3/2 Approximation Algorithm for the Channel Assignment Problem in All Optical Networks Supporting DWDM (Dense Wavelength Division Multiplexing)

Sumit Chopra [*] Neelima Gupta [†] S. N. Maheshwari[‡]

October 5, 2002

Abstract

In this paper, we address the issue of efficiently allocating wavelengths to communication requests in Dense Wavelength Division Multiplexing (DWDM) all optical networks. We study the bidirected binary tree topology for the network. We assume that the pattern of communication requests is such that all the requests are from leaf-to-leaf and each directed edge of the tree is fully loaded. For this specific instance of requests, we present a simple deterministic greedy algorithm that guarantees a $3/2-$ *approximation and improves upon the previously known bound of* $5/3$ *on the approximation factor.*

1 Introduction

Optical fiber is rapidly becoming the standard transmission medium for backbone networks, since it can provide the required data rate, error rate and delay performance necessary for high speed networks of next generation [1]. Networks using optical transmission and maintaining optical data paths through the nodes are called *all-optical networks.* In an all-optical network, once the data stream has been transmitted in the form of light, it continues without conversion to electronic form until it reaches its destination.

Optical Bandwidth is the number of available wavelengths. An important engineering problem is to establish communication between pairs of nodes so that the data stream travels on the same wavelength on all the links and the total number of wavelengths used is minimized; this is known as the *wavelength routing problem* [2].

[*]Department of Computer Science, Hansraj College, Delhi University, New Delhi 110007, India. spchopra@mantraonline.com

[†]Department of Computer Science, Hansraj College, Delhi University, New Delhi 110007, India. neelima_research@yahoo.com

[‡]Department of Computer Science and Engineering, IIT Delhi, New Delhi 110016, India. snm@cse.iitd.ernet.in

We model the underlying fiber network as a directed graph, where the vertices are the nodes of the network and the links are optical fiber connecting nodes. Communication requests are ordered pairs of nodes, which are to be thought of as transmitter-receiver pairs.

In this paper we consider the bidirected binary tree topology. Since in a tree topology the path between a pair of nodes is unique a call request r can be thought of as a directed path between two vertices x and y of the network. Given a set R of requests, the maximum number of paths (the maximum load) through any directed link is denoted by L_{max}. Two calls r and r' are said to be in *conflict* if their corresponding paths go through the same fiber link. Calls using the edge of the tree in different directions do not interfere with each other. The goal is to assign wavelengths to the set of requests such that no two calls that are in conflict get the same wavelength and the number of required wavelengths is minimized. Thus the wavelength routing problem reduces to *wavelength assignment problem.* We refer to wavelengths as colors, and we can view the wavelength assignment problem as a path coloring problem. Let OPT be the minimum number of needed wavelengths or colors. Then, we have $L_{max} \leq OPT$.

The path coloring problem in trees has been proved to be NP-hard in [3], thus the work on the topic mainly focuses on the design and analysis of approximation algorithms. Known results are expressed in terms of L_{max}, Raghavan and Upfal [4] studied the undirected version of the problem. The directed version was first considered in [5]. Mihail et al. presented approximation algorithms for trees, rings and trees of rings. They give an upper bound of $15/8 L_{max}$. Kaklamanis and Persiano [6] and independently Kumar and Schwabe [7] improved the upper bound to $7/4 L_{max}$. The best known upper bound is $5/3 L_{max}$ [8]. Erlebach and Jansen [3] have proved that the wavelength routing problem is NP-complete even for binary directed trees. Simple deterministic algorithms that achieve the $5/3 L_{max}$ upper bound in binary trees are presented in [10] and [11]. The best known lower bound for the problem is $5/4 L_{max}$ [7], i.e., there exists a binary tree T of depth 3 and a set of paths R of load L_{max} on T that cannot be colored with less than $5/4 L_{max}$ colors.

All the above algorithms are deterministic and greedy in the sense that they visit the tree in a top to bottom manner and at each node v color all paths that touch v and are still uncolored; moreover, once a path has been colored it is never colored again. In the contex of WDM routing, greedy algorithms are important as they are simple and more importantly, they are amenable of being implemented easily and fast in a distributed environment. Kaklamanis et al. [8] prove that no greedy algorithm can achieve better performance ratio than $5/3 L_{max}$.

In an attempt to beat the $5/3 L_{max}$ lower bound for deterministic greedy algorithms, Auletta et. al. [12] define the class of randomized greedy algorithms for the problem. They obtained the first randomized algorithms that uses at most $7/5 L_{max} + o(L_{max})$ colors for binary trees of depth $o({L_{max}}^{1/3})$ with high probability. They also presented an existential upper bound of $7/5 L_{max} + o(L_{max})$ that holds on any binary tree.

The online version of the problem has also been studied in which the requests are processed as and when they come. The performance of an online algorithm is measured in terms of *optimal offline algorithm.* An online algorithm is said to be ρ - competitive if $\rho = sup \frac{\text{Cost}_{\text{ON}}}{\text{Cost}_{\text{OPT}}}$. The problem has been studied for the following topologies: line, rings, trees and meshes. An $\Omega(n^{\epsilon})$ lower bound has been obtained for arbitrary networks by Bartal, Fiat and Leonardi [13]. The lower bound holds even for randomized algorithms. An $O(\log n)$ - competitive algorithm for directed tress has been proposed by several authors [14, 15]. An almost matching determinstic $\Omega(\log n/\log\log n)$ lower bound has also been proved by Bartal and Leonardi [14] .

In this paper we present a simple deterministic greedy algorithm for a specific instance of requests for a binary tree topology that uses no more than $3/2L_{max}$ colors. The pattern of communication requests is such that all the requests are from leaf-to-leaf and each directed edge of the tree is fully loaded.

2 The Algorithm

In this section we present a simple deterministic algorithm for the problem. The topology of the network which we consider is a bidirected binary tree. The set R of requests (source-destination pair) possesses two properties. First, all requests start and terminate at the leaves of the tree. Second, through each directed edge exactly L_{max} requests pass i.e., each directed edge of the tree is fully loaded. This network can be thought of as L_{max} independent networks, each of which is capable of supporting only one wavelength (color). Note that the topology of each of these networks is identical to the orignal network. We call these networks as *bins*. The idea is to pack the requests in these bins so that no two requests sharing an edge in the same direction are packed in the same bin. Notice that running a simple algorithm to compute maximum edge disjoint paths(MEDP) on a bin and repeating the process with the remaining set of requests will not work here. See for instance Figure 1. An MEDP algorithm will first put 8 paths in each of the first $L_{max}/2$ bins, then 4 paths in each of next $L_{max}/2$ bins and then 2 paths in each of the next L_{max} bins thereby using a total of $2L_{max}$ bins whereas we are looking for an algorithm that uses no more than $3/2L_{max}$ bins.

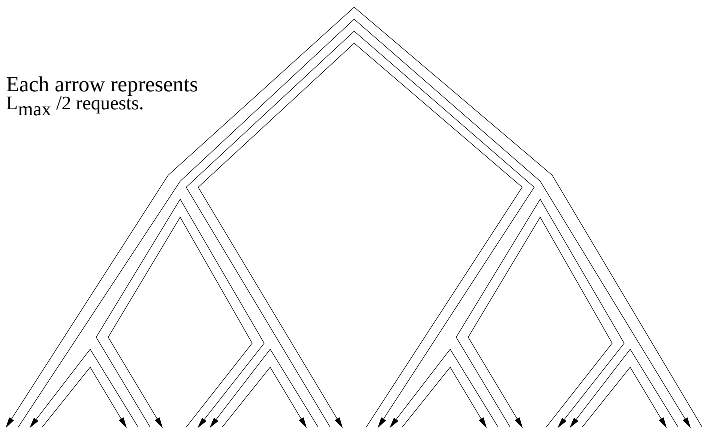

Figure 1:

Our algorithm is greedy and proceeds in phases in a BFS manner. In each phase we process the nodes at a particular level of the tree one by one.

Let v be any node of the tree. Let its parent, the left child and the right child be denoted by $p(v)$, $l(v)$ and $r(v)$ respectively. The paths that touch the node v and its two children are called the *medium paths* with respect to v. The paths that touch v and its parent node are called the *long paths* with respect to v. See Figure 2.

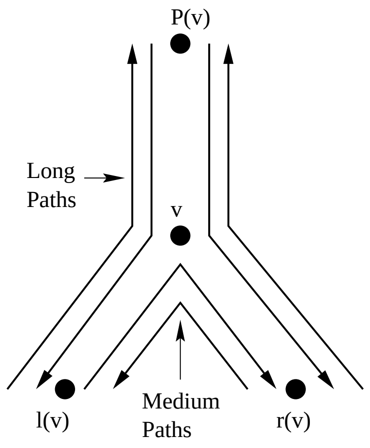

Figure 2:

When we process a node v all the long paths through v have already been assigned a bin i.e., they have already been colored. The aim is to pack the medium paths through v. Here, we classify our bins in three categories : One, that does not have a path through $(v, p(v))$ - we call such bins as *free* with respect to v. Second, that has a pair of paths, in opposite directions, through $(v, p(v))$ and the pair parts at v (i.e. one path traverses the edge $(v, l(v))$ and the other traverses $(v, r(v))$). Third, that has a pair of paths, in opposite directions, through $(v, p(v))$ and they do not part at v (i.e. either both traverse $(v, l(v))$ or both traverse $(v, r(v))$. A bin cannot have a single path through any edge.

Procedure process_v is repeated for every node in the tree in a BFS manner starting from the root.

```
Procedure process_v($v$)

1. For every bin of type two do
       Pack it with an edge disjoint medium path through $v$.

2. do
       Pick a pair of medium paths through $v$ in opposite directions and assign
       them to a bin of first type.
   until no more medium paths are left.
```

Notice that when we process the root, there are no long paths so the assumption that long paths have already been colored holds vacuosly, also all the bins are of type one i.e. all the bins are *free*.

Once all the medium paths corresponding to a node v at level i have been exhausted, the algorithm repeats the same procedure for all the other nodes at the same level and then proceeds to the next level.

3 Analysis

For the specific instance of leaf-to-leaf requests with each directed link being fully loaded, the picture at any node v of the tree is as shown in the Figure 3. Clearly, at least one of x and $L_{max} - x$ is $\leq L_{max}/2$. Without loss of generality we may assume $x \leq L_{max}/2$ (for otherwise we shall call $L_{max} - x$ as y, x becomes $L_{max} - y$ and we work with y).

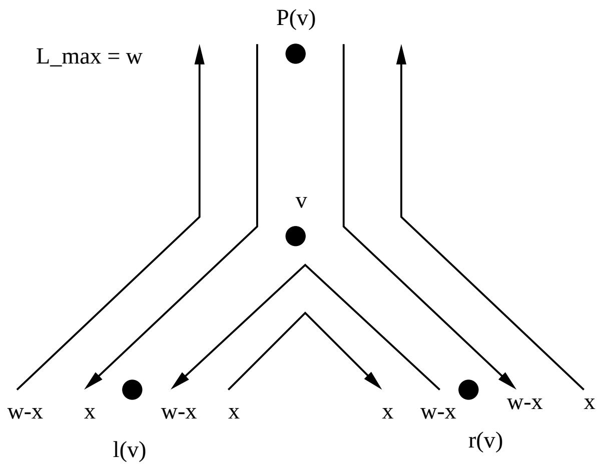

Figure 3:

Claim 3.1 *Paths through an edge occur in pairs in bins.*

Proof: The proof is by induction.

Consider paths through an edge $(v, p(v))$. Let $v' = p(v)$. Assume that all the paths through $(v', p(v'))$ occur in pairs in the bins. This is vacuosly true when v' is the root. See Figure 4.

Consider a bin B at v'. If B is of type two, then B has a single path through $(v, p(v))$, now at v' the algorithm assigns a single medium path (of course edge disjoint) to B and hence B gets a pair of paths through $(v, p(v))$. If B is of type one then we assign a pair of medium paths through v' (or assign no medium paths if we have exhausted all the pairs) and hence either a pair of paths or no paths through $(v, p(v))$. If B is of type three, then by classification either it has no path through $(v, p(v))$ or it has a pair of paths through $(v, p(v))$.

Claim 3.2 *We have sufficient number of medium paths to pack all bins of type two.*

Proof: Consider a node v as shown in Figure 3. The bins of type two are of two types. See Figure 5.

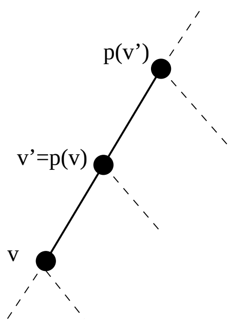

Figure 4:

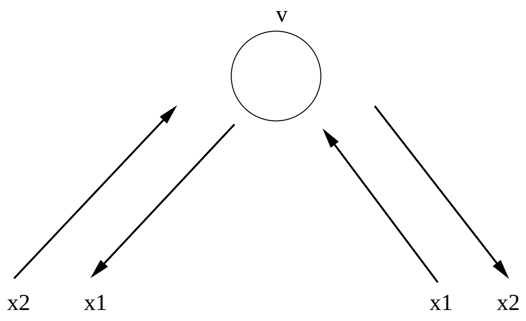

Figure 5:

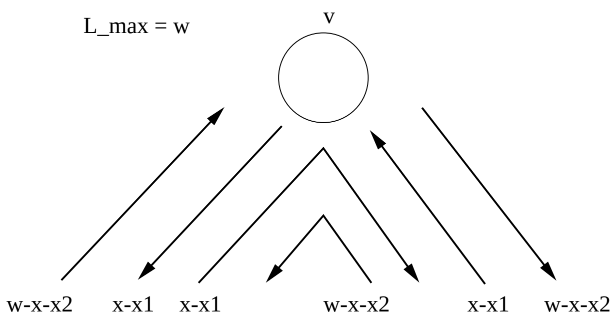

Figure 6:

The set B^1 of bins that have downward paths through $(v, l(v))$ and upward paths through $(v, r(v))$ and the set B^2 of bins that have upward paths through $(v, l(v))$ and downward paths through $(v, r(v))$. Let B^1 has x_1 bins and B^2 has x_2 bins. Then $x_1 \leq x$ and $x_2 \leq L_{max} - x$ (from Figure 3). Since we have x medium paths from $l(v)$ to $r(v)$ and $L_{max} - x$ medium paths from $r(v)$ to $l(v)$ the claim follows.

Claim 3.3 *Assume that we have $X = 3/2L_{max}$ number of bins. After packing bins of type two we are left with equal number of medium paths in opposite directions and the number of such pairs is no more than the number of bins of type one.*

Proof: The picture at node v after packing bins of type two is as shown in Figure 6.

Since all the long paths left are non-parting we must have $x - x_1 = L_{max} - x - x_2$. Hence the first part follows.

Next, since the paths are occuring in pairs, the number of bins having pairs of paths through $(v, p(v))$ is at most L_{max}. That is, the number of bins of type one is at least $L_{max}/2$. (This is the most important observation here which does not hold for the general instances.) Since there are exactly $x - x_1$ pairs of medium paths left through v and $x \leq L_{max}/2$, it follows that the number of pairs of medium paths is no more than the number of bins of type one.

4 The General Instance

In the previous section we obtained a greedy algorithm that gives an approximation factor of 3/2 for a fully loaded leaf-to-leaf instance. However this algorithm, as it is, cannot be used to color a general instance (not necessarily a leaf-to-leaf) or even to color a general leaf-to-leaf instance. And a general leaf-to-leaf instance cannot be converted to a fully loaded leaf-to-leaf instance. In fact it has been shown that for a general leaf-to-leaf instance as well as for a general instance, no local greedy can obtain an approximation factor better than $5/3L_{max}$. Hence an approach that is non-local in nature is required.

In this section we present our algorithm for the general instance in binary trees. Our algorithm has a non-local structure, in the sense that while coloring the paths touching a node, we simultaneously look at two consecutive levels.

We divide our algorithm in two steps, the Preprocessing step and the Coloring step.

4.1 The Preprocessing Step

This step is similar to the Preprocessing Procedure given in [12]. First, the general instance is transformed into a fully loaded instance by adding single-hop paths at the directed arcs that are not fully loaded. Next, the non-leaf nodes of the tree are traversed in a BFS manner. During this traversal, let us be at a node v. Consider the set of paths that touch node v. They are of two types, those that either terminate or orignate at v and those that pass through. Combine the pairs of paths of the form (v_1, v) and (v, v_2) to create one path (v_1, v_2) that passes through v. After doing this, the instance satisfies the following three properties [12]

1. Each directed link is fully loaded.

2. For every node v, paths that orignate or terminate at a node v appear on only one of the three arcs adjacent to v.

3. For every node v, the number of paths that orignate from v is equal to the number of paths that terminate at v.

Assumption: For simplicity we assume that the terminating-orignating paths appear only on the parent arcs of all the nodes. The case that they appear on the other two arcs will be handeled later.

4.2 The Coloring Step

The coloring step is based on the idea similar to the one used to color the fully loaded leaf-to-leaf instance. Note that the leaf-to-leaf coloring guarantees that at every edge, there are exactly two paths (in opposite direction) that are colored with the same color. However this same idea cannot be directly applied to the general instance. The problem is that if paths are allowed to terminate in between, then this pairing of paths through the edges is not guaranteed. When at a node v, if one tried to pair the paths in the same way as in leaf-to-leaf (pairing the long paths together and the terminating paths together), then it may happen that at one of the child we get a complete pairing (each color is used to color a pair of paths in opposite direction) and on the other child we have paths, each of which is colored with different color (i.e. no pairing at all). Thus it may happen that at a node a large number of colors show up and we may not have any control over the number of colors. See fig @@1. Thus what is needed is some form of balance on both the children.

When at a node v, we keep in mind the nature of paths that are going down the two children and those that are terminating at the children. We color the paths in such a way so that at each of the child node, roughly half of the terminating paths are paired (colored with the same color) with the long paths on that node. The remaining half of the long paths are paired amongst themselves and the terminating paths are paired amongst themselves. This way a sort of balance is maintained amongst the two children.

But note that, even with such a pairing we still may not be able to guarantee even a single pairing amongst the long paths of one of the child nodes i.e. all the long paths are colored differently. However in such a case we strongly conjecture (and also prove for number of cases) that the number of such paths, hence the total number of colors used at that node is bounded and that if such a scene happens, then at the next level, at the subtree rooted at that node, such a pairing is guaranteed to happen.

Clearly this type of coloring strategy is non-local. Since at a node v, we are using the information associated with the paths passing through the edges below the child nodes. Indeed we shall show that the adversay argument given in [11] fails in this case. Let us be at a node v. Let one of its child (say the right child) be denoted by v' and its two children by $l(v')$ and $r(v')$. At v', two types of paths appear, those that are terminating and orignating at v' and those that are going down (the long ones) to its two children. It may happen that all these paths are of different color. What the adversary does, is it tries to split these paths in such a way that any greedy algorithm is forced to to use new colors to color the medium paths at successive levels. However, we show that if number of such paths is bounded then we use no more than $3w/2$ colors at this level and that our strategy guarantees pairing of paths at the next level. Let the number of such paths be $2p$ (p paths in either direction). Let $2p < w$ (our conjecture). Then at worst what could happen, is that the adversary splits the paths in the way shown in fig @@2. All the down coming p paths go to one child of v' (say $l(v')$) and the p upgoing paths go to the other child ($r(v')$). And there will be $w-p$ paths that need to be colored with a new set of colors. But since $2p < w$, we have $(w-p) < w/2$. Thus the number of colors which we have used at this level is no more that $w + w/2 = 3w/2$. Once such a thing happens at this level, our strategy guarantees that some pairing will occur at the next level, hence the adversary argument will fail.

References

[1] P. E. Green. Fiber Optic Communication Networks. *Prentice-Hall*, 1992.

[2] A. Aggarwal, A. Bar-Noy, D. Coppersmith, R. Ramaswami, B. Schieber and M. Sudan. Efficient Routing and Scheduling Algorithms for Optical Networks. *Journal of the ACM*, Vol. 43, No. 6, 1996, pp. 973-1001.

[3] T. Erelebach and K, Jansen. Scheduling of Virtual Connections in Fast Networks. In *Proc. of the 4th Workshop on Parallel Systems and Algorithms*, pp. 13-32, 1996.

[4] P. Raghavan and E. Upfal. Efficient Routing in All-Optical Networks. *Proc. of the 26th Annual Symposium on Theory of Computing (STOC '94)*, 1994, pp. 133-143.

[5] M. Mihail, C. Kaklamanis and S. Rao. Efficient Access to Optical Bandwidth. *Proc. of the 36th Annual Symposium on Foundations of Computer Science (FOCS '95)*, 1995, pp. 548-557.

[6] C. Kaklamanis and P. Persiano. Efficient Wavelength Routing on Directed Fiber Trees. *Proc. of the 4th European Symposium on Algorithms (ESA '96)*, LNCS 1136, Springer Verlag, 1996, pp. 460-470.

[7] V. Kumar and E. Schwabe. Improved Access to Optical Bandwidth in Trees. *Proc. of the* 8th *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '97)*, 1997, pp. 437-444.

[8] C. Kaklamanis, P. Persiano, T. Erlebach and J. Jansen. Constrained Bipartite Egde Coloring with Applications to Wavelength Routing. *Proc. of the* 24th *International Colloquium on Automata, Languages, and Programming (ICALP '97)*, LNCS 1256, Springer Verlag, 1997, pp. 493-504.

[9] T. Erlebach and K. Jansen. Call Scheduling in Trees, Rings and Meshes. *Proc. of the* 30th *Hawaii International Conference on Systems Sciences*, 1997.

[10] I. Caragiannis, C, Kaklamanis and P. Persiano. Bounds on Optical Bandwidth Allocation on Directed Fiber Tree Topologies. 2nd *Workshop on Optics and Computer Science (WOCS '97)*, 1997.

[11] K. Jansen. Approximation Results for Wavelength Routing on Directed Trees. 2nd *Workshop on Optics and Computer Science (WOCS '97)*, 1997.

[12] V. Auletta, I. Caragiannis, C. Kaklamanis, and P. Persiano. Randomized Path Coloring on Binary Trees. In *Proc. of the* 3rd *International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX '00)*, LNCS 1913, Springer, pp. 60-71, September 2000.

[13] Y. Bartal, A. Fiat and S. Leonardi. Lower bounds for on-line graph problems with application to on-line circuit and optical routing. In *Proc. of the* 28th *ACm Symposium on Theory of Computing*, pp. 531-540, 1996.

[14] Y. Bartal and S. Leonardi. On-line routing in all-optical networks. In *Proc. of the* 24th *International Colloqium on Automata, Languages and Programming*, LNCS 1256, pp. 516-526. Springer-Verlag, 1997.

[15] O. Gerstal, G. H. Sasaki and R. Ramaswami. Dynamic channel assignment for WDM optical networks with little or no wavelength conversion. In *Proc. of the* 36th *Allerton Conference on Comminication, Control and Computin*, 1996.