2003

Optimal placement of add-drop multiplexers in a multi-ring network

Karen A. Kelly
Lehigh University

Kelly, Karen A.

Optimal Placement of Add/Drop Multiplexers in a Multi-Ring Network

May 2003

Optimal Placement of Add/Drop Multiplexers in a Multi-Ring Network

by

Karen A. Kelly

A Thesis

Presented to the Graduate and Research Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science.

in

Information and Systems Engineering

Lehigh University

May 2003

This thesis is accepted and approved in partial fulfillment of the requirements for the Master of Science.

April 25, 2003
Date

Thesis Advisor

Chairperson of Department

Table of Contents

List of Tables

List of Figures

Abstract

The interconnected stacked ring problem arises in the capacity planning of telecommunications networks using Synchronous Optical NETwork (SONET) technology. Given a topology of interconnected, bidirectional SONET rings and a set of demands between node pairs, the objective of the problem is to determine how to route each demand and where to place add-drop multiplexers (ADMs) on each stack of each ring in such a way as to minimize the total number of ADMs required. In this thesis, three integer programming formulations are presented for the problem. The formulations are extensions of path-based and edge-based formulations for the multi-commodity flow problem. A set of constraints that strengthens the linear programming relaxation bound is identified. Solution algorithms using the CPLEX 8.1 Callable Library are implemented and computational experiments are conducted to compare the formulations. The computational results suggest that the path-based formulations are better than the edge-based formulation based on computation time in combination with modeling flexibility. Future research directions, including ideas regarding column generation implementations, are discussed.

1. Introduction

Many telecommunication networks today are deploying the technology known as the Synchronous Optical NETwork (SONET). One of the main reasons why SONET is appealing is because it is a self-healing network technology. Self-healing means that there are two strands of fiber on each physical ring, the working ring, which handles all of the traffic, and the protection ring, which remains on standby. With this arrangement, if there is a failure on the working ring, SONET is capable of automatically detecting the failure and transferring control to the protection ring quickly, within 50 milliseconds. This ability to react quickly to problems has become increasingly important due to the fact that individual links carry so much bandwidth, and even a single link failure would cause many disruptions.

A popular configuration for SONET is a bidirectional ring, in which nodes are connected by a ring (or multiple rings) of fiber and traffic can be routed in both the clockwise and counterclockwise directions. A node is able to send, receive or relay messages using a device called an add-drop multiplexer (ADM). ADMs are required at any node where traffic originates or terminates. For the SONET architecture, the cost of the ADMs placed at the nodes in the network is a much larger proportion of the total cost than the cost of the fiber along the links.

Given a communications network with a set of nodes interconnected by a ring of optical fiber, a typical problem faced by a network planner is to determine how to

route a set of point-to-point demands through the network in such a way as to minimize cost. In the operations research literature, the network planning problem has been modeled in two ways. The ring loading problem, which is the problem of minimizing the number of stacked SONET rings necessary to carry a given amount of traffic, has been studied by a number of researchers [1, 2, 3]. In the ring loading problem, it is assumed that every node on every stack is equipped with an ADM. The cost function of this problem only approximates the cost of installing a SONET ring network, because it is based solely on the cost of the capacity of fiber needed to satisfy all the demand. While the cost of fiber can be an important consideration, the cost of the electronic equipment, the ADMs, tends to dominate the total cost. Therefore, a cost function that considers the number of ADMS is appropriate. Several researchers [4, 5, 6, 7] have studied this problem of minimizing the number of ADMs in a SONET ring, which is commonly known as the ring grooming problem, but consider just a single ring.

In this thesis, we extend the ring grooming problem to a multi-ring SONET network. The multi-ring SONET network consists of multiple bidirectional rings that are always connected to at least one other ring. Interconnected rings share two common nodes, which are referred to as interconnection nodes. In a multi-ring topology, ADMs are required at interconnection nodes where traffic is transferred from one ring to another as well as at nodes where traffic originates and terminates. Given a set of point-to-point demands and a description of the network topology, the objective of the interconnected stacked ring problem is to determine the optimal route for each

demand, the optimal number of stacks for every ring and the location of ADMs on each stack for every ring. The main contribution of this thesis is the exact formulation of the interconnected stacked ring problem, which has not appeared previously in the literature.

Since the rings are bidirectional, demand can be routed in either direction around the ring. The total capacity used on each link is the sum of all of the flows in both directions along that link. A requirement of SONET technology is that the same amount of capacity must be installed on each link of a ring. Therefore, the link of the ring with the maximum load determines the required capacity on each stack of a ring.

The organization of this thesis is as follows. In Section 2, the relevant literature is reviewed. In Section 3, a formal statement of the problem is given, and three formulations are presented. Also, an example problem is shown and differences among the formulations are discussed. In Section 4, the data required and the design of the computational experiments are discussed. In Section 5, the results of testing are shown and evaluated. In Section 6, conclusions are presented and areas for future research are discussed.

2. Literature Review

The interconnected stacked ring problem is an extension of two problems that have appeared in the operations research literature. During the past decade, a number of

researchers have studied the ring loading problem and the ring grooming problem. However, all of the papers in the literature have considered the single ring version. Recently, a multi-ring version of the problem was presented [8] and a heuristic solution method was described. This thesis is an extension of that work.

The ring loading problem involves an undirected network in a single ring configuration with nodes indexed clockwise from 1 to n and a set of bidirectional demands d_{ij} where $i<j$. The objective of the problem is to determine a routing direction (clockwise or counterclockwise) for each demand in order to minimize the capacity of the ring. The cost function of this problem only approximates the cost of installing a SONET ring network, because it is based solely on the cost of the capacity of fiber needed to satisfy all the demand.

There are two versions of the ring loading problem that can be considered, one in which demands are not allowed to split and one in which they can split. The version of this problem in which demands are unable to split (demand must be completely routed in a clockwise or counterclockwise direction), is known as an origin-destination integer multi-commodity flow problem, which is NP-hard. See [2]. The version of the problem in which demands may be split into integer parts is equivalent to an integer multi-commodity flow problem, which in general is NP-hard. See [3]. This problem is much more difficult than the one in which demands may be split arbitrarily. In [1], Cosares, Deutsch, Saniee and Wasem describe the SONET Toolkit, which is a tool that reads in data about the network, its embedded capacity,

the available equipment, the customer demands and the protection requirements and produces a topology that satisfies the demands and protection requirements. The model in the SONET Toolkit attempts to minimize the total cost of the network. This problem is very different from the problem in this thesis since it is assumed that the topology is given here whereas the topology in the SONET toolkit is to be determined.

The other standard problem is the ring grooming problem. This problem deals with "stacked" rings, where each stack is routed over the same cycle of optical fiber cables, but each stack serves only a subset of the nodes along the cycle. This problem routes all of the demands in such a way as to minimize the number of ADMs used in the network. The cost of the network is proportional to the number of ADMs in the ring grooming problem. In [5], an approximation algorithm for uniform traffic is designed and a new lower bound for the case of uniform traffic is computed. An exact algorithm based on integer column generation is provided in [6]. A unidirectional version of this problem is described in [7]. The authors describe a heuristic design methodology that "builds" stacks by incrementally adding ADMs and assigning demands. Finally, [4] presents a modified version of the problem in which the authors are presented with a set of nodes and demands and are trying to design the topology of the network. This can be viewed as a complex version of a vehicle routing problem with a single depot (the hub of the network) and multiple vehicles (each tracing out a ring). The main difference in their problem is that the cost function used is not associated with the number of ADMs, instead a link that is used

by some ring family incurs a fixed cost plus a variable cost per stack associated with that family.

3. Formulations

In this section, three formulations are presented for the interconnected, stacked ring problem. Given a topology of interconnected, bidirectional SONET rings and a set of demands between node pairs, the objective of the problem is to determine how to route each demand and where to place ADMs on each stack of each ring in such a way as to minimize the total number of ADMs required.

The formulations are extensions of the multi-commodity flow problem, which is a network flow problem in which there are multiple commodities, each with their own origin and destination, that share the capacity on an arc. There are two standard formulations for multi-commodity flow problems, a path-based approach and an edge-based approach.

In the path-based model, for each demand pair, all possible paths between the origin and destination are generated. The model determines the amount of flow to assign to each path for each commodity. From a practical standpoint, the path-based formulation is appealing in that it is easy to restrict the set of candidate paths to a set of paths that actually would be implemented. One problem however is that, as the size of the network increases and the number of demands increases, the set of

candidate paths becomes extremely large. Generally, for anything other than toy-sized problems, the number of path variables makes the explicit formulation computationally intractable, but column generation techniques can be developed to solve instances effectively.

There are two versions of the path-based formulation. In one version, a node pair's demand is allowed to be split in integer quantities across multiple paths. In another version, a node pair's demand must be assigned to exactly one path. (In the literature, the no-splitting version is referred to as the origin-destination integer multi-commodity flow problem.)

In the edge-based approach, the model determines the amount of flow for each commodity on each arc. Candidate paths are not predefined for each demand. Instead the end-to-end connection between a demand's origin and destination is guaranteed by flow conservation constraints that state that for every demand the flow out of a node minus the flow in to a node equals the demand quantity at the origin, equals the negative of the demand quantity at the destination, and equals zero at all other nodes. Because of this structure, the number of variables in the edge-based approach is greatly reduced compared to the path-based approach; however, the number of constraints is much larger than that of the path-based approach.

3. 1. Path-Based Formulation (allows demand splitting)

In this section, the path formulation that allows demands to be split across multiple paths is presented. The decision variables are (1) for each demand k in K, the amount of flow to assign to each path p in P_k, the set of all possible paths for demand k, (2) whether or not capacity is allocated on stack s in S_r of ring r in R, and (3) whether or not to place an ADM at node n in N_r on stack s in S_r of ring r in R.

Sets

K	Set of demands
R	Set of rings
S_r	Set of stacks on ring r, $\forall r \in R$
N_r	Set of nodes on ring r, $\forall r \in R$
A_r	Set of arcs on ring r, $\forall r \in R$
P_k	Set of paths for demand k, $\forall k \in K$

Parameters

Q	Capacity per stack in OC-1 units (192 to represent OC-192)
d_k	Quantity of demand k, $\forall k \in K$
δ_{kprsm}	A value of 1 indicates path p for demand k uses arc m on stack s of ring r, $\forall p \in P_k$, $\forall k \in K$, $\forall m \in A_r$, $\forall s \in S_r$, $\forall r \in R$; 0 otherwise
γ_{kprsn}	A value of 1 indicates path p for demand k requires an ADM at node n on stack s of ring r, $\forall p \in P_k$, $\forall k \in K$, $\forall n \in N_r$, $\forall s \in S_r$, $\forall r \in R$; 0 otherwise

Variables

y_{kp}	Flow on path p for demand k, $\forall p \in P_k$, $\forall k \in K$
c_{rs}	1 if capacity is allocated on stack s of ring r, $\forall s \in S_r$, $\forall r \in R$; 0 otherwise
a_{rsn}	1 if an ADM is needed at node n on stack s of ring r, $\forall n \in N_r$, $\forall s \in S_r$, $\forall r \in R$; 0 otherwise

Objective

$$\text{Minimize } \sum_r \sum_s \sum_n a_{rsn} \quad (1)$$

Constraints

$$\sum_p y_{kp} = d_k \qquad \forall k \in K \quad (2)$$

$$\sum_p \sum_k \delta_{kprsm}\, y_{kp} \leq Q\, c_{rs} \qquad \forall m \in A_r,\ \forall s \in S_r,\ \forall r \in R \quad (3)$$

$$\sum_p \sum_k \gamma_{kprsn}\, y_{kp} \leq Q\, a_{rsn} \qquad \forall n \in N_r,\ \forall s \in S_r,\ \forall r \in R \quad (4)$$

$$y_{kp} \in Z_+ \qquad \forall p \in P_k,\ \forall k \in K \quad (5)$$

$$c_{rs} = \{0,1\} \qquad \forall s \in S_r,\ \forall r \in R \quad (6)$$

$$a_{rsn} = \{0, 1\} \qquad \forall n \in N_r,\ \forall s \in S_r,\ \forall r \in R \quad (7)$$

The objective function (1) minimizes the total number of ADMs that are used in the network. The first set of constraints (2) routes all the demand, stating that the sum of the flow quantities on all the different paths for a certain demand must equal the demand quantity for that demand. The second set of constraints (3) are the capacity constraints, stating that the sum of all of the flows on all of the paths that use a link cannot exceed the capacity on that ring. (Note that the ring capacity is determined by the link with the maximum total flow.) The third set of constraints (4) are the ADM placement constraints, stating that ADMs must be placed where demands originate and terminate and where traffic changes rings. Finally, the y_{kp} variables and the c_{rs} variables must be non-negative integer values and the a_{rsn} variables must be binary.

3.2. Path-Based Formulation (does not allow demand splitting)

In this section, the path formulation that requires demand to be assigned to exactly one path is presented. This formulation is very similar to the formulation presented in

Section 3.1. The sets, parameters, the ADM and capacity variables and the objective function remain the same. The differences come in the definition of the path (flow) variables and the network constraints involving the path variables. The decision variables are binary: for each demand k, a variable y_{kp} takes on the value 1 if path p in P_k, the set of all possible paths for demand k, is selected and 0 otherwise. The formulation is shown below.

Objective

$$\text{Minimize } \sum_r \sum_s \sum_n a_{rsn} \qquad (1)$$

Constraints

$$\sum_p y_{kp} = 1 \qquad \forall k \in K \qquad (8)$$

$$\sum_p \sum_k \delta_{kprsm}\, y_{kp}\, d_k \leq Q\, c_{rs} \qquad \forall m \in A_r,\ \forall s \in S_r,\ \forall r \in R \qquad (9)$$

$$\sum_p \sum_k \gamma_{kprsn}\, y_{kp}\, d_k \leq 2\, Q\, a_{rsn} \qquad \forall n \in N_r,\ \forall s \in S_r,\ \forall r \in R \qquad (10)$$

$$y_{kp} = \{0, 1\} \qquad \forall p \in P_k,\ \forall k \in K \qquad (11)$$

Notice that with the new variables, the path variables must be multiplied by the demand quantity d_k to get the flow amount on a path. Otherwise, the purpose and functionality of each of the constraints remains the same.

This path formulation may provide a solution that requires fewer ADMs than the no-splitting formulation. Allowing demands to be split can be more capacity efficient, but for some applications, it may not be appropriate.

3. 3. Edge-Based Formulation (allows demand splitting)

In the edge-based formulation that allows demands to be split across multiple paths, the decision variables are the amount of flow for demand k in K on arc (i, j) in A, whether or not capacity is allotted on stack s in S_r of ring r in R, and whether or not to place an ADM at node n in N.

There is a difference in the definition of the d_{kn} parameter, when compared to the path-based model. In this model for every path and node, if node n is the origin of demand k the value of d_{kn} is the quantity of demand k, if node n is the destination of demand k the value of d_{kn} is negative of the quantity of demand k, otherwise, the value of d_{kn} is zero.

Another difference in the edge-based model is the use of the special set, which is the set of super nodes. A super node is a dummy node that is introduced into the network. For each node in the network there is one super node that is connected to that node on all stack levels. So for every node n, an extra pair of arcs are introduced, one arc from n to its super node and one arc from the super node to n. The super node is assigned the d_{kn} values and routes the demands to the proper stack allowing the quantity to be split over the stacks. The edge-based formulation is modeled as shown below.

Sets

K	Set of demands
R	Set of rings
S_r	Set of stacks $\forall r \in R$
N	Set of nodes
special	Set of super nodes
$clockwise_{rs}$	Set of clockwise arcs, $\forall s \in S_r$, $\forall r \in R$
$counterccc_{rs}$	Set of counterclockwise arcs, $\forall s \in S_r$, $\forall r \in R$
connect	Set of arcs that connect each node to its corresponding super node

$ring\_arcs = clockwise_{rs} \cup counterccc_{rs}$

$all\_arcs = ring\_arcs \cup connect$

Parameters

Q	capacity per channel (OC-192)
d_{kn}	Quantity of demand k at node n, $\forall k \in K$, $\forall n \in N$

Variables

f_{kij}	Flow on arc (i,j) for demand k, $\forall (i,j) \in A$, $\forall k \in K$
c_{rs}	Capacity for stack s on ring r, $\forall s \in S_r$, $\forall r \in R$
a_n	Indicates if an ADM is needed at node n, $\forall n \in N$

Objective

Minimize $\sum_r \sum_s \sum_n a_{rsn}$

Constraints

$\sum_{(i,j) \in all\_arcs} f_{kij} - \sum_{(j,i) \in Hn} f_{kji} = d_{kn}$ $\quad \forall k \in K, \forall n \in N$

$\sum_{k \in K} (f_{kij} + f_{kji}) \leq Q * c_{rs}$ $\quad \forall s \in S_r, \forall r \in R, \forall (i,j) \in clockwise_{rs}$

$\sum_{k \in K} \sum_{(i,n) \in all\_arcs:\, i \in special} f_{kin} \leq a_n * 2 * Q$ $\quad \forall n \in N$

$\sum_{k \in K} \sum_{(n,j) \in all\_arcs:\, j \in special} f_{knj} \leq a_n * 2 * Q$ $\quad \forall n \in N$

$c_{rs} \in Z_+$ $\quad \forall s \in S, \forall r \in R$

$f_{kij} \in Z_+$ $\quad \forall (i,j) \in A, \forall k \in K$

$a_n = \{0, 1\} \qquad \forall n \in N_r$

The objective function minimizes the total number of ADMs that are used in the network. The first set of constraints are flow conservation constraints. The second set of constraints are the capacity constraints. The third and fourth set of constraints places ADMs at the necessary nodes. Finally, the c_{rs} variables and the f_{kij} variables must be non-negative integer values and the a_{rsn} variables must be binary.

3.4. Example

In this section, a small example is shown to illustrate some of the features of the formulations.

Consider the 10 node, 2 ring network in Figure 1, with 4 demands: $d_{02} = 96$, $d_{18} = 108$, $d_{58} = 96$ and $d_{79} = 96$. The maximum capacity for each stack on each ring is 192.

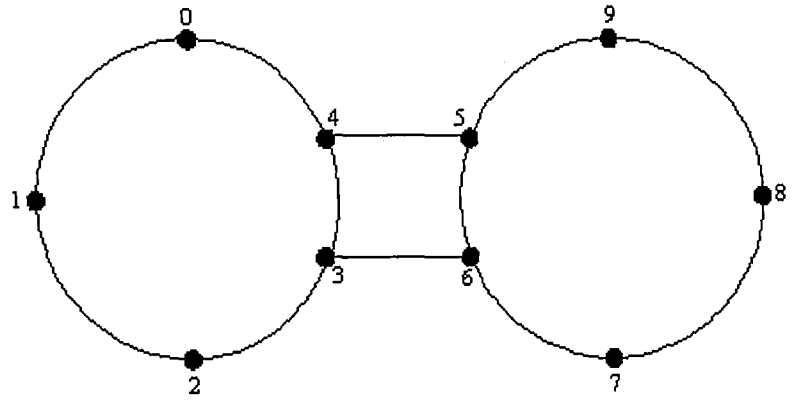

Figure 1 – 10 node, 2 ring network

Figures 2, 3 and 4 show the ADM placements that were determined using the path-based formulation that allows splitting, the path-based formulation that does not allow splitting and the edge-based formulation, respectively. The black dots on the rings

signify the placement of an ADM at that location, whereas a white dot signifies that an ADM is not placed at that location. Also, in the case of the no-splitting path-based formulation (Figure 3), two stacks of capacity are required on ring 1 (the left ring). The outer circles represent stack 1 on each ring and the inner circles represent stack 2 on each ring.

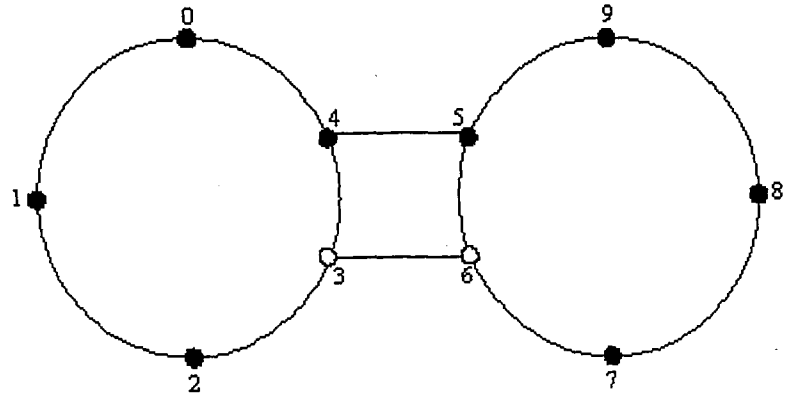

Figure 2 – Path-based solution when splitting is allowed

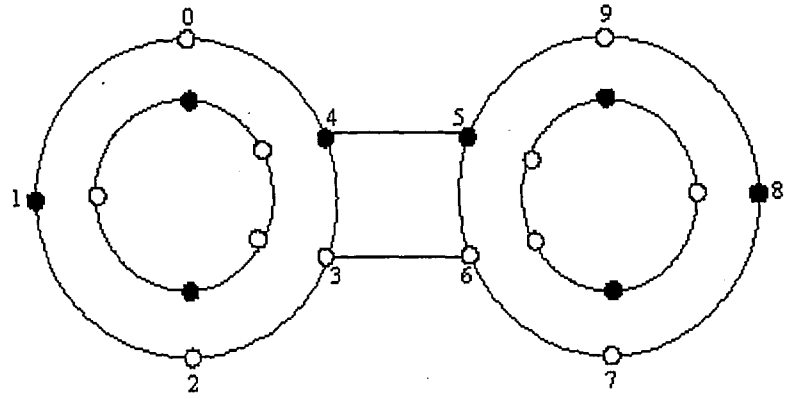

Figure 3 – Path-based solution when splitting is not allowed

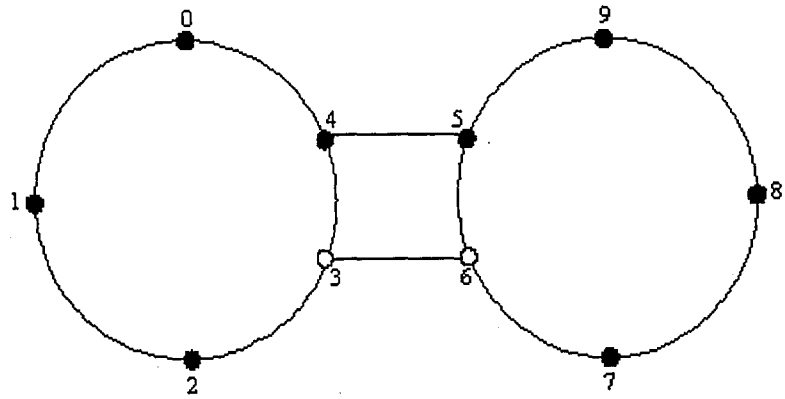

Figure 4 – Edge-based solution

With the objective function of minimizing the number of ADMs required, all three formulations give the same answer. The placement of the ADMs and the paths taken by the demands differ slightly. Although this example solves both easily and quickly, it does not take long before the problems become difficult. The difficulty stems from the fact that the formulations are integer programs. When solving integer programs with branch-and-bound, it is important to have a strong linear programming relaxation bound.

Consider the path-based formulation that allows splitting. The variables are y_{kp}, a_{rsn} and c_{rs}, and both a_{rsn} and c_{rs} must be binary (0,1). In the LP relaxation solution, for the example here, the y_{kp} and the c_{rs} variables all are integer, but the a_{rsn} variables were fractional. While this finding was promising because it showed that fewer variables than initially thought need to be changed from real numbers to integers, it was also found that the LP relaxation bound is weak. The LP relaxation solution assigns a_{rsn} values to be very small decimal values, only large enough to satisfy the constraints. The a_{rsn} variables are set in the constraints

$$\sum_p \sum_k \gamma_{kprsn}\, y_{kp} \leq 2\, Q\, a_{rsn} \qquad \forall n \in N_r,\ \forall s \in S_r,\ \forall r \in R$$

For the example, the LP relaxation solution has the a_{rsn} values shown in Table 1.

	ring	stack	node			ring	stack	node	
a	0	0	0	90/384 = 0.234375	a	1	0	0	102/384 = 0.265625
a	0	0	1	108/384 = 0.28125	a	1	0	1	102/384 = 0.265625
a	0	0	2	90/384 = 0.234375	a	1	0	2	90/384 = 0.234375
a	0	0	3	102/384 = 0.265625	a	1	0	3	204/384 = 0.53125
a	0	0	4	6/384 = 0.0156	a	1	0	4	90/384 = 0.234375
a	0	1	0	6/384 = 0.0156	a	1	1	0	0
a	0	1	1	0	a	1	1	1	0
a	0	1	2	6/384 = 0.0156	a	1	1	2	6/384 = 0.0156
a	0	1	3	0	a	1	1	3	0
a	0	1	4	0	a	1	1	4	6/384 = 0.0156

Table 1 – ADM variable values from the LP relaxation solution

In order to force these initial a_{rsn} values to be closer to 1, the following constraints were added to the formulation.

$$\sum_p \gamma_{kprsn}\, y_{kp} \leq d_k * a_{rsn} \qquad \forall n \in N_r, \forall s \in S_r, \forall r \in R, \forall k \in K$$

These constraints are valid for the integer program and strengthen the LP relaxation bound. For this example, the original LP relaxation bound was 2.63, and the strengthened LP relaxation bound was 8, which is the optimal integer objective function value.

3.5. Comparison of the Formulations

These models all will minimize the equipment and capacity requirements as a result of minimizing the total number of ADMs. Since the length of the path that a demand follows is not considered, some demands will be routed the "long way" around rings as opposed to a shortest path approach. This will result in a more balanced loading of the rings.

There are a number of differences in the formulations. In the path-based formulation, one input is a set P_k, which represents all of the possible paths from the origin to the destination of demand k. In the edge-base formulation, the paths are determined as a result of the flow conservation constraints, which prevent demands from starting and terminating at the wrong nodes. There are benefits to both of these methods.

From a realistic standpoint, having all the paths clearly defined (as in the path formulations) makes it easier to see which route a demand is traversing. Also, the paths are well-defined to require that demand not be able to change between stacks within a single ring. The stack level can change, however, when a demand is changing rings. Since special wavelength translation hardware is needed to translate the wavelength in order for it to switch stacks, it makes sense that allowing the demand to jump from stack to stack would be costly and inefficient. Because of the way that the edge-based model is implemented, it is possible for a demand to switch stacks within a ring by looping through the supernode associated with a node. It does not appear that this was an issue in the computational experiments.

From a modeling standpoint, the main problem with the path-based method is the explosion of data that results when all the possible paths are generated. Because any path that is generated has to be replicated for all of the stacks, the number of variables grows rapidly. This problem is avoided when an edge-based formulation is used. Two remedies are possible. First, if a more sophisticated method of choosing allowable paths were used, rather than just generating all of them and adding them to

the model, it may be possible to solve larger problems. However, when all of the path variables are added to the formulation, only small problems can be solved in a reasonable amount of time.

A good means of comparing the two formulations is by considering the number of constraints and variables in each formulation. The edge-based formulation has nK+SRclockwise+2n constraints and K+SR+n variables and the path-based formulation has K+mSR+nSR constraints and SR+PK+NSR variables, where n is the number of nodes, K is the number of demands, S is the number of stacks, R is the number of rings, clockwise is the number of clockwise arcs, m is the total number of arcs and P is the number of paths.

4. Test Data Generation

Test data was randomly generated in a 100 by 100 square. X and Y coordinates were generated within the square for every node in the network. The nodes were then connected to form rings. Every ring was connected to at least one other ring by having two common nodes. An example is show in Figure 5.

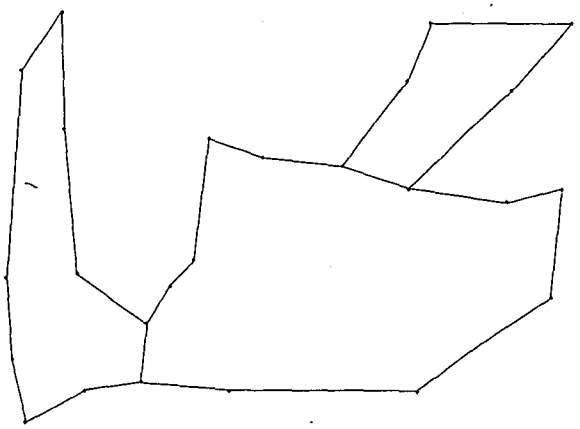

Figure 5 – An Example of a 25 Node Network.

Demand pairs were also randomly generated. The origin and destination were randomly generated to range between 0 and N-1. The quantity of demand for that pair was generated to be between 3 and 24. Any demand pair that had the same node for the origin and destination were thrown out. Also any demand pair that had already been defined was thrown out to avoid duplicate quantities between a demand pair.

Paths were generated for the path-based model using a program written in C++. The program reads in three files, one listing the nodes and which ring they were located on, one listing the set of arcs and one listing the origin, destination and quantity for each demand. The program then takes each demand one at a time and using a depth-first search algorithm, determines all of the possible paths from origin to destination. Each path is subjected to two rules in order to help reduce the number of possible paths. The first rule is that if the origin and destination of the demand are on the same ring, the path used can not leave that ring. The second rule is that if a path

needs to traverse multiple rings, it can not return to a ring that it has already traveled on. These two rules combined eliminate paths that cycle around rings and also eliminate paths that are unnecessarily long. After a path is found that does not violate either of these two rules, the list of nodes that comprise the path is written out to a data file.

5. Results

Each of the three formulations was solved for 24 different problem instances, for a total of 72 instances. Half of the data sets were comprised of networks that contained 15 nodes laid out over 2 rings and the other half were networks with 25 nodes laid out over 3 rings. Three different 15 node examples and three different 25 node examples were generated as described in Section 4. Demand origins, destinations and quantities also were randomly generated. For each of the 15 node networks, a set of 50 demands and a set of 100 demands were generated. For each of the 25 node networks, a set of 25 demands and a set of 50 demands were generated. A second set of demands were created for each test problem by taking each set of demand, keeping the same origin and destination pairs, but multiplying the quantity of the demand by 2. This makes it easy to see what happens when the only change in the test data is the volume of quantity trying to be routed.

For each of the instances, the optimal objective function value, the value of the LP Relaxation bound, the CPU time and the number of branch and bound nodes were

compiled into a table and can be seen in Table 2. Across the three formulatons, the final objective function values are close, usually within one or two ADMs. There are a couple reasons for the differenced. First, as explained in Section 3.3, there is a slight difference in the way in which the edge-based model and path-based models represent the network. In the path-based models, each interconnection node is represented by two nodes, one node on each of the rings that it joins. In the edge-based model, each interconnection node is modeled as just one node that serves both rings. This difference in modeling most likely causes the edge-based model to return a better optimal objective function value in some of the tests. Second, some of the objective function values are not provably optimal, because a time limit was used in running the models. Therefore, if a test was terminated prematurely, it may not have yet reached the true optimal.

All of the test problems were run with a time limit of 3600 seconds (1 hour) for the edge-based and 7200 seconds (2 hours) for the path-based in order to prevent test problems from running for an extensive period of time. It quickly became apparent that the edge-based model was running much slower than both the path-based models, especially the path-based model that allows demand splitting. Notice that not one of the edge-based instances completed within one hour, whereas some of the same splitting path-based instances completed in less than one second. Based on the CPU time required to find an optimal solution, it appears as though the path-based model that allows for demand splitting is a good model to be use for this design problem, because it runs the quickly and provides a comparable objective function value.

The path-based models could be made even more efficient through the use of column generation. Column generation is a technique that can be used to solve linear programs in which there are a large number of variables. In column generation, a subset of paths are initially included in the model. Then, using the associated dual variable information, a separate optimization problem is solved to determine if a better solution could be attained by adding additional paths to the model. Implementing a column generation scheme would reduce the amount of data required and as a result (most likely) lead to shorter CPU times.

Table 3 shows the difference in the number of paths that were selected in the two versions of the path-based model. Allowing demands to be split increases the number of paths in the solution but results sometimes in fewer ADMs.

The path-based formulations include the set of constraints to improve the LP relaxation bound. As a result, on average, the LP relaxation bound as a percentage of the integer solution is 87.06% for the splitting path-based and 88.26% for the no-splitting path-based formulation. In contrast, the LP relaxation bound as a percentage of the integer solution is 19.47%.

			Optimal Objective Function*			LP Relaxation Bound			CPU Time			# of Branch-and-Bound Nodes		
Nodes	Ex.	Demand	Edge	Path1**	Path2	Edge	Path1	Path2	Edge	Path1	Path2	Edge	Path1	Path2
15	1	50	16	17	17	2.7	17	17	3600	0.03	0.04	13615	0	0
15	1	50x2	19	23	23	14.8	17	17	3600	39.53	7201.32	4362	5925	1710106
15	1	75	24	21	21	4.3	17	17	3600	289.88	3599.6	2803	6684	85894
15	1	75x2	20	34'	34	4.3	19	19	3600	562.64	7200.63	638	2592	24104
15	2	50	15	17	17	2.3	17	17	3600	0.03	0.05	377	0	0
15	2	50x2	18	21	21	4.7	17	17	3600	82.44	332.59	4872	9405	58287
15	2	75	23	21'	21	3.9	17	17	3600	664.66	7200.69	3414	21100	409178
15	2	75x2	32	29'	29	7.7	19	19	3600	336.89	7200.37	1244	5600	65229
15	3	50	16	17	18	2.9	17	17	3600	0.04	53.03	1101	0	24232
15	3	50x2	20	22'	23	5.8	17	17	3600	120.09	7201.26	832	29907	1709108
15	3	75	25	23'	23	4.1	17	17	3600	120.26	7200.47	1585	2537	160493
15	3	75x2	39	44	xx	8.2	20	xx	3600	120.5	xx	289	291	xx
25	1	25	24	26	26	2.1	26	26	3600	0.04	0.03	28548	0	0
25	1	25x2	25	26	26	4.2	26	26	3600	0.09	0.59	16852	7	0
25	1	50	29	29	29	3.2	28	28	3600	51.43	115.23	2362	3266	5807
25	1	50x2	46	32'	32	6.4	28	28	3600	725.28	7201.59	1205	2000	15572
25	2	25	26	26	26	1.8	26	26	3600	0.03	0.06	27666	0	0
25	2	25x2	26	26	26	3.7	26	26	3600	0.04	0.48	24811	0	0
25	2	50	30	29	29	3.8	29	29	3600	1.85	4.4	4752	2	15
25	2	50x2	34	33'	35	7.7	29	29	3600	2248.23	7201.65	3010	7300	28101
25	3	25	23	23	23	2.2	23	23	3600	0.04	0.06	26912	0	0
25	3	25x2	25	25	25	4.4	23	23	3600	45.08	247.53	15451	3450	31154
25	3	50	30	29	29	3.6	29	29	3600	3.66	3.54	2221	3	1
25	3	50x2	36	32	32	7.2	29	29	3600	1223.33	7202.32	1448	66832	14352

* Test problems were set to terminate after running for either 3600 or 7200 seconds. Therefore any problems with a CPU time equal to either 3600 or 7200 may have stopped before it reached an optimal objective function. Therefore, the values are an upper bound on the optimal value for the problem.

** Path1 is the path-formulation that allows splitting and Path2 is the path-formulation that does not allow splitting.

' The problem was terminated after the best integer value was found to be less than or equal to the optimal value for the Path2.

xx Since the no-splitting path-based model requires more stacks to be able to find a feasible solution, more stacks were added to the data file. However, this made this problem too large to be solved.

Table 2 - Results

			Number of Paths	
Nodes	Ex.	Demand	Path1*	Path2
15	1	50	52	50
15	1	50x2	73	50
15	1	75	79	75
15	1	75x2	270	75
15	2	50	50	50
15	2	50x2	62	50
15	2	75	113	75
15	2	75x2	120	75
15	3	50	52	50
15	3	50x2	58	50
15	3	75	84	75
15	3	75x2	101	75
25	1	25	25	25
25	1	25x2	27	25
25	1	50	51	50
25	1	50x2	73	50
25	2	25	25	25
25	2	25x2	26	25
25	2	50	52	50
25	2	50x2	84	50
25	3	25	25	25
25	3	25x2	33	25
25	3	50	53	50
25	3	50x2	59	50

* Path1 is the path-formulation that allows splitting and Path2 is the path-formulation that does not allow splitting.

Table 3 – Comparison of the Number of Paths Used

6. Conclusions

Both formulation methods, when given the same set of demand pairs and the same network topology, accurately routed demands in order to minimize the total number of ADMs used in the network. Both took a large amount of CPU time; however the edge-based model was substantially slower. Since the functionality of the interconnection nodes was modeled differently in the two methods, the same optimal routing is not always reached.

Other than looking purely at data results, there are other reasons why the path-based model would be a better way of modeling a network. First of all, a path-based model is easier to conceptualize. You can see what possible routes the demands can take. This makes modeling the problem easier. Also, when working with engineers who are not knowledgeable about different modeling techniques, they may not understand the concept of flow conservation constraints and how they are utilized in the edge-based model, making it different to understand how the model is actually routing demand. Finally, the use of a path-based model allows for more control over how demands get routed through the system. For example, suppose a customer has new demand that they wish to route through a network. Perhaps they already have demand in that same network and so in a attempt to diversify, ask that their new demand not be routed on the same path as their other demand. This request can not be granted with certainty when using the edge-based model since there is no control over how the demands get routed. However, in the path-based model, the request is

easily met by simply excluding the current path from the list of possible paths that the new route can traverse. All these facts together show that when comparing path-based models and edge-based models, the path-based model is the better method to use for the routing of demand through a multi-ring network.

In the future, more work needs to be done involving path generation and the selection of possible paths. For this problem, all paths were generated, duplicated across all stacks and included in the modeling. This made solving even a 25 node problem very difficult and time consuming. A better way of selectively deciding what paths should be allowable, possibly using column generation, is a practical next step.

7. References

[1] Cosares, S., D. N. Deutsch, I. Saniee and O. J. Wasem. 1995. "SONET Toolkit: A Decision Support System for Designing Robust and Cost-Effective Fiber-Optic Networks," *Interfaces*, Vol. 25, No. 1 (January-February), pp. 20-40.

[2] Vachani, R., A. Shulman and P. Kubat. 1996. "Multicommodity Flows in Ring Networks," *INFORMS Journal on Computing*, Vol. 8, No. 3 (summer), pp. 235-242.

[3] Schrijver, A., P. Seymour and P. Winkler. 1998. "The Ring Loading Problem," *SIAM J. Discrete Math.*, Vol. 11, No. 1, (February), pp. 1-14.

[4] Klincewicz, J. G., H. Luss and D. C. K. Yan. 1998. "Designing Tributary Networks with Multiple Ring Families," *Computers Ops Res.*, Vol. 25, No. 12, pp. 1145-1157.

[5] Chow, T. Y. and P. J. Lin, "The Ring Grooming Problem," *Discrete Applied Math.*, to appear.

[6] Sutter, A., F. Vanderbeck and L. Wolsey. 1998. "Optimal Placement of Add/Drop Multiplexers: Heuristic and Exact Algorithms," *Operations Research*, Vol. 46, No.5 (September-October), pp. 719-728.

[7] Gawande, M. and J. G. Klincewicz. 1999. "Designing a family of bi-directional self-healing rings," Part of the SPIE Conference on Performance and Control of Network Systems III, Boston, Massachusetts.

[8] Berger, R. T. 2001. Models and Algorithms for the Design of Hierarchical Ring Networks. Technical Report. Level 3 Communications, Broomfield, CO.

8. Brief Biography

Karen A. Kelly was born of William J. and Linda A. Kelly on April 18, 1980 in Bristol, Pennsylvania. She attended Lehigh University as an undergraduate, receiving a B.S. in Industrial Engineering in June 2002. She remained at Lehigh as a graduate student as part of the President's Scholar Program. She is a member of the Tau Beta Pi Engineering Honor Society and the Alpha Pi Mu Industrial Engineering Honor Society. Following completion of her M.S. degree in Information and Systems Engineering, she will be going to work for marketRX in Bridgewater, NJ as a statistical analyst.

END OF TITLE