Dynamic Wavelength Assignment for WDM All-Optical Tree Networks

Poompat Saengudomlert, Eytan H. Modiano, and Robert G. Gallager*
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139
tengo@mit.edu, *modiano@mit.edu*, *gallager@mit.edu*

Abstract

We develop an on-line wavelength assignment (WA) algorithm for a WDM tree network with N end nodes. The algorithm dynamically supports all $\mathbf{k}$-port traffic matrices, where $\mathbf{k}$ denotes an integer vector $[k_1, ..., k_N]$ and end node i, $1 \le i \le N$, can transmit at most k_i wavelengths and receive at most k_i wavelengths. Our algorithm is rearrangeably nonblocking, uses the minimum number of wavelengths, and requires at most $d^* - 1$ lightpath rearrangements per new session request, where d^* is the degree of the most heavily used node in the worst-case traffic scenario. We observe that the number of lightpath rearrangements per new session request does not increase as the amount of traffic $\mathbf{k}$ in the network increases by an integer scaling factor. In addition, we found that wavelength converters cannot reduce the number of wavelengths required to support $\mathbf{k}$-port traffic in a tree network.

1 Introduction

In a wavelength division multiplexed (WDM) network, the fiber bandwidth is divided into multiple frequency bands often called wavelengths. Using reconfigurable optical switches at the network nodes, some wavelengths can be selected at each node for termination and electronic processing, and others selected for optical bypass. In an *all-optical* network, each traffic session optically bypasses electronic processing at each node on its path other than the source node and the destination node. One important benefit of this architecture is a significant cost saving from the use of fewer and/or smaller electronic switches.

Without optical wavelength conversion, routing of traffic sessions is subjected to the *wavelength continuity constraint*, which dictates that the lightpath corresponding to a given session must travel on the same wavelength on all links from the source node to the destination node. Using wavelength converters potentially allows the network to support a larger set of traffic. However, such converters are likely to be expensive. Hence, several researchers assume no wavelength conversion in the problem of routing and wavelength assignment (RWA). Similarly, we shall assume no wavelength conversion.

The RWA problem is an important problem in resource management for all-optical networks, and has received a lot of attention. We can categorize existing results in literature into two groups based on whether static or dynamic provision of routes and wavelengths is performed. We shall focus on dynamic RWA.

To model dynamic traffic, session arrivals can be assumed to form stochastic processes [1, 2]. In addition, session lifetimes are probabilistic. The goal is usually to develop an on-line RWA algorithm which minimizes the average blocking probability for a new session request given a fixed number of wavelengths in the network. We refer to this type of problem formulation as the *blocking* formulation. Due to the complexity in

*This work was supported by the National Science Foundation (NSF) under Grant ANI-0073730.

computing blocking probabilities, some approximations are made to simplify the analysis. For example, session arrivals on different links are assumed to be independent [1, 3], or correlated among adjacent links in the same fashion throughout the network [2]. Based on such approximations, several dynamic RWA heuristics have been developed [4, 5].

Alternatively, a different problem formulation, referred to as the *nonblocking* formulation, assumes prior knowledge of the set of all the traffic matrices to be supported [6, 7, 8, 9]. In [7], the set of traffic matrices is characterized by the maximum link load in the network. In [6, 8, 9], the set of traffic matrices is characterized by the numbers of tunable transmitters and tunable receivers at each end node, i.e. a node which sources and/or sinks traffic sessions. A new session is said to be *admissible* if its arrival results in a traffic matrix which is still in the set of supportable traffic. The goal is to develop an on-line RWA algorithm which does not block any admissible session and uses the minimum number of wavelengths.

If we allow some existing lightpaths to be rearranged in order to support a new session, the corresponding RWA algorithm is said to be *rearrangeably nonblocking.* If we allow no rearrangement of any existing lightpath in order to support a new session, the corresponding RWA algorithm is said to be *wide-sense nonblocking.* Note that if an RWA algorithm is wide-sense nonblocking, it is also rearrangeably nonblocking.

We shall adopt a nonblocking formulation. As in [6, 8, 9], the supportable traffic set is defined by the number of tunable transmitters and tunable receivers at each end node. We model the traffic as a session-by-session arrival and departure process in which sessions arrive and depart one at a time, and each session utilizes a full wavelength. Our goal is to design an on-line RWA algorithm which is rearrangeably nonblocking, uses the minimum number of wavelengths, and requires few rearrangements of existing lightpaths in order to support each new session request.

In this paper, we present an on-line RWA algorithm for tree networks. Since there is no routing problem in a tree network, our RWA algorithm only has to perform wavelength assignment (WA) and will be referred to as a WA algorithm. Since a tree topology can be embedded in any connected topology, we hope that our analytical approach can later be used in the extension of this work to other types of network topologies.

In section 2, we define the set of **k**-port traffic based on the number of tunable transmitters and tunable receivers at each end node, and formulate the WA problem for **k**-port traffic in a tree network. In section 3, we describe our on-line WA algorithm and prove its correctness. Finally, we summarize the results and point out future research directions in section 4.

2 Problem Formulation

Consider a WDM all-optical tree network with no wavelength conversion. Adjacent nodes are connected by two fibers, one in each direction. All fibers contain the same number of wavelengths. Assume there are N end nodes, which are the leaf nodes of the tree. (Later on, we shall discuss the cases with non-leaf end nodes.) Assume that each traffic session has a rate of one wavelength. At a given time, only one session can use a specific wavelength in a fiber, but multiple sessions on the same wavelength can use the same node. Leaf node i, $1 \leq i \leq N$, is equipped with k_i fully tunable transmitters and k_i fully tunable receivers. Consequently, at any time, node i can transmit at most k_i wavelengths and receive at most k_i wavelengths. Such a traffic matrix is said to belong to a set of **k**-*port traffic*, where $\mathbf{k} = [k_1, k_2, ..., k_N]$. We make one assumption on **k**-port traffic.

Assumption 1 *Let* $k_{max} = \max_{1 \leq i \leq N} k_i$*. Assume that* $k_{max} \leq \left(\sum_{1 \leq i \leq N} k_i \right) /2$.

Assumption 1 is reasonable since the node with k_{max} transmitters (receivers) can transmit (receive) at most $(\sum_{1 \le i \le N} k_i) - k_{max}$ wavelengths to (from) all the other nodes. Therefore, k_{max} need be no greater than $(\sum_{1 \le i \le N} k_i) - k_{max}$.

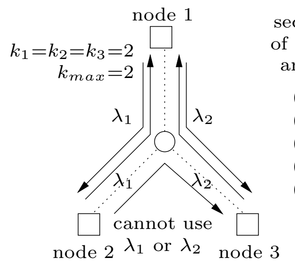
We model dynamic traffic as a session-by-session arrival and departure process in which sessions arrive and depart one at a time. In other words, a transition from one traffic matrix to another results from either a session arrival or a session departure. A new session request is admissible if the resultant traffic matrix is still in the set of **k**-port traffic. For convenience, throughout the paper, a new session is assumed to be admissible.

We want to design an on-line WA algorithm which supports **k**-port traffic in a rearrangeably nonblocking fashion, uses the minimum number of wavelengths, and requires few rearrangements of existing lightpaths in order to support each new session request. Our algorithm will be centralized in nature. We assume that traffic does not change too frequently and the WA algorithm always has correct knowledge of the current WA in the network. In addition, we assume there is sufficient time for lightpath rearrangements between successive transitions of the traffic matrix.

3 On-Line WA Algorithm

3.1 Star Networks

We first present an on-line WA algorithm for star networks. This algorithm is later extended for tree networks. Fig. 1 shows an example of a star network with 3 leaf nodes connected through a central hub.



sequence of session arrivals	corresponding sequence of WA steps
(1,2)	(1,2) on λ_1
(2,1)	(2,1) on λ_1
(1,3)	(1,3) on λ_2
(3,1)	(3,1) on λ_2
(2,3)	(2,3) not on λ_1 or λ_2

(i, j) denotes a session from node i to node j.

Figure 1: An example in which a greedy approach requires more than k_{max} wavelengths.

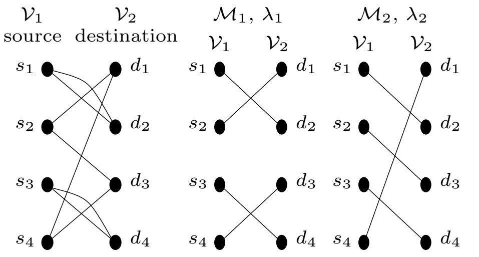
Let $L_{\mathbf{k}}$ and $W_{\mathbf{k}}$ denote the minimum number of wavelengths which, if provided in each fiber, can support **k**-port traffic with full wavelength conversion at all nodes and without wavelength conversion respectively. It is clear that $L_{\mathbf{k}} \le W_{\mathbf{k}}$. Notice that $L_{\mathbf{k}}$ and $W_{\mathbf{k}}$ are the number of wavelengths required to support *any* traffic matrix in the **k**-port set. Thus, for a specific traffic matrix, we may need fewer wavelengths than in the worst-case. To derive $L_{\mathbf{k}}$, consider the fiber from the node with traffic parameter k_{max} to the hub node. This fiber must support up to k_{max} wavelengths, which is the maximum link load. It follows that $L_{\mathbf{k}} = k_{max}$.

We shall construct an on-line WA algorithm using k_{max} wavelengths. This algorithm implies that $W_{\mathbf{k}} = L_{\mathbf{k}} = k_{max}$. Fig. 1 illustrates an example scenario in which an on-line greedy WA algorithm fails to support an instance of **k**-port traffic using k_{max} wavelengths. In this example, $N = 3$, $\mathbf{k} = [2, 2, 2]$, and the traffic matrix to be supported is uniform all-to-all traffic, i.e. each node sends one session to each of the other nodes. In Fig. 1, the same wavelength is assigned to the oppositely directed sessions between the same pair of nodes, e.g. sessions (1,2) and (2,1) on wavelength λ_1. After assigning wavelengths λ_1 and λ_2 to sessions (1,2), (2,1), (1,3), and (3,1), neither λ_1 nor λ_2 can be assigned to (2,3). It follows that more than $k_{max} = 2$ wavelengths are used. This example tells us that the WA algorithm design is not trivial. In addition, to use the minimum number of wavelengths, we may need to support the oppositely directed sessions between the same pair of nodes on different wavelengths.

Our algorithm is based on bipartite matchings. For a given traffic matrix, we construct the *traffic bipartite graph*, denoted by $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$, as follows. We consider each leaf node as a distinct source node and destination node. The set of nodes $\mathcal{V}_1$ contains the N source nodes. The set of nodes $\mathcal{V}_2$ contains the N destination nodes. In the set of edges $\mathcal{E}$, an edge between node i in $\mathcal{V}_1$ and node j in $\mathcal{V}_2$ exists for each traffic session from source i to destination j. Fig. 2a shows an example of the traffic bipartite graph and its traffic matrix. Note that there may be multiple edges between the same pair of nodes. For example, since there are two sessions from source 1 to destination 2, there are two parallel edges between s_1 in $\mathcal{V}_1$ and d_2 in $\mathcal{V}_2$ in Fig. 2a.

destination

$$\text{source}\quad \begin{array}{c|cccc} & d_1 & d_2 & d_3 & d_4 \\ \hline s_1 & 0 & 2 & 0 & 0 \\ s_2 & 1 & 0 & 1 & 0 \\ s_3 & 0 & 0 & 0 & 2 \\ s_4 & 1 & 0 & 1 & 0 \end{array}$$



(a) traffic matrix and its traffic bipartite graph   (b) bipartite matchings $\mathcal{M}_1$ and $\mathcal{M}_2$ assigned to wavelengths λ_1 and λ_2

Figure 2: Traffic bipartite graph and its matchings.

A matching in a bipartite graph, or in short a bipartite matching, is a subset $\mathcal{M}$ of $\mathcal{E}$ such that no two edges in $\mathcal{M}$ are adjacent. A matching $\mathcal{M}$ is said to *saturate* the set $\mathcal{V}_1$ if, for every node in $\mathcal{V}_1$, there is an edge in $\mathcal{M}$ incident on that node. A matching $\mathcal{M}$ which saturates the set $\mathcal{V}_1$ is called a *perfect matching.* In Fig. 2b, the matchings $\mathcal{M}_1$ and $\mathcal{M}_2$ are two different perfect matchings in $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$.

Observe that the sessions in a bipartite matching can be supported on a single wavelength without wavelength collision. To see this, note that, in a matching, at most one edge is incident on each source (destination) node. Thus, in each fiber to (from) the hub node, every wavelength is used at most once. Our algorithm will assign a single matching to a single wavelength. We shall refer to the matching assigned to wavelength λ_1 simply as the bipartite matching of λ_1. Fig. 2b shows example bipartite matchings of specific wavelengths. We next state a known lemma related to bipartite graphs [10].

Lemma 1 *In a bipartite graph $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$ with maximum node degree m, we can color the edges in $\mathcal{E}$ so that no two adjacent edges have the same color using m colors.*

Consider coloring the edges in a bipartite graph $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$ as suggested by lemma 1. Since no two adjacent edges have the same color, the edges with the same color form a bipartite matching. Thus, we can restate lemma 1 as follows.

Lemma 2 *In a bipartite graph $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$ with maximum node degree m, the set $\mathcal{E}$ can be partitioned into m disjoint bipartite matchings.*

Lemma 2 can be used to argue that k_{max} wavelengths are sufficient to support any traffic matrix in the **k**-port set. Given a traffic matrix, we can write down the corresponding traffic bipartite graph in which each node has degree at most k_{max}. By lemma 2, the set of edges can be partitioned into k_{max} disjoint bipartite matchings. The sessions in each matching can be supported on a single wavelength. Thus, k_{max} wavelengths are sufficient to support any **k**-port traffic matrix.

The main idea of our on-line WA algorithm involves keeping k_{max} disjoint bipartite matchings of k_{max} wavelengths such that each traffic session corresponds to an edge in one bipartite matching. When a session departs, we simply remove its corresponding lightpath from the network. When a new session, say (i, j), arrives, we find one wavelength that is not used by source i, and one wavelength that is not used by destination j. If the two wavelengths are the same, we can support the new session without any lightpath rearrangement. Otherwise, we rearrange some existing lightpaths on the two wavelengths to support the new session. The following lemma makes the above discussion concrete and states an upper bound on the number of lightpath rearrangements.

Lemma 3 *In a bipartite graph $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$ with $|\mathcal{V}_1| = |\mathcal{V}_2| = V$, given a new edge (s_i, d_j), $s_i \in \mathcal{V}_1$, $d_j \in \mathcal{V}_2$, a matching $\mathcal{M}_1$ of wavelength λ_1 which is not incident on s_i, and a matching $\mathcal{M}_2$ of wavelength λ_2 which is not incident on d_j, there exist two disjoint matchings which cover all the edges in $\mathcal{M}_1$ and $\mathcal{M}_2$ as well as the new edge (s_i, d_j).*

In addition, these two disjoint bipartite matchings can be assigned to λ_1 and λ_2 so that the number of lightpath rearrangements is at most $V - 1$.

Proof: Consider the bipartite graph $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E}')$, where $\mathcal{E}' = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \{(s_i, d_j)\}$. Observe that each node has degree at most 2. From lemma 2, $\mathcal{E}'$ can be partitioned into two disjoint matchings $\mathcal{M}'_1$ and $\mathcal{M}'_2$.

Without loss of generality, assume that $(s_i, d_j) \in \mathcal{M}'_1$. Let set $\mathcal{P}$ contain the edges in $\mathcal{M}_1$ assigned to $\mathcal{M}'_2$ and the edges in $\mathcal{M}_2$ assigned to $\mathcal{M}'_1$. Let set $\mathcal{Q}$ contain the edges in $\mathcal{M}_1$ assigned to $\mathcal{M}'_1$ and the edges in $\mathcal{M}_2$ assigned to $\mathcal{M}'_2$. Notice that $\mathcal{P}$ and $\mathcal{Q}$ are disjoint and $\mathcal{P} \cup \mathcal{Q} = \mathcal{M}_1 \cup \mathcal{M}_2$. Since there are at most $2V - 2$ edges in $\mathcal{M}_1 \cup \mathcal{M}_2$, it follows that $|\mathcal{P}| + |\mathcal{Q}| \leq 2V - 2$. If $|\mathcal{P}| \leq V - 1$, assigning $\mathcal{M}'_1$ to λ_1 and $\mathcal{M}'_2$ to λ_2 yields the desired result. Otherwise, it is true that $|\mathcal{Q}| \leq V - 1$. In this case, assigning $\mathcal{M}'_1$ to λ_2 and $\mathcal{M}'_2$ to λ_1 yields the desired result. $\square$

In [11], we provide an efficient algorithm to find two disjoint matchings in a bipartite graph with maximum node degree 2. The algorithm has the running time $O(V)$.

The following is our on-line WA algorithm for a star network with $\mathbf{k}$-port traffic which uses k_{max} wavelengths in each fiber, is rearrangeably nonblocking, and requires at most $N - 1$ lightpath rearrangements per new session request.

Star WA algorithm: (Use k_{max} wavelengths.)

Session termination: Simply remove its lightpath from the network.

Session arrival: When a new session arrives, proceed as follows. (Assume that the new session is (i, j).)

Step 1: If there is a wavelength, denoted by λ_0, which is used by neither source i nor destination j, then assign the new session to λ_0. In this case, no lightpath rearrangement is made. Otherwise, proceed to step 2.

Step 2: Find a wavelength, denoted by λ_1, which is not used by source i, i.e. its bipartite matching $\mathcal{M}_1$ is not incident on s_i, and another wavelength, denoted by λ_2, which is not used by destination j, i.e. its bipartite matching $\mathcal{M}_2$ is not incident on d_j. (Since the new session is admissible, there are at most $k_{max} - 1$ sessions from source i. Since there are k_{max} available wavelengths, it follows that λ_1 exists. By the same argument, λ_2 always exists.)

Modify the WA of only the sessions on λ_1 and λ_2. Contruct the traffic bipartite graph $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E}')$, where $\mathcal{E}' = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \{(s_i, d_j)\}$. Partition $\mathcal{E}'$ into two disjoint matchings (c.f. lemma 2). Since $|\mathcal{V}_1| = |\mathcal{V}_2| = N$, lemma 3 tells us that the two matchings can be assigned to λ_1 and λ_2 such that at most $N - 1$ existing lightpaths need to be rearranged.

The construction of the star WA algorithm implies the following theorem.

Theorem 1 *For the star network with N nodes and $\mathbf{k}$-port traffic,*

$$W_{\mathbf{k}} = L_{\mathbf{k}} = k_{max} = \max_{1 \le i \le N} k_i.$$

In addition, there exists, by construction, an on-line WA algorithm which uses k_{max} wavelengths in each fiber and requires at most $N-1$ lightpath rearrangements per new session request.

3.2 Arbitrary Tree Networks

In this section, we extend the star WA algorithm to create an on-line WA algorithm for tree networks. In a given tree network, assume there are $N > 2$ end nodes which are the leaf nodes of the tree. We shall ignore all the non-leaf nodes with degree 2 since their removal does not change the WA problem. We describe a tree by a set of nodes $\mathcal{N}$ and a set of bidirectional links $\mathcal{T}$.

We first determine $L_{\mathbf{k}}$, the minimum number of wavelengths which, if provided in each fiber, can support $\mathbf{k}$-port traffic given full wavelength conversion at all nodes. Each link e in the tree corresponds to a cut which separates N leaf nodes into two sets, denoted by $\mathcal{N}_{e,1}$ and $\mathcal{N}_{e,2}$. The maximum possible traffic, in wavelength units, in a fiber across this link is equal to $\min(\sum_{i\in\mathcal{N}_{e,1}} k_i, \sum_{i\in\mathcal{N}_{e,2}} k_i)$. The overall maximum possible traffic across any link, denoted by w^*, is the value of $L_{\mathbf{k}}$ given below.

$$L_{\mathbf{k}} = w^* = \max_{e\in\mathcal{T}} \min\left(\sum_{i\in\mathcal{N}_{e,1}} k_i, \sum_{i\in\mathcal{N}_{e,2}} k_i\right) \tag{1}$$

Let $W_{\mathbf{k}}$ be the minimum number of wavelengths which, if provided in each fiber, can support $\mathbf{k}$-port traffic with no wavelength conversion. We shall construct an on-line WA algorithm using w^* wavelengths. This algorithm implies that $W_{\mathbf{k}} = L_{\mathbf{k}} = w^*$. We shall refer to w^* as the *worst-case* number of wavelengths since w^* wavelengths are necessary and sufficient to support *any* traffic matrix in the $\mathbf{k}$-port traffic set. Since a star network is also a tree network, Fig. 1 shows that the WA algorithm design is not trivial.

Let e^* denote the link associated with w^*. Note that there may be multiple choices for e^*. We shall refer to e^* as the *bottleneck link* since it is the link with the maximum load under the worst-case traffic.

Link e^* separates the leaf nodes into two sets $\mathcal{N}_{e^*,1}$ and $\mathcal{N}_{e^*,2}$. Without loss of generality, choose $\mathcal{N}_{e^*,1}$ such that the sum of k_i's in this set is w^*. We assume for now that $\mathcal{N}_{e^*,2}$ contains multiple leaf nodes, as illustrated in Fig. 3. Define the *bottleneck node* v^* to be the end point of e^* opposite to $\mathcal{N}_{e^*,1}$, i.e. the subtree connected to v^* by e^* has the sum of k_i's equal to w^*, as illustrated in Fig. 3.

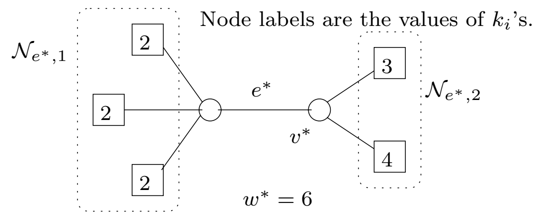


Figure 3: Definition of the bottleneck node v^*.

We shall refer to each subtree connected to v^* as a *top-level subtree.* Note that a top-level subtree can be a single node. Let d^* be the degree of v^*. Since v^* is a non-leaf node, $d^* \ge 3$. It follows that there are $d^* \ge 3$ top-level subtrees, as illustrated in Fig. 4a.

If the set $\mathcal{N}_{e^*,2}$ contains a single node, we have the scenario illustrated in Fig. 4b. In this case, assumption 1 implies that the value of k_i for the leaf node in $\mathcal{N}_{e^*,2}$ is equal to w^*. We argue that, with $N > 2$ leaf nodes, this scenario can be transformed to the scenario in Fig. 4a by exchanging the roles of $\mathcal{N}_{e^*,1}$ and $\mathcal{N}_{e^*,2}$. Therefore, we shall shall consider only the scenarios in which v^* exists and $d^* \geq 3$, as illustrated in Fig. 4a. Note that the location of the bottleneck node v^* depends on the specific tree network and the traffic vector $\mathbf{k}$, but not on the current traffic matrix being supported. The following lemma provides useful properties regarding w^*.

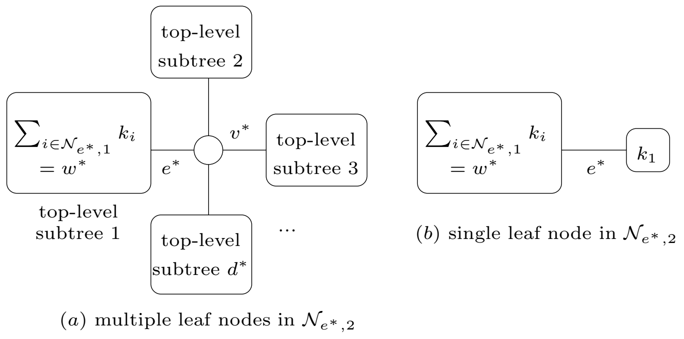

top-level subtree 2
$\sum_{i\in\mathcal{N}_{e^*,1}} k_i = w^*$
v^*
top-level subtree 3
e^*
top-level subtree 1
top-level subtree d^*
...
(a) multiple leaf nodes in $\mathcal{N}_{e^*,2}$
$\sum_{i\in\mathcal{N}_{e^*,1}} k_i = w^*$
e^*
k_1
(b) single leaf node in $\mathcal{N}_{e^*,2}$

Figure 4: The bottleneck node v^* and the top-level subtrees.

Lemma 4 *Let K_j denote the sum of k_i's in top-level subtree j. Let $K = \sum_{1\leq j\leq d^*} K_j$. Then, (1) For all $1 \leq j \leq d^*$, $K_j \leq w^*$, and (2) $K/d^* \leq w^* \leq K/2$.*

Proof: Number the d^* top-level subtrees from 1 to d^* such that top-level subtree 1 is connected to v^* by e^*. By the definition of v^*, we know that $K_1 = w^*$. For $2 \leq j \leq d^*$, consider the link e_j which isolates top-level subtree j from v^*. Let $\mathcal{N}_{e_j,1}$ contain the leaf nodes in top-level subtree j, and $\mathcal{N}_{e_j,2}$ contain all the other leaf nodes. Consequently, $\sum_{i\in\mathcal{N}_{e_j,1}} k_i = K_j$. In addition, $\sum_{i\in\mathcal{N}_{e_j,2}} k_i = K - K_j > K_1 = w^*$ since there are at least three top-level subtrees. From the definition of w^* in (1), we must have that $K_j \leq w^*$, or else e_j instead of e^* would be the bottleneck link. It follows that $K_j \leq w^*$ for $2 \leq j \leq d^*$. Thus, $K_j \leq w^*$ for all $1 \leq j \leq d^*$.

From the definition of w^*, it is clear that $w^* \leq K/2$. The lower bound follows from statement 1 of the lemma, i.e. $K = \sum_{1\leq j\leq d^*} K_j \leq d^* w^*$. □

As in the star WA algorithm, the algorithm in this section is based on bipartite matchings. The main difference has to do with what a node in a bipartite graph represents. In the star WA algorithm, a node represents a single source or a single destination. In this section, a node represents a set of sources or a set of destinations in a top-level subtree.

For a given traffic matrix, we construct the *top-level subtree bipartite graph*, denoted by $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$, as follows. We consider each leaf node as one distinct source and one distinct destination. Number the d^* top-level subtrees from 1 to d^*. The set $\mathcal{V}_1$ contains d^* abstract nodes, denoted by $\mathcal{S}_1$, $\mathcal{S}_2$, ..., $\mathcal{S}_{d^*}$. Node $\mathcal{S}_i$, $1 \leq i \leq d^*$, represents the set of sources contained in top-level subtree i. Similarly, the set $\mathcal{V}_2$ contains d^* abstract nodes, denoted by $\mathcal{D}_1$, $\mathcal{D}_2$, ..., $\mathcal{D}_{d^*}$. Node $\mathcal{D}_j$, $1 \leq j \leq d^*$, represents the set of destinations contained in top-level subtree j. In the set of edges $\mathcal{E}$, an edge from node $\mathcal{S}_i$ in $\mathcal{V}_1$ to node $\mathcal{S}_j$ in $\mathcal{V}_2$ exists for each traffic session from a source in top-level subtree i to a destination in top-level subtree j. Fig. 5 shows an example of the top-level subtree bipartite graph and its traffic matrix. Note that there may be multiple edges between the same pair of nodes. For example, since there are two sessions from top-level subtree 3 to top-level subtree 4, there are two parallel edges between the set of sources $\mathcal{S}_3$ and the set of destinations $\mathcal{D}_4$ in Fig. 5d.

Define a *local* session to be a traffic session whose source and destination are in the same top-level subtree. Accordingly, a *non-local* session has its source and its destination in different top-level subtrees. A non-local session has to travel through the bottleneck node v^*, whereas a local session does not have to travel all the way to v^* and back to its destination, i.e. each session never uses the same link twice in the opposite directions. A non-local session corresponds to an edge from some node $\mathcal{S}_i$ in $\mathcal{V}_1$ and some node $\mathcal{D}_j$ in $\mathcal{V}_2$, where $i \neq j$. On the other hand, a local session corresponds to an edge between some node $\mathcal{S}_i$ in $\mathcal{V}_1$ and node $\mathcal{D}_i$ in $\mathcal{V}_2$. For example, the top-level subtree bipartite graph in figure 5d contains seven non-local sessions and one local session. The local session is from a source in top-level subtree 2 to a destination in the same top-level subtree.

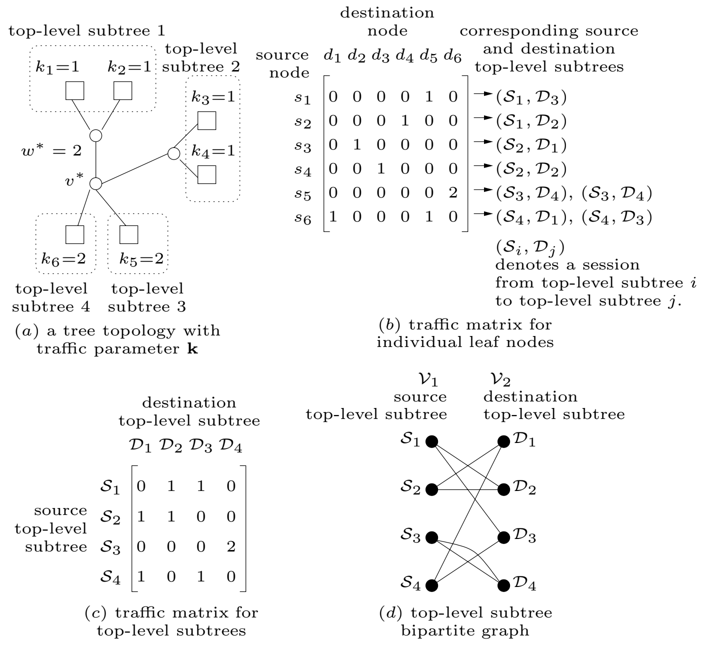

top-level subtree 1
k_1=1
k_2=1
top-level subtree 2
k_3=1
k_4=1
w^* = 2
v^*
k_6=2
k_5=2
top-level subtree 4
top-level subtree 3
(a) a tree topology with traffic parameter **k**
destination node
source node
d_1 d_2 d_3 d_4 d_5 d_6
s_1 0 0 0 0 1 0 → $(\mathcal{S}_1, \mathcal{D}_3)$
s_2 0 0 0 1 0 0 → $(\mathcal{S}_1, \mathcal{D}_2)$
s_3 0 1 0 0 0 0 → $(\mathcal{S}_2, \mathcal{D}_1)$
s_4 0 0 1 0 0 0 → $(\mathcal{S}_2, \mathcal{D}_2)$
s_5 0 0 0 0 0 2 → $(\mathcal{S}_3, \mathcal{D}_4)$, $(\mathcal{S}_3, \mathcal{D}_4)$
s_6 1 0 0 0 1 0 → $(\mathcal{S}_4, \mathcal{D}_1)$, $(\mathcal{S}_4, \mathcal{D}_3)$
corresponding source and destination top-level subtrees
$(\mathcal{S}_i, \mathcal{D}_j)$ denotes a session from top-level subtree i to top-level subtree j.
(b) traffic matrix for individual leaf nodes
destination top-level subtree
$\mathcal{D}_1$ $\mathcal{D}_2$ $\mathcal{D}_3$ $\mathcal{D}_4$
source top-level subtree
$\mathcal{S}_1$ 0 1 1 0
$\mathcal{S}_2$ 1 1 0 0
$\mathcal{S}_3$ 0 0 0 2
$\mathcal{S}_4$ 1 0 1 0
(c) traffic matrix for top-level subtrees
$\mathcal{V}_1$ source top-level subtree
$\mathcal{V}_2$ destination top-level subtree
$\mathcal{S}_1$ $\mathcal{S}_2$ $\mathcal{S}_3$ $\mathcal{S}_4$
$\mathcal{D}_1$ $\mathcal{D}_2$ $\mathcal{D}_3$ $\mathcal{D}_4$
(d) top-level subtree bipartite graph

Figure 5: Top-level subtree bipartite graph.

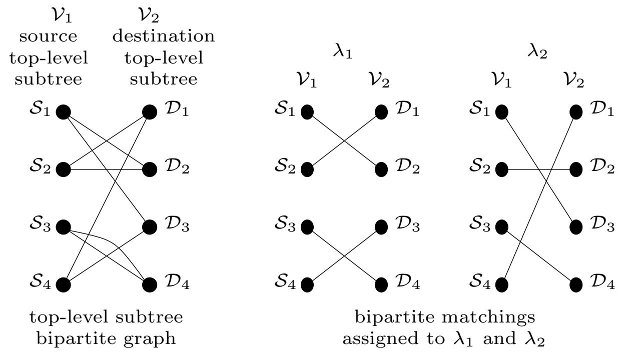

$\mathcal{V}_1$ source top-level subtree
$\mathcal{V}_2$ destination top-level subtree
$\mathcal{S}_1$ $\mathcal{S}_2$ $\mathcal{S}_3$ $\mathcal{S}_4$
$\mathcal{D}_1$ $\mathcal{D}_2$ $\mathcal{D}_3$ $\mathcal{D}_4$
top-level subtree bipartite graph
λ_1
λ_2
$\mathcal{V}_1$ $\mathcal{V}_2$
bipartite matchings assigned to λ_1 and λ_2

Figure 6: Bipartite matchings of specific wavelengths.

Observe that the sessions belonging to the same matching in the top-level subtree bipartite graph can be supported on a single wavelength without wavelength collision. To see this, note that any two sessions in a matching are transmitted from different top-level subtrees and to different top-level subtrees. If these two sessions travel in the same top-level subtree, one session must be transmitted from that subtree while the other session must be received in that subtree. These two sessions traverse links in the same top-level subtree in the opposite directions and do not collide.

Our algorithm will assign a single bipartite matching to a single wavelength. We shall refer to the matching assigned to wavelength λ_1 as the bipartite matching of λ_1. Fig. 6 shows example bipartite matchings of specific wavelengths.

We now argue that w^* wavelengths are sufficient to support any traffic matrix in the **k**-port set. From statement 1 of lemma 4, each top-level subtree can transmit at most w^*

wavelengths and receive at most w^* wavelengths. Thus, for a given a traffic matrix, each node in the corresponding top-level subtree bipartite graph has degree at most w^*. By lemma 2, the set of edges can be partitioned into w^* disjoint bipartite matchings. The sessions in each matching can be supported on a single wavelength. Thus, w^* wavelengths are sufficient. Notice that, by finding w^* disjoint matchings, we provide the WA for both local and non-local sessions simultaneously.

The main idea of our on-line WA algorithm involves keeping w^* disjoint bipartite matchings of w^* wavelengths such that each traffic session corresponds to an edge in one bipartite matching. When a session departs, we simply remove its corresponding lightpath from the network. When a new (local or non-local) session arrives, we update the WA by finding up to two wavelengths whose bipartite matchings can be reassigned to include the new session.

The following is our on-line WA algorithm for a tree network with **k**-port traffic which uses w^* wavelengths in each fiber, is rearrangeably nonblocking, and requires at most $d^* - 1$ lightpath rearrangements per new session request.

Tree WA algorithm: (Use w^* wavelengths.)

Session termination: Simply remove its lightpath from the network.

Session arrival: When a new session arrives, proceed as follows. Assume that the new session is from a source in top-level subtree i to a destination in top-level subtree j. When $i = j$, the new session is local. Otherwise, it is non-local. In either case, follow the same procedures below.

Step 1: If there is a wavelength, denoted by λ_0, which is used by neither a source in top-level subtree i nor a destination in top-level subtree j, then assign the new session to λ_0. In this case, no lightpath rearrangement is made. Otherwise, proceed to step 2.

Step 2: Find a wavelength, denoted by λ_1, which is not used by any source in top-level subtree i, i.e. its bipartite matching $\mathcal{M}_1$ is not incident on $\mathcal{S}_i$, and another wavelength, denoted by λ_2, which is not used by any destination in top-level subtree j, i.e. its bipartite matching $\mathcal{M}_2$ is not incident on$\mathcal{D}_j$. (Since the new session is admissible, there are at most $w^* - 1$ sessions from top-level subtree i. Since there are w^* available wavelengths, it follows that λ_1 exists. By the same argument, λ_2 always exists.)

Modify the WA of only the sessions on λ_1 and λ_2. Contruct the top-level subtree bipartite graph $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E}')$, where $\mathcal{E}' = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \{(\mathcal{S}_i, \mathcal{D}_j)\}$. Partition $\mathcal{E}'$ into two disjoint matchings (c.f. lemma 3). Since $|\mathcal{V}_1| = |\mathcal{V}_2| = d^*$, lemma 3 tell us that the two matchings can be assigned to λ_1 and λ_2 such that at most $d^* - 1$ existing lightpaths need to be rearranged.

The construction of the tree WA algorithm implies the following theorem.

Theorem 2 *For an arbitrary tree network with* **k***-port traffic among N leaf nodes and the bottleneck node v^* with degree d^*,*

$$W_{\mathbf{k}} = L_{\mathbf{k}} = w^* = \max_{e \in \mathcal{T}} \min \left(\sum_{i \in \mathcal{N}_{e,1}} k_i, \sum_{i \in \mathcal{N}_{e,2}} k_i \right).$$

In addition, there exists, by construction, an on-line WA algorithm which uses w^* wavelengths in each fiber and requires at most $d^* - 1$ lightpath rearrangements per new session.

Theorem 2 tells us that wavelength conversion cannot decrease the wavelength requirement for **k**-port traffic in an arbitrary tree topology. In addition, if we scale the traffic vector **k** by an integer factor, then the location of the bottleneck node v^* remains fixed, and the upper bound on the number of lightpath rearrangements per new session request does not increase. Finally, from statement 2 of lemma 4, among the tree topologies with N leaf nodes, the minimum value of the worst-case number of wavelengths w^* is at least $(\sum_{1\leq i\leq N} k_i)/d^*$. The tree topologies with w^* close to this lower bound are the ones in which each top-level subtree has the sum of k_i's approximately equal to $(\sum_{1\leq i\leq N} k_i)/d^*$. Roughly speaking, it is desirable to have all the top-level subtrees support an equal amount of traffic.

4 Conclusion

We developed an on-line WA algorithm for dynamic $\mathbf{k}$-port traffic in a WDM all-optical tree network. The algorithm is rearrangeably non-blocking, uses the minimum number of wavelengths, and requires at most $d^* - 1$ lightpath rearrangements per new session request, where d^* is the degree of the most heavily used node in the worst-case traffic scenario. Most of the complexity in our on-line WA algorithm involves partitioning the edges in a bipartite graph with maximum node degree 2 into two disjoint matchings.

We observed that the number of lightpath rearrangements per new session request does not increase as the amount of traffic $\mathbf{k}$ in the network increases by an integer scaling factor. In addition, we found that, for an arbitrary tree topology, the minimum numbers of wavelengths required to support $\mathbf{k}$-port traffic with full wavelength conversion at all nodes and without wavelength conversion are the same. This implies that the use of wavelength converters will not decrease the required number of wavelengths.

Our future goal is to develop an on-line RWA algorithm for other types of network topologies. Since a tree network can be embedded in any connected network, we hope that our analytical approach in this paper can be extended to create an on-line RWA algorithm for other types of network topologies.

References

[1] A. Birman, "Computing approximate blocking probabilities for a class of all-optical networks," *IEEE JSAC*, vol. 14, no. 5, pp. 852-857, June 1996.

[2] S. Subramaniam et al., "All-optical networks with sparse wavelength conversion," *IEEE/ACM Trans. on Networking*, vol. 4, no. 4, pp. 544-557, August 1996.

[3] R.A. Barry and P.A. Humblet, "Models of blocking probability in all-optical networks with and without wavelength changers," *IEEE JSAC*, vol. 14, no. 5, pp. 858-867, June 1996.

[4] L. Li and A.K. Somani, "Dynamic wavelength routing using congestion and neighborhood information," *IEEE/ACM Trans. on Networking*, vol. 7, no. 5, pp. 779-786, October 1999.

[5] Y. Zhu, G.N. Rouskas, and H.G. Perros, "A path decomposition algorithm for computing blocking probabilities in wavelength routing networks," *IEEE/ACM Trans. on Networking*, vol. 8, no. 6, pp. 747-762, December 2000.

[6] R.K. Pankaj, *Architectures for Linear Lightwave Networks*, MIT Ph.D. Thesis, 1992.

[7] O. Gerstel, G. Sasaki, S. Kutten, and R. Ramaswami, "Worst-case analysis of dynamic wavelength allocation in optical networks," *IEEE/ACM Trans. on Networking*, vol. 7, no. 6, pp. 833-845, December 1999.

[8] A. Narula-Tam, P.J. Lin, and E.H. Modiano, "Efficient routing and wavelength assignment for reconfigurable WDM networks," *IEEE JSAC*, vol. 20, no. 1, pp. 75-88, January 2002.

[9] P. Saengudomlert, E.H. Modiano, and R.G. Gallager, "On-line routing and wavelength assignment for dynamic traffic in WDM ring and torus networks," *IEEE Infocom*, April 2003.

[10] C. Berge, *Graphs.* North-Holland, 1985.

[11] P. Saengudomlert, *Architectural Study of High-Speed Networks with Optical Bypassing*, MIT Ph.D. Thesis, September 2002.