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

Zhenghao Zhang and Yuanyuan Yang

***Abstract*—In this paper, we study on-line wavelength assignment in wavelength-routed WDM networks under both unicast and multicast traffic where nodes in the networks have wavelength conversion ability. Since wavelength converters are still expensive and difficult to implement, we consider the case where nodes in networks have only a limited number of converters that are shared by all input channels. We study the problem of setting up connections in such networks using minimum number of wavelength converters. For unicast traffic, we first study the problem of setting up a lightpath on a given link-path with minimum number of conversions. We give a simple 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 method that needs to construct an auxiliary graph and apply the Dijkstra's algorithm. We also consider the problem of setting up a lightpath while using wavelength converters at nodes with fewer available converters only when necessary, and give an $O(tk)$ time algorithm. We then generalize this technique to WDM networks with arbitrary topologies and give an algorithm that sets up an optimal lightpath network-wide in $O(Nk + Lk)$ time, where N and L are the number of nodes and links in the network, respectively. We also consider multicast traffic in this paper. 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 an optimal 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 multicast conversion model with which the output of the wavelength converter is split-table to save the usage of converters. We show that under this model the problem of setting up an optimal light-tree is NP-hard and then give efficient heuristics to solve it approximately.**

***Index Terms*—Multicast, on-line algorithms, optical networks, routing, shared wavelength converter pool, unicast, wavelength assignment, wavelength conversion, wavelength division multiplexing (WDM).**

I. Introduction and Background

OPTICAL networks with wavelength division multiplexing (WDM) are now widely regarded as the future backbone network for communications because of the huge bandwidth of optical systems. In a WDM network, nodes are connected by optical fiber links and on each link there are multiple wavelengths serving as independent data channels. To send information from the source to the destination, along the path between these two nodes, a wavelength channel must be selected on each link to make a *lightpath* [19]. Without *wavelength converters*, the lightpath must be on the same wavelength, i.e., be *wavelength continuous*, which limits the number of connections that can be simultaneously set up in the network. However, with wavelength converters, the lightpath does not have 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, wavelength converters are still expensive and difficult to implement today. Therefore, to save wavelength converters, it is more cost-effective to use a converter pool consisting of a relatively small number of converters and let them be shared by all inputs than to give each input channel its own wavelength converter, because at any moment it is highly unlikely that all input wavelengths will need wavelength conversion [4]. A consequence of this is that when setting up connections, it is desirable to use as few converters as possible. In this paper, we will study several related problems under this scenario and give efficient algorithms for setting up light connections using fewer wavelength converters whenever possible for both unicast and multicast traffic.

We focus on the *on-line* (or *dynamic*) version of the problem, which means that the scheduler does not know the traffic intensities between the nodes *a priori*, and when a connection request arrives, it tries to satisfy the request optimally based on the current state of the network. 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 the problem where the traffic intensities between all pairs of nodes in the network are known 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, [4], [7], [9], [12], [17]. It can be solved by breaking 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 lightpath on the link-path [4], [17]. Alternatively, the problem can be solved by jointly considering the two subproblems [7], [9], [12]. The second approach will give better results but may be hard to realize especially in a large network. We will mainly follow the

Manuscript received April 18, 2005; revised November 17, 2005; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor A. Fumagalli. This work was supported in part by the U.S. National Science Foundation under Grants CCR-0073085 and CCR-0207999.

Z. Zhang was with the Department of Electrical and Computer Engineering, State University of New York at Stony Brook. He is now with the Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: zhhzhang@cs.cmu.edu).

Y. Yang is with the Department of Electrical and Computer Engineering, State University of New York, Stony Brook, NY 11794 USA (e-mail: yang@ece.sunysb.edu).

Digital Object Identifier 10.1109/TNET.2006.890113

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 uses 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 the Dijkstra's algorithm. In this paper, we will study the same problem but will give a completely different and more direct way for solving it without using auxiliary graphs, and the resulting algorithm, called the Longest Segment Algorithm, is much simpler than the algorithm given in [4] and has linear time complexity of $O(tk)$ where t is the number of nodes on the path and k is the number of wavelengths per fiber. We will also consider the case when some nodes, called the "critical nodes", have fewer available wavelength converters than others, and study the problem of setting up a lightpath using converters in critical nodes only when necessary and give the Label Extending Algorithm that runs in $O(tk)$ time as well.

In addition, using similar ideas as Label Extending Algorithm, we will also give an algorithm for a special case of the *joint* routing and wavelength assignment problem. In [7] and [9] this problem was solved by first assigning costs to wavelength channels and wavelength converters and then finding an optimal lightpath which is defined as a lightpath with minimum total cost, including the wavelength cost and the conversion cost. In this paper, we will consider the same problem, however, we make the following simplifications: 1) wavelength channels have the same cost; 2) wavelength conversions have the same cost; 3) the cost of wavelength conversion is much higher than the cost of wavelength channel. The reasons for these simplifications, respectively, are: 1) wavelength channels typically have the same bandwidth; 2) when converting a wavelength to different wavelengths with full-range wavelength converter, although the physical characteristics such as power consumption and tuning time may differ, the differences are typically not large and should be hidden to upper layer protocols; 3) wavelength channels are more abundant than wavelength converters and propagation loss of the optical signals can be compensated for by optical amplifiers that are much cheaper than wavelength converters. Therefore, though only a special case of the more generalized problem, the problem we consider clearly has sufficient practical interest. 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 [7] and [9] the more generalized problem needs at least $O(Nk^2 + Lk + Nk\log(Nk))$ time.

Multicast, which is to send information from one source to multiple destinations, is also considered in this paper. In a communication network, a multicast connection is usually realized by establishing a multicast *tree* covering all nodes involved, where a tree is defined as a connected graph with no cycles. In WDM networks, since there are multiple wavelengths on a link, multicast can be supported by finding a *light-tree* [19]. Finding an optimal light-tree in WDM networks is NP-hard [1], [2], and can be solved, similarly to unicast, by breaking into two sub-problems and solving them one by one: 1) find a link-tree that covers all 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 [1], [2], and we will focus on the second subproblem in this paper.

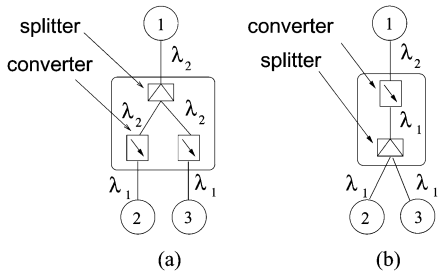

Fig. 1. Under the Converter Spilt Model, one wavelength converter can be saved.

In [2], a linear time algorithm was given for setting up a light-tree in a link-tree using dynamic programming method, however, it was under a different scenario without trying to minimize the total number of conversions. In [1], a linear time algorithm was given for setting 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 multicast model that will reduce the number of wavelength conversions. In [1], it was 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, two wavelength converters have to be used, 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, only one wavelength converter is needed to convert λ_2 to λ_1, thus saving one converter, as shown in Fig. 1(b). In this paper, the two multicast conversion models are called the Converter No-Split Model and the Converter Split Model, respectively. 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. Through simulation studies we will also show that with our heuristics, the Converter Split Model saves a significant number of converters comparing to the Converter No-Split Model.

The rest of the paper is organized as follows. Section II studies unicast problems, in which Section II-A gives the Longest Segment Algorithm for setting up a lightpath on a given link-path

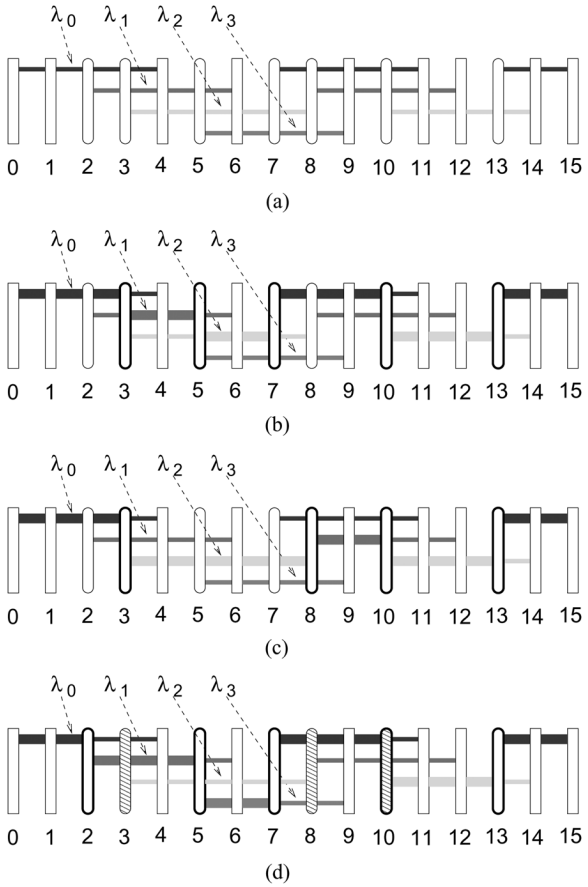

Fig. 2. The state of a link-path with 16 nodes and 4 wavelengths. Wavelength convertible nodes are shown as rectangles with round corners. Wavelength channels chosen by the algorithms are shown as wide line segments. Wavelength convertible nodes performing wavelength conversion are shown as rectangles with heavy edges. (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. The shaded nodes, nodes 3, 8, and 10, are critical nodes.

with minimum number of wavelength conversions, Section II-B gives the Label Extending Algorithm for setting up a lightpath on a given link-path while considering the availabilities of wavelength converters at different nodes, and Section II-C gives the Label Searching Algorithm for setting up an optimal lightpath network-wide. Section III studies multicast problems and gives heuristics for setting up a light tree on a given link-tree with minimum number of wavelength conversions. Finally, Section IV concludes the paper.

II. Wavelength Assignment for Unicast

In a network, a *link-path* is defined as several consecutive links. If wavelength λ_i is currently unused on all links along a link-path, these wavelength channels 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, where wavelength convertible nodes are shown as 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 as wide line segments and nodes that perform wavelength conversions are shown as rectangles with heavy edges.

TABLE I
Longest Segment Algorithm

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

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.

The source node is at one end of the path and is given index 0. Other nodes are indexed according to their distances to the source. For example, if there are t links between the source and the destination, the nodes are indexed as $0, 1, \ldots, t$. Node u is said to be able to *find* node v with $v-u$ hops if $v > u$ and there exists a wavelength continuous segment from u to v. The *furthest reachable wavelength convertible node* from node u, or abbreviated as the *FRC* node from u, is defined as the wavelength convertible node with largest index that can be found by u. If $v > u$ and there is a lightpath from u to v, v can be *reached* 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, and the FRC node from node 0 is node 3. Node 5 can be reached by node 0, although it cannot be found by it.

The algorithm is described in Table I. The basic idea is to extend the current lightpath to as far as possible to reach more nodes in each step, until the destination node is reached. To extend the lightpath is to choose a wavelength convertible node that has already been reached as the *extending point*, and extend the path to the node with the largest index (the furthest away from the source) that can be found by the extending point, because all such nodes can be reached by the source node. The extending point is chosen as the FRC node from the current extending point. Initially, the source node is the first extending point.

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 less converter is used than that in Fig. 2(b).

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

Proof: Clearly the theorem is true if the destination can be found by the source. In the following we consider the case when wavelength conversion has to be used for the source to reach the

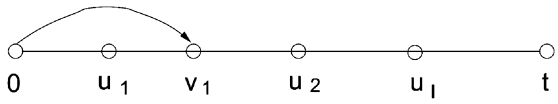

Fig. 3. If node 0 can reach node v_1 and v_1 is wavelength convertible, u_1 cannot be the FRC node from node 0.

destination. Let the lightpath found by the algorithm be Ψ. Ψ will consist of several, say, $I+1$, wavelength continuous segments, denoted by $[0, u_1], [u_1, u_2], \ldots, [u_I, t]$, 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. Consider the first wavelength continuous segment in Ψ. We claim that from node 1 to u_1, there is at least one wavelength conversion on Φ. Suppose the claim is not true, that is, on Φ, none of node $1, 2, \ldots, u_1$ converts wavelength. Since there must be at least one wavelength conversion in Φ, the conversions are carried out by nodes with larger indexes than u_1. Let the first such node be v_1. Note than $v_1 > u_1$, and 0 can find v_1, which contradicts the fact that u_1 is the FRC node from 0. Therefore, there must be at least one wavelength conversion from node 1 to u_1 on lightpath Φ.

Similarly, we can show that in each of the first I segments of Ψ, there must be at least one wavelength conversion on lightpath Φ. Therefore, the number of conversions used in Ψ is no more than that in Φ. Since Φ can be any lightpath, Ψ must use the minimum number of conversions. ■

1) Implementation and Complexity of the Longest Segment Algorithm: The Longest Segment Algorithm can be implemented as follows. The state of a link can be represented by a $1 \times k$ binary vector, where an element is "1" if the corresponding wavelength channel is not used on that link and "0" otherwise. The link vector of the link between node x and $x+1$ is denoted as $L_{x,x+1}$.

Note that only a series of AND operations over the link vectors are needed to check whether t can be found by a node x: starting with an all "1" vector, AND with $L_{x,x+1}, L_{x+1,x+2}, \ldots, L_{t-1,t}$. If the result is not all zero after ANDing $L_{t-1,t}$, t can be found by x. If the result becomes all zero after ANDing, say, $L_{y,y+1}$, then x can find as far as y. In this case suppose the FRC node from x is x' which will be the next extending point. Note that $x' \leq y < x''$ where x'' is the FRC node from x'. As a result, a link vector will be involved in the AND operation at most twice. Thus, the running time for finding the minimum number of conversions is $O(t)$.

From the proof of Theorem 1 it is clear that the extending points should carry out wavelength conversions. To set up the lightpath, it is also needed 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 Considering the Availabilities of Converters at Different Nodes

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. This problem can be formalized as follows. Categorize wavelength convertible nodes into two classes: If the number of available converters at a node is less than a threshold, the node is considered "critical"; otherwise, it is "noncritical". Wavelength conversion taking place at critical nodes and noncritical nodes are called critical conversion and noncritical conversion, respectively. An assignment is measured by a pair of integers, (c, n), called the *cost* of the lightpath, where c is the number of critical conversions and n is the number of noncritical conversions. Assignment 1 is better than assignment 2 if its cost is lexicographically smaller than the cost of assignment 2, that is, either if $c_1 < c_2$ or if $c_1 = c_2$ and $n_1 < n_2$. Later in this subsection we will also simply say that (c_1, n_1) is smaller than (c_2, n_2) and denote the relation as $(c_1, n_1) \prec (c_2, n_2)$. The optimal assignment is defined as the one with the smallest cost.

TABLE II
LABEL EXTENDING ALGORITHM

```
Extend the lightpath at the source.
while t is not labeled
  if non-critical nodes were labeled in the last extension
    Let u be the one with the largest index.
    Extend the lightpath at u.
  else
    if the last extending point w is non-critical
      Let the label of w be the "search label"
    else
      Let the label following w be the "search label"
    end if
    while TRUE
      Among critical nodes with the search label and larger
      indices than w, find v, the one with the largest index.
      if v exists
        Extend the lightpath at v
        break from the inner wile loop.
      else
        Set new search label be the label
        following the current search label
        if no such label
          t cannot be reached. exit.
        end if
      end if
    end while
  end if
end while
```

The algorithm for finding the optimal assignment is shown in Table II. The basic idea is to give labels to nodes where a label represent the cost of the minimum cost path from the source node to this node, until the destination is labeled. Like the Longest Segment Algorithm, in each step, it will choose a node as the extending point, say, u, and will give labels to nodes with larger indexes than u that can be found by u based on the label of u. Let the label of u be (c, n). If u is a noncritical node, the label of the new nodes will be $(c, n+1)$; if u is a critical node, the label of the new nodes will be $(c+1, n)$. At the first step, the source is the extending point and all nodes that can be found by the source are given label (0,0).

The algorithm chooses the new extending point as follows. If in the last extension some noncritical nodes were labeled, the one with the largest index will be the next extending point. Otherwise, suppose the last extending point is w. It will first determine the *search label*. If w is a noncritical node, the search label will be the label of w; if w is a critical node, the search label will be the one *following* that of w, where the label of node v is said to *follow* the label of w if v is labeled in the next step after w is labeled. It will choose u as the new extending point, where u is larger than w and is the critical node with the search label and is the one with the largest index.

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 noncritical 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. There is no noncritical node labeled at this step. The algorithm sets the search label to be (0,2), and finds that nodes 8 and 9 have label (0,2) and larger indexes than 7. Node 8 is the next extending point and gives the label (1,2) to node 12. Since node 12 is not wavelength convertible, the algorithm searches among nodes with label (0,3), which are nodes 10 and 11, and node 10 becomes the new extending point and gives the label (1,3) to node 13. Node 13 is a noncritical node and can find node 15. The algorithm then terminates.

The following two facts can be easily verified: 1) when u is labeled, all nodes with indexes less than u must have been labeled, and 2) if $u < v$ and v can be reached from the source with (c, n) conversions, u can be reached from the source with no more than (c, n) conversions.

The following theorem says that the Label Extending Algorithm finds the optimal wavelength assignment.

Theorem 2: The following two invariants hold throughout the execution of the algorithm: 1) The label given to each node is the smallest cost to reach this node, or "the correct label", and 2) If some nodes are given label (c, n) in an extension, all nodes that can be reached with cost no more than (c, n) have been labeled by that extension.

Proof: We prove it by induction on the steps of extension. The two invariants are obviously true at the first step when all nodes that can be found by the source are labeled (0,0). Suppose it is also true for the first $H-1$ extensions.

Consider the H_{th} extension. If it extends at a noncritical node u with label (c, n), some nodes will be given label $(c, n+1)$. If the first invariant is not true, then among the newly labeled nodes, there exists a node, say, v, that can be reached from the source with (c', n') conversions where $(c', n') \prec (c, n+1)$. Note that c' cannot be smaller than c, since if so, the label of u will not be correct which contradicts our induction hypothesis that the first invariant is true. Thus, $c' = c$ and $n' < n+1$. But this contradicts our induction hypothesis that the second invariant is true by step $H-1$. Thus, the first invariant is true at step H.

To see the second invariant is also true at step H, suppose there are some node, say, z, that can be reached with no more than $(c, n+1)$ conversions but not labeled. Since z cannot be found by u, on the lightpath from the source to z, the last wavelength conversion must take place at a node, say, w, where $w > u$. Note that w must have been labeled by step $H-1$, since if not, it must be the case that w can only be reached with more than (c, n) conversions and thus z cannot be reached with only $(c, n+1)$ conversions. Since u is the noncritical node with label (c, n) with the largest index, w can only be a critical node with label (c, n). However, in this case, z cannot be labeled as $(c, n+1)$. Thus, the second invariant is also true at step H.

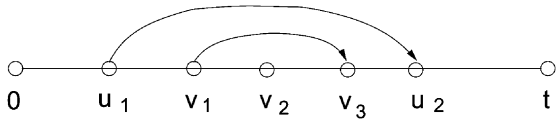

Fig. 4. If u_1 can find u_2, the Label Extending Algorithm will not convert wavelength at v_2.

If the H_{th} extension extends at a critical node, say, u, with label (c, n), some nodes will be labeled as $(c+1, n)$. If the first invariant is not true, some newly labeled nodes, say, v, can be reached with (c', n') conversions where $(c', n') \prec (c+1, n)$. Since v was not labeled before the H_{th} extension, v can only be found by nodes with an index no less than p where p is the node with the smallest index with label (c, n). By the induction hypothesis, to reach these nodes, at least (c, n) conversions are needed. By the algorithm, there are no noncritical nodes among these nodes. It follows that v cannot be reached with less than $(c+1, n)$ conversions.

If the second invariant is not true, there is a node w that can be labeled as $(c+1, n)$ but not labeled. Note that since w cannot be found by u, it can only be found by nodes with larger indexes than u. Suppose along the lightpath from the source to w, the last wavelength conversion takes place at q. Since the first invariant is true at the H_{th} step, q cannot be among the nodes labeled at the H_{th} step or have a larger index than them. Thus, q is among the nodes with larger indexes than u and is labeled before the H_{th} step. By the algorithm, q must be a critical node and by the induction hypothesis and the choice of u, to reach q, more than (c, n) conversion is needed. Thus, the label of w cannot be $(c+1, n)$. ■

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.

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 two converters. Consider the example shown in Fig. 4 where $W = 2$. A contradiction can be found immediately if the Label Extending Algorithm uses more than three conversions between u_1 and u_2: since there is a wavelength continuous segment between u_1 and u_2, v_1 can find v_3 without converting wavelength at v_2 and thus the wavelength converter at v_2 could not have been used. With similar arguments, it can be shown 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

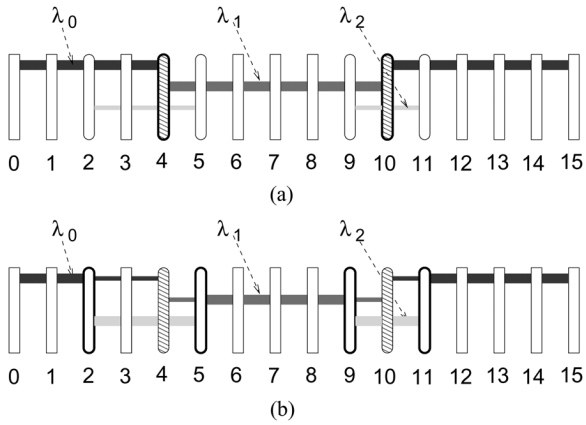

Fig. 5. The bound in Theorem 3 is tight. 4 and 10 are critical nodes. (a) Longest segment algorithm. (b) Label extending algorithm.

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.

1) Implementation and Complexity of the Label Extending Algorithm: The state of links can still be represented by k-bit binary vectors. The algorithm does two jobs: searching for wavelength convertible nodes and give labels to nodes. The former needs only $O(t)$ time, where t is the number of links between the source and the destination, since a node is checked by the algorithm for no more than a constant number of times. The latter can also needs only $O(t)$ time. To see this, note that to determine which nodes can be labeled is to do AND operations on the k-bit binary vectors until the ANDed result becomes all 0. We note that in Label Extending Algorithm, one vector can be involved in this operation at most 3 times. To see this, consider the time when the algorithm finished an extension and labeled some nodes. The vectors of the links connecting the newly labeled nodes would have been ANDed exactly once. If among the newly labeled nodes there is a noncritical node, these vectors will be ANDed at most one more time. If there is no noncritical node, these vectors will be ANDed at most two more times. This is because, for example, consider the first time when the algorithm failed to find a noncritical node. There can be at most two sets of nodes with indexes larger than the last extending point: nodes with labels $(0, n-1)$ and $(0, n)$ where $(0, n)$ is the label given to the newly labeled nodes. In each of these two sets, the algorithm will choose at most one node as extending point, which is why the vectors will be ANDed at most two more times. Similar facts can be verified for the rest extensions. Hence, a vector is involved in the AND operation no more than 3 times and the algorithm needs only $O(t)$ time in labeling.

Unlike the Longest Segment Algorithm, not all extending points perform wavelength conversion. To determine where to carry out wavelength conversion, each node remembers the node that labeled it as its parent. At first, the parent of t will carry out wavelength conversion. After that the parent of the node that has been most recently determined to carry out wavelength conversion will carry out wavelength conversion, until such node becomes the source node 0. Similar to the Longest Segment Algorithm, the time for setting up a lightpath is $O(tk)$, where k is the number of wavelengths per fiber.

TABLE III
LABEL SEARCHING ALGORITHM

Label all nodes that can be found by s.
$I \leftarrow 0$;
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
  $I \leftarrow I+1$
end while

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 scheduler, because the entire network has to be searched to find the optimal lightpath.

We still measure a lightpath by a pair of integers called the cost of the lightpath denoted as (c, h), where c is the number of wavelength conversions and h is the number of hops. For reasons described in Section I, a lightpath is considered better than another if its cost 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. The optimal lightpath is defined as a path with the lexicographically smallest cost.

The Label Searching Algorithm for solving this problem is shown in Table III. The idea is simple and can be roughly understood as equivalent to doing Breadth First Search (BFS) starting at wavelength convertible nodes. Let s be the source and t be the destination. At first, all nodes are not labeled and the algorithm will do k BFS starting at s, one for each wavelength, and give label $(0, p)$ to nodes that can be found by s with a minimum of p hops. The is the first "round" of searching. If the destination has been labeled, the optimal lightpath has been found. Otherwise, it will do another round of searching starting at the labeled wavelength convertible nodes to label more nodes. If no more nodes can be labeled and the destination is still not labeled, it will start a new round of searching at the newly labeled wavelength convertible nodes, and so on, until the destination is labeled.

To be more specific, suppose after the first round, u is a wavelength convertible node with the smallest label, say, $(0, h)$. Note that all unlabeled nodes that can be found by u with one hop can be labeled as $(1, h+1)$. Since first, it must take at least one wavelength conversion for the source to reach them. If they can be reached with one conversion but less than $h+1$ hops, suppose along one of such paths wavelength conversion takes place 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 found with one hop from wavelength convertible nodes with label $(0, h)$, the algorithm starts to label nodes that can be found with two hops from wavelength convertible nodes with label $(0, h)$ and nodes that can be found with one hop from wavelength convertible nodes with label $(0, h+1)$. All such nodes can be labeled as $(1, h+2)$, for reasons similar to those described earlier. After that, the algorithm gives label $(1, h+i)$ to nodes that can be found with $i-j$ hops from wavelength convertible nodes with label $(0, h+j)$ for all $0 \leq j < i$, until all nodes that can be found by wavelength convertible nodes labeled in the first round have been labeled.

If the destination is still not labeled, a new round of searching will be needed, 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 until no new nodes can be labeled. Since the label of a node is the cost of the optimal lightpath from the source to it, when the destination is labeled, the desired optimal lightpath is found.

To establish the lightpath, at first, t will check on which wavelength it was first found, and then trace back to u_1 on this wavelength where u_1 is the wavelength convertible node that found t. u_1 will check on which wavelength it was first found, and use this wavelength to trace back to u_2 which is the node that found it, and so on, until the path reaches the source s. The complexity of this algorithm is $O(Nk + Lk)$, where N is the number of nodes, L is the number of links, and k is the number of wavelengths 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 on different network topologies by simulation. The connection requests are assumed to arrive at nodes according to a Poisson process, and the duration of a connection follows the exponential distribution with an average of 1 time unit. The traffic intensities between every pair of nodes are assumed to be the same. The network performance is measured by overall *blocking probability* as a function of the arrival rate at a node. Each point in the figures is obtained after generating 1 000 000 requests. In figures, the First Fit Algorithm, the Longest Segment Algorithm, the Label Extending Algorithm, and the Label Searching Algorithm are referred to as "FF", "LSeg", "LExt", and "LSear", respectively. In all simulations the number of wavelengths per fiber is 16.

Fig. 6 shows the results for a bidirectional ring network with 16 nodes. Each node has eight wavelength converters, and the threshold of the Label Extending Algorithm is set to be 2. The Label Searching Algorithm is not shown because it is the equivalent to the Longest Segment Algorithm in a ring network where there is no routing problem involved. It can be observed that both the Longest Segment Algorithm and the Label Extending Algorithm outperform the First Fit Algorithm by a significant amount when the arrival rates are not too high. However, to much of our surprise, the performance of the Longest Segment Algorithm and the performance the Label Extending Algorithm are almost the same. By examining the data we found that while the Label Extending Algorithm uses less converters at critical nodes, overall it uses more converters and reduces the number and the length of wavelength continuous segments in the network and forces future lightpaths to use more converters. This offsets the benefits of using less converters at critical nodes and makes the performance of the two algorithms very close.

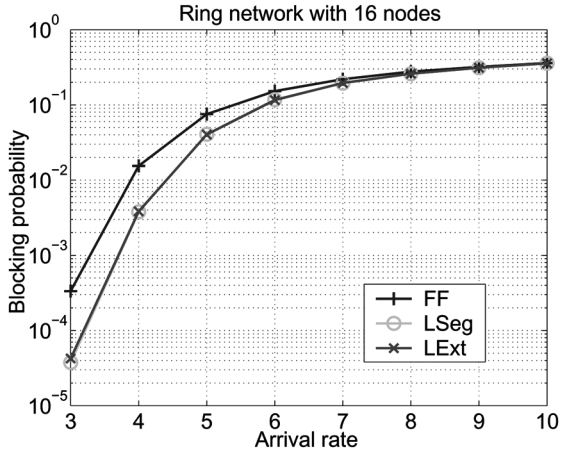

Fig. 6. Blocking probability of a bidirectional ring network with 16 nodes and 16 wavelengths per fiber.

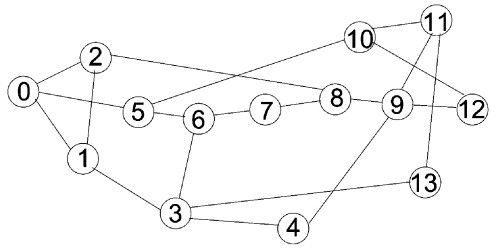

Fig. 7. The NSF network.

More results are shown in Figs. 8 and 9. Fig. 8 shows the results for the well-known 14-node NSF network which is shown in Fig. 7. In the simulations of Fig. 8, each node has 8 wavelength converters, the threshold of the Label Extending Algorithm is 2, and each pair of nodes keeps up to four link-disjoint paths as candidate link-paths. Fig. 9 shows the results for a randomly generated network with 50 nodes where the average node degree is 5. In the simulations of Fig. 9, each node has 16 wavelength converters, the threshold of the Label Extending Algorithm is 8, and each pair of nodes keeps up to eight link-disjoint paths as candidate link-paths. From these two figures, similar facts can be observed as in Fig. 6. In addition, as expected, the performance of the Label Searching Algorithm is better than others, especially when the arrival rate is small. However, we believe the Label Searching Algorithm may not be suitable for very large networks, since it may require every node to have full knowledge of the network, which is hard and expensive to realize in large networks.

III. Wavelength Assignment for Multicast

In this section, we study on-line wavelength assignment problem in WDM networks for multicast traffic. We will consider setting up a light-tree in a given link-tree using minimum number of converters. First, we introduce some definitions and notations.

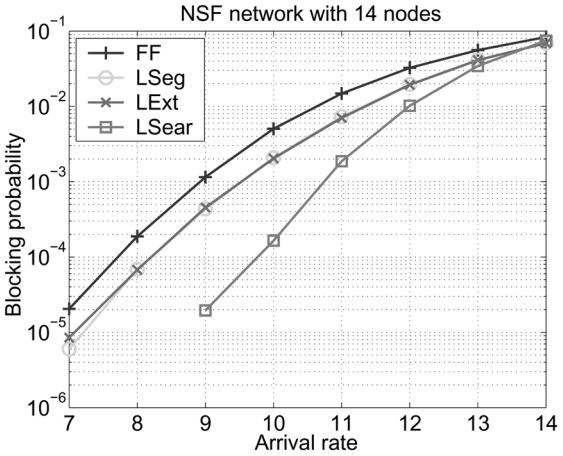

Fig. 8. Blocking probability of the NSF network with 16 wavelengths per fiber.

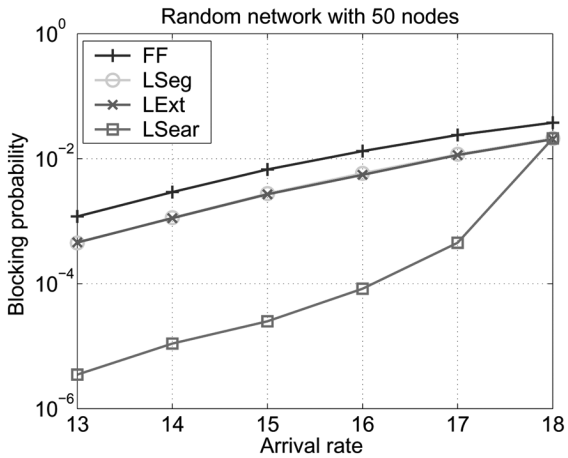

Fig. 9. Blocking probability of a random network with 50 nodes and 16 wavelengths per fiber, where the average node degree is 5.

TABLE IV
LIST OF SYMBOLS IN SECTION III

$\Delta_B(i)$:	minimum number of converters needed when B is entered 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 fork node r when the light-tree reaches r on λ_i.

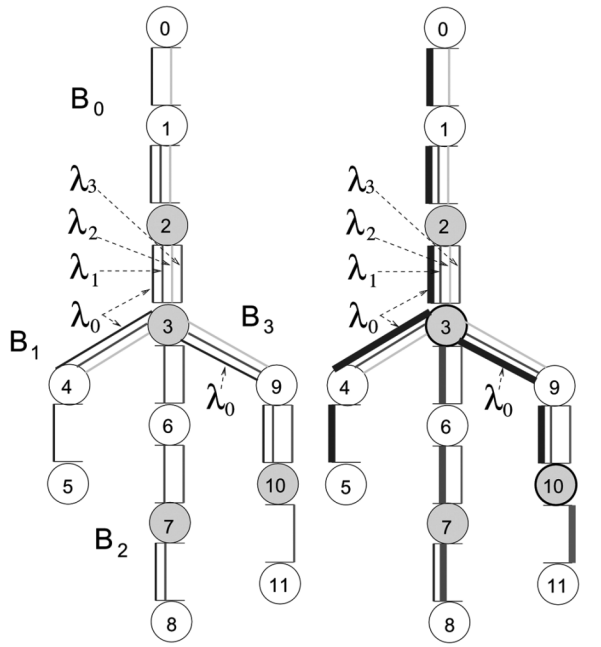

Fig. 10. A link-tree with four wavelengths per fiber and the optimal light-tree, where 2, 3, 4, and 10 are wavelength convertible nodes.

Definitions and Notations

For a tree denoted as T, a node with *out degree* more than one will be called a *fork node*. Each part of the tree starting at a fork node is called a *branch* and the fork node is called the *root node* of this branch. Since a fork node is the root node of more than one branches, the root node of a branch is not counted as a node in the branch. A tree may have many branches, and a branch itself may also have *child* branches. A branch with 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 its root node there are i fork nodes, including the root node of the branch. The *tree level* is the maximum branch level.

We say the light-tree *enters* a branch B on λ_i if on the first link of B λ_i is used in the light-tree. We say B can be *covered* by λ_i if there exists a wavelength assignment in B such that there is a lightpath from the root of B to every node in B when the light-tree enters B on λ_i. The number of wavelength conversions used in B in such an assignment is called the covering cost of this assignment, and the minimum cost among all assignments is defined as $\Delta_B(i)$. $\Delta_B(i) = \infty$ if λ_i cannot cover B, which may happen, for example, when λ_i on the first link has already been occupied. Let $\delta_B = \min\{\Delta_B(i)\}$ among all $i \in [0, k-1]$. If $\Delta_B(i) = \delta_B$, λ_i is called *optimal entering wavelength* of branch B. For simplicity, the set of $\Delta_B(i)$ for all $i \in [0, k-1]$ is denoted as $\Delta_B()$.

When the light-tree reaches a fork node r on λ_i, we say all child branches of r can be covered if there is a wavelength assignment such that there is a lightpath from r to every node in its child branches. The number of wavelength conversions needed in such an assignment, including the conversions at r and the conversions in its child branches, is called the covering cost of this assignment, and the minimum cost among all assignments is defined as $\Upsilon_r(i)$. $\Upsilon_r(i)$ can be ∞, for example, when there is no available converter at r and λ_i cannot cover one of r's child branches. The set of $\Upsilon_r(i)$ for all $i \in [0, k-1]$ is denoted as $\Upsilon_r()$.

Table IV lists some of the frequently used notations. Also, throughout this section, we use B to denote the branches, t to denote the root of a branch, and r to denote the fork node of a branch.

Fig. 10 is an example for illustrating the definitions and notations. The link-tree is shown in the left part of the figure. It has 12 nodes, where node 0 is the root of the tree and node 3 is the fork node. Node 3 has three child branches denoted as B_1, B_2, and B_3, respectively, and node 3 is the root node of all these

TABLE V
ALGORITHM FOR FINDING OPTIMAL LIGHT-TREE

```
i ← tree level;
while i ≥ 0
    Find Υ() for all fork nodes only in non-simple
    branches of level i;
    Find Δ() for all branches of level i;
    i ← i − 1;
end while
return δ_T;
```

branches. 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 of B_2 is λ_1. $\Upsilon_3(0) = 2$ since λ_0 covers B_1 at cost 0, covers B_3 at cost 1, and can be converted to λ_1 at node 3 to cover B_2 at cost 0. In fact, $\Upsilon_3() = \{2, 2, 4, 4\}$.

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

A. Outline of the Algorithm

The idea of our algorithm, as shown in Table V, is quite simple: Find $\Delta_T()$ in a "bottom-up" way where T is the tree. Basically, we start from branches at the lowest level that are all simple branches, and find their $\Delta_B()$. Then "move up" one level, find $\Delta_B()$ of higher level branches according to the $\Delta_B()$ of their child branches obtained in the previous round. Keep on moving up until $\Delta_B()$ of the highest level, i.e., $\Delta_T()$, is found. The minimum number of conversions needed in a light-tree is δ_T. At this time, the minimum covering cost of each branch for each wavelength can also be determined and the optimal light-tree can be established.

The two major jobs the algorithm does are finding $\Delta_B()$ and $\Upsilon_r()$. In the following, we first describe how to find $\Delta_B()$.

B. Finding $\Delta_B()$

Our way of finding $\Delta_B(i)$ for λ_i is to apply the Longest Segment Algorithm in Table I. To find $\Delta_B(i)$ for a simple branch B, we can mask all wavelengths other than λ_i on the first 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 found by the algorithm is $\Delta_B(i)$.

When B is not a simple branch, $\Delta_B(i)$ can be found as follows. Let t be the root node of the branch and let r be the fork node of the branch. Let Ω be the set of nodes in B excluding r and those in the child branches of r. Note that at this time, $\Upsilon_r()$ should have been found. There are two cases. 1) There is no wavelength convertible node in Ω. In this case, clearly $\Delta_B(i) = \Upsilon_r(i)$ if there is a wavelength continuous segment on λ_i from t to r, otherwise $\Delta_B(i) = \infty$. 2) There are wavelength convertible nodes in Ω. Suppose the one nearest to r is u. Consider the set of wavelengths on which there is a wavelength continuous segment from u to r and let $\epsilon = \min \Upsilon_r(j)$ where λ_j is a wavelength in this set. Again, we mask all other wavelengths on the first link except λ_i and apply the Longest Segment Algorithm from t to r, and suppose η_i converters are needed and the last extending point is v. For all wavelengths on which there is a wavelength continuous segment from v to r, if there is λ_l where $\Upsilon_r(l) = \epsilon$, 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.

Clearly, it takes $O(|\ \Omega\ |)$ time to find $\Delta_B(i)$. Since this is required for all k wavelengths, overall the time spent on this job is $O(Nk)$ where N is the total number of nodes in the tree.

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

Theorem 4: If there are wavelength convertible nodes in Ω, if $\Delta_B(j) \neq \infty$, then either $\Delta_B(j) = \delta_B$ or $\Delta_B(j) = \delta_B + 1$.

Proof: Suppose the closest wavelength convertible node to root t is w. For any wavelength λ_j, if $\Delta_B(j) \neq \infty$, there must be a wavelength continuous segment on λ_j from t to w. If $\Delta_B(j) \neq \delta_B$, suppose the optimal light-tree on this branch leaves w on λ_l. We can convert λ_j to λ_l at w, and hence use a total of no more than $\delta_B + 1$ converters. ■

Note that the property in Theorem 4 was unaware of in [1] and part of the algorithm in [1] is unnecessary.

C. Finding $\upsilon_r()$

In this subsection, we will study the problem of finding $\Upsilon_r()$. We will show that the problem is NP-hard, since it is related to the *Set Covering Problem* which is known to be NP-hard. We show that it can be solved approximately by consecutively running three greedy algorithms shown in Tables VII, VIII, and IX, respectively. For convenience, we call them Greedy 1, Greedy 2, and Greedy 3.

1) Determining Feasibility: The first question is to determine feasibility: Suppose there are C converters left at fork node r, can all child branches of r be covered while using no more than C converters at r, if the light-tree reaches r on a certain wavelength?

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

Proof: To see this, we will need the *Set Covering Problem* which is known to be NP-complete: Given a whole set $S = \{e_0, e_1, \ldots, e_{m-1}\}$ and a collection of subsets denoted as $S_0, S_1, \ldots, S_{h-1}$, does there exist C subsets such that every element is in at least one of the C subsets?

Given any instance of the Set Covering Problem, construct an instance of this problem as follows. Let $k = h + 1$ and let the wavelength that reaches r be λ_{k-1}. Wavelength λ_l where $0 \leq l \leq k-2$ will correspond to subset S_l. The fork node r will be given m child branches where branch B_j, $0 \leq j \leq m-1$, will correspond to element e_j. No branch can be covered by λ_{k-1}. If $e_j \in S_l$, B_j can be covered by λ_l. It is then clear that a *yes* answer to this constructed instance will give a *yes* answer to the instance of the Set Covering Problem and vice versa. ■

As can be seen from the proof, this problem is actually exactly the set covering problem. The set covering problem can be solved approximately by a simple greedy algorithm shown

TABLE VI
GREEDY SET COVER ALGORITHM

```
C' ← 0;
while Π ≠ ∅
    find S_l that covers the most elements in Π;
    if no such subset can be found break.
    Remove all elements in Π that can be covered by S_l;
    C' ← C' + 1;
end while
```

TABLE VII
GREEDY 1

```
Let Π be the set of branches that cannot be covered by λ_i;
For every λ_l, let S_l be a subset where an element in Π
is in S_l if that branch can be covered by λ_l.
Apply the Greedy Set Cover Algorithm to find C'.
If C' ≤ C return yes else return no.
```

TABLE VIII
GREEDY 2

```
Let branch B_j be an element if Δ_Bj(i) ≠ δ_Bj.
For every λ_l, let S_l be a subset where element B_j is
in subset S_l if Δ_Bj(l) = δ_Bj.
Apply the Greedy Set Cover Algorithm, find C'.
return C' + Σ_{j=1}^{m} δ_Bj
```

in Table VI where the whole set is denoted by Π. In each step, it 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 $C'/C_{\text{opt}} \leq \ln m$. Recent results show that no polynomial algorithm has a ratio smaller than $\ln m$[16].

The algorithm for determining the feasibility when the light-tree reaches r on λ_i is shown in Table VII. If $C' \leq C$, the child branches can be covered and the algorithm returns a yes.

2) Minimizing Conversion Cost: Suppose Greedy 1 determines that the request is feasible. The next question is to find a wavelength assignment with minimum cost. Note that if the minimum covering costs of the child branches are either 0 or ∞ for any wavelength, this problem reduces to the optimization version of the problem studied in Theorem 5. Therefore, we have

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

We hereby give a greedy algorithm to solve this problem shown in Table VIII. Note that if $\Delta_{Bj}(i) = \delta_{Bj}$, B_j can be covered with minimum cost without wavelength conversion at r, therefore, the optimal assignment must enter branch B_j with λ_i. Thus, only branches where $\Delta_{Bj}(i) \neq \delta_{Bj}$ need to be considered. The input to this algorithm is $\Delta()$ for all such child branches. The output is the number of wavelength converters needed. Note that in the algorithm, a branch must be entered via an optimal entering wavelength. We next show that

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

Proof: Given any optimal assignment, if the entering wavelength of a branch denoted by λ_p is not an optimal entering wavelength, we can convert λ_i to an optimal entering wavelength at r denoted by λ_l, 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. Since the assignment given by Greedy 2 also enters every branch via its optimal entering wavelength, the difference between the optimal algorithm and the greedy algorithm is the number of wavelength converters used at r. Denote them as C_{opt} and C', respectively. As explained previously, $C'/C_{\text{opt}} \leq \ln m$. Therefore, the performance ratio is

$$\frac{C' + \sum_{j=0}^{m-1} \delta_{Bj}}{C_{\text{opt}} + \sum_{j=0}^{m-1} \delta_{Bj}} \leq \frac{C'}{C_{\text{opt}}} \leq \ln m$$

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

3) Minimizing Conversion Cost With Limited Number of Available Converters at Fork Node: Greedy 2 tries to minimize the total number of wavelength conversions, however, it does not consider the total number of available converters at r. If $C' \leq C$, i.e., the number of conversions it uses at r is no more than the number of available converters, Greedy 2 gives a feasible assignment. Otherwise, the assignment is not feasible and we have to use other method to find an assignment that uses as few converters as possible under the constraint that no more than C converters at r are used.

Clearly, the problem studied in Theorem 6 is a special case of this problem. Thus, we have:

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 IX gives an algorithm to solve this problem approximately when the light-tree reaches r on λ_i. The idea is to start with a feasible assignment which is the one found by Greedy 1, and use wavelength conversions at r to reduce the number of conversions in each step. In the description of the algorithm, a wavelength is "unused" if no wavelength converter is used at r to convert λ_i to it, otherwise it is "used". If the entering wavelength of B_j is λ_l, the *weight* of B_j is defined as $\Delta_{Bj}(l) - \delta_{Bj}$. If a converter is used at r to convert λ_i to a wavelength λ_p, if $\Delta_{Bj}(p) < \Delta_{Bj}(l)$, B_j will update its entering wavelength as λ_p and its weight will be reduced by $\Delta_{Bj}(l) - \Delta_{Bj}(p)$.

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 minimum covering cost of the optimal entering wavelength and those of other wavelengths may be unbounded. However, note that if there are wavelength converters on every branch, as shown in Theorem 4, the minimum covering costs differ at most

TABLE IX
GREEDY 3

Start with the feasible assignment found by Greedy 1. Suppose C' wavelength conversions are used at r. **while** $C' \leq C$ Find an unused wavelength that reduces the maximum amount of total weight of the branches. **if** no such wavelength can be found **break**. Use this wavelength and update the weights of the branches. $C' \leftarrow C' + 1$. **end while**

TABLE X
FINDING $\Upsilon_r(i)$

Run Greedy 1. **exit if** not feasible. Run Greedy 2. **return** the assignment given by Greedy 2 **if** it is feasible. Run Greedy 3. **return** the assignment found by Greedy 3.

by one, and in this case, it is not hard to show that the cost difference between the assignment found by Greedy 3 and the optimal assignment is no more than $C(1 - 1/\ln m) + m$.

We next show that:

Theorem 9: In each iteration in the loop of Greedy 3, the cost difference between the assignment found by Greedy 3 and the optimal assignment in the child branches (excluding the wavelength conversions at the fork node) will decrease exponentially at a rate no less than $(1 - 1/C)$.

Proof: Suppose when Greedy 1 has finished, the difference between the greedy and the optimal in the child branches is ω_0. Therefore, if all C_{opt} wavelengths in the optimal assignment to which λ_i is converted at r are used, the difference will decrease by exactly ω_0. Thus, at least one of the wavelengths in the optimal assignment will reduce the difference by ω_0/C_{opt}. Since the greedy algorithm always chooses the wavelength that decreases the maximum amount of branch weight, ω_1 which is the difference after the first execution of the loop in Greedy 3 satisfies $\omega_1 \leq \omega_0(1 - 1/C_{\text{opt}})$. In other words, ω_1 is at most a fraction of $(1 - 1/C_{\text{opt}})$ of ω_0. The same is also true for all following steps. Thus, the decreasing rate is no less than $(1 - 1/C)$, since $C \geq C_{\text{opt}}$. ■

The procedure for finding $\Upsilon_r(i)$ approximately for a wavelength λ_i is summarized in Table X.

For a fork node with m child branches, to find $\Upsilon_r(i)$, Greedy 1 and Greedy 2 need $O(mk)$ time and Greedy 3 needs $O(m^2k)$ time. Thus, 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.

D. Applying to the Converter No-Split Model

For completeness, we briefly discuss how to apply our algorithm to the Converter No-Split Model. First, note that under this model the definitions of the tree branch, the fork node, and correspondingly $\Delta_B()$ and $\Upsilon_r()$ are exactly the same and the optimal assignment can still be found in the bottom-up way as described in Table V. Also note that finding $\Delta_B()$ under the Converter No-Split Model is exactly the same as under the Converter Split Model. However, finding $\Upsilon_r()$ is no longer NP-hard. For example, to find $\Upsilon_r(i)$, we can first use λ_i to cover all branches at cost defined as $\Delta_{Bj}(i) - \delta_{Bj}$ for branch B_j (the cost can be ∞). Then we can find the branch with the largest cost and choose to use a wavelength converter at r to convert λ_i to its optimal entering wavelength to cover it at minimum cost, and repeat this until the costs of all branches become 0 or until there are no converters left at r. Note that this simple method finds $\Upsilon_r(i)$ because the output of wavelength converter cannot be split and only one branch needs to be considered at a time. As a result, our algorithm finds the optimal light-tree in a given link-tree in linear time.

In [1] a linear time algorithm for finding an optimal light-tree in a given link-tree under the Converter No-Split Model was also given. However, although both are linear time algorithms, our algorithm is much simpler and needs less computation. This is due to the major difference between the two algorithms: the basic element of our algorithm is a tree branch while the basic element of the algorithm in [1] is an individual node, and we only compute $\Delta()$ for a branch and $\Upsilon_r()$ for a fork node while [1] computes similar quantities for every node.

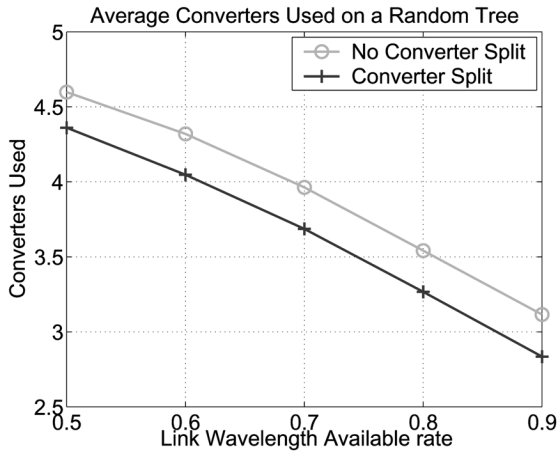

Fig. 11. Average number of converters needed to set up a light-tree in a random link-tree with average 39.4 nodes.

E. Performance Study for Multicast Algorithms

We have implemented our algorithm in software and applied it to randomly generated trees. In our simulations, on average the tree has three levels and 39.4 nodes, and there are a total of 16 wavelengths on the links. With probability 0.75, a tree node is wavelength convertible, and the number of available wavelength converters is randomly chosen from 1 to 10. Wavelength channels on the links are randomly chosen to be available or not, and the overall percentage of availability varies from 0.5 to 0.9. The data is collected after 100 000 runs. Both the results for Converter Split Model and the Converter No-Split Model are shown in Fig. 11. It can seen that the Converter Split Model saves about 6%–10% of the converters, which is the improvement we have achieved over [1].

IV. Conclusion

In this paper, we have studied the problem of setting up connections on-line 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 simple new algorithm that solves it in $O(tk)$ time, where t is the number of links and k is the number of wavelengths. We also considered the problem of setting up a lightpath while using wavelength converters at nodes with fewer available converters only when necessary, and gave an $O(tk)$ time algorithm. We then generalized this technique to WDM networks with arbitrary topologies and gave an algorithm that sets up an optimal lightpath network-wide in $O(Nk + Lk)$ time, 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 a light-tree on a given link-tree with minimum conversion cost. We proposed a new multicast conversion model that allows the output of the converter to split, which can save the conversion cost considerably. We showed that this problem is NP-hard, and then gave efficient heuristics to solve it approximately.

Acknowledgment

The authors would like to thank Prof. E. M. Arkin of SUNY at Stony Brook for the suggestions on the Set Covering Problem.

References

[1] B. Chen and J. Wang, "Efficient routing and wavelength assignment for multicast in WDM networks," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 1, pp. 97–109, Jan. 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, Oct. 2002.

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

[4] Y. Zhang *et al.*, "An efficient heuristic for routing and wavelength assignment in optical WDM networks," in *Proc. IEEE Int. Conf. Communications (ICC)*, 2002, vol. 5, pp. 2734–2739.

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

[6] B. Mukherjee, "WDM optical communication networks: Progress and challenges," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 10, pp. 1810–1824, Oct. 2000.

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

[8] K.-C. Lee and V. O. K. Li, "A wavelength-convertible optical network," *J. Lightw. Technol.*, vol. 11, no. 5, pp. 962–970, May-Jun. 1993.

[9] I. Chlamtac, A. Farag, and T. Zhang, "Lightpath (wavelength) routing in large WDM networks," *IEEE J. Sel. Areas Commun.*, vol. 14, no. 5, pp. 909–913, Jun. 1996.

[10] W. J. Goralski, *Optical Networking and WDM*, 1st ed. New York: McGraw-Hill, 2001.

[11] R. Ramaswami and K. N. Sivarajan, *Optical Networks: A Practical Perspective*, 1st ed. New York: Academic, 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, Apr. 1998.

[13] C. Law and K. Y. Siu, "Online routing and wavelength assignment in single-hub WDM rings," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 10, pp. 2111–2122, Oct. 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, Apr. 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, Apr. 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," in *Proc. STOC '97*, 1997, pp. 475–484.

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

[18] Z. Zhang and Y. Yang, "On-line optimal wavelength assignment in WDM networks with shared wavelength converter pool," in *Proc. IEEE INFOCOM*, 2005, pp. 694–705.

[19] L. H. Sahasrabuddhe and B. Mukherjee, "Light trees: Optical multicasting for improved performance in wavelength routed networks," *IEEE Commun. Mag.*, vol. 37, no. 2, pp. 67–73, Feb. 1999.

Zhenghao Zhang received the B.Eng. and M.S. degrees in electrical engineering from Zhejiang University, China, in 1996 and 1999, respectively. From 1999 to 2001, he worked in industry as an Embedded System Engineer. He received the Ph.D. degree in electrical engineering from the State University of New York at Stony Brook in 2006.

He is now a Postdoctoral Research Fellow in the Computer Science Department, Carnegie Mellon University, Pittsburgh, PA. His research interests include network security, computer systems, scheduling algorithm design, performance analysis, optical networking and wireless networking.

Yuanyuan Yang received the B.Eng. and M.S. degrees in computer science and engineering from Tsinghua University, Beijing, China, and the M.S.E. and Ph.D. degrees in computer science from The Johns Hopkins University, Baltimore, MD.

She is a Professor of computer engineering and computer science at the State University of New York at Stony Brook. Her research interests include high-speed networks, optical and wireless networks, and parallel and distributed computing and systems, Her research has been supported by the National Science Foundation (NSF) and U.S. Army Research Office (ARO). She has published extensively in major journals and refereed conference proceedings, and holds six U.S. patents in these areas.

Dr. Yang has served as an editor for IEEE Transactions on Parallel and Distributed Systems and is currently an editor for the *Journal of Parallel and Distributed Computing*. She has served on NSF review panels and program/organizing committees of numerous international conferences in her areas of research.