INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Enhanced Model Representations for an Intra-Ring Synchronous Optical Network Design Problem Allowing Demand Splitting

Hanif D. Sherali, J. Cole Smith, Youngho Lee,

To cite this article:
Hanif D. Sherali, J. Cole Smith, Youngho Lee, (2000) Enhanced Model Representations for an Intra-Ring Synchronous Optical Network Design Problem Allowing Demand Splitting. INFORMS Journal on Computing 12(4):284-298. http://dx.doi.org/10.1287/ijoc.12.4.284.11884

INFORMS Journal on Computing
Vol. 12, No. 4, Fall 2000

0899-1499/100/1204-0284 $05.00

Enhanced Model Representations for an Intra-Ring Synchronous Optical Network Design Problem Allowing Demand Splitting

HANIF D. SHERALI / *Department of Industrial and Systems Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, Email: hanifs@vt.edu*

J. COLE SMITH / *Department of Industrial and Systems Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, Email: josmit13@vt.edu*

YOUNGHO LEE / *Department of Industrial Engineering, Korea University, Sung-Buk Ku Anam Dong 5-1, Seoul 136-701, Korea, Email: yhlee@kuccnx.korea.ac.kr*

(Received: July 1999; revised: January 2000, April 2000; accepted: May 2000)

In this paper, we consider a network design problem arising in the context of deploying synchronous optical networks (SONET) using a unidirectional path switched ring architecture, a standard of transmission using optical fiber technology. Given several rings of this type, the problem is to find an assignment of nodes to possibly multiple rings, and to determine what portion of demand traffic between node pairs spanned by each ring should be allocated to that ring. The constraints require that the demand traffic between each node pair should be satisfiable given the ring capacities, and that no more than a specified maximum number of nodes should be assigned to each ring. The objective function is to minimize the total number of node-to-ring assignments, and hence, the capital investment in add-drop multiplexer equipments. We formulate the problem as a mixed-integer programming model, and propose several alternative modeling techniques designed to improve the mathematical representation of this problem. We then develop various classes of valid inequalities for the problem along with suitable separation procedures for tightening the representation of the model, and accordingly, prescribe an algorithmic approach that coordinates tailored routines with a commercial solver (CPLEX). We also propose a heuristic procedure which enhances the solvability of the problem and provides bounds within 5–13% of the optimal solution. Promising computational results are presented that exhibit the viability of the overall approach and that lend insights into various modeling and algorithmic constructs.

In this paper, we consider a network design problem arising from the deployment of synchronous optical networks (SONET). The SONET is a standard of transmission technology used in optical fiber networks. The typical capacity of current technology permits the transmission of 2.4 Gbps over a single fiber, which is equivalent to over 38,000 voice circuits (see Wu 1992 for technical details). Thus, a failure in even a single link may result in a tremendous loss in customer service. In response, telecommunication companies are adopting SONET ring architectures, where the structure of the ring promotes an enhanced survivability of the network. The problem that arises in this context is to determine cost-effective topological designs and deployment strategies for the SONET rings.

This paper addresses such a SONET ring design problem faced by network planners at any regional operating company that incorporates this technology in its territory. In order to route the traffic between the clients or nodes assigned to a ring in a SONET network configuration, we need a special type of equipment that is capable of adding and dropping the traffic to be installed at each node. This device is called a SONET *add-drop multiplexer* (ADM). Since the cost of SONET ADMs has dropped quite significantly over the past few years, a fully ADM-based network having no digital cross-connect systems (DCS) has become quite an attractive alternative architecture for providing link survivable topologies, especially for interoffice facility networks in larger metropolitan areas. In particular, the demand for high capacity transport has grown very rapidly, especially for a metropolitan region within a local access transport area. The main concept that characterizes the routing of traffic using this SONET technology is that the high capacity interchange among demand nodes can be grouped into a few clusters that can then be transported more efficiently and economically using a direct connection in a ring, instead of using the traditional hierarchical routing scheme inherent in a DCS-based network. Note that due to the capacity limit of SONET equipments such as an OC-48 ring (which, as defined and explained in greater detail below, allows only 48 channels in a unidirectional configuration and 24 channels in a bidirectional configuration with two fibers), we need to partition the traffic demand into several clusters of nodes. The problem considered here then becomes one of finding an optimal partition of the traffic demand in the network, while minimizing the total ADM cost. In our context, we consider only unidirectional rings having a fixed capacity.

To describe the problem more precisely, consider a set N

Subject classifications: Philosophy of modeling, integer programming.
Other key words: SONET ring, valid inequalities

of nodes in the network, indexed by $i \in N \equiv \{1, \ldots, n\}$, where $n \geq 2$. Let d_{ij} be the traffic demand (number of required channels to carry the traffic) between node $i \in N$ and node $j\ (> i) \in N$. Accordingly, define an edge set $E = \{(i, j)\colon i < j,\ d_{ij} > 0\}$ comprised of such demand pairs. Furthermore, let $m \geq 2$ be the number of rings we are permitted to install, and let b be the capacity of each ring, measured by the number of available (unidirectional) channels (some further elucidation on this parameter is provided below). Also, let $R \geq 2$ be the maximum number of ADMs (and hence nodes) allowed to be installed on a single ring. Then, the ring design problem of concern is to assign the nodes to rings, and to route the traffic between the nodes, so as to minimize the total number of ADMs, while satisfying the ring capacity restrictions as well as the demands for all pairs of nodes in the edge set E, possibly splitting demand over several rings. Note that inter-ring traffic is not permissible.

The ring architecture considered in this paper is a *unidirectional path switched ring* (UPSR) of the type OC-24 or OC-48 having respectively a maximum capacity of 24 or 48 channels. Each channel is of type STS1 (*synchronous transport signal*), and has a bit transmission rate of 51.84 Mega-bits per second (equivalent to 810 voice circuits). In this unidirectional ring architecture, the ADMs equipped at each node are connected by a pair of optical fibers, one for the *working path* and the other for *protection*. In the case of OC-48 rings, for example, each of these fibers has a capacity of 48 channels and the same (integral) number of channels on each fiber are dedicated to serve the demand signal traffic between each pair of nodes assigned to the ring. Hence, in effect, the capacity b is allocated or distributed in a dedicated fashion among the pairs of nodes on the ring. In practice, when the model is used for planning purposes, due to the stochasticity in demand as well as the uncertainty in future growth in demand over the typical horizon of five years, the value of b is assumed to be 30–80% of the maximum capacity, depending on the uncertainty and the risk attitude of the carrier. Hence, in our computations, the value of b varies in the range [15, 40] over the test cases.

Whenever a signal transmission originates between any pair of nodes assigned to the ring, the working path propagates this as a *primary* signal in the clockwise direction, and the protection fiber carries an *identical secondary* signal in the counterclockwise direction. Consequently, at the receiving node, two identical signals are observed. During normal operation, only the primary signal is used, although both signals are monitored for alarms and maintenance signals, with a switch occurring in case of interruptions or a loss in data being detected along the working path. Of course, if a hardware failure or a fiber cut occurs in the working path, service is sustained by switching to select the secondary signal being propagated along the protection fiber. Hence, this pair of fibers improves both the reliability and the survivability of the network. Note also that whenever a link fails in either fiber, by monitoring the loss in the clockwise or counterclockwise transmission between pairs of nodes, the failed link can be isolated. We also comment here that in the case of *bidirectional line switched rings* (BLSR), the ring architecture might be composed of four fibers, or even of only two fibers at half the maximum capacity, and moreover, the channel allocation among node pairs occurs on a link-by-link basis in either the clockwise or counterclockwise direction. Consequently, the modeling of capacity consumption is quite different in this case (see Dell'Amico et al. 1999, for example), although for conservative network planning purposes, even when bidirectional rings are deployed, the capacity concept of UPSR rings is frequently employed.

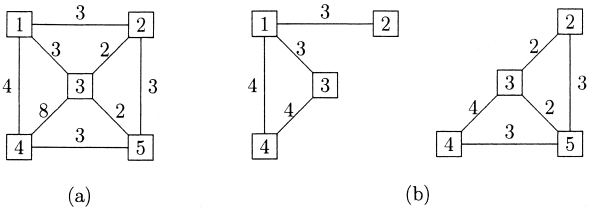

Figure 1. An example of the problem ($b = 14$, $R = 4$).

For example, consider Figure 1 that illustrates the nature of the problem for an instance having $n = 5$ nodes and $m = 2$ rings, with capacity parameters $b = 14$ and $R = 4$. Part (a) of Figure 1 depicts the demand pattern, and part (b) displays a feasible solution of two clusters that require a total of eight ADMs. The numbers against the links represent the allocated traffic demands. Note that the demand between nodes 3 and 4 is split between the two rings in this example. Observe also that the solution to this problem essentially prescribes clusters of (possibly overlapping) sets of nodes along with an allocation of demand traffic between pairs of nodes that are assigned to each ring. The solution does not prescribe an actual configuration by which the nodes in each cluster should be connected in order to form a ring structure. This latter problem, referred to as a *physical ring design problem*, is itself a complex optimization problem that considers existing fiber connections and variable costs of new fiber connections along with fixed costs of inserting suitable non-ADM nodes for the purpose of relaying traffic, and is typically solved subsequent to determining the foregoing clusters of nodes to be assigned to each ring (Lee et al. 1999a, 1999b).

Wu and Burrowes (1990) and Wu (1992) describe in detail the advantages of SONET ring architectures over the traditional hubbing network in terms of cost and survivability. Wasem et al. (1994) have developed SONET design software considering demand growth over time. They consider two types of network architectures, a self-healing ring (SHR) structure and a hubbing network having point-to-point diverse protection systems in order to incorporate SHR into an existing hubbing network. Cosares and Saniee (1994) present a SONET ring design model and propose heuristic procedures for minimizing the traffic load on each ring. Myung et al. (1997) have developed a polynomial-time procedure for minimizing the traffic load on each ring, where this traffic may be split among the rings for each node pair. For the case of non-split traffic among rings, Karunanithi and Carpenter (1997) describe a genetic algorithm to minimize traffic loads. In contrast, we consider in this paper the problem of designing a SONET ring architecture that minimizes the number of

ADMs to be installed, subject to ring load and capacity constraints for a single period, and present a new solution procedure for finding an optimal solution.

Laguna (1994) has proposed a mixed-integer programming (MIP) model of a logical SONET clustering problem that permits inter-ring traffic, and has obtained promising computational results using a tabu search algorithm. Goldschmidt et al. (1998) also consider the problem that allows inter-ring traffic and develop a model to minimize the total number of rings used. (We show later that in our context, this is not equivalent to minimizing the number of ADMs placed on the rings.) Since inter-ring traffic is prohibited in our problem, we call our problem an *intra-ring design problem*. For the non-split case of our problem, Lee et al. (1999a, 1999b) discuss a branch-and-cut algorithm, and Sutter et al. (1998) utilize a column generation approach to solve this problem. Note that this non-split case facilitates the routing and management of traffic and the administration of the network, but being more restrictive than the case that permits demand splitting, it could result in more capital intensive solutions. Moreover, demand splitting provides greater flexibility in dealing with peak demand surges as well as with future growth in the demand pattern. For this reason, most real-world implementations of the SONET ring technology accommodate the splitting of demand. In our computational experience on both these classes of problems (see Section 6), we found that the non-split case problems are relatively harder to solve, and are more likely to result in infeasible or somewhat more expensive solutions than the split-demand case considered herein.

The remainder of this paper is organized as follows. In Section 1 we model the problem as a mixed-integer program and evaluate several alternative formulations for this problem. We provide additional insights into the structure of this problem in Section 2 by establishing the NP-hardness of the problem, even for specific classes of demand graphs. Next, we present certain algorithmic details for the problem that pertain to preprocessing routines and branching priorities in Section 3, and in Section 4, we describe several classes of valid inequalities along with suitable separation procedures for further tightening the representation of the problem. A heuristic procedure for facilitating the branch-and-bound process and providing upper bounds to the problem is developed in Section 5. Finally, Section 6 provides a summary of our computational results, and Section 7 concludes the paper.

1. Problem Formulation

To model the split-demand SONET ring design optimization problem, we define a primary set of decision variables x_{ik} $\forall i \in N$, $k \in M \equiv \{1, \ldots, m\}$, where $x_{ik} = 1$ if node i is assigned to ring k, and 0 otherwise. We also define a secondary set of decision variables f_{ijk} $\forall (i, j) \in E$ $(i < j)$ and $k \in M$, representing the fraction of demand between the node pair (i, j) that is assigned to ring k. Letting

$$S_i = \{\rho \in E\colon \rho = (i, j) \text{ or } \rho = (j, i) \text{ for some } j\}, \quad (1)$$

the ring design problem (RD) can be stated as follows.

$$\textbf{RD}\colon \text{Minimize} \sum_{i \in N} \sum_{k \in M} x_{ik} \quad (2a)$$

subject to

$$\sum_{k \in M} f_{\rho k} = 1 \forall \rho \in E \quad (2b)$$

$$\sum_{\rho \in E} d_\rho f_{\rho k} \leqslant b \forall k \in M \quad (2c)$$

$$\sum_{i \in N} x_{ik} \leqslant R \forall k \in M \quad (2d)$$

$$0 \leqslant f_{\rho k} \leqslant x_{ik} \forall i \in N,\ k \in M,\ \rho \in S_i \quad (2e)$$

$$x \text{ binary}. \quad (2f)$$

The objective function seeks to minimize the total number of node-to-ring assignments. Since each such assignment involves the installation of an ADM, this objective is equivalent to minimizing the total ADM capital plus deployment costs (assuming that these costs per ADM unit are the same at all node locations). Constraint (2b) requires that the demand between each node pair be satisfied across all the rings, Constraint (2c) requires that the total demand assigned to each ring should not exceed the ring demand capacity as discussed in the introduction, while (2d) requires the number of nodes assigned to each ring to be no more than the allowable limit R. Finally, (2e) states that any portion of the demand between nodes i and j may be satisfied on ring k only if both nodes i and j are placed on ring k, and (2f) represents logical restrictions. For feasibility to (2), note that we must necessarily have

$$m \geqslant \left\lceil \frac{\sum_{\rho \in E} d_\rho}{b} \right\rceil \text{ and } Rm \geqslant n. \quad (3)$$

Observe that Problem RD bears some resemblance to capacitated facility location, multiple-knapsack resource allocation, or bin packing problems (see Nemhauser and Wolsey 1988 for a general description of such problems). Here, the rings play the role of facilities or knapsacks or bins, each having a capacity of satisfying b demand units, but unlike typical problems of the foregoing type, also having an additional capacity restriction of accommodating at most R nodes. The customers in this problem are represented by pairs of nodes ρ, having respective demands for d_ρ resource units, for $\rho \in E$. The problem involves determining an assignment of each node to one or more facilities/knapsacks/bins, along with an allocation of customer demands based on this assignment, such that the total number of node placements are minimized, subject to the node cardinality and demand resource capacity restrictions. Hence, this problem possesses certain specialized as well as generalized features with respect to the foregoing related classes of problems, which motivates the need for an independent analysis.

To begin with, consider the following result which ex-

Table I. Description of Test Sets

	n	m	b	R	# Demand Pairs
Set 1	7	4	15	4	8
Set 2	10	6	25	5	15
Set 3	13	7	40	5	24

poses the inadequacy of the formulation RD and prompts the generation of improved model representations.

Proposition 1. *If RD is feasible, then an optimal solution to its LP relaxation $\overline{RD}$ is given by*

$$\bar{x}_{ik} = \frac{1}{m}\ \forall (i, k),\ \bar{f}_{\rho k} = \frac{1}{m}\ \forall \rho, k. \tag{4}$$

Proof. Feasibility of (4) to $\overline{\text{RD}}$ is readily verified using (3). The objective value for this solution (4) is equal to n, while from (2e, 2b), we have for any feasible solution that

$$\sum_i \sum_k x_{ik} \geqslant \sum_i \sum_k f_{\rho(i)k} = \sum_i (1) = n \tag{5}$$

where for each $i \in N$, $\rho(i)$ is some node pair from E that contains i. Noting the objective function (2a) and (5), we deduce that (4) is an optimal solution to $\overline{\text{RD}}$. This completes the proof. ■

In the algorithmic design described in Sections 3–5, there are various alternative strategies for performing different steps or operations that we will be evaluating computationally. Toward this end, we generated three sets of 15 test problems each. Each of these sets contains problems having different sizes to illustrate the effect of the different strategies on solving problems of increasing difficulty. Table I lists the problem specifications for each set of problems. The demand values were generated as follows: 80% of the demands were generated uniformly between 1 and 5, while the other 20% were generated uniformly between 1 and 25. This scheme, adopted from Lee et al. (1999a, 1999b), generates more difficult problems than a simple uniform generation scheme. Also, the selected values of n, b, and R are typical to those encountered in practice. The values for m were selected feasible to Equation (3) such that the ratio of m to n lies between 0.5 and 0.6, as is common in such planning scenarios.

One difficulty with solving problem RD is the number of alternative optimal solutions it possesses because of the fact that for any design, there are several equivalent (identical) designs that can be obtained by simply changing the indexing of the rings. This symmetry can hopelessly encumber a branch-and-bound process. We therefore investigated several alternative formulations to RD which incorporate hierarchical structures for reducing the problem symmetry of RD by enforcing branching decisions or by requiring certain allocation functions to take on monotone values over the set of rings. One such formulation places a hierarchy on the number of nodes that are assigned to each ring, where the first ring is designated to receive the most nodes, followed by ring 2, and so on. We denote this formulation as RDS and accordingly, in lieu of (2d), we include

$$R \geqslant \sum_{i \in N} x_{i1} \geqslant \sum_{i \in N} x_{i2} \geqslant \ldots \geqslant \sum_{i \in N} x_{im}. \tag{6}$$

Table II shows a comparison of the effectiveness of the formulations RD and RDS with respect to the average number of branch-and-bound nodes enumerated by CPLEX 6.0, and the corresponding cpu time on a SUN Ultra 10 Workstation, using the foregoing test-bed of 45 problems. We denote RDS(RD) as the computational statistics for RDS corresponding to the set of problems solvable by RD. Note that RD performs quite poorly in comparison with RDS due to number of different symmetric solutions that must be eliminated by the branch-and-bound process. Hence, we will adopt the model RDS to study further enhancements and algorithmic strategies.

Remark 1. Note that there are several different constraint hierarchies that we may impose to reduce the symmetry of this problem. One such hierarchy could place the most demand on ring 1, then on ring 2, and so on. Also, for the hierarchy used in formulation of RDS itself, wherever there exist ties in the number of nodes assigned, we could employ the aforementioned demand hierarchy to break ties. (A single function can be derived to enforce this two-level hierarchy.) Another option might be to substitute the use of total demand assigned to each ring in the hierarchy with the total *fractions* of such demands. Alternatively, we may enforce a hierarchy based on the sum of node indices assigned to each ring, or to reduce the likelihood of any remnant symmetry, we may enforce a hierarchy on the squared sum of node indices assigned to each ring. Although RDS is perhaps milder than several of these alternative hierarchies in breaking symmetry, we discovered empirically that it results in a relatively superior computational performance. The reason for this might be that it adds a set of constraints to the problem in which the nonzero variable coefficients are all $\pm$ 1's, exhibiting a network structure, which tends to encourage integrality of solutions.

Before proceeding with any further analysis of this SONET design problem, let us point out a practical aspect of the demand allocation scheme. Note that given integral values of the demands d_ρ, $\rho \in E$, and the capacity b, we would expect that an optimal solution to Problem RDS, if it exists, would also have integral values of the portions of individual node-pair demands allocated to each ring. The following result establishes this fact, and shows that a linear programming based branch-and-bound algorithm would automatically determine such an optimal allocation scheme. Notationally, any non-subscripted variable below represents the vector of the set of subscripted variables corresponding to the same symbol.

Proposition 2. *Consider Problem RDS, possibly augmented with additional valid inequalities in terms of the x-variables, and assume that the data is all integral and that an optimum exists. Define*

$$y_{\rho k} \equiv d_\rho f_{\rho k} \tag{7}$$

Table II. Comparison of Formulations

Formulation	Set 1			Set 2			Set 3		
	A	B	C	A	B	C	A	B	C
RD	592.00	2.31	15	13982.4	124.70	15	161543.27	2792.59	9
RDS	344.33	1.28	15	2219.33	31.86	15	113378.60	2926.21	15
RDS(RD)	344.33	1.28	15	2219.33	31.86	15	30606.56	1158.86	9

A, Average number of nodes enumerated by the branch-and-bound tree. B, Average CPU Time (in seconds) required by the algorithm. C, Number of problems (out of 15) solved within a limit of 5 hours. The averages in columns A and B are for problems solved within 5 hours.

to be the portion of demand d_ρ that is allocated to ring k, $\forall \rho \in E$ and $k \in M$. If (x^*, f^*) solves Problem RDS, where f^* is an extreme point of the feasible region to RDS with x fixed at x^*, then the corresponding vector y^* given by (7) has all integral components.

Proof. Consider Problem RDS with x fixed at x^*, and for each $\rho \equiv (i, j) \in E$, let $M_\rho = \{k \in M: x^*_{ik} = x^*_{jk} = 1\}$. The feasible region of RDS for $x \equiv x^*$ is then given by

$$F \equiv \left\{ f \geqslant 0: \sum_{k \in M} f_{\rho k} = 1 \forall \rho \in E,\ \sum_{\rho \in E} d_\rho f_{\rho k} \leqslant b\ \forall k \in M,\ f_{\rho k} \equiv 0\ \forall \rho \in E,\ k \notin M_\rho \right\}.$$

Now, consider the transformation of F under the substitution (7). This yields the set

$$Y \equiv \left\{ y \geqslant 0: \sum_{k \in M} y_{\rho k} = d_\rho \forall \rho \in E,\ \sum_{\rho \in E} y_{\rho k} \leqslant b\ \forall k \in M,\ y_{\rho k} \equiv 0\ \forall \rho \in E,\ k \notin M_\rho \right\}.$$

Since (7) represents a nonsingular linear transformation, the set of extreme points of the polyhedra F and Y defined above are in a one-to-one correspondence. Moreover, the set Y possesses a (transportation) network structure (see Bazaraa et al. 1990, for example), and hence has integral extreme points, given that $Y \neq \emptyset$. Therefore, the vector y^* corresponding to the solution f^* specified in the proposition is integral valued, and this completes the proof. ■

2. Computational Complexity Analysis

Define the following *decision problem* **SONET-DP** based on the synchronous optical network ring optimization problem (RD or RDS) introduced in the foregoing section. **SONET-DP**: Given m rings, each with a capacity of accommodating R nodes and routing b demand units; given a demand graph $G(N, E)$ having node set $N = \{1, \ldots, n\}$ and edge set $E = \{(i, j), i < j$: there exists a demand $d_{ij} > 0$ between node pairs i and $j\}$, and given κ ADMs, does there exist an assignment of nodes to rings (permitting a node to be assigned to more than one ring), along with an allocation of demand between assigned pairs of nodes on each ring, such that the total demand is satisfied, the ring capacity restrictions are satisfied, and the total number of node-to-ring assignments (objective value) is less than or equal to κ?

Proposition 3. *Problem SONET-DP is NP-Complete, even if the demand graph G is restricted to be a forest graph having only two nodes per component.*

Proof. The problem SONET-DP is clearly in class NP because given any yes-instance of the problem, there exists an assignment of nodes to rings from among the finite population of all possible polynomial-length assignment vectors for which an answer of yes can be verified in polynomial time by solving a linear programming (LP) problem to determine the associated feasible routing of demand. Hence, to demonstrate NP-completeness of SONET-DP, we need to show that it is NP-Hard (Garey and Johnson 1979). We accomplish this via a polynomial reduction from the NP-Complete *Partition Problem* (PP) (Garey and Johnson 1979). Problem PP considers a given finite set of p nonnegative integers $\{a_1, \ldots, a_p\}$ indexed by $A = \{1, \ldots, p\}$, where $S = \sum_{i=1}^{p} a_i$ is even, and seeks if there exists a partition $\{A', A - A'\}$ of A such that

$$\sum_{i \in A'} a_i = \sum_{i \in A - A'} a_i = S/2. \tag{8}$$

Given any instance of PP, let us construct the following equivalent instance of SONET-DP. Let $m = 2$, $n = 2p$, $R = 2p$, $b = S/2$, $E = \{(i, p + i), i = 1, \ldots, p\}$ with $d_{i,p+i} = a_i\ \forall i = 1, \ldots, p$, and let $\kappa = n \equiv 2p$. Note that the size of SONET-DP is polynomially related to that of PP. Moreover, if PP has an answer yes given by $A' \subseteq A$, then by assigning the nodes i and $p + i\ \forall i \in A'$ to one ring, and the remaining nodes to the other ring, we would satisfy the demand and the capacity constraints, and obtain an objective value of $n = 2p$. Conversely, if SONET-DP has a yes-answer for $\kappa = n = 2p$, then since this means that the demand is satisfied without replicating any node on the two rings, we must have a solution in which for some $A' \subset A$, the set of nodes $\{i$ and $p + i\ \forall i \in A'\}$ are assigned to one ring and the remaining nodes $\{i$ and $p + i\ \forall i \in A - A'\}$ are assigned to the other ring, such that (8) holds true. Hence, PP would then have a solution. Therefore, a given instance of PP is a yes-instance if and only if the polynomially transformed instance of SONET-DP is a yes-instance, and so, SONET-DP is NP-Hard. Noting that the transformed instance of SONET-DP has a forest demand

graph having two nodes per component, this completes the proof. ■

Remark 2. By the nature of the proof, both the present case considered in this paper, as well as the special case in which demand cannot be split among rings, are NP-Complete. Note that for the non-split demand case, the transformed instance of SONET-DP would have a feasible solution regardless of $\kappa \geqslant n$ if and only if the given instance of PP is a yes-instance. For the split demand case of this instance, by assigning one suitable pair of nodes to both the rings, we can always find a feasible solution to SONET-DP when $\kappa \geqslant n + 2$.

Proposition 4. *SONET-DP is NP-Complete even if G is restricted to be a star tree graph.*

Proof. Following the proof of Proposition 3, given PP, construct an instance of SONET-DP having $m = 2$, $n = p + 1$, $R = p + 1$, $b = S/2$, $E = \{(i, p + 1), i = 1, \ldots, p\}$ with $d_{i,p+1} = a_i$ for $i = 1, \ldots, p$, and let $\kappa = p + 2 \equiv (n + 1)$. Then if PP has a solution given via $A' \subseteq A$, SONET-DP has a solution in which the nodes $\{A', p + 1\}$ are on one ring and the nodes $\{A - A', p + 1\}$ are on the other ring. Conversely, if SONET-DP has a solution, since it is impossible to achieve this without assigning some node to both the rings by the nature of this given instance, we must have the number of node allocations equal to $(n + 1) \equiv p + 2$. Since node $p + 1$ must necessarily belong to both rings, the remaining nodes $\{1, \ldots, p\} \equiv A$ must be partitioned into a set A' that resides on one ring, and the set $A - A'$ that resides on the other, such that (8) holds true. Hence, a given instance of PP is a yes-instance if and only if the polynomially transformed instance of SONET-DP is a yes-instance. This completes the proof. ■

Remark 3. It is insightful to note that the following situation can occur in the context of the problem SONET-DP. Let $G(N, E)$ be a connected demand graph, and let m be the fewest number of rings for which a feasible solution exists. Then the fewest number of ADM allocations for a feasible solution with m available rings could exceed that if more than m rings are available. The following example illustrates this fact.

Example 1. Consider a forest demand graph having two nodes per component, with $m = 2$, $n = 6$, $R = 4$, $b = 3$, $E = \{(1, 2), (3, 4), (5, 6)\}$, and with all demands equal to 2. The optimal solution to this problem uses 8 ADMs, with ring 1 being assigned to handle all of the demand between nodes 1 and 2 and half the demand between nodes 3 and 4, and ring 2 being assigned to handle all of the demand between nodes 5 and 6 and half the demand between nodes 3 and 4. However, if $m = 3$, then the optimal solution would use only 6 ADMs, with each ring being assigned to each demand pair.

3. Algorithmic Details: Preprocessing and Branching Priorities

In this section we discuss some specific algorithmic strategies for solving the SONET ring design problem. In particular, we describe a specialized preprocessing step for fixing certain x variables based on logical tests derived from RDS, and we also describe node ordering strategies designed to enforce integrality first on certain critical variables in a branch-and-bound process.

3.1 Preprocessing

The special preprocessing routine that we implement to further tighten the representation of Problem RDS is based on the following results. Proposition 5 below examines a node having the highest traffic demand with the other nodes, or having positive traffic demand with the highest number of nodes, and prescribes a minimal number of rings to which this node must necessarily be assigned.

Proposition 5. *Consider a node i_d having a highest total traffic demand with the other nodes, i.e., $i_d \in \operatorname{argmax}_{i \in N}\{\Sigma_{\rho \in E:\, i \in \rho} d_\rho\}$. Let $r_d = \lceil \Sigma_{\rho \in E:\, i_d \in \rho} d_\rho / b \rceil$. Then the following inequality is valid:*

$$\sum_k x_{i_d k} \geqslant r_d. \tag{9}$$

Also, let $i_a \in N$ be a node having the maximum degree in G (denoted $deg(i_a)$), and let $r_a = \lceil deg(i_a)/(R - 1) \rceil$. Then

$$\sum_k x_{i_a k} \geqslant r_a \tag{10}$$

must also hold true. Thus, by letting $r \equiv r_d$ and $i \equiv i_d$ if $r_d \geqslant r_a$, and $r \equiv r_a$ and $i \equiv i_a$ otherwise, we may fix $x_{i1} = x_{i2} = \ldots = x_{ir} = 1$ in any optimal solution.

Proof. The validity of (9) and (10) follows from (2c, e, f) and (2d, f), respectively. Given that the identified node i must be assigned to at least r rings, we may mitigate effects of symmetry by pointedly assigning it to rings $1, \ldots, r$ without any loss of generality. This completes the proof. ■

Remark 4. Note that if we fix node i on rings 1 through r as in Proposition 5, the hierarchical loading constraints of the form (6) must be imposed separately on the two sets of rings $\{1, \ldots, r\}$ (if $r \geqslant 2$), and $\{r + 1, \ldots, m\}$ (if $r \leqslant m - 2$).

The next result given below characterizes part of an optimal solution, given that the demand graph has a component that can be feasibly assigned to a single ring, and given a sufficient number of admissible rings. Based on this, Corollaries 1 and 2 prescribe optimal solutions for related special cases.

Proposition 6. *Let $G(N, E)$ represent the demand graph for Problem RDS, and suppose that $G'(N', E')$ is a component (maximal connected subgraph) of G that satisfies*

$$|N'| \leqslant R \text{ and } \sum_{\rho \in E'} d_\rho \leqslant b. \tag{11}$$

Then, if $m \geqslant 1 + \Sigma_{\rho \in E - E'} \lceil d_\rho / b \rceil$, we will have that all the nodes in N' along with the demand represented by E' will be allocated to one single ring in any optimal solution.

Proof. On the contrary, suppose that more than one ring contains nodes from N' in some optimal solution (x^*, f^*). Since G' is connected, there exists a pair of rings for which

Table III. Comparison of Procedure with and without Preprocessing

	Set 1		Set 2		Set 3	
Preprocessing	Nodes	CPU Time	Nodes	CPU Time	Nodes	CPU Time
Without	310.07	1.38	2,730.60	34.40	63,992.79	1,350.74
With	126.07	0.70	903.27	14.83	28,077	1,082.63

Note: The numbers represent averages over the test problems solved for each set.

some node in N' appears in both these rings. Hence, we have

$$\sum_{i\in N'} \sum_{k\in M} x^*_{ik} \geq |N'| + 1. \tag{12}$$

Note that there exists at least one ring r, say, that either has no nodes assigned to it or that contains nodes only from N'. Otherwise, all the m rings would contain at least a pair of nodes from $N - N'$, and an improved solution would result by simply assigning G' to one ring, and then assigning the demand d_ρ for each $\rho \in E - E'$ to $\lceil d_\rho/b \rceil$ separate rings, because $m \geq 1 + \Sigma_{\rho\in E-E'} \lceil d_\rho/b \rceil$. But now, by moving all the demand on G' and all the nodes in N' to this ring r, we would maintain feasibility due to (11), while the objective value would strictly fall due to (12), since no nodes in N' would repeat in this revised solution. This is a contradiction, and the proof is complete. ■

Corollary 1. *If G is such that (11) is satisfied for each of its components, and if the number of components of G' is no more than m, then an optimal solution is obtained by assigning each component of G along with its associated demand to a single, separate ring.*

Proof. Note that the stated solution is feasible and has an objective function value of n. Since n is a lower bound on the problem, this completes the proof. ■

Corollary 2. *If G is such that it can be partitioned into subgraphs where each subgraph is a union of some components of G, and if each subgraph $G'(N', E')$ satisfies (11) with m being at least equal to the number of such subgraphs, then an optimal solution is obtained by assigning each subgraph along with its associated demand to a single, separate ring.*

Proof. Similar to that of Corollary 1. ■

The overall preprocessing routine for RDS that is performed prior to inputting it into CPLEX may then be specified as follows.

Step 1. If Corollary 1 applies, construct an optimal solution and stop.

Step 2. Check if Corollary 2 applies by assigning components in order of nonincreasing total demand weight to a ring that has the highest residual capacity. If this reveals that Corollary 2 is applicable, construct the corresponding optimal solution and stop.

Step 3. If Proposition 6 is applicable, assign G' to a single ring and reduce the problem to solving the residual problem having a demand graph $G - G'$ on the remaining $(m - 1)$ rings.

Step 4. Apply Proposition 5, revising the hierarchy constraints as necessary based on Remark 4.

In order to make the 45 test problems generated sufficiently challenging, we avoided cases where Proposition 6 or Corollaries 1 and 2 would be applicable. Yet, using only Step 4 of the above routine, and even though r rarely exceeded two in these test problems, the computational advantage of this preprocessing routine turned out to be significant, as evident from Table III. In these computations, formulation RDS has been used, along with an initial heuristic solution derived as discussed in Section 5, prior to inputting it into CPLEX. Note that Table III actually describes the average performance of 14 of the 15 test problems from Set 3, since one problem from Set 3 could not be solved without this preprocessing routine due to memory restrictions on the branch-and-bound tree.

3.2 Priorities for Selecting Branching Variables

This subsection discusses different branching variable selection priority rules that we experimented with in our solution procedure. Given branching priorities for groups of variables, the branch-and-bound strategy implemented within CPLEX will branch on a lower priority variable only if all the relatively higher priority variables are integer valued at the current solution. Here, the following branching priority rules were investigated.

B1 Branch on the set of m variables $\{x_{ik}, k = 1, \ldots, m\}$ associated with the lowest degree node $i \in N$ first, then on the set of x-variables associated with the next lowest degree, and so on.

B2 Same as B1, except that instead of sorting nodes by their degrees, nodes are sorted by the sum of the demands on the incident edges, henceforth referred to as the *demand degree* of a node.

B3 Same as B1, except that instead of sorting nodes by their degrees, nodes are sorted by the products of their respective degree and demand degree.

B4 Same as B1, but where ties are broken between nodes having the same degree by selecting a lower demand degree node first.

B5 Same as B1, but where ties are broken between nodes having the same degree by selecting a higher demand degree node first.

B6 Same as B5, but now, for each group of m variables $\{x_{ik}, k = 1, \ldots, m\}$ for $i \in N$, an additional priority is

Table IV. Comparison of Branching Rules on Different Formulations

Formulation	Order	Set 1		Set 2		Set 3	
		Nodes	CPU Time	Nodes	CPU Time	Nodes	CPU Time
RDS	B0	126.07	0.70	903.27	14.83	30,744.53	1,206.32
RDS	B5	59	0.47	797.13	12.33	24,235.47	1,027.38
RDS	B7	83.07	0.50	574.13	9.45	22,073.13	859.55
RDI	B0	137.80	0.74	1,209.73	22.83	47,906.20	2,222.55
RDI	B5	91.87	0.59	584.20	11.16	19,653.33	897.26
RDI	B7	64.87	0.49	618.80	11.67	38,330.60	1,146.17

imposed that considers the variables within each such group in the stated order.

B7 Same as B6, but where the variables within each group are prioritized in the reverse order. (Note that the ordering scheme imposed by B6 and B7 attempts to investigate the joint effect of such a strategy used in concert with the hierarchical constraints of (6).)

There were several other branching variable prioritization schemes that were explored. For example, rules B1 through B5 were reversed with respect to the ordering prescribed by B1, but this produced vastly inferior results. Evidently, nodes having relatively smaller degrees of interactions have a greater tendency to fractionate while consuming portions of residual demand capacities on more than one ring, and resolving their integrality hastens the completion of the overall solution. Among nodes having the same degree, it is more critical to resolve integrality on those having a relatively higher demand degree. Furthermore, because the hierarchical constraint (6) imposes a greater number of node assignments to lower indexed rings, the resolution of integrality on the assignments made to the relatively higher indexed rings tends to induce integrality on the remainder of the solution. In addition, we attempted various combinations of the hierarchical constraints embodied by (6) and Remark 1 with the above branching priority rules. One such additional formulation imposes a hierarchy on the squared sum of node indices assigned to a ring. To formalize this hierarchy, we create the formulation RDI by adding the following constraints to RD:

$$\sum_{i\in N} i^2 x_{i1} \geqslant \sum_{i\in N} i^2 x_{i2} \geqslant \ldots \geqslant \sum_{i\in N} i^2 x_{in}. \tag{13}$$

Table IV illustrates the effectiveness of certain key or promising combinations that are worthy of attention, using the preprocessing scheme of Section 1.1 and the heuristics of Section 5. Here, B0 denotes the default branching order used by CPLEX. Preliminary tests demonstrated that B5 and B7 were the most effective branching priority rules. Note also that RDI becomes an attractive formulation when used in concert with B5. The best combination strategy appears to be to use RDS with B7.

4. Classes of Valid Inequalities

In this section, we derive two classes of valid inequalities, along with separation routines, designed to further tighten the continuous feasible region for Problem RDS. Constraints (2c) and (2d) motivate the construction of these inequalities, as expounded by the following propositions. The proofs for these results are similar to those for the case of non-split demand allocations analyzed in Lee et al. (1999a, 1999b), and are therefore omitted for brevity. However, the separation routines for generating effective members of these classes of inequalities are new, and are therefore presented in greater detail below. We also investigated another class of valid inequalities by constructing a higher dimensional partial convex hull formulation of RD with respect to each binary variable along with constraints (2c, e, f) using the Reformulation-Linearization Technique of Sherali and Adams (1994), and then surrogating subsets of the remaining constraints using the optimal dual multipliers to derive strong valid constraints in the space of the original variables. However, because the formulation RDS is relatively tight, the effort to generate such additional constraints far outweighed the incremental strength imparted to the model representation, and hence did not reveal an overall benefit. In contrast, the following classes of valid inequalities significantly enhanced the solution approach, because they are sufficiently strong and can be efficiently generated as shown below.

Proposition 7. *Suppose that $G'(N', E')$ is a connected subgraph of $G(N, E)$ such that $|N'| > R$. Let $r = \lceil (|N'| - 1)/(R - 1) \rceil$. Then the following inequality is valid for Problem RDS:*

$$\sum_{i\in N'} \sum_{k\in M} x_{ik} \geqslant |N'| + (r - 1). \tag{14}$$

■

Proposition 8. *Suppose that $G'(N', E')$ is a connected subgraph of $G(N, E)$, where E' is induced by N', such that $\Sigma_{\rho\in E'} d_\rho > b$. Then the following inequality is valid for Problem RDS:*

$$\sum_{i\in N'} \sum_{k\in M} x_{ik} \geqslant |N'| + (r - 1) \tag{15}$$

where

$$r = \left\lceil \sum_{\rho\in E'} \frac{d_\rho}{b} \right\rceil. \tag{16}$$

Also, if $r = 2$ in (16) and G' does not contain any node of articulation (i.e., a node which when removed along with its

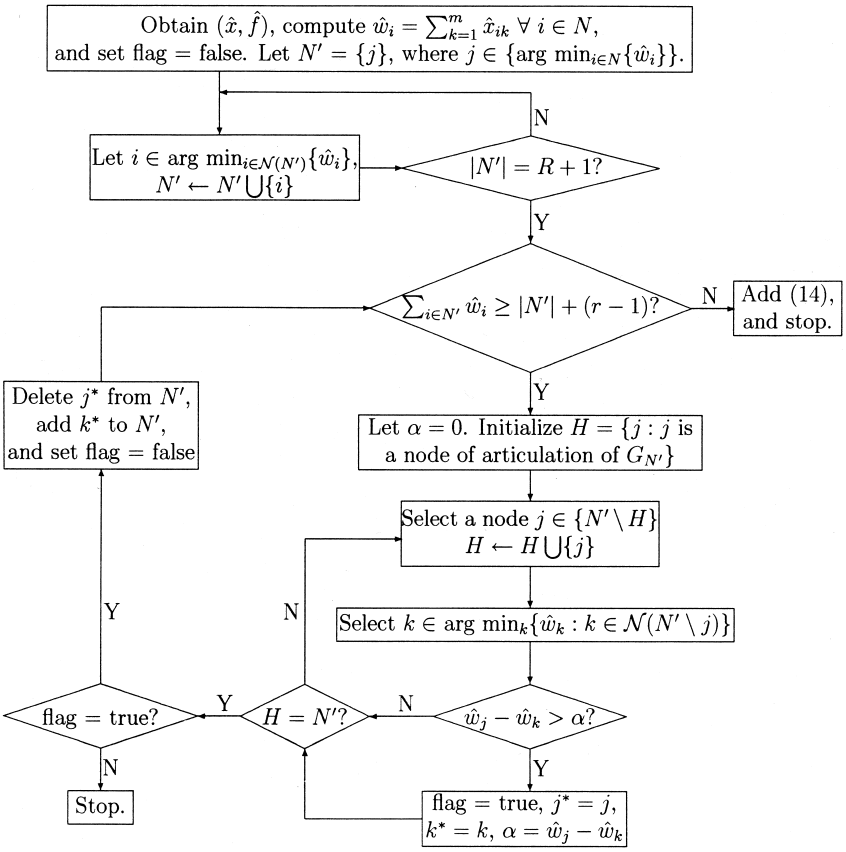

Figure 2. Growth procedure for generating (14).

incident arcs would disconnect the remainder of G′ into more than one component), then the following inequality is valid:

$$\sum_{i \in N'} \sum_{k \in M} x_{ik} \geqslant |N'| + 2. \tag{17}$$

■

For each of the inequalities of type (14) and (15), an appropriate subgraph G' of G must be chosen. The following separation routines identify such subgraphs for generating these constraints based on the linear relaxation solution obtained for Problem RDS. Accordingly, let $(\hat{x}, \hat{f})$ denote the solution obtained for the linear relaxation $\overline{\text{RDS}}$ of RDS. Define y_i to be equal to one if node $i \in N$ is included in G', and zero otherwise, and define $\hat{w}_i = \Sigma_{k=1}^{m} \hat{x}_{ik}$ for each node $i \in N$. Then we can conceptualize the following separation problem for generating (14) via Proposition 7.

$$\text{Minimize} \sum_{i=1}^{n} \hat{w}_i y_i \tag{18a}$$

$$\text{subject to} \sum_{i=1}^{n} y_i \geqslant R + 1 \tag{18b}$$

$$N' = \{i \in N\colon y_i = 1\} \text{ induces a connected subgraph} \tag{18c}$$

$$y \text{ binary.} \tag{18d}$$

If the optimal value of (18) is less than the right-hand side of (14) for the subgraph generated by the optimal solution y to this problem, the corresponding inequality (14) would be a valid cut that deletes the current solution to $\overline{\text{RDS}}$. One possible heuristic solution procedure for solving this problem is to select a seed node, and to then grow a subgraph from this node by selecting the lowest $\hat{w}_i$-valued nodes from among all the nodes that are reachable from the current subgraph. This greedy heuristic is described in Figure 2. Here, we define $\mathcal{N}(N') = \{i \in N\colon i \notin N', i$ is adjacent to some node in $N'\}$, and $G_{N'}$ to be the subgraph induced by nodes N' on the graph with edge set E. After the seed subgraph is grown to $|N'| = R + 1$, this procedure attempts to minimize the objective function in (18) by selecting a node j to be removed from N' and a node k to be added to N' such that the new subgraph is still connected. This is ensured by requiring that j is not a node of articulation in $G_{N'}$, and that k is connected to some node other than j in N'. These exchanges are repeated until either a cutting plane is generated, or until no such exchange that reduces the objective function value can be found. Alternatively, Figure 3 describes a heuristic which shrinks G one node at a time by removing the node having the largest $\hat{w}_i$ value at each iteration, such that the graph remains connected, having more

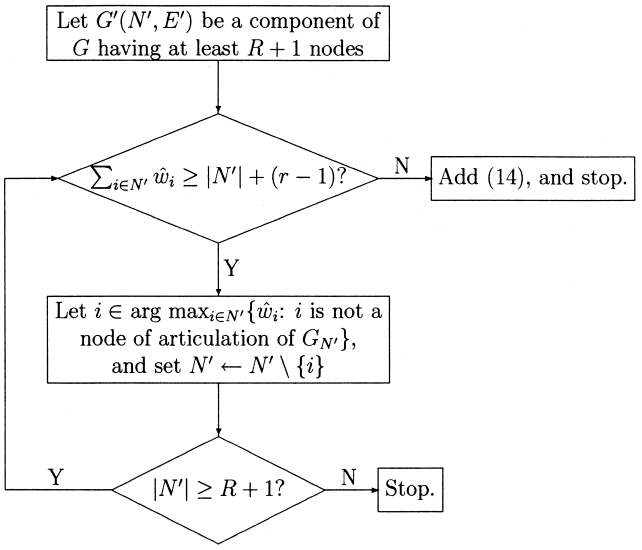

Figure 3. Shrinking procedure for (14).

than R nodes. Each of these procedures checks to see if the current solution is violated by Equation (14) for each valid $G' \equiv G_{N'}$ that is generated during the procedure.

Similar procedures may be devised for generating valid inequalities of the type (15) or (17). Here, instead of satisfying the requirement of including more than R nodes in G', we wish to satisfy a requirement on the total demand in G'. Thus, we wish to construct G' using nodes that contribute a minimal value to the objective of type (18a) relative to the demand they add to the subgraph. Accordingly, define $\delta_i = \Sigma_{j \in N'(i)} d_{ij}$, where $N'(i)$ is the set of nodes adjacent to i in G'. The separation problem for this case can be conceptualized as follows.

$$\text{Minimize} \sum_{i=1}^{n} \hat{w}_i y_i \tag{19a}$$

$$\text{subject to} \sum_{i \in N'} \delta_i \geq 2b + 1 \tag{19b}$$

The subgraph $G_{N'}$ induced by $N' = \{i \in N$:

$$y_i = 1\} \text{ is connected} \tag{19c}$$

$$y \text{ binary.} \tag{19d}$$

The proposed heuristics for solving (19) are similar to those for solving (18). Note that the graph $G' \equiv G_{N'}$ in this context. Figures 4 and 5 provide details for the proposed separation routines. Note that a version of these procedures could be run *a priori* using $\hat{w}_i \equiv 1\ \forall i \in N$ to determine small subgraphs which exhibit the property of having demand greater than b. The resulting solution can be used to generate the corresponding valid inequalities (15) or (17) that can be added to the model RDS prior to solving its initial relaxation.

We ran our test-bed of 45 problems on the formulation RDS with and without the valid inequalities generated using our separation routines in the following manner. We ran each of the routines in Figures 2 through 5, and augmented the model with each constraint generated. After updating the solution to the new linear programming relaxation, we re-ran the cutting plane generation routines. This was continued until no more cutting planes could be generated in this fashion. The average number of valid inequalities generated via this procedure is 0.53 for Set 1, 3.4 for Set 2, and 2.53 for Set 3. Table V presents the results obtained, exhibiting a significant decrease in the average number of branch-and-bound nodes generated as well as in the overall computational effort by incorporating these valid inequalities. This table demonstrates that the valid inequalities based on the demand capacity constraints embodied by (15) and (16) account for most of the tightening of the problem. The inclusion of valid inequalities also reduces the amount of memory necessary to store the branch-and-bound tree, thus enabling the solution of larger problems on systems having a limited memory capacity.

In a separate experiment, we also tested the effectiveness of incorporating (9) and (10) into the model as valid inequalities. Note that such valid inequalities can be actually generated for each node $i \in N$ in a similar fashion. However, neither the strategies of including these constraints directly in the model for each $i \in N$, nor including only that subset of these inequalities which delete the current fractional solution improved the efficiency of the algorithm. Of these two strategies, the latter appears to help slightly with the more difficult class of problems in Set 3, resulting in an average number of nodes enumerated and average cpu time of 17384.13 and 859.28 seconds, respectively.

5. Heuristic Procedures

In order to derive strong upper bounds for the SONET problem, we developed two heuristic procedures. The first heuristic solves a series of increasingly restricted linear programs, fixing at least one node-to-ring assignment at each step. The second heuristic is a greedy construction procedure. We also augmented these heuristics with an improvement procedure that attempts to reduce the node-to-ring assignments for any given feasible solution.

LP Rounding Heuristic

Step 1: To initialize, set some tolerance level $T = .99$, and let HLP be the linear relaxation $\overline{\text{RDS}}$ to RDS. Proceed to Step 2.

Step 2: Solve HLP and obtain an optimal solution $\hat{x}$. If no feasible solution exists, stop and return no solution. If $\hat{x}$ is integer feasible, stop and return $\hat{x}$ as the prescribed heuristic solution. Otherwise, proceed to Step 3.

Step 3: For each $\hat{x}_{ik} \geq T$, fix $x_{ik} = 1$ within HLP. Proceed to Step 4.

Step 4: Find the maximum fractional value $\hat{x}_{ik}$, and fix $x_{ik} = 1$ in HLP. Return to Step 2.

Note that Step 4 ensures that at least one additional node-to-ring assignment is made in each loop of the algorithm, thus ensuring finite convergence of the process. Also, note that while it is possible that some more than necessary

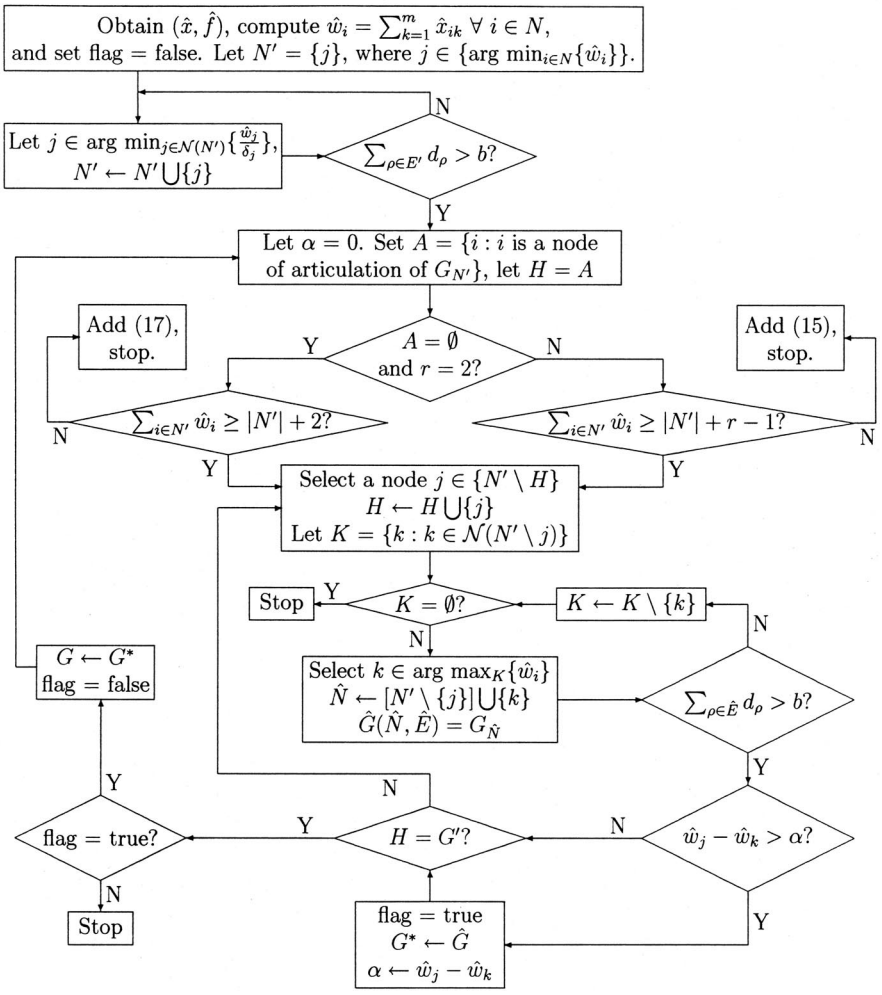

Figure 4. Growth procedure for (15) or (17).

assignments result in the solution following the final iteration of this heuristic, the improvement routine described below is designed to identify and delete such assignments.

Construction Heuristic

In essence, this procedure searches at each iteration for the node which, when added to the current nodes on a ring, contributes the most additional demand. If the additional demand exceeds the demand capacity for a ring, then demand edges are evaluated one by one to consider their inclusion in the current ring. In addition, a threshold value T is used to determine whether or not a partial demand should be added to the current ring, in case the full demand allocation on an edge exceeds the residual capacity. This is done only if the residual capacity exceeds T and if the remaining unsatisfied demand on this edge after the partial allocation is made will also exceed T. This check tends to obviate the situation where a partial demand is added to a ring via a node assignment to simply consume its residual capacity, but at the undesirable expense of having to incur a perhaps unnecessary additional node-to-ring allocation cost. Figure 6 provides a flow-chart to describe the details of this procedure. In this procedure, G' represents the current subgraph of G whose edge set E' is comprised of the as-yet unsatisfied demands. Also, given any node set $L \subseteq N$ currently assigned to a ring under consideration, and given a node $j \in N$, we define $E'_{j,L}$ to be the set of demand edges in E' that involve node j and some node in L. Accordingly, we define $D_j \equiv \Sigma_{\rho \in E'_{j,L}}\ d_\rho$ to be the total demand corresponding to the edge set $E'_{j,L}$. The procedure examines the inclusion of a suitable node j in L on the current ring, along with the allocation of a subset of the demand D_j as outlined above, and detailed in Figure 6.

Remark 5. For the configurations produced by the foregoing heuristics, it is possible for two or more rings to have enough spare capacity such that they could be merged onto one ring. However, the number of rings is generally not a limiting factor in this design problem, although if the number of available rings is indeed limited, we can perform a quick check for merging ring allocations in Step 2 of this procedure.

Improvement Routine

The improvement routine is designed to examine the feasibility of rearranging nodes on rings so as to reduce the number of node-to-ring (ADM) assignments for a given

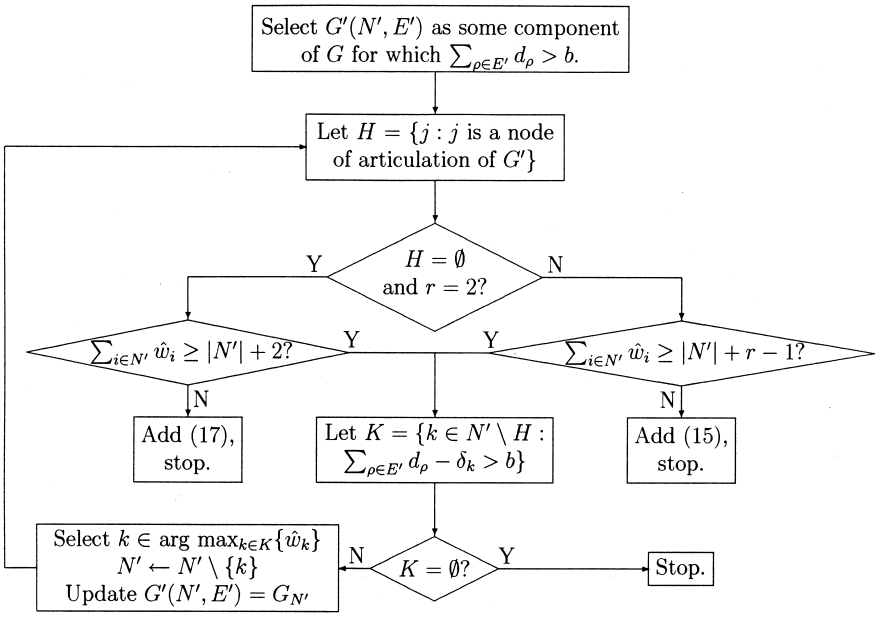

Figure 5. Shrinking procedure for (15) or (17).

Table V. Evaluating the Effect of the Valid Inequalities (VI)

	Set 1		Set 2		Set 3	
VI	Nodes	CPU Time	Nodes	CPU Time	Nodes	CPU Time
No VIs	126.07	0.70	903.27	14.83	30,744.53	1,206.32
With (14)	75.13	0.54	675.93	13.46	20,327.00	1,035.21
With (15) & (16)	69.33	0.50	544.27	11.00	22,094.60	868.08
With all VIs	69.80	0.52	674.50	13.31	18,170.13	861.92

Note: The numbers represent average values over the 15 problems from each set.

solution. This routine solves a mixed-integer program (denoted as IMIP) in which all the x_{ik} variables that are presently zero in the incumbent solution are fixed at zero, but the remaining variables are free to switch values from 1 to 0 if necessary. Solving this restricted problem thus checks for possible consolidations of demands for replicated nodes while reducing the corresponding node-to-ring assignments. IMIP is solved very quickly in practice, with CPLEX usually giving an integer feasible solution at node zero itself. Naturally, if neither heuristic yields a feasible solution, the improvement routine is not performed.

Tables VI and VII present computational results pertaining to the heuristics proposed in this section. These test runs were made with formulation RDS, along with preprocessing. (Note that this choice affects only the LP Rounding heuristic.) Although the construction heuristic often outperforms the LP rounding heuristic, since both these routines are computationally inexpensive, we execute both these procedures and use the resulting solution that has the minimum objective value. Table VII demonstrates the advantage of using IMIP to improve the best upper bound obtained on the problem via this pair of heuristics, especially for larger problems.

Furthermore, in order to assess the effectiveness of the heuristic procedure within the overall algorithm, we ran formulation RDS on the test bed of problems along with branching order B7, preprocessing, and valid inequalities, both with and without this heuristic. Table VIII displays the results obtained. Observe that as the problem size increases, the importance of implementing the proposed heuristic procedure becomes more evident.

6. Computational Summary

In this section, we present a summary computational analysis demonstrating the overall viability of the solution procedure proposed in this paper. Table IX displays the results for Formulation RDS using CPLEX's default settings, as well as for the "Proposed Algorithm" comprised of solving Formulation RDS using the branching priority order B7, with the prescribed cutting plane generation scheme, preprocessing routine, and heuristic procedures. A comparison of these

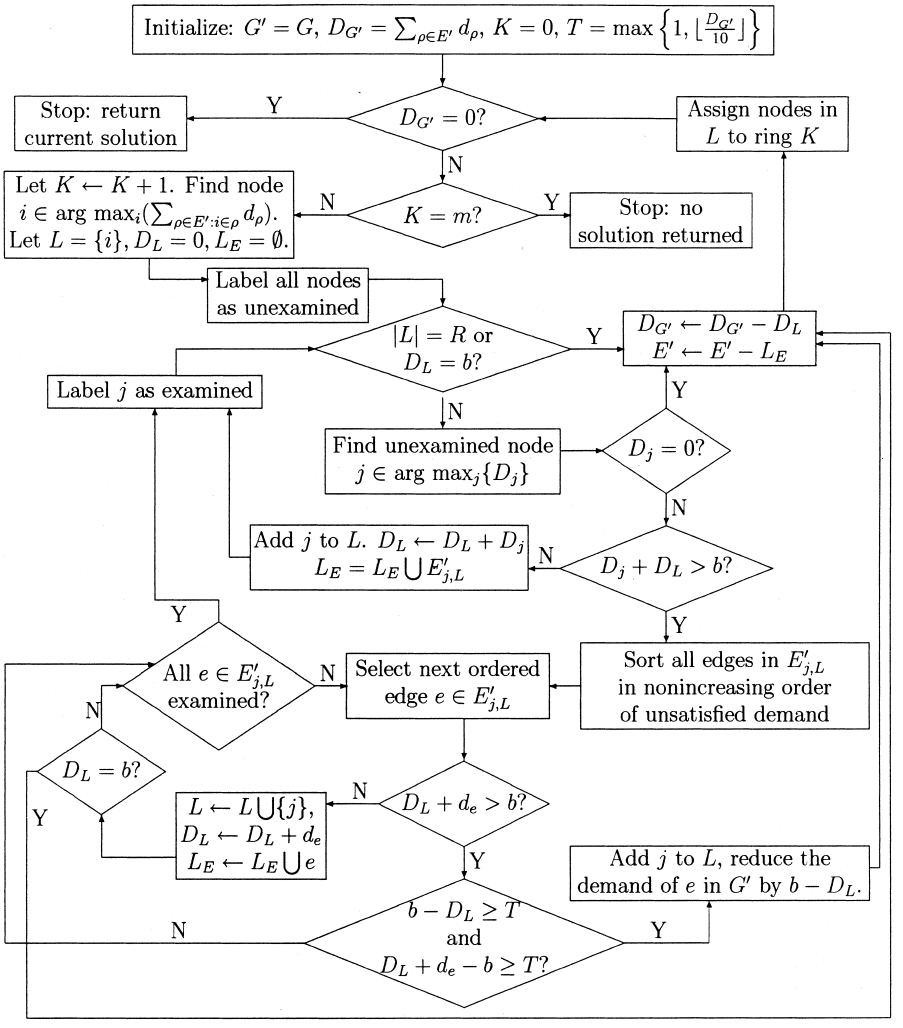

Figure 6. Construction heuristic.

Table VI. Comparison of LP Rounding and Construction Heuristics

	Set 1		Set 2		Set 3	
Heuristic	Avg Obj	% Infeasible	Avg Obj	% Infeasible	Avg Obj	% Infeasible
LP rounding	12.0	33	19.93	0	31.5	20
Construction	10.38	47	18	0	25.47	0

results reveals a strong dominance of implementing the proposed algorithmic enhancements versus using CPLEX's defaults. The savings in cpu time using our algorithm over CPLEX's defaults is considerable: 60.9% for Set 1, 70.3% for Set 2, and 70.6% for Set 3. Furthermore, the savings in the number of branch-and-bound nodes enumerated are 75.9% for Set 1, 74.1% for Set 2, and 80.5% for Set 3.

In a separate experiment, we compared the results and performance of a restricted version of this algorithm on the split-demand case of this problem versus the non-split case. Note that for the non-split case, all f variables are binary valued, which in turn implies that the x variables are binary at optimality. Since the branching orders and heuristics specified for the split-demand case are not relevant to the solution of the non-split case, these algorithmic strategies were omitted. On the other hand, the customized preprocessing step (see Lee et al. [1999a, 1999b] for the non-split demand case) and the generation of valid inequalities were uniformly included for each case in Formulation RDS. The results depicted in Table X demonstrate that the non-split

Table VII. Effectiveness of the Improvement Routine

	Set 1		Set 2		Set 3	
	Upper Bound	% Gap	Upper Bound	% Gap	Upper Bound	% Gap
Before IMIP	11.09	5.17	17.73	13.68	25.47	15.41
After IMIP	11.09	5.17	17.13	8.55	25.07	13.60

Note: The numbers represent average values over the 15 problems from each set.

Table VIII. Evaluating the Effect of the Heuristic Procedure

	Set 1		Set 2		Set 3	
Heuristic	Nodes	CPU Time	Nodes	CPU Time	Nodes	CPU Time
Without	68.87	0.37	673.33	10.52	26,100.87	1,057.36
With	83.07	0.50	574.13	9.45	22,073.13	859.55

Note: The numbers represent average values over the 15 problems from each set.

Table IX. Results for the Formulation RDS with Default CPLEX Settings Versus the Proposed Algorithm

	Set 1				Set 2				Set 3			
	A		B		A		B		A		B	
Prob.	Nodes	CPU	Nodes	CPU	Nodes	CPU	Nodes	CPU	Nodes	CPU	Nodes	CPU
1	59	0.29	4	0.23	1929	32.25	2271	27.71	27663	1133.63	5844	209.54
2	1254	4.08	248	1.00	1599	23.73	781	13.75	4568	172.77	1654	89.23
3	0	0.08	0	0.13	306	4.67	52	2.02	41812	1418.35	4696	151.54
4	228	0.94	26	0.46	390	5.46	16	1.50	177271	7596.58	50167	1814
5	65	0.35	4	0.25	1746	24.96	323	5.67	185454	7978.25	36487	1358.83
6	58	0.35	0	0.21	7427	117.63	2344	35.50	40387	1265.56	9001	343.54
7	100	0.40	18	0.42	534	9.48	91	2.75	60796	1896.67	13966	568.96
8	163	0.65	0	0.27	1539	20.65	77	2.65	9690	472.60	441	23.38
9	56	0.31	0	0.25	3664	57.63	750	12.94	158713	4749.42	25504	701.71
10	241	1.02	24	0.38	1540	21.58	136	3.42	490806	2185.75	8501	375.48
11	1858	6.17	247	1.10	5063	58.88	363	6.52	39075	2081.90	5015	316.77
12	176	0.88	45	0.40	6260	80.19	532	9.94	39162	1377.33	6025	213.40
13	472	1.71	32	0.33	605	9.75	342	6.73	12306	610.96	2052	65.06
14	132	0.63	9	0.29	393	6.92	330	6.23	191658	7920.33	61115	2337.29
15	303	1.27	589	1.83	295	4.15	204	4.42	221318	3032.98	100629	4324.19
Avg	344.33	1.28	83.07	0.50	2219.33	31.86	574.13	9.45	113378.60	2926.21	22073.13	859.55

A, CPLEX default. B, Proposed algorithm.

case is much harder to solve than the split-demand case, especially for larger problems. The row entitled "ratio" gives the ratio of the measure for the non-split case divided by the measure for the split-demand case. Note that 8 of the 15 problems in Set 1 were infeasible for the non-split case. Hence the averages recorded for this case are for the seven problems solved by both the scenarios. Also, all 15 of the problems in Set 3 had identical optimal objective function values for the two cases, although the ring configurations and the demand allocations were not necessarily identical.

7. Summary and Conclusions

In this paper, we have developed an exact cutting plane and enumeration algorithm for a SONET design problem deploying unidirectional path switched rings, modeling it as a mixed-integer program and exploiting certain problem characteristics to enhance its solution capability. We first showed that a straightforward branch-and-bound strategy to solve a traditional mixed-integer formulation for this problem performs poorly, primarily due to the large number of symmetric solutions that the enumeration process must sift through

Table X. Evaluating the Effect of the Non-Split Restriction

Demand Splitting	Set 1			Set 2			Set 3		
	Nodes	CPU Time	Objective	Nodes	CPU Time	Objective	Nodes	CPU Time	Objective
Not allowed	65.71	0.35	9.86	1,830.13	27.59	33.71	70,292.13	2,780.49	47.29
Allowed	33.14	0.23	9.86	691.40	12.76	33.43	24,039.67	1,098.80	47.29
Ratio	1.97	1.52	1.00	2.65	2.16	1.01	2.92	2.53	1.00

Note: The numbers represent average values over the problems solved from each set (7, 15, and 15, respectively).

in determining an optimum. To alleviate this, we investigated several different methods that tend to mitigate the effects of symmetry by imposing certain valid configuration hierarchies, and demonstrated that the inclusion of such additional constraints significantly improves the performance of the algorithm.

We then investigated the effectiveness of various preprocessing routines and of different variable branching priorities. The proposed specialized preprocessing routines implemented led to a sharp decrease in the branch-and-bound effort, and an appropriate prioritized ordering scheme for selecting branching variables also moderately decreased the execution time for the algorithm, especially when used in concert with certain hierarchical constraints. Next, we examined the generation of cutting planes from various classes of valid inequalities, designing suitable separation routines for deriving strong cutting planes that delete the solution to the current linear programming relaxation. Incorporating these cuts was also shown to enhance the effectiveness of the solution algorithm. While these cutting planes were implemented only at node zero to allow the use of a powerful commercial optimization package (CPLEX), it might be worthwhile to explore a specialized branch-and-cut algorithm in which such valid inequalities are generated at other nodes of the enumerative tree as well. In addition, we designed two heuristic procedures along with an improvement routine for providing useful upper bounds on the problem and reducing the amount of effort required by the branch-and-bound process. These procedures typically determined solutions having an objective value within 5–13% of the optimum, thereby promoting their use for possibly obtaining good solutions for larger problems which may not be solvable to exact optimality. Finally, we provided some insights into the non-split demand allocation version of the problem considered in this paper, showing that this case leads to problems that are relatively harder to solve, and its reduced flexibility is more likely to result in infeasible instances or somewhat more expensive optimal solutions.

Acknowledgments

This material is based upon work supported by the *National Science Foundation* under Grant Number DMI-9812047 and by the *Korea Science and Engineering Foundation* under Grant Number 981-1015-084-2. The authors also thank the two anonymous referees, the Associate Editor, and the Area Editor for their constructive comments and suggestions that have led to an improved presentation.

References

Bazaraa, M.S., J.J. Jarvis, H.D. Sherali. 1990. *Linear Programming and Network Flows.* John Wiley & Sons, New York.

Cosares, S., I. Saniee. 1994. An optimization problem related to balancing loads on SONET rings. *Telecommunication Systems* **3** 165–181.

Dell'Amico, M., M. Labbé, F. Maffioli. 1999. Exact solution of the SONET ring loading problem. *Operations Research Letters* **25** 119–129.

Garey, M., D. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, San Francisco.

Goldschmidt, O., A. Laugier, E. Olinick. 1998. SONET/SDH ring assignment with capacity constraints. Working Paper, Department of Industrial Engineering and Operations Research, University of California, Berkeley, CA.

Karunanithi, N., T. Carpenter. 1997. SONET ring sizing with genetic algorithms. *Computers and Operations Research* **24** 581–591.

Laguna, M. 1994. Clustering for the design of SONET rings in interoffice telecommunications. *Management Science* **40** 1533–1541.

Lee, Y., J. Han, S. Kim. 1999a. A physical ring design problem of synchronous optical networks for mass market multimedia telecommunication services. *INFORMS Fall Conference*, Philadelphia, PA, November 7–10.

Lee, Y., H.D. Sherali, J. Han, S. Kim. 1999b. A branch-and-cut algorithm for solving an intra-ring synchronous optical network design problem. Working Paper, Department of Industrial Engineering, Korea University, Seoul, Korea. (To appear in *Networks*.)

Myung, Y.-S., H.-G. Kim, D.-W. Tcha. 1997. Optimal load balancing on SONET bidirectional rings. *Operations Research* **45** 148–152.

Nemhauser, G.L., L.A. Wolsey. 1988. *Integer and Combinatorial Optimization*. John Wiley & Sons, New York.

Sutter, A., F. Vanderbeck, L. Wolsey. 1998. Optimal placement of add/drop multiplexers: heuristic and exact algorithms. *Operations Research* **46** 719–728.

Wasem, O., T.-H. Wu, R. Cardwell. 1994. Survivable SONET networks-design methodology. *IEEE Journal of Selected Area in Communications* **12** 205–212.

Wu, T.-H. 1992. *Fiber Network Service Survivability: Architecture, Technologies, and Design*. Artech House, New York.

Wu, T.-H., M. Burrowes. 1990. Feasibility study of a high-speed SONET self-healing ring architecture in future interoffice fiber networks. *IEEE Communications Magazine* **28** 33–43.