \min \quad & \textstyle\sum_e c_e x_e & & \text{(IP1)} \\ \text{s.t.} \quad & \bar{x} \in \{0,1\}^{|E|} \text{ supports } r_{ij} \text{ flow between } (i,j) \text{ in } E - \{(i,j)\} & & \text{for all } i,j \end{aligned}$$

We relax the integrality constrains to obtain a linear program (LP1), and claim that the integrality gap is at most 4, the same as the approximation guarantee for the algorithm above. Jain [18] showed that the integrality gap of the natural cut formulation for SN is 2. The following linear programming relaxation (LP2) for SN is flow based and is equivalent to the cut formulation, and hence its integrality gap is also 2.

$$\begin{aligned} \min \quad & \textstyle\sum_e c_e x_e & & \text{(LP2)} \\ \text{s.t.} \quad & \bar{x} \in [0,1]^{|E|} \text{ supports } r_{ij} \text{ flow between } (i,j) \text{ in } E & & \text{for all } i,j \end{aligned}$$

Note that the optimal solutions to (LP2) for the instances I_1 and I_2 cost no more than an optimal solution to (LP1) for I. This, combined with the fact that (LP2) has an integrality gap of at most 2, gives the following result.

Lemma 2. *The integrality gap of (LP1), the LP for the CSN problem is upper bounded by* 4.

3 Backup Allocation

In this section we show an $O(1)$ approximation for the problem of computing the cheapest backup network for a given network. Let $G = (V, E)$ be the underlying base network and let $G^p = (V^p, E^p)$, a subgraph of G, denote the primary network. We are also given a real valued function u^p on the edge set E that gives the primary bandwidth provisioned on the edges. Our goal is to find an edge set $E^b \subseteq E$ and a function $u^b : E^b \to \mathbb{R}^+$ such that E^b backs up the network G^p for single link failures. Note that we are working in the uncapacitated case which implies that we can buy as much bandwidth as we want on any edge of E and the cost for buying b units of bandwidth on edge e is $b \cdot c_e$, where c_e is the cost of e.

Let $u^p_{\max} = \max_{e \in E^p} u^p(e)$. Our algorithm for backup allocation given below is based on scaling the capacities and solving the resulting uniform capacity problems separately.

– Let $E_i^p = \{e \in E^p \mid u^p(e) \in [2^i, 2^{i+1})\}$. For all $e \in E_i^p$, round up $u^p(e)$ to 2^{i+1}.
– For $1 \leq i \leq \lceil \log u_{\max}^p \rceil$, *independently* backup E_i^p.

Let E_i^b be the edges for backing up E_i^p and u_i^b be the backup bandwidth on E_i^b. Note that rounding up the bandwidths of E_i^p causes the the backup allocation problem on E_i^p to be a uniform problem. The lemma below states that solving the problems separately does not cost much in the approximation ratio.

Lemma 3. *Let α be the approximation ratio for the uniform capacity backup allocation problem. Then there is an approximation algorithm for the backup allocation problem with ratio 4α.*

Proof. Let E^{r*} be an optimal solution for backup allocation, with u^{r*} being the bandwidth on E^{r*}. For $1 \leq i \leq \lceil \log u_{\max}^p \rceil$ construct solutions E_i^{r*}, where $e \in E_i^{r*}$ with capacity $u_i^{r*}(e) = 2^{i+1}$ if $u^{r*}(e) \geq 2^i$. Clearly $\sum_i u_i^{r*}(e) \leq 4u^{r*}(e)$, and so $\sum_i c(E_i^{r*}) \leq 4c(E^{r*})$. However, note that E_i^{r*} is a feasible backup for E_i^p, since every edge in E^{r*} of bandwidth at least 2^i lies E_i^{r*} with bandwidth 2^{i+1}. Hence, for each i, using the approximation algorithm for the uniform case for E_i^p would give us a solution with cost at most $\alpha c(E_i^{r*})$. This completes the proof.

However, the backup allocation problem for the uniform bandwidth case can be approximated to within $\alpha = 4$. Given a set of edges E^p with uniform bandwidth $u^p(e) = U$ that need to be backed up, the problem can be scaled so that $u^p(e) = 1$ for $e \in E^p$. This is just a problem of finding, for $(i,j) \in E^p$, a path between i and j that does not use the edge (i,j) itself, which in turn is the problem described in Section 2 with $r_{ij} = 1$ for $(i,j) \in E^p$. Combining this with Lemma 3, we get a 16-approximation algorithm for the backup allocation problem.

The ratio of 4α can be improved to $e\alpha$ by randomness: instead of grouping by powers of 2, grouping can be done by powers of e (with a randomly chosen starting point). This technique is fairly standard by now (e.g., [21, 13]) and the details are deferred to the final version.

Theorem 1. *Given any G^p, there is a $4e \simeq 10.87$-approximation for the backup allocation problem for G^p.*

3.1 Integrality Gap of an LP relaxation

We showed an $O(1)$ approximation for the backup allocation problem. We now analyze the integrality gap of a natural LP relaxation for the problem and show that it is $\Theta(\log n)$. This will allow us to analyze an algorithm for simultaneous allocation of primary and backup networks in the next section. The LP formulation uses variables y_e which indicate the backup bandwidth bought on edge e.

We relax the requirement that the flow uses a *single* path.

$$\begin{array}{lll} \min & \sum_e c_e y_e & \text{(LP3)} \\ \text{s.t.} & \bar{y} \text{ supports } u^p_e \text{ flow between } (i,j) \text{ in } E - \{e\} & \text{for all } e \in E^p \\ & y_e \geq 0 & \end{array}$$

We now analyze the integrality gap. Recall the definition of E^p_i as the set of edges in E^p such that $u^p(e) \in [2^i, 2^{i+1})$. As before we round up the bandwidth of these edges to 2^{i+1}. Let $x_e(i) = \min\{1, y_e/2^i\}$. Note that $x_e(i) \in [0,1]$. We claim the following.

Proposition 1. *The variables $x_e(i)$ are feasible for the uniform bandwidth backup allocation problem induced by E^p_i where the bandwidths are scaled to* 1.

From the analysis in Section 2.1 it follows that the integrality gap of (LP1), the LP for the uniform bandwidth problem is at most 4. Hence we can find a solution that backs up the edges in E^p_i with cost at most $4\sum_e c_e y_e$. Since we have to only look at $\lceil \log u^p_{\max} \rceil$ values of i, there is a solution that backs up all edges in E^p with cost at most $4 \log u^p_{\max} \sum_e c_e y_e$. We can make the upper bound on the integrality gap $O(\log n)$ via a simple argument. We set $x_e(i) = 0$ if $y_e/2^i \leq 1/n^3$, otherwise we set $x_e(i) = \min\{1, (1 + 1/n)y_e/2^i\}$. It is straightforward to argue that Proposition 1 still holds for the variables $x_e(i)$ defined in this modified fashion. The cost goes up by a $(1 + 1/n)$ factor. Each edge e participates in the backup of at most $O(\log n)$ groups E^p_i, hence the overall cost is at most $O(\log n)$ times the LP cost. This gives us the following theorem.

Theorem 2. *The integrality gap of (LP3) is $O(\min\{\log n, \log u^p_{\max}\})$.*

The following theorem shows that our analysis is tight.

Theorem 3. *The integrality gap of (LP3) is $\Omega(\log n)$.*

Proof. We construct a graph G with the required gap as follows. The graph consists of a binary tree T rooted at r with some additional edges. The cost of each edge in T is 1. The additional edges go from leaves to the root and each of them is of cost d, where d is the depth of T. Primary bandwidth is provisioned only on the edges of T and is given by $u^p(e)$: for an edge e at depth $d(e)$, $u^p(e) = 2^d/2^{d(e)}$. Backup bandwidth allocation defined by the following function $u^b(e)$ is feasible for (LP3): $u^b(e) = 1$ for each edge e that goes from a leaf to the root and $u^b(e) = u^p(e)$ for each edge of T. It is easy to check that the cost of this solution is $O(d2^d)$. We claim that any path solution to the backup of T in G has a cost of $\Omega(d^2 2^d)$. To prove this claim we observe that in any path backup solution the number of edges from the leaves to the root of backup capacity 2^{d-i} is at least 2^i. This follows since there are 2^i edges of capacity 2^{d-i} in T each of which could fail and each of them requires a backup edge from a leaf in its subtree to the root. The subtrees are disjoint and hence these back up edges cannot be shared. We set d to be $\log n$ to obtain the desired bound.

We note that the primary network in the above proof is valid for both the point to point demand model and the VPN hose model. That the former is true is clear: every edge implicitly defines a point to point demand between its end points of value equal to the primary bandwidth allocated to the edge. To see that the above primary network is valid in the VPN hose model consider the leaves of T as demand points, each with a bandwidth bound of 1.

4 Simultaneous Primary & Backup Allocation

In this section we examine the problem of simultaneously building a primary network and the backup network of minimum overall cost. We have already seen an $O(1)$ approximation algorithm to provide backup given the primary network. We adopt a natural two-phase strategy where we build the primary network first and then build a backup network for it. If α is the approximation guarantee for the problem of building the primary network then we obtain an $O(\alpha \log n)$ approximation for building the primary and backup networks. For the two primary networks of interest, namely the pairwise demand model and the VPN hose model, we have $O(1)$ approximation algorithms for building the primary network, hence we obtain an $O(\log n)$ approximation for the combined problem.

An $O(\log n)$ approximation: We analyze the two-stage approach for primary and backup allocation. Let G^p be the subgraph of G that is chosen in this first step. We provide backup for this network using the algorithm described in Section 3. To analyze this algorithm we use the LP relaxation (LP3) for the backup allocation problem. In the following lemma we will be using extra capacity on the edges of provisioned network itself. Note that this is allowed.

Lemma 4. *Let u^p be any solution to the primary problem. Let u^{p*} and u^{r*} be the primary and backup in some optimal solution. Then, $u^p + u^{p*} + u^{r*}$ is a feasible solution for (LP3), the LP relaxation for the backup of u^p.*

Proof. We assume that the solution u^p is minimal. Let $e = (i, j)$ be such that $u^p(e) > 0$. Since we have a minimal solution it implies that there exists some traffic between terminals that requires at least $u^p(e)$ flow on e. Let the flow paths that use e in that traffic be $P_1, P_2, \ldots, P_k$ and let f_i be the flow on P_i. In the graph G^p, let X be the set of terminals that are connected to i if the edge (i, j) is removed from the paths, and let Y be those terminals that are connected to j. Let P_h connect terminal x_h to terminal y_h where $x_h \in X$ and $y_h \in Y$. We need to argue that in the graph with out the edge (i, j), we can a send a flow of value $u^p(e)$ from i to j with capacities defined by $u^p + u^{p*} + u^{r*}$. We do this as follows. We send flow f_i from i to each x_i using capacities $u^p(e)$. Since the optimum solution is resilient against single edge failures, for $1 \leq i \leq k$, there must exist flow paths that can route a flow of f_i units from x_i to y_i, none of which use the edge (i, j). Since $\sum_i f_i = u^p(e)$, it follows that we can route a flow of $u^p(e)$ from i to j without using (i, j) in the capacitated graph defined by $u^p + u^{p*} + u^{r*}$.

Theorem 4. *The two-stage approach yields an $O(\alpha \log n)$ approximation to the combined primary and backup allocation problem where α is the approximation ratio for finding the primary allocation.*

Proof. Let P be the cost of the primary allocation and B the cost of backup allocation in the two stage approach. From the approximation guarantee on finding P, we have $P \leq \alpha\,\mathrm{OPT}$. From Lemma 4, it follows that there is a feasible (LP3) relaxation for the backup allocation problem of value at most $P + \mathrm{OPT}$, hence at most $(\alpha+1)\,\mathrm{OPT}$. From Theorem 2, the backup solution we obtain is at most $O(\log n)$ times the LP value. Hence, $B = O(\alpha \log n)\,\mathrm{OPT}$ and the theorem follows.

Corollary 1. *There is an $O(\log n)$ approximation for the combined primary and backup allocation problems for the pairwise demand model and the VPN hose model.*

It turns out that the two-stage approach loses an $\Omega(\log n)$ factor even if the first step obtains a primary network of optimum cost; the example in the proof of Theorem 3 demonstrates this.

5 Backup for Tree Networks

In this section, we consider the case when the provisioned network T is a tree. Furthermore, we require that when an edge e fails, the network $T - \{e\} + P(e)$ also be a tree. We show that finding a minimum cost backup network in this case is at least as hard as the group Steiner problem [14] on trees, which in turn is $\Omega(\log n)$-hard. We also give an algorithm whose approximation ratio is within constant factors of the approximation ratio for the group Steiner problem on trees, which at most $O(\log^2 n)$ due to Garg et al. [14].

There is a slight difference in the manner in which we define the approximation ratio in this section. Let $T = G^p = (V^p, E^p)$ be the provisioned network as usual, and let E^b be the backup edges, with u^p and u^b be the primary and backup bandwidths as usual. We shall consider the cost of the solution to be $\sum_e (u^p(e) + u^b(e))$, and our approximation will be with respect to this measure. The problem turns out to be hard, even when the measure of goodness is taken to be $\sum_e u^b(e)$. We omit the proof of the following theorem.

Theorem 5. *The tree backup problem is at least as hard as the group Steiner problem on trees.*

5.1 Approximation Algorithm

Let $T = (V, E)$ be the already provisioned tree. When an edge e fails, it splits T into two components, and $P(e)$ must be a path between these two components which is internally node disjoint from the tree T. We must reserve enough bandwidth on the edges in the graph such that the tree formed thus can support traffic between the demand nodes.

Our basic strategy is the same as in Section 3: Let E_i be the set of edges in T on which the bandwidth u^p lies in the interval $[2^i, 2^{i+1})$. Let $u^p_{\max}$ lie in the interval $[2^s, 2^{s+1})$. Our algorithm will proceed in stages — in the i^{th} stage, we will "protect" the edges in E_{s+1-i}. When we have already protected the edges in $E_{i+1}, \ldots, E_s$ by reserving bandwidth on some edges, we contract edges in $E_{i+1} \cup \ldots \cup E_s$. This will not affect our performance by more than a constant, since the bandwidth we may later reserve on some edge e in this set will be at most $\sum_{j \leq i} 2^{i+1} \leq 4u^p(e)$. Let T_i and G_i be the resulting tree and base graph after the contraction. We shall now consider protecting the edges in E_i, using the edges of G_i.

The procedure has a few conceptual steps, so let us describe these here, with the intent of convincing the reader. The proofs closely follow the ideas mentioned here, but are omitted due to lack of space.

- It can be seen that the edges of E_i form a "spider"; i.e., there is a root r, and a collection of paths $\{P_i\}_{i=1}^k$ which meet at r but are otherwise node-disjoint. This is because of the structure of the VPN trees as given in [16], where the allocated demand on the edges from the root to the leaves is non-increasing.
- The graph G_i can be transformed into a graph G'_i of which T_i is a spanning tree. All non-T_i edges go between vertices of T_i. This can be done so that the backup solutions for T_i in G_i and in G'_i can be translated between each other with only a constant factor difference in cost.

  This transformation is based on the following idea: we consider the backup edges for E_i which do not belong to T_i; these must form a tree (else we could drop one of the edges). Take an Eulerian tour of this tree and consider the subpaths between consecutive nodes in T_i; these can be replaced by new edges with the appropriate weight, and all other vertices can be disposed off. Note that the optimal solution in this new instance will be at most twice the optimal solution before, since the Eulerian tour counted every edge twice. One has to take care that there may be backup edges in T_i itself, and accounting for these requires a slightly more complicated argument, which we omit here.
- We now have a simpler problem: a graph G'_i, with a spanning tree T_i in which we need to find a tree backup for the edges of E_i, which form a spider. Let r_i be the root of T_i, and let P_j's be the paths of the spider. Also let $T_{i,j}$ be the subtree of T_i hanging off P_j. (See Figure 1 for a picture.) Call a non-tree edge a *back* edge if both its end points belong to the same tree $T_{i,j}$, and a *cross edge* otherwise. For example, the edge e_b is a back edge, and e_c a cross edge in the figure. Now each edge e of E_i has a *savior edge* $sav(e)$ which is used to connect the two components formed if e fails. A crucial fact is that if a cross edge from $T_{i,j}$ to $T_{i,j'}$ is a savior for some edge e in P_j, then it is a savior for all edges on P_j which are above e. Hence, fixing the lowest edge e in P_j whose savior is a cross edge implies that all edges above it are also saved by that same savior edge, and all edges below it on P_j must be saved by back edges. The cost $Q(e)$ of saving the rest by back edges depends

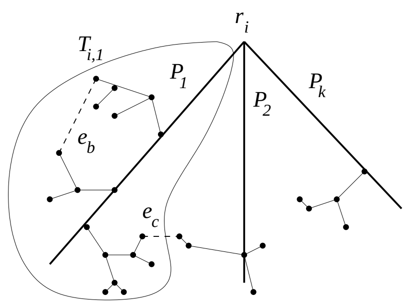

Fig. 1. The tree T_i, with tree edges shown in solid lines and non-tree edges in dotted ones.

on the portion of T_i attached to these edges and the back edges between this portion; note that this is entirely independent of the rest of the problem. Suppose we know, for each edge $e \in \cup_j P_j$, the cost $Q(e)$. (We shall discharge this assumption later.) Then the cost of backing up all the edges in E_i consists of the following: for each P_j, picking the single cross edge (say going to $T_{i,j'}$) which is going to be savior (and reserving 2^{i+1} capacity on it), and reserving 2^{i+1} capacity on the edges in $T_{i,j'}$ from the other end of this edge to the root r_i. Of course, we have to add the cost of saving the edges that were not saved by these cross edges to the solution.

– We now claim that this can be modeled as a minor variant of the group Steiner tree problem with vertex costs. Each vertex $v \in T_i$ which is the endpoint of some cross edge e from $T_{i,j}$ is belongs to the group S_j and has a "cost" $2^{i+1}c_e + Q(e')$, where e' is the lowest edge in P_j that can be saved by e. (Note that S_j must be a multi-set, with the vertex v occurring several times in S_j with various costs, if there are several such cross edges.) As a pedantic aside, there may be no such cross edge, and so r_i also belongs to S_j with cost equal to saving P_j entirely with back edges. This is done for every vertex and every value of j. Now the objective is to find the minimum cost subtree of T_i that contains the root and hits every group S_j at least once, where we also have to pay for the vertex of S_j picked in the tree. It is fairly easy to see that this can be transformed into a regular group Steiner tree problem, and the algorithm of Garg et al. [14] then gives us a $O(\log^2 n)$ approximation.
– There is one more assumption to be discharged: we have to show how to compute all the $Q(e)$. We will not be able to do this optimally, but we give a constant factor approximation for this as well. Since these are independent problems, let us consider the case when we want to find the cost of backing up P_1 using only back edges. We would like to just use the Eulerian trick done above to reduce the problem to edges only between vertices of P_1, and then find the least cost augmentation. The technical problem that arises is that that the optimal solution could be using edges in $T_{i,j} - P_j$, and

doing this trick naively could result in our paying several times for this reservation, when paying once would have sufficed. We can however show that we pay for no such edge more than twice, and hence the cost of the min-cost augmentation is off by a factor of at most 2. Of course, we can only compute the augmentation within a factor of 2 of optimal, and hence we can compute $Q(e)$ within a factor 4 of optimal.

6 The Online Optimization Problem

In this section we consider a unit-capacity MPLS primary and backup allocation problem which is motivated by the online problem of choosing the best primary and backup paths for demands arriving one by one. Suppose that we are given source and destination vertices, denoted by s and t, respectively. The goal is to simultaneously provision a primary path p from s to t and a backup set of edges q of minimum overall cost. Since we are dealing with a single source-sink pair we can scale the bandwidth requirement to 1, hence all edges have unit capacity, i.e., the primary and backup edge sets are disjoint. We require that for any failure of an edge $e \in p$, $q \cup p - \{e\}$ contains a path from s to t. We call this problem SSSPR (Single Source Sink Provisioning and Restoration). Note that this requirement is slightly different from the backup model discussed earlier in the paper; here, we do not insist on local restoration. The backup edges together with the primary edges are required to provide connectivity from s to t. This problem is in the spirit of the work of Kodialam and Lakshman [19, 20]. As explained before, we model the online nature of the problem by using two different costs. Formally, there are two non-negative cost functions associated with the edges: cost function c_1 denotes the cost of provisioning a primary edge the primary, and cost function c_2 denotes the cost of an edge when used as a backup edge. We assume that $c_1(e) \geq c_2(e)$ for all edges $e \in E$.

Let p be a primary path from the source s to the destination t. The following procedure due to Suurballe [22] computes a minimum cost backup set of edges for a given primary path p. The idea is to direct the edges on the path p in the "backward" direction, i.e., from t to s and set their cost to be zero. All other edges are replaced by two anti-symmetric arcs. For each edge e which is replaced by arcs a and a^-, the cost of both a and a^- is set to $c_2(e)$. We now compute a shortest path q from s to t. It can be shown that the edges of q that do not belong to p define a minimum cost local backup [22].

A 2-approximation algorithm for the SSSPR problem can be obtained as follows. First, find a shortest path p from s to t with respect to the c_1-cost function. Then, use Suurballe's [22] procedure to compute an optimal backup q to the path p with respect to the c_2-cost function. We show below that p and q together induce a 2-approximate solution.

Theorem 6. *The two stage approach yields a 2-approximation to SSSPR.*

Proof Sketch. Let OPT be the cost of an optimal primary and backup solution and let $P = \sum_{e \in p} c_1(e)$ be the cost of p and $Q = \sum_{e \in q} c_2(e)$ be the cost of q. It is clear that $P \leq$ OPT since we find the cheapest primary path. We next argue that $Q \leq$ OPT. Consider Suurballes [22] algorithm to find the optimum backup path for p. As described earlier, the algorithm finds a shortest path in a directed graph obtained from G and p. We can express the computation of this shortest path as a linear program L in a standard way – essentially as a minimum cost flow computation of sending one unit from s to t. The main observation is that any primary path p' and a q' that backs up p' yield a feasible solution to linear program L. We omit the formal proof of this observation but it is not difficult to see. In particular, this holds for the set of edges of p^* and q^*, where p^* is an optimal primary path and q^* is a set of edges that back up p^*. Therefore, it follows that $Q \leq \sum_{e \in p^* \cup q^*} c_2(e) \leq \sum_{e \in p^*} c_1(e) + \sum_{e \in q^*} c_2(e)$. Here is where we crucially use the assumption that $c_2(e) \leq c_1(e)$ for all e. Hence, $Q \leq$ OPT and the theorem follows.

Although we provide an approximation algorithm, we note that it is not known whether the problem is NP-hard or not.

6.1 Linear Programming Formulations for SSSPR

We provide two linear programming relaxations of SSSPR. The first formulation is based on cuts and the second formulation is based on flows. We show that the second formulation dominates the first one on all instances.

A *cut* in a graph G is a partition of V into two disjoint sets V_1 and V_2. The edges of the cut are those edges that have precisely one endpoint in both V_1 and V_2. Let T be a subgraph of G which is a tree. A cut (V_1, V_2) of G is a *canonical cut* of G with respect to T if there exists an edge $e \in T$, decomposing T into T_1 and T_2, such that $T_1 \subseteq V_1$ and $T_2 \subseteq V_2$.

Let p be a primary path from the source s to the destination t. It follows from Suurballe's [22] procedure that a set of edges q is a backup to a path p if it covers all the canonical cuts of p. This leads us to the following linear programming formulation which is based on covering cuts. For an edge e, let $x(e)$ denote the primary indicator variable and let $y(e)$ denote the backup indicator variable.

$$\begin{array}{rrcll}
\min & \sum_{e \in E} c_1(e) \cdot x(e) \ + \ c_2(e) \cdot y(e) & & & \text{(Cut-LP)} \\
\text{s.t.} & \sum_{e \in C} (x(e) + y(e)) & \geq & 2 & \text{for all } \{s,t\}\text{-cuts } C \\
& \sum_{e \in C} x(e) & \geq & 1 & \text{for all } \{s,t\}\text{-cuts } C \\
& x(e) + y(e) & \leq & 1 & \text{for all } e \in E \\
& x(e), y(e) & \geq & 0 & \text{for all } e \in E
\end{array}$$

It is not hard to see that the value of an optimal (fractional) solution to Cut-LP is a lower bound on the value of an optimal integral solution to SSSPR. We now present a second linear programming formulation of SSSPR which is based on flows. Our formulation relies on the following lemma.

Lemma 5. *Let p be a primary path from s to t and let q be a set of backup edges. Replace each edge from p and q by two parallel anti-symmetric unit capacity arcs. Then, two units of flow can be sent from s to t.*

This leads us to the following bidirected flow relaxation. We replace each edge by two parallel anti-symmetric unit capacity arcs. Denote by $D = (V, A)$ the directed graph obtained. The goal is to send two units of flow in D from s to t, one from each commodity, while minimizing the cost. Denote the two commodities by blue and red, corresponding to primary and backup edges, respectively. The cost of the blue commodity on an arc a (obtained from edge e) is equal to $c_1(e)$. The cost of the red commodity on an arc a (obtained from edge e) is defined as follows. Suppose there is blue flow on arc a^- of value f. Then, red flow on a up to value of f is *free*. Beyond f, the cost of the red flow is $c_2(e)$.

$$\begin{array}{rll}
\min & \sum_{e\in E} c_1(e)\cdot f_1(e) + c_2(e)\cdot f_2(e) & \text{(Flow-LP)}\\
\text{s.t.}\ \ \bar{x} & \text{supports a unit flow } (f_1) \text{ between } s \text{ and } t & \\
\bar{y} & \text{supports a unit flow } (f_2) \text{ between } s \text{ and } t & \\
 & f_1(e) \geq \max(f_1(a), f_1(a^-)) \qquad \text{for all } e = (a, a^-) & \\
 & f_2(e) \geq \max((f_2(a) - f_1(a^-)), 0) + \max((f_2(a^-) - f_1(a)), 0) & \\
\end{array}$$

for all $e = (a, a^-)$

$$\begin{array}{rl}
x(a) + y(a) \leq 1 & \text{for all } a \in A\\
x(a), y(a) \geq 0 & \text{for all } a \in A
\end{array}$$

Given a solution to the SSSPR problem, Lemma 5 tells us how to obtain a two-commodity flow solution from it. We claim that the cost of the two-commodity flow solution is equal to the cost of the solution to the SSSPR problem. Notice that the blue flow costs the same as the blue edges in the SSSPR solution. The cost of the red flow is zero on arcs which are obtained from blue edges. On other edges, the cost of the red flow and the cost of the SSSPR solution are the same. Therefore, the value of an optimal (fractional) solution to the Flow-LP is a lower bound on the value of an optimal integral solution. We now prove that Flow-LP dominates Cut-LP.

Theorem 7. *For any instance of the SSSPR problem, the cost of the optimal solution produced by Flow-LP is at least as high as the cost of the optimal solution produced by Cut-LP.*

Proof. We show that given a feasible solution to Flow-LP, we can generate a feasible solution to Cut-LP without increasing the cost. Consider edge $e \in E$ which is replaced by two anti parallel arcs a and a^- in Flow-LP. Without loss of generality, we can assume that at most one of $\{f_1(a), f_1(a^-)\}$ is non-zero and at most one of $\{f_2(a), f_2(a^-)\}$ is non-zero. Define $x(e) = f_1(e)$ (or $x(e) = f_1(a) + f_1(a^-)$) and $y(e) = \min(f_2(e), 1 - f_1(e))$. We show that $\{x(e), y(e)\}_{e\in E}$ defines a feasible solution for Cut-LP. Let the x-capacity (y-capacity) of a cut be the sum of the variables $x(e)$ ($y(e)$) taken over the edges e belonging to the cut. Clearly, the x-capacity of all $\{s, t\}$-cuts is at least one, since flow f_1 in D sends one unit of flow from s to t. It remains to show that the x-capacity together with the y-capacity of all $\{s, t\}$-cuts is at least two.

Consider a particular $\{s, t\}$-cut C. Decompose flow function f_1 in D into flow paths, each of flow value ε. Let $n(k)$ denote the number of flow paths in the decomposition that use precisely $2k+1$ edges from C. Clearly, $\sum_{k=0}^{\infty} n(k)\cdot\varepsilon = 1$, and so the contribution of flow f_1 in D to the x-capacity of C is

$$\begin{aligned}\textstyle\sum_{k=0}^{\infty}(2k+1)\cdot n(k)\cdot\varepsilon &= 2\sum_{k=0}^{\infty} k\cdot n(k)\cdot\varepsilon + \sum_{k=0}^{\infty} n(k)\cdot\varepsilon \\ &= 2\textstyle\sum_{k=0}^{\infty} k\cdot n(k)\cdot\varepsilon + 1.\end{aligned}$$

Suppose $\sum_{k=0}^{\infty} k\cdot n(k)\cdot\varepsilon < 1/2$, otherwise we are done. The red flow in D, f_2, can send for "free" flow of value at most $\sum_{k=0}^{\infty} k\cdot n(k)\cdot\varepsilon$ using arcs belonging to cut C. (For each arc a carrying blue flow of value ε, red flow of value ε can be sent on a^- for free.) Therefore, the red flow must send flow of value at least $1-\sum_{k=0}^{\infty} k\cdot n(k)\cdot\varepsilon$ using capacity "paid" for by f_2. (Note that for this flow we have $y(e) = f_2(e)$ for all edges e.) Hence, the y-capacity of C is at least $1-\sum_{k=0}^{\infty} k\cdot n(k)\cdot\varepsilon$, yielding that the capacity of cut C (x-capacity and y-capacity) is $1+2\sum_{k=0}^{\infty} k\cdot n(k)\cdot\varepsilon + 1 - \sum_{k=0}^{\infty} k\cdot n(k)\cdot\varepsilon \geq 2$, thus completing the proof.

Integrality Gap: It is not hard to show that a fractional solution to both formulations can be rounded to an integral solution while increasing the cost by at most a factor of 2. The proof is along the lines of the proof for the combinatorial 2-approximation algorithm presented earlier.

We also have an example of an instance where there is a (multiplicative) gap of at least 5/4 between the optimal solution to Flow-LP and any integral solution.

Acknowledgments

We would like to thank Rajeev Rastogi for providing us a copy of [17], and for useful discussions.

References

1. A. Bremler-Barr, Y. Afek, E. Cohen, H. Kaplan and M. Merritt. Restoration by Path Concatenation: Fast Recovery of MPLS Paths. In *Proceedings of the ACM PODC '01*; also in *Proceedings of the ACM SIGMETRICS '01 conference (2-page Poster)*. 2001.
2. A. Balakrishnan, T. Magnanti, and P. Mirchandani. Network Design. *Annotated Bibliographies in Combinatorial Optimization*, M. Dell'Amico, F. Maffioli, and S. Martello (eds.), John Wiley and Sons, New York, 311-334, 1997.
3. A. Balakrishnan, T. Magnanti, J. Sokol, and Y. Wang. Modeling and Solving the Single Facility Line Restoration Problem. Working Paper OR 327-98, Operations Research Center, MIT, 1998. To appear in *Operations Research*.
4. A. Balakrishnan, T. Magnanti, J. Sokol, and Y. Wang. Telecommunication Link Restoration Planning with Multiple Facility Types. To appear in *Annals of Operations Research*, volume "Topological Network Design in Telecommunications" edited by P. Kubat and J. M. Smith.

5. D. Bienstock and G. Muratore. Strong Inequalities for Capacitated Survivable Network Design Problems. *Math. Programming*, 89:127–147, 2001.
6. G. Brightwell, G. Oriolo and F. B. Shepherd. Reserving resilient capacity in a network. In SIAM J. Disc. Math., **14**(4), 524–539, 2001.
7. G. Brightwell, G. Oriolo and F. B. Shepherd. Reserving Resilient Capacity with Upper Bound Constraints. Manuscript.
8. G. Dahl and M. Stoer. A Cutting Plane Algorithm for Multicommodity Survivable Network Design Problems. *INFORMS Journal on Computing*, 10, 1-11, 1998.
9. Bruce Davie and Yakov Rekhter. *MPLS: Technology and Applications.* Morgan Kaufmann Publishers, 2000.
10. Nicholas G. Duffield, Pawan Goyal, Albert G. Greenberg, Partho P. Mishra, K.K. Ramakrishnan, and Jacobus E. van der Merwe. A flexible model for resource management in virtual private networks. In *Proceedings of the ACM SIGCOMM, Computer Communication Review*, volume 29, pages 95–108, 1999.
11. Lisa Fleischer, Adam Meyerson, Iraj Saniee, F. B. Shepherd and Aravind Srinivasan. Near-optimal design of MPλS tunnels with shared recovery. DIMACS Mini-Workshop on Quality of Service Issues in the Internet, 2001.
12. A. Fumagalli and L. Valcarenghi. IP restoration vs. WDM Protection: Is there an Optimal Choice? *IEEE Network*, 14(6):34-41, November/December 2000.
13. M. Goemans and J. Kleinberg. *An improved approximation ratio for the minimum latency problem.* In *Proceedings of 7th ACM-SIAM SODA*, pages 152–157, 1996.
14. Naveen Garg, Goran Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group Steiner tree problem. *Journal of Algorithms*, 37(1):66–84, 2000. (Preliminary version in: *9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 253–259, 1998).
15. Michel X. Goemans, Andrew V. Goldberg, Serge Plotkin, David B. Shmoys, Éva Tardos, and David P. Williamson. Improved approximation algorithms for network design problems. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 223–232, 1994.
16. Anupam Gupta, Amit Kumar, Jon Kleinberg, Rajeev Rastogi, and Bülent Yener. Provisioning a Virtual Private Network: A network design problem for multicommodity flow. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 389–398, 2001.
17. Giuseppe F. Italiano, Rajeev Rastogi and Bülent Yener. Restoration Algorithms for Virutal Private Networks in the Hose Model. *Infocom 2002*.
18. Kamal Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001. (Preliminary version in: *39th Annual Symposium on Foundations of Computer Science*, pages 448–457, 1998).
19. M. Kodialam and T. V. Lakshman. Minimum Interference Routing with Applications to MPLS Traffic Engineering. *Infocom 2000*, pages 884-893, 2000.
20. M. Kodialam and T. V. Lakshman. Dynamic Routing of Bandwidth Guaranteed Tunnels with Restoration. *Infocom 2000*, pages 902-911, 2000.
21. R. Motwani, S. Phillips and E. Torng. *Non-clairvoyant scheduling. Theoretical Computer Science*, 130:17–47, 1994.
22. J. W. Suurballe. Disjoint paths in a network. *Networks*, Vol. 4: 125-145, 1974.
23. David P. Williamson, Michel X. Goemans, Milena Mihail, and Vijay V. Vazirani. A primal-dual approximation algorithm for generalized Steiner network problems. *Combinatorica*, 15(3):435–454, 1995. (Preliminary version in: *25th Annual ACM Symposium on Theory of Computing*, pages 708–717, 1993).
24. D. Zhou and S. Subramaniam. Survivability in Optical Network. *IEEE Network*, 14(6):16-23, November/December 2000.