SIAM J. DISCRETE MATH.
Vol. 11, No. 1, pp. 1-14, February 1998

001

THE RING LOADING PROBLEM*

ALEXANDER SCHRIJVER†, PAUL SEYMOUR‡, AND PETER WINKLER§

Abstract. The following problem arose in the planning of optical communications networks which use bidirectional SONET rings. Traffic demands $d_{i,j}$ are given for each pair of nodes in an n-node ring; each demand must be routed one of the two possible ways around the ring. The object is to minimize the maximum load on the cycle, where the load of an edge is the sum of the demands routed through that edge.

We provide a fast, simple algorithm which achieves a load that is guaranteed to exceed the optimum by at most 3/2 times the maximum demand, and that performs even better in practice. En route we prove the following curious lemma: for any $x_1, \ldots, x_n \in [0,1]$ there exist $y_1, \ldots, y_n$ such that for each k, $|y_k| = x_k$ and

$$\left|\sum_{i=1}^{k} y_i - \sum_{i=k+1}^{n} y_i\right| \leq 2.$$

Key words. SONET ring, load balancing, optical network design

AMS subject classifications. 90B10, 90C10, 94C15

PII. S0895480195294994

1. Introduction.

Around the world, billions of dollars are being spent by telephone operating companies to replace copper circuits with optical fiber, vastly increasing potential bandwidth and opening the network to multiple data-types—including video. The dominant technological standard in the United States is the Synchronous Optical NETwork (SONET) [1]. In one very popular configuration called a SONET ring, nodes (typically telephone central offices) are connected by a ring of fiber, with each node sending, receiving, and relaying messages by means of a device called an add-drop multiplexer (ADM).

SONET rings enjoy several advantages over other network configurations. The vertex-symmetry of the rings ensures that nodes play the same role and are similarly equipped, and the connectivity of the cycle protects against failure of either a link (that is, an edge) or a node. Thus, a major task of network-planning software, including Bellcore's SONET Toolkit™ [2], is to identify groups of nodes which can be turned into SONET rings in such a way as to satisfy traffic demands in a cost-efficient manner.

The capacity of a SONET ring varies from ring to ring but is the same for each link of a ring, and the cost of a ring (all other factors being equal) is an increasing function of its capacity. It is not the fiber itself but the ADMs which limit bandwidth. However, the effect is the same: for each SONET ring there is a capacity C such that no link of the ring may carry more than C units of traffic.

In some SONET rings all traffic is routed clockwise (unless a fault has occurred) and the capacity is selected so as to handle the sum of all the point-to-point demands

* Received by the editors October 6, 1995; accepted for publication (in revised form) October 9, 1996. The majority of the research described in this paper was conducted while the second and third authors were employed by Bellcore; the first was a visiting consultant.

http://www.siam.org/journals/sidma/11-1/29499.html

† Mathematics Center, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands (lex@cwi.nl).

‡ Department of Mathematics, Princeton University, Princeton, NJ 08544 (pds@math.princeton.edu).

§ Bell Labs 2C-379, 700 Mountain Avenue, Murray Hill, NJ 07974 (pw@lucent.com).

between nodes of the ring. Such "unidirectional" SONET rings will not concern us here.

In bidirectional rings, however, a routing is chosen independently for each pair of nodes, and all traffic between those nodes (in either direction) is sent by that route. Clearly, bidirectional rings are much more bandwidth-efficient; for example, when demands are uniform they can carry four times the traffic of a unidirectional ring having the same capacity.

In order to compute the capacity required for a proposed bidirectional SONET ring, the planning software must route the projected traffic demands in such a way as to minimize, or at least approximately minimize, the maximum load on any link. The problem is described formally below. We remark that the actual capacity selected for a proposed ring is further adjusted to allow for failures and abnormal demands, and that there is a discrete set of standard capacities from which to choose; but these considerations do not change the objective.

2. Notation and terminology. The problem is formally stated as follows:

RING LOADING

INSTANCE: Ring size n and nonnegative integers $d_{i,j}$, $1 \le i < j \le n$.
QUESTION: Find a map $\phi : \{(i,j) : 1 \le i < j \le n\} \longrightarrow \{0,1\}$ which minimizes $L = \max_{1 \le k \le n} L_k$, where

$$L_k = \sum \{d_{i,j} : \phi(i,j) = 1 \text{ and } k \in [i,j)\} + \sum \{d_{i,j} : \phi(i,j) = 0 \text{ and } k \notin [i,j)\}.$$

The notation "$[i,j)$" is used here for the half-closed integer interval $\{i, i+1, \ldots, j-1\}$.

To make RING LOADING a decision problem as in [7], we append a target value T to the instance and ask whether there is a ϕ for which $L \le T$.

Each $d_{i,j}$ is called a *demand*, and the map ϕ is called a *routing*. Setting $\phi(i,j) = 0$ amounts to routing the traffic between nodes i and j the "back" way, that is, through the link $\{n, 1\}$. When $\phi(i,j) = 1$ we say that the (i,j)th demand has been routed through the "front."

The routing induces a *load* L_i on each link $\{i, i+1\}$, namely, the sum of the demands routed through that link. The largest load is the *ringload* L, the quantity to be minimized.

3. Theory and reality. The decision form of RING LOADING is clearly in the class NP since the routing provides a witness which is only $\binom{n}{2}$ bits long. In fact, RING LOADING is an integer multicommodity flow problem (the reader is referred to [4] for a survey on such problems); in general such problems are NP-complete, but we are dealing with a very special case.

Technically, the input size for an instance of RING LOADING is slightly more than

$$\lceil \log n \rceil + \lceil \log T \rceil + \sum_{1 \le i < j \le n} \lceil \log d_{i,j} \rceil,$$

relative to which RING LOADING is NP-complete. A simple reduction is available from the PARTITION problem [7, p. 223], in which positive integers $a_1, \ldots, a_m$ are given and the question is whether one can divide them into two groups of equal sum. Put $n = m+3$, $d_{i,m+2} = a_i$ for $1 \le i \le m$, and $d_{m+1,m+2} = d_{m+2,m+3} = \sum a_i/2$. Set all other demands equal to zero, and let $T = \sum a_i$. Then a good routing must send

$d_{m+1,m+2}$ and $d_{m+2,m+3}$ the short way (front) and must partition the other demands so that $L_{m+1} = L_{m+2} = T$. This solves PARTITION, and vice versa.

An even easier reduction—with just two nodes—was given by Cosares and Saniee [3] and was made possible by their slightly more general RING LOADING formalization in which more than one demand per node pair is allowed. (The positive results to follow are also easily extended to cover the more general formulation; we prefer the more restrictive version for notational reasons.)

However, the reduction from PARTITION says nothing about the tractability of RING LOADING in practice, because PARTITION is solvable in time polynomial in $m \cdot \max a_i$ and actual demand sizes for RING LOADING are not large numbers. In fact, traffic demands are estimates to begin with, and the range 0 to 100 units is typically adequate. Thus, we may even take the maximum demand D to be bounded by a reasonable *constant*. The size n of a SONET ring is currently restricted to about 20. With these parameters, an instance of PARTITION can be solved using dynamic programming by hand!

Modest as the parameters are, however, they do not permit exhaustive search of the $2^{\binom{n}{2}}$ possible routings, and the PARTITION-to-RING LOADING reduction does not appear to permit reversal. As far as we know, any of the following three statements may be true (see [7] for descriptions of CLIQUE and CHROMATIC NUMBER):

- RING LOADING (like PARTITION) can be solved in time polynomial in n and D.
- RING LOADING (like CLIQUE) can be solved in time polynomial in n but only if a bound on the maximum demand D is fixed.
- RING LOADING (like CHROMATIC NUMBER) is NP-complete even for (some) fixed D.

Mercifully, the $D = 1$ case is solvable in time polynomial in n. The proof is due to Frank [5] and is explained nicely in [6]; it relies on a theorem of Okamura and Seymour [9]. This case is important because in some cases demands *can* be split, but only at integral values, and can thus be regarded as a multiplicity of unit demands. In fact, as we shall demonstrate, our approximation algorithm for RING LOADING actually solves this case exactly.

We do not have a fast exact algorithm, either in theory or in practice, for the RING LOADING problem with $D>1$. Fortunately, in practice, a reasonable approximate solution to RING LOADING was acceptable. There was no room for compromise on the issue of computation time: the RING LOADING problem had to be solved in a matter of seconds at most, because it was part of a frequently called subroutine for determining the cost of *proposed* SONET rings. The full program considers enormous numbers of potential SONET rings and is supposed to work on run-of-the-mill serial computers.

To be precise, we sought an algorithm A with the following three properties, listed in order of importance:

1. A must be fast.
2. A should provide a solution to RING LOADING which exceeds the optimum load by no more than about 5% in most cases.
3. A should, if possible, come with a performance guarantee for both (1) and (2).

As it turns out, these properties were obtainable with a fairly simple algorithm whose efficiency does not much depend on D (the demands can be treated as real numbers).

4. Linear relaxation. The "relaxed" version of RING LOADING, in which demands may be split (that is, sent partly around the front, partly around the back), is formulated as follows:

RELAXED RING LOADING

INSTANCE: Ring size n and nonnegative integers $d_{i,j}$, $1 \le i < j \le n$.
QUESTION: Find a map $\phi^* : \{(i,j) : 1 \le i < j \le n\} \longrightarrow [0,1]$ which minimizes $L^* = \max_{1 \le k \le n} L_k^*$, where

$$L_k^* = \sum\{\phi^*(i,j)d_{i,j} : k \in [i,j)\} + \sum\{(1-\phi^*(i,j))d_{i,j} : k \notin [i,j)\}.$$

Since this is now a linear programming problem, it is solvable in polynomial time [8]. In fact, we shall see that a solution to RELAXED RING LOADING can be obtained in a very fast greedy fashion, even if we demand the additional property described in Proposition 4.1.

It is useful to think of demands geometrically as weighted chords in a circle representing the SONET ring. Two demands $d_{g,h}$ and $d_{i,j}$, with $g < h$ and $i < j$, are said to *cross* if all of the indices are distinct and if exactly one of i and j lies in (g,h); otherwise they are said to be *parallel*. In particular, demands such as $d_{i,j}$ and $d_{i,k}$, which share a node, are parallel.

A link which lies between two chords representing parallel demands is said to be "between" the demands. Finally, a routing ϕ^* for the RELAXED RING LOADING problem is said to *split* a demand $d_{i,j}$ if $0 < \phi^*(i,j) < 1$.

PROPOSITION 4.1. *Let ϕ^* be a routing for an instance of* RELAXED RING LOADING *which achieves the optimal load L^* and is also minimal, in the sense that no other routing has $L_i \le L_i^*$ for every i and $L_j < L_j^*$ for some j. Then no link which lies between two parallel demands will carry traffic from both demands.*

Proof. Assume otherwise, letting link $\{k, k+1\}$ carry a quantity a of traffic from demand $d_{g,h}$ and $b \ge a$ from $d_{i,j}$. After rerouting a quantity a of traffic from each demand so as to no longer pass through the kth link, no link suffers an increased load. This contradicts the minimality of ϕ^*. □

Proposition 4.1 fails for RING LOADING as can be seen from the example in Fig. 1, where $n = 8$ and the nonzero demands are $d_{2,3} = d_{1,4} = 1$ and $d_{6,7} = d_{5,8} = 2$. The optimal $\{0,1\}$-assignment sends both $d_{1,4}$ and $d_{5,8}$ the long way around the ring, achieving $L = 3$; no other assignment can do better than $L = 4$. What is significant, however, is that the proposition does hold in the case of $\{0,1\}$ demands.

We can turn RELAXED RING LOADING into a decision problem in a more general way than before. We append to the instance a capacity C_i for each link $\{i, i+1\}$ and ask whether there is a routing ϕ^* for which $L_i \le C_i$ for each i. In the following it will be useful to regard node labels as integers modulo n, so that, for example, the link $\{n, 1\}$ is also written $\{n, n+1\}$ and the half-open interval $(g,h]$ is interpreted as $\{g, g+1, \ldots, n-1, n, 1, 2, \ldots, h-1\}$ if $h < g$.

Each pair of links $\{g, g+1\}$, $\{h, h+1\}$, with $g < h$, constitutes a *cut* of capacity $C_g + C_h$ in the network. We may think of a cut as a chord connecting the midpoints of the links $\{g, g+1\}$ and $\{h, h+1\}$; if a demand $d_{i,j}$ crosses this chord, any routing will contribute load $d_{i,j}$ to the cut's two links. Thus, if the instance is solvable, then $D_{g,h} \le C_g + C_h$, where

$$D_{g,h} := \sum\{d_{i,j} : i \le g \text{ and } j \in (g,h], \text{ or } i \in (g,h] \text{ and } j > h\}$$

is the total traffic demand across the cut. The following converse is a special case of the Okamura–Seymour theorem [9]; we give a simple proof here.

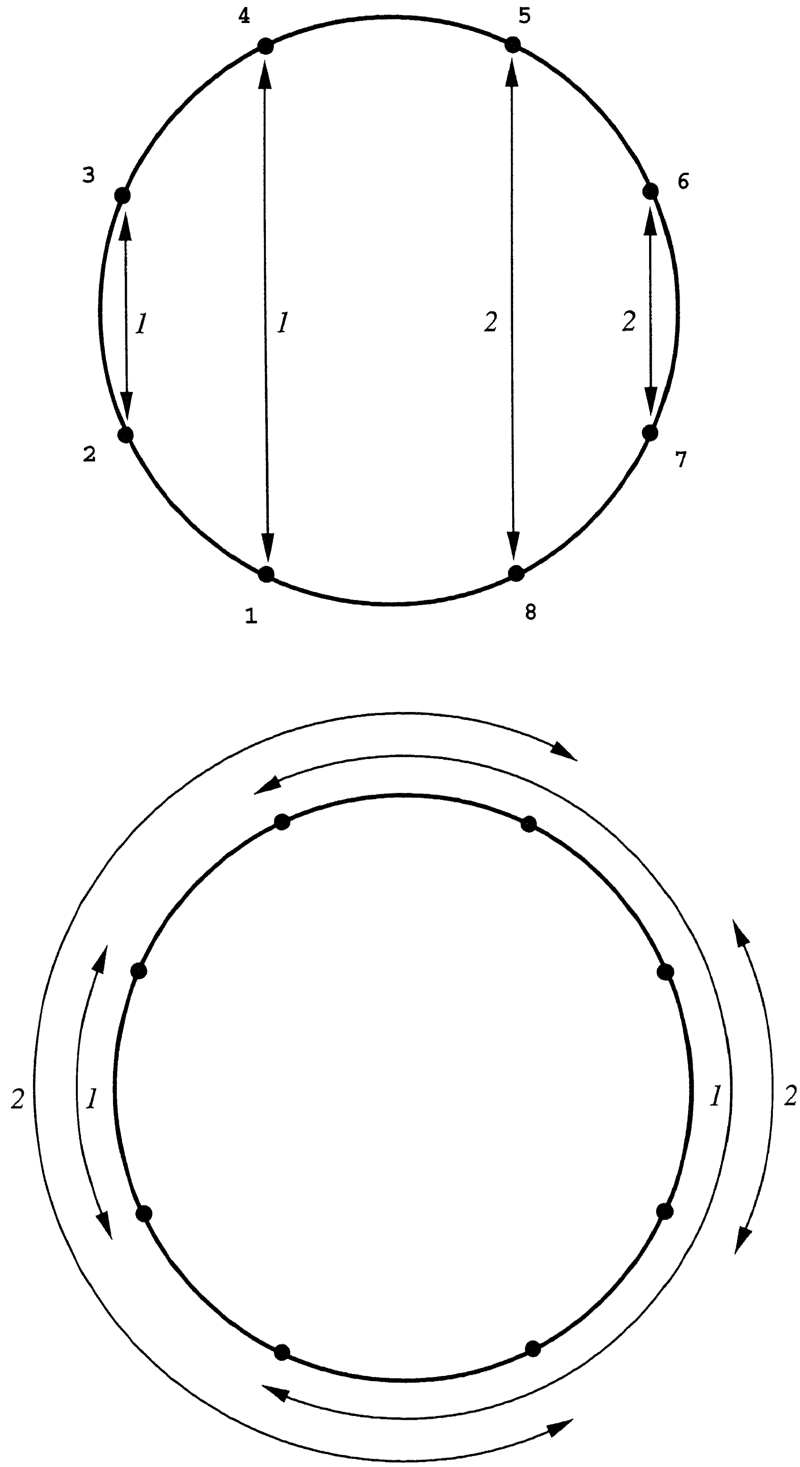


FIG. 1. *An instance of RING LOADING with optimal solution.*

PROPOSITION 4.2. *If $D_{g,h} \le C_g + C_h$ holds for each cut then there is a solution to* RELAXED RING LOADING *satisfying the capacity constraints.*

Proof. It will be useful in what follows to allow "cuts" of the form $\{g, g\}$, with capacity $2C_g$ and demand $D_{g,g} = 0$. The cut constraint for these cuts is thus equivalent to nonnegativity of the link capacities.

Assume the theorem fails and fix a counterexample with n minimal and, subject to the minimality of n, having the least possible number of nonzero demands.

Choose any nonzero demand—say, $d_{i,j}$—with $i < j$, and let $\{g, h\}$ minimize $M = D_{g,h} - C_g - C_h$ subject to $i \le g < h < j$; thus, $\{g, h\}$ is the tightest cut in the front route for $d_{i,j}$. (A cut $\{g, h\}$ is said to be "tight" if $D_{g,h} = C_g + C_h$.)

We propose to send $\min(d_{i,j}, M/2)$ of the demand $d_{i,j}$ around the front and, if $d_{i,j} > M/2$, send the remaining $d_{i,j} - M/2$ around the back. When the capacities have been decreased accordingly, we will have a new RELAXED RING LOADING instance with one less nonzero demand. If the new instance still satisfies the cut constraints, it will contradict minimality of the counterexample, proving the theorem.

Suppose that in the new instance some cut is violated. That cut must lie on the back route for $d_{i,j}$, since this demand has already been accounted for in cuts which it crosses, and cuts on the front route have sufficient slack by choice of M. Then we have a cut $\{g', h'\}$ with $[g', h') \cap [i, j) = \emptyset$ such that

$$D_{g',h'} + 2(d_{i,j} - M/2) > C_{g'} + C_{h'}$$

where all quantities are computed in the original instance.

Call the $\{g, h\}$ cut and the $\{g', h'\}$ cut "straight" and consider also the "diagonal" cuts $\{g, g'\}$ and $\{h, h'\}$. Every demand must cross at least as many of the two diagonal cuts as the two straight cuts, while $d_{i,j}$ crosses both diagonal cuts and neither straight cut. Hence,

$$\begin{aligned} &D_{g,g'} + D_{h,h'} \\ &\ge D_{g,h} + D_{g',h'} + 2d_{i,j} \\ &> C_g + C_h - 2(M/2) + C_{g'} + C_{h'} - 2(d_{i,j} - M/2) + 2d_{i,j} \\ &= C_g + C_{g'} + C_h + C_{h'} \end{aligned}$$

so that one of the diagonal cuts must have violated the cut constraint.

Note that nonviolation of cuts of the form $\{g, g\}$ assures us that the given routing of $d_{i,j}$ is actually possible, i.e., that no link capacity will become negative afterward. □

Given a set of demands, we now wish to find an assignment ϕ^* which minimizes L^* and satisfies the conclusion of Proposition 4.1. This can be done quickly by putting each link in a tight cut as follows.

First we compute the $\binom{n}{2}$ values $D_{g,h}$, $1 \le g < h \le n$; let the largest of these be M. Then $L^* \ge M/2$, but the ring with all capacities set to $M/2$ satisfies the cut constraint, so in fact $L^* = M/2$. We now take the links in any order (say, $\{1, 2\}$ through $\{n, 1\}$) and lower their capacities as much as possible; that is, define capacities $\{C_i\}$ recursively by

$$C_g = \max\left(\max h < g(D_{g,h} - C_h), \max h > g(D_{g,h} - M/2)\right),$$

noting that $C_g \ge 0$.

No realizable set $\{C'_i\}$ of capacities can have $C'_i \le C_i$ for every i and $C'_j < C_j$ for some j, since the least such j would be part of a bad cut. Hence any feasible assignment ϕ^* for these capacities is a minimal solution of the original RELAXED RING LOADING instance, and Proposition 4.1 applies. In particular, if $S = \{\{i,j\} : d_{i,j}$ is split by $\phi^*\}$, then every pair of chords in S crosses and, therefore, $|S| \le n/2$.

In fact, after reducing the capacities as above we can solve RELAXED RING LOADING to route each demand all front or all back until only mutually pairwise crossing demands remain. To see this, assume that there is still a parallel pair of unrouted demands and choose a link between them; fix a tight cut containing that link. At most one of the two parallel demands crosses the cut; the other can, and indeed must, be routed to miss the cut entirely.

In summary, our algorithm for solving RELAXED RING LOADING proceeds as follows:

1. Compute the $\binom{n}{2}$ values $D_{g,h}$, and $L^* = M/2$.
2. Compute minimal capacities $\{C_i\}$ as described above.
3. While there are pairs of parallel demands, find tightest cuts and route demands all front or all back, resetting capacities accordingly.
4. When only crossing cuts remain, route as much as possible by the front and the remainder by the back.

The running time of this procedure is approximately of order kn^2, where k is the number of nonzero demands; this is very fast for the parameter sizes that we require. See [10] for an even faster solution to problems akin to RELAXED RING LOADING.

In any case, our solution to RELAXED RING LOADING ends with at most only $n/2$ of the demands split. It therefore seems natural to compute ϕ^* and then "unsplit" the demands in S as gently as possible in order to get a near-optimal $\{0,1\}$ assignment for RING LOADING. This is exactly what we do.

5. Unsplitting. Henceforth ϕ^* will be a fixed, minimal solution to RELAXED RING LOADING with a set of split demands S as above. We seek a solution ϕ to RING LOADING which agrees with ϕ^* when $\phi^*(i,j) \in \{0,1\}$ and for which $L - L^*$ is as small as possible, where L is the ringload of ϕ.

If node i is not an endpoint of a split demand, then the difference between the loads on links $\{i-1,i\}$ and $\{i,i+1\}$ will not change as we pass from ϕ^* to ϕ. Hence, for the purpose of determining ϕ, we may as well delete vertex i and combine the two former links to form a single link whose load under the relaxed assignment is taken to be $\max(L^*_{i-1}, L^*_i)$. Proceeding in this fashion for each vertex not involved in a split demand, we are reduced to the case where n is even and $S = \{\{i, i+m\} : 1 \le i \le m\}$, with $m = n/2$.

Let us define u_i to be the amount of demand $d_{i,i+m}$ sent by ϕ^* via the front route, and v_i via the back, so that $u_i,\ v_i > 0$ and $u_i + v_i = d_{i,i+m}$. If ϕ routes $d_{i,i+m}$ by the front, then each link $\{j, j+1\}$ with $j \in [i, i+m)$ has its load incremented by v_i (the amount formerly sent around the back) relative to the relaxed assignment ϕ^*, while the rest of the link loads are decremented by v_i. Similarly, if demand $d_{i,i+m}$ is sent by the back route, the load of each link in $[i, i+m]$ is decreased by u_i while the rest are incremented by the same amount.

Hence if we set $z_i = v_i$ when $\phi(i, i+m) = 1$ and $z_i = -u_i$ otherwise, we have

$$L_j = L_j^* + \sum_{\substack{i\in[1,m]\\ j\in[i,i+m)}} z_i - \sum_{\substack{i\in[1,m]\\ j\in[i+m,i)}} z_i.$$

Notice that $L_j + L_{j+m} = L_j^* + L_{j+m}^*$ for all j. Thus $L \le 2L^*$ for *all* choices of ϕ, duplicating the performance ratio claimed by Cosares and Saniee [3], but we will do much better by choosing ϕ judiciously.

The optimal ϕ can be found by dynamic programming, but in practice we try every ϕ and choose the best one! There are at most $2^{n/2}$ choices for ϕ, a list easily exhausted for all currently contemplated SONET ring sizes. In effect, for our values of n (up to 32, possibly) the line between tractability and intractability lies not between polynomial and exponential but between exponential in n and exponential in n^2.

Our embarrassment, as theorists, is assuaged somewhat by the fact that there *is* a polynomial algorithm for finding an assignment ϕ which achieves the performance guaranteed by the following theorem.

THEOREM 5.1. *Let ϕ^* be a minimal solution, with ringload L^*, to the relaxed version of an instance of* RING LOADING. *Let D be the maximum magnitude of the demands split by ϕ^*. Then there is a $\{0,1\}$ assignment ϕ with ringload L which agrees with ϕ^*, except on split demands, and which satisfies $L - L^* \le \frac{3}{2}D$.*

Proof. We define z_i (hence ϕ) inductively, ensuring that $\sum_{i=1}^k z_i \in [-D/2, D/2]$ for all k, $1 \le k \le m$. This is always possible since, once $z_1, \ldots, z_{k-1}$ are defined and the partial sum $s = \sum_{i=1}^{k-1} z_i$ lies in the required interval, the two possible values of $\sum_{i=1}^k z_i$ lie on both sides of s and differ by only $u_k + v_k \le D$.

Put

$$M_k := \sum_{i=1}^k z_i - \sum_{i=k+1}^m z_i = 2\sum_{i=1}^k z_i - \sum_{i=1}^m z_i \in \left[-\frac{3}{2}D, \frac{3}{2}D\right]$$

and

$$M := \max_{1\le k\le m} |M_k|,$$

then

$$L - L^* \le \max_j (L_j - L_j^*) = M \le \frac{3}{2}D. \qquad \square$$

The greedy unsplitting given in the proof of Theorem 5.1, when appended to our solution to RELAXED RING LOADING, gives the polynomial-time approximation algorithm which we call "Algorithm A."

Of course, the true optimum L^{opt} for the original RING LOADING problem is at least equal to L^*, so the theorem guarantees an additive error of at most a constant $(3/2)$ times the maximum original demand irrespective of the value of n.

How good is this performance guarantee? This method can never achieve a multiplicative performance bound better than 2 relative to L^*, since the "square example" with $n = 4$, $d_{1,3} = d_{2,4} = 1$, and other demands 0 gives $L^* = 1$, $L^{\text{opt}} = 2$. Nor can we hope to get a factor better than 4/3 relative to L^{opt} due to the example shown in Fig. 1.

However, for larger n, if demands average $D/2$ in size then the typical demand adds $n/4 \cdot D/2$ to the total load when routed the short way; thus we expect the sum of the loads of all the links to be approximately $\binom{n}{2} \cdot n/4 \cdot D/2 \approx (D/16)n^3$, giving $L^* \ge (D/16)n^2$. Next to an optimum of order n^2, an additive error which does not depend on n at all looks pretty good; but we must again remember that n is never very large. For $n = 16$ this analysis allows a relative error of $(\frac{3}{2}D)/(16D) \approx 9\%$, which is not so impressive. Of course this is pessimistic; the Cosares-Saniee algorithm allows 100% error in theory but does far better in practice. In any case, it would clearly be worth some effort to determine whether the constant 3/2 is best possible, and we tackle this problem in the last section.

First, however, we return to the $\{0,1\}$ demands case.

6. {0,1} Demands.

In this section it will not complicate notation to allow many demands between two nodes of the ring, each of magnitude 1; we also allow capacities C_i for the links, not necessarily equal. A cut $\{g,h\}$ is said to be *even* if $C_g+C_h-D_{g,h} \equiv 0 \pmod 2$. In [6] feasibility is shown to be equivalent to the cut condition together with the following parity condition.

Parity condition. For every pair of links g, h, if g and h are each in a tight cut, then the cut $\{g,h\}$ is even.

THEOREM 6.1. *In the $\{0,1\}$ case, if we put $C_i = L$ for each link i, then* RING LOADING *is feasible with ringload $\leq L$ if and only if the cut and parity constraints are satisfied. If only the cut constraint is satisfied, then the optimal ringload is $L+1$. In any case, the algorithm A described above finds an optimal assignment.*

Proof. It is straightforward to verify that if a demand is assigned (all front or all back) without violating the cut condition, then the truth value of the parity condition is preserved. Since the parity condition is met when all demands are assigned, necessity is clear.

On the other hand, suppose that demands are assigned in accordance with Algorithm A until all remaining demands require splitting. Suppose there is at least one left, say, $d_{i,j}$; then there must be parallel cuts $\{g,h\}$ and $\{g',h'\}$ on each side of $d_{i,j}$ with $C_g + C_h - D_{g,h} = C_{g'} + C_{h'} - D_{g',h'} = 1$. Since the diagonal cuts must be tight, the parity condition is (twice) violated.

It remains only to observe that if the cut constraint is satisfied when $C_i = L$, then at $C_i = L+1$ we also satisfy the parity constraint since all the cuts have slack. □

7. The constant.

Let β be the infimum of all reals α such that the following combinatorial statement holds: For all positive integers m and nonnegative reals $u_1, \ldots, u_m$ and $v_1, \ldots, v_m$ with $u_i + v_i \leq 1$, there exist $z_1, \ldots, z_m$ such that for every k, $z_k \in \{v_k, -u_k\}$ and

$$\left| \sum_{i=1}^{k} z_i - \sum_{i=k+1}^{m} z_i \right| \leq \alpha.$$

Then β is the "right" constant for Theorem 5.1, i.e., $L - L^* \leq \beta D$ for some choice of ϕ. Note that any choice of rational values for the u_i's and v_i's can actually occur (up to constant factor) from an instance of RING LOADING, since we can construct one as follows. Let M_j be the load actually incurred by link $\{j, j+1\}$ when the demands $d_{i,i+m} = u_i + v_i$ are split u_i front and v_i back. Let M be huge, and postulate additional "short" demands $d_{j,j+1} = M - M_j$ for each j, $1 \leq j \leq 2m$. Then any optimal RELAXED RING LOADING solution will send all the short demands by the one-link route; however, the sum of the link loads due to the other demands is constant since each has two routes of the same length. Thus splitting the other demands as given, so as to obtain the same load M on every link, is optimal, and it is easy to see that no other splitting can achieve uniform load.

We already know $\beta \le 3/2$ and the square example, where $m = 2$ and $u_1 = v_1 = u_2 = v_2 = 1/2$, shows that $\beta \ge 1$. (In fact, u_i's and v_i's chosen uniformly at random subject to the given constraints also force $\beta \ge 1$.)

The special case where $u_i = v_i$ for each i is interesting for several reasons. This means that ϕ^* is sending exactly half of each demand $d_{i,i+m}$ each way around the ring, giving us no clue how to unsplit them. Furthermore, this is the case which arises when (as in the square example) all of the nonzero demands in the original RING LOADING instance are mutually crossing.

The case $u_i = v_i$ thus gives rise to a new ring loading problem as well as the following new constant.

CROSSED RING LOADING

INSTANCE: Ring size $2m$ and nonnegative reals d_i, $1 \le i \le m$.
QUESTION: Find a map $\phi : \{1, 2, \ldots, m\} \longrightarrow \{0, 1\}$ which minimizes $L = \max_{1 \le k \le 2m} L_k$, where

$$L_k = \sum\{\phi(i) d_i : \ k \in [i, i+m)\} + \sum\{(1 - \phi(i)) d_i : \ k \notin [i, i+m)\}.$$

Note that we have allowed real demands here (rationals would be fine, too) in order to handle nonintegral splits produced by a previous linear programming phase.

We define γ be the infimum of all reals α such that the following combinatorial statement holds: For all positive integers m and $x_1, \ldots, x_m \in [0,1]$ there exist $y_1, \ldots, y_m$ such that for every k, $|y_k| = x_k$ and

$$\left| \sum_{i=1}^{k} y_i - \sum_{i=k+1}^{m} y_i \right| \le 2\alpha.$$

(Note that we have rescaled the combinatorial statement so that the x_i's lie in the unit interval instead of $[0, 1/2]$.)

We have $1 \le \gamma \le \beta \le 3/2$. For lack of a counterexample, the authors were moved to conjecture publicly that both constants are equal to 1. After an embarrassingly long interval we found a simple proof, given below, that $\gamma = 1$; thus we have the following theorem.

THEOREM 7.1. *Let K be the sum of the demands of an instance of* CROSSED RING LOADING. *Then there is an assignment ϕ (which can be found in time polynomial in m and the length of the demand descriptions) whose ringload L satisfies $L - K/2 \le D$.*

Proof. We must show that given $x_1, \ldots, x_m \in [0,1]$ there are $y_1, \ldots, y_m$ such that for every k, $|y_k| = x_k$ and

$$\left| \sum_{i=1}^{k} y_i - \sum_{i=k+1}^{m} y_i \right| \le 2.$$

As in the asymmetric case, we can obtain a bound of 3 instead of 2 by greedy assignment; in this case that amounts to putting $y_k = x_k$ when $\sum_{i=1}^{k-1} y_i \le 0$ and $y_k = -x_k$ otherwise. We generalize this algorithm by choosing a real w instead of 0 as the "empty sum."

Specifically, for fixed $w \in [-1, 1]$, define y_k inductively by $y_k = x_k$ when $w + \sum_{i=1}^{k-1} y_i \le 0$ and $y_k = -x_k$ otherwise. Then $w + \sum_{i=1}^{k} y_i \in [-1, 1]$ for all k; let $f(w) := w + \sum_{i=1}^{m} y_i$.

Suppose that $f(w) = -w$; then

$$\begin{aligned}
&\sum_{i=1}^{k} y_i - \sum_{i=k+1}^{m} y_i \\
&= 2\sum_{i=1}^{k} y_i - \sum_{i=1}^{m} y_i \\
&= 2\left(w + \sum_{i=1}^{k} y_i\right) - \left(w + \sum_{i=1}^{m} y_i\right) - w \\
&\in [-2, 2]
\end{aligned}$$

as desired.

Since $f(-1) + (-1) \leq 0 \leq f(1) + 1$, the existence of a w for which $f(w) = -w$ would follow from the intermediate value theorem if f were continuous. Of course this is not the case; whenever a partial sum hits 0, some y_i's change sign and $f(w)$ may jump. (Since we have chosen y_i positive when the partial sum is 0, f will be continuous from the left.) However, it turns out that the *absolute value* of f is continuous.

Note first that when no partial sum is at 0, the derivative $f'(w)$ is 1. On the other hand suppose that $w = w_0$ is chosen such that one or more of the partial sums is zero; in particular, let $k \geq 0$ be minimal such that $w + \sum_{i=1}^{k} y_i = 0$. Then for sufficiently small ε, the signs of y_j and $w + \sum_{i=1}^{j} y_i$, for $j > k$, flip as we move from $w = w_0$ to $w = w_0 + \varepsilon$. Hence, taking $j = m$, we have that $\lim_{w \to w_0^+} f(w) = -f(w_0)$.

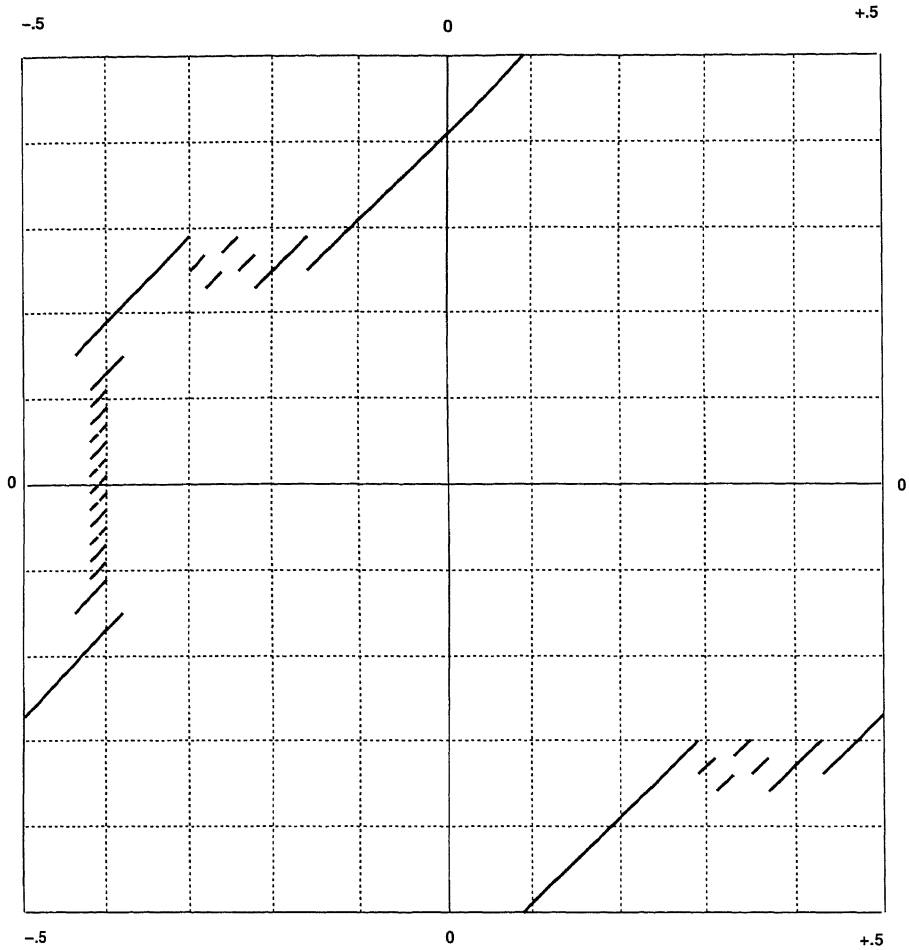
It follows that when any partial sum hits zero we will have $\lim_{w \to w_0^+} f(w) = -f(w_0)$; thus the function g given by $g(w) = |f(w)|$ will be continuous everywhere and differentiable except at finitely many points. The graph of g is a zig-zag, with derivative 1 where $g(w) = f(w)$ and -1 where $g(w) = -f(w)$.

Of course, if we define h by $h(w) = -w$, then the graph of h is a line of slope -1 from $(-1, 1)$ to $(1, -1)$ which must intersect the graph of g. Moreover, it must either intersect at a point where $g'(w) = 1$ or coincide with a segment of the graph of g of slope -1, in which case the leftmost point of the segment lies also on the graph of f. Either way we have a point w at which $-w = g(w) = f(w)$.

To complete the proof we need to demonstrate a fast algorithm for finding this w. To do this we set the y_i's one at a time while keeping a solution w in range. Specifically, at stage j we have values $y_1, \ldots, y_j$ fixed and $a_j \leq w \leq b_j$, with $g(a_j) \leq -a_j$ and $g(b_j) \geq -b_j$; of course this holds at stage 0 with $a_0 = -1$, $b_0 = 1$. At stage $j + 1$, if $a_i + \sum_{i=1}^{j} y_i$ and $b_i + \sum_{i=1}^{j} y_i$ are both positive, then perforce we set $y_{j+1} = -x_{j+1}$; if both are ≤ 0, then we put $y_{j+1} = x_{j+1}$. In either of these cases we set $a_{j+1} = a_j$ and $b_{j+1} = b_j$.

Otherwise $s := -\sum_{i=1}^{j} y_i$ lies in the half-open interval $[a_j, b_j)$. If $g(s) < -s$, we put $y_{j+1} = -x_{j+1}$ and set $a_{j+1} = s$ and $b_{j+1} = b_j$; if $g(s) \geq -s$, put $y_{j+1} = x_{j+1}$ and set $a_{j+1} = a_j$ and $b_{j+1} = s$. In any case the inductive conditions are preserved, the intervals $[a_j, b_j]$ are nested downward, and at stage m all the y_i's are correctly set. □

We have shown $\gamma = 1$, but the proof above will not work for β, as $g = |f|$ is no longer continuous in the asymmetric case. Even so, we may gain by replacing f by a multivalued function F, defined by $z \in F(w)$ if $z = w + \sum_{i=1}^{n} z_i$ for any $z_1, \ldots, z_n$ which keep the sums $w + \sum_{i=1}^{k} z_i$ within bounds.



$i=$	1	2	3	4	5	6	7	8	9	10	11	12	13
u_i	.17	.71	.25	.71	.23	.75	.76	.21	.75	.21	.73	.25	.50
v_i	.83	.23	.75	.21	.77	.21	.24	.73	.25	.71	.27	.71	.50

FIG. 2. *An example in which additive error of D cannot quite be achieved.*

Then the graph of F will be a union of slope-1 line segments, each corresponding to an assignment of z_i's. The sum of the lengths of these segments will be at least $\sqrt{2}$ since $F(w)$ always takes on at least one value, and in practice—and in virtually any random model—the segments will practically always intersect the line from $(-.5, .5)$ to $(.5, -.5)$ at least once, providing a solution to RING LOADING which is within D of L^*.

However, it is just barely possible to choose values u_i and v_i for which the diagonal line sneaks through between the line segments of the graph of F. A set of such values, for $m = 13$, is given in Fig. 2 along with the corresponding graph of F. On the graph, each of the 2^{13} routings is represented by a diagonal line segment, often null,

TABLE 1

n	k	C%	C-B	B=C	A-C	A=C	A-B	A=B	Bt	At	Ct
8	28	100%	.0054	63%	.0110	19.4%	.0160	10.7%	.0001	.0002	.002
12	66	99%	.0013	85%	.0036	21.2%	.0051	19.5%	.0004	.0005	.1
16	120	96%	.0003	94%	.0017	22.3%	.0023	20.7%	.0016	.0016	.45
20	190	93%	.00014	96%	.0010	26.2%	.0015	23.6%	.0036	.0038	.78
24	276	93%	.00002	99%	.0007	27.2%	.0008	24.8%	.007	.007	.84
28	378	92%	.00000	99%	.0004	28.3%	.00056	25.5%	.013	.012	.92
32	496	92%	.00000	99%	.0002	29.2%	.00037	26.4%	.02	.019	1.1

indicating the final sum $w + \sum_{i=1}^{m} z_i$ as a function of w, for just those values of w for which all partial sums $w + \sum_{i=1}^{k} z_i$ lie between $-.5$ and $.5$.

With this general definition of the multifunction F, a crossing of the diagonal is necessary as well as sufficient to get a solution within 2 of L^*. Hence the example shows that β is at least 1.01. This lower bound can certainly be raised somewhat but it is far from clear that the true value of β is anywhere near 3/2.

8. Conclusions. Experimental results show that indeed our proposed algorithm is adequately fast and, when applied to random examples small enough to compute L^{opt}, produces a ringload very close to optimal. We have never managed to produce a random example with $L > L^* + D$ even though our theorem guarantees only $L \leq L^* + \frac{3}{2}D$, and we doubt such an instance will ever be seen in practice.

Hence, even though the mathematics refuses to cooperate, *we* guarantee $L \leq L^* + D$.

Table 1 above exhibits the results of testing our algorithm, which we call "Algorithm A," on uniformly random data. Alongside "A" we ran a linear programming algorithm, "Algorithm B," in order to compute the lower bound given by the RELAXED RING LOADING solution. To find the optimum ringload and for purposes of comparison, we also tested "Algorithm C," which recursively looks for an optimal solution. In most cases Algorithm C was not enormously slower than A, but it became hopelessly stuck in some cases, leaving us with no value for the optimal ringload.

For each set of parameters, 1000 cases were run. The interpretation of the columns of the table is as follows:

n: number of nodes in the ring,
k: number of demands,
C%: percentage of runs in which the optimum was found,
C-B: average error of LP bound relative to optimum,
B=C: percentage of runs in which LP bound = optimum,
A-C: average error of our algorithm relative to optimum,
A=C: percentage of cases in which A hit the optimum,
A-B: average error of LP bound relative to A,
A=B: percentage of cases in which A achieves LP bound,
Bt: average running time for the LP algorithm,
At: average running time for Algorithm A,
Ct: average running time for Algorithm C.

The fourth through seventh columns are computed only for those rounds in which the optimum was found; that creates a bias, especially for the column labelled B=C, since we will probably never get equality when Algorithm C fails. The run time for Algorithm C includes cases where it failed to find the optimum, and it was terminated after 10 seconds of CPU time on any one run.

Acknowledgments. The authors have had the benefit of valuable conversations with Noga Alon and Milena Mihail.

REFERENCES

[1] J. BABCOCK, *SONET: A practical perspective*, Business Comm. Rev., Sept. (1990), pp. 59–63.

[2] S. COSARES, I. SANIEE, AND O. WASEM, *Network planning with the SONET toolkit*, Bellcore EXCHANGE, Sept./Oct. (1992), pp. 8–13.

[3] S. COSARES AND I. SANIEE, *An optimization problem related to balancing loads on SONET rings*, Telecommunications Systems, 3 (1994), pp. 165–181.

[4] A. FRANK, *Connectivity and network flows*, in Handbook of Combinatorics, R. Graham, M. Grötschel, and L. Lovász, eds., Elsevier, Amsterdam, 1995, pp. 111–177.

[5] A. FRANK, *Edge-disjoint paths in planar graphs*, J. Combin. Theory Ser. B, 38 (1985), pp. 164–178.

[6] A. FRANK, T. NISHIZEKI, N. SAITO, H. SUZUKI, AND E. TARDOS, *Algorithms for routing around a rectangle*, Discrete Appl. Math., 40 (1992), pp. 363–378.

[7] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, 1979.

[8] L. G. KHACHIYAN, *A polynomial algorithm in linear programming*, Soviet Math. Dokl., 20 (1979), pp. 191–194.

[9] H. OKAMURA AND P. D. SEYMOUR, *Multicommodity flows in planar graphs*, J. Combin. Theory Ser. B, 31 (1981), pp. 75–81.

[10] H. RIPPHAUSEN-LIPA, D. WAGNER, AND K. WEIHE, *Efficient algorithms for disjoint paths in planar graphs*, in Combinatorial Optimization, DIMACS Series in Discrete Math. Theoret. Comput. Sci. 20, W. Cook, L. Lovász, and P. Seymour, eds., AMS, Providence, RI, 1995, pp. 295–354.