Path Decomposition under a New Cost Measure with Applications to Optical Network Design*

Elliot Anshelevich[1]** and Lisa Zhang[2]

[1] Department of Computer Science, Cornell University, Ithaca NY.
eanshel@cs.cornell.edu.
[2] Bell Laboratories, 600 Mountain Avenue, Murray Hill NJ.
ylz@research.bell-labs.com.

Abstract. We introduce a problem directly inspired by its application to DWDM (*dense wavelength division multiplexing*) network design. We are given a set of demands to be carried over a network. Our goal is to choose a route for each demand and to decompose the network into a collection of edge-disjoint simple paths. These paths are called *optical line systems*. The cost of routing one unit of demand is the number of line systems with which the demand route overlaps; our design objective is to minimize the total cost over all demands. This cost metric is motivated by the need to avoid O-E-O (*optical-electrical-optical*) conversions in optical transmission as well as to minimize the expense of the equipment necessary to carry the traffic.

For given line systems, it is easy to find the optimal demand routes. On the other hand, for given demand routes designing the optimal line systems can be NP-hard. We first present a 2-approximation for general network topologies. As optical networks often have low node degrees, we offer an algorithm that finds the optimal solution for the special case in which the node degree is at most 3. Our solution is based on a local greedy approach.

If neither demand routes nor line systems are fixed, the situation becomes much harder. Even for a restricted scenario on a 3-regular Hamiltonian network, no efficient algorithm can guarantee a constant approximation better than 2. For general topologies, we offer a simple algorithm with an $O(\log K)$- and an $O(\log n)$-approximation where K is the number of demands and n is the number of nodes. For rings, a common special topology, we offer a 3/2-approximation.

1 Introduction

The constantly changing technology continues to present algorithmic challenges in optical network design. For example, the SONET (*Synchronous Optical NETwork*) technology has motivated a rich body of combinatorial work on rings, e.g. [4, 10, 19, 20]; and the WDM (*wavelength division multiplexing*) technology has inspired numerous algorithms in coloring and wavelength assignment, e.g. [11, 12, 15, 17]. In this paper we introduce a novel problem directly motivated by its applications to DWDM (*dense WDM*) network design. At the essence of this problem, however, lies a fundamental graph theory question. Our problem involves partitioning a graph into edge-disjoint paths with certain properties while minimizing a natural, yet so far largely unstudied, cost function.

The state-of-the-art DWDM technology allows more than one hundred wavelengths to be multiplexed on a single fiber strand and allows signals to travel thousands of kilometers optically. One design methodology in building an optical network using DWDM technology is to partition the

* Submission to the Design and Analysis Track.
** Work partially done while visiting Bell Labs.

network into a collection of paths which are called *optical line systems*. Such path decompositions have many advantages. For example, the linear structure simplifies wavelength assignment [21], engineering constraints [13], and requires less sophisticated hardware components. Therefore *line-based* optical networks are quite popular in today's optical industry.

In a line-based network an optical signal is transmitted within the optical domain as long as it follows a line system. However, when it switches from one line system to another, an O-E-O (*optical-electrical-optical*) conversion takes place. Such conversions are slow, expensive, and defeat the advantages of optical transmission. Therefore, our first design objective is to avoid O-E-O conversions as much as possible.

Naturally, our second objective is to design networks of low cost. As is usual in the case of optical network design, we cannot control the physical layout of the network since optical fibers are already installed underground. We are given a pre-existing network of *dark fiber*, i.e. optical fiber without any hardware for sending, receiving, or converting signals. The hardware cost consists of two parts, the *base cost* and the *transport cost*. The base cost refers to the cost of the equipment necessary to form the line systems. An optical line system begins with an *end terminal*, followed by an alternating sequence of fibers and OADM (*optical add/drop multiplexers*) and terminates with another end terminal. Roughly speaking, an OADM is built upon two back-to-back end terminals. Instead of terminating an optical signal an OADM allows the signal to pass through within the optical domain. An OADM costs about twice as much as an end terminal [7]. As a result, independent of how the network is partitioned into line systems, the base cost stays more or less the same. We can therefore ignore the base cost in our minimization. The transport cost refers to the cost of transmitting signals, in particular the cost incurred by the equipment needed to perform O-E-O conversions. Once the signal is formatted in the optical form it is transmitted along a line system for free. The number of O-E-O converters for a signal is linearly proportional to the number of line systems that the signal travels through. Therefore, minimizing O-E-O conversions serves the dual purpose of network cost minimization and providing (nearly) all-optical transmission.

Our Model. Let us describe the model in more detail. We are given a dark fiber network and a set of demands to be carried over the network. This network is modeled by an undirected graph $G = (V, E)$. Demands are undirected as well. Each demand is described by a source node and a destination node and carries one wavelength (or one unit) of traffic. We can have duplicate demands with the same source-destination pair. Let K be the sum of all demands. There are two design components, i) partitioning the edge set E into a set of edge-disjoint line systems, and ii) routing each demand.

A routing path of a demand consists of one or more *transparent sections*, where each transparent section is the intersection of the routing path and a line system. It is worth pointing out here that if multiple units of demand switch from one line system to another, each demand unit requires its individual O-E-O converter. Therefore, our objective is minimizing the total number of transparent sections summed over all demands, which is equivalent to minimizing the network equipment cost and minimizing O-E-O conversions.

Figure 1 gives a dark fiber network and two of the many possible line system designs. Suppose one unit of demand travels from A to F and three units of demand travel from C to E. The only possible routing paths are $ABCDF$ and CDE respectively. The solution in the middle results in $1 \times 4 + 3 \times 2 = 10$ transparent sections and the solution on the right results in $1 \times 1 + 3 \times 2 = 7$ transparent sections only.

One complication in routing can arise under the following circumstances. When an optical signal travels between two nodes on a line system it stays within the optical domain only if it follows the line system path defined by the orientation of the OADMs. Otherwise, O-E-O conversions may take

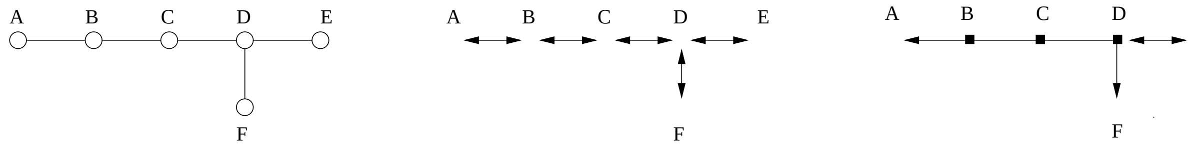

Fig. 1. (Left) A dark fiber network. (Middle) Each link is a separate line system. (Right) $ABCDF$ is a line system and DE is a line system. The arrows stand for end terminals and solid boxes stand for OADMs.

place. For example, the line system path in Figure 2 is $ABCFEDCG$. To travel between B and G the direct path BCG is expensive since it does not follow the line system and incurs an O-E-O conversion at node C; the "free" path $BCFEDCG$ is non-simple since it visits C twice. Naturally, network carriers avoid expensive demand routes. On the other hand, they also need simple demand routes for reasons such as easy network management. We therefore restrict ourselves to line systems that induce simple demand routes only and call such line systems *proper*. More precisely, a *proper* line system corresponds to a path in which each node appears at most once, excluding its possible appearance(s) as the end points.

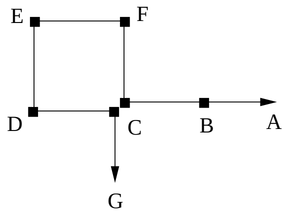

Fig. 2. An improper line system $ABCFEDCG$.

We also emphasize that wavelength assignment is not an issue in this context. A feasible wavelength assignment requires a demand to be on the same wavelength when it stays on one line system. However, it is free to switch to a different wavelength whenever it hops onto a different line system. Naturally, for wavelength division multiplexing, two demands sharing a link need to have distinct wavelengths. Given such constraints, as long as the number of demands per link respects the fiber capacity, i.e. the number of wavelengths per fiber, a feasible wavelength can be found efficiently using the simple method for interval graph coloring. (See e.g. [6].) In this paper we assume infinite fiber capacity, since the capacity of a single fiber strand is as large as one terabit per second. From now on, we focus on our objective of cost minimization without having to consider wavelengths.

Our Results. In this paper we begin with optimizing line system design assuming demand routes are given. (We note that for given line systems finding the optimal routing paths is trivial.) We then focus on the general case in which neither routes nor line systems are given. In the interest of space we include most of our proofs in the Appendix.

– In Section 2 we assume simple routing paths for all demands are given. For an arbitrary network with node degree higher than a certain constant c, finding the optimal line systems that minimize the total number of transparent sections is NP-hard. Fortunately, optical networks typically are sparse. Perhaps as our most interesting result, when the network has maximal node degree at most 3, we present an optimal solution based on an elegant greedy approach.

For an arbitrary network topology, we also present a 2-approximation algorithm. This 2-approximation is the best possible with the particular lower bound that we use.

- In Section 3 neither demand routes nor line systems are fixed. We first show that this general case is much harder. In fact, even for a very restricted case on a 3-regular Hamiltonian network, no algorithm can guarantee a constant approximation better than 2. We then present a simple algorithm that guarantees an $O(\log n)$- and $O(\log K)$-approximation for arbitrary networks, where n is the number of nodes in the network and K is the number of demands.
  In Section 3.2 we focus on the ring topology. Rings are of particular interest since often the underlying infrastructure is a ring for metro-area networks. We give a 3/2-approximation. If the total demand terminating at each node is bounded by a constant, we present an optimal solution.
- Finally, in Section 4 we discuss the relation of *supereulerian* graphs to our problem. A graph is supereulerian if there is a tour that visits every node but not necessarily every link. If we allow a line system to visit its nodes multiple times, then for a supereulerian network we can use this tour as a single spanning line system. For example, 4-edge-connected graphs are supereulerian which implies that they always have an optimal solution of cost K.

Related Work. Until now, the design of line-based networks has only been studied empirically, e.g. [5, 7]. However, the problem of ATM virtual path layout (VPL) is relevant. VPL aims to find a set of virtual paths in the network such that any demand route can be expressed as a concatenation of virtual paths, see e.g. [22, 2]. The model for VPL differs from ours in several aspects. For example, our line systems form a *partition* of the network links whereas virtual paths are allowed to *overlap*. Also, our demand routes can enter and depart from any point along a line system whereas demand routes for VPL have to be concatenations of *entire* virtual paths. This makes a huge difference in the model, since the extra constraint that all routes must be concatenations of whole line systems can increase the optimal solution by a factor of K. Finally, we are concerned with minimizing the *total* number of transparent sections, while VPL papers are traditionally concerned with minimizing the *maximum* hop count, with notable exceptions such as [9].

2 Designing Line Systems with Specified Routing

First we would like to point out that if line systems are given, it is easy to find the optimal route between any two nodes. The optimal route between any two nodes is the one that goes through as few line systems as possible and thus results in as few transparent sections as possible. We can find this route by calculating the shortest path according to an appropriate distance function. Let us refer to the optimal routes with respect to a set of line systems as the routing *induced* by the line systems.

If instead we are given the demand routes, and our goal is to find the optimal set of proper line systems, the problem becomes NP-hard. In fact, as we show in the Appendix, even for networks with bounded degree the problem remains NP-hard. Therefore, we consider approximation algorithms. Indeed, we present a 2-approximation algorithm for arbitrary network topologies.

2.1 A Lower Bound

We describe a *configuration* at each node, i.e. how the links incident to the node are connected to one another by line systems. For node u let $E(u)$ be the set of links incident to u. Each link $e \in E(u)$ is matched to at most one other link in $E(u)$. If e is matched to $f \in E(u)$ it means e and f are on the same line system connected by an OADM at u. If e is not matched to any link in $E(u)$ it means e terminates a line system at u. It is easy to see that the node configurations imply a set of paths that partition the network links. Of course, these resulting paths may not correspond to proper line systems.

Suppose every demand route is given. We offer a natural algorithm to configure each node. For every pair (u, v) and (v, w) of adjacent links, we define the *through traffic* $T(uvw)$ as the number of demand units routed along u, v and w. For each node v, we match the links incident to v such that the total through traffic along the matched link pairs is maximized. We refer to this algorithm as Max Thru. Although Max Thru may define closed loops or improper line systems, it has the following property.

Lemma 1. *For given demand routes, the total number of transparent sections generated by* Max Thru *is a lower bound on the cost of the optimal feasible solution.*

2.2 2-Approximation for Proper Line Systems

To find proper line systems given simple demand routes, we begin with the Max Thru algorithm and obtain a set of line systems that are not necessarily proper. If a line system is proper, we leave it as is. Otherwise, we cut the line system as follows. We traverse the line system from one end to the other and record every node that we visit in a *sequence*. If the line system is a closed loop we start from an arbitrary node and finish at the same node. (For example, the node sequence for the line system drawn in Figure 2 is $ABCFEDCG$.) If a node u appears multiple times in the node sequence, we mark the first appearance of u with an open parenthesis "(", the last appearance of u with a closed parenthesis ")", and every other appearance of u with a closed and an open parenthesis ")(". All these parentheses are labelled u. We *match* the ith open parenthesis labelled with u with the ith closed parenthesis also labelled with u. Each matching pair of parentheses represents a section of the line system that would induce non simple routing paths. We put down parentheses for every node that appears multiple times in the sequence.

We use these parentheses to determine where to cut an improper line system. We say two parentheses form an *inner-most pair* if the left parenthesis is "(", the right one is ")" and they do not contain any other parenthesis in between. Note the two parentheses in an inner-most pair do not have to have the same label. We now find the pair of inner-most parentheses that is also left most, and cut the sequence at the node v where the closed parenthesis from the selected pair sits. We remove every matching pair of parentheses that contains v. We repeat the above process until no parentheses are left. We refer to this algorithm as Cut Paren, which has 2 desirable properties as stated in Lemmas 2 and 3.

Lemma 2. *The* Cut Paren *algorithm defines a set of proper line systems.*

Lemma 3. *Given simple routing paths, each transparent section defined by* Max Thru *is cut into at most* 2 *pieces by* Cut Paren.

Proof. We first observe that a transparent section of a simple demand path cannot contain any matching pair of parentheses. Now consider any demand and let T be a transparent section of the demand defined by Max Thru. Suppose Cut Paren cuts T at node v. We claim that Cut Paren cannot cut T again. For the purpose of contradiction let us assume that T is cut again at v'. Let u' be where the matching open parenthesis sits. Note that u' and v' cannot contain v since this pair of matching parentheses at u' and v' would be removed after the sequence is first cut at v. Note also that v and v' cannot contain u' since otherwise T would contain the matching pair of parentheses at u' and v' which contradicts the fact that T is transparent. Therefore, Cut Paren cannot possibly cut T at v' after cutting it at v.

By Lemma 2 Cut Paren defines a set of proper line systems. By Lemmas 1 and 3 we know that Cut Paren is at most twice as expensive as any optimal solution. Hence,

Theorem 1. *Given simple routing paths,* Cut Paren *is a 2-approximation algorithm.*

Using the lower bound given by Lemma 1, no algorithm can beat the approximation ratio of 2 since the optimal solution can cost twice as much as the infeasible solution generated by Max Thru. For example, this can happen on a cycle. Suppose there is a unit demand between any two neighboring nodes u and v and the demand is routed along the "long" path excluding link uv. In this example, as the number of nodes grows, the gap between the optimal cost and the lower bound given by Max Thru approaches 2.

So far we have presented an algorithm that can cut an improper line system with a guaranteed approximation ratio of 2. In fact, we can always cut an improper line system L optimally. For each node v in the node sequence of the line system, we use a binary variable x_v to indicate whether or not node v is cut. The objective is to minimize $\sum_v c_v x_v$ where c_v is the number of transparent sections of L that go through node v at this point in the node sequence. In other words, the objective is to minimize the total number of times that the transparent sections on this line system are cut. The constraints are $\sum_{v \in P} x_v \geq 1$ for every P where P is the set of nodes enclosed by a set of matching parentheses. In other words, the constraints make sure that each loop in the line system is cut. It can be shown that the coefficient matrix of the constraints is *totally unimodular* (TUM). Therefore, the linear relaxation of the above integer program always has an integral optimal solution [16], and so we have the following lemma.

Lemma 4. *Given a set of transparent sections on an improper line system, we can efficiently cut the line system so that the transparent sections are cut into as few pieces as possible.*

We do not know if Lemma 4 can lead to a better approximation ratio than 2. We know that even if we cut the improper line systems generated by Max Thru optimally, we can still end up with a solution that costs 5/4 as much as the optimal solution with proper line systems. An example is given in the Appendix.

2.3 Networks of Node Degree 3

The situation is far better when the network only has nodes of low degree. In particular, if the network has maximal degree of at most 3, we can indeed find an optimal set of proper line systems for any given simple demand routes. This is fortunate since optical networks are typically sparse, and only have a small number of nodes of degree greater than 3. Again, we begin with Max Thru. Since the node degree is at most 3, we observe that a resulting line system is improper only if it is a simple closed loop, i.e. a closed loop that induces simple routes only. We now examine each node u on the loop, which we call L. Let x and y be u's neighboring nodes on the loop L and z be u's neighboring node off the loop (if such a z exists). Without loss of generality, let us assume the through traffic $T(xuz)$ is no smaller than the through traffic $T(yuz)$. (If z does not exist, then $T(xuz) = T(yuz) = 0$.) An alternate configuration at u would be to cut xuy at u and connect xuz, and we refer to this operation as a *swap*. This swap would increase the total number of transparent sections by $I(u)$ where $I(u) = T(xuz) - T(xuy)$. If node $u' \in L$ has the minimum increase $I(u')$ then we perform the swap operation at u'. We swap exactly one node on each closed loop. We refer to this algorithm as Greedy Swap. Since every node has degree at most 3, Greedy Swap eliminates all loops and never creates new ones. Therefore, the Greedy Swap algorithm defines a set of proper line systems.

Theorem 2. *The* Greedy Swap *algorithm is optimal for given demand routing on networks with node degree at most 3.*

Proof. We first note that no matter how Greedy Swap breaks ties in decision making the resulting solutions have the same total number of transparent sections. Given a solution by an optimal algorithm Opt, we now show that Greedy Swap produces an identical solution under some tie breaking. In particular, when executing Greedy Swap let us use the optimal solution for tie breaking as follows. When executing Max Thru if multiple configurations maximize the through traffic at u and if one of these configurations is given by Opt, then Max Thru configures u like Opt. Suppose Max Thru creates a simple loop L and multiple ways of swapping a node on L yield the same amount of minimum increase. If some of these swaps result in an identical configuration for the node $u \in L$ being swapped as in the optimal solution, then Greedy Swap executes one of these particular swaps on L.

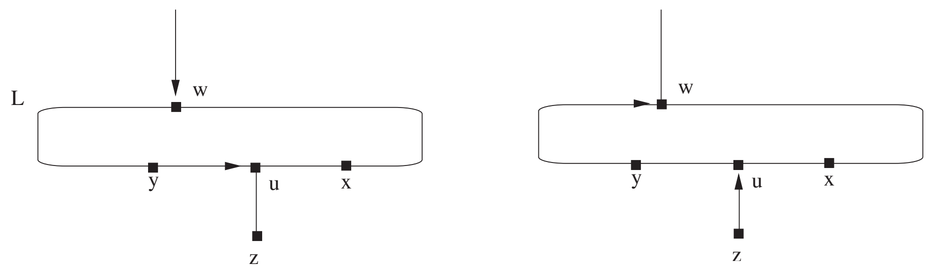

Fig. 3. (Left) OPT. (Right) Result of Greedy Swap.

Now let us compare the line systems produced by Greedy Swap against the optimal solution. We first examine each node u for which Max Thru and Greedy Swap have the same configuration. We claim that Opt has the same configuration at u as well. Let us focus on the case in which u has three neighbors x, y and z. (The cases in which u is degree 1 or 2 are simpler.) If Max Thru does not connect xuy, xuz or yuz, Opt must have the same configuration at u since the through traffic $T(xuy)$, $T(xuz)$ and $T(yuz)$ must be all zero and Opt is used for tie breaking. Otherwise, let us assume without loss of generality that Max Thru connects xuy. For the purpose of contradiction, let us assume that Opt connects xuz. By the construction of Max Thru and the tie breaking rule, we have $T(xuy) > T(xuz)$. If Opt reconfigures node u by connecting xuy instead, a loop L must be formed or else we would have a better solution than Opt (see Figure 3). On the loop L there must be a node $v \neq u$ such that Greedy Swap and Opt configure v differently, since otherwise Greedy Swap would contain this loop L. Let w be the first such node, i.e. all nodes on L between u and w in the clockwise direction are configured the same by Greedy Swap and Opt.

If w has the same configuration in Max Thru and Greedy Swap, then by the definition of Max Thru and by the tie breaking rule, the configuration of w with Max Thru must be strictly better than its configuration with Opt. Hence, by reconfiguring Opt at nodes u and w like Greedy Swap, we get a better solution than Opt which is a contradiction. Therefore, as shown in Figure 3, w must be a node that was cut by Greedy Swap on a loop generated by Max Thru that included u, which implies that $I(w) \leq I(u)$. In fact, it must be the case that $I(w) < I(u)$ by the tie breaking rule. Therefore, if we reconfigure Opt at nodes u and w like Greedy Swap, we once again get a better solution than Opt. This is a valid solution, since we cannot create any loops in Opt by reconfiguring both u and w. Hence, if MAX THRU and GREEDY SWAP have the same configuration for node u, OPT has the same configuration at u as well.

We finally examine each node u that MAX THRU and GREEDY SWAP configure differently. This node u can only exist on a closed loop L created by MAX THRU and each closed loop can only have one such node. For every node $v \in L$ and $v \neq u$, MAX THRU and GREEDY SWAP configure v in the same way by the construction of GREEDY SWAP. Hence, by our argument above, OPT configures v in the same way as well. Since OPT contains no loops, it has to configure node u like GREEDY SWAP.

3 The General Case

The general case in which neither demand routes nor line systems are given is much harder. Contrary to Theorem 2, the general case is NP-hard even for the following restricted instance on 3-regular networks. Suppose there is a unit demand between a common source node r and every other node in G. Since G is 3-regular, we have a proper line system that visits every node in G if and only if G has a Hamiltonian path. Since deciding the existence of a Hamiltonian path for 3-regular graphs is NP-hard [8], deciding whether or not the optimal solution is at most $n-1$ is NP-hard as well. Furthermore, we also have a stronger result as stated in Theorem 3.

Theorem 3. *If routes are not specified, no algorithm can guarantee a constant approximation ratio better than 2 for our problem even if the network is 3-regular and Hamiltonian and even if all demands originate from the same source node.*

3.1 A Logarithmic Approximation

In this section we present a logarithmic approximation for the general case. Let T be any tree that spans all source nodes and destination nodes. We choose an arbitrary node r in T to be the root of the tree. We route every demand from its source to r and then to its destination along edges of T.

We arrange the line systems as follows. Consider a node y, and let z be its parent node and $C(y)$ be the set of child nodes. For every child node $x \in C(y)$, let $N(x)$ be the size of the subtree rooted at x. If x' is the child node that has the largest subtree then the only line system through node y is along zyx'. We now bound the number of transparent sections from any node to the root r. Let $f(x)$ be the maximum number of transparent sections from any node in the subtree rooted at x to the parent node y of x. Then,

$$f(y) = \max_{x \in C(y), x \neq x'} \{f(x'), f(x)+1\}.$$

Note that $N(y) = 1 + \sum_{x \in C(y)} N(x)$. Since $N(x') \geq N(x)$ for any $x \in C(y) - \{x'\}$, we have $N(x) < N(y)/2$. This means whenever the number of transparent sections increases by 1, the size of the subtree is cut by at least a half. Hence, the maximum number of transparent sections from a node in T to the root r is at most $\log n$ where n is the number of nodes in T. For any demand to go from its source node to its destination node the number of transparent sections is at most $2\log n$. This is also true because the path decomposition we have formed is a *caterpillar decomposition* [14].

If we redefine $N(x)$ to be the total number of demands that have either source nodes or destination nodes in the subtree rooted at x, then the same construction of the line systems as above would bound the maximum number of transparent sections to $2\log(2K)$ where K is the total number of demands. Since the total cost of any solution must be at least K, we have the following theorem.

Theorem 4. *For the general case where demand routes are not specified, we can find a* $2 \log n$*-approximation and a* $2\log(2K)$ *approximation in polynomial time.*

Notice that this approximation ratio is tight comparing against the lower bound, K, that we are using. To see this, consider a balanced binary tree with one unit demand between the root and every leaf node. The optimal solution costs $\Theta(K \log n)$ or $\Theta(K \log K)$.

3.2 Rings

The ring topology is of particular interest since the underlying infrastructure of metro-area networks is often a ring [18]. In this context, we have a set of *core* nodes connected in a ring and each core node is attached to an *access* node, as in Figure 4(Left). Among other things, the access nodes have the capability of multiplexing row-rate local traffic streams into high-rate optical signals. Such access nodes are particularly useful when end users are far away from the core ring. In this ring network, all demands are between access nodes. It is easy to see that each demand has two routes, each access node has one configuration and each core node has three configurations which completely determine the layout of the line systems. Under this constrained scenario, it is reasonable to expect that we can do much better than the logarithmic approximation of Section 3.1. Indeed, we obtain the following result.

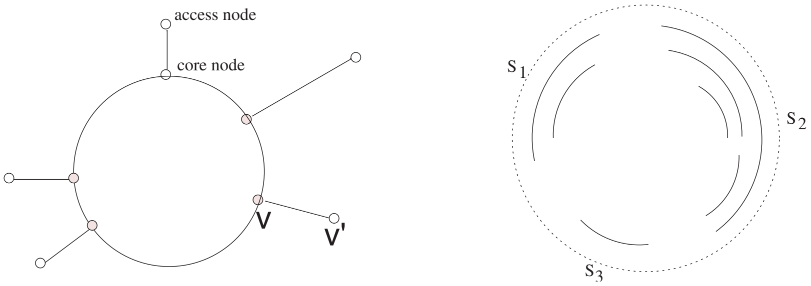

Fig. 4. (Left) A ring network. (Right) A possible consistent routing of S and the sets S_j.

Theorem 5. *In ring networks, with no multiple demands between node pairs, we can find a* $3/2$*-approximation in polynomial time.*

To prove the above theorem we show that either we can find an optimal solution efficiently or the following *whole ring solution* serves as a $3/2$ approximation. We denote a core node by variables such as v and the access node attached to v by v'. If u and w are neighboring core nodes of v then the three configurations at v are denoted by $v'vu$, $v'vw$ and uvw. The first two configurations mean v and its access node v' are on the same line system; the third configuration means that vv' is a separate line system and u, v and w are on a line system. In the whole ring solution, vv' is a separate line system for every v. All the core nodes form a single line system around the ring and it is cut open at an arbitrary core node. It is easy to see that in the whole ring solution we can route the demands so that each demand requires 3 transparent sections.

Before we describe our algorithm we begin by describing *crossing*, *non-crossing* demands and their properties. For a demand that starts at v' and ends at w', we refer to v and w as the *terminals* of this demand. We call a pair of demands a *crossing* pair if the four terminal nodes are distinct and one terminal of a demand always appears in between the two terminals of the other demand. For a demand in a crossing pair, if it switches line systems at a terminal node that belongs to its crossing partner we call this switch a *normal* jump. Otherwise, the switch is called an *extra* jump. Because of the layout of crossing demands, we know that without any extra jumps, a pair of crossing demands requires 4 transparent sections no matter how it is routed.

Now we consider a pair of non-crossing demands. If their routes do not overlap or overlap at one contiguous section only, then we say the demand paths are *consistent*. Note that two routes overlap even if they share one common node. If the demand paths of all pairs of non-crossing demands are consistent, we say the solution has consistent routing. We establish the following two properties about routing non-crossing demands consistently.

Lemma 5. *There is an optimal solution that we call OPT, for which the routing is consistent.*

Lemma 6. *For a set of mutually non-crossing demands S, the number of ways to route the demands consistently is linear in $|S|$.*

The Algorithm. We now describe our 3/2-approximation algorithm. From the set of demands, we take out crossing pairs of demands, one pair at a time in arbitrary order, until we have no crossing pairs left. Let C be the set of crossing pairs of demands that we have removed, and S be the set of non-crossing demands that are left. By Lemma 5 we know that OPT must use consistent routing, and by Lemma 6 we can efficiently enumerate all possible consistent routings for demands in S. Hence, if C is empty we know how to find an optimal solution and we are done. Otherwise, we assume from now on that we know how OPT consistently routes the demands in S. We also know that the routing of S results in a collection of disjoint intervals $I_1, I_2, \ldots$ around the ring, where each interval I_j consists of links that are on the routes of demands of S. Let $S_j \subseteq S$ be the set of demands whose routes are included in I_j, as illustrated in Figure 4(Right). We denote by $c(S_j)$ the total number of transparent sections that the demands in the set S_j use in OPT. Each set S_j obeys the following lemma.

Lemma 7. *We have $c(S_j) \geq 2|S_j| - 1$. The equality happens only if at each border node s of I_j, the configuration of s is $s'sv$ where $v \in I_j$. Finally, if I_j is the entire ring, then $c(S_j) \geq 2|S_j|$.*

If $c(S_j) = 2|S_j| - 1$ and the interval I_j causes no extra jumps for any demand in C, we call the set S_j *thrifty*. Of course, we do not know a priori whether or not a set is thrifty. We consider each set S_j one by one. Supposing that S_j is thrifty, we show in Lemma 8 that the routes of demands in C are determined as well. Given the routes of all demands, we can find the optimal line systems by Theorem 2. If none of the S_j are thrifty, we use the whole ring solution which has a cost of $3K$. Now that we have obtained a total of $O(x)$ solutions where x is the number of sets S_j, we choose the least expensive one among them. If it is the case that OPT contains no thrifty set, the solution we have found is no worse than the whole ring solution which guarantees a 3/2-approximation by Lemma 8. On the other hand, if OPT contains some thrifty set, our solution is guaranteed to find this set through enumeration and is therefore optimal. This proves Theorem 5.

Lemma 8. *If OPT does not contain a thrifty set, the whole ring solution is a 3/2-approximation. If OPT contains a thrifty set, then we can find the routing of all demands of C in OPT.*

Proof. If there are no thrifty sets, consider the total number of transparent sections in OPT. As noted above, each crossing pair has exactly 4 transparent sections without considering the extra jumps. Each S_j has either at least $2|S_j|$ transparent sections in total, or has $2|S_j| - 1$ transparent sections and also causes an extra jump of a crossing demand in C on I_j. Therefore, in total the number of transparent sections in OPT is at least $2K$. The whole ring solution requires 3 transparent sections per demand. Therefore, it is a $3/2$ approximation.

Now suppose S_1 is thrifty. By Lemma 7, we know the interval I_1 is not the whole ring. Consider a demand d_1 in C. Let P_1 be its route in OPT, and let d_2 be its crossing partner. If both terminals of d_1 are contained in the interval I_1, then it must be the case that $P_1 \subseteq I_1$. Otherwise, d_1 would jump at both border nodes of I_1 by Lemma 7. One of these jumps has to be extra since d_1 and d_2 are crossing and therefore the two border nodes of I_1 cannot both be terminals of d_2. Similarly, if neither terminal of d_1 is contained in I_1, it must be the case that $P_1 \cap I_1 = \emptyset$. Otherwise, d_1 would jump at both border nodes of I_1. Again, one of these jumps has to be extra. In both cases we know how d_1 is routed.

If exactly one terminal of d_1 is contained in I_1, then the border node of I_1 that is contained in P_1 must be a terminal of d_2 in order to avoid an extra jump. If only one border node of I_1 is a terminal node of d_2, then we know which route d_1 takes. Otherwise, both border nodes of I_1 must be terminal nodes of d_2. In this case d_1 has two routes to consider. Given that we have assumed there are no multiple demands with the same terminal pairs, only one demand in C can have two possible routes if S_1 is thrifty. We can easily enumerate both possibilities.

This finishes our 3/2-approximation algorithm for rings. However, if the demand originating at every node of the ring is bounded by some constant k, we can find the optimal solution in polynomial time, even if the same source-destination pair can have multiple demands. The proof can be found in the Appendix.

4 Variations and Open Problems

In this paper, we introduced a new cost metric which measures the cost of routing a demand by the number of line systems that the demand travels through. We presented a collection of results. However, many questions remain open and some variations deserve further attention.

Proper Line Systems. The algorithm we have presented for line system partitioning with given demand routes guarantees a 2-approximation. However, we have no example in which our algorithm gives an approximation ratio worse than $5/4$, or any reason to believe that a different algorithm may not do even better. If the demand routes are not given, we only have an $O(\log n)$-approximation that compares against a rather trivial lower bound, K, on OPT. In fact, we know that on a binary tree OPT can be as much as $\Theta(K \log n)$. Therefore, a much better approximation seems likely. The only inapproximability result says that the approximation ratio cannot be better than 2, so even a small constant approximation may be possible.

It is worth noting that our model makes several simplifications. Most notably, we assume that optical fibers have infinite capacity and that line systems can have arbitrary length. It would be quite interesting to introduce one or both of these complications into our model and see if good algorithms are still possible.

Improper Line Systems and Supereulerian Networks. We have restricted the line systems to be proper in this paper for reasons discussed in the Introduction. If we relax our requirement so that each line system may repeat nodes in any manner without repeating any links, this relaxed problem remains fascinating. In fact, it is closely related to *supereulerian* graphs, a survey on which can be found in [3]. See the Appendix for further discussion on this topic.

Acknowledgements

We would like to thank Chandra Chekuri, Jon Kleinberg, and Peter Winkler for productive and illuminating discussions, and thank Steve Fortune for his insight in defining the model.

References

1. C. Bazgan, M. Santha, and Z. Tuza. On the approximability of finding a(nother) Hamiltonian cycle in cubic Hamiltonian graphs. *Journal of Algorithms*, 31:249 – 268, 1999.
2. J.-C. Bermond, N. Marlin, D. Peleg, S. Pérennes. Virtual path layouts with low congestion or low diameter in ATM networks. In *Proceedings of 1ère Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*, 1999, pp. 77-82.
3. P. A. Catlin. Supereulerian graphs: a survey. *Journal of Graph theory*, 16(2):177 – 196, 1992.
4. S. Cosares and I. Saniee. An optimization problem related to balancing loads on SONET rings. *Telecommunications Systems*, 3:165 – 181, 1994.
5. B. Doshi, R. Nagarajan, N. Blackwood, S. Jothipragasam, N. Raman, M. Sharma, and S. Prasanna. LIPI: A lightpath intelligent instantiation tool: capabilities and impact. *Bell Labs Technical Journal*, 2002.
6. P. Fishburn. *Interval orders and interval graphs.* Wiley and Sons, New York, 1985.
7. S. Fortune, W. Sweldens, and L. Zhang. Line system design for DWDM networks. Submitted.
8. M. R. Garey and D. S. Johnson. *Computers and intractability - A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York, 1979.
9. O. Gerstel and A. Segall. Dynamic maintenance of the virtual path layout. In *Proceedings of IEEE INFOCOM'95*, April 1995.
10. S. Khanna. A polynomial-time approximation scheme for the SONET ring loading problem. *Bell Labs Technical Journal*, 1997.
11. J. Kleinberg and A. Kumar. Wavelength conversion in optical networks. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 566 – 575, January 1999.
12. V. Kumar and E. Schwabe. Improved access to optical bandwidth in trees. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 437 – 444, January 1997.
13. W. Lee. Personal communication. 2003.
14. J. Matoušek. On embedding trees into uniformly convex banach spaces. *Israel Journal of Mathematics*, 114:221 – 237, 1999.
15. M. Mihail, C. Kaklamanis, and S. Rao. Efficient access to optical bandwidth. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 548 – 557, 1995.
16. C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization.* Dover Publications, Mineola, New York, 1998.
17. P. Raghavan and E. Upfal. Efficient routing in all-optical networks. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 134–143, 1994.
18. R. Ramaswami and K. Sivarajan. *Optical networks A practical perspective.* Morgan Kaufmann, San Francisco, CA, 1998.
19. A. Schrijver, P. D. Seymour, and P. Winkler. The ring loading problem. *SIAM Journal of Discrete Math*, 11(1):1 – 14, 1998.
20. G. Wilfong and P. Winkler. Ring routing and wavelength translation. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 333 – 341, January 1998.
21. P. Winkler and L. Zhang. Wavelength assignment and generalized interval graph coloring. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, January 2003.
22. S. Zaks. Path Layout in ATM Networks - A Survey. In *Networks in Distributed Computing*, M. Mavronicolas, M. Merritt, and N. Shavit, Eds., DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, 1998, pp. 145-160.

Appendix

Theorem 6. *For networks with node degree bounded by 12 or above, with demand routes given, it is NP-hard to find the optimal proper line system arrangement.*

Proof. The reduction is from 3-SAT. Suppose we are given a 3-SAT formula with variables x_i and clauses C_j, with k clauses. For each pair (i, j), form the following gadget. It is a star with node $v(i, j)$ at the center, and with 12 legs of length 2. Denote the nodes at the edges of the legs by $t(i, j, 1) \ldots t(i, j, 12)$ and the nodes in the middle of the legs by $t'(i, j, 1) \ldots t'(i, j, 12)$. There is a demand from node $t(i, j, \ell)$ to node $t(i, j, \ell + 1)$ of size a with the specified route in the gadget going through $v(i, j)$. There is also such a demand from node $t(i, j, 12)$ to node $t(i, j, 1)$.

Choose a to be really large (larger than all the demands we are going to add later put together). Notice that for each gadget as described above, exactly half the demands must have to use two transparent sections. If a solution is optimal, it must not cut any more demands of size a. Therefore, the configuration of each node $t'(i, j, \ell)$ must be to link the two edges coming out of it together into a single line system, and there are only two possible configurations for $v(i, j)$. These are the matching linking $t'(i, j, \ell)$ to $t'(i, j, \ell + 1)$ for ℓ odd, and the matching linking $t'(i, j, \ell)$ to $t'(i, j, \ell + 1)$ for ℓ even (with $t'(i, j, 12)$ linked to $t'(i, j, 1)$). The first case corresponds to x_i being true, and the second to x_i being false.

However, we want to make sure that all gadgets that correspond to x_i are in the same configuration in the optimal solution. Therefore, we join each gadget corresponding to (i, j) with the gadget corresponding to $(i, j + 1)$ as follows. We form many gadgets we call *crosses*, which is a star with node v in the center and 4 legs of length 1, which we label legs 1 through 4. There is a demand of size a from the edge of leg 1 to leg 2, and from the edge of leg 3 to leg 4. Therefore, in the optimal solution, we know exactly what the configuration of v must be. Add edges $(t(i, j, 7), t(i, j + 1, 5))$ and $(t(i, j, 9), t(i, j + 1, 3))$. Also add an edge from $t(i, j, 8)$ to leg 2 of a cross, and an edge from $t(i, j + 1, 4)$ to leg 3 of the same cross. Form demands of size b between $(t'(i, j, 7), t'(i, j + 1, 5))$ and $(t'(i, j, 9), t'(i, j + 1, 3))$. Also form these demands from $t'(i, j, 8)$ and $t'(i, j + 1, 4)$ to v, the node in the center of the cross. The routes of these demands are along the newly formed edges. This insures that if two adjacent nodes $v(i, j)$ and $v(i, j + 1)$ have different configurations, then some demand of size b must use two transparent sections, otherwise we would have an improper line system. If we make b larger than the sum of all demands we have not mentioned yet, then in any optimal solution, the configurations of all $v(i, j)$ for fixed i must be the same.

For each clause C_j, form the following gadget. Suppose the variables that appear in C_j are x_{i_1}, x_{i_2}, and x_{i_3}. Take the star gadgets with the nodes $v(i_1, j)$, $v(i_2, j)$, and $v(i_3, j)$ in the center. Also take a not-yet used cross with v at the center. We add 4 edges to link these 4 gadgets together so they form a cycle. If x_i appears as a positive literal in C_j, we use the nodes $t(i, j, 1)$ and $t(i, j, 12)$ for this linkage, otherwise we use the nodes $t(i, j, 1)$ and $t(i, j, 2)$. In the cross, we use legs 1 and 3. For example, if $C_j = \overline{x_{i_1}} \vee x_{i_2} \vee x_{i_3}$, then we add edges $(t(i_1, j, 1), t(i_2, j, 2))$, $(t(i_2, j, 1), t(i_3, j, 2))$, $(t(i_3, j, 1), leg1)$, and $(leg3, t(i_1, j, 12))$. For each of these new edges, we also add a demand of size 1 from the corresponding t' nodes with the route along the newly added edge. In the above example, we would add demands $(t'(i_1, j, 1), t'(i_2, j, 12))$, $(t'(i_2, j, 1), t'(i_3, j, 12))$, $(t'(i_3, j, 1), v)$, and $(v, t'(i_1, j, 2))$.

We know that in any optimal solution, the demands of size a and b each use only one transparent section, and this is always possible. Therefore, we forget about their cost, since it is a constant, and consider as the cost of the optimal solution the number of transparent sections used by the demands of size 1. Now we show that the formula is satisfiable iff the cost of the optimal solution here is $4k$, i.e. if there is an way so that each demand of size 1 only uses one transparent section.

If the formula is satisfiable, and we configure each $v(i, j)$ corresponding to the satisfying assignment (the configurations of all the other nodes are fixed so as not to cut demands of size a and b), then each demand of size 1 uses only one transparent section. This forms proper line systems, since the cycle formed by the clause is not a single line system, it is cut by some gadget corresponding to the literal that is true in the clause.

To the contrary, if there is a solution where each demand of size 1 uses a single transparent section, then the cycle formed by the clause gadget is a single line system unless there is some gadget corresponding to a literal in the clause that cuts it. There must be such a gadget for each clause since otherwise the cycle formed by this clause gadget would not be a proper line system. Therefore, since the configurations on all literal gadgets must be the same in all clauses, there must be a literal that is true in every clause, and therefore there is a satisfying assignment.

Proof of Lemma 1. Let $T(v) = \sum_{uv,vw \in E(v)} T(u,v,w)$ be the total through traffic at v and let $M(v) = \sum_{(uv,vw):matching\ pair} T(u,v,w)$ be the total through traffic along the matched link pairs of v. The number of demands that have to switch from one line system to another at node v is $T(v) - M(v)$. Since $T(v)$ is fixed given fixed demand routes and Max Thru maximizes $M(v)$ at each node, no algorithm can outperform Max Thru. □

Proof of Lemma 2. The line systems resulting from Cut Paren have had all matching parentheses cut in the middle. The cut could be made at the node corresponding to the closing parenthesis. With the node sequence as above, a node can appear in the node sequence of these line systems at most once, except the first and last node might be the same. Therefore, Cut Paren generates proper line systems. □

Lower bound for optimal cutting of Max Thru. We do not know if Lemma 4 can lead to a better approximation ratio than 2. We know that even if we cut the improper line systems generated by Max Thru optimally, however, we can still end up with a solution that costs 5/4 as much as the optimal solution with proper line systems.

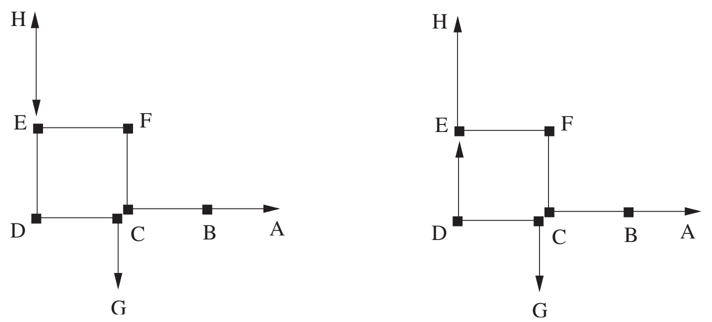

Fig. 5. (Left) The improper line systems generated by Max Thru. (Right) The optimal proper line systems.

This is true in the example in Figure 5, with x units of demand along the route $FEDCG$, x units of demand along the route $ABCFE$, and $x-1$ units of demand along the route $HEFC$. Max Thru generates the line systems in Figure 5(Left), and so the optimal cutting of these line systems form a solution that costs $5x-2$. However, the optimal solution in Figure 5(Right) only costs $4x-1$. Therefore, we get a lower bound of 5/4 as $x \to \infty$ for the approximation ratio of the above algorithm.

Notice that if we use an algorithm (like GREEDY SWAP) that reconfigures an improper line system instead of just cutting it, we would obtain the optimal solution in this example. However, there are problems with this approach in general, because it is difficult to determine that we are not forming new improper line systems while cutting and reconfiguring the old ones.

Proof of Theorem 3. In this proof, we use the following result from [1]:

Theorem [Bazgan-Santha-Tuza] *The longest path problem cannot be approximated within a constant factor in 3-regular Hamiltonian graphs.*

As before, we have a unit demand between a common source node r and every other node in the network. Consider any α-approximation algorithm and the set of line systems that the algorithm produces. Suppose the line system that contains node r has cn nodes where $c \leq 1$. The total number of transparent sections in this solution is at least $(cn-1)+2(n-cn)$, since every demand that is not terminated in the line system that contains r needs at least 2 transparent sections.

Since the network is Hamiltonian, the optimal solution is $n-1$ in which the line system is the Hamiltonian path and every demand has 1 transparent section. Since the algorithm of interest guarantees an α-approximation, its solution is at most $\alpha(n-1)$. Therefore, we have that,

$$(cn-1)+2(n-cn) \leq \alpha(n-1).$$

If α is a constant smaller than 2, then the above inequality implies that c is a constant at least $2-\alpha$. Therefore, we have found a path in the 3-regular Hamiltonian graph whose length at least a constant fraction of the optimal value of n. This contradicts the inapproximability theorem of Bazgan, Santha, and Tuza. □

Proof of Lemma 5. Consider a pair of non-crossing demands AD and BC. Without loss of generality let us assume the four end nodes appear clockwise on the ring in the order of A, B, C and D. If the two demands are routed inconsistently, then AD takes the route clockwise from A to D and BC takes the route counterclockwise from B to C, as shown in Figure 6. We show in the following that, without reconfiguring the line systems at any node, we can always reroute either AD or BC so that the cost of the resulting solution is never increased.

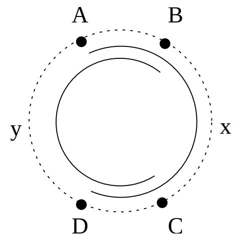

Fig. 6. A pair of non-crossing demands AD and BC that are routed inconsistently.

Suppose that v_1, v_2, and v_3 are consecutive nodes on the ring in clockwise order. We say that v_2 is configured "R" if its configuration is $v_2'v_2v_1$, that it is configured "L" if its configuration is $v_2'v_2v_3$, and that it is configured "T" otherwise. Let x be the number of nodes between B and C (clockwise from B to C, excluding B and C) that are configured "R" or "L", i.e. the number of line system switches clockwise from B to C. Similarly, let y be the number of nodes counterclockwise from A to D that are configured "R" or "L". Without loss of generality let us assume that $x \leq y$.

The following table enumerates the 9 possible configurations at nodes B and C and the number of line system switches that demand BC needs when routed counterclockwise from B to C (column ccw) and if rerouted clockwise from B to C (column cw). It is easy to see in the first 6 cases that rerouting BC in the clockwise direction either improves the solution or makes it no worse.

	B	C	ccw	cw
1	T	T	$\geq y+2$	$x+2$
2	L	R	$\geq y+2$	x
3	L	L	$\geq y+1$	$x+1$
4	R	R	$\geq y+1$	$x+1$
5	T	R	$\geq y+2$	$x+1$
6	L	T	$\geq y+2$	$x+1$
7	T	L	$\geq y+1$	$x+2$
8	R	T	$\geq y+1$	$x+2$
9	R	L	$\geq y$	$x+2$

For cases 7 and 8, if $y \geq x+1$ or if column ccw is larger than $y+1$ then we can reroute BC without making the solution worse. Hence, for cases 7 and 8 we consider the situation in which $x = y$ and column ccw is exactly $y+1$. When column ccw is exactly $y+1$ the configuration at A and D must be both "T". As a result, the number of switchings demand AD needs when routed clockwise from A to D equals $x+3$; the number of switchings equals $y+2$ when AD is rerouted counterclockwise from A to D. We can therefore reroute AD to improve the solution.

For case 9, if $y \geq x+2$ we can reroute BC. Hence, we consider $x = y$ and $y = x+1$. For $x = y$, we can reroute AD since when routed counterclockwise AD needs at least $x+2$ line systems switchings and when rerouted clockwise AD needs $y+2$. For $y = x+1$, if column ccw is larger than y we can then reroute BC. Otherwise, column ccw is exactly y and the configuration at A and D must be both "T". We can reroute AD since when routed counterclockwise AD needs at least $x+4$ line systems switchings and when rerouted clockwise AD needs $y+2$ switchings. Therefore, rerouting AD improves the solution.

In the above argument we have assumed that A and B are distinct nodes and C and D are distinct nodes. The proof is also applicable if either $A = B$ or $C = D$ holds. □

Proof of Lemma 6. Let $s \subset S$ be any subset of mutually noncrossing demands. We assume inductively that we can enumerate all consistent routing of s in time linear in $|s|$. Consider any demand $d \in S$. By the definition of consistent routing, fixing a route P for d also fixes all routes for demands for which both terminals are contained in P. Let S_P be the set of the remaining demands, i.e. those for which both terminals are outside P, or for which one terminal is outside and one terminal is on the border of P. No demands with one terminal inside P and one outside can exist, since we assumed that all demands are non-crossing. Inductively, we can enumerate all consistent routings of S_P in time linear in $|S_P|$. If P and Q are two routes for d we can enumerate consistent routings for S in time linear in $|S_P| + |S_Q|$. Since S_P and S_Q are disjoint, the running time is linear in $|S|$. □

Proof of Lemma 7. Consider a set S_j with a consistent routing and the corresponding interval I_j. Order the nodes of I_j in clockwise order. We further subdivide I_j into subintervals $A_1, A_2, \ldots, A_k$ as follows. Each A_i starts at node v_{i-1} and end at node v_i (in clockwise order), with v_0 being the first node of I_j and v_k being the last node of I_j. Take the longest route P of the demands of S_j that starts at the terminal corresponding to the first node of I_j, and make $P \cap I_j$ be A_1. Denote the demand with route P as d_1. Then we take the longest route of the demands that starts at v_1'

and continues clockwise, and then let the intersection of that route with I_j be A_2. If we continue in this manner, we obtain the above decomposition, where each A_i is formed from the route of some demand d_i.

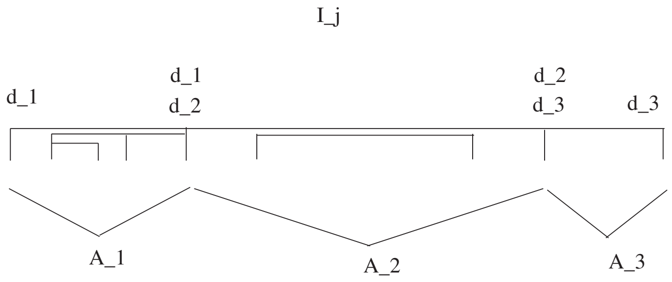

Fig. 7. The division of I_j into subintervals. The terminals of d_i are labeled with "d_i".

Each route of a demand in S_j is contained entirely within one of the intervals A_i. This is because if a demand route begins at v_{i-1}, it must be contained in A_i since we took the longest route when we formed A_i. If a demand route begins at a node between v_{i-1} and v_i, then it must also end at such a node, since otherwise it would form a crossing pair with the route we used to create A_i. Since the routing is consistent, the demand route must be entirely contained in A_i.

Let R_i be the demands of S_j with routes in A_i. We first show that $c(R_i) \geq 2|R_i|-1$. Suppose that there are x demands of R_i with only one transparent section. Assume d_i is not one of these, since then $c(R_i) \geq 1+2(|R_i|-1)$. These demands must have disjoint routes, since if their routes intersect, one of the two demands must switch line systems, and therefore form a second transparent section. The demands cannot have both terminals in common since we assume that there are no multiple demands between the same terminals. Therefore, all x of these demands have disjoint routes. Then, d_i must have at least $2x-1$ transparent sections, since it has to switch line systems every time it enters and every time it leaves a route of one of these x demands. Moreover, d_i can only have $2x-1$ transparent sections if the configuration of v_i is $v_i'v_iv$ with $v \in A_i$ and the configuration of v_{i+1} is $v_{i+1}'v_{i+1}w$ with $w \in A_i$. Therefore, as desired,

$$c(R_i) \;\geq\; x + (2x-1) + 2(|R_i| - x - 1) \;=\; 2|R_i| + x - 1 \;\geq\; 2|R_i| - 1.$$

Now consider the cost of the entire set S_j. We know that each set R_i can only cost $2|R_i|-1$ if the necessary conditions at the border nodes of A_i hold. However, since each A_i and A_{i+1} share a border node, only one of then can have the necessary configuration, so at least one of them must have higher cost. Each shared border node increases the total cost by 1, since it makes one of the adjacent sets R_i have 1 higher cost than before (if both border conditions are not met for R_i, then by the same argument as above, $c(R_i) \geq 2|R_i|+1$). There are $k-1$ shared border nodes, and therefore,

$$c(S_j) \;\geq\; (k-1) + \sum_i (2|R_i| - 1) \;=\; (k-1) + 2(\sum_i |R_i|) - k \;=\; 2|S_j| - 1.$$

Notice that if I_j is the entire ring, then there are k border nodes, so $c(S_j) \geq 2|S_j|$. □

Solution for ring with bounded demand per node. If the demand originating at every node of the ring is bounded by some constant k, then we can find the optimal solution in polynomial time, even if the same source-destination pair can have multiple demands. Suppose OPT partitions the core nodes into 2 or more line systems. Let us consider the set of core nodes v that are configured in the form of $v'vw$. The set has at least two nodes, one of which, say v, has to have through traffic $T(uvw)$ no higher than k. Otherwise, we can reconfigure v to uvw. The resulting line systems are proper and better than OPT. We consider the core nodes one by one. For v to have through traffic no higher than k, a set of at most k demands can be routed through v and there are at most $\sum_{j\leq k}\binom{K}{j}$ such sets. We can enumerate these $O(K^k)$ possibilities and find the optimal line systems for each possible routing.

Suppose OPT puts all the core nodes in one line system and one end of the line system is configured $v'vw$. In OPT every demand that does not have v as a terminal is routed to avoid v; every demand that has v as a terminal is routed in the direction of vw.... Therefore, we enumerate all possibilities of the end of the line system and there are $O(n)$ of them. For each possibility, the routing is fixed and the optimal solution can be found. Thus, we have enumerated all possible solutions in polynomial time. We choose the least expensive one.

Discussion on Supereulerian Graphs A graph is *supereulerian* if it has a spanning Eulerian subgraph, i.e. if there is a tour that visits every node but not necessarily every link. (A survey of results on supereulerian graphs can be found in [3]). Therefore, a graph is supereulerian exactly when we can connect all nodes using a single line system. For the case in which all node pairs have positive demand, determining if a graph is supereulerian is equivalent to determining if the optimal solution is K, the total demand. We can draw on results about supereulerian graphs to answer this question for certain types of networks. For example, all networks that have 2 edge-disjoint spanning trees are supereulerian, and therefore can be connected by a single line system. This is also true for 4-edge-connected graphs.

Nevertheless, determining if a graph is supereulerian is far from sufficient for our purpose. Even decomposing a graph into as few Eulerian subgraphs as possible does not reflect our cost function, which depends on the layout of the line systems, not just how many line systems there are. It would be interesting to identify sufficient graph conditions for which our problem can be solved optimally or near optimally.