

Enhancing Robustness Under Dual-Link Failures

Sandeep Kour Ahuja and Srinivasan Ramasubramanian
Department of Electrical and Computer Engineering
University of Arizona, Tucson, AZ 85721
{sandeepa, srini}@ece.arizona.edu

Abstract—In this paper, we show that minimizing the product of path lengths results in minimizing the probability of connection failure between a source and destination given two links have failed in the network. We, therefore, analyze the problem of finding disjoint paths between two nodes with the minimum path length product. We formulate the problem as a Mixed Integer Quadratic Programming (MIQP) problem. We develop an iterative heuristic based on k -shortest path approach. We demonstrate through extensive simulations that only a few iterations are sufficient to obtain paths whose path length product is comparable to that of optimal obtained by solving the MIQP.

I. INTRODUCTION

Optical networking employing dense wavelength division multiplexing (DWDM) has emerged as a dominant technology in communication networks due to the massive bandwidth it offers. Using the DWDM technology, a single fiber can carry multiple wavelengths and each wavelength can carry up to 40 Gbps (OC-768). Hence, a fiber can support a total capacity of more than 1 Tbps. Providing resilience against failures to these high-speed networks becomes indispensable as a failure of a link or node results in the loss of huge amount of data revenue.

In any network, there are two kind of failures: link failures and node failures. Link failures can occur due to a fiber cuts, component failures (e.g., inline amplifier) or human errors. Node failures result in failures of multiple links passing through the failed node. Node failures are rare, caused by power failures and catastrophic events, such as flood, fire, earthquake etc.

In order to protect connections from single link failures in a network, several techniques have been developed ([1], [2], [3], [4], [5], [6] etc.). These protection approaches can be classified into *path protection* and *link protection*. In path protection schemes each connection has an active (primary) path on which a connection is established, and a *disjoint* backup (secondary) path on which the connection will be re-established if the active path fails. Alternatively, the traffic between the given pair of nodes in the network maybe divided equally over the two disjoint paths so that if a node or link on one of the paths fails, not all of the traffic is lost. Backup paths may be provided in a *failure dependent* or *failure independent* fashion. Failure dependent path protection assigns multiple backup paths to a connection and selects one depending on the failed link in the working path. Failure independent path protection assigns a single end-to-end disjoint backup path for a source-destination pair and does not need to know exactly

which link has failed in the active path. In link protection schemes, backup paths are found around a failed link. Link protection provides faster recovery than path protection because failure recovery is performed around the failed link only. The resource requirement for link protection is more compared to path protection strategies. On the other hand, path protection typically lower end-to-end propagation delay for the re-established connection [1], [7], [8]. In this paper, we focus on failure independent path protection with end-to-end link-disjoint backup paths.

Link-disjoint paths may be found in several ways [9], [10], [11], [12], [13], [14], [15]. In [9], a polynomial time algorithm for computing K node disjoint paths such that the sum of the path lengths is minimum is developed. In [10], an improved algorithm (s-t algorithm) is developed to select link-disjoint paths from one source node to n destination nodes could be obtained in a single Dijkstra-like computation. Bhandari [14] developed an algorithm for the shortest pair between a given pair of nodes in the network which is node-disjoint. In all these proposed schemes the objective was to find a pair of link- or node-disjoint paths with minimum sum of path lengths. The intuition behind optimizing the sum of the length of the disjoint paths is that it reduces the resources utilized. In addition, shorter paths increase network reliability under single link/node failures.

A. Motivation

Two link-disjoint (node-disjoint) paths with minimum sum of the path lengths minimizes the probability the path is affected given that a link (node) has failed. However, minimizing the sum of the disjoint path lengths does not necessarily improve the robustness of the path under dual-link failures. Consider two link-disjoint paths \mathcal{P}_1 and \mathcal{P}_2 from a source to destination with lengths h_1 and h_2 , respectively. Let L denote the number of links in the network. We compute the robustness of the paths to dual-link failures by evaluating the probability that both the primary and backup paths fail given that two links have failed in the network, denoted by P_f . Given two link failures, both the paths (and hence the connection) fail when exactly one link fails in each path and no other link in the network fails. Assume, for simplicity, that all the links have same failure probability, p . Assuming that link failures are independent, we have:

$$P_f = \frac{\binom{h_1}{1}p(1-p)^{h_1-1}\binom{h_2}{1}p(1-p)^{h_2-1}(1-p)^{L-h_1-h_2}}{\binom{L}{2}p^2(1-p)^{L-2}}$$

$$= \frac{h_1 h_2}{\binom{L}{2}}$$

We observe that the probability that both the paths get disconnected is minimized if the product of the path lengths is minimized rather than the sum of the path lengths.

Now, consider the case when the link failure probabilities are not identical. Assume that the lifetime of link i is exponentially distributed with parameter λ_i . Let $\Lambda = \sum_{i=1}^L \lambda_i$. The probability that the first failure is link i and the second failure is link j is given by:

$$\begin{aligned} P_{ij} &= \int_{t=0}^{\infty} (1 - e^{-\lambda_i t}) e^{-(\Lambda - \lambda_i - \lambda_j)t} \lambda_j e^{-\lambda_j t} dt \\ &= \frac{\lambda_j}{\Lambda - \lambda_i} - \frac{\lambda_i}{\Lambda} \end{aligned}$$

Similarly, probability that the first failure is link j and the second failure is link i is given by:

$$P_{ji} = \frac{\lambda_i}{\Lambda - \lambda_j} - \frac{\lambda_j}{\Lambda}$$

The probability that the first two link failures are i and j (without ordering) is given by:

$$\begin{aligned} Q_{ji} = Q_{ij} &= P_{ij} + P_{ji} \\ &= \frac{\lambda_i}{\Lambda - \lambda_j} + \frac{\lambda_j}{\Lambda - \lambda_i} - \frac{\lambda_i + \lambda_j}{\Lambda} \\ &= \frac{\lambda_i \lambda_j}{\Lambda} \left[\frac{1}{\Lambda - \lambda_i} + \frac{1}{\Lambda - \lambda_j} \right] \\ &\approx \frac{2\lambda_i \lambda_j}{\Lambda^2} \end{aligned}$$

The above approximation is valid when the number of links is large in the network and the variance of failure rates across the links is not significantly high.

Now, given that two links have failed, the probability that the first two link failures are i and j (without ordering) is given by:

$$\begin{aligned} R_{ji} = R_{ij} &= \frac{Q_{ij}}{\sum_{i=1}^L \sum_{j=i+1}^L Q_{ij}} \\ &= \frac{\lambda_i \lambda_j}{\sum_{i=1}^L \sum_{j=i+1}^L \lambda_i \lambda_j} \end{aligned}$$

Given two links have failed, the probability that two disjoint paths \mathcal{P}_1 and \mathcal{P}_2 fail is given by:

$$\begin{aligned} P(f) &= \sum_{i \in \mathcal{P}_1 \text{ and } j \in \mathcal{P}_2} R_{ij} \\ &= \frac{\left(\sum_{i \in \mathcal{P}_1} \lambda_i \right) \left(\sum_{j \in \mathcal{P}_2} \lambda_j \right)}{\sum_{i=1}^L \sum_{j=i+1}^L \lambda_i \lambda_j} \end{aligned}$$

The above shows that the probability both the paths are affected given two links have failed is proportional to the product of the failure rates of the paths.

B. Example

Figure 1 shows an example network with arbitrary link costs in which two link-disjoint paths between nodes A and D need to be obtained. The link weights (costs) indicate the parameter of the exponential lifetime distribution for the links. The link-disjoint paths that have minimum sum of path costs, referred to as the MINSUM paths, are A-B-H-D and A-G-C-D with a path cost sum value as 24 and product value as 144. The link-disjoint paths that have the minimum product of path costs, referred to as the MINPROD paths, are A-B-C-D and A-E-F-D with path cost sum value as 26 and product value as 120. It may be observed that the sum of the path costs of MINPROD paths are greater than that of MINSUM paths while the product of the MINPROD paths are lower than that of MINSUM paths. If the dotted lines in the Figure 1 are viewed as multi-hop path with the link weight as the hop length, then the example illustrates computing paths with minimum path length product.

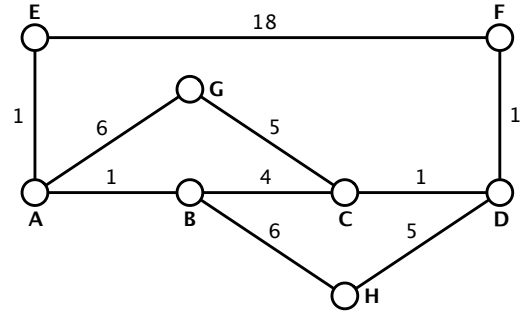


Fig. 1. Example network with arbitrary link costs. The dotted lines may be viewed as a multi-hop path with the link weight as the hop length.

In addition to improving the robustness to dual-link failures, obtaining a path with minimum path cost product helps in minimizing the resource requirement in the networks. It may be observed that the length of the shortest path in the MINPROD paths is shorter than the length of the shortest path in the MINSUM paths. Thus, if the shortest of the two paths is used as the primary path, then the resources committed in the network is minimized with the MINPROD approach. Although the total resources committed to the backup path may be more in the MINPROD approach, these resources may be shared with the backup paths of those connections if the primary paths are link-disjoint.

The goal of this paper is to identify link-disjoint paths with minimum path length (cost) product. To this end, we formulate the problem as a mixed integer quadratic program (MIQP) and develop an iterative heuristic approach based on k -shortest path algorithm. We demonstrate through extensive simulations that we can achieve the minimum path length (cost) product close to that obtained by MIQP with smaller value of k .

The rest of the paper is organized as follows. In Section II, we develop the MIQP formulation and a heuristic approach. In Section III, we present the simulation results. Section IV concludes the paper.

II. COMPUTING DISJOINT PATHS WITH MINIMUM PATH LENGTH (COST) PRODUCT

Consider a network $G(V, E)$, where V is the set of nodes and E is the set of links in the network. Consider a connection between source s and destination d . Let P_1 be the primary path and P_2 be the backup path between s and d , such that P_1 and P_2 are link-disjoint. We assume that the probability of failure is same for all the links and these failures are independent of each other.

A. MIQP Formulation

We formulate the problem of identifying link-disjoint paths with minimum path cost product as a network flow problem as shown in Figure 2. Let X and Y be two flows in the network $G(V, E)$ from source s to destination d . $x_{u,v}$ and $y_{u,v}$ are two decision binary variables which denote if link (u, v) is traversed by flow X and Y , respectively. For each link $(u, v) \in E$, $x_{u,v}$ ($y_{u,v}$) is one if flow X (Y) traverses the link, otherwise it is set to zero. One unit of flow X and one unit of flow Y are routed from source s to the destination d . We assume that each link $(u, v) \in E$ has one unit of capacity. DC is the disjointedness constraint which ensures that no link in the network is traversed by both flows X and Y . $FC1$ and $FC2$ are flow conservation constraints which ensure that one unit of flow X and one unit of flow Y are routed from source s to destination d , respectively. We assume that a link (u, v) has a cost of $C_{u,v}$ ($C_{u,v} = 1$, if the cost is number of hops). The cost of a flow is equal to the sum of the cost of the links traversed by it. Our objective is to minimize the product of the path costs.

The links traversed by flow X and Y provides the two disjoint paths from the given source node s to destination node d such that the product of the path costs is the minimum among all possible disjoint path pairs between the nodes s and d .

The complexity associated with MIQP grows exponentially with the number of links in the network. We, therefore, develop an iterative heuristic for computing the solutions for large networks.

B. MINPROD Heuristic

The basic idea behind the heuristic is to identify the set of k shortest paths between a source-destination pair, obtain the disjoint path corresponding to each of the path from the set (using flow augmentation), and then select the path pair with minimum path cost product.

The pseudo-code for the MINPROD heuristic is presented in Figure 3. The input is an undirected graph $G(V, E)$; the source node s ; the destination node d ; and the number of shortest paths K to be identified. The algorithm returns two disjoint paths P_1 and P_2 with the minimum path cost product between s and d . We initialize the cost of each path, h_1 and h_2 , and their product as ∞ . We associate a cost for all links in the network, $w(u, v) \forall (u, v) \in E$. The heuristic first finds a set of k successive shortest paths, with respect to $w(\cdot)$, between s and d using the well known k -Shortest Path (KSP) algorithm [16], [17] and arranges these paths in the increasing order of their

Objective:

$$\text{Minimize} \left(\sum_{(u,v) \in E} x_{u,v} C_{u,v} \right) \left(\sum_{(u,v) \in E} y_{u,v} C_{u,v} \right)$$

Subject to:

Disjointedness Constraint (DC):

$$x_{u,v} + y_{u,v} + y_{v,u} \leq 1 \quad \forall (u, v) \in E$$

Flow Constraint 1 (FC1):

$$\begin{aligned} \sum_{v:(u,v) \in E} x_{u,v} - \sum_{v:(v,u) \in E} x_{v,u} &= 1; & u = s \\ &= 0; & u \in V - \{s, d\} \\ &= -1; & u = d. \end{aligned}$$

Flow Constraint 2 (FC2):

$$\begin{aligned} \sum_{v:(u,v) \in E} y_{u,v} - \sum_{v:(v,u) \in E} y_{v,u} &= 1; & u = s \\ &= 0; & u \in V - \{s, d\} \\ &= -1; & u = d. \end{aligned}$$

Binary variables:

$$\begin{aligned} x_{u,v} &= \{0, 1\}, \quad \forall (u, v) \in E \\ y_{u,v} &= \{0, 1\}, \quad \forall (u, v) \in E \end{aligned}$$

Fig. 2. MIQP formulation for computing link-disjoint paths from source s to destination d with minimum path cost product.

path costs in \mathcal{K} . The heuristic starts with the shortest (first) path p_1 in \mathcal{K} and remove all the links from $G(V, E)$ which forms the path p_1 . Then it uses Dijkstra's algorithm [18] in the residual graph to find the shortest path q_1 , with respect to $w(\cdot)$, from the source s to the destination d . This is the shortest disjoint path to the path p_1 . Let $temp_h1$ be the cost of path p_i , $temp_h2$ be the cost of path q_i and min_h1h2 be the minimum path cost product. The heuristic saves the product $temp_h1 * temp_h2$ as min_h1h2 , the paths p_1 as P_1 and q_1 as P_2 , and $temp_h1$ as h_1 and $temp_h2$ as h_2 . Now, it iteratively repeats the same procedure of choosing a path p_i sequentially from \mathcal{K} , removing all the links from the network $G(V, E)$ and finding the shortest disjoint path q_i corresponding to p_i and compare the path cost product of p_i and q_i with the saved min_h1h2 . If any path pair in these iterations have $temp_h1 * temp_h2$ lesser than the saved product min_h1h2 , it updates the saved paths and corresponding path costs and path cost product. If the heuristic finds a path pair having $temp_h1 * temp_h2$ same as the saved min_h1h2 , it chooses the path pair with lesser path cost sum. The heuristic terminates under one of the two conditions: (1) it comes across a path p_i in \mathcal{K} such that the cost of p_i is greater than the saved h_2 , or (2) all the paths in the set \mathcal{K} are exhausted.

The complexity of the KSP algorithm is $\mathcal{O}(K|E|\log(K|V|) + K^2|E|)$ [19] and that of the Dijkstra's

algorithm is $\mathcal{O}(|V|\log(|V|) + |E|)$ [18] with Fibonacci heap as priority queue, where $|V|$ is the number of nodes in the network, $|E|$ is the number of links in the network, and K is the number of successive shortest paths to be considered. The procedure Update-Paths has a complexity of $\mathcal{O}(1)$.

```

MinProd( $G(V, E), s, d, k, P_1, P_2$ )
1. Initialization:  $h_1 = \infty, h_2 = \infty, \min h_1 h_2 = \infty,$ 
    $\text{sum} h_1 h_2 = \infty.$ 
2.  $\mathcal{K} = \mathbf{KSP}(G(V, E), s, d, w(\cdot), k)$ 
3. For each path  $p_i \in \mathcal{K}$ , do
4.    $\text{temp}_{h_1} = \text{cost of the path } p_i,$ 
5.   If ( $\text{temp}_{h_1} > h_2$ ),
6.     Return( $P_1, P_2$ ).
7.    $E' = E - \{(u, v) : (u, v) \in p_i\},$ 
8.    $q_i = \mathbf{Dijkstra's Algorithm}(G(V, E'), s, d, w(\cdot)),$ 
9.    $\text{temp}_{h_2} = \text{cost of the path } q_i,$ 
10.   $\text{temp}_{\min h_1 h_2} = \text{temp}_{h_1} * \text{temp}_{h_2},$ 
11.  If ( $\text{temp}_{\min h_1 h_2} < \min h_1 h_2$ ),
12.    Update-Paths( $\text{temp}_{\min h_1 h_2}, p_i, q_i, h_1, h_2,$ 
    $\text{temp}_{h_1}, \text{temp}_{h_2}, \min h_1 h_2, P_1, P_2$ ),
13.  Else if ( $\text{temp}_{\min h_1 h_2} == \min h_1 h_2$ ),
14.    If ( $(\text{temp}_{h_1} + \text{temp}_{h_2}) < \text{sum} h_1 h_2$ ),
15.      Update-Paths( $\text{temp}_{\min h_1 h_2}, p_i, q_i,$ 
    $h_1, h_2, \text{temp}_{h_1}, \text{temp}_{h_2}, \min h_1 h_2,$ 
    $P_1, P_2$ ),
16.  Return( $P_1, P_2$ ).

```

Function Update-Paths updates saved pair of disjoint paths, their corresponding path costs and the product of their path costs.

Fig. 3. Pseudocode for MINPROD algorithm.

III. PERFORMANCE EVALUATION

We compare the performance of the MINPROD heuristic by comparing the results to that obtained from the MIQP. In addition, we compare the results of the MINPROD heuristic with that of MINSUM algorithm that computes disjoint paths to minimize the sum of the path costs based on flow-augmenting technique. We evaluate the performance on four network topologies: (i) ARPANET, (ii) NSFNET, (iii) NJ-LATA, and (iv) 4×4 MESH network as shown in Figure 4. We also compare the obtained path lengths to the path lengths of disjoint paths with the minimum path length sum.

The solution for the MIQP was obtained using ILOG CPLEX 8.1 ([20]). With the MINPROD heuristic, we find a pair of disjoint paths with minimum path length product by varying the value of K , where K is the number of successive shortest paths considered for the primary path. We calculate average path length for the primary paths as well as backup paths for different values of k and compare it to the optimal results obtained by solving the MIQP.

Table I shows the average path lengths of primary and backup paths obtained using the MINPROD heuristic and the optimal values obtained by solving the MIQP for minimizing the path length product. It is observed that the deviation of the heuristic results from the optimal results drops significantly with the number of successive paths considered. It is also observed that there is no improvement in the results when more than three successive shortest paths are considered.

We compare the average of the sum of the path lengths obtained from the MINPROD heuristic to that obtained from the MINSUM algorithm in Table II. We observe that the results of both the approaches are the same, indicating that the MINSUM paths are also the MINPROD paths for these smaller networks.

We evaluate the performance of the MINPROD algorithm with random link weights. In our simulation, we assign link weights according to a uniform distribution in the range [10, 50]. The comparison of the results with random link weights is shown in Table III. The last row in the table provides the average sum of the path costs and average product of the path costs for the MINSUM algorithm. Similar to earlier results, it is observed that the path cost products are minimized with very few iterations. In addition, the difference between the average sum of the cost of MINPROD and MINSUM paths are negligible. The results indicate that robustness to dual-link failures may be achieved using path protection strategy employing link-disjoint paths with negligible increase in resource utilization.

IV. CONCLUSION

This paper illustrates that the robustness of a connection may be improved by selecting disjoint paths with minimum path cost product. We formulate the problem of computing disjoint paths with minimum path length product as a mixed-integer quadratic program and develop a heuristic solution based on K -shortest path. We demonstrate that in most cases, the MINPROD heuristic gives optimal solution in a fewer number of iterations.

ACKNOWLEDGMENT

The research developed in this paper is supported by National Science Foundation under grants 0325979, 0435490, and EEC-0333046.

REFERENCES

- [1] T. Wu, "Emerging technologies for fiber network survivability," *IEEE Communication Magazine*, vol. 33, no. 2, pp. 58–59, 62–74, February 1995.
- [2] M. Medard, S. G. Finn, and R. A. Barry, "WDM loop-back recovery in mesh networks," in *Proceedings of IEEE INFOCOM*, March 1999, pp. 752–759.
- [3] G. Ellinas, G. Halemariam, and T. Stern, "Protection cycles in WDM networks," *IEEE Journal of Selected Areas in Communication*, vol. 8, no. 10, pp. 1924–1937, October 2000.
- [4] W. D. Grover, *Mesh-Based Survivable Networks: Options and Strategies for Optical, MPLS, SONET and ATM networking*. Prentice Hall, 2003.
- [5] S. Ramamurthy, L. Sahasrabudde, and B. Mukherjee, "Survivable WDM mesh networks," *Journal of Lightwave Technology*, vol. 21, no. 4, pp. 870–883, April 2003.
- [6] D. Zhou and S. Subramaniam, "Survivability in optical networks," *IEEE Network*, vol. 14, no. 6, pp. 16–23, November 2000.

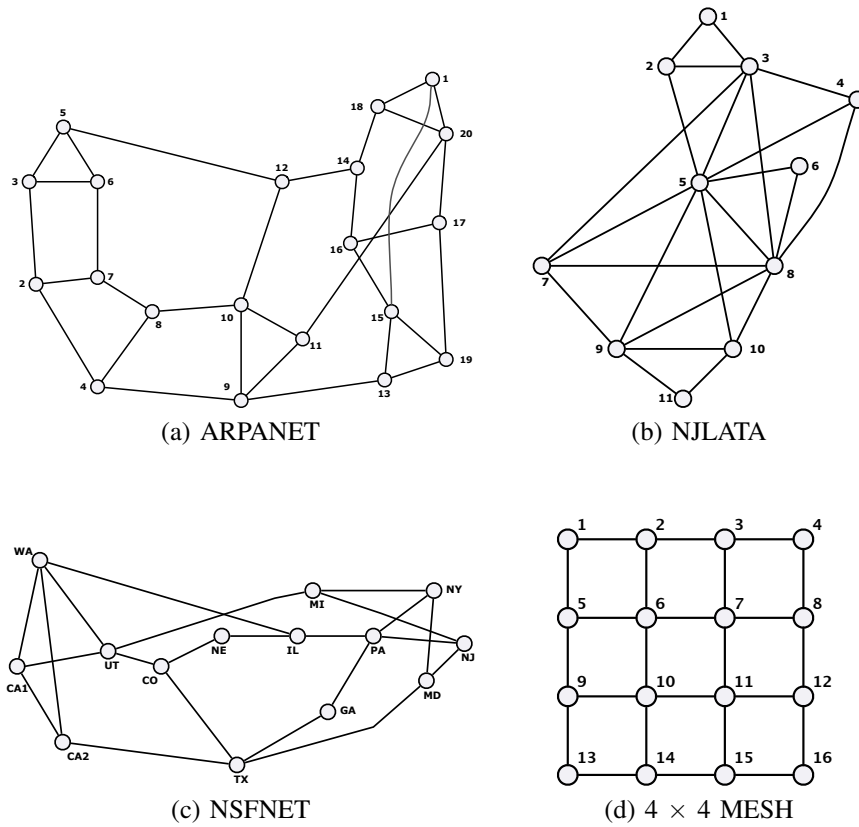


Fig. 4. Topologies considered for performance evaluation.

TABLE I
COMPARISON FOR AVERAGE PRIMARY AND BACKUP PATH LENGTHS.

K	ARPANET				NJ-LATA				NSFNET				4 X 4 mesh network			
	Average Primary Path Length		Average Backup Path Length		Average Primary Path Length		Average Backup Path Length		Average Primary Path Length		Average Backup Path Length		Average Primary Path Length		Average Backup Path Length	
	MIQP	MinProd Heuristic	MIQP	MinProd Heuristic	MIQP	MinProd Heuristic	MIQP	MinProd Heuristic	MIQP	MinProd Heuristic	MIQP	MinProd Heuristic	MIQP	MinProd Heuristic	MIQP	MinProd Heuristic
1	2.8	2.79	4.24	4.32	1.75	1.75	2.31	2.35	2.08	2.08	3.25	3.25	2.67	2.67	3.47	3.77
2	2.8	2.8	4.24	4.24	1.75	1.75	2.31	2.31	2.08	2.08	3.25	3.25	2.67	2.67	3.47	3.52
3	2.8	2.8	4.24	4.24	1.75	1.75	2.31	2.31	2.08	2.08	3.25	3.25	2.67	2.67	3.47	3.47

TABLE II
COMPARISON FOR THE SUM OF AVERAGE PRIMARY AND BACKUP PATH LENGTHS

Topology	Average Sum of Primary and Backup Path Lengths for MIQP	Average of Minimum Sum of Primary and Backup Path Lengths
ARPANET	7.04	7.04
NSFNET	5.33	5.33
NJ-LATA	4.05	4.05
4 X 4 Mesh Network	6.13	6.13

TABLE III
COMPARISON FOR AVERAGE PRIMARY AND BACKUP PATH COST

K	MinProd							
	ARPANET		NJ-LATA		NSFNET		4 X 4 mesh network	
	Average Sum of Path Cost	Average Product of Path Cost	Average Sum of Path Cost	Average Product of Path Cost	Average Sum of Path Cost	Average Product of Path Cost	Average Sum of Path Cost	Average Product of Path Cost
1	202.03	10763.56	98.38	2637.85	155.22	5672.14	185.25	9195.73
2	200.94	10634.02	97.93	2616.36	155.22	5672.14	182.19	8955.58
3	200.78	10628.09	97.93	2616.36	155.22	5672.14	180.91	8811.64
4	200.8	10628.09	97.93	2616.36	155.22	5672.14	180.91	8811.64
5	200.8	10628.09	97.93	2616.36	155.22	5672.14	180.91	8811.64
6	200.8	10628.09	97.93	2616.36	155.22	5672.14	180.91	8811.64
7	200.8	10628.09	97.93	2616.36	155.22	5672.14	180.91	8811.64
8	200.8	10628.09	97.93	2616.36	155.22	5672.14	180.91	8811.64
9	200.8	10628.09	97.93	2616.36	155.22	5672.14	180.91	8811.64
10	200.8	10628.09	97.93	2616.36	155.22	5672.14	180.91	8811.64
MinSum	200.25	10746.90	97.93	2717.516	155.21	5683.47	180.85	8832.49

[7] S. Ramasubramanian and A. Harjani, "DIVERSION: A trade-off between link and path protection strategies," in *Proceedings of 9th Conference on Optical Network Design and Modelling*, February 2005, pp. 321-334.

[8] R. Ramaswami and K. N. Sivarajan, *Optical networks: A practical perspective*. Morgan Kaufmann, 2002.

[9] J. W. Suurballe, "Disjoint paths in a network," *Networks*, vol. 4, no. 2, pp. 125-145, 1974.

[10] J. W. Suurballe and R. E. Tarjan, "A Quick Method for Finding Shortest Pairs of Disjoint Paths," *Networks*, vol. 14, no. 2, pp. 325-333, 1984.

[11] J. K. Wolf, A. M. Viterbi, and G. S. Dixon, "Finding the best set of k paths through a trellis with application to multitarget tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 25, pp. 287-296, March 1989.

[12] C. Cheng, S. P. R. Kumar, and J. J. Garcia-Luna-Aceves, "A distributed algorithm for finding k disjoint paths of minimal total length," in *Proceedings 28th Annual Allerton Conference on Communication Control and Computing*, October 1990.

[13] D. Sidhu, R. Nair, and S. Abdallah, "Finding disjoint paths in networks," in *Proceedings of the Conference on Communications Architecture and Protocols*, vol. 21, August 1991, pp. 287-296.

[14] R. Bhandari, "Optimal diverse routing in telecommunication fiber networks," in *Proceedings IEEE INFOCOM*, vol. 3, June 1994, pp. 1498-1508.

[15] Y. Bejerano, Y. Breitbart, A. Orda, R. Rastogi, and A. Sprintson, "Algorithms for Computing QOS paths with Restoration," *IEEE/ACM Transactions on Networking*, vol. 13, pp. 648-661, June 2005.

[16] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows: Theory, Algorithm, and Applications*. Prentice Hall Inc., 1993.

[17] G. Eppstein, "Finding the k shortest paths," in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, vol. 1, Nov 1994, pp. 154-165.

[18] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Prentice Hall Inc., 1998.

[19] E. I. Chong, S. R. Maddila, and S. T. Morley, "On finding single-source single-destination k shortest paths," in *Proceedings of Seventh International Conference on Computing and Information (ICCI '95)*, July 1995, pp. 40-47.

[20] Ilog cplex 8.1. [Online]. Available: <http://www.ilog.com/products/cplex>