Approximation Algorithms for Wavelength Assignment

Vijay Kumar* Atri Rudra†

Abstract

Winkler and Zhang introduced the FIBER MINIMIZATION problem in [3]. They showed that the problem is NP-complete but left the question of approximation algorithms open. We give a simple 2-approximation algorithm for this problem. We also show how ideas from the Dynamic Storage Allocation algorithm of Buchsbaum et al [1] can be used to give an approximation ratio arbitrarily close to 1 provided the problem instance satisfies certain criteria. We also show that these criteria are necessary to obtain an approximation scheme. Our 2-approximation algorithm achieves its guarantee unconditionally.

We generalize the problem to a ring network and give a $2 + o(1)$-approximation algorithm for this topology. Our techniques also yield a factor-2 approximation for the related problem of PACKING INTERVALS IN INTERVALS, also introduced by Winkler and Zhang in [3].

1 Introduction

Winkler and Zhang in [3] introduced the FIBER MINIMIZATION problem on a linesystem, which relates to the efficient construction of an optical fiber network to meet a specified collection of demands. Consider n links or edges connected in a line. The set of demands are intervals — each needs to use some consecutive links. An optical fiber, which is of wavelength μ and spans some consecutive links, can carry demands such that links required by the demands are contained within the links the fiber spans, and no two demands which require the same link are assigned the same wavelength. The goal is to get a set of fiber intervals such that the minimum total fiber length is used.

In Section 2.1, we give a 2-approximation algorithm for the FIBER MINIMIZATION problem. This problem is similar to the Dynamic Storage Allocation (DSA) problem in some aspects, and we employ techniques from the DSA literature, such as those of Gergov [2] and Buchsbaum et al. [1], to tackle it. Ideas from [1] yield an approximation scheme for this problem in Section 2.2.

We extend these results to obtain other approximations for related problems. In Section 3 give a factor $2+o(1)$ approximation algorithm for FIBER MINIMIZATION on a ring. We investigate the related problem of PACKING INTERVALS IN INTERVALS — also posed in [3] — in Section 4, and obtain a 2-approximation algorithm.

*Strategic Planning and Optimization Team, Amazon.com, Seattle, WA. email: vijayk@amazon.com

†Department of Computer Science and Engineering, University of Washington, Seattle, WA. email: atri@cs.washington.edu

2 The FIBER MINIMIZATION problem

Formally, consider a linesystem of n links $e_1, e_2, \cdots, e_n$. We represent each demand by $d_j = [l_j, r_j]$ for $l_j \leq r_j$ if it requires the links $e_{l_j}, \cdots, e_{r_j}$; the set of demands is denoted by D. We say demands d_j and $d_{j'}$ intersect (or overlap) if either $l_j \leq l_{j'} \leq r_j$ or $l_{j'} \leq l_j \leq r_{j'}$. A fiber interval f is represented by $f = [l_f, r_f]$ for $l_f \leq r_f$ if it spans the edges $l_f, \cdots, r_f$. The goal is to construct a set F of fiber intervals (each capable of carrying μ different wavelengths) of minimum total length $(\sum_{F \ni f=[l_f,r_f]}(r_f - l_f + 1))$ such that D can be packed in F. A packing of D in F is an assignment of each demand $d_j = [l_j, r_j]$ to a fiber $f = [l_f, r_f] \in F$, and a wavelength $\omega \in \{1, \cdots, \mu\}$ within f, such that $[l_j, r_j] \subseteq [l_f, r_f]$ and no two intersecting demands are assigned the same wavelength in the same fiber.

2.1 A 2-approximation algorithm

As specified above, the input D to the FIBER MINIMIZATION problem consists of demands d_j, each of which is an interval of the form $[l_j, r_j]$, where $l_j \leq r_j \in \{1, \cdots, n\}$. For each link e_i, we define $L_D(i) = |\{d_j \in D : i \in [l_j, r_j]\}|$ (WLOG we assume that $L_D(i)$ is a multiple of μ) and $L_D^{max} = \max_i L_D(i)$. The algorithm uses a $L_D^{max} \times n$ matrix H to keep track of wavelength assignments. For each $k \in \{1, \cdots, L_D^{max}\}$, the row H_k is referred to as the k^{th} *row* and finally would correspond to a wavelength in some fiber interval. We say that link e_i is colored c_i in row k if $H_{k,i} = c_i$. For any row k and color c, we call a interval $[l, r]$ in the kth row a c *segment* if $H_{k,l-1} \neq c$, $H_{k,r+1} \neq c$ and $\forall i \in [l, r]$, $H_{k,i} = c$.

The algorithm, which we refer to as the **Simple** algorithm, is specified in Figures 1 and 2. In **Phase 1**, the algorithm constructs the matrix H and then derives a "packing" P_D from H. We say that demand $d_j \in D$ is packed in P_D if $< j, k > \in P_D$ for some $k \in \{1, \cdots, L_D^{max}\}$. P_D is not a feasible solution initially — the assignments of some demands overlap, but only to a limited extent. In **Phase 2**, the overlaps are taken care of and a valid packing derived.

In a nutshell, this is how the algorithm works. At the beginning of Phase 1 (Figure 1), each matrix entry $H_{k,i}$ is colored either *red* (which means that no demand can be placed on edge e_i in the kth row), *green* (which means that atmost one demand can be placed on edge e_i in the kth row) or *blue* (which means overlapping demands can be placed on edge e_i in the kth row). Each edge e_i is "alloted" $L_D(i)$ rows, which is why the first $L_D^{max} - L_D(i)$ rows of e_i are colored *red* in Step 2. All other intervals are colored *green*. Initially all green segments are "available". In Step 3, the algorithm iterates over all possible rows and in each iteration looks for an "available" green segment over which some "unpacked" demand can be placed (maybe partially) in the following way: this demand can possibly intersect blue segments in that row but it must not intersect with any other green segment. The placement of a demand can fragment an "available" green segment into smaller "available" green segments (the edges common to the placed demand and the green segment are no longer available). The iteration is complete when no demand can be placed on any available green segment. Edges which were colored blue in the current row or did not have any demand placed on them in the current row are colored blue in the next row. Note that this implies that if an edge becomes blue in one row then it remains blue for all the subsequent iterations. Phase 1 of the algorithm is complete when D_{max} iterations are complete. We will show in Lemma 4 that after the first phase, all demands are "packed".

Further in Lemma 5 we show that a demand may intersect, if at all, with no more than one

demand in a blue segment. This suggests Phase 2 (Figure 2) of the algorithm where demands are finally packed into fiber intervals. Consider μ consecutive rows and consider maximal intervals of consecutive edges which are not colored red in any of the μ rows. In each such segment, every one of the μ rows has, by Lemma 5, at most two demands conflicting over any particular link. Thus, creating two fiber intervals corresponding to such a segment is sufficient to accommodate all the demands.

Phase 1

1. $J \leftarrow D$, $P_D \leftarrow \emptyset$.
2. **for** $(k \leftarrow 1; k \leq L_D^{max}; k \leftarrow k+1)$
   for $(i \leftarrow 1; i \leq n; i \leftarrow i+1)$
   $H_{k,i} \leftarrow$ *red* if $k \leq L_D^{max} - L_D(i)$; otherwise $H_{k,i} \leftarrow$ *green*.
3. **for** $(k \leftarrow 1; k \leq L_D^{max}; k \leftarrow k+1)$
   (a) $G \leftarrow \{[l,r] : H_{k,j}$ is green for all $j \in [l,r]$; $[l,r]$ is maximal$\}$ (G is the set of all maximal green intervals in the kth row)
   (b) **while** $\exists\ [l,r] \in G$
      i. **if** $\exists\ d_j = [l_j, r_j] \in J$ such that (d_j is an unpacked demand)
         A. $[l,r] \cap [l_j, r_j] \neq \emptyset$ (d_j intersects $[l,r]$), and
         B. $\forall w \in [l_j, r_j]$, either $w \in [l,r]$ or $H_{k,w} =$ *blue* (d_j intersects no other interval in G)

      then
         Add $< j, k >$ to P_D, delete d_j from J, and for all $i \in [l_j, r_j] \cap [l,r]$ set $H_{k+1,i} =$ green.
         delete d_j from J
         delete $[l,r]$ from G
         if $l < l_j$ add $[l, l_j]$ to G
         if $r_j < r$ add $[r_j, r]$ to G

      else
         for all $i \in [l,r]$ set $H_{k,i} =$ *blue*.
   (c) **for** $(i \leftarrow 1; i \leq n; i \leftarrow i+1)$
      if $H_{k,i} =$ *blue* **then** set $H_{k+1,i} \leftarrow$ *blue*.

Figure 1: Phase 1 of **Simple** algorithm

The following three Lemmas follow directly from the way the matrix H is manipulated in the algorithm of Figure 1.

Lemma 1 *At the end of Phase 1, if* $H_{k,i} =$ green *for any* k *and* i, *then for all* $1 \leq k' < k$, $H_{k',i} =$ green *or* $H_{k',i} =$ red.

Proof: If $H_{k',i}$ is colored blue, then $H_{j,i}$ gets colored blue for all $j > k'$ due to repeated execution of Step 3(c); in particular $H_{k,i}$ gets colored blue, which contradicts the assumption on $H_{k,i}$ in the lemma. ■

$F_D \leftarrow \emptyset$.
for $(k = 1; k \leq L_D^{max}; k = k + \mu)$

1. $\forall i \in \{1, \cdots, n\}$, $c[i] \leftarrow black$;
2. $\forall i \in \{1, \cdots, n\}$, if $\exists j \in [k, k + \mu)$ such that $H_{j,i} = red$ then $c[i] \leftarrow red$.
3. for each maximal interval $I = [l, r] \subseteq [1, n]$ such that $i \in [l, r] \Rightarrow c[i] = black$,
   (a) Create two fiber segments $f_1(I)$ and $f_2(I)$ and add them to F_D.
   (b) **for** all $j \in [k, k + \mu)$,
      i. Let $S_j \leftarrow \{d_i : d_i \subseteq I, < i, j > \in P_D\}$. Let $s_{j,1}, s_{j,2}, \cdots, s_{j,N}$ be an ordering of S_j such that for any two demands $s_{j,a} = [l_a, r_a]$ and $s_{j,b} = [l_b, r_b]$, $a < b \Rightarrow r_a \leq r_b$.
      ii. **for** $i = 1, 2, \cdots, N$,
         - Assign $s_{j,i}$ the wavelength $j - k + 1$.
         - Assign $s_{j,i}$ the fiber segment $f_2(I)$ if there exists some demand $s_{j,u}$ assigned wavelength $j - k + 1$ and fiber segment $f_1(I)$, such that $s_{j,i} \cap s_{j,u} \neq \emptyset$; otherwise, assign $s_{j,i}$ the fiber segment $f_1(I)$.

Figure 2: Phase 2 of **Simple** algorithm

Lemma 2 *At the end of Phase 1, if* $H_{k,i} =$ red *for any* k *and* i*, then for all* $1 \leq k' < k$, $H_{k',i} =$ red.

Proof: This is ensured by Step 2, where $H_{k,i}$ is colored red for all k less than a certain value. The color red is not employed at any other step in the algorithm; nor is it ever replaced by any other color. ■

Lemma 3 *At the end of Phase 1,* $L_D(i) = |\{k : H_{k,i} \neq \text{ red }\}|$.

Proof: This follows from how the coloring decision is made at Step 2, and the fact that the set $\{(k, i) : H_{k,i}$ is *red*$\}$ is invariant over the later steps of the algorithm. ■

We now show that all demands in D are "packed" in P_D after Phase 1.

Lemma 4 *At the end of Phase 1,* J *is empty.*

Proof: Assume that this is not the case, and there is a demand $d_t = [l_t, r_t] \in J$ at the end of Phase 1. One of the following three cases must arise:

Case 1 There is one $i \in [l_t, r_t]$ such that $H_{L_D^{max},i} =$ red. By Lemma 2, for all $k \in [1, L_D^{max}]$, $H_{k,i} =$ red. Lemma 3 implies that $D(i) = 0$, which contradicts the fact that d_t uses link i.

Case 2 There exists $i \in [l_t, r_t]$ such that $H_{L_D^{max},i} =$ green. It follows from Lemmas 1 and 2 and the coloring criterion of Step 2 that for all $k \in \{1, \cdots, L_D^{max} - L_D(i)\}$, $H_{k,i} =$ red; and for all $k' \in \{L_D^{max} - L_D(i) + 1, \cdots, L_D^{max}\}$, $H_{k',i} =$ green.

For each such k', it must be the case that in the k'th iteration of the **for** loop in Step 3, some demand $d_j \ni e_i$ was placed in row k', since otherwise $H_{k',i}$ would have been colored blue at Step 3(b). In all there would be $L_D(i)$ such demands, one for each value of k'. Including them, and including d_t, there are at least $L_D(i)+1$ demands that use link e_i, which contradicts the definition of $L_D(i)$.

Case 3 For all $i \in [l_t, r_t]$, $H_{L_D^{max},i}$ = blue. We complete the proof by showing that d_t would have been placed by the algorithm in some row. Consider $k^* = \min\{k : \forall j \in [l_t, r_t], H_{k,j} = \text{blue}\}$. By the choice of k^*, there exists an interval $[l, r] \subseteq [l_t, r_t]$ such that in some iteration of the **while** loop of Step 3(b),

- $[l, r] \in G$, and
- for all $i \in [l_t, r_t] - [l, r]$, $H_{k^*,i}$ =blue, and
- for all $j \in [l, r]$, $H_{k^*,j}$ is colored blue by the **else** clause of Step 3(b)(i).

This can only happen when there is no demand d_j which is suitable for placing over $[l, r]$ in row k^*: as indicated by conditions A and B in the **if** statement. However, d_t is precisely such a demand; it is unpacked, it intersects $[l, r]$, and [by the choice of k^*] it does not anymore intersect any other interval in G. Thus, instead of coloring $[l, r]$ blue, d_t should have been placed over it; and this completes our proof by contradiction. ■

We now show that the packing P_D has a nice property: no link is used by more than two demands that are placed in the same row. In other words, no three demands conflict simultaneously.

Lemma 5 $\forall (i, k)$, $|\{j :< j, k >\ni e_i\}| \leq 2$.

Proof: It is easy to see from the way intervals are added to and deleted from G that only one demand is placed on a green segment, that is, demands do not overlap over green segments. Thus, it follows that overlaps can only take place over blue segments (as no demands are placed over red segments).

Consider such a segment $[l, r]$ in row k. Among the demands that are placed over this segment, there can be at most one demand that contains $H[k, l-1]$, and at most one demand that contains $H[k, r+1]$ — that is because both $H[k, l-1]$ and $H[k, r+1]$ are non-blue and thus can not have overlapping demands placed on them. By placement rules [3(b)i.A.] and [3(b)i.B.], no demand is contained within a blue segment. Thus, no more than two demands can be placed over $[l, r]$ in row k. ■

We next show that Phase 2 outputs a valid solution.

Lemma 6 *Phase 2 (Figure 2) produces a valid packing of D in the set of F_D.*

Proof: Consider a demand $d_i = [l_i, r_i]$. Let d_i be placed in row j where $j \in [(h-1)\mu, h\mu)$ for some j and h — that is, $< i, j >\in P_D$.

First of all, let us verify that d_i is assigned a fiber interval in Phase 2. Consider the hth iteration of the **for** loop (Figure 2) in Phase 2. At Step 3(a), an interval $I \ni [l_i, r_i]$ would indeed be created,

except if $c[u] = black$ for some $u \in [l_i, r_i]$. Is that possible? For that to be the case, there must exist some $j' \in [(h-1)\mu, h\mu)$ such that $H_{j',u} = red$.

Recall how certain elements of H are colored *red* at Step 2 in Phase 1 (Figure 1). If $H_{j',u}$ is *red*, then $H_{v,u}$ is *red* for all $v \leq L_D^{max} - L_D(u)$, and it is not *red* for any other v. Since $L_D^{max} - L_D(u)$ is a multiple of μ (we have assumed the load at any link to be a multiple of μ), it follows that $H_{v,u}$ is *red* either for all $v \in [(h-1)\mu, h\mu)$, or for none of those values of v. In the latter case, the desired fiber interval I is indeed created; while the former case is easy to rule out, since it implies that $H_{j,u}$ is *red* as well, which is not possible given that $< i, j > \in P_D$ and $u \in [l_i, r_i]$ (no d_i can be placed over a *red* interval in Phase 1).

Note that demand d_i is assigned wavelength $j - (h-1)\mu + 1$, and one of the fibers $f_1(I)$ and $f_2(I)$. It remains to be verified that no other demand is assigned the same wavelength and the same fiber. We do this by pointing out that the set of demands that have been assigned wavelength $j-(h-1)\mu+1$ on $f_1(I)$ or $f_2(I)$ is exactly the set of demands d_k for which $< k, j > \in P_D$. This set of demands has been characterized by Lemma 5: any given link is contained in no more than two of these demands. This set of demand can be thought of as a collection of line intervals. Packing them into $f_1(I)$ and $f_2(I)$ without conflict is akin to 2-coloring the corresponding interval graph (whose clique number is 2). It is well-known and easy to see that a legal 2-coloring can be obtained simply by scanning the intervals left-to-right, and greedily assigning them one of two colors. Note that this is precisely what Step 3(b)ii of Phase 2 (Figure 2) attempts to do. ■

Let L be defined as $\frac{\sum_{i=1}^{n} L_D(i)}{\mu}$. For any set F of fiber intervals, let L_F be the total length of fiber used in F.

The following lemma shows that F_D is a pretty good solution.

Lemma 7 $L_{F_D} = 2L$.

Proof: Let us denote a fiber interval f by $[l_f, r_f]$, and for any link e_i, let $L_{F_D}(i) = |\{f \in F_D : i \in [l_f, r_f]\}|$. We will prove a stronger claim: for all e_i, $L_{F_D}(i) = 2\frac{L_D(i)}{\mu}$. The lemma follows by summing over all links.

Consider Step 3 of Phase 2 (Figure 2), where fiber intervals are created in pairs ($f_1(I)$ and $f_2(I)$). This step is repeated in the L_D^{max}/μ iterations of the **for** loop. A link e_i will not be contained in these fiber intervals for iterations $1, 2, \cdots, \frac{L_D^{max} - L_D(i)}{\mu}$, and included in both intervals of a pair ($f_1(I)$, $f_2(I)$) in all subsequent iterations. This is because $H_{v,i}$ is *red* for all $v \leq L_D^{max} - L_D(i)$, and not *red* for any other v, as we saw in the proof of Lemma 6. This implies that during the first $\frac{L_D^{max} - L_D(i)}{\mu}$ iterations, $c[i]$ is *red*, and thus the fiber intervals created do not include link e_i. In other words, link e_i is contained in exactly $2\frac{L_D(i)}{\mu}$ fiber intervals. ■

Clearly, L is a trivial lower bound on the length of the optimal set of fiber intervals for D. Thus:

Theorem 1 *Our algorithm is a 2-approximation algorithm for the FIBER MINIMIZATION problem.* ■

2.2 An Approximation Scheme

In this section we employ the *boxing* technique of Buchsbaum et al. [1] to the FIBER MINIMIZATION problem. The application of a result of [1] yields an approximation scheme for our problem. Let us first briefly describe *boxing*, which is applied in [1] to the Dynamic Storage Allocation problem. Let Z be a set of *jobs*, where each job i is a triple of start time l_i, end time r_i and height h_i. To *box* Z means placing the jobs in a box b starting from time $l_b = \min\{l_j : j \in Z\}$, ending at time $r_b = \max\{r_j : j \in Z\}$ and of height $h_b \geq \sum_{j \in Z} h_j$. A *boxing* of Z into a set B of boxes is a partition of Z into $|B|$ subsets, each of which is then boxed into a distinct $b \in B$. At any time t, let $L_Z(t)$ denote $\sum_{j \in Z: l_j \leq t \leq r_j} h_j$, and let $L_B(t) = \sum_{b \in B: l_b \leq t \leq r_b} h_b$.

We will be working with jobs of unit height, for which the algorithm in Section 2.1 of [1] comes with the following performance guarantee:

Theorem 2 *[1] Given a set Z of jobs, each of height 1, an integer box-height parameter H, and a sufficiently small positive ϵ, there exists a set B of boxes, each of height H, and a boxing of Z into B such that for all time t:*
$L_B(t) \leq (1 + 4\epsilon)L_Z(t) + O(\frac{H \log H}{\epsilon^2} \log \frac{1}{\epsilon})$.

The key insight here is that each demand in the FIBER MINIMIZATION problem can be viewed as a job of unit height, and packing of demands into fiber intervals is analogous to the boxing of a collection of jobs. This leads us to the following bound, where $L_F(t)$ denotes as before the number of fibers in F containing the link e_t and $L_D(t)$ denotes the number of demands in D using link e_t.

We will use the following lemma:

Lemma 8 *Given a set D of demands and a sufficiently small positive ϵ, there exists a set of fiber intervals F and a packing of the demands of D into F such that for all links e_t:*
$L_F(t) \leq (1 + 4\epsilon)\frac{L_D(t)}{\mu} + O(\frac{\log \mu}{\epsilon^2} \log \frac{1}{\epsilon})$.

Proof: A straightforward reduction maps an instance D of the FIBER MINIMIZATION problem to an instance of boxing. Corresponding to every demand $[l, r]$ in D, let there be a job with a start time of $l - 1$ and end time of r. Each link e_i is mapped to the time interval $[i - 1, i]$, and fiber intervals of wavelength $\mu = H$ map to boxes of height $H = \mu$.

Consider the boxing algorithm in Section 2.1 of [1]. As observed above, a demand $[l, r]$ in D corresponds to a job $(l - 1, r)$ in the DSA setting. Further note that a box of height μ in the DSA setting corresponds to a fiber interval. The set of fiber intervals F and set of demands D map directly to the set B of boxes and set Z of jobs, respectively, in the boxing instance. Observe that $L_D(t) = L_Z(t)$ and $L_F(t) = \frac{L_B(t)}{\mu}$.

The lemma follows directly from an application of Theorem 2 to the boxing instance Z with $H = \mu$. ■

Noting that $L_F = \sum_{i=1}^{n} L_F(i)$ is the total length of fiber used by the algorithm, and $L_D = \frac{\sum_{i=1}^{n} L_D(i)}{\mu}$, we have

Theorem 3 $L_F \leq (1 + 4\epsilon)L_D + O(n)$ *for sufficiently small positive constant ϵ.*

Proof: Using Lemma 8 and summing over all links. The last term on the right hand side in Lemma 8 is a constant, which leads to the $O(n)$ upon summation. ■

As in Section 2.1, we observe that the length of F^*, the optimal set of fiber intervals, $L_{F^*} \geq L_D$. Thus, algorithm of [1] has a competitive ratio arbitrarily close to 1 provided $L_D = \Omega(n)$.

Next, we look at the $O(n)$ term in Theorem 3 more carefully.

2.3 A Lower Bound

It is easy to see that the $O(n)$ additive term in the statement of Theorem 3 cannot be done away with for any approximation scheme. Consider the set D_{bad} of demands over $2n+1$ links. D_{bad} contains one copy of demand $[1, 2n+1]$, and $\mu-1$ copies each of demands $[1, n+1]$ and $[n+1, 2n+1]$.

Clearly, $L_{D_{bad}} \leq 2n+2$ while L_{F^*}, the value of the optimal solution, is $3n+2$. That is, there can be no positive constant $\delta < \frac{1}{2}$ such that $L_{F^*} < (1+\delta)L_{D_{bad}} + o(n)$ for all n.

3 FIBER MINIMIZATION in a Ring

Next, we look at what is perhaps the most natural generalization of the FIBER MINIMIZATION problem. Ring topologies are very commonly encountered and widely studied in optical routing literature. Consider a ring of n links where each demand d_j in the demand set D is an arc $[l_j, r_j]$ which requires links $e_{l_j}, e_{l_j+1 \bmod(n)}, \cdots, e_{r_j}$. Fiber intervals are now arcs each of which can support μ different wavelengths. The goal is to find the set of fiber arcs with the minimum total length.

The straightforward technique of partitioning an arc coloring problem into two interval coloring problems and taking the union of the two solutions would seem to directly obtain an approximation ratio of twice of that of the approximation ratio for the problem on a line system (that is, a ratio of four in this case). However, a small tweak gives a $(2+\epsilon)$-approximation ratio algorithm with two invocations of the **Simple** algorithm of Figures 1 and 2. Arbitrarily pick a link e_i and consider the set of demands using link e_i, $D^i = \{d_j \in D : i \in \{l_j, l_j + 1 \bmod(n), \cdots, r_j\}\}$. Now run the **Simple** algorithm on both D^i and $D - D^i$ to get sets of fiber intervals (arcs) F_{D^i} and F_{D-D^i}. Define $F_D = F_{D^i} \cup F_{D-D^i}$. Due to (possible) rounding[1] "errors" we now have for each link e_i, $L_{F_D}(i) \leq 2L_D(i) + 1$. Thus we have:

Theorem 4 *The combined algorithm gives $L_{F_D} \leq 2L_{F_D^*} + n$, where F_D^* is the optimal set of fibers arcs for D.*

Theorem 4 implies that if $L_{F_D^*} \geq \frac{1}{\epsilon}n$, then the combined algorithm achieves an approximation ratio of $2+\epsilon$.

The ideas developed in the paper so far can be applied to the related problem of PACKING INTERVALS IN INTERVALS [3].

[1]For each arc e_i atmost 2 fiber intervals containing e_i from the two solutions can be merged in the solution for the original problem.

4 PACKING INTERVALS IN INTERVALS

Winkler and Zhang also introduced the PACKING INTERVALS IN INTERVALS problem in [3]. Here we are given a set of demands D and a set of fiber intervals F, and the goal is to determine if D can be packed in F. Consider the optimization version of this decision problem. What is the smallest value of μ such that F can accommodate D? Our techniques imply a 2-approximation algorithm for this problem in the following sense:

Theorem 5 *If D can be packed in F using no more than $\frac{\mu}{2}$ wavelengths in each fiber, then there exists an algorithm which can pack D in F utilizing no more than μ wavelengths in each fiber.*

Proof outline: A small modification to the **Simple** algorithm is required. Let F be the set of fiber intervals and for each edge e_i define $L_D(i) = \frac{\mu}{2} |\{f : F \ni f = [l_f, r_f] \text{ and } i \in [l_f, r_f]\}|$. As in Section 2.1, L_D^{max} is defined as $\max_i L_D(i)$. Run Step 1 and 2 of Figure 1 with these values of $L_D(i)$ and L_D^{max}. Run Step 3 of Figure 1 with the given set of demands D. Execute Phase 2 (Figure 2 using $\frac{\mu}{2}$ in place of μ, and in the output F_D merge each $(f1(I), f2(I))$ pair.

Using arguments similar to ones used to prove Lemma 4, one can show that if D can be packed in F using no more than $\frac{\mu}{2}$ wavelengths in each fiber then J is empty after the execution of Phase 1 . Similarly, analogues of Lemmas 5, 6 and 7 can be proved if the assumption of the theorem statement is valid. The definition of $L_D(i)$ and the fact that the constructed F_D is actually F completes the proof. A detailed proof is deferred for lack of space. ■

5 Conclusions

We presented a clean 2-approximation algorithm for the FIBER MINIMIZATION problem on a linesystem. We also apply techniques from [1] to give an approximation scheme for this problem. Based upon our 2-approximation algorithm, we obtain good approximations for the related problems of FIBER MINIMIZATION on a ring and PACKING INTERVALS IN INTERVALS.

Interesting open problems include investigating the FIBER MINIMIZATION problem on other network topologies, particularly those common in optical fiber networks, such as trees and meshes.

References

[1] A. Buchsbaum, H. Karloff, C. Kenyon, N. Reingold, and M. Thorup. Opt versus load in dynamic storage allocation. In *Proc. of STOC 03*, 2003.

[2] J. Gergov. Algorithms for compile-time memory optimization. In *Proc. of 10th SODA*, 1999.

[3] P. Winkler and L. Zhang. Wavelength assignment and generalized interval graph coloring. In *Proc. of SODA 03*, 2003.