On-Line Optimal Wavelength Assignment in WDM Networks with Shared Wavelength Converter Pool

Zhenghao Zhang and Yuanyuan Yang
Department of Electrical & Computer Engineering, State University of New York, Stony Brook, NY 11794, USA

Abstract— **In this paper we study on-line wavelength assignment in wavelength-routed WDM networks under both unicast and multicast traffic. We assume nodes in the networks have wavelength conversion ability. Since wavelength converters are still expensive and difficult to implement, we consider the networks that have only a limited number of converters in each node, and the converters are shared by all input channels at the node. We consider how to set up connections in such networks using as few wavelength converters as possible. For unicast traffic, we first study the problem of setting up a lightpath on a given link path with minimum number of conversions, and give a new algorithm that solves it in $O(tk)$ time, where t is the number of links on the path and k is the number of wavelengths per fiber, as compared to the best known existing algorithm that runs in at least $O(t^2k)$ time. We also consider the case when nodes have different conversion priorities, and give an $O(tk)$ time algorithm for setting up a lightpath on a given link path while converting wavelength at higher priority nodes only when necessary. We then generalize this technique to WDM networks with arbitrary topologies and present an algorithm that sets up an optimal lightpath network-wide in $O(Nk + Lk)$ time by checking the state of the entire network, where N and L are the number of nodes and links in the network, respectively. For multicast traffic, finding an optimal multicast light-tree is known to be NP-hard and is usually solved by first finding a link tree then finding a light tree on the link tree. Finding a link tree is also NP-hard and has been extensively studied. Thus, we focus on the second problem which is to set up a light tree on a given link tree with minimum number of conversions. We propose a new and more practical multicast conversion model, where the output of the wavelength converter can be split. As can be seen, the new model can save the usage of converters considerably. We first show that this problem is NP-hard and then give efficient heuristics to solve it approximately.**

I. Introduction and Background

Optical networks with wavelength division multiplexing (WDM) are now widely regarded as the backbone network for future communication networks because of the huge bandwidth of optical systems. In a WDM network, nodes are connected by optical fiber links. On each link there are multiple wavelengths carrying independent data. To transmit information from one node to another, a *lightpath* needs to be set up along the links connecting the source to the destination. Without *wavelength converters*, a lightpath must use the same wavelength throughout the path. With wavelength converters, a lightpath does not need to be on the same wavelength and can consist of several consecutive wavelength continuous segments, with wavelength conversion carried out at the junction nodes. It has been shown that by adding wavelength conversion ability network performance can be greatly improved [15].

However, at current time wavelength converters are still expensive and difficult to implement. Therefore, as suggested by [4], at an intermediate node, using a converter pool that can be shared by all input channels is more cost-effective than giving each input channel its own wavelength converter. This is because it is highly unlikely that every input wavelength will need wavelength conversion at the same time. The number of wavelength converters at a node can be far less than the total number of input wavelength channels, therefore, when setting up connections, it is desirable to use as few converters as possible. In this paper we study several related problems on wavelength scheduling under this scenario and give efficient algorithms to find wavelength assignment that uses less wavelength converters whenever possible for both unicast and multicast traffic. We focus on how to solve the problem *on-line* (or *dynamically*), which means that the traffic intensity between nodes is not previously known to the schedulers, and when a connection request comes, the scheduler seeks to find ways to satisfy it optimally based on the current network state. It is different from what is usually referred to as the Routing and Wavelength Assignment (RWA) problem, which can be regarded as the *off-line* or *static* version of our problem where the traffic intensity between every pair of nodes in the network is known and given in advance [14]. Apparently, the speed requirement for on-line scheduling is far more critical.

We first consider unicast traffic, i.e., there is one source and one destination in a connection request. The problem of finding lightpaths for unicast connection requests in WDM networks has been extensively studied in recent years, see, for example, [12], [9], [7], [4], [17]. It can be solved by breaking it into two subproblems: the routing problem which is to find a link path in the network connecting the source to the destination, and the wavelength assignment problem which is to find a light path on the link path [4], [17]. Alternatively, the problem can be solved by jointly considering the two subproblems [12], [9], [7]. The second approach will give better results but is very time consuming especially for a

0-7803-8968-9/05/$20.00 (C)2005 IEEE

large network. We will mainly follow the first approach. In particular, we will give new algorithms for the wavelength assignment problem.

The problem of finding a lightpath on a given link path is usually solved by applying the First Fit Algorithm which starts from the source node and finds the first available wavelength channel to reach to the next node. This may cause unnecessary wavelength conversions, which is especially undesirable in an environment where wavelength converters are scarce. In [4] the problem of finding a lightpath using minimum number of converters was studied and solved by constructing an auxiliary graph and then applying Dijkstra's algorithm, which has time complexity of at least $O(t^2k)$ where t is the number of nodes on the path and k is the number of wavelengths per fiber. In this paper we will solve exactly the same problem but in a completely different and more direct way without using auxiliary graphs, and the resulting algorithm, called the Longest Segment Algorithm, has linear time complexity of $O(tk)$. We will also consider the case when some of the nodes have higher conversion priorities than others, and study the problem of setting up a lightpath using converters in higher priority nodes only when necessary, and give an algorithm that runs in $O(tk)$ time as well.

We will also study similar problems as those in [9], [7] and give an algorithm that sets up an optimal lightpath network-wide by checking the state of the entire network. In [9], [7] a cost represented by a real number is assigned to each wavelength channel and also to each wavelength converter, and the optimal lightpath is defined as a lightpath with minimum total cost, including the wavelength channel cost and the conversion cost. What we study here is a special, albeit may be of more practical interest case of it. We assume that each wavelength channel has the same cost and each wavelength converter also has the same cost, since wavelength channels typically have the same bandwidth and *full range* wavelength converters are capable of converting a wavelength to any other wavelengths at similar cost. Furthermore, we assume that the costs of wavelength converters are much higher than the costs of wavelength channels, since the number of wavelength channels on a fiber link is growing very rapidly and 256 channels on a fiber have been reported, and propagation loss can be compensated by optical amplifiers which are much cheaper than wavelength converters. This reduces the problem in [9], [7] to finding a lightpath with minimum total number of conversions and, under this condition, minimum total number of hops. We will give an algorithm that runs in $O(Nk + Lk)$ time to solve this problem, where N and L are the number of nodes and links in the network, respectively. Note that in [9], [7] the more generalized problem needs at least $O(Nk^2 + Lk + Nk\log(Nk))$ time.

We will also consider multicast traffic, which is to send information from one source to multiple destinations. In a communication network, a multicast connection is usually realized by establishing a multicast tree covering all the nodes involved. Finding an optimal light tree in WDM networks is NP-hard [2], [1], and can be solved by breaking it into two sub-problems and solving them one by one: (1) find a link tree that covers all the nodes involved; then (2) find a wavelength assignment in this link tree to construct a light tree. Finding a link tree in a network is still NP-hard and has been extensively studied [2], [1], and we will focus on the second subproblem in this paper.

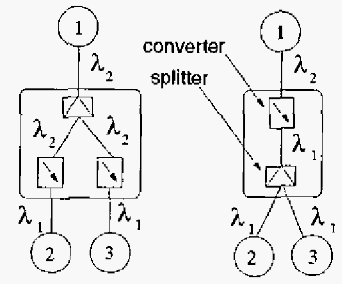


Fig. 1. Under Converter Split Model, one wavelength converter can be saved.

[2] gave a linear time algorithm to set up a light tree in a link tree using a dynamic programming method, but under a different scenario and did not try to minimize the total number of conversions. [1] gave a linear time algorithm to set up a light tree using minimum number of conversions. In this paper we will also consider the problem of finding a light tree with minimum number of conversions, but under a new and more practical multicast model. In [1], it is assumed that the output of the wavelength converter cannot be split. That is, if an intermediate node of the tree has m branches, the light signal is first split into m copies, and each copy is either sent directly into a branch or first converted to another wavelength by a *separate* wavelength converter and then sent into a branch. In this paper, we propose a new model which allows the output of the converter to split. This is not technologically difficult and does not increase the splitting cost defined in [1], however, as will be seen, the new model can reduce the conversion cost considerably. For example, in Fig.1, suppose node 1 wants to send information to nodes 2 and 3 through an intermediate node, but on the link between node 1 and the intermediate node only λ_2 is available and on the links from the intermediate node to nodes 2 and 3 only λ_1 is available. If the output of the converter cannot be split, we will have to use two wavelength converters, both converting λ_2 to λ_1, as shown in Fig.1(a). On the other hand, if the output of the converter can be split, we can use one wavelength converter to convert λ_2 to λ_1, then split the output of the converter and send them to the destinations, thus saving one converter, as shown in Fig.1(b). We will first show that when the output of the converter can be split, the optimal wavelength assignment problem is NP-hard and then give efficient heuristics to solve it approximately. We will call the multicast conversion model in this paper the "Converter Split Model" and call the model in [1] the "No Converter Split Model." It should be mentioned that the algorithm presented in this paper can also solve the wavelength assignment problem under the No Converter Split Model in linear time but its implementation is much simpler, since we can apply our Longest Segment Algorithm developed

for unicast to find the properties of tree branches in a more efficient and more explicit way and we only compute such properties for the branches of the tree while in [1], such properties were computed for every node.

The rest of the paper is organized as follows. Section II describes the algorithms for unicast traffic, including the algorithm for setting up a light path on a link path with minimum number of wavelength conversions, the algorithm for setting up a light path on a link path when nodes have different conversion priorities, and the network wide routing and wavelength conversion algorithm. Section III describes the algorithms for multicast traffic. Section IV concludes the paper.

II. Wavelength Scheduling for Unicast

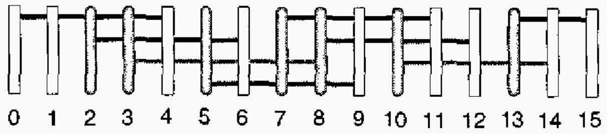
In a network, we define a *path* as several consecutive links. If wavelength λ_i is currently unused on several consecutive links, those wavelength channels on λ_i are called a *wavelength continuous segment on* λ_i. A *lightpath* is defined as several consecutive wavelength continuous segments, with the constraint that the junction node where two segments join should be wavelength convertible.

For example, Fig.2(a) shows a link path with 16 nodes and 4 wavelengths per fiber. λ_0 to λ_3 are represented by blue, red, green, and purple line segments, respectively. Wavelength convertible nodes are shown in rectangles with round corners. There is a wavelength continuous segment on the first wavelength, λ_0, from node 0 to node 4. The lightpath connecting node 0 to node 15 found by the First Fit Algorithm is shown in Fig.2(b), where wavelength channels chosen by the algorithm are shown in wider line segments and nodes that perform wavelength conversions are shown as rectangles with heavy edges.

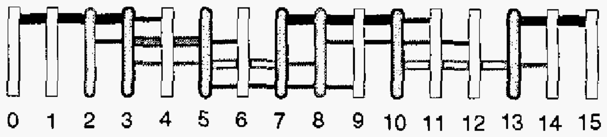
A. Longest Segment Algorithm for Setting up a Lightpath with Minimum Number of Conversions

We now give the Longest Segment Algorithm for setting up a lightpath along a given link path with minimum number of conversions. First we introduce some notations. We first give the source node index 0. Then any other node on this path is denoted by the number of links from the source to it. In this way, assuming there are t links between the source and the destination, the nodes are indexed as $0, 1, \dots, t$. We say node u *finds* node v if $v > u$ and there exists a wavelength continuous segment from u to v. We define *furthest reachable wavelength convertible node* from a node u, or abbreviated as the *FRC* node, as the furthest wavelength convertible node that can be found by u. For example, in Fig.2(a), node 0 finds node 4 because there is a wavelength continuous segment on λ_0 from 0 to 4. The FRC node from node 0 is node 3.

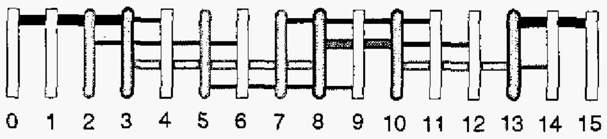
The algorithm is described in Table 1. It is a greedy algorithm. In each step, it will try to find the longest wavelength continuous segment starting at current *extending point* x. Initially, x is set to be the source node. If the destination is reached, the algorithm returns; otherwise, it sets the FRC node as the next extending point and repeats. Every FRC node performs wavelength conversion.



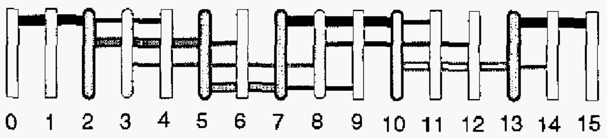
(a) A link path with 16 nodes.



(b) Assignment found by the First Fit Algorithm.



(c) Assignment found by the Longest Segment Algorithm.



(d) Assignment found by the Label Extending Algorithm.

Fig. 2. The state of a link path with 16 nodes and assignments found by different algorithms. .

TABLE 1
LONGEST SEGMENT ALGORITHM

```
$x \leftarrow 0$
while $x$ cannot reach $t$
        find $u$, the FRC node from $x$.
        if no such $u$ exists
                exit the while loop
        end if
        $x \leftarrow u$
end while
```

For example, the assignment found by the Longest Segment Algorithm for the link path in Fig.2(a) is shown in Fig.2(c). In the first step, node 0 can find as far as 4 on λ_0, and the FRC node is 3. Therefore the extending point at the next step is 3. Then 3 can find as far as 8 on λ_2, and 8 is the FRC node and will be the next extending point. This is carried on until 15 is found. Note that one fewer converter is used than that in Fig.2(b).

Theorem 1: The Longest Segment Algorithm finds an lightpath on a given link path using the minimum number of converters.

Proof. If the destination t can be found by the source 0, the algorithm will find a wavelength continuous segment to connect 0 to t. Therefore the claim is true if no wavelength conversion is needed. In the following we consider the case when wavelength conversion has to be used. Let the lightpath found by the algorithm be Ψ. Ψ will consist of several, say,

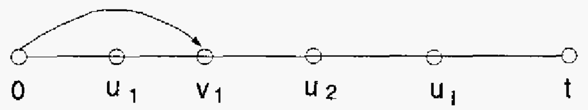


Fig. 3. If node 0 can reach node v_1 and v_1 is wavelength convertible, u_1 cannot be the FRC node of node 0.

$I+1$, wavelength continuous segments, and we denote them by $[0, u_1]$, $[u_1, u_2]$, ..., $[u_I, t]$, with wavelength conversion at u_i, $i \in [1, I]$, as shown in Fig.3. To show this algorithm is indeed optimal, let Φ be any other lightpath. Now consider the first wavelength continuous segment in Ψ. We claim that from node 0 to u_1, Φ has at least one wavelength conversion. Suppose the claim is not true. Then on path Φ none of the nodes 0, 1, ..., u_1 convert wavelength. Since Φ has to use some wavelength conversions to connect 0 to t, the conversions are carried out by nodes with longer distances to 0 than u_1. Let the first such node be v_1. We have $v_1 > u_1$, and 0 finds v_1 by a wavelength continuous segment. This contradicts the fact that u_1 is the FRC node of 0.

Next consider the second wavelength continuous segment of Ψ, $[u_1, u_2]$. We claim that from u_1+1 to u_2, Φ has at least one wavelength conversion. Suppose this is not true. Let v_1 be a node where Φ converts wavelength and $v_1 \le u_1$. From previous discussions we know that there must exist such a node, and let v_1 be the one closest to u_1. If there is no conversion from u_1+1 to u_2, the next wavelength conversion node in Φ following v_1, denoted by v_2, satisfies $v_2 > u_2$. There must be a wavelength continuous segment between v_1 and v_2. Therefore, there must also be one between u_1 and v_2. This contradicts the fact that u_2 is the FRC node of u_1.

This argument can be carried on, and will lead to the conclusion that in each of the first I segments of Ψ, Φ uses at least one wavelength conversion. Therefore the number of conversion used in Ψ is no more than that in Φ. Since Φ can be any lightpath, Ψ also uses no more wavelength conversions than the optimal lightpath. Hence, Ψ is an optimal lightpath. ■

We can represent the state of each link by a $1 \times k$ binary vector, where an element being '1' means the corresponding wavelength channel is available on that link. We use $L_{x,x+1}$ to represent the link vector of the link between node x and $x+1$. To check whether t can be found by a node x, we only need to perform a series of *AND* operations, starting with an all '1' vector and then *AND* with $L_{x,x+1}$, then with $L_{x+1,x+2}$, $L_{x+2,x+3}$, until the result becomes all zero or until t has been found. At the same time, while "moving forward", we can check whether a node is wavelength convertible, and always keep the FRC node as the one that was reached most recently. If the *AND* result becomes all zero at v, we start the *AND* operation from u, the FRC node. Note that $L_{u,u+1}$ to $L_{v,v+1}$ will be involved in the *AND* operation one more time. However, they will not be involved in the next and following rounds, since the FRC node of u must have a larger index than v. Hence, a link state vector is *ANDed* no more than twice, and the running time for finding the total number of conversions is $O(t)$. To set up the lightpath, we need to find wavelength continuous segments to connect the successive extending points. This can be done by scanning through the elements of the *AND* results and choose the first '1'. As there are k wavelengths and the number of extending points cannot exceed t, the running time for setting up a lightpath is $O(tk)$.

B. Label Extending Algorithm for Setting up a Lightpath when Nodes Have Different Conversion Priorities

So far we have considered finding a lightpath using minimum number of conversions. There are situations when some of the wavelength convertible nodes should convert wavelength only when extremely necessary. For example, when a node has very few converters left, to set up a lightpath, it is intuitively better to convert wavelength at other nodes whenever possible. Another case is when the network is hybrid: some of the nodes have full conversion ability, i.e, have a number of converters equal to the number of input channels; some have partial conversion ability, i.e., have a limited number of converters less than the number of input channels; and some have no conversion ability. In this case, we should use wavelength converters at nodes with full conversion ability whenever possible.

This problem can be formalized as follows. Categorize wavelength convertible nodes into two classes: one with higher conversion priority called the "critical nodes" and the other with lower conversion priority called the "non-critical nodes". In the first case discussed above, a node is critical when the number of converters left is lower than a threshold. In the second case, a node is critical if it only has partial conversion ability. An assignment is then measured by a pair of integers, (C, N), where C is the number of critical conversions along the path and N is the number of non-critical conversions along the path. We say assignment 1 is better than assignment 2 if (C_1, N_1) is lexicographically smaller than (C_2, N_2), that is, if $C_1 < C_2$ or $C_1 = C_2$ but $N_1 < N_2$. The optimal assignment is defined as the one with the lexicographically smallest (C, N).

For a given link path, the optimal wavelength assignment algorithm considering the conversion priorities of the nodes is shown in Table 2. We here define the set of nodes that can be reached by the source via a lightpath as the *reachable set* and denote it by R. The algorithm will try to extend R at a wavelength convertible node chosen as the extending point in each step until t can be added to R or until R cannot be extended any further. To extend R at node u is to add to R all the nodes not previously in R but can be found by u. Also, a label is given to all the newly found nodes: if the extending point, say, u, has label (c, n), nodes that added to R by extending at u are labeled as $(c, n+1)$ if u is a non-critical node and $(c+1, n)$ otherwise. At the first step, the source is regarded as the extending point and all the nodes that can be reached by the source without conversion is added to R. All such nodes are given label $(0, 0)$. Note that if u is the extending point and v can be added to R by extending at u, all w where $u < w < v$ can also be added to R. It follows

TABLE 2

LABEL EXTENDING ALGORITHM

```
Extend $R$ at the source.
Give all the nodes in $R$ label $(0,0)$.
while $t$ is not labeled
    Among all the newly labeled nodes, find $u$,
    the non-critical node with the largest index.
    if $u$ exists
        Extend $R$ at $u$.
        Give label to nodes accordingly.
    else
        For nodes that have larger indices than the last
        extending point and have not been critical-searched
        before, let $(c,n)$ be the smallest label.
        while no non-critical nodes were labeled
            Find $v$, which a critical node with the largest
            index among nodes with label $(c,n)$.
            if $v$ exists
                Extend $R$ at $v$.
                Give label to nodes accordingly.
            else
                Set $(c,n)$ to be the next smallest label
                that have not been searched.
                exit if no such label.
            end if
        end while
    end if
end while
```

that if node u is in R, all the nodes with smaller indices than u must also be in R.

In the algorithm, if in an extension some non-critical nodes were added to R or were labeled, the next extending point will be the newly labeled non-critical node with the largest index. Otherwise, the algorithm will find the set of nodes with the smallest label among the nodes that have larger indices than the last extending point and have not been scanned for critical nodes before. It will do a search, called the "critical-search" among these nodes to find a critical node. If there is one, this node is set to be the next extending point. Otherwise, the algorithm does the critical-search in the set of nodes with a larger label. If all labeled nodes have been explored and no node can be set to be the extending point, the algorithm returns and t is not reachable.

For example, Fig.2(d) shows the assignment found by the Label Extending Algorithm. Nodes 3, 8 and 10 are critical nodes. In the first step, nodes 1, 2, 3 and 4 are labeled as $(0,0)$, because they can be found by 0 on λ_0. Since a non-critical node, node 2, was labeled, it is set to be the next extending point. Node 2 gives label $(0,1)$ to nodes 5 and 6, and node 5 becomes the next extending point. Node 5 gives label $(0,2)$ to nodes 7, 8 and 9, and node 7 becomes the next extending point. Node 7 gives label $(0,3)$ to nodes 10 and 11. Since there is no non-critical node labeled at this step, the algorithm should perform a critical-search among the nodes with larger indices than the previous extending point. These nodes are 8 and 9, with label $(0,2)$; and 10 and 11, with label $(0,3)$. The algorithm will start at nodes with the smallest label, and node 8 gives label $(1,2)$ to node 12. Since node 12 is not wavelength convertible, the algorithm checks nodes 10 and 11, and 10 gives label $(1,3)$ to node 13. 13 is a non-critical node and can find 15. The algorithm then terminates.

We next show that this algorithm is optimal because the label given to every node is the minimum number of conversion needed to reach it.

Theorem 2: The following two invariants hold throughout the execution of the algorithm: (1) The algorithm labels as many nodes as possible in each extension; (2) The label given to each node is the optimal number of conversion needed to reach this node.

Proof. We prove it by induction on the steps of extension. This is obviously true at the first step when all the nodes that can be reached from the source without conversion are labeled $(0,0)$. Now suppose it is also true for the first H extensions. Now consider the $(H+1)_{th}$ extension. Invariant 1 is easy to see since we always use the one with the largest index as the extending point. In the following we show Invariant 2 also holds.

If there is a non-critical node labeled as (c,n) at the H_{th} extension, according to the algorithm, we choose the one with the largest index, say, u, as the next extending point, and all the nodes that can be found by u are labeled as $(c,n+1)$. Now if this label is not correct for one of the nodes, say, z, then z can be reached from the source by a lightpath with less than $(c,n+1)$ conversions. Let the last wavelength conversion along this path occur at node x, and first suppose x is non-critical. Then x can be reached with less than (c,n) conversions. We first claim that $u > x$, since otherwise, u can be reached by the source with less than (c,n) conversions which contradicts the induction hypothesis. If $x < u$, after step H, x must have been labeled and according to the induction hypothesis it has been given a correct label, say, (c',n') which is less than (c,n). Now consider when we first label x. Since x is a non-critical node, the next step must use a non-critical node with label (c',n') as extending point. This means that z must have been labeled at this step, since there is a wavelength continuous segment from z to x. This contradicts the fact that z has not been labeled until step $H+1$.

Now if along the lightpath which reaches z with less than $(c,n+1)$ conversions, the last conversion is critical. Let this node be y. y must have been labeled, and suppose it was labeled as (c'',n'') where $c'' < c$. If we had used one of the critical nodes with a label no less than (c'',n'') as an extending point, z must have been labeled upon this extension, because of the wavelength continuous segment from y to z and that these nodes are closer to z than y is. But if we had not, then we only used nodes with smaller labels than (c'',n'') as extending points up to the H_{th} extension. Consider v which is the extending point that gave label (c,n) at the H_{th} extension. Suppose the label of v is (c''',n'''). v cannot be non-critical, since otherwise $c''' = c > c''$. Hence, $c''' = c-1$ and $n''' = n$. In this case, we must have $c'' = c''' = c-1$ and $n'' > n''' = n$, since y has a label larger than v. This is a contradiction.

Now consider when the $(H+1)_{th}$ extension is a critical extension. Suppose the extending point has label (c,n). Then all the newly labeled nodes are given label $(c+1,n)$. Suppose that the invariant is not true, that is, there exists a lightpath reaching a newly labeled node z with less than $(c+1,n)$

conversions. We choose z to be such a node with the smallest index. Again consider the last wavelength conversion node on this path and let it be w. We first show that w must have been labeled after step H. If it is not, then it must be labeled at step $H+1$, since its index must be smaller than z and z was labeled at step $H+1$. In this case, due to the choice of z, w must be labeled correctly. Therefore to reach w at least $(c+1, n)$ conversions are needed. This contradicts the hypothesis that there exists a lightpath reaching z converting wavelength at w with less than $(c+1, n)$ conversions.

We next show that after step H w cannot have a label larger than (c, n). This is because if so, w must be a critical node since it has been labeled before but was not chosen to be an extending point. Suppose w has label (c', n'). Then $(c'+1, n')$ must be smaller than $(c+1, n)$ while (c', n') is larger than (c, n). It is not hard to see that these two conditions cannot be both satisfied. It the case where w has a smaller label than (c, n), following the same arguments as the case when step $H+1$ extends at a non-critical node, we can obtain a contradiction. ■

Next we analyze the complexity of this algorithm. We can still represent the state of links with k-bit binary vectors. The algorithm does two things: extend the reachable segment and search for wavelength convertible nodes. The latter is done in $O(t)$ time, because we scan one node at most twice, once for searching the non-critical node and the other for searching the critical node. The extending is to do *AND* operations on the k-bit binary vectors. We show that one particular vector can be involved in this operation at most 3 times. Consider the time when the algorithm finished an extension and labeled some nodes. The vectors for the links connecting these nodes would then have been *ANDed* exactly once. If among these nodes there is a non-critical node, these vectors would be *ANDed* at most one more time. If there is no non-critical node, the number of *AND* operations for a vector will be determined by the number of extensions that originated from the nodes with smaller labels. It turns out that there can be at most one such an extension. This is because, at the first time when the algorithm failed to find a non-critical node after an extension which, say, labeled nodes as $(0, n)$, there can be at most two types of nodes with indices larger than the last extending point: nodes with labels $(0, n-1)$ and $(0, n)$, and it will search only among them for a critical node as the next extending point. In each extension some new nodes will be labeled, but there will be at most one set of nodes which may contain the next extending point with a smaller label than those of newly labeled nodes. The same is true for the following extensions. Hence, it takes $O(t)$ time for the algorithm to find where to convert wavelengths, where t is the number of links between the source and the destination. The time to set up a lightpath is $O(tk)$, where k is the number of wavelengths per fiber.

One might be concerned on that the Label Extending Algorithm may end up using too many converters at non-critical nodes. The next theorem gives the bound of the number of converters used by the Label Extending Algorithm.

Theorem 3: The total number of converters used by the Label Extending Algorithm is at most twice of the Longest Segment Algorithm.

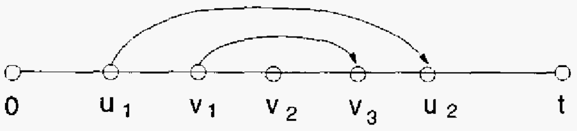


Fig. 4. If u_1 can reach u_2, then v_2 cannot have been not been used.

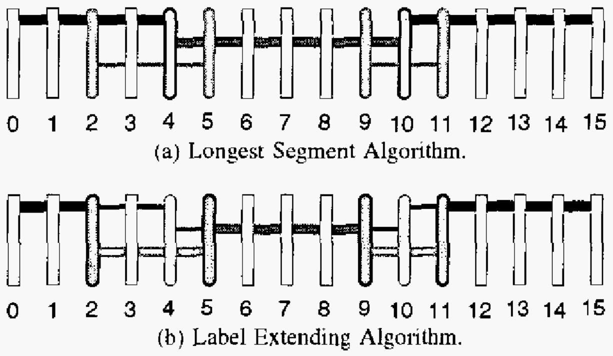


Fig. 5. The bound in Theorem 3 is tight. 4 and 10 are critical nodes.

Proof. Consider the assignment given by the Longest Segment Algorithm. Suppose it uses W converters at nodes, say, u_1 to u_W, and these nodes divide the path into $W+1$ segments. We show that in the segment between any u_i and u_{i+1}, the Label Extending Algorithm can use no more than 2 converters. We show this by contradiction. Suppose $W=2$, as shown in Fig.4. Since there is a wavelength continuous segment between u_1 and u_2, v_1 can reach v_3 without converting wavelength at v_2 and thus v_2 could not have been used. By similar arguments, we can show that there can be at most one converter used by the Label Extending Algorithm in the segment between the source and u_1 and the segment between u_W and the destination. Thus, at most $2(W-1)+2 = 2W$ converters are used. ■

The example in Fig.5 shows that the bound is tight.

C. Network-Wide Dynamic Routing and Wavelength Assignment Algorithm

A more complicated case is when the routing is dynamic, in other words, the source can choose any path network-wide to connect to the destination. This will potentially improve the network performance, in the mean time, it also poses more challenges to the scheduling. We now need to search the entire network to find the optimal lightpath.

We measure a lightpath by a pair of integers, (c, h), where c is the number of wavelength conversions and h is the number of hops. A lightpath is considered better than another if its measure is lexicographically smaller than the other, that is, if it uses less wavelength conversions, or, if it travels less hops when the number of wavelength conversions are the same, for reasons described in Section I. The optimal lightpath is defined as a path with the lexicographically smallest measure.

We propose an algorithm, called the Label Searching Algorithm, to solve this problem, and the algorithm is shown in Table 3. The idea of it is simple and is similar to the Label Extending Algorithm. Let s be the source and t be the destination. In the first round, s gives label $(0, h)$ to all the nodes that can be reached from it with no wavelength conversion and a minimum of h hops. If the destination has been labeled, the optimal lightpath has been found. Otherwise wavelength conversions need to be used.

Let the u be the wavelength convertible node with the smallest label, say, $(0, h)$. All unlabeled nodes that can be reached by u with one hop can be labeled as $(1, h + 1)$. Because first, apparently, they can be reached by the source with one conversion and $h + 1$ hops. They cannot be reached without wavelength conversion because otherwise they would have been labeled previously. Now if they can be reached with one conversion but less than $h + 1$ hops, suppose along one of such paths wavelength conversion occurs at node v. v must have a label less than $(0, h)$, which contradicts the fact that u is the labeled wavelength convertible node with the smallest label.

After labeling all nodes that can be reached with one hop from wavelength convertible nodes with label $(0, h)$, the algorithm moves on to label nodes that can be reached with two hops from wavelength convertible nodes with label $(0, h)$ and nodes that can be reached with one hop from wavelength convertible nodes with label $(0, h+1)$. All such nodes can be labeled as $(1, h + 2)$, for similar reasons as described earlier. In the following, in step i, the algorithm gives label $(1, h+i)$ to nodes that can be reached with $i - j$ hops from wavelength convertible nodes with label $(0, h+j)$ for all possible j, until all the nodes that can be reached from wavelength convertible nodes labeled in the first round without converting wavelength have been labeled. By this time we have found the optimal lightpath to all the nodes that can be reached with no more than one wavelength conversion.

If the destination is still not reached, we begin a new round of labeling, starting from the wavelength convertible node that was labeled in the previous round with the smallest label. This process is repeated until the destination is labeled or no new nodes can be labeled.

It can be seen that in the I_{th} round of labeling, all the nodes that can be reached with a minimum of I wavelength conversions are labeled. It follows that if there is a lightpath from the source to the destination, such a path will be found. And since the label we give always reflects the measure of the optimal lightpath from the source to this node, when the destination is labeled, we have found the desired optimal lightpath.

In round I, to find nodes certain hops away from wavelength convertible nodes and label them, we can maintain a search list. In the first step, the wavelength convertible nodes labeled in the previous round with the smallest label, say, (I, h) are added to the list. After this, for each node u in the list, we add all unlabeled nodes adjacent to u to the list and label them as $(I+1, h+1)$ and remove u from the list. Then all wavelength convertible nodes with label $(I, h + 1)$ are added to the list, and repeat the process.

TABLE 3
LABEL SEARCHING ALGORITHM

```
Label all nodes that can be reached from $s$
without conversion.
$I \leftarrow 0$;
while $t$ is not labeled
    Suppose $(I, h)$ is the minimum label of wavelength
    convertible nodes labeled in the previous round.
    $i \leftarrow 1$
    do
        Give label $(I + 1, h + i)$ to all unlabeled nodes
        that are a minimum of $i - j$ hops away from some
        wavelength convertible nodes with label $(I, h + j)$
        for all possible $j$.
        $i \leftarrow i + 1$
    while new nodes have been labeled
end while
```

To establish the lightpath, each node records which node first reaches itself on a certain wavelength. For example, if u is in the search list and on the link connecting u and v, λ_1 is free, we record at v that it was reached from u by λ_1. After this, we mask λ_1 on this link, since we have used this channel for searching. Each node also records the wavelength by which it is first reached. After t is labeled, it will first find the node that labeled it, and then use the information above to trace back to establish the lightpath to s. Clearly, the complexity of this algorithm is $O(Nk + Lk)$, where N is the number of nodes, L is the number of links in the network, and k is the number of wavelength per fiber, since a link and a node is checked no more than k times.

D. Performance Study for Unicast Algorithms

We have implemented the algorithms and studied their performances under different network topologies. We assume that the connection requests arrive at the network according to a Poisson process, and the traffic intensities between every pair of nodes are the same. The duration of a connection follows exponential distribution with parameter 1. The network performance is measured by overall *blocking probability*, as a function of the arrival rate at each node. In Fig.6 and Fig.8, "FF" denotes the First Fit Algorithm, "LSeg" denotes the Longest Segment Algorithm, "LExt" denotes the Label Extending Algorithm and "LSear" denotes the Label Searching Algorithm.

We first show the results for a bidirectional ring network with 16 nodes and 16 wavelengths per fiber. Ring network was chosen first because it is a widely used topology. The second reason is that, for any pair of nodes in the ring, there are exactly two possible routes connecting them. Thus there is no routing problem needs to be solved and to set up a lightpath we can simply apply the wavelength assignment algorithm on these two routes. This makes it perfect for comparing the impact of wavelength assignment algorithms on network performance. We assume that there are 8 wavelength converters in each node, and the threshold of the Label Extending Algorithm is 2. In Fig.6, we can see that both Longest Segment Algorithm

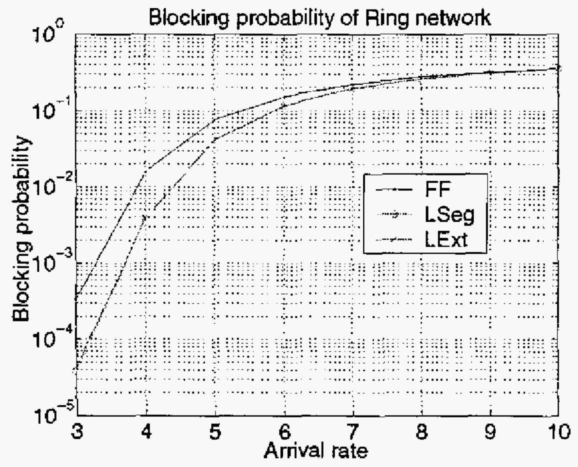


Fig. 6. Blocking probability of a bidirectional ring network with 16 nodes and 16 wavelengths per fiber.

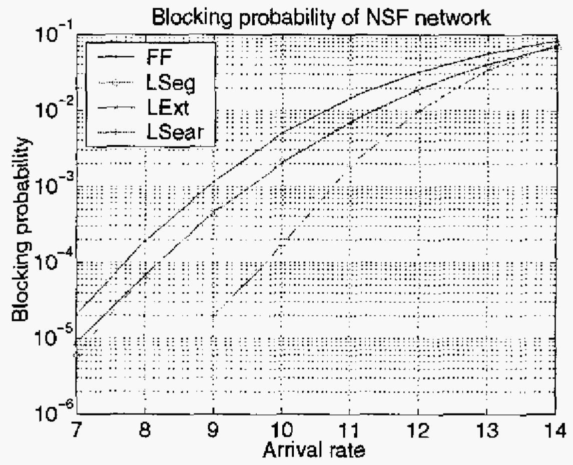


Fig. 8. Blocking probability of the NSF network with 16 wavelengths per fiber.

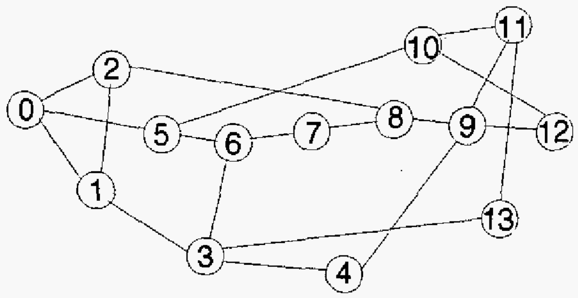


Fig. 7. The NSF network.

and the Label Extending Algorithm outperform the First Fit Algorithm by a large amount when the arrival rates are not too high. An interesting phenomenon is that the performance of the first two algorithms is almost the same under this scenario.

We also used the well known 14-node NSF network, shown in Fig. 7, as a testing topology, and the results are shown in Fig.8. We still assume that there are 16 wavelengths per fiber and there are 8 wavelength converters in each node, and the threshold of the Label Extending Algorithm is 2. For each pair of nodes, we keep up to 4 link disjoint paths as candidate link paths. Similar trends can be observed as in Fig.6. However, as expected, the performance of the Label Searching Algorithm is better than the others.

III. Wavelength Scheduling for Multicast

We now move onto the on-line wavelength assignment problem in WDM networks under multicast traffic. In this section we consider setting up a light tree in a given link tree for a multicast connection using the minimum number of converters. We first give definitions and notations that will be used in the section.

A. Definitions and Notations

For a tree denoted by T, we define a node with *out degree* more than one as the *ramification node*. Each part of the tree starting at a ramification node is called a *branch*. The ramification node is called the *root node* of this branch. A tree may have many branches, and a branch itself may also have child branches. A branch containing no child branches is called a *simple branch*. The tree itself can also be regarded as a branch, with its root node being the source node of the multicast connection. A branch is of level i if along the path from the source node to the root node of it there are i ramification nodes, including the root node of the branch. The tree itself is of level 0. The *tree level* is the maximum branch level. A node is said to belong to a branch if there is no ramification node along the path from the root node of the branch to it.

We now introduce the following two notions. A light tree, when extending to a branch B, may choose from the k wavelengths to enter this branch. The cost associated with each wavelength called the *covering cost* and measured by the minimum number of wavelength converters used if the light tree enters this branch on this wavelength, is denoted by $\Delta_B(i)$ for λ_i where $i \in [1, k]$. If $\Delta_B(i) \neq \infty$, we say λ_i *covers* B at cost $\Delta_B(i)$. If λ_i covers a branch, we say the branch can be *entered* by λ_i. In a light tree, if the first link of a branch is on λ_i, we say the tree *enters* this branch on λ_i. For branch B, we use δ_B to denote $\min\{\Delta_B(i)\}$. If $\Delta_B(i) = \delta_B$, we say λ_i is the *optimal entering wavelength* of this branch. For simplicity, we use $\Delta_B()$ to denote the set of $\Delta_B(i)$ for all $i \in [1, k]$.

For a ramification node r, we use $\Upsilon_r(i)$ to denote the minimum number of wavelength conversions needed to cover all its child branches if the light tree reaches this node on wavelength λ_i where $i \in [1, k]$. We use $\Upsilon_r()$ to denote the set of $\Upsilon_r(i)$ for all $i \in [1, k]$. Table 4 listed these notations. Also, throughout this section, we use B to denote the branches, and use r to denote the ramification node of a branch, and use t to denote the root of a branch.

Fig.9 is an example for illustrating the definitions and the notations. The link tree is shown in the left of the figure. It has 12 nodes. Node 0 is the root of the tree. Node 3 is the ramification node. Node 3 has 3 child branches, denoted as B_1, B_2 and B_3. Node 3 is the root node of all these branches.

TABLE 4

List of symbols in Section III

$\Delta_B(i)$	:	minimum number of converters used when the
	:	tree enters B on λ_i.
δ_B	:	$\min \Delta_B(i)$ for all $i \in \{0, 1, \ldots, k-1\}$.
$\Upsilon_r(i)$	:	minimum number of converters needed to
	:	cover all child branches of ramification node r
	:	when the light tree reaches r on λ_i.

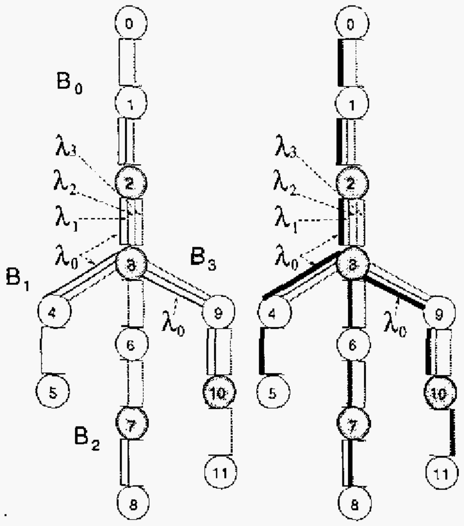


Fig. 9. A link tree with 4 wavelengths per fiber and the optimal light tree, where 2, 3, 4 and 10 are wavelength convertible nodes.

B_1, B_2 and B_3 are all simple branches of level 1. B_0 is a branch of level 0 and is the tree itself. The tree level is 1.

It is not difficult to see that $\Delta_{B_1}() = \{0, \infty, \infty, \infty\}$, $\Delta_{B_2}() = \{\infty, 0, \infty, 1\}$, $\Delta_{B_3}() = \{1, 1, \infty, \infty\}$. $\delta_{B_2} = 0$, and the optimal entering wavelength for B_2 is λ_1. Under the Converter Split Model, $\Upsilon_3(0) = 2$, $\Upsilon_3(1) = 2$, $\Upsilon_3(2) = 3$, $\Upsilon_3(3) = 3$. As an example, $\Upsilon_3(0) = 2$ since λ_0 is the optimal entering wavelength for B_1 and B_3, and covers these two branches at total cost of 1, and it can be converted to λ_1 at node 3 which covers B_2 at cost 0.

In this simple example, it can be seen that for B_0, $\delta_{B_0} = \Delta_{B_0}(0) = 2$, and we should use λ_0 as the entering wavelength of it. The optimal light tree is shown in the right of the figure where wavelength channels in the optimal light tree are shown in wider line segments.

B. Outline of the Algorithm

The idea of our algorithm is simple: Find $\Delta_T()$ recursively where T is the tree. This can be implemented in a "bottom-up" way, as shown in Table 5. Basically, we start from branches at the lowest level which are all simple branches, and find $\Delta_B()$ for them. Then "move up" one level, find $\Delta_B()$ for branches at this level according to the $\Delta_B()$ of their child branches found in the previous round. Then keep on moving up until reach the highest level. The minimum number of conversions needed to cover the tree is δ_T. Once δ_T is obtained, we can determine which wavelength to enter a branch and the optimal light tree can be established.

TABLE 5

Algorithm for finding optimal cover cost of a light tree T

```
$i$ ← tree level;
while $i \geq 0$
    Find $\Upsilon()$ for all non-simple branches of level $i$ ;
    Find $\Delta()$ for all branches of level $i$;
    $i \leftarrow i - 1$;
end while
return $\delta_T$;
```

The major difference between our algorithm and the algorithms in [1] is that the basic element of our algorithm is a tree branch while the basic element of the algorithms in [1] is an individual node. We only compute $\Delta()$ for a branch, while [1] computes it for every node. As a result, our algorithm is much simpler and needs less computation.

C. Finding $\Delta_B()$

We first explain how to find $\Delta_B()$. Our way of finding $\Delta_B(i)$ for λ_i is to apply the Longest Segment Algorithm for unicast in Table 1, and takes $O(t)$ time for a branch with t nodes. Overall, the time spent on this task is $O(Nk)$ where N is the total number of nodes in the tree.

Finding $\Delta_B(i)$ for a simple branch B is easy. We can mask all wavelengths other than λ_i on the fist link of B and apply the Longest Segment Algorithm, by regarding the root of the branch as the source node and the leaf node of the branch as the destination node. The number of converters needed found by the algorithm is $\Delta_B(i)$.

When B is not a simple branch, $\Delta_B(i)$ can be found as follows. Let the root node of the branch be t and let the ramification node of the branch be r. At this time, $\Upsilon_r()$ should have been found. There are two cases: (1) If there is no wavelength converters left in the nodes of the branch, $\Delta_B(i) = \Upsilon_r(i)$ if there is a wavelength continuous segment on λ_i from t to r, otherwise $\Delta_B(i) = \infty$; (2) If there are wavelength convertible nodes from t to r, suppose the one nearest to r is u. Consider the set of wavelengths that have wavelength continuous segments from u to r. Let $\epsilon = \min \Upsilon_r(j)$ for all λ_j in this set. Again we mask all other wavelengths on the first link except λ_i and apply the Longest Segment Algorithm, and suppose η_i converters are needed and the last extending point is v. For all wavelengths that have wavelength continuous segments from v to r, if there is λ_j where $\Upsilon_r(j) = \epsilon$, then apparently, $\Delta_B(i) = \eta_i + \epsilon$. Otherwise, $\Delta_B(i) = \eta_i + \epsilon + 1$, since we can convert the wavelength at u to one of the wavelengths that achieve ϵ.

We have the following theorem concerning $\Delta_B()$.

Theorem 4: If there are wavelength converters on a branch, $\Delta_B()$, excluding those who are ∞, differ at most by one.

Proof. Suppose $\Delta_B(i) = \delta_B$. If there are wavelength convertible nodes in the branch, suppose the one closest to the root t is w. For any wavelength other than λ_i, say, λ_j, if there is no wavelength continuous segment on λ_j from t to w, $\Delta_B(i) = \infty$. Otherwise, suppose the lightpath for λ_i leaves

w on λ_l. We can convert λ_j to λ_l at w, and hence cover the branch at cost no more than $\Delta_B(i)+1$. ■

Note that the property in Theorem 4 was unaware of by [1] and part of the algorithm in [1] is unnecessary. Also note that our method for finding $\Delta_B()$ can be applied to both the No Converter Split Model and the Converter Split Model. Thus, we have obtained a simpler solution for the same problem defined in [1] as well.

D. Finding $\Upsilon_r()$

In this subsection we will try to solve the problem of finding $\Upsilon_r()$ under converter split model. We show that the problem is hard and even hard to approximate, because they are inherently related to the *Set Covering Problem* which is known to be NP-hard and no polynomial time algorithm has a log ratio. Nevertheless, simple greedy algorithms will produce good results and will save conversion cost in most of the cases.

We show that the problem can be solved by consecutively running three greedy algorithms, shown in Tables 7, 8 and 9, respectively, and for convenience, we call them Greedy 1, Greedy 2 and Greedy 3. For a ramification node with m child branches, to find $\Upsilon_r(i)$ for a wavelength λ_i, all greedy algorithms run in $O(m^2k)$. The overall time to find a light tree is thus $O(N^2k^2)$, where N is the number of nodes in the tree, and k is the number of wavelengths.

1) Determining Feasibility: Consider a ramification node r, and suppose it has m child branches, denoted as $B_1, B_2, \ldots, B_m$, and $\Delta_{Bj}()$ is known for all $j \in [1, m]$. Suppose there are C converters left at node r. We first show that

Theorem 5: It is NP-complete to decide whether all child branches of r can be covered using no more than C converters at r.

Proof. The problem can be formalized as follows. First note that for a branch B_j with $\Delta_{Bj}(i) \neq \infty$, it can be covered by λ_i without wavelength conversion. Hence we only need to consider the rest of the branches. Suppose there are m' of them. We regard each of them as an element of a set with m' elements, denoted as S. We also regard each wavelength as a subset of S, denoted as S_l for $1 \leq l \leq k$. An element B_j is in subset S_l if $\Delta_{Bj}(l) \neq \infty$, i.e., if λ_l can cover branch B_j. Each subset can cover more than one elements and each element can be covered by more than one subsets. The question then becomes: Given these m' elements and k subsets, can a group of no more than C subsets be found such that each element is in at least one of the subsets? This is exactly the *Set Covering Problem* which is NP-complete [16]. ■

A simple greedy algorithm shown in Table 6 can be used to solve the set covering problem approximately. In each step, this algorithm finds a subset that covers the maximum number of uncovered elements. It has an $O(\ln m')$ performance ratio, where m' is the number of elements, which means that if the elements can be covered with a minimum of C_{opt} subsets, the number found by the algorithm, C', satisfies $C'/C_{opt} \leq \ln m'$. Recent results show that no polynomial algorithm has a ratio smaller than $\ln m'$ [16].

TABLE 6
GREEDY SET COVER ALGORITHM

```
$C' \leftarrow 0$;
while $\Pi \neq \emptyset$
        find $S_l$ which covers the most elements in $\Pi$;
        if no such subset can be found break.
        Remove all elements in $\Pi$ that can be covered by $S_l$;
        $C' \leftarrow C' + 1$;
end while
```

TABLE 7
GREEDY 1

```
Let $\Pi$ be the set of branches that cannot be covered by $\lambda_i$;
For every $\lambda_l$, let $S_l$ be a subset where an element in $\Pi$
is in $S_l$ if that branch can be covered by $\lambda_l$.
Apply the Greedy Set Cover Algorithm. find $C'$.
If $C' \leq C$ return yes else return no.
```

We can transform our problem into a set cover problem directly and use the algorithm in Table 6 to determine the feasibility, as shown in Table 7. If $C' \leq C$ then child branches can be covered. Note that because the problem is NP-complete, this approximation algorithm may turn down some request even if it is feasible. However, since this algorithm achieves the best possible performance ratio to the optimal algorithm, it is the best we can do for this problem.

2) Minimizing Conversion Cost: Suppose the Greedy 1 determines that the request is feasible. The next question is then to find a wavelength assignment that uses the minimum number of converters. We show that

Theorem 6: It is NP-hard to find a wavelength assignment to cover all child branches of r at minimum total cost.

Proof. This problem can be formalized as follows. Given a set S with m elements, each representing a branch, and k subsets, each representing a wavelength. Let c_{lj} be the cost of using subset S_l to cover element B_j, in our case $c_{lj} = \Delta_{Bj}(l)$. Let c'_l be the cost of using subset S_l, in our case, $c'_l = 0$ if $l = i$ and $c'_l = 1$ if $l \neq i$. The problem then becomes finding a group of subsets to cover all elements with minimum cost.

This problem is NP-hard, because given any instance of the set covering problem, we can transform it into an instance of this problem with the same elements and subsets plus an additional subset S_i. For a subset S_l in the original problem, let $c_{lj} = 0$ for all element B_j and $c'_l = 1$. For S_i, let $c_{ij} = \infty$ for all element B_j and $c'_i = 0$. Therefore it is not hard to see that if we can solve this problem optimally, we can also solve the set covering problem optimally. ■

We hereby give a greedy algorithm to solve this problem, shown in Table 8. First note that if $\Delta_{Bj}(i) = \delta_{Bj}$, we can cover this branch with minimum cost and without wavelength conversion at r. Therefore, the optimal assignment must enter branch B_j with λ_i. Thus we need only to consider branches where $\Delta_{Bj}(i) \neq \delta_{Bj}$. The input to this algorithm is $\Delta()$ for all child branches. The output is the number of wavelength converters needed to cover all the child branches if the light tree reaches r on λ_i.

TABLE 8

Greedy 2

```
Let all branches where $\Delta_{Bj}(i) \neq \delta_{Bj}$ be an element.
For every $\lambda_l$, let $S_l$ be a subset where element $B_j$ is
in subset $S_l$ if $\Delta_{Bj}(l) = \delta_{Bj}$.
Apply the Greedy Set Cover Algorithm, find $C'$.
return $C' + \sum_{j=1}^{m} \delta_{Bj}$
```

We can see that by the algorithm, we only enter a branch via an optimal entering wavelength. We next show that

Theorem 7: The performance ratio of the algorithm shown in Table 8 is $\ln m$, where m is the number of branches.

Proof. Given any optimal assignment, if for a branch the entering wavelength λ_p is not an optimal entering wavelength, we can convert λ_i to an optimal entering wavelength λ_l at r, and enter this branch on λ_l. This new assignment should still be optimal, since $\Delta_{Bj}(p) \geq \Delta_{Bj}(l) + 1$. Hence there exists an optimal assignment where every branch is entered via its optimal entering wavelength. Note that the assignment we give also enters every branch via its optimal entering wavelength. Therefore, the difference between the optimal and the greedy algorithms is the number of wavelength converters used at r. Denote them as C_{opt} and C', respectively. With previous discussions, we know that $C'/C_{opt} \leq \ln m$. Therefore the performance ratio is

$$\frac{C' + \sum_{j=1}^{m} \delta_{Bj}}{C_{opt} + \sum_{j=1}^{m} \delta_{Bj}} \leq \frac{C'}{C_{opt}} \leq \ln m$$

since $C' \geq C_{opt}$ and $\sum_{j=1}^{m} \delta_{Bj} \geq 0$. Also note that this bound is tight, when $\sum_{j=1}^{m} \delta_{Bj} = 0$. ■

3) Minimizing Conversion Cost for the Case of Limited Number of Available Converters: Greedy 2 tries to minimize the total number of wavelength conversions. Note that this is a simplified approach to solving our problem, as it does not consider the constraint that there are a total of C available wavelength converters. If $C' \leq C$, or the number of converts used at r is no more than the number of available converters, Greedy 2 gives a feasible scheduling. Otherwise, we might have to use other methods to find a feasible scheduling while trying to reduce the number of converters used.

This problem can be formalized as follows. Given a set S with m elements and k subsets, let c_{lj} be the cost of using subset S_l to cover element B_j and let c'_l be the cost of using subset S_l. Find a group of no more than C subsets to cover all elements with minimum cost. Here in our particular problem, c_{lj} is $\Delta_{Bj}(l)$ which takes non-negative integer values, and c'_l is 1 if $l \neq i$ and 0 otherwise.

By using a similar method to the proof of Theorem 6, we can transform an instance of the set covering problem to an instance of this problem, and show that

Theorem 8: It is NP-hard to find a wavelength assignment to cover all child branches of r at minimum total cost while using no more than C converters at r.

Table 9 gives an algorithm to solve it approximately. The idea is to start with a feasible scheduling which is the one found by Greedy 1, and use wavelength conversions at r to reduce the number of conversions step by step. Note that if we choose to use a converter to convert λ_i to an unused wavelength λ_p, for branch B_j, if under current assignment the entering wavelength is λ_l, the cost of B_j will be reduced by an amount of $[\Delta_{Bj}(l) - \Delta_{Bj}(p)]^+$.

TABLE 9

Greedy 3

```
Use Greedy 1 to find a feasible scheduling.
Suppose $C'$ wavelength conversions are used at $r$.
Let $w(B_j)$ be the cost of branch $B_j$.
$w(B_j) \leftarrow \Delta_{Bj}(l) - \delta_{Bj}$ where $\lambda_l$ is the entering
wavelength of $B_j$ under current assignment.
while $C' \leq C$
    Find an unused wavelength that reduces the
    maximum total cost of the branches.
    if no such wavelength can be found break.
    Update the cost of the branches.
    $C' \leftarrow C' + 1$.
end while
```

Unlike the previous two greedy algorithms, we cannot find a bound for this algorithm. The reason is that due to its greedy nature, the algorithm will try to make sure that all branches are covered first, and may choose to enter a branch not using the optimal entering wavelength while the difference between the covering cost of the optimal entering wavelength and that of other wavelengths can be unbounded. However, note that if there are wavelength converters on every branch, the covering costs differ at most by one, as shown in Theorem 4. In this case, it is not hard to show that the difference between the cost of the assignment found by Greedy 3 and that of the optimal algorithm is no more than $C(1 - 1/\ln m) + m$.

In Greedy 3, when Greedy 1 has finished, we will choose a wavelength to reduce the total cost in each of the following steps. We next show that,

Theorem 9: After finishing Greedy 1, in the following steps the difference between the cost of the assignment found by Greedy 3 and that of the optimal algorithm will decrease exponentially at a rate no less than $(1 - 1/C)$.

Proof. Suppose when Greedy 1 has finished, the cost difference between the greedy and the optimal in the m branches is ω_0. Therefore, if we add in all C'_{opt} wavelengths in the optimal scheduling, the cost will decrease by exactly ω_0. Thus, at least one of the wavelengths in the optimal scheduling will reduce the total cost by ω_0/C'_{opt}. Since by the greedy algorithm we always choose the wavelength that decreases the maximum amount of cost, we have $\omega_1 \leq \omega_0(1 - 1/C'_{opt})$. In other words, ω_1 is at most a fraction of $(1 - 1/C'_{opt})$ of ω_0. The same is also true for all following steps. Thus, the decreasing rate is no less than $(1 - 1/C)$, since $C \geq C'_{opt}$. ■

E. Performance Study for Multicast Algorithms

To compare the performance of the multicast wavelength assignment algorithms, we applied the algorithms on randomly generated trees with average 39.4 nodes, and the results as a function of the percentage of the availability of the wavelength channels on each link are shown in Fig.10. We can see that by allowing the output of the converter to split, our algorithm saves about 6% to 10% of the converters.

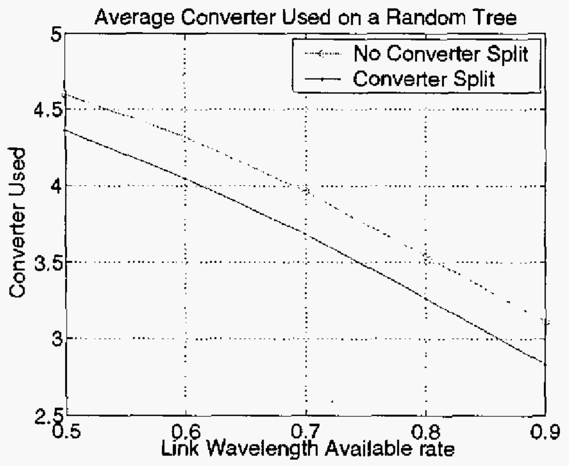


Fig. 10. Average number of converters needed to set up a light tree in a link tree with average 39.4 nodes.

IV. Conclusions

In this paper we studied the problem of on-line setting up connections in WDM networks at minimum conversion cost. We considered both unicast and multicast traffic, and improved existing results significantly. For unicast, we first considered the problem of setting up a lightpath on a given link path with minimum number of conversions, and gave a new algorithm that solves it in $O(tk)$ time, where t is the number of links and k is the number of wavelengths, as compared to the best known existing algorithm that runs in at least $O(t^2k)$ time. We also considered the case when nodes have different conversion priorities and gave an $O(tk)$ time algorithm for setting up a light path while converting wavelength at higher priority nodes only when necessary. We then generalized this technique to WDM networks with arbitrary topologies and presented an algorithm that sets up an optimal lightpath network-wide in $O(Nk + Lk)$ time by checking the state of the entire network, where N is the number of nodes in the network and L is the number of links in the network. For multicast, we focused on the problem of setting up an optimal light tree on a given link tree with minimum conversion cost. We proposed a new multicast conversion model which allows the output of the converter to split and can save the conversion cost considerably. We showed that this problem is NP-hard, and then gave efficient heuristics to solve it approximately. Our method can also be applied to solve the problem when output of the converter is not allowed to split in linear time and is much simpler to implement than existing algorithms.

V. Acknowledgements

The authors would like to thank Professor Esther Arkin of the State University of New York at Stony Brook for the valuable suggestions on the set covering problem. This work was supported in part by the U.S. National Science Foundation under grant numbers CCR-0073085, CCR-0207999 and ECS-0427345.

References

[1] B. Chen and J. Wang, "Efficient routing and wavelength assignment for multicast in WDM networks," *IEEE JSAC*, vol.20, no.1, pp.97-109, 2002.

[2] R. Libeskind-Hadas and R. Melhem, "Multicast routing and wavelength assignment in multihop optical networks," *IEEE/ACM Trans. Networking*, vol.10, no.5, pp.621-629, 2002.

[3] D.-N Yang and W. Liao "Design of light-tree based logical topologies for multicast streams in wavelength routed optical networks," *IEEE INFOCOM 2003*, vol. 1, pp. 32-41, 2003.

[4] Y. Zhang, et. al, "An efficient heuristic for routing and wavelength assignment in optical WDM networks," *IEEE ICC 2002*, vol.5, pp.2734-2739, 2002.

[5] L. Ruan, et. al, "Converter placement supporting broadcast in WDM optical networks," *IEEE Trans. Computers*, vol.50, no.7, pp.750-758, 2001.

[6] B. Mukherjee, "WDM optical communication networks: progress and challenges," *IEEE JSAC*, vol.18, no.10, pp.1810-1824, 2000.

[7] W. Liang and X. Shen, "Improved lightpath (wavelength) routing in large WDM networks," *IEEE Trans. Communications*, vol.48, no.9, pp.1571-1579, 2000.

[8] K.-C. Lee and V.O.K. Li, "A wavelength-convertible optical network," *Journal of Lightwave Technology*, vol.11, no.5, pp.962-970, 1993.

[9] I. Chlamtac, A. Farag, and T. Zhang, Lightpath (wavelength) routing in large WDM networks, *IEEE JSAC*, vol.14, pp.909-913, 1996.

[10] W.J. Goralski, *Optical Networking and WDM*, 1st Ed., McGraw-Hill, 2001.

[11] R. Ramaswami and K. N. Sivarajan, *Optical Networks: A Practical Perspective*, 1st Ed., Academic Press, 2001.

[12] A. Mokhtar and M. Azizoglu, "Adaptive wavelength routing in all-optical networks," *IEEE/ACM Trans. Networking*, vol.6, no.2, pp.197-206, 1998.

[13] C. Law and K.Y. Siu; "Online routing and wavelength assignment in single-hub WDM rings," *IEEE JSAC*, vol.18, no.10, pp.2111-2122, 2000.

[14] A.E. Ozdaglar and D.P. Bertsekas, "Routing and wavelength assignment in optical networks," *IEEE/ACM Trans. Networking*, vol.11, no.2, pp.259-272, 2003.

[15] E. Karasan and E. Ayanoglu, "Effects of wavelength routing and selection algorithms on wavelength conversion gain in WDM optical networks," *IEEE/ACM Trans. Networking*, vol.6, no.2, pp.186-196, 1998.

[16] R. Raz and S. Safra, "A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP, *STOC '97*, pp.475-484, 1997.

[17] H. Harai, M. Murata and H. Miyahara, "Performance of alternate routing methods in all-optical switching networks," *Proc. INFOCOM'97*, pp. 517-525, April 1997.