

# Optimization Problems in SONET/WDM Ring Architecture

Rahul Shah\*

December 11, 1998

## Abstract

We provide a model and some solution techniques for some problems arising in the design of survivable telecommunication networks utilizing fiber-optics based techniques. Given a set of network nodes and a set of possible links between them, we decide where to build links and where to place multiplexers. For this architecture to be survivable, we use ring architecture combined with point to point diverse path architecture. We explore computational complexities and provide some heuristic algorithms for the mathematical programming formulations of some problems.

## 1 Introduction

This essay briefly summarizes my work as a summer intern at Bellcore during summer 1998. I worked on some problems in survivable network design with Network Architecture group. In this essay my main emphasis is to give the reader a flavor of theoretical background along with some practical approaches for solving these problems. The essay shows some theorems along with empirical results.

The main objective in network planning is to design a survivable network using SONET<sup>1</sup>/WDM<sup>2</sup> equipment. We are given, as input, a set of nodes, distances between them, a set of possible links (forming a 2-connected graph) and a set of demands between pairs of nodes in terms of bit-rate. Our objective is to design a survivable architecture which would satisfy the demand constraints and would also be economical. By survivability, we mean that we want our architecture to survive single component (node or link) failure. So to route each demand we maintain two disjoint paths in our architecture between each pair of nodes having a non-zero demand between them. For switching between different possible routes we need multiplexers at nodes.

Of various design alternatives for 2-connected graphs, we use the ring architecture, which uses Add-Drop Multiplexers (ADMs) at some nodes. It may not always be possible to route a ring through given set of nodes in 2-connected graph. In some cases it may not be very economical

---

\*Summer Intern at Bellcore, e-mail: sharahul@paul.rutgers.edu

<sup>1</sup>Synchronous Optical Network

<sup>2</sup>Wavelength Division Multiplexing

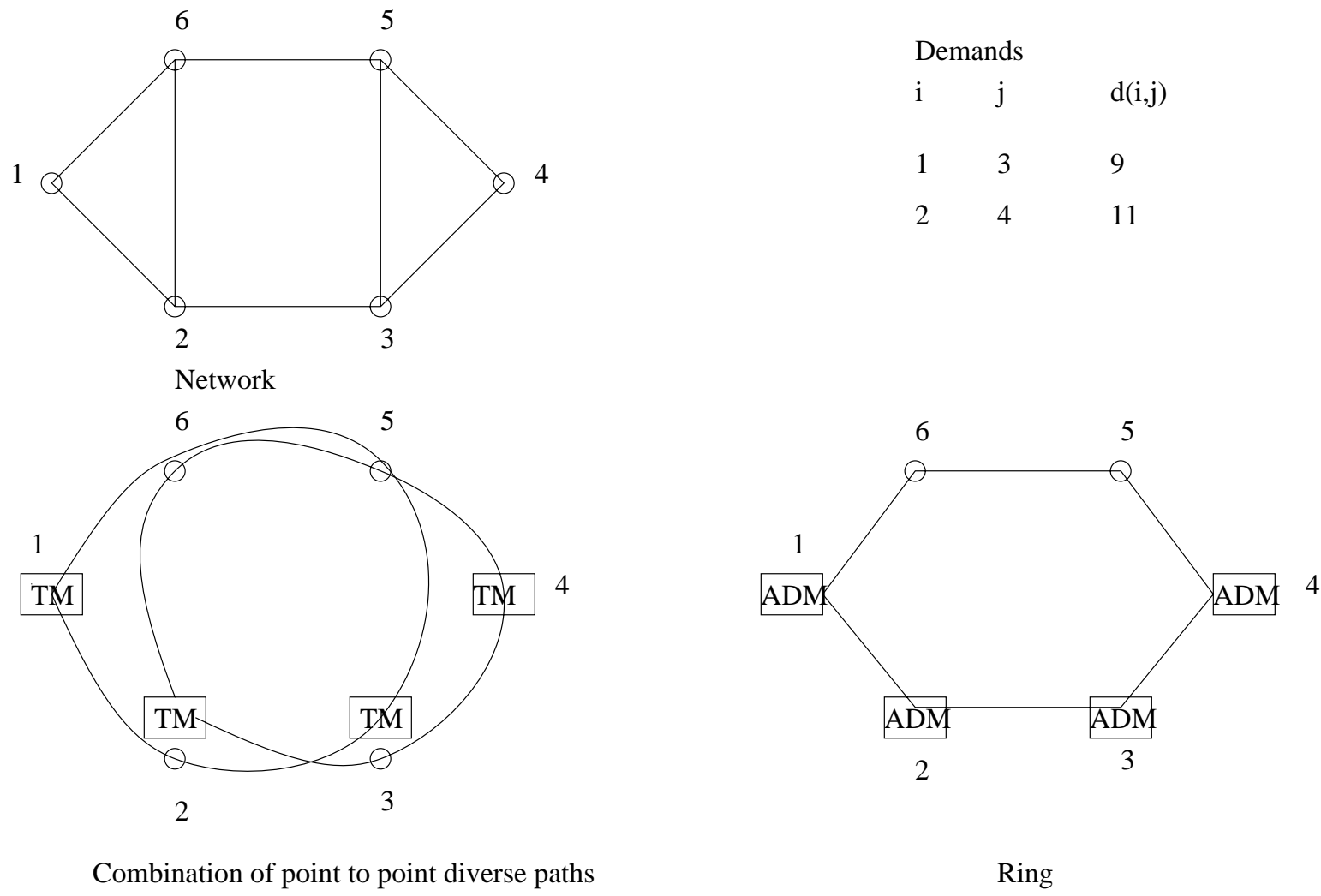


Figure 1: Various Design Alternatives

to do so. For such cases we use point-to-point diverse paths architecture which uses Terminal Multiplexers (TMs) at end-points. Figure 1 shows various design alternatives.

In overview we first select set of nodes which have the highest demand exchanges between themselves. We call this set *community of interest* (CoI). Then, given this set of nodes, we try to find a cycle passing through these nodes in the graph of possible links. This is to solve the *Ring-routing Problem* (RRP). Then we have two options for routing each demand in the ring. So we try to find the optimal routing of demands in the ring. This is to solve the *Ring-loading Problem* (RLP). This is discussed in section 3. Once we know the load on the ring, we find the bit-rates of components (multiplexers, fibers, amplifiers, regenerators etc). Then if that ring is economical enough, we retain it and remove all demands covered by this ring and try different ring-routing or ring node selection.

To determine whether the ring is economical or not we compare its cost with some benchmark cost. A typical benchmark could be the cost of point-to-point diverse paths architecture cost for the same set of nodes. But this usually gives preference to larger rings even when combination of smaller rings is more economical. This is because the benchmark cost is proportional to  $n^2$ , where  $n$  is the number of nodes in the ring, and hence it is very high for larger rings, indicating

that larger rings are economical. To correct this, another preferable benchmark could be the cost of *dual-hub architecture* which will be discussed in section 4. In section 2, we give some problem formulations. Section 5 summarizes the work.

## 2 Problem Formulations

Here we give the mathematical programming formulations and definitions of some problems. These are as given in [CDSW] and [ST].

### 2.1 Ring Routing Problem

Given a network  $G$  with node set  $N$ , an arbitrary subset  $M$  of  $N$ , and directed links with cost/distances/weights  $c_{ij}$  (some  $c_{ij}$  are infinite, indicating that the directed link does not exist), the ring routing problem can be formulated as follows:

RRP( $G, M$ ):

$$\text{Minimize } \sum_{i,j \in N} c_{ij} x_{ij}$$

subject to

$$\sum_{j \in N} x_{ij} = 1, \text{ whenever } i \in M \tag{1}$$

$$\sum_{i \in N} x_{ij} = 1, \text{ whenever } j \in M \tag{2}$$

$$\sum_{j \in N} x_{ij} \leq 1, \text{ whenever } i \in N - M \tag{3}$$

$$\sum_{i \in N} x_{ij} \leq 1, \text{ whenever } j \in N - M \tag{4}$$

$$\sum_{i,j \in S \subset M} x_{ij} < |S|, \text{ whenever } S \subset M, S \neq M \tag{5}$$

$$x_{ij} \in \{0, 1\} \tag{6}$$

Here,  $x_{ij} = 1$  indicates that nodes  $i$  and  $j$  are connected on a solution cycle and  $x_{ij} = 0$  indicates otherwise. Constraint sets (1), (3), (2) and (4) ensure that each member of the special set  $M$  is on a tour, and constraint set (5) prevents improper cycles (known as subtour elimination constraints). Constraint set (6) ensures that the variables are 0 – 1. A minimum cost of infinity for this problem indicates the absence of a cycle.

We do not discuss the solution to this problem here. However, we notice that it is very similar to Traveling Salesperson Problem (TSP) but on specific selected node subset. Hence various TSP heuristics can be used for solving this problem effectively. [W-91] discusses one such solution technique.

## 2.2 Ring Loading Problem

For a given cycle  $C$  through the set  $M$  of  $m$  nodes in an arbitrary network  $G$ , where between each node pair  $(i_k, j_k)$  there is a demand  $d_k \geq 0$  units, the ring loading problem is formulated as follows:

$RLP(G, M)$ :

Minimize  $z$

subject to

$$x_k + \hat{x}_k = 1, \quad k = 1, \dots, D \quad (7)$$

$$\sum_{k \in k_1(l)} d_k x_k + \sum_{k \in k_2(l)} d_k \hat{x}_k \leq z, \quad l = 1, \dots, m \quad (8)$$

$$x_k, \hat{x}_k \in \{0, 1\} \quad (9)$$

where  $k_1(l) = \{k | i_k \leq l \leq j_k - 1\}$

and  $k_2(l) = \{k | j_k \leq l \text{ or } l < i_k\}$

and  $D$  is the total number of demands.

Constraint sets (7), in conjunction with integrality constraints (9), state that each demand is routed in either clockwise ( $x_k$ ) or counterclockwise ( $\hat{x}_k$ ) direction. Constraint sets (8) state that  $z$ , the optimal capacity of the ring, must be as large as the loads on each of the links. For link  $(l, l+1)$  the load is the sum of  $d_k$  for clockwise routed demands between nodes  $i$  and  $j$  such that  $i \leq l \leq j - 1$  and counter clockwise routed demands between nodes  $i$  and  $j$  such that  $j \leq l$  or  $i > l$ . We discuss the solution to this problem in section 3.

## 2.3 Shortest Pairs of Disjoint Paths

Given  $G = (V, E)$ , a directed graph containing  $n$  vertices, one of which is a distinguished source vertex, and  $m$  edges, each with a non-negative length, we find the shortest pairs of edge-disjoint paths from the source to all other vertices.

In section 4, we describe an algorithm which does this in  $O(m \log_{(1+m/n)} n)$  time. We also discuss the applicability of this algorithm to dual-hub benchmark cost computation.

## 3 Ring Loading Problem

Here we describe some solution techniques for the ring loading problem (RLP) as formulated in section 2. We have to route each demand in either a clockwise or a counterclockwise direction, so that the maximum total demand passing through a link, called the *load of the ring*, is minimized. There are  $2^D$  possible solutions to RLP and we have to choose the one which minimizes the load. This problem is discussed in [CS-95].

### 3.1 Problem Complexity

The version of RLP where the integrality constraints (9) on  $x_k$ 's and  $\hat{x}_k$ 's are relaxed to

$$x_k, \hat{x}_k \geq 0$$

is equivalent to the case where the splitting of demands between the two directions is allowed. We know that this version could be solved in polynomial time [Kh79].

When demand splitting is not allowed, as is the case for ring designs being studied here, the problem is NP-complete. The proof of this is as follows. Consider an arbitrary case of the partition problem, which was shown to be NP-complete by Karp[K-72]. The problem is stated as follows: PARTITION PROBLEM:

INSTANCE :  $m$  integers  $s_1, s_2, \dots, s_m$ .

QUESTION : Is there a partition of  $\{1, 2, \dots, m\}$  into sets  $A_1$  and  $A_2$  such that  $\sum_{i \in A_1} s_i = \sum_{j \in A_2} s_j$  ?

This problem is easily cast as an instance of the integer RLP. To see this, consider a ring with two nodes, 1 and 2. Let demand  $d_k$  be set to  $s_k$ , for  $k = 1, \dots, m$  and assume that each demand originates at node 1 and terminates at node 2. Any feasible solution to this instance of ring loading problem partitions the set of demands to visit either link (1, 2) or link (2, 1). The corresponding load on the ring must therefore be at least  $(1/2) \sum s_i$ . If the optimal solution to the RLP is equal to  $(1/2) \sum s_i$ , then we can conclude that the answer to the partition problem is YES. Otherwise, the answer must be NO.

### 3.2 A Factor 2-Approximation Algorithm for RLP

Let me present a simple factor 2-approximation algorithm for ring loading problem that I came up with. The approach follows the principle of LP-duality. We can also get the same factor by LP-rounding. Let me first describe briefly how LP-rounding gives factor 2. Then we shall look at another factor 2-approximation algorithms using LP-duality which can be further generalised and used in practical heuristic algorithm.

The first step in approximation algorithm design is finding a good lower bound on optimal value  $OPT$ . Let  $OPT'$  be the optimal value of the LP-relaxation of RLP. Trivially,  $OPT' \leq OPT$ . Also we observe that  $OPT \leq 2OPT'$  : if we solve LP-relaxation of RLP with all demands doubled we get  $2OPT'$  as the optimal value and now if we retain only that part of each demand in LP (the solution to LP is splitting of each demand into two parts) which is at least half the demand, we get an upper bound on  $OPT$ . So theoretically, factor 2-approximation can be as simple as that but we shall now discuss another simple factor 2-approximation algorithm which follows from LP-duality and motivates a general family of 2-approximation algorithms which is called as weight based techniques. The theorems presented here are from [CS-95]. But here we present a different, simpler proof.

**Minimum-Link-Routing** : Route each demand  $d_k$  with origin node  $i_k$  and destination node  $j_k$  in the direction that has the smaller number of links, (i.e., the smaller of  $j - i$  and  $m - (j - i)$ ), with ties broken arbitrarily.

**Lemma 1** *The ring load associated with minimum-link-routing is at most the ring load associated with the routing in which each demand is routed in the direction opposite to that in minimum-link-routing.*

**Proof :** Let  $z_1$  be the ring load associated with minimum-link-routing and let  $z_2$  be the ring load associated with routing opposite to minimum-link-routing. Let link  $(i, i + 1)$  be a link with load  $z_1$  in minimum-link-routing. Consider the link  $(j, j + 1)$  opposite to this link where  $j = (i + \lfloor m/2 \rfloor) \bmod m$ . Now since the routing which gives load  $z_1$  follows minimum-link-routing, all the demands which form this load when rerouted in opposite direction pass through link  $(j, j + 1)$ . Therefore,  $z_2 \geq z_1$ .  $\square$

**Theorem 1** *The ring load associated with the minimum-link-routing is bounded above by twice that of optimal routing.*

**Proof :**

Let  $z^*$  denote the ring load associated with the optimal routing and let  $x_k^*$  be the values of decision variables  $x_k$  in the optimal routing. Let  $\bar{z}$  and  $\bar{x}_k$  be the corresponding values for minimum-link-routing. Now, without loss of generality, assume  $\bar{x}_k = x_k^*$  for  $k = 1, \dots, l$  and  $\bar{x}_k = 1 - x_k^*$  for  $k = l + 1, \dots, D$ . Let  $S$  be the set of all demands  $d_k$  where  $k = 1, \dots, D$ . Let  $S_1$  be the set of demands which have the same routing as in optimal (i.e. all demands  $d_k$  for  $k = 1, \dots, l$ ) and  $S_2 = S - S_1$  be the set of remaining demand which have routing opposite to the optimal. Let  $z_1$  be the load of the ring with demands in  $S_1$  routed by minimum-link-routing and  $z_2$  be the same for  $S_2$ . Let  $z'_2$  be the load of ring if each demand in  $S_2$  is rerouted in opposite direction than minimum-link-routing (i.e. as in the optimal routing of  $S$ ). Then we have

$$\bar{z} \leq z_1 + z_2 \tag{10}$$

Also,

$$z_1 \leq z^* \tag{11}$$

And from lemma 1 we have

$$z_2 \leq z'_2 \leq z^* \tag{12}$$

from (10),(11) and (12) we get,

$$\bar{z} \leq 2z^* \tag{13}$$

$\square$

Next question we address is, "Is there a family of tight examples for factor 2?" The answer is yes. Consider a ring with 5 nodes with 2 demands of unit size each between node pair (1, 2) and (1, 3) respectively. The minimum-link-routing gives 2 units as ring load while the optimal load is 1 unit.

To complete this discussion, we discuss a generalization of the minimum-link-routing, which is known as weight-based-routing. A weight-based-routing is any routing that determines a non-negative weight for each of the links and then routes each demand according to these weights. That is, each demand is routed in the direction where sum of the weights of links through which it passes is minimum. The minimum-link-routing is a particular case of this when all the weights are equal. This provides the same approximation guarantee and similar kind of family of tight examples exists. If we see the weights as the lengths of the links, we can see that the same proof works for proving factor 2. The only change is in the definition of opposite link, which is now the opposite link in the sense of distance. We state the following theorem without a formal proof.

**Theorem 2** *The ring load associated with any weight-based-routing is bounded above by twice that of the optimal routing.*

$\square$

### 3.3 RLP, its LP-relaxation and Dual

In this section, we discuss the LP-Relaxation of Integer Programming formulation (7),(8),(9) of Ring Loading Problem. We show a particular way of representing the dual linear program which will be used in the next section.

The LP-relaxation of the integer program given by equations (7),(8) and (9) can be written as :

Minimize  $z$

subject to

$$x_k + \hat{x}_k = 1, \quad k = 1, \dots, D \quad (14)$$

$$z - \sum_{k \in k_1(l)} d_k x_k - \sum_{k \in k_2(l)} d_k \hat{x}_k \geq 0, \quad l = 1, \dots, m \quad (15)$$

$$x_k, \hat{x}_k \geq 0 \quad (16)$$

where  $k_1(l) = \{k | i_k \leq l \leq j_k - 1\}$

and  $k_2(l) = \{k | j_k \leq l \text{ or } l < i_k\}$

and  $D$  is the total number of demands.

Now, let  $w_l, l = 1, \dots, m$  be dual variables corresponding to constraint set (15) and  $u_k, k = 1, \dots, D$  be the dual variables corresponding to constraint set (14). Then we get the Dual of this LP-relaxation as:

$$\text{Maximize } \sum_{k=1}^D u_k$$

subject to

$$\sum_{l=1}^m w_l = 1 \quad (17)$$

$$u_k \leq \sum_{l=i_k}^{j_k-1} d_k w_l \quad \text{whenever } k = 1, \dots, D \quad (18)$$

$$u_k \leq \sum_{l=j_k}^m d_k w_l + \sum_{l=1}^{i_k-1} d_k w_l \quad \text{whenever } k = 1, \dots, D \quad (19)$$

$$w_l \geq 0 \quad \text{whenever } l = 1, \dots, m \quad (20)$$

Only constraints from set (14) of LP-relaxation contribute to the dual objective function. Constraint (17) is obtained from the variable  $z$  in the LP-relaxation, (18) are obtained from

variables  $x_k$  and (19) are obtained from variables  $\hat{x}_k$ . We can replace the constraint sets (18) and (19) by

$$u_k = \text{MIN} \left( \sum_{l=i_k}^{j_k-1} d_k w_l, \sum_{l=j_k}^m d_k w_l + \sum_{l=1}^{i_k-1} d_k w_l \right) \quad \text{whenever } k = 1, \dots, D$$

This can be done because we know that in the optimal case, since we are maximizing the sum of  $u_k$ 's, each  $u_k$  will take the highest value allowed by the constraints (18) and (19). Further, if we replace the variables  $u_k$ 's by variables  $y_k$ 's, where  $y_k = u_k/d_k$ , we get the representation of the dual which will appear in the next section.

### 3.4 Some Practical Heuristic Algorithms

Here we discuss two heuristic improvement techniques over factor 2-approximation algorithm. The first approach is a greedy optimization of minimum-link-routing. At each stage, this algorithm finds a demand whose rerouting would result in a decrease of load, and then reroutes it. If there is no such demand, it tries the same thing with a pair of demands, i.e. it tries to reroute two demands at a time. It stops when there is neither a single demand which can be rerouted nor such a pair of demands. We call this algorithm OPT-2.

Another approach, which is an iterative weight-based-routing, is the dual-ascent method. This is as discussed in [CS-95]. It exploits the LP-relaxation of RLP and its associated dual LP, namely: DRL:

$$\text{Maximize } \sum_{k=1}^D d_k y_k$$

subject to

$$\begin{aligned} \sum_{l=1}^m w_l &= 1 \\ y_k &= \text{MIN} \left( \sum_{l=i_k}^{j_k-1} w_l, \sum_{l=j_k}^m w_l + \sum_{l=1}^{i_k-1} w_l \right) && \text{whenever } k = 1, \dots, D \\ w_l &\geq 0 && \text{whenever } l = 1, \dots, m \end{aligned}$$

The variables  $y_k$  represent the shadow prices for the constraints in set 7. That is, they measure the sensitivity of the optimal load to the amount demanded between each node pair. The variables  $w_l$ , which are associated with constraint set (8), measure the sensitivity on a per link basis.

Any feasible solution to DRL provides a lower bound on the optimal value of RLP.

$$\sum_{k=1}^D d_k y_k \leq z \quad \text{where } z \text{ is optimal link load}$$

The dual-ascent approach works on to find acceptable values for  $w_1, \dots, w_m$  by starting with some feasible assignment (for instance  $w_l = 1/m, l = 1, \dots, m$  as in minimum-link-routing), and successively changing the weights in a way that increases the value of the dual objective function,  $\sum d_k y_k$ , until some stopping criteria are met. We first route the demands according to the weights

Demands		
i	j	d(i,j)
1	2	2
1	4	5
2	3	9
2	4	11
3	4	4
3	5	3

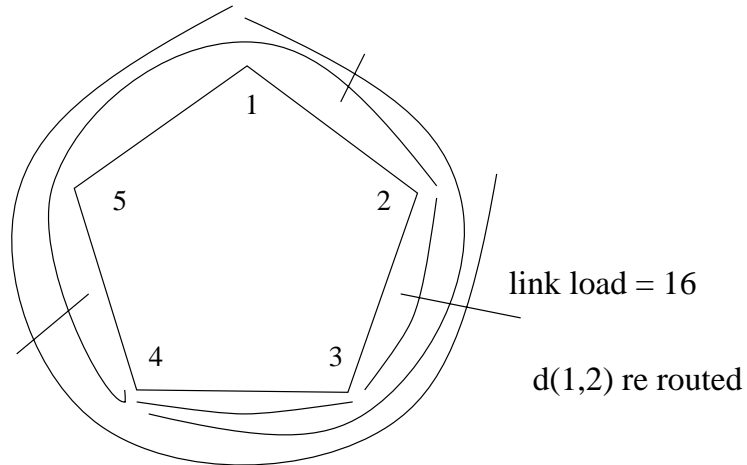
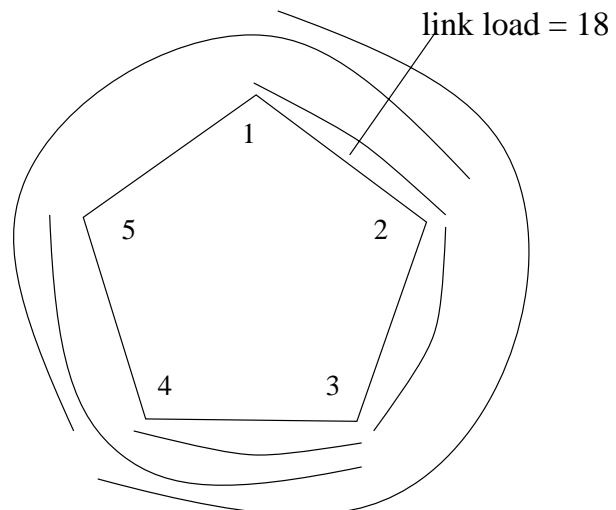
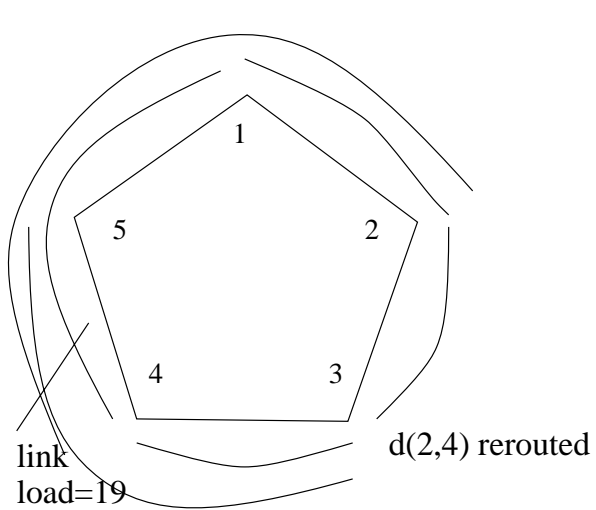
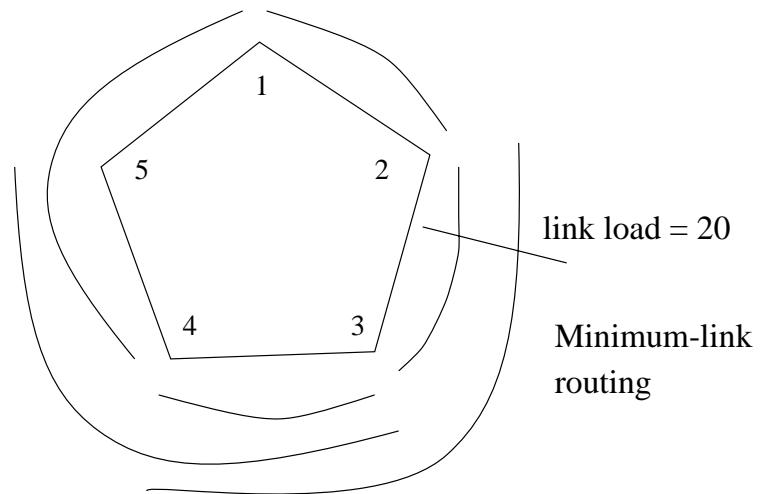


Figure 2: Steps in execution of OPT-2

and then at each step we pick some "pivot" node  $i$ . Let  $d_1$  be the sum of demands originating at  $i$  routed in clockwise direction and  $d_2$  is the same in counter-clockwise direction. If  $d_1 > d_2$  then set  $w_i = w_i + \delta, w_{i-1} = w_{i-1} - \delta$ , where  $0 \leq \delta \leq w_{i-1}$ . Otherwise if  $d_1 < d_2$  then set  $w_i = w_i - \delta, w_{i-1} = w_{i-1} + \delta$ , where  $0 \leq \delta \leq w_i$ . Now the demands are rerouted according to this new set of weights. The value of  $\delta$  is selected so that only one demand gets rerouted by this weight adjustment. This is repeated up to some predetermined number of iterations or up to the point when optimum of DRL is found.

Note that an increase in the dual objective does not necessarily mean a decrease in the ring load. So we have to maintain the solution which gave the minimum ring load. Essentially, this is like exploring the solution space near the dual LP optimal to find the best possible approximate solution.

This kind of dual-ascent approach has been successfully applied in variety of applications, including facility location ([BK77],[E-78]), network design ([BMW89]) and database location ([FH80]).

We note that both OPT-2 and dual-ascent method give results within a 10% approximation error range most of the times. The table shows approximate loads and running times for some typical ring and demand sizes. Figure 2 demonstrates the iterations of algorithm OPT-2 for  $m = 5$  and  $D = 6$ . There is no assignment of weights which can give optimum of 16 in this example, so the dual-ascent does not do as well in this example.

Problem Size (m,D)	OPT-2		Dual Ascent		OPTIMAL load
	load	time(s)	load	time(s)	
(5,6)	16	0.03	18	0.03	16
(7,11)	18	0.05	18	0.06	18
(7,22)	45.5	0.06	45.5	0.11	45
(10,30)	85	0.09	77	0.17	77
(13,28)	96	0.06	96	0.16	96
(15,50)	136	0.16	130	0.32	126

## 4 Dual Hub Benchmark

Dual Hub architecture is an architecture in which two nodes are selected as hubs. Then routing of any demand between node  $i$  and  $j$  is through a hub. We require two hubs for network survivability. One hub is reserved as a protection hub. This architecture consists of terminal multiplexers at nodes and digital cross connect system at hubs. For this, we require from each node a pair of vertex disjoint paths going to two different hubs in the undirected graph  $G$  of possible links. We convert this problem to finding a pair of link-disjoint paths from single source  $s'$  to every other vertex in a directed asymmetric<sup>3</sup> graph  $G'$ .

For this we first add a new vertex  $s$  and two new edges  $(s, h_1)$  and  $(s, h_2)$  where  $h_1, h_2$  are the hubs in  $G$ . Now for each vertex  $v$  in this graph add two vertices  $v_1, v_2$  and a directed link  $(v_1, v_2)$  in  $G'$  and for every undirected edge  $(u, w)$  add two directed links  $(u_2, w_1)$  and  $(w_2, u_1)$ . A pair of link disjoint paths from  $s_2$  to  $v_1$  in  $G'$  corresponds to a pair of vertex disjoint paths from  $v$  to each of the hubs  $h_1$  and  $h_2$  in  $G$ .

We now concentrate on the algorithm to find a pair of link-disjoint paths from vertex  $s$  to vertex  $v$  in a directed asymmetric graph.

<sup>3</sup>if  $(u, v)$  is an edge then  $(v, u)$  is not an edge

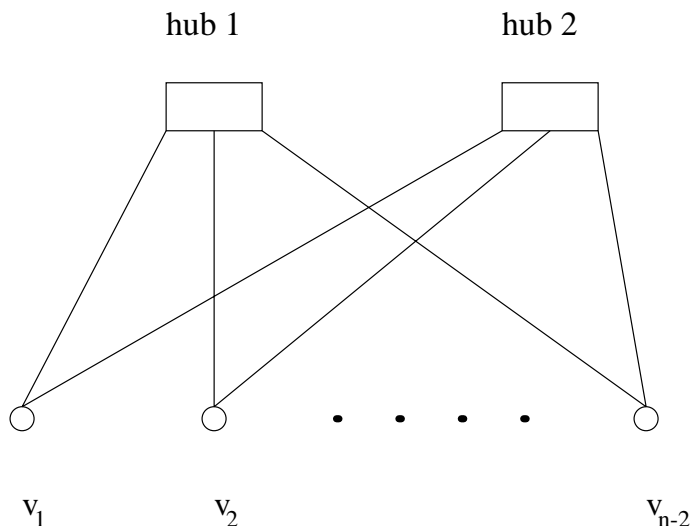


Figure 3: Dual Hub Architecture

#### 4.1 An Algorithm for Shortest Pairs Problem

Given a directed asymmetric graph  $G = (V, E)$  with a special node  $s$  and cost  $c(u, v)$  on each edge, we describe an algorithm ([ST]) to find two link disjoint paths from  $s$  to all other vertices. We use following steps, consisting of two passes of Dijkstra's algorithm [D-59].

**Step 1:** Find a shortest path tree  $T$  rooted at  $s$ . Such a tree contains, for every vertex  $v$ , a shortest path from  $s$  to  $v$ . Compute  $d(s, v)$ , the shortest distance from  $s$  to  $v$  for every vertex  $v$ .

**Step 2:** Transform the length of every edge  $(u, w)$  in  $G$  by defining  $c'(u, w) = c(u, w) - d(s, w) + d(s, u)$ . Call this graph  $G'$ .

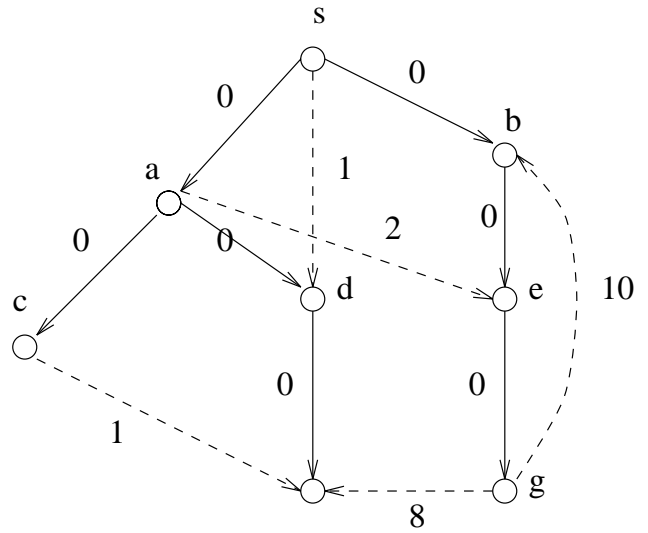
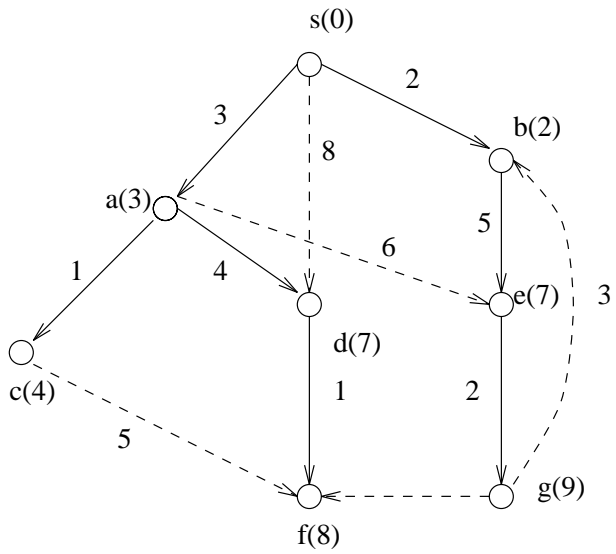
**Step 3:** For each vertex  $v$  construct a graph  $G_v$  by reversing the directions of edges on the shortest path from  $s$  to  $v$  in  $G'$ . Now find a new shortest path from  $s$  to  $v$  in  $G_v$ . So we have pair of paths from  $s$  to  $v$ . Eliminate common links from these two paths to get a shortest pair of link disjoint paths from  $s$  to  $v$ .

The pair of paths returned by this algorithm has the shortest total distance. For proof see [ST]. Figure 4.1 shows the execution of this algorithm. Dijkstra's algorithm can be implemented in  $O(m \log_{(1+m/n)} n)$  time using  $d$ -heaps data structure ([J-77],[T-83]). So this algorithm also requires  $O(m \log_{(1+m/n)} n)$  time to compute one such pair.

#### 4.2 An Efficient Algorithm for Single Source-All vertex Shortest Pairs of Disjoint Paths

We note that step 3 of the algorithm in section 4.1 has to be repeated  $n - 1$  times because  $G_v$  for each vertex  $v$  is different. So doing it this way takes  $O(nm \log_{(1+m/n)} n)$  time. We obtain a faster algorithm by realizing that these graphs  $G_v$  for different vertices  $v$  are related. We can, in effect, run Dijkstra's algorithm in parallel on all of them, obtaining  $d_v(s, v)$  for all vertices  $v$  in single Dijkstra like calculation. This gives us  $O(m \log_{(1+m/n)} n)$  time algorithm. The details of the algorithm and implementation issues can be obtained in [ST].

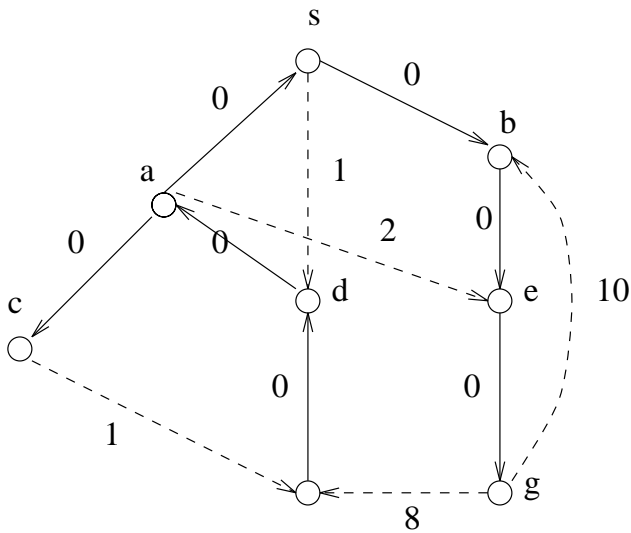
This algorithm was implemented for dual-hub benchmark cost computation.



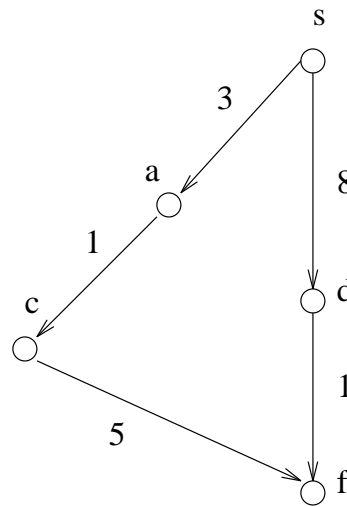
G with  $c(u,v)$  for each edge and  $d(s,v)$  for each vertex

———> Tree edges  
 - - - -> Non tree edges

G' : Shortest path from s to f is s,a,d,f with distance 8



G<sub>f</sub>: Shortest path from s to f is s,d,a,c,f with distance 2



Shortest pair of paths formed after eliminating common links. Total distance = 18 = 2\*8+2

Figure 4: Steps in Shortest pair of disjoint paths algorithm

## 5 Conclusions

Here we have presented some algorithms used in SONET/WDM network architecture design. WDM network design tool is being developed at Bellcore. Some of the algorithms are directly taken from SONET toolkit, some are modified either for improvement or for appropriate applicability to WDM equipment and there are some new algorithms developed for finding optimal WDM cost and optimal placement of amplifiers and regenerators. The algorithm for the ring loading problem has been improved while the one for dual-hub benchmark is newly implemented.

## 6 Acknowledgements

The author would like to thank Dr. Ondria Wasem for her guidance during the work, Dr. Vasek Chvatal for his supervision of this essay and Avni Rambhia for proof-reading this essay.

## References

- [CDSW] S. Cosares, D.N. Deitsch et al, "SONET Toolkit: A Decision Support System for Designing Robust and Cost-Effective Fiber-Optic Networks", Interfaces, Vol. 25, No. 1,1995.
- [W-91] O.J. Wasem, "An Algorithm for Designing Rings for Survivable Fiber-Optic Networks", IEEE Transactions on Reliability, Vol. 40, No. 4,1991.
- [CS-95] S. Cosares, I. Saniee, "An optimization problem related to balancing loads on SONET rings", Telecommunication Systems,1995.
- [K-72] R. Karp, "Reducibility Among Combinatorial Problems", in Complexity of Computer Computations, 1972.
- [Kh79] L. Khachiyan, "A Polynomial Algorithm in Linear Programming", Soviet Math. Dokl.,1979.
- [AHU] A. Aho, J. Hopcroft, J. Ullman, The Design and Analysis of Computer Algorithms. Addison-Wesley,1974.
- [J-77] D. Johnson, "Efficient algorithms for shortest paths in sparse networks", Assoc. Comput. Mach. 24,1977.
- [T-83] R. Tarjan, "Data structures and network algorithms", Soc. Ind. Appl. Math, 1983.
- [D-59] E. Dijkstra, "A note on two problems in connexion with graphs", Numer. Math. 1,1959.
- [BK77] O. Bilde, J. Krarup, "Sharp lower bound and efficient algorithms for simple plant location problem", Annals of Disc. Math 1,1977.
- [BMW89] A. Balakrishnan, T. Magnanti, R. Wong, " A dual-ascent procedure for large scale network design", Operations Research 37,1989.
- [E-78] D. Erlenkotter, "A dual-based procedure for uncapacitated facility location", Operations Research 26,1978.
- [FH80] M. Fisher, D. Hochbaum, "Database location in computer networks", Journal of the ACM 27,1980.

- [GJ79] M. Garey, D. Johnson, Computers and Intractability: A Guide to NP-Completeness, W.H. Freeman, 1979.
- [ST] J. Suurballe, R. Tarjan, "A Quick Method for Finding Shortest Pairs of Disjoint Paths", Networks, Vol. 14, 1984.