DIMACS Technical Report 97-02
January 1997

Demand Routing and Slotting on Ring Networks

by

Tamra J. Carpenter[1]
Bellcore
445 South Street
Morristown, New Jersey 07960

Steven Cosares
Dept. of Mathematics
Dowling College
Oakdale, New York 11769

Iraj Saniee[2]
Bellcore
445 South Street
Morristown, New Jersey 07960

[1]Permanent Member
[2]Permanent Member

DIMACS is a partnership of Rutgers University, Princeton University, AT&T Research, Bellcore, and Bell Laboratories.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

We describe an important class of problems that arise in the economic design of "survivable" networks. Such networks are capable of accommodating all of the traffic between pairs of locations, even if some arbitrary link or node in the network is rendered unusable. Cycles play an important role in the design of survivable networks because they represent two-connected subnetworks of minimal size. To cost-effectively utilize cycles, we must determine the minimum capacity required for the links in the cycle, subject to constraints on how traffic must be routed and how capacity must be utilized. Depending upon the situation being modeled, different versions of the problem arise. The least restrictive versions are solvable in polynomial time, while the more restrictive (and more realistic) versions are NP-hard. This paper focuses on variants of the problem in which time-slot assignment constraints are enforced to model the operation of the equipment placed at the nodes of a SONET ring. We present several simple heuristic methods for addressing the problem, and we show that they are 2-approximation algorithms.

Demand Routing and Slotting on Ring Networks

Tamra Carpenter, Steven Cosares, and Iraj Saniee

Abstract: We describe an important class of problems that arise in the economic design of "survivable" networks. Such networks are capable of accommodating all of the traffic between pairs of locations, even if some arbitrary link or node in the network is rendered unusable. Cycles play an important role in the design of survivable networks because they represent two-connected subnetworks of minimal size. To cost-effectively utilize cycles, we must determine the minimum capacity required for the links in the cycle, subject to constraints on how traffic must be routed and how capacity must be utilized. Depending upon the situation being modeled, different versions of the problem arise. The least restrictive versions are solvable in polynomial time, while the more restrictive (and more realistic) versions are NP-Hard. This paper focuses on variants of the problem in which time-slot assignment constraints are enforced to model the operation of the equipment placed at the nodes of a SONET ring. We present several simple heuristic methods for addressing the problem, and we show that they are 2-approximation algorithms.

1. Introduction

Today's fiber optic networks are able to concentrate large volumes of traffic onto a relatively small number of nodes and links. As a result, the failure of a single network element can potentially interrupt a large amount of traffic. For this reason, ensuring survivability through fault-tolerant network design is an important network design objective.

In a "survivable" network, the links have sufficient capacity and are inter-connected in such a way that much of the expected network demand is guaranteed to be satisfied, even if some arbitrary link or node in the network is rendered unusable. For telecommunications applications, survivable designs are essential because many service providers wish to guarantee their ability to establish and maintain connections in a timely manner, even during network component failures. Although implementing such features may be expensive, it is thought to be a cost effective alternative to relying on a network that has a high potential for service disruption [NYT]. Service providers that do not offer guarantees that their networks are protected from unforeseen circumstances are likely to lose clients to competitors who do.

In the next section, we provide a brief overview of popular methods for protecting network traffic. These methods include protecting demands on rings that join a group of nodes in order that they may share protection capacity. To do this in the most cost-effective manner, the demands must be packed onto a ring with the smallest possible capacity. Our paper focuses on the resulting ring sizing problem. Ring sizing issues have recently been considered by several

authors ([CS94], [SSW95], [VSKW95]), but without the so-called "slotting" constraints that we consider here (and discussed in [CCS94]). These constraints better-model the way that most current equipment utilize ring capacity to satisfy demands. In this paper, we formulate the Demand Routing and Slotting Problem [DRSP] that includes these constraints, and we discuss a variety of heuristic methods that provide feasible solutions with good quality guarantees. The heuristics that we consider are "two-phased" methods that first determine demand routing and then assign demands to slots. By breaking up the problem in this fashion, we can exploit algorithms previously derived for the individual components. The main contributions of this paper are to: 1) introduce the Demand Routing and Slotting Problem; 2) to present several heuristic methods for obtaining solutions for DRSP; and 3) to demonstrate quality guarantees for these heuristics.

In the next section, we begin our discussion of DRSP by describing the network planning context in which it arises. In section 3, we formally state DRSP and discuss its complexity. We note that less restrictive variants of DRSP can be solved in polynomial time while the more restrictive variants, representing equipment presently in use, are NP-hard. In Sections 4 and 5 we discuss the routing and slotting components of DRSP. In Section 4 we present a basic slotting algorithm that was proposed by Tucker [Tuck75]. In Section 5, we present several methods for determining the routing of demands in a ring network. By combining these algorithms with the slotting algorithm of the previous section, we demonstrate that a number of simple techniques assure DRSP solutions that use no more than twice the optimal number of slots and in many cases are within a potentially tighter bound based on the linear relaxation. The last section concludes with discussions about future issues related to DRSP.

2. Designing Survivable Networks

In this section, we briefly introduce some common mechanisms for survivability planning. Our primary objective is to provide the reader with the general context in which the Demand Routing and Slotting Problem arises.

There are a few basic approaches to providing survivability in a network design that differ in the degree to which protection capacity is dedicated to protect particular demands. One method is to fully dedicate protection capacity for each specific demand. In this framework, two individual and non-intersecting routes are reserved for every point-to-point demand requiring protection.

The links in either route have enough capacity to satisfy the demand. The "back-up" path is not utilized unless a link or node in the "primary" path is disconnected. This method is popular because it is easy to implement, it requires fairly inexpensive node equipment, and it provides virtually immediate failure recovery. However, the resulting network has a large amount of under-utilized capacity, so this approach is best when the number of point-to-point demands requiring protection is small or when network capacity is cheap.

In an alternative approach, the back-up capacity is not dedicated to any particular demands, rather it is shared by all of them. Extra link capacity is placed throughout the network and special switching equipment is placed at key nodes so that, within seconds, traffic may be redirected to avoid a broken or congested link or node location. The equipment necessary to implement this approach is more sophisticated and expensive, but the associated network designs are not only survivable, but are also robust to "demand uncertainty". That is, the network can satisfy the demand for connections that were not originally anticipated in demand forecasts. For some network services, this feature is as important as survivability because of the difficulty in obtaining reliable forecasts. The problems of determining appropriate link capacities and finding optimal locations for the switching equipment to implement this approach are among the most challenging of those encountered by the authors. (See [San94] [BCM94] [SDC94] for examples.)

In a third approach, the demand locations are partitioned into sets called "communities" and protection capacity is dedicated to each community. That is, the demands within each community share protection capacity allocated to the community as a whole. A community would likely be a set of nodes that are highly inter-connected and have a relatively large number of point-to-point demands among them. A typical community would contain between 3 and 16 nodes. Once an appropriate community is established, protection can be assured by finding a simple cycle or "ring" that connects the nodes of the community and then placing enough capacity to satisfy and protect the demands within the community. Topologically, a ring is a simple cycle passing through each node of the community. An example is given in Figure 1. Rings offer survivability because every pair of nodes on the ring has exactly two routes between them so that the network remains connected if there is a failure. Whenever a broken link or node is detected, the equipment on the ring redirects traffic to avoid the failure.

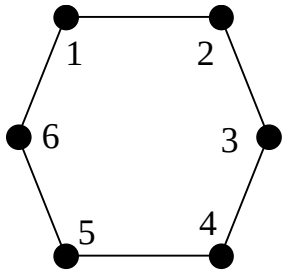

Figure 1: 6 node ring

An important technology in fiber-optic networks, called "the Synchronous Optical NETwork" or "SONET", gives rise to special equipment that are capable of effectively performing the switching tasks required for these types of protection in less than one tenth of a second. In [CDSW95] the authors describe the SONET Toolkit, which is a decision support system for providing survivability in SONET networks that was developed at Bellcore. The system includes various mechanisms for providing survivability, but it relies heavily on ring deployment for cost-effectiveness. The ring sizing problem is one particular subproblem that arises in the ring selection and deployment component of this tool.

3. Notation and Problem Formulations

The variant of ring sizing that we consider is special multi-commodity flow problem that we call the "Demand Routing and Slotting Problem" [DRSP]. We now describe DRSP in detail. Our task is to allocate capacity to serve the individual demands assigned to a ring with the ultimate goal of serving them on the cheapest possible ring. One of the implications of using SONET equipment is that all of the links on the ring must have the same capacity, which we refer to as the capacity of the ring. Not surprisingly, larger rings cost more, so building the cheapest ring boils down to determining the minimum capacity ring that satisfies the demands. In practice, rings are deployed in small number of discrete sizes, and we would select the smallest one that meets or exceeds the minimum capacity requirement to afford survivability. Thus, it is actually this minimum capacity requirement that DRSP seeks to determine.

DRSP represents the operation of SONET equipment with constraints that govern the way point-to-point demands utilize ring capacity. One such constraint, that we refer to as the “slotting constraint”, requires that each unit of demand stay in the same "slot" throughout its path on the ring. That is, if the units of capacity are numbered, 1, 2, 3,..., then a "slot" is defined to be the

unit of ring capacity associated with a specific number. This corresponds to terms like "channel" or "band" in radio or microwave transmission, or a "lane" in a highway or racetrack. Certain equipment can only keep track of a particular demand if it does not change slots within its route. Such equipment is said to require “time-slot” assignment of demands. In contrast, equipment that allows movement of demand between slots is said to have time-slot interchange capability. In particular, we note that the add-drop multiplexers commonly deployed in rings do, in fact, require that each unit of demand be assigned to a single slot along its entire route. Failure to model this slot assignment may lead to selection of an infeasible ring size.

A second constraint that is often imposed by telecommunication carriers dictates that demands cannot be "split" in their routing. That is, they must be routed entirely clockwise or counter-clockwise on the ring. The “Ring Loading Problem”, studied by [CS94] and [SSW95], is the variant of the ring sizing problem that forbids the splitting of demands but does not impose slot assignments. The DRSP formulation that we now state includes both types of constraints.

3.1. Notation

Consider a ring containing nodes labeled 1,2,...,n (proceeding clockwise), and edge set E = {(1,2), (2,3),..., (n-1,n), (n,1)}. Suppose that every edge in E has the same capacity “budget”, C. (The value of C must be at least as large as the minimum number of slots required, but is not necessarily equal to that value.) It is possible to number the capacity units in each link 1,2,...,C, where slot c is the particular unit of ring capacity labeled with the number c. The ring is to be used to satisfy K different “point-to-point demands” between the nodes. Each demand comes with a pair of distinct end-points, node i and node j ($i<j$), and a demand quantity d. A pair of nodes may have any number of distinct point-to-point demands between them. For each demand k, we define $E^+(k)$ to be the set of edges visited if the demand is routed in the clockwise direction. $E^-(k)$ is the complementary set of edges associated with the counter-clockwise route. For instance, if demand k is between node i and node j, then $E^+(k)$ is {(i, i+1), (i+1,i+2)…(j-1,j)} and $E^-(k)$ is {(j,j+1),..., (n,1),...(i-1,i)}. Note that the clockwise routing never uses edge (n,1), while the counter-clockwise routing always uses it.

For each edge e on the ring we define $K^+(e)$ to be the set of demands $\{k \mid e \in E^+(k)\}$, i.e., the set of demands that would pass edge e if routed clockwise. Similarly, $K^-(e)$ is the set of demands that would pass edge e if routed in the counter-clockwise direction. Further, we assume that a demand does not route over its endpoints in either the clockwise or counter-clockwise routing.

Notice that removing any two edges in the cycle cuts the network into two components. Let $D(e,f)$ be the set of demands whose endpoints are separated by the cut formed by edges e and f. Let $T(e,f)$ equal the total amount of demand in $D(e,f)$. Finally, let e^*, f^* denote the edges in a cut that separates the maximum total demand and let $T^*=T(e^*,f^*)$.

3.2. DRSP Formulation

The objective in the Demand Routing and Slotting Problem is to route all of the point-to-point demands on the ring while utilizing as few slots as possible. The constraints are that each demand must be routed entirely in one direction and that no demand can change slots within its route. The decision variables for the problem are defined as follows: Let y_k be a variable whose value is 1 if demand k is routed clockwise, or 0 if it is routed counter-clockwise. Let z_c be a variable set to 1 if slot c on the ring is utilized, or 0 if it is not. Let $x^+(k,c)$ be a variable set to 1 if any unit of demand k is routed clockwise using slot c, or 0 if none are. Finally, let $x^-(k,c)$ be a variable set to 1 if any unit of demand k is routed counter-clockwise using slot c, or 0 if none are.

Formulated as a variant of an integer multi-commodity flow problem, DRSP can be stated as follows:

$$\text{Minimize } C^* = \sum_c z_c$$

subject to:

$$\sum_c x^+(k,c) = d_k\, y_k \qquad k=1,\ldots,K$$

$$\sum_c x^-(k,c) = d_k\, (1-y_k) \qquad k=1,\ldots,K$$

$$\sum_{k\in K^+(e)} x^+(k,c) + \sum_{k\in K^-(e)} x^-(k,c) \le z_c \qquad e=1,\ldots,n;\ c=1,\ldots,C$$

$$x^+(k,c),\ x^-(k,c),\ z_c,\ y_k \in \{0,1\}$$

The first two constraints coupled with the integrality constraints state that, for each demand k, d_k slots are utilized and that all of the demand units are routed in either the clockwise or counter-clockwise direction. The third constraint states that the demand units assigned to a particular slot on the ring cannot overlap at any edge. We point out that there are a number of alternative formulations for DRSP; we selected this particular one for its simplicity. Alternative formulations may include additional constraints that strengthen the formulation for linear programming-based solution methods, but these issues are not addressed in this paper.

DRSP is a computationally difficult problem. A proof of the following result mimics the one in [CS94].

Theorem 1: DRSP is NP-hard.

Theorem 1, combined with the fact that the ring sizing component of a planning tool like the SONET Toolkit [CDSW95] may be called thousands of times during a network planning session, makes heuristic methods a natural (and warranted) way to proceed.

Several variations of the DRSP model may arise because of differing assumptions about the capabilities of equipment being represented or because of different requirements on the solution. The most realistic variants are still computationally difficult. The "Ring Loading" problem is one such variant that relaxes the slotting constraints but still requires each demand to be routed entirely in one direction. Ring Loading is described in greater detail in Section 5.4. One polynomially solvable variant of DRSP may be viewed as a relaxation of Ring Loading in which the demands are allowed to split into integral parts to utilize both the clockwise and counter-clockwise paths. This case can be modeled as an instance of Ring Loading with unit-sized demands. For instance, if the demand between a pair of nodes is equal to d, then d individual unit-sized demands can be represented in the formulation. It turns out that this variant is equivalent to an integer multi-commodity flow problem on a ring network which was shown to be polynomially solvable for a more general class of networks that includes rings in [Fra85]. The specific case of ring networks is described in [FNSST], [VSKW96], and [SSW95].

3.3. The Linear Relaxation of DRSP

If we relax both the slotting constraints and the one-way-routing constraints, then our constraints must simply ensure that the demands are routed. This problem is equivalent to a continuous variable multi-commodity flow problem on a ring network. It allows for any fraction of the demand units to be routed in the clockwise direction while the remainder must be routed in the counter-clockwise direction. This problem is a linear program that can be used to provide lower bounds on the optimal value of DRSP and can also be used to suggest feasible solutions for DRSP, which we discuss in Section 5.3. This linear relaxation of DRSP can be formulated as follows: let x_k represent the amount of demand k (possibly fractional) routed in the clockwise direction. The total capacity required for the ring to be capable of satisfying the demands is the solution to the linear relaxation problem [LRP] stated as follows:

Minimize z

subject to:

$$\sum_{k \in K^+(e)} x_k + \sum_{k \in K^-(e)} (d_k - x_k) \leq z \qquad e = 1,\ldots,n$$

$$0 \leq x_k \leq d_k$$

Since this is a linear program, we know that it is polynomially solvable. However, solving LRP using standard LP techniques does not take advantage of the problem structure afforded by routing in such a simple network. Specialized polynomial-time algorithms for this relaxation of DRSP have been proposed in [VSKW96] and [SSW95]. Both of these approaches make use of results presented in [OS81] for a more general class of networks. The results presented in [OS81] imply the following properties of the solution to the linear program.

LRP Properties: The optimal solution to LRP has the following properties:

i) The optimal objective value, z^*, is equal to 1/2 T^*.

ii) The demands d_k where $k \notin D(e^*,f^*)$ are not split. Further, they are routed in the direction that avoids both edge e^* and edge f^*.

That is, the cut with the largest demand across it determines the optimal value of the linear program. Clearly, this value, z^*, provides a lower bound to the required capacity for the more

restrictive versions of DRSP. This bound on the objective of DRSP is the tightest possible, because there is an instance for which the bound is achieved. Consider the case where there is a single demand unit between node *i* and node *i*+1, for *i* = 1 ... *n*-1 and between node *n* and node 1. (An example where *n*=6 is shown in Figure 2.) Clearly the optimal solution to DRSP is 1 because the demands can be satisfied using a single slot. The value of T* is 2 because T(*e*,*f*) is 2 for any pair of edges *e* and *f*, so the optimal solution to the linear relaxation, *z**, is 1.

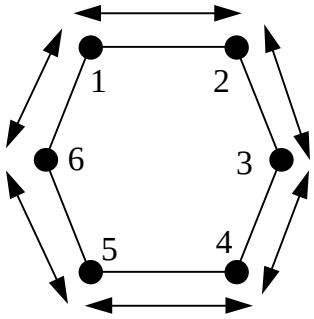

Figure 2: 6 demands that occupy one slot

3.4. DRSP Components

An instance of DRSP may be viewed as having two components. The "Routing" component is to determine, for each of the K demands, whether to route in the clockwise or counter-clockwise direction. The "Slotting" component is to find an assignment of each unit of the routed demands to individual slots so that the total number of slots used is minimized. Clearly, there are 2^K possible solutions to the Routing Problem. The objective of DRSP is to select a particular routing that minimizes the number of slots required in the associated slotting problem.

These components are not separable in an optimal approach to DRSP, but since each has been studied previously, results about each of their solutions can be incorporated into an intuitively appealing heuristic approach to DRSP. Such an approach would be a "two-phased" method in which a routing is determined in the first phase and a slotting based on this particular routing is determined in the second. In the next two sections, we discuss the components of routing and slotting, and we discuss the quality of resulting heuristics for DRSP.

4. The Slotting Problem

We begin our discussion with the Slotting Problem and assume that some solution to the Routing Problem is provided. That is, each demand is provided with a direction on the ring. For any such solution, the objective of the Slotting Problem is to place the individual demand units into as few slots as possible, while avoiding "collisions". If two or more demand units are assigned to the same slot, they cannot overlap at any edge.

Observation 1: The Slotting Problem is equivalent to optimally coloring the vertices of an associated circular arc graph.

We demonstrate this connection by noting that a given routing of demands around a ring may be viewed as a collection of arcs on a circle. The graph associated with overlaps in such a collection of arcs is called a "circular arc graph" (CAG) and is constructed as follows: Associate a vertex in the CAG with every unit of routed demand (i.e., circular arc) and place a link in the CAG between any two vertices whose associated routings overlap at any edge. Figure 3 provides an example of a demand routing and its associated CAG.

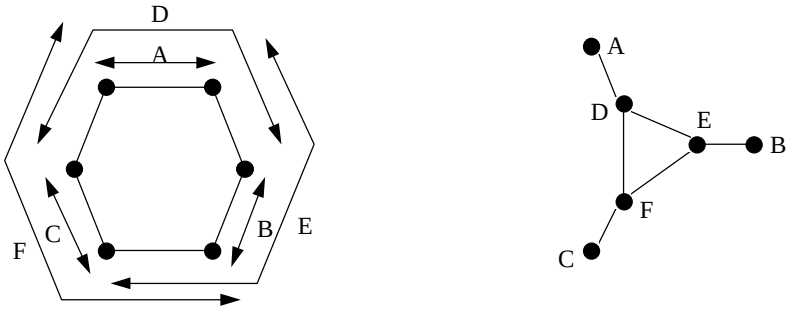

Figure 3: A demand routing and its associated CAG.

A vertex coloring of a graph is any assignment of colors to vertices such that no two adjacent vertices are assigned the same color. Since nodes in the CAG correspond to demand units, and two nodes in the CAG are adjacent precisely when their associated demands overlap, we see that a feasible coloring in the CAG is equivalent to a feasible slotting for the demands. Thus, by finding the minimum coloring in the associated CAG, we find the optimal slotting. The problem of finding the chromatic number of an arbitrary CAG, and hence, the minimum number of slots for an arbitrary set of routed demands, is NP-hard [GJMP].

However, a particular subset of circular arc graphs, called "interval graphs", can be optimally colored in linear time. An interval graph is the overlap graph associated with a set of intervals along a line. A CAG in which there is at least one point on the circle that is not covered by an arc is an interval graph. For our problem, an interval graph is formed whenever there is a link or node that has no demand routed across it.

An interval graph can be optimally colored using a simple greedy algorithm that proceeds along the range from left to right and assigns the first available color to each new interval encountered. The chromatic number of an interval graph is equal to the maximum overlap at any point along the interval, ([Gol80]). If we define the "load" on an edge of the ring to be the number of demand units routed over it, the following can be said for the Slotting Problem:

Lemma 1: The optimal number of slots, when the routing is equivalent to an interval graph, is equal to the maximum of the loads experienced at the edges of the ring.

The simplicity of interval graph coloring is exploited in a well-known heuristic for coloring arbitrary circular arc graphs proposed in [Tuck75]. The routine, adapted for the Slotting Problem, is as follows:

Slotting Heuristic:

Step 1: Find the node *i* having the least overlapping demand. Place each of the overlapping demand units into an empty slot.

Step 2: The remaining demand units form a set that corresponds to an interval graph. These demands can be slotted by using the linear time interval graph coloring scheme that proceeds around the ring (in either direction) starting at node *i* and assigns the first available slot to each demand unit encountered.

For the case where the routing of demands corresponds to an interval graph, this heuristic provides an optimal coloring. In the more general cases, the number of slots used by this heuristic is within twice the optimal number ([Tuck75]).

5. The Routing Problem.

We now address the problem of determining a routing for a given set of demands. Once a routing is established, the associated Slotting Problem can be solved (possibly heuristically) to provide a feasible solution to DRSP, and hence, an upper bound on its optimal objective value. In this section, we present several methods for constructing routings and then we apply the slotting algorithms described in the previous section in order to obtain DRSP solutions with provable quality bounds.

5.1. Edge Avoidance Routing

We present the routing heuristics in order of increasing complexity. The simplest routing approach is to route every demand in the same direction. Notice that if every demand were routed in the clockwise direction, then edge $(n,1)$ would have no load on it. That is, the edge is "avoided" by the routing. Thus, the routing forms an interval graph which can be optimally slotted to provide a feasible solution to DRSP. This routing method can be slightly generalized by routing to avoid an arbitrary edge, instead of just $(n,1)$. We call the resulting routings "edge-avoidance" routings and note that they share the following property:

Lemma 2: In a routing that avoids edge f, the load induced on edge e ($e \neq f$) is equal to $T(e,f)$.

Proof: Let $L(e)$ denote the load on edge e. Note that $L(e) \geq T(e,f)$ because every demand in $D(e,f)$ must, by definition, cross the cut involving e and f. Since none cross edge f, they must all route over edge e. Now note that no demand that is not in $D(e,f)$ is routed over either e or f in this routing. To see this, observe that both endpoints of a demand that is not in $D(e,f)$ lie on the same side of the cut. Thus, if such a demand were routed over edge e, it would also have to be routed back over edge f, which is not possible. Thus, the demands routed on e are precisely those in $D(e,f)$ so $L(e) = T(e,f)$. ■

Lemma 3: The routing that avoids edge f induces a load of no more than $2z^*$ on any edge.

Proof: $T(e,f) \leq T^* = 2z^*$, for all e, so the result follows immediately from Lemma 2. ■

Lemma 4: For any edge f, the routing that avoids edge f can be slotted using no more than $2z^*$ slots.

Proof: Since the routing that avoids f forms an interval graph, Lemmas 1 and 3 provide the result. ■

By combining edge-avoidance routing with an algorithm for interval graph coloring, we can see that it's easy to construct feasible solutions to DRSP that require no more than 2z* slots. These solutions provide a bound on the objective of DRSP.

Theorem 2: The optimal objective to DRSP, C*, is bounded by T* = 2 z*.

Proof: Consider routing all demands in a clockwise direction so that edge (n,1) is avoided. This is an edge-avoidance routing, which Lemma 4 assures can be slotted using no more than 2z* slots. Since there is a feasible solution whose value is no more than 2z*, we have that C* ≤ 2z*.■

We point out that the bound on the objective of DRSP provided by Theorem 2 is the tightest possible, because there is an instance for which C* = T*. Consider the case where the number of nodes is even, i.e., n = 2m. Suppose there is a single demand between nodes i and i+m, for i=1,...,m, as illustrated in Figure 4 when m=3.

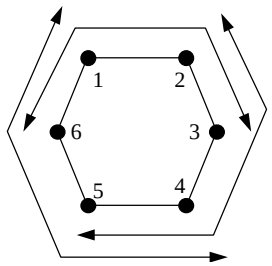

Figure 4: Demands that require separate slots

Notice that each of the m demands overlaps with every other demand so that m slots are required to prevent collisions, regardless of the routing. Since the value of T* in this case is also equal to m, the bound is tight. Note that this implies that in assessing the performance of DRSP heuristics with respect to z*, 2z* will be the best performance guarantee we can hope to achieve. The edge-avoidance heuristic provides a very simple method for obtaining DRSP solutions that are within this bound. We can be somewhat more precise by noting that a ρ -approximation algorithm for DRSP is a polynomial-time algorithm guaranteed to deliver a solution that requires at most ρ times the optimal number of slots. Since determining an edge-avoidance routing and coloring the resulting interval graph can be done in polynomial time, and since the resulting solution is guaranteed to use T* ≤ 2C* slots, combining the two algorithms yields a 2-approximation

algorithm for DRSP. Furthermore, the example in Figure 2 demonstrates that ρ cannot be reduced from 2 because the edge-avoidance routing requires two slots, when the optimal requires just one.

In a practical setting, we might want to apply the edge-avoidance routing method several times in succession in the hope of finding progressively better solutions. Since the number of ring nodes is relatively small, that suggests the following simple iterative heuristic for obtaining solutions to DRSP: calculate the number of slots required for routings that avoid each of the edges in the ring and use the one that requires the fewest slots. Let S(*f*) denote the number of slots required by a routing that avoids edge *f*. Let $\overline{f}$ be the edge for which S($\cdot$) is minimized. Let *e* be an edge for which the load in the routing that avoids $\overline{f}$ is S($\overline{f}$). We know that S($\overline{f}$) = T(*e*, $\overline{f}$) ≤ T*. In some cases, the total number of slots required by a solution obtained in this fashion may be considerably smaller than T*. For instance, consider the following result.

Theorem 3: Given a routing R whose maximum link load is L and whose minimum link load is *l*, there is an edge-avoidance routing whose load is no more than L+*l*.

Proof: Suppose that R induces load *l* on edge *e*. If we reverse the direction of all demand routed over *e*, we now have a routing that avoids edge *e*. At worst, we have increased the load on every edge other than *e* by *l*. Therefore, we have constructed an edge avoidance routing whose load is at most L+*l*. ■

5.2. Weight-based Routing

Edge-avoidance routings are special cases of a more general class of "weight-based" routings that we consider now. In a weight-based routing method, each edge in the ring is assigned a non-negative weight. In a "positive-weight" routing, each of the edges is assigned a strictly positive weight. Given a set of weights, each demand is routed in the direction whose edges have minimum total weight. Ties can be broken according to some selection rule. We adopt the rule that ties are routed in the direction that contains the fewest edges, and if there is still a tie, it may be broken arbitrarily.

Notice that the "edge-avoidance" method is a special weight-based method that can be implemented by assigning a weight of 1 to the avoided edge and a weight of 0 to the remaining

edges. The popular "minimum-hop" routing (the routing that traverses the fewest edges) is a weight-based routing method that is effected by assigning a weight of $1/n$ to each edge.

Weight-based methods offer heuristics that are intuitively more appealing than the simpler edge-avoidance methods and they still offer a good performance bound. In the remainder of this subsection, we describe the properties of weight-based routings and their associated slottings. Our goal is to establish the result that the entire class of weight-based routings can be slotted using no more than $2z^*$ slots. We begin by introducing a property that, in fact, guarantees the result.

To begin, we will follow the description in [SSW95] and consider the demands geometrically, so that each can be viewed as a chord of a circle representing the ring. Given this representation, two demands are said to be either *parallel* or *crossing*. For instance, demands between *i* and *j* and *k* and *l* cross if their indices are all distinct and exactly one of *i* or *j* lies in (*k,l*), otherwise the demands are parallel. Figure 5 provides an example.

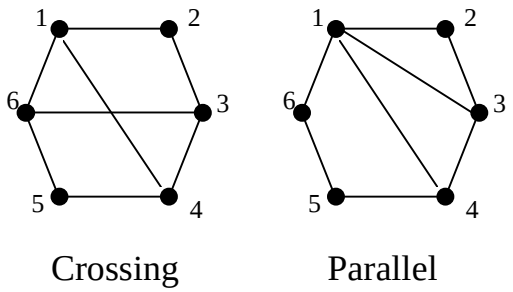

Figure 5: Example of crossing and parallel demands

An edge that lies between two parallel demands is said to be "between" them. In the example in Figure 5, (3,4) is between the two parallel demands.

Property 1: (Parallel Routing) We say that a routing R has the *parallel routing property* if there is no link that is between two parallel demands that carries load from both of them.

We will say that a routing that has the parallel routing property is a "parallel routing". The parallel routing property assures that the routing of any two demands can overlap in at most one continuous interval on the ring. To see this, note that crossing demands overlap at exactly one interval in their routing, regardless of which routes are selected. Parallel demands exhibit three

routing cases that are illustrated in Figure 6. They are: 1) the routings do not overlap; 2) one routing contains the other; or 3) the routings overlap only the ends. The parallel routing property assures that the third case does not occur. We state this more formally, in the following lemma.

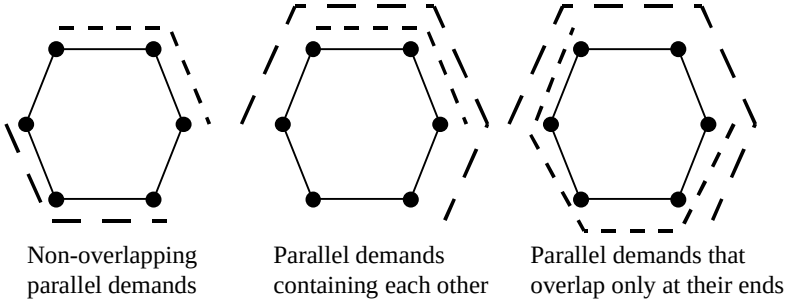

Figure 6: Examples of routing parallel demands.

Lemma 5: No two demands in a parallel routing overlap in two disjoint intervals. (Thus, the third case illustrated in Figure 6 cannot occur in a parallel routing.)

Proof:: First, note that crossing demands cannot overlap in more than one interval, so we need only consider parallel demands. Thus, assume that we have parallel demands (i,j) and (k,l), as illustrated in Figure 7. There may be links between the two demands on either side. For simplicity, we'll use label e to represent (possibly several) links between the demands on one side and f to represent links between the demands on the other side. (See Figure 7.) It may also be the case that either e or f is empty. Finally, let's assume that the demands do not have both endpoints in common (i.e. e and f are not both empty). If they have the same endpoints then they clearly overlap completely or not at all. Without loss of generality, we can further assume that node i is distinct from k and l and that it is 1, so that $i<j\leq k<l$. Now, there are only four possibilities to consider. If both demands route clockwise, they do not overlap. If one routes clockwise and the other routes counter-clockwise, then the counter-clockwise routed demand contains the clockwise one. If both demands route counter-clockwise, they would overlap both e and f, but these edges are between the two demands, so this cannot happen in a parallel routing unless both e and f are empty (in which case the endpoints for the two demands were the same and they do not overlap anyway). ■

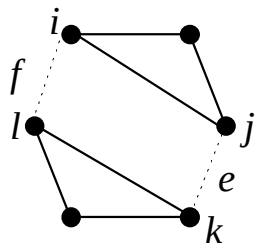

Figure 7: Two parallel demands in ring.

We can now show that a weight-based routing that applies the stated tie-breaking criterion is a parallel routing.

Theorem 4: A positive weight routing is a parallel routing.

Proof: To demonstrate that this is the case, assume the opposite so that we may have two parallel demands X and Y that each route over an edge that is between them in a positive-weight routing R. Let's let A denote the edges that are between the two demands on one side and let C denote the edges between them on the other side, such that at least one of A and C is non-empty. Finally, let B denote the remaining edges in the routing of X and let D denote the remaining edges in the routing of Y, as illustrated in Figure 8. The routing of X is made up of segments A, B, and C and the routing of Y is made up of segments A, D and C. If we let $w()$ denote a weight function so that $w(A)$ is the weight of segment A, then the weight of routing X is $w(X) = w(A) + w(B) + w(C)$. Since X is a weight-based routing, we know that

$$w(A) + w(B) + w(C) \leq w(D).$$

Similarly for Y, we know that

$$w(A) + w(D) + w(C) \leq w(B).$$

Together these two statements imply that

$$w(A)+w(C) \leq 0,$$

which contradicts the assumption that R is a positive weight routing. ■

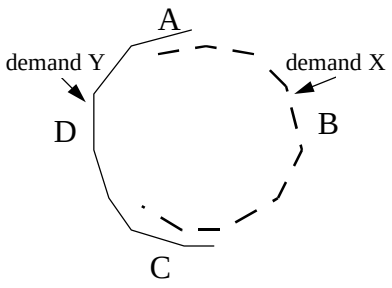

Figure 8

Corollary 1: A weight-based routing that applies the fewest hop criterion to break ties is a parallel routing.

Proof:: Using the notation from the previous theorem, the only way for parallel demands X and Y to both route over edges that are between them is for the weight on those edges to be zero. In such a case, both routings for both demands have equal weight. To break the ties, we apply the fewest hop criterion. By applying the fewest hop criterion, we are, in essence, routing all demands that "tied" under the original weight function according to a new positive-weight function, which guarantees parallel routing. ■

We will now demonstrate that parallel routings can be slotted using at most 2z* slots. In order to prove this result, we use a property of parallel routings that is assured by the following lemma.

Lemma 6: If R is a parallel routing, then for every edge e there exists a node *p* such that there is no demand routed over both e and *p*.

Proof: Without loss of generality, assume that *e* is (n,1), where n is the highest numbered node. Let X denote the demand that extends the farthest past *e* moving clockwise and let Y denote the demand that extends the farthest past *e* moving counter-clockwise. Finally, let p_2 denote the endpoint of X on the clockwise side of *e* and let p_1 denote the endpoint of Y on the counter-clockwise side of *e*. Thus, p_1 is the lowest numbered node encountered by moving counter-clockwise along a demand that overlaps *e* and p_2 is the highest numbered node encountered by moving clockwise along a demand that overlaps *e*. Figure 9 provides an example.

For the Lemma to be false, it must be the case that every node of the ring is overlapped by either X or Y or both. Assume that this is the case. For all nodes to be overlapped by at least one of X and Y, it must be true that $p_1 < p_2$. This means that X and Y overlap the interval (p_1, p_2). Since R is a parallel routing and X and Y also overlap (n,1) we know that the two intervals cannot be disjoint. That is, either X must overlap [p_2 ,n], or Y must overlap [1, p_1]. However, that would imply that either X or Y overlaps every link of the ring, which cannot occur because demands have distinct endpoints. ■

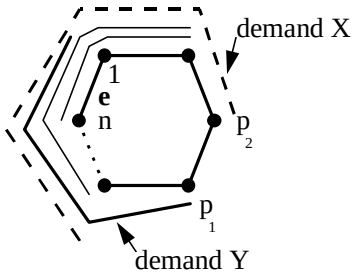

Figure 9

Lemma 6 provides that the demands that overlap edge e in a parallel routing form an interval graph that leaves at least one node uncovered. Note that when we reverse the direction of the demands that overlap *e*, they must overlap every point that they did not overlap in the original routing. In particular, they must all overlap a node *p* that is provided by Lemma 6. This property is now used to demonstrate our main result.

Theorem 5: If R is a parallel routing, then it can be slotted to provide a solution to DRSP that requires no more than T* = 2*z** slots.

Proof: Let e denote the edge that has the maximum load under routing R and let L denote this amount. Let l denote the minimum load induced by R at any point in the ring. Further let D(e) denote set of demands that route over edge e in R.

Notice that the routing that avoids e differs from R only in its routing of the demands in D(e). Thus, the routing that avoids *e* is effected by reversing the direction of the demands in D(e). When we do this, Lemma 6 provides that they will all overlap a common node *p*. Since the load

on p was at least l in routing R, we know that the load on p in the edge avoidance routing is *at least* L+l. Lemma 4 assures that the edge avoidance routing can be slotted to provide a solution to DRSP that requires at most 2z* slots, so L + $l \leq 2z$*.

To establish the theorem, note that the slotting heuristic provided in Section 4 requires no more than L + l slots for R. It uses l slots for the demands overlapping the point of least overlap and no more than L slots for the remaining demands. Since L+$l \leq 2z$*, we have the result. ■

Corollary 2: A weight-based routing (using an appropriate tie-breaking method) can be slotted using no more than 2z* slots.

Proof: The result follows directly from the fact that weight-based routings are parallel routings.■

The slotting that demonstrates the result of Theorem 5 is obtained using the algorithm given in the previous section [Tuck75]. This is clearly a polynomial-time algorithm for slotting. Since weight-based routing can be performed in linear time, the result of Corollary 2 assures that weight-based routing followed by the basic slotting algorithm provides a 2-approximation algorithm for DRSP. Weight-based routings include edge-avoidance routings, so we know that the factor of two cannot be reduced. In section 5.4, we demonstrate that this is also the case for positive-weight routings.

5.2.1. Real-time Routing and Slotting

Weight-based schemes, such as minimum-hop, are widely used in practice because they require no off-line computation. Once a weighting is established, the routing for each demand becomes fixed and is independent of the routing of other demands. Thus, weight-based routing is an "on-line" method for routing.

If we combine edge-avoidance routing with an on-line scheme for interval graph coloring [KT81], we have an on-line method for obtaining DRSP solutions. On-line algorithms for interval graph coloring can be guaranteed to deliver solutions that use no more than 3 times the optimal number of colors, which is best possible [KT81]. Since edge-avoidance guarantees that the resulting interval graph has a maximum load of no more than 2z*, the combined approach delivers a solution to DRSP that uses no more than 6z* slots.

We can also combine more general weight-based routings with an on-line algorithm for circular arc graph coloring [MHR93] to obtain a second on-line approach for DRSP. The CAG coloring algorithm [MHR93], however is only guaranteed to deliver a solution that is within a factor of 4 of the optimal. This combined with the fact that we cannot guarantee that an optimal CAG coloring will require fewer than $2z^*$ colors implies that this method will have a worse theoretical performance than the simpler one described above. However, it may perform well in practice.

The important issue of real-time performance of algorithms for routing and slotting has recently begun to be addressed. Some initial simulation studies of real-time routing and slotting have been conducted and described by Wilson [Wil95].

5.3. Routing based on the Linear Relaxation Problem

In the context of SONET Ring technology, various approaches to demand routing have been investigated [CS94], [SSW95], [VSKW96]. Each of them solves some relaxation of DRSP that does not include slotting, but the solutions obtained by these methods may still be of use in constructing DRSP solutions. In the remainder of this section, we consider two such methods. The first is based on solving the linear relaxation problem (LRP) which relaxes both splitting and slotting constraints. The second is based on solving the Loading problem which relaxes only the slotting constraints. Unlike the previous routing methods whose route selections are independent of the particular demands populating the ring, these methods tailor routing choices based upon the demands provided. That is to say that these methods may choose to route a particular point-to-point demand either clockwise or counter-clockwise depending on the other demands that need to be routed.

Since the Linear Relaxation problem LRP is a specially structured linear program that is relatively easy to solve, it is reasonable to utilize its solution to suggest routings for at least some subset of the demands in heuristic approaches to DRSP. Recall that in the solution to the relaxation, any demand d_k where $k \notin D(e^*,f^*)$ is routed in the direction that avoids both edge e^* and edge f^*. The remaining demands may or may not be split between the two directions. We can build a routing from this one by maintaining the route chosen by LRP for any demand that does not split and "unsplitting" the remaining demands. One way to do this would be to simply

route each demand entirely in the direction that contains the largest portion of the LRP routing. (In case of a tie, we could choose either direction.)

It turns out that we don't really need to worry about the particular unsplitting method that we use if our goal is simply to get a quality bound on DRSP solutions derived from the LRP routing. Both [SSW95] and [VKSW96] provide polynomial-time algorithms that solve LRP and split very few demands. (They split at most $n/2$ of them.) In addition, [SSW95] shows that the routing they obtain exhibits the parallel routing property. As a result, *any* unsplitting of their solution is also a parallel routing. We can see this by observing that an unsplitting introduces no new routes. It simply selects a subset of those used in the given solution. Therefore, if the given solution is a parallel routing, any unsplitting of it must be as well. Since the algorithm in [SSW95] enables us to compute a parallel routing that solves LRP, the following theorem is an immediate consequence of Theorem 5.

Theorem 6: There exists a solution to LRP that can be unsplit arbitrarily and then slotted using no more than $2z^* = T^*$ slots.

The [SSW95] algorithm for routing combined with arbitrary unsplitting and the Slotting algorithm in Section 4 provides a polynomial-time approach for obtaining a feasible DRSP solution based on the linear relaxation. Since Theorem 6 assures that these solutions require no more than twice the optimal number of slots, the combined approach is a 2-approximation algorithm for DRSP.

5.4. Routing based on the Loading Problem

Finally, we consider building solutions for DRSP from those of the "Ring Loading" problem. The Ring Loading problem arises when we assume that the equipment placed at the nodes of the ring is capable of performing "slot interchanges". In such cases, it is not necessary that a demand stay in the same slot within its route, so the slotting constraints can be removed. The Ring Loading problem refers to this version of DRSP in which the slotting constraints are removed but splitting is still forbidden. Since a particular routing of the demands puts a load on each edge that is equal to the number of demand units passing through it, the objective in the Loading problem is to determine the routing that minimizes the maximum of the loads experienced at the

edges. The Loading problem is described in detail in [CS94] and [SSW95]. It is NP-Hard, but some reasonable heuristics are available. In particular, [SSW95] offers an approach that assures a solution whose load is no more than $z^* + 3/2$ D, where D is the amount of the largest demand, and [CS94] describes a "dual-ascent" heuristic that seems to perform well in practice.

The Loading problem can be used in a heuristic approach to DRSP because one can assume that a routing of demands that minimizes link loads would be likely to require a relatively small number of slots as well. Indeed, it is certainly true that optimal link load is a lower bound on the number of slots required. In the absence of a more "integrated" approach to DRSP, this method is intuitively appealing. The approach seeks a routing that induces a small link load and then slots based on this routing. Implicit in this approach is the assumption that link load provides a good surrogate for the number of slots required. While it seems intuitively reasonable that such a method would perform better than a simple-minded scheme like edge-avoidance, theoretically its performance is no better. In the remainder of this section, we examine the theoretical performance of methods that select a routing based on minimizing link load. For the sake of analysis, we ignore the fact that the Ring Loading problem is NP-hard and assume that we can determine a routing that minimizes the maximum link load.

Theorem 7: Let R* be a routing that minimizes the maximum link load and let L* be the optimal load. R* can be slotted using no more than 2L* slots.

Proof: Assign slots to R* using the Slotting algorithm in Section 4. So, let *p* be the point of least overlap on the ring. We know that the overlap at *p* is no more than L*, so the algorithm assigns the demands that overlap *p* to L* slots. The remaining demands form an interval graph whose maximum overlap is no more than L*, so these demands can be assigned to L* slots by interval graph coloring. ■

The optimal slotting would be at least as good as the heuristic one obtained in the proof of Theorem 7, so clearly $C^* \leq 2\ L^*$. Since $2\ L^* \geq 2\ z^* = T^*$, the performance bound that we demonstrate for this approach is no better than that of the considerably simpler "edge-avoidance" heuristic. While it may be possible to improve this bound to $2z^*$, the routing that minimizes load is not necessarily a parallel routing, so we cannot employ Theorem 5 to do it. Figure provides a small example in which the minimum load routing is not a parallel routing.

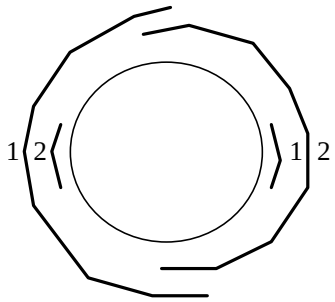

Figure 10: Optimal load routing that is not a parallel routing

Finally, we can show that even if there were polynomial-time algorithms for solving the Ring Loading problem and then coloring the resulting circular arc graph, the combined approach could be no better than a 2-approximation algorithm, i.e., no better than edge-avoidance. This is because examples exist for which the routing associated with the optimal loading requires relatively many slots. We now illustrate the point.

Observation 2: Routing optimally with respect to load followed by slotting optimally may yield a solution to DRSP that uses twice the optimal number of slots.

We demonstrate this claim with an example. Let m be an odd number and consider a ring with $n = 2m$ nodes. The following demands must be routed in the ring: 1) two individual unit-sized demands between node i and node $i+m$, for $i=1..m$; and 2) a demand for two units between node i and node $i+1$, for all odd i.

Let L(R) denote the load induced by routing R and let R* be a routing that minimizes load. Since the linear program LRP is also a relaxation of the Loading problem, the following bound applies: $L(R^*) \geq z^* = T^*/2 = (2m+2)/2 = m+1$. Thus, the optimal load is at least $m+1$.

Consider the following routing of demands. Each of the demands between node i and node $i+1$ is routed in the clockwise direction along edge $(i,i+1)$. Both units of the unit-sized demands are routed in the direction that overlaps the fewest "short" demands. (An example with $m=3$ is illustrated in Figure 11.) The maximum load associated with this routing is $m+1$, so it is optimal to the Loading problem. (Moreover, it is the unique optimal.)

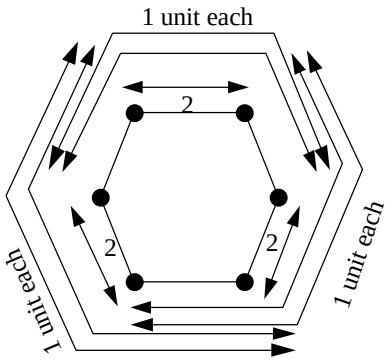

Figure 11: Demand routing that yields optimal load.

In the solution to the associated Slotting Problem, each of the unit-sized demands must be placed in its own slot, otherwise it would collide with another unit-sized demand, so the number of slots associated with this routing must be at least 2*m*.

A different feasible solution to DRSP routes one of the unit demands between node *i* and node *i*+*m* in the clockwise direction and the other unit in the counter-clockwise direction. The remaining demands are routed across edge (*i*, *i*+1). (See the illustration in Figure 12.) The load associated with this routing is *m*+2, i.e., not optimal to the Loading problem, but with this routing the two unit-sized demands between a pair of nodes can share the same slot, so the total number of slots required is *m*+2. The ratio between the two solutions is 2*m*/*m*+2 which approaches 2 as m increases, so the performance bound is asymptotically tight. This illustrates that any method based on the optimal load routing can be no better than a 2-approximation algorithm. ■

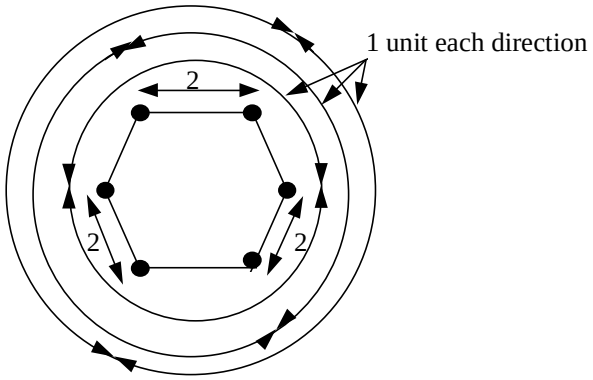

Figure 12: Routing that requires m+2 slots.

Finally, in section 5.2 we showed that weight-based routing combined with the standard slotting algorithm provides a 2-approximation method for solving DRSP. We showed that the factor of 2

cannot be improved for non-negative weights. This example enables us to verify that this is also the case for positive-weight routings. By assigning weight w to even numbered edges and weight $2w$ to odd numbered edges (where $w = 1/3m$), the weight-based routing for this example is the same as the optimal load routing that requires $2m$ slots. Thus, the performance bound is also tight for positive-weight routings.

6. Concluding Remarks

The introduction to this paper describes some general approaches used to protect the demands in a communications network. A “divide and conquer” scheme to address this complex and difficult problem gives rise to a number of subproblems, some of which are themselves quite complex. DRSP represents one such problem. The remainder of the paper describes this new problem in detail and presents some basic heuristic approaches to solve it. Further study of the problem will likely yield more sophisticated and effective heuristics for its solution. In an approach to solve a typical network design problem, there are a large number of potential rings to consider. A "Route and Slot" heuristic, like any of the ones described in this document, may be called as a subroutine hundreds, or even thousands, of times. So, we recommend restricting attention to relatively simple heuristics that find their solutions in a short amount of time. Since z^*, the optimal objective value for the linear relaxation, and T^*, the maximum total demand across a cut, are the factors that contribute to the tightest bounds on C^*, the optimal objective value of DRSP, it is reasonable to use these values to establish performance bounds on the feasible solutions obtained by such methods.

Recall that the true objective of the Routing Problem is to give rise to a circular-arc graph with the smallest possible chromatic number, hence the smallest number of required slots. The routing methods whose objective is to minimize the maximum load on the edges have been suggested because they results in graphs with what is hoped to be a small chromatic number. If there is an alternative method for quickly anticipating the chromatic number of an arbitrary circular-arc graph or a better surrogate such as the maximum clique size, an alternative objective

to the Routing Problem might be appropriate. There is reason to believe that this could provide superior solutions to DRSP.

7. Acknowledgments

The authors wish to thank Brian Wilson, Peter Winkler, Tom Trotter, and Nate Dean for many helpful discussions on various topics related to rings. The first author thanks Leslie Hall for pointing out reference [FNSST], for sparking the example we present in Section 5.4 and for many fun and elucidating conversations about rings.

8. References

[Bab90] J. Babcock, "SONET: A Practical Perspective", *Business Communications Review*, September 1990, pp. 59-63.

[BCM94] D. Bai, T. Carpenter, J. Mulvey, “Stochastic Programming to Promote Network Survivability”, *Technical Report* SOR-94-14, Department of Civil Engineering and Operations Research, Princeton University, 1994.

[CDSW95] S. Cosares, D. Deutsch. I. Saniee, O. Wasem, "SONET Toolkit: A Decision Support System for the Design of Robust and Cost-Effective Fiber-Optic Networks", *Interfaces* 25, 1995.

[CS94] S. Cosares and I. Saniee, "An Optimization Problem Related to Balancing Loads on SONET Rings", *Telecom. Systems* 3, 1994.

[CCS94] S. Cosares, T. Carpenter, and I. Saniee, “Static Routing and Slotting of Demand in SONET Rings”, presented at the TIMS/ORSA Joint National Meeting, Boston, MA, 1994.

[Fra85] A. Frank, “Edge-disjoint Paths in Planar Graphs”, *Journal of Combinatorial Theory*, Series B 39, pp. 164-178, 1985.

[FNSST] A. Frank, T. Nishizeki, N. Saito, H. Suzuki, E. Tardos, “Algorithms for Routing Around a Rectangle”, *Discrete Applied Mathematics* 40, pp. 363-378, 1992.

[GJMP80] M.R. Garey, D.S. Johnson, G.L. Miller, C.H. Papadimitriou, "The Complexity of Coloring Circular arcs and Chords", *SIAM J. Alg. Disc. Meth.* 1, pp. 216-227, 1980.

[Gol80] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, Inc, San Diego, Ca., 1980.

[KT81] H. Kierstead and W. T. Trotter, “An Extremal Problem in Recursive Combinatorics”, Congressus Numeratum, 33, pp. 143-153, 1981.

[MHR93] M. V. Marathe, H. B. Hunt, and S. S. Ravi, “Efficient Approximation Algorithms for Domatic Partition and On-line Coloring of Circular Arc Graphs”, Technical Report, Dept. of Computer Science, SUNY Albany, Albany, NY, 1993.

[NYT] “Much of East Coast Phone Service is Disrupted by Jersey Cable Break”, *New York Times*, November 19, 1988, p. 1, and “AT&T Acting to Thwart New Mass Disruption”, *New York Times*, November 20, 1988, p. 37.

[OS81] H. Okamura and P. Seymour, “Multicommodity Flows in Planar Graphs”, *Journal of Combinatorial Theory*, Series B 31, pp. 75-81, 1981.

[Rit90] G.R. Ritchie, "SONET Lays the Roadbed for Broadband Networks", *Networking Management* 1990.

[San94] I. Saniee, “Optimal routing in self-healing communications networks”, *International Transactions in Operations Research*, 3, pp. 187-195, 1996.

[SSW95] A. Schrijver, P. Seymour, and P. Winkler, “The Ring Loading Problem”, to appear in *SIAM J. Disc. Math*.

[VSKW96] R. Vachani, A. Shulman, P. Kubat, J. Ward, “Multicommodity Flows in Ring Networks”, *INFORMS Journal on Computing*, 8, pp. 235-242, 1996.

[SDC94] S. Sen, R. Doverspike, S. Cosares, "Network Planning with Random Demand", *Telecom. Systems*, 3, 1994.

[Tuck75] A. C. Tucker, “Coloring a Family of Circular Arc Graphs”, *SIAM Journal Appl. Math* 29, pp. 493-502, 1975.

[Wil95] B. J. Wilson, “Managing Connections in SONET Bi-directional Rings”, presented at the Third INFORMS Telecommunications Conference, Boca Raton, Florida, 1995.