PERGAMON Mathematical and Computer Modelling 0 (2004) 1–0
www.elsevier.com/locate/mcm

Analysis of Backup Route Reoptimization Algorithms for Optical Shared Mesh Networks

Z. Bogdanowicz
US Army Armament Research and Development Center
Picatinny Arsenal, New Jersey, USA
`zbogdan@pica.army.mil`

S. Datta
Electrical Engineering, Princeton University
Princeton, New Jersey, USA
`datta@princeton.edu`

(Received July 2003; revised and accepted February 2004)

Abstract—In an optical mesh network operation, lightpaths are setup when requested, without prior knowledge of future requests. The concerns of increasing the sharing of backup paths and reducing capacity consumption are addressed on a best-of-now basis. As the number of lightpaths increases, it is possible to reroute existing paths to increase sharing further and free up more network capacity. This paper proposes several rerouting algorithms and compares their performance on test networks. All the described algorithms reroute the backup paths only and do not affect ongoing service. Also, the effect of network load factor and the number of lightpaths on the rerouting gain is examined. © 2004 Elsevier Science Ltd. All rights reserved.

Keywords—Rerouting, Re-optimization, Mesh restoration.

1. INTRODUCTION

The rapid expansion of the internet in the last decade has been made possible largely by optical backbone networks with high capacity and reliability. As capacity requirements continue to grow rapidly and unpredictably, mesh architecture has emerged as the solution for fast and efficient deployment of capacity in long-haul optical backbone networks. An optical mesh network consists of optical switches, interconnected by fiber links containing several optical channels. The basic service provided by the network is to set up an optical circuit, called a *lightpath*, across the network connecting two access nodes. An optical channel (could be a wavelength) is reserved in each link along the lightpath. If the switch has wavelength conversion capability, the lightpath may use different wavelengths in different links.

Clients at the edge of the network (e.g., IP/ATM routers) send large amounts of data using these lightpaths, hence, a restoration mechanism is needed to ensure reliable communication in

We would like to thank the referees for valuable suggestions that enhanced the contents of this paper.

the event of the failure of any network component. A common approach is to set up a dedicated backup path at the time of setting up the primary lightpath, and switch traffic to the backup path in the event of any failure in the primary path. This method, called the *1+1 protection* scheme, requires at least twice the amount of network capacity than the unprotected case. An alternate approach is to set up sharable backup paths. Two or more backup paths can share the same optical channel in a link, if their corresponding primary paths are link disjoint. Since the same link failure would not affect disjoint primary paths simultaneously, their backups are assured of getting a channel without contention in the event of any single link failure. This method, called the *shared mesh protection* scheme, significantly reduces the extra capacity requirement for protection. This paper is focused on shared mesh protected networks only.

In a typical scenario, a network in operation receives requests for lightpaths from time to time on hourly or daily basis, and accordingly sets up suitable primary and backup paths for the received request. Concerns of increasing the sharing of backup paths and avoiding capacity bottlenecks in the network makes the task of optimal routing more complicated than a simple shortest path approach. The routing decisions are made on a best-of-now basis, with no knowledge of the lightpaths to be set up in future. With time, the selection of routes becomes more and more suboptimal, and it is possible to reroute many of them to free up more network capacity and reduce blocking probability of new requests. Such a backup route reoptimization is expected to be performed on either weekly or monthly basis in a typical network. This paper describes the rerouting problem in some detail, outlining different algorithms, and dependence on network characteristics. As the rerouting of primary paths may cause short service disruptions, in this paper, we will focus on rerouting backup paths only. Also, in our model, wavelength conversion doesn't incur any extra cost, as is the case for OEO switches.

The rest of the paper is organized as follows. Section 2 mentions some of the available literature related to the subject. Section 3 describes the network model and defines the rerouting problem. In Section 4, we propose several heuristic algorithms to find a good solution to this problem. In Section 5, we present the performances of the proposed algorithms and follow it up with a study on the effect of network load and number of lightpaths on the rerouting gain. The paper is concluded in Section 6.

2. PRIOR WORK

The problem of selecting routes and wavelengths for one or a set of lightpaths in a WDM optical network is known as the *routing and wavelength assignment problem* (RWA). The problem could be a static (offline) one where the entire set of lightpath requests is known in advance, or a dynamic (online) one where lightpaths are routed as and when they are requested. An extensive survey of literature on the RWA problem for unprotected paths is provided in [1]. Reference [2] describes heuristics and ILP formulation for the routing problem for a network model similar to the one discussed in this paper, i.e., a mesh protected optical network, where the switches are wavelength conversion capable. References [3] and [4] describe other approaches for routing protected paths.

The issue of rerouting existing lightpaths in a dynamic routing scenario was addressed in [5] and [6], for unprotected lightpaths. In [2], a rerouting approach has been described for mesh protected paths. All of these papers use the rerouting step only when a new request is unable to find a feasible path and it becomes necessary to reroute some existing paths to free up capacity from the critical links on the path of the new request. This paper addresses the issue of re-optimizing the routes as a maintenance measure, not triggered by any particular new request, seeking to free up as much network resources as possible without any disruptions to the ongoing service. Some justification for such an approach has been provided in Section 5.2. Also, we do not alter the primary routes of the existing paths, and instead reconfigure only the backup paths which do not carry any traffic under normal circumstances.

3. PROBLEM DEFINITION

3.1. Network Model

The network model consists of a set of n nodes and m links, each link of a certain cost and capacity. A link consists of a set of channels connecting two nodes. The cardinality of such a set of channels defines capacity of a link. There are k bi-directional lightpaths in the system, each of which has been already assigned a primary and backup path using the online scheme described in Section 3.2. The objective is to remove the existing backup paths, and find a new feasible set of backup paths which minimizes some cost criterion.

Cost of Routing. The total number of channels used in the network for all backup paths is used as the cost criterion. The number of channels used by the primary paths is never changed by rerouting backups and, hence, is ignored.

3.2. Initial Assignment Algorithm

The initial primary and backup path assignment is done by addressing the lightpath requests one by one, in a given order. To find the best routes for one lightpath request, the shortest path between the terminal nodes is selected as the primary. Then, the links in the graph are examined for backup sharing possibilities.

(1) All links on the primary path are assigned infinite cost since the backup must be link disjoint with the primary. In addition, all not operational (i.e., failed) links are assigned infinite cost.
(2) If an operational link contains a channel which is being used by one or more of the other backup paths already established, and that channel can be shared by the new backup path (in other words, if the primary is link disjoint with the primaries of the other backup paths using that channel, and this channel is not used by any active backup path), then, that link is assigned zero cost.
(3) If no such sharable channel is found, then, an operational link is assigned its original cost.

After transforming the graph in this manner, the shortest path between the terminal nodes is selected as the backup path for the lightpath request. The process is continued for the next lightpath request after reverting to the original link costs. This process is sensitive to the initial ordering of the lightpath requests, and this fact will be used in the permutation based rerouting approaches described in Section 4.1.

4. HEURISTIC ALGORITHMS

The defined rerouting problem being NP-complete, several heuristic approaches were developed to find a good solution. They can be broadly classified as *permutation based* and *inheritance based* algorithms.

4.1. Permutation Based Rerouting Algorithms

As the routing algorithm in Section 3.2 is sensitive to the ordering of lightpath requests, different permutations of the list lead to different routing solutions with different costs. A routing solution is a set of primary and backup paths associated with each lightpath request. The principle assumption (or wish) behind the permutation based approach is as follows.

Given an algorithm to reroute an ordered list of lightpath requests (such as Algorithm A defined below), each permutation of the list corresponds to a particular routing solution, and the permutation that produces the least cost solution shall be selected.

In that case, the problem reduces to finding the best permutation. An exhaustive solution to this problem is still computationally intensive, which involves running the routing algorithm on $k!$ ordered lists. The rest of this section describes some of the heuristic approaches to seek the best solution from a reduced subset of permutations.

Algorithm A. The algorithm to reroute an ordered list of lightpath requests is similar to the initial assignment algorithm in Section 3.2, except that the primary paths are already routed. It is as follows.

```
Step 1. Remove all backup paths
Step 2. For each lightpath request,
            (a) Adjust link costs as in Section 3.2
            (b) Find shortest path as backup
            (c) Revert link costs
```

4.1.1. Random permutations

The simplest approach is to generate a certain number of random permutations of the list of lightpath requests, run the Algorithm A on each permutation, and keep the least cost solution. We do not pretend that it is a very sophisticated algorithm, but it serves as a good benchmark to evaluate other algorithms, and it executes in bounded time.

4.1.2. Binary reversal

The binary reversal algorithm is as follows.

```
Step 1. Mark the full list of lightpath requests as active, and route the list using Algorithm A.
Step 2. Reverse the ordering of the first half of the active region, route the full list using
Algorithm A and retain this ordering if it leads to a reduction in cost.
Step 3. Repeat the same for the second half.
Step 4. If the reversal of the first half had led to higher reduction on the cost than the reversal
of the second half – mark the first half as the new active region. Otherwise mark the second half.
Step 5. If the active region is empty, stop. Otherwise go to Step 2.
```

This scheme examines $2 \log_2 k$ permutations to search for the least cost solution.

4.1.3. GA-1

Genetic algorithms (GA) have been used extensively in all kinds of optimization problems including telecommunication networks [7,8], and has time and again proved its efficiency in seeking the minima in otherwise intractable solution spaces. The special case when each solution can be represented by a particular permutation of given elements, has been particularly well studied and off the shelf algorithms are available [9]. The main aspects of any GA are as follows.

Population. Start with a certain set of different solutions, from which new solutions can be created by suitably combining or changing them, hoping that it will progressively lead to better and better solutions.

Crossover. Given two solutions, create a new valid solution using them such that there is a possibility of inheriting the good properties of both parents.

Mutation. Given a solution, change it in a certain way to create a new valid solution.

Natural Selection. Eliminate higher cost solutions from the population to make way for newly created ones.

We used the following GA schematic, where each solution is represented by a particular permutation of the lightpath requests.

Step 1. Create a population of 20 members by *Josephus Permutation* [10] (with certain skip parameter) of the original list with random seeds.

Step 2. Select two members randomly from the population and generate two children by their crossover.

Step 3. Mutate each child with 50% probability.

Step 4. Randomly select two of the 10 costliest members of the population and replace them with the new children.

Step 5. Repeat Steps 2–4 for a certain number of iterations, and pick the least cost member of the population as the final solution.

To crossover two members, we use the *Partial Matched Crossover* method described in [9], where two crossover positions in the list are determined randomly and the segment between the two positions in each of the parent lists is replaced by the corresponding segment in the other parent. To make sure that the child is also a valid list with each element appearing once and only once, every time an element A is replaced by another element B—the original place of B in the list is replaced by A. To mutate, we made i random swaps in the list, where i is also a random number between 1 and $k = 3$, which makes sure that the list changes significantly, but not beyond recognition. A random swap is switching the position of two randomly selected elements in the list.

4.2. Inheritance Based Algorithms

Clearly, the permutation based schemes suffer from the following disadvantages.

(1) Merely reordering the list (always routed with Algorithm A, or any given algorithm for that matter) may not yield a solution close to the optimum.

(2) Once a set of routes have been determined by Algorithm A, subsequent attempts do not use the route information or attempt to improve upon them by directly modifying them. The inheritance-based schemes try to use the result of a previous routing and obtain a better set of routes.

4.2.1. Multiple pass heuristic

The simplest of inheritance-based schemes is the use of multiple passes. In Algorithm A each backup is routed using the knowledge of the backup paths of lightpath requests earlier in the list, but not of those later in the list. We can increase sharing by running another pass through the list, each time removing only one backup path, readjusting the link costs according to the sharing and disjoint constraints, and rerouting the backup path along the current shortest path. In fact, we can go on and on making several passes through the list, but our studies showed that beyond the second pass, gains diminish. By itself, the multiple pass scheme does not perform very well as it only finds a set of routes close to the original set. However, we found it useful to incorporate a second pass in Algorithm A and call it Algorithm B, which is outlined below. In our final implementation of all the rerouting schemes described in the previous section, Algorithm A was replaced by Algorithm B.

Algorithm B. The algorithm to reroute the backup paths of a given ordered list of lightpath requests is as follows.

4.2.2. GA-2

An inheritance based version of the GA, described in Section 4.1.3, is to use a population of different routing solutions (initially generated from different ordered lists using Algorithm B),

```
Step 1. Remove all backup paths.
Step 2. For each lightpath request,
            (a) Adjust link costs as in Section 3.2
            (b) Find shortest path as backup
            (c) Revert link costs
Step 3. For each lightpath request,
            (a) Remove its backup path
            (b) Adjust link costs as in Section 3.2
            (c) Find shortest path as backup
            (d) Revert link costs
```

instead of a population of the ordered lists themselves. In other words, instead of crossing over two ordered lists to create another ordered list, we try to cross over two sets of backup paths to create a new set of backup paths. The challenge is to design suitable crossover and mutation operations, which yield children that are valid routing solutions themselves, obeying link capacity restrictions.

Soft Parent Crossover. A single point of crossover is randomly selected in the list of lightpaths. The backups of all lightpaths prior to the crossover point are routed exactly as in the first parent (hard parent). The remainder of the lightpaths are considered one by one as in Step 3 of Algorithm B, except that the links used in the second parents (soft parent) backup path are also set to zero cost to tempt the child to use the same (or partially similar) path. In this manner, we derive a valid routing solution, parts of which are similar to either parents.

Cheap Node Mutation. All links to/from a randomly selected node are set to zero cost and all backups are rerouted as in Step 3 of Algorithm B. Other than the radically different crossover and mutation mechanisms, the general schematic of GA-2 is similar to GA-1.

5. RESULTS

5.1. Performance Comparison

We used several small and large test networks (A1, A3, A5, C1, E2) with randomly generated lightpath requests, to run the different rerouting schemes described. For the random permutation and the genetic algorithms, we ran 200 iterations in each run.

Rerouting Gain. The performance parameter mentioned in our results is the percentage reduction in the cost of routing after running the rerouting algorithm. In other words, the percentage reduction in the number of channels occupied by backup paths in the network.

Table 1. Rerouting gain of 4 algorithms on several test networks.

Test Network	A1	A3	A5	C1	C2
Number of Nodes	9	15	17	8	53
Number of Links	12	28	41	12	59
Number of Lightpaths	5000	9000	300	8000	3301
Rerouting Gain					
Random Permutation	0.53	1.43	16.67	0.0	0.0
Binary Reversal	0.48	1.43	16.05	0.0	0.0
GA-1	0.63	1.52	16.98	0.0	0.0
GA-2	0.63	1.82	19.75	3.31	3.78

From the results shown in Table 1, we can conclude that the GA-2 algorithm consistently outperforms the others. The other three algorithms yield almost similar results, with the GA-1 being marginally better. On a Pentium III 733 MHz Windows NT PC, the execution time of all the algorithms ranged from a few seconds to about ten minutes (for large lists) making them suitable for practical operations. The results for case A5 also throw up some observations about the effect of the number of lightpaths on rerouting gains. Smaller numbers tend to have better gains than the larger ones. To obtain better insight, we used the GA-2 algorithm on the same network with varying size of demand lists, and varying load factor, the results of which are reported next.

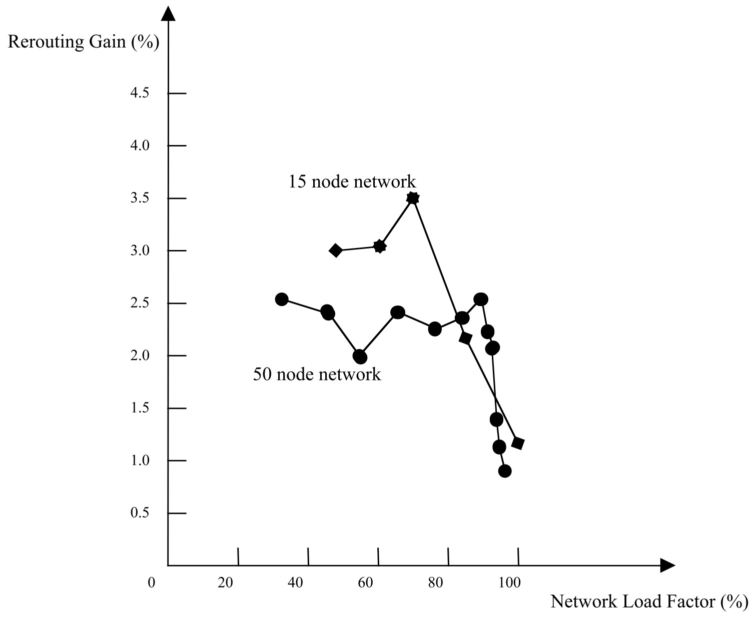

Figure 1. Dependence of rerouting gain on network loadfactor.

5.2. Load Factor

We define network load as the fraction of optical channels occupied in the network by either primary or backup paths, out of the total number available. For performance evaluation, we start with a given network with a set of lightpaths already routed (both primary and backup), and change the available spare capacities in the links (i.e., number of available spare channels in the links) to create different instances of the problem with different load factors. GA-2 was used as the rerouting scheme. Figure 1 shows the variation in rerouting gain with increasing network load for two networks. The gain starts out flat and starts to diminish after around 75% load region. The reason for this can be attributed to the reduced flexibility in moving existing backup paths to other routes because of the reduction in spare capacities available in links. This observation suggests that, in a dynamic lightpath routing scenario we should invoke the rerouting procedure before the network load reaches 75% (or some other mark, dependent on the network characteristic). Waiting for the first request to be blocked before attempting rerouting might be too late and reduce the efficiency of the rerouting process. This is particularly true if the rerouting process is restricted from changing certain things in the original allocation, like the primary paths in our version of the problem.

5.3. Number of lightpath requests

The other study we did was to compare the performance of the GA-2 rerouting scheme for lightpath request lists of different size, for the same network, and maintaining a similar network load factor. Figure 2 shows the variation in rerouting gain with increasing number of lightpath requests for two networks, both maintaining a load factor of around 70% in all test runs.

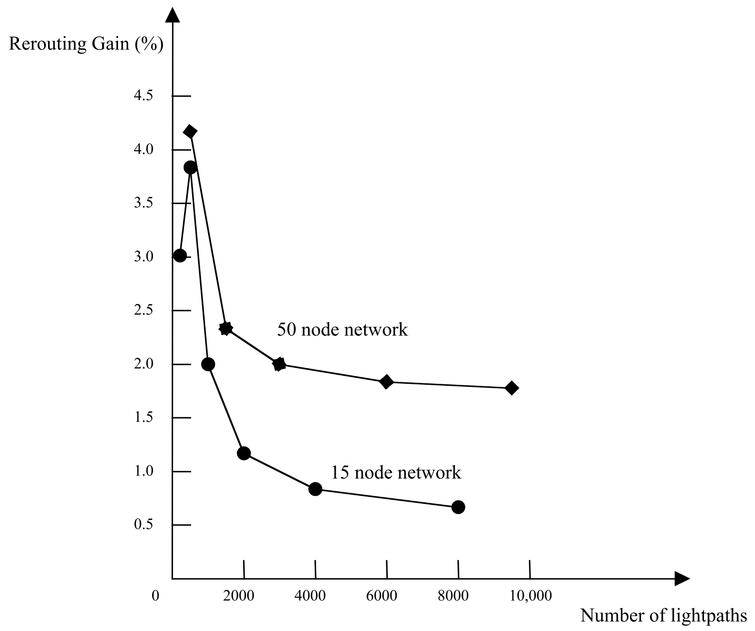

Figure 2. Dependence of rerouting gain on number of lightpaths.

The graph shows a consistent decline in the gain with increasing list size, suggesting either or both of two possibilities.

(1) A network with a large number of lightpaths has less potential for rerouting gain, as even simple online routing schemes can produce solutions not far from the optimum.

(2) The rerouting algorithms used by us are less efficient in finding a good solution of the rerouting problem, as the solution space becomes increasingly larger.

Only an optimal rerouting algorithm such as an Integer Linear Programming (ILP) approach can resolve the above question, even though they might be impractical for field operation.

6. CONCLUSIONS AND FURTHER WORK

We defined and discussed a rerouting problem for optical backbone networks using shared mesh protection, to address the issue of reducing the blocking of new lightpath requests. Several schemes were described of which a genetic algorithm based scheme, which developed new routing solutions from a population of existing routing solutions, was found to perform best.

We also showed that the rerouting gain in a network tends to decline after a certain level of network load has been reached. The rerouting gain also decreases with an increase in the number of lightpaths in the network.

As part of future work, we will investigate the effect of rerouting on the blocking probability of future requests. We will also try to develop ILP based schemes to explore the rerouting gain potential of networks, even though such algorithms may not be suitable for practical operation because of long execution times.

REFERENCES

1. J.Y. Yoo and S. Banerjee, Design, analysis and implementation of wavelength routed all-optical networks: Routing and wavelength assignment approach, `comsoc.org/pub/surverys/Yoo/yoo.htm`, May, 1997.
2. B.T. Doshi *et al.*, Optical network design and restoration, *Bell Labs Technical Journal* **4** (1), (1999).
3. G. Mohan *et al.*, Lightpath restoration in WDM optical networks, *IEEE Network*, (Nov/Dec 2000).
4. I. Chlamtac, A. Farago and T. Zhang, Lightpath routing in large WDM networks, *J. Selected Areas in Commun.* **14**, 909–913, (June 1996).
5. K.C. Lee and V.O.K. Li, A circuit rerouting algorithm for all-optical wide area networks, *IEEE INFOCOM*, 954–961, (1994).

6. G. Mohan and C. Siva Ram Murthy, A time optimal wavelength rerouting algorithm for dynamic traffic in WDM networks, *J. Lightwave Technol.* **17** (3), (March 1999).
7. J.Zhao *et al.*, A genetic algorithm for the design of multipoint connections in a local access network, *Intl. Conf. on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA)*, University of Sheffield UK, (September 1995).
8. L.A. Cox, J.R. Sanchez and L. Lu, Cost savings from optimized packing and grooming of optical circuits: Mesh vs. ring comparisons, *Optical Networks Magazine* **72**–**90**, (May-June 2001).
9. S. Ladd, *Genetic Algorithms in C++*, M & T Books, New York, (1996).
10. T.H. Cormen *et al.*, *Introduction to Algorithms*, MIT Press, Cambridge, (1989).