Flow Splitting Approach for Path Provisioning and Path Protection Problems

Rauf Izmailov
NEC USA, Inc.,
C&C Research Laboratories
rauf@nec-lab.com

Dragos Niculescu
Rutgers University,
Computer Science Department
110 Frelinghuysen Road
Piscataway NJ 08854, USA
dnicules@cs.rutgers.edu

Abstract

We consider off-line versions of path provisioning and path protection problems for general circuit switched networks. Both problems deal with a given network topology and a list of integral demand flows. The objective is to route the flows and to allocate the bandwidth in a way that minimizes the total amount of bandwidth used for working and protection paths. We consider path-based protection where, in case of a single link failure, all the flows utilizing the failed link can be rerouted to a precomputed set of paths. We demonstrate that flow splitting can bring significant advantages for both provisioning and protection problems. Since the problem is NP-complete, we propose and analyze two simple heuristics. We show that one of these heuristics performs almost as well as the optimal solution.

Keywords: path protection, linear programming, flow splitting, routing, recovery, provisioning, optical network.

1 Introduction

Routing, protection and restoration of various types of data flows have been studied extensively in the last two decades (see [1] and its references). These problems have been addressed in the general framework of circuit switching, and for specific technologies, such as ATM, MPLS, and WDM switching (see [2], [3], [4] and their references).

There have been two basic approaches in providing protection: link-oriented, and path-oriented. In the link-oriented approach, the upstream node of the failed link is responsible for failure detection and finding alternate routes [5]. In the path-oriented approach, sets of disjoint paths are precomputed for all source-destination pairs, and each working path is protected by an alternative disjoint path [5].

In this paper, we consider the path-oriented approach, as it yields lower overhead for the protection paths, and is also suitable for global optimization. Once a failure is detected on a link, all the flows using the link for their working paths have to switch to preallocated corresponding protection paths. This protection method has both centralized and distributed phases.

Centralized planning takes into consideration all traffic demands and the network topology in order to establish where flows are routed under normal conditions (provisioning problem) and what alternative paths they are to use under each failure scenario (protection problem). To set up restorable bandwidth, it is sufficient to make sure the protection path is disjoint from all working paths pertaining to a flow. This introduces protection overhead, which can be alleviated by the bandwidth sharing among some of the protection paths.

The distributed phase takes place in the event of a failure, when intelligent switching elements choose the spare paths to restore the affected flows (the flows utilizing the failed link). The switches base their decisions on precomputed (during the first phase) tables, specific to the failure (a link or a group of links).

We concentrate on the centralized phase and address the problem of finding working and protection paths for the general circuit switching network for which all source-destination flows and their integer demands are assumed to be known. The approach presented here is especially appropriate for DWDM networks, where the discrete nature and large bandwidth of individual wave-bands naturally translate into integer demands.

We explore the effects of flow splitting on provisioning and protection problems. Flow splitting can improve the efficiency of provisioning: a path for the flow with bandwidth requirement X may not be as efficient (it may not even exist) as two paths for two sub-flows (with the same source and destination), each with bandwidth requirement $X/2$. Flow splitting can also improve the efficiency of protection: the protection path, disjoint from all working paths, has to be only as large as the largest working path, thus reducing the protection overhead.

The paper is organized in the following way. Section 2 introduces definitions and formulates different versions of provisioning and protection problems. Section 3 outlines two heuristic algorithms. Section 4 examines the results obtained from simulation. The paper ends with a few concluding remarks in Section 5.

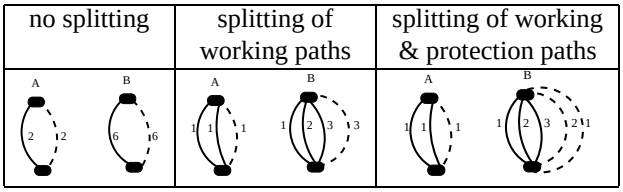

Figure 1: Flow split example.

2 Problem formulation

The provisioning problem of routing integral flows through a given network (also known as the integer multi commodity flow problem) is NP-complete [6]. Several heuristics [7], [8] and approximations were proposed for it. If arbitrary (fractional) splitting were practical, linear programming could provide an optimal solution in polynomial time. Solutions to the provisioning problem depend on whether flows can be split or not, or whether the sub-flows may be fractional or integer. While flows may be split in some scenarios [9], fractional solutions are not practical in most network problems, so linear programming is usually not an option.

If splitting is formulated as a fractional linear program solvable in polynomial time, a non-splitting approach is formulated as a integer linear problem solvable in exponential time. Although the impractical fractional splitting approach gives a better solution than a non-splitting approach, an integer splitting heuristic may provide a compromise feasible solution with reasonably good performance. This approach is further analyzed in the paper.

The protection problem and its requirement for spare capacity allocation adds complexity to the provisioning problem. A scheme in [10] relies on an optimal provisioning precomputed for all configurations with the failed components eliminated. In case of a failure, the corresponding precomputed configuration is used to route the affected flows. Integer programming formulations and relaxations are used in [11], [4] to select appropriate predetermined protection paths for all source-destination pairs. Being NP-hard, integer programming is only usable with small scale problems. The heuristics proposed in [12] can be used for selection from the available paths. In order to reduce the centralization and the amount of information needed for the online flow admission, successive survivable routing is proposed and analyzed in [13].

For the protection problem, the flow splitting approach can be used not only for working paths, but for the protection path as well. Splitting the working path has the advantage of reducing the amount to be protected. As illustrated in Figure 1, while 6 units of bandwidth have to be allocated on the protection path in the first case, only 3 units would be sufficient in the second case. This is referred in the literature as $1:N$ protection (one protection path may protect N working paths). Splitting the protection paths does not reduce the amount of bandwidth to be protected, but it reduces the bandwidth requirement of individual protection paths, which can lead to better bandwidth sharing. Note that while splitting working flows would reduce the amount to be allocated for protection, a larger number of working paths increases the probability of working path intersection, thus reducing the possibilities of sharing for the protection paths. Splitting of both types of paths (working and protection) could also be closer to the optimal fractional solution. The splitting of protection paths is similar to $M:N$ protection [3]. However, the advantage of having more resilience and less protection overhead does hot come for free: network nodes have to handle more sub-flows.

In this section, we describe four problems covering different versions of provisioning and protection problems, with and without path splitting, Throughout the rest of the paper, we use the following notations:

V	set of nodes in the network
E	set of links in the network
K	set of source-destination flows
P	set of paths for the flows
D	number of disjoint paths for a flow
d_{ij}	capacity of link (ij)
δ_{ij}^p	$\begin{cases} 1 & \text{if link } (ij) \text{ is used by path } p \\ 0 & \text{otherwise} \end{cases}$
l_p	length (number of hops) of path p
q^k	demand amount for flow k
$P(k)$	set of paths available for flow k

The allocated bandwidth for working and protection paths is denoted by x and by z, respectively (details are specified for each problem separately); x and z are the only variables used in all formulations of different provisioning and protection problems. δ_{ij}^p describe the collection of disjoint paths available for each flow to use for provision and protection. This set is precomputed offline, therefore δ_{ij}^p are constants in all formulations below.

2.1 Provisioning Problem

The provisioning problem (PP) is a well-known integer multi commodity flow problem: given a set of disjoint source-destination paths for each commodity (flow), find an allocation without splitting the flows that would minimize the total amount of bandwidth used in the network (bandwidth-hop product), while satisfying all flow requirements and the capacity constrains in the links. The variables of PP are denoted by x_p^k where

$$x_p^k = \begin{cases} 1 & \text{if flow } k \text{ is allocated to path } p \\ 0 & \text{otherwise} \end{cases}$$

The formal description of PP is as follows.

$$\text{Minimize} \quad \sum_{k \in K} \left(\sum_{p \in P(k)} x_p^k q^k l_p \right) \qquad (1)$$

subject to

$$\sum_{p \in P(k)} x_p^k = 1, \forall k \in K, \quad (2)$$

$$\sum_{k \in K} \sum_{p \in P(k)} x_p^k \delta_{ij}^p q^k \leq d_{ij}, \forall (ij) \in E. \quad (3)$$

The objective (1) of PP is to minimize the total bandwidth-hop product. Equations (2) are provisioning conditions, while inequalities (3) enforce the capacity constraints for each network link. Since PP is a 0-1 integer program with P variables and $K+E$ constraints, is solvable for problems of small size.

2.2 Provisioning with Splitting Problem

The provisioning with splitting problem (PSP) is an extension of PP when flow splitting is allowed. The variables of PSP are denoted by x_p^k where

$x_p^k =$ bandwidth of working flow k allocated to path p.

The formal description of PSP is as follows.

$$\text{Minimize} \quad \sum_{k \in K} \left(\sum_{p \in P(k)} x_p^k l_p \right) \quad (4)$$

subject to

$$\sum_{p \in P(k)} x_p^k = q^k, \forall k \in K, \quad (5)$$

$$\sum_{k \in K} \sum_{p \in P(k)} x_p^k \delta_{ij}^p \leq d_{ij}, \forall (ij) \in E. \quad (6)$$

The objective (4) of PSP is to minimize the total bandwidth-hop product. Equations (5) are provisioning conditions, while inequalities (6) enforce the capacity constraints for each network link. Since PSP is a linear program with P variables and $K+E$ constraints, it is solvable for most practical scenarios. Performance of PSP provides an upper bound for the performance that can be obtained through flow splitting.

2.3 Provisioning and Protection Problem

The provisioning and protection problem (PPP) requires selection of protection paths, disjoint from the corresponding working paths. Working and protection paths are selected from the set $P(k)$ of disjoint paths precomputed for each source-destination pair k. Protection bandwidth may be shared between paths that protect non-intersecting working paths. For any given link, the amount to be reserved for protection paths depends not only on their bandwidth reservations, but also on the relationship between their corresponding working paths. We associate a variable z_{ij} with this amount in each link (ij). This variable has to satisfy two conditions: (1) it has to fit with the working flows in the link (ij); (2) it has to be able to serve any possible combination of protection flows allocated to link (ij). The variables of PPP are x_{pr}^k and z_{ij} where

$$x_{pr}^k = \begin{cases} 1 & \text{if flow } k \text{ is allocated to working path } p \text{ and protection path } r \\ 0 & \text{otherwise} \end{cases}$$

$z_{ij} =$ bandwidth of link (ij) used for protection

The formal description of the problem is as follows.

$$\text{Minimize} \quad \sum_{ij \in E} z_{ij} + \left(\sum_{k \in K} \sum_{p \in P(k)} \sum_{r \in P(k), p \neq r} x_{pr}^k l_p q^k \right) \quad (7)$$

subject to

$$\sum_{p \in P(k)} \sum_{r \in P(k), p \neq r} x_{pr}^k = 1, \forall k \in K, \quad (8)$$

$$\sum_{p \in P(k)} \sum_{r \in P(k), p \neq r} x_{pr}^k \delta_{ij}^p q^k \leq \leq d_{ij} - z_{ij}, \forall (ij) \in E, \quad (9)$$

$$\sum_{k \in K} \sum_{p \in P(k)} \sum_{r \in P(k), p \neq r} \delta_{ij}^r x_{pr}^k \delta_{ef}^p q^k \leq \leq z_{ij}, \ \forall (ij), (ef) \in E. \quad (10)$$

The objective (7) is to minimize the bandwidth-hop product and the total overhead required to protect all working paths against any single link failure. While the provision (8) and capacity (9) constraints remain similar to the previous cases (PP and PSP), a new set of constraints are needed to size z_{ij}. Constraints (10) are needed for each pair of links (ij) and (ef) so that z_{ij} could protect all the working flows using (ef) and (ij) in their working and protection flows, respectively. Since PPP is a 0-1 integer program with $E+PD$ variables and $K+E+E^2$ constraints, these large numbers make PPP prohibitive complex even for small size problems.

2.4 Provisioning and Protection with Splitting

The provisioning and protection with splitting problem (PPSP) is an extension of PPP when flow splitting of working and protection paths is allowed. The variables z_{ij} of

PPSP are the same as those for PPP, and the variables x_{pr}^k are defined as

$$x_k^{pr} = \text{bandwidth of working flow } k \text{ allocated to path } p \text{ and protection path } r$$

The formal description of the problem is as follows.

$$\text{Minimize} \quad \sum_{ij \in E} z_{ij} \; + \; + \left(\sum_{k \in K} \sum_{p \in P(k)} \sum_{r \in P(k), p \neq r} x_{pr}^k l_p \right) \quad (11)$$

subject to

$$\sum_{p,r \in P(k), p \neq r} x_{pr}^k = q^k, \forall k \in K, \quad (12)$$

$$\sum_{k \in K} \sum_{p \in P(k)} \sum_{r \in P(k), p \neq r} x_{pr}^k \delta_{ij}^p \leq \leq d_{ij} - z_{ij}, \forall (ij) \in E, \quad (13)$$

$$\sum_{k \in K} \sum_{p \in P(k)} \sum_{r \in P(k), p \neq r} \delta_{ij}^r x_{pr}^k \delta_{ef}^p \leq \leq z_{ij}, \;\; \forall (ij), (ef) \in E. \quad (14)$$

As in the previous problems, (12) makes sure each demand is satisfied by its fractions. Equations (13) are capacity constraints for each link: all working paths and the protection capacity z_{ij} for link (ij) cannot exceed the link capacity. The way in which the protection bandwidth for different flows is overlapping is expressed by (14): for any two links (ij) and (ef), all flows using (ij) and (ef) for their protection and working paths, respectively, have to be protected by z_{ij} if the link (ef) fails. PPSP has $2P + E$ variables and $2K + E + E^2 + P$ constraints. It is solvable for fractional solutions in a time ranging from minutes to hours, depending on $\sum q^k$, on a high end workstation (Sun Ultra 10, 440Mhz) for networks with about 1200 links. We use PPSP both as a reference point and a base for our heuristics.

3 Heuristics

The provisioning and protection problems depend on one another. The selection of provisioning paths with splitting can limit the routing choices for the protection paths; the selection of a protection path affects the bandwidth sharing with the subsequently allocated working paths are established. Although it is plausible to assume that the initial choice of disjoint paths (i.e., the sets $P(k)$) has a more significant impact on performance than the actual working paths allocation, we do not analyze this issue in the paper, concentrating on the optimization of protection paths instead.

Since our objective is to assess the general benefits of flow splitting for provisioning and protection problems, we selected simple heuristics and compare their performance against the corresponding optimal solutions (with and without flow splitting). The heuristics are based on the objectives of the analyzed problems. For the provisioning problem, the objective is to balance the load on the existing paths, maximize residual capacity, and minimize interference between working paths as described in [8]. For the protection problem, the objective is to maximize interference, leading to bandwidth sharing for protection paths.

We use the following definitions to measure the interference for working paths and protection paths: a bottleneck link is the one that has the minimum residual capacity along a disjoint path; a link shared between paths X and Y is *critical* in X if it is a bottleneck in Y. A critical link has the property that any flow routed through it will decrease the residual capacity in some other path. The heuristics used in this paper call for utilizing routes with few critical links for working paths and routes intersecting many other protection paths for protection paths.

One of the approaches, "`Sorting heuristic`", first sorts the available paths in a way that minimizes interference (number of critical links or number of working flows intersected). During the allocation process each path has the associated figure for the number of critical links and the number of flows whose protection paths are being intersected. Working flow allocation splits evenly its working component into a predefined number $S < D$ of sub-flows distributed across D available disjoint paths. A protection path is then allocated on one of the remaining $D - S$ paths disjoint from S paths. The protection path is selected in a way that maximizes its intersection with other protection paths. If splitting is also employed for the protection path, the available paths may be sorted based on their number of intersections with other protection paths. This would allow choosing paths with high potential for sharing. This heuristic can also be applied online, since we evaluate one flow at a time, based on the above mentioned criteria.

Another approach is to use the solution of a fractional linear program and subsequent rounding of the solutions. This approach may lead overload the links that have many subunit allocations. A more practical solution is to truncate the solutions, and then allocate the cumulative left bandwidth using the mentioned sorting methods. We use this approach as a second heuristic, called "`truncate heuristic`".

4 Simulations

The networks we used for simulation in our validation tests are based on topologies of actual networks: the IP backbone network shown in Figure 2, and the Kyoto University network shown in Figure 3. The IP backbone network consists of 12 nodes and 42 links, all of the same capacity. The Ky-

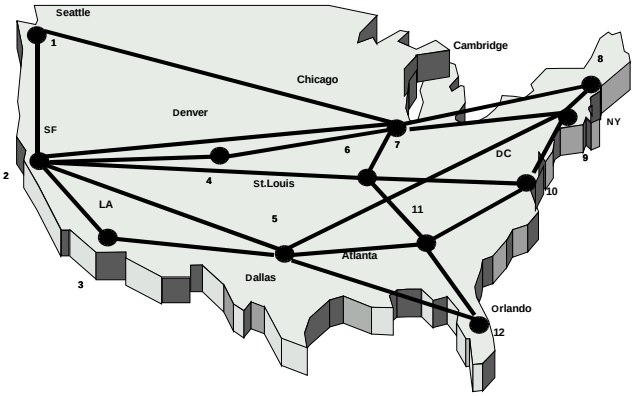

Figure 2: IP Backbone.

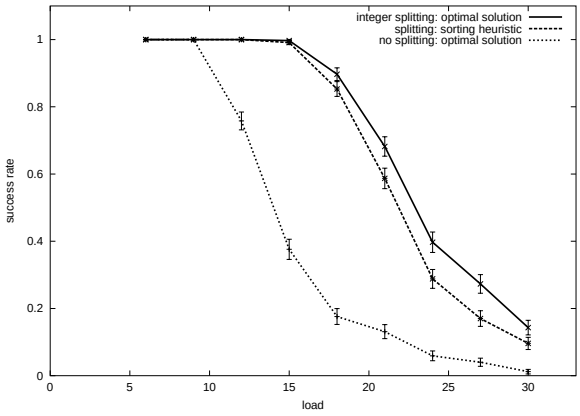

Figure 4: Provisioning (IP Backbone).

.

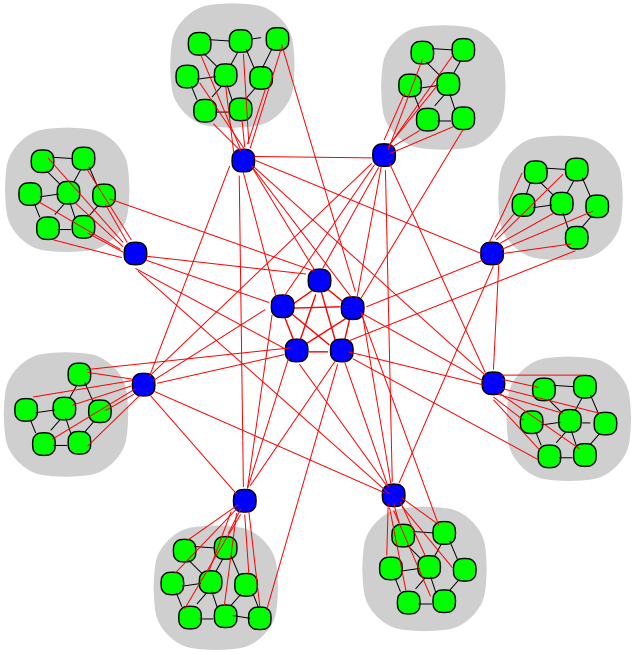

Figure 3: Kyoto University network

oto network has 78 nodes and 1122 links, also of identical capacity.

As shown in [14], the "k-shortest paths" method for generating disjoint paths is fast and reasonably close to the optimum solution. For each source-destination pair k, we ran Dijkstra's shortest path algorithm, and then iterated the procedure on the network with the newly discovered path pruned. The set of paths obtained in this manner forms the set $P(k)$ of paths available to each flow k.

For the sorting heuristic in the IP backbone network, we split the flows evenly into $S = 3$ working sub-flows, since at least $D = 4$ disjoint paths are available for each source-destination pair. Although this arrangement uniquely determined the protection path, the performance was better than the one observed for the case where flows were split into $S = 2$ working sub-flows (this providing $D - S = 2$ potential protection paths for each flow). This effect is due to the fact that the simulated network topology did not allow much bandwidth sharing, and savings from reduced protection path choices were more important than those from shared protection paths. In case of Kyoto University network, each source-destination pair can have at least $D = 8$ disjoint paths. For the sorting heuristic, we split the flows evenly into $S = 5$ working paths, so that the protection path was chosen from the remaining $D - S = 3$ paths. We used the freeware program solver `lp_solve`[15] both for the integer and linear programs.

Two series of simulations were conducted to test the effectiveness of the flow splitting. The first series involved only provisioning, and used the sorting heuristic, with respect to the number of critical links for each path. The second series covered both provisioning and protection and used both heuristics described in Section 3. The traffic matrices were generated in a way that created the Zipf [16] distribution of different flows. Performance was measured as the probability that randomly generated traffic matrix of a given load can be provisioned and protected in the network by the proposed algorithms. The amount of protection bandwidth overhead can also be used as a performance metric, but it is actually reflected in the success rate. When the amount of working bandwidth is subtracted from the network, the amount and distribution of the protection bandwidth determines the success of the allocation.

For the provisioning problem, we compared the optimal non-splitting case (PP) and the optimal integer splitting (PSP). The results are shown in Figure 4 are shown with their 95% confidence intervals. The upper curve corresponds to PSP, and the lower curve to PP. By sorting the flows based on their number of intersections with working flows, we can achieve performance close to the optimal for an average load.

For the protection problem, we compared the sorting and

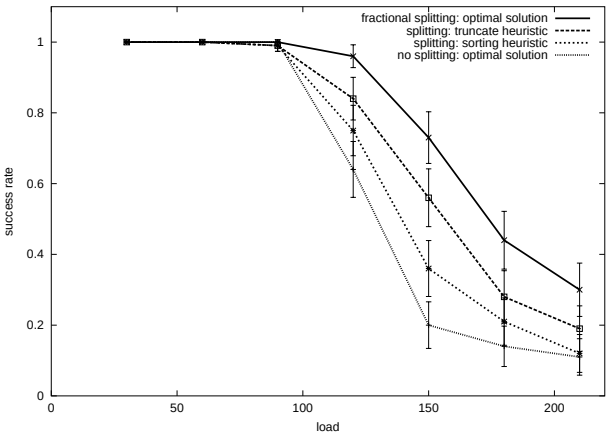

Figure 5: Provision and protection (IP Backbone).

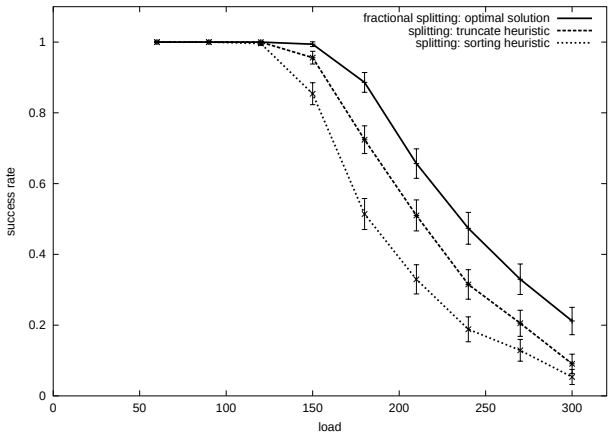

Figure 6: Provision and protection (Kyoto University network).

truncating heuristics against the optimal solutions obtained from PPP and PPSP. Due to the prohibitive running time (one instance of the integer program PPSP, solved with a brute force method, takes about 8 hours on a Pentium III at 500Mhz), we only ran 100 trial problems for each point and the calculated confidence intervals are 90%. As shown in Figure 5, flow splitting can increase the success rate by 30% in the average load case (compared with no flow splitting). The fractional splitting performance is not a tight upper bound since fractional flows are not usually an option in most network problems.

We ran similar tests for the Kyoto University model; results are shown in Figure 6. For this large network (Figure 3), solving PPP is not feasible due to the large number of variables. This problem can only be solved in the fractional domain. Since we did not have the no splitting case for this network, we could not carry out the lower bound comparison for this case. However, the sorting heuristic can serve as a comparison reference here. PPSP is solvable for fractional solutions for the Kyoto network in reasonable time (a couple of hours), depending on the load.

5 Conclusions

We demonstrated the advantages of flow splitting for both provisioning and protection problems. We proposed two heuristics and compared their performance with that of optimal solutions. We showed that simple flow splitting heuristics can improve the success rate by 30-50% over optimal approaches that do not use flow splitting. We also expanded the flow splitting approach to protection paths, which further improved performance by better bandwidth sharing among different protection paths.

In our future work, we plan to further improve the performance of our algorithms by exploring additional flow splitting and routing options.

References

[1] T.E.Stern, “Multiwavelength Optical Networks: A Layered Approach”, Prentice Hall, 1999.

[2] A. Iwata, R. Izmailov and B. Sengupta, “Alternative Routing Methods for PNNI Networks With Partially Disjoint Paths,” *GLOBECOM 1998*.

[3] V. Sharma, B. Crane, S. Makam, K. Owens, C. Huang, F. Hellstrand, J. Weil, L. Andersson, B. Jamoussi, B. Cain, S. Civanlar, A. Chiu, “Framework for MPLS-based Recovery”, Internet draft (work in progress), draft-ietf-mpls-recovery-frmwrk-03.txt July 2001.

[4] Y.Miyao and H.Saito, “Optimal design and evaluation of survivable WDM transport networks”, IEEE Journal on Selected Areas in Communications, vol. 16, no. 7, September 1998.

[5] R. Doverspike and B. Wilson, "Comparison of capacity efficiency of DCS network restoration routing techniques", Journal of Network and System Management, vol. 2, no. 2, pp 95-123, 1994.

[6] R.M. Karp, “On the computational complexity of combinatorial problems”, Networks, 5, pp 45-68, 1975.

[7] C. Frei and B. Faltings, “Abstraction and Constraint Satisfaction Techniques for Planning Bandwidth Allocation” , *INFOCOM 2000.*

[8] M. Kodialam and T.V. Lakshman, “Minimum interference routing with applications to MPLS traffic engineering”, *INFOCOM 2000.*

[9] S. Biswas, R. Izmailov and B. Sengupta, “Connection Splitting: An Efficient Way of Reducing Call Blocking in ATM,” IEEE/ACM Transactions on Networking, Vol. 8, No. 5, pp 655-666, 2000.

[10] B. Coan, W. Leland, M. vecchi, A. Weinrib and L. Wu, “Using distributed topology update and preplanned configurations to achieve trunk network survivability”, in IEEE Transactions on Reliability, vol. 40, no. 4, October 1991.

[11] R.R.Irascho, M.H. MacGregor and W.D. Grover, “Optimal capacity placement for path restoration in mesh survivable networks”, IEEE/ACM Transactions on Networking, pp. 325-226, June 1998.

[12] M. Kodialam and T.V. Lakshman, “Dynamic routing of bandwidth guaranteed tunnels with restoration”, *INFOCOM 2000.*

[13] Y. Liu, D. Tipper and P. Siripongwutikorn, “Approximating optimal spare capacity allocation by successive survivable routing”, I*NFOCOM 2001.*

[14] D.A. Dunn, W.D. Grover and M.H. MacGregor, “Comparison of k-shortest paths and maximum flow routing for network facility restoration”, in IEEE Journal on Selected Areas in Communications, vol. 12, no. 1, January 1994.

[15] `ftp://ftp.es.ele.tue.nl/pub/lp_solve`

[16] B. Hill, “Zipf’s Law and Prior Distribution for the Composition of a Population”, Journal of the Americal Statistical Association, September 1970, Vol. 65, No. 331, pp. 1220-1232.