RAIRO Operations Research
Will be set by the publisher

TWO TREE-BASED ALGORITHMS FOR NETWORK SPARE CAPACITY DESIGN

Hovhannes A. Harutyunyan, Calin D. Morosan and Yunzan Zhang[1]

Abstract. Survivable network design has become increasingly important due to the need for reliable communication service. Its main purpose is to provide cost-efficient spare capacity reservation at certain survivability level. In this paper, we introduce two pre-planned path restoration algorithms for spare capacity design in mesh-like networks. First one is a spanning tree based algorithm, which needs less spare capacity than the well known hierarchical tree algorithm while keeping the same level of restorability. The second algorithm is a cycle based tree algorithm with backup parents and extra cycle edges, which forms cycles with the original spanning tree edges. Simulation results show that this algorithm works much better than the other two algorithms on restorability. Both algorithms have time complexity $O(n^3)$ and space complexity $O(n^2)$, where n is the total number of nodes in the network.

Keywords: network survivability, spare capacity design, restorability

Introduction

In the modern society, we are increasingly becoming dependent on the exchange of information. Multimedia communication, Internet, on-line banking, e-commerce are just a few examples of how we use networks daily. To assure reliable network communications we need to design a survivable network with suitable spare capacity which enables fault recovery via rerouting of traffic in the the case of a link failure.

October 24, 2005.

[1] Department of Computer Science, Concordia University, Montreal, Canada, {haruty, cd_moros, y_zhang}@cs.concordia.ca

Traditionally, there are two components that make a network survivable, namely *network design and capacity allocation*, and *traffic management and restoration* [6]. Network design and capacity allocation techniques try to cover system level failures, such as network link failures, by placing sufficient spare capacity on suitable spare links. The most important component, spare capacity design, is a design of the spare topology and the corresponding capacity, such that the network can reroute the demand if a link is lost due to a failure. Traffic management and restoration procedures seek to redirect the network load when the failures occur, such that the connections affected by the failure are restored while maintaining the network stability. These two components are complementary and cooperate to achieve reliable service upon failures.

Some metrics have been introduced in order to measure the spare capacity design and restorability performance:

- *Restorability* is the percentage of the working bandwidth that could be restored in case of a single link failure.
- *Work capacity* is defined as the sum of work bandwidth on every link using the shortest path routing.
- *Spare capacity* is defined as the sum of spare capacity on every spare link.

Based on the time necessary to allocate spare resources, survivability strategies can be classified as *pre-planned* and *dynamic methods*. The pre-planned methods have the advantage of guaranteing the survivability upon predicted failures. However, the pre-planned resources will be wasted if none of the predicted failures happens. The dynamic methods try to allocate the spare resources when the failure happens. In order to achieve better resource utilization the dynamic methods use the spare resources in case of failure. However, the survivability assurance is jeopardized because the requested resource might not be available when it is needed upon failures. In [11] these two methods have been employed together in order to obtain a good restorability. They called it *"2-step" restorability*, which means that the total restoration ratio is achieved when all useful preplanned paths are first exploited, followed by needed on-demand search for available paths using unallocated links plus unneeded spare capacity.

According to the initial locations during the rerouting process, restoration schemes are classified as *link restoration* and *path restoration*. In general, path restoration is known to require less spare capacity than link restoration. No matter which method is used, one is interested in determining a set of backup routes for the working traffic and the amount of spare capacity required on the backup routes to achieve a desired level of traffic restoration for a specific failure scenario. A typical goal is to determine a spare capacity placement such that all the normal traffic can be restored for any single failure in the network [1]. Some related work in network restoration and capacity allocation can be found in [15], [3], [5], [9], [12], [13], [7], [10], and [8].

Mesh-restorable networks, as opposed to ring-like networks, can be more capacity-efficient in terms of the total spare capacity. The term *mesh-restorable* refers to the

ability of the rerouting mechanism to exploit an arbitrary topology, in a mesh-like (as opposed to ring-like) way, using diverse rerouting mechanisms [11].

Mesh-based survivability is more capacity-efficient because each unit of spare capacity is reusable in many ways, across many different failures. Signals that pass through a failed link can be rerouted through many diverse paths. In this way, when considered in total, mesh-based survivability design requires smaller amounts of spare capacity compared to ring-based networks. In general, rings use very simple mechanisms and are thus simple and fast, but relatively capacity-inefficient. Mesh networks can be far more capacity-efficient, but need considerably longer time for a path restoration.

For spare capacity design, a spanning tree is a tree which covers all network nodes, with only one path between any node-pair. Network trees, a kind of mesh-restorable networks, can provide more flexibility and present a tradeoff between the restorability and restoration time. Trees are local by nature since adding or dropping a branch would have a limited impact over the other branches, as opposed to rings or cycles which lose their ring form if a single link on the ring is removed. This fact provides a potentially better chance in trees for dealing with multiple failures [4].

The two tree-based algorithms proposed in this paper use both pre-planned and path restoration methods. The basic spanning tree based algorithm generates a maximum spanning tree with backup parents, which needs much less spare capacity than the hierarchical tree algorithm presented in [4] while the restorability is similar to the hierarchical tree algorithm. The cycle based tree algorithm with backup parents and extra cycle edges produces a basic tree topology with backup parents and extra cycle edges, which form cycles with the original spanning tree edges. This one can get much higher restorability than the hierarchical tree algorithm or the spanning tree algorithm, with a reasonable increasing spare capacity. The most important objective is to achieve as large as possible restorability with a small spare capacity used. The two proposed algorithms have time complexity $O(n^3)$ and space complexity $O(n^2)$.

1. Definitions and Background

Formally, any network can be modelled as an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. Two nodes $u \in V$ and $v \in V$ are *adjacent* if there is an edge $e \in E$, such that $e = (u, v)$. We also say that node u and v are *neighbors*. A node in the graph represents a station in a network and an edge represents a connection between two stations, which creates a link between that two nodes. Throughout this paper we will use the terms edge and link interchangeably. A path p in a graph $G = (V, E)$ is a sequence of nodes of the form $p = v_1, v_2...v_n, (n \geq 2)$, in which each node v_i is adjacent to node v_{i+1}, where $1 \leq i \leq n-1$. The path p is also a sequence of edges or links. The length of a path is the number of edges in the path. A graph $G = (V, E)$ is said to be

connected if there is a path between any two nodes of G. We will assume that the network is represented by a connected graph.

The restorability is the percentage of restorable network traffic upon failures. In our work, we use the average fraction $\frac{restorable\ traffic\ capacity}{affected\ traffic\ capacity}$ to denote the term *restorability*, over all possible one link failures. For the pre-planned restoration algorithm, this measures the effectiveness of a pre-planned method in providing immediate restoration without requiring any on-demand computation for dynamic restoration.

We compare our algorithms to a previous known tree-based algorithm called *hierarchical tree algorithm*, which is presented in detail in [4]. This is a pre-planned path restoration algorithm and is based on the idea of a *hierarchical protection tree*, which provides hierarchical layering of the network. It defines a special spanning tree on which the links are arranged in a hierarchical form based on link capacity. In a hierarchical tree, the connection between every node and its parent has a larger bandwidth than the link between the same node and its children. The time complexity of the hierarchical tree algorithm is $O(n^3)$ [14].

2. The spanning tree algorithm

In this section we introduce a typical maximum weight spanning tree algorithm, which can achieve very close restorable percentage level when compared to the above hierarchical tree algorithm, but needs much less spare capacity. We will further refer to it as S.T. algorithm.

How to define the weight on each link to generate pre-planned trees? First approach can be based on the number of times the spare links can contribute to the shortest backup path. Initially, we find each shortest backup path upon any one-link failure and than we compute the total needed spare capacity according to the shortest backup path and assign this value to the associated weight for every link. Another possibility is to simply assign the work capacity on the link as its weight, then to construct the maximum spanning tree according to the obtained weights. Using these two methods on some examples we achieve similar restorability. Since the first method needs more time and space, we select the second one in our algorithms.

Once the link weights are assigned, Prim-Jarnik's algorithm is executed to find a maximum weight spanning tree, instead of a minimum weight spanning tree. A single copy of this tree is then configured as a pattern (unit link capacity) for the spare network. For any single link failure, we use this pattern to find all the existing spare backup paths and to assign the needed spare capacity to the spare capacity of the link. In the last step, following every possible one link failure, the maximum needed spare capacity per link will be the final designated link spare capacity.

Formally, the S.T. algorithm can be written as follows:

```
Input:  graph G(V, E)
Output:  spare capacities of E

assign the weight of each link according to its work capacity
compute maximum spanning tree T
assign unit link capacities for each link of T
for each link of T considered as a failure
    find all the existing spare backup paths
    assign the spare capacity to that link
end for
```

If the network spare topology is already given, it can be used as the basic spare graph. We can select our maximum spanning tree from all the possible connected spare links; the assigned link weight is its spare capacity limit. Then we assign to the tree link enough spare capacity (not exceeding the limit) in order to get a higher restorability. If we only have the network topology itself, we will use the S.T. algorithm to get the spare capacity allocation. In this paper, we consider the problem where no spare resource has been allocated in the given network topology.

As in the hierarchical tree algorithm, further refereed as the H.T. algorithm, we also use backup parents as a possible selection for another available backup spare path. Unlike the H.T. algorithm, the table with backup parents is established after the construction of the spanning tree, keeping the unused adjacent nodes for every node, including their parent node. Only the nodes that lose their connection to the tree need to switch to backup parents for other possible valid backup paths.

3. The cycle based tree algorithm

In this section, we present a new algorithm that we called *cycle based tree algorithm* for spare capacity design, further refereed as the S.T.C. algorithm. This algorithm can guarantee a higher restorability than the spanning tree algorithm described above. The algorithm constructs a spanning tree with backup parents and extra cycle edges that form cycles with the original spanning tree edges. The backup parents table keeps the unused adjacent edges for every node. When the nodes lose their connection to the tree, they can be switched to backup parents for other possible valid backup spare paths. More specific, we add some redundant edges in order to the basic spanning tree to achieve higher restorability with very little additional time complexity and reasonable increased spare capacity.

Here, one traffic demand between each node-pair in the network was assumed. The allocated working capacity for each network was determined using shortest path routing. The required spare capacity is calculated to provide fault tolerance for any single link failure.

3.1. The S.T.C. algorithm description

The S.T.C. algorithm is similar to the S.T. algorithm described in Section 2. In the first step we compute the shortest working path for every pair of node-pair, using Dijkstra's algorithm, assuming that the cost of every link on the traffic path is equal to 1. Then we get the connection cost table by calculating every link cost on every working path. Having this table we construct the maximum spanning tree with backup parents as in Section 2. In order to add the extra cycle links, we sort all the links in descending order into an array, according to the cost on every link. We take out the largest link from the sorted edge array and check whether it will form a cycle with the other links already on the tree. If there is no cycle, then we add this link to the tree-based links array. Otherwise we check whether we can add this link as a cycle link, that is, no other cycle contains this link. If it can be a cycle link, then we add it to the cycle links array. We continue until there are no more links in the sorted array. Now we can get the spare path for every node-pair just from the spanning tree. These are the original spare paths. In the next step we get backup parents for every node by checking all adjacent nodes and links and recording the nodes with maximum bandwidth as backup parents. In this way we obtain backup spare paths for every node-pair from the spanning tree links, the backup parents table, and the cycle links array.

Formally, the S.T.C. algorithm can be written as follows.

```
Input:  graph G(V, E)
Output:  backup spare paths, backup parents table, link cycles
Step 1: for each node-pair (v, u) in G
          compute the shortest working path of (v, u) using Dijkstra
        end for
        for each link e in E
          compute the weight of e according the working path
        end for
Step 2: sort the links from E in an array A, in descending order
Step 3: for each link e in A
          add e to the tree T
          if e forms a cycle
            if e belongs only to one cycle
              add e to T
              delete e from A
            end if
          end if
        end for
Step 4: for each node-pair (v, u) in T
          get the spare path of (v, u) from T
        end for
Step 5: for each node v in V
          get backup parents of v
        end for
```

Step 6: **for** `each node-pair` (v, u) `in` T
`get the backup spare path`
end for
`construct the cycle edge array`

From all the edges of the spanning tree and the cycle edges, we can get the cycles using the BFS (Bread First Search) algorithm. Then, from these cycles, we have two backup spare paths for every node-pair appeared on any cycle. We cannot guarantee that every node-pair on the graph will have these backup spare paths but if the node-pair has the backup spare paths, it means that it can cover any single failure on their working path or spare path due to the two link-disjoint backup spare paths. This is why we added these cycle edges to the spanning tree described in Section 2.

When we compute the restorability upon any possible single link failure we should check whether there exists another available spare path for any affected working path. If such a path exists, then the traffic on this working path is protected by the spare topology. The spare path can be derived either directly from the tree, from the the tree and the cycle edges, or from the backup parent table. Finally, using the definition described in Section 1, we compute the restorability given by the S.T.C. algorithm.

3.2. Time and space complexity

We denote by n and m the number of nodes, respectively edges, in graph. Following the formal description of the S.T.C. algorithm we have the complexity of each step:

- Step 1: $O(n^3)$.
- Step 2: $O(m \log m)$.
- Step 3: $O(mn))$.
- Step 4: $O(n^3)$.
- Step 5: $O(n^2)$.
- Step 6: $O(n^3)$.

Considering $m \in O(n^2)$ in the worst case we conclude that the time complexity of the S.T.C. algorithm is in $O(n^3)$. Since we have to store each pair of nodes the space complexity is $O(n^2)$.

4. Simulations and comparisons

For the tree-based algorithms used for simulation one traffic demand between each node-pair in the network was assumed.

We have used 4 topology models to do the simulation:

- Mesh-like topology, randomly generated, using three different values for the number of nodes (10, 17, and 21)
- AS-level Internet topology (generated by Inet) with 100 and 200 nodes

TABLE 1. Mesh-like random topology

Graph Model : mesh-like random topology							
	Work	**Restorability**			**Spare Capacity**		
n	**Cap.**	**H. T.**	**S. T.**	**S.T.C.**	**H. T.**	**S. T.**	**S.T.C.**
13	324	74.69%	74.69%	89.82%	181	181	207
17	640	74.53%	74.53%	86.56%	351	336	368
20	966	79.30%	79.30%	90.06%	545	526	569

TABLE 2. AS-level Internet topology

Graph Model : AS-level Internet topology (generated by Inet)							
	Work	**Restorability**			**Spare Capacity**		
n	**Cap.**	**H. T.**	**S. T.**	**S.T.C.**	**H. T.**	**S. T.**	**S.T.C.**
200	83112	63.13%	63.16%	77.79%	21053	15582	32759
200	83470	62.32%	62.34%	82.60%	20497	15490	40732
200	83582	61.80%	61.81%	81.46%	19908	12991	38740
200	83902	61.95%	61.98%	88.90%	19810	13292	47279
200	84774	61.24%	61.25%	78.21%	19726	14045	33224

- Random plat topology (generated by GT-ITM) with 100 and 200 nodes
- WAN-MAN-LAN topology (generated by Tiers) with 100 and 200 nodes

The detailed description of these models can be found in [2].

We have run over 200 experiments using the four topology models above. In Tables 1–4 and Figures 1–3 we present some of the results for each model. The meaning of the abbreviations in the tables is:

- n – The number of nodes in the graph
- H.T. – Hierarchical Tree algorithm with backup parents
- S.T. – Spanning Tree algorithm with backup parents
- S.T.C. – Spanning Tree algorithm with backup parents and cycle edges

The confidence interval for restorability varies from 0.3% to 5% for different models and different number of nodes. The confidence interval for spare capacity varies from 0.5% to 6% for different models and different number of nodes. For the seek of clarity we show the confidence interval only for the results regarding the S.T.C algorithm in Figures 1–3.

TABLE 3. Random plat topology

Graph Model : Random Plat (generated by GT-ITM)							
	Work	**Restorability**			**Spare Capacity**		
n	**Cap.**	**H. T.**	**S. T.**	**S.T.C.**	**H. T.**	**S. T.**	**S.T.C.**
100	22660	88.66%	88.27%	90.82%	5238	5046	5430
100	23086	88.16%	87.66%	90.68%	5873	5475	5899
100	24468	88.79%	88.22%	91.14%	6515	6391	6839
100	28676	84.22%	83.77%	86.65%	8518	8308	8734
100	31088	81.31%	81.49%	84.57%	10120	10065	10533

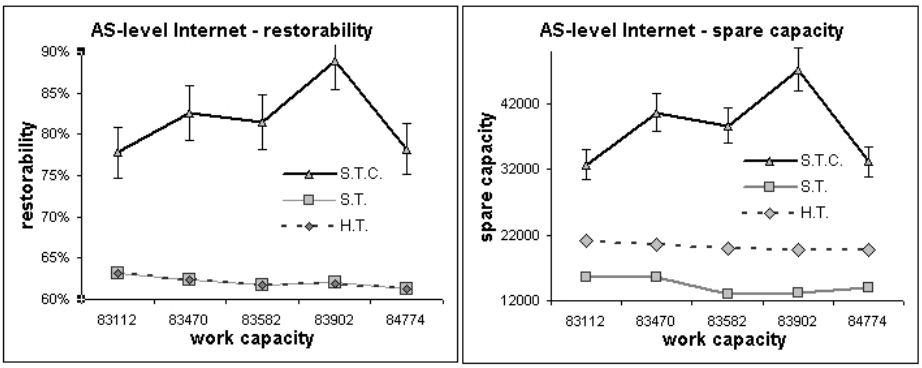
TABLE 4. WAN-MAN-LAN topology

Graph Model : 3 levels–WAN-MAN-LAN (generated by Tiers)							
	Work	**Restorability**			**Spare Capacity**		
n	**Cap.**	**H. T.**	**S. T.**	**S.T.C.**	**H. T.**	**S. T.**	**S.T.C.**
100	51546	44.24%	45.48%	45.98%	33576	33871	34021
100	52540	39.26%	41.51%	41.66%	40703	41117	41148
100	55688	36.34%	37.86%	38.16%	31467	32319	32421
100	56762	31.81%	34.28%	34.43%	33410	35124	35184
100	56922	31.55%	33.76%	33.91%	34609	35229	35272

Within the same complexity, for every model, the spanning tree algorithm needs almost always less spare capacity than the hierarchical tree algorithm. This is caused by the fact that the spare path in hierarchical tree is sometimes longer than that of in spanning tree. This tendency can be seen also in Figures 1-3, part a (left side). Although this difference can reach, in some cases, nearly 35% (Table 2), the differences in restorability are not significant.

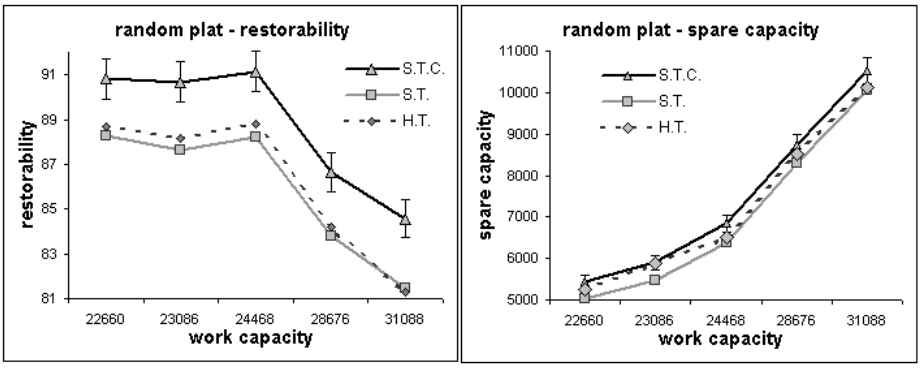
Although the restorability and the spare capacity can be very different for various network models, the restorability of the cycle based tree algorithm is always higher than the other two algorithms whereas the increased spare capacity is proportional to the increased restorability. This tendency can be seen in Figures 1-3, part b (right side). Note here that the increasing in spare capacity for AS-level Internet topology is particularly higher than for the other models.

The network redundancy can have a great effect on the restorability. The differences in restorability between the cycle based tree algorithm and the other two



a b

FIGURE 1. Restorability and spare capacity for AS-level Internet topology (200 nodes)



a b

FIGURE 2. Restorability and spare capacity for random plat topology (100 nodes)

is bigger in networks having higher redundancy than in networks having lower redundancy. This happens because more redundant edges can generate more cycles, which means that we can get more possible backup spare paths.

5. CONCLUSIONS

In this paper, we introduce two new spare capacity design algorithms: the spanning tree algorithm with backup parents (S.T.) and the spanning tree algorithm with backup parents and cycle edges (S.T.C.). We compare these algorithms to a

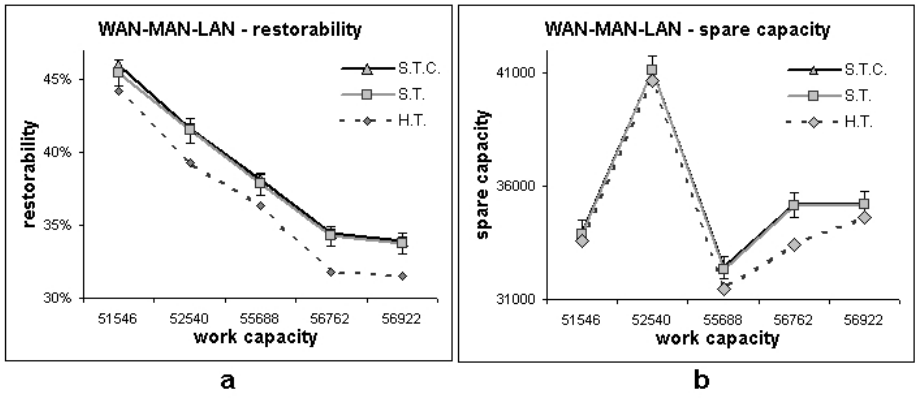


FIGURE 3. Restorability and spare capacity for WAN-MAN-LAN topology (100 nodes)

previous known algorithm, namely the hierarchical tree algorithm (H.T.). All of them can be used as pre-planned path restoration algorithms.

The spanning tree based algorithm needs less spare capacity than the hierarchical tree algorithm, improving in some cases with nearly 35% the spare capacity while the differences in restorability are not significant. The cycle based tree algorithm with backup parents and cycle edges gives clearly better results than the other two in terms of restorability, with a proportional increased spare capacity.

The restorability and the spare capacity can be very different for various network models. We used 4 topology models for simulations (mesh-like topology, AS-level Internet topology, random plat topology, and WAN-MAN-LAN topology). Within the same complexity, for each model, the restorability of the S.T.C. algorithm is always higher than the other two algorithms. In one example the difference even reaches nearly 27%. The time complexity of our algorithms is in $O(n^3)$ and the space complexity is in $O(n^2)$.

REFERENCES

[1] A. Al-Rumaih, D. Tipper, Y. Liu, and B. A. Norman. Spare Capacity Planning for Survivable Mesh Networks,in *Proceedings of the IFIP-TC6 / International Conference on Broadband Communications, High Performance Networking, and Performance of Communication Networks*, (2000) LNCS 1815:957–968.

[2] K. Calvert, M. Doar, and E. W. Zegura, Modeling Internet Topology, *IEEE Communications Magazine*, 35(6) (1997) 160–163.

[3] W. D. Grover and D. Stamatelakis. Cycle-Oriented Distributed Preconfiguration: Ring-like Speed with Mesh-like Capacity for Self-planning Network Rrestoration, in *Proceedings of IEEE International Conference on Communication (ICC'98)*, 1 (1998) 537–543.

[4] S. S.-Heydari and O. Yang. A Tree-based algorithm for protection/restoration in optical mesh networks, in *Proceedings of CCECE 2001*, (2001) 1169–1174.

[5] R. R. Iraschko, M. H. MacGregor, and W. D. Grover. Optimal Capacity Placement for Path Restoration in STM or ATM Mesh-Survivable Networks, *IEEE/ACM Transactions on Networking*, 6(3) (1998) 325–336.

[6] Y. Liu. Space Capacity Allocation: Model, Analysis and Algorithm, *Ph.D dissertation*, University of Pittsburgh, USA (2001).

[7] Y. Liu, D. Tipper. Spare Capacity Allocation for Non-linear Cost and Failure Dependent Path Restoration, In *Proceeding of the Third International Workshop on Design of Reliable Communication Networks, DRCN 2001*, Budapest, Hungary, (2001).

[8] Y. Liu, D. Tipper, P. Siripongwutikorn. Approximating optimal spare capacity allocation by successive survivable routing. *IEEE/ACM Transactions on Networking (TON)*, 13(1) (2005) 198–211.

[9] M. Médard, S. G. Finn, R. A. Barry, and R. G. Gallager. Redundant Trees for Preplanned Recovery in Arbitrary Vertex-Redundant or Edge-Redundant Graphs, *IEEE/ACM Transactions on Networking*, 7(5) (1999) 641–652.

[10] E. Modiano and A. N.-Tam, Survivable Lightpath Routing: A New Approach to the Design of WDM-Based Networks, *IEEE Journal on Selected Areas in Communications*, 20(4) (2002) 800–809.

[11] D. Stamatelakis and W. D. Grover. Network Restorability Design Using Pre-configured Trees, Cycles, and Mixtures of Pattern Types, *TRlabs Technical Report TR-1999-05*, (1999).

[12] D. Stamatelakis and W. D. Grover. IP Layer Restoration and Network Planning Based on Virtual Protection Cycles, *IEEE Journal on Selected Areas in Communications*, 18(10) (2000) 1938–1949.

[13] L. Szymanski and O. W. W. Yang, Spanning tree algorithm for spare network capacity, in *Proceedings of CCECE 2001*, (2001) 447-453.

[14] Y. Zhang, S. S.-Heydari and O. Yang. Complexity Analysis of the Hierarchical Tree Algorithm, In *Proceedings of the* 21^{st} *Biennial Symposium on Communications*, (2002) 46–50.

[15] B. Yener, Y. Ofek and M. Yung. Design and Performance of Convergence Routing on Multiple Spanning Trees, in *Proc. of Globecom'94*, 1 (1994) 169–175.