An Improved Algorithm for Optimal Lightpath Establishment on a Tree Topology

Guoliang Xue, *Senior Member, IEEE*, Weiyi Zhang, Jian Tang, and Krishnaiyan Thulasiraman, *Fellow, IEEE*

***Abstract*— Routing and wavelength assignment (RWA) aims to assign the limited number of wavelengths in a wavelength-division multiplexed (WDM) optical network so as to achieve greater capacity. In a recent paper [6], Datta *et al.* studied the problem of establishing a set of disjoint lightpaths on a tree topology using a single wavelength to maximize the total traffic supported by the chosen set of lightpaths. They discussed applications of this problem to RWA and presented a dynamic programming algorithm which optimally solves this problem in $O(n^4+n\mathcal{D}^3)$ time, where n is the number of nodes in the network and $\mathcal{D}$ is the maximum node degree. In this paper, we present an improved algorithm with a time complexity of $O(n^2+n\mathcal{D}^2)$.**

***Index Terms*— WDM networks, wavelength assignment, maximum matching, graph algorithms.**

I. Introduction

ALL-OPTICAL networks transmit information on a very high speed, broadband optical path [2], [10], [17]. One promising all-optical wide-area network architecture is the wavelength-division multiplexed (WDM) network with circuit-switching and wavelength routing [15], [16].

In WDM networks, the fiber bandwidth is partitioned into many wavelength channels, each of which can be utilized to carry independent data connections. Reference [2] introduced the concept of a lightpath, which is an all-optical connection that is wavelength-routed from source node to destination node; i.e., there is no electronic processing at intermediate nodes of the path. In the absence of all-optical wavelength conversion, the same wavelength must be used to carry the traffic on each link of the lightpath. In one-hop (also called single-hop) routing schemes, connections between nodes are established using a single lightpath (one-hop refers to a single virtual hop, not a single physical hop). Due to the wavelength continuity constraint, one-hop connections are often difficult to establish. Alternatively, multi-hop connections can be established, where the end-to-end connection is comprised of multiple lightpaths, and a different wavelength may be used for each of the lightpaths. Multi-hop connections, however, require electronic processing at some of the intermediate nodes, which potentially adds to the cost of the network. The focus of this paper is an efficient algorithm for one-hop routing, without wavelength conversion, on a tree topology. This algorithm can be used as a foundation for more general routing and wavelength assignment, as was shown in [6], where the goal is to carry most of the traffic using one-hop connections while being efficient in the number of wavelengths utilized.

Given communication or lightpath requests, finding routes for the lightpath requests and assigning wavelengths to each of the links on the lightpaths satisfying certain performance criteria is known as the routing and wavelength assignment (RWA) problem. The RWA problem is known to be NP-hard [8] for general network topologies [2]. In view of the importance of this problem, many approaches have been proposed in the literature. We refer the readers to [1], [11], [12], [14], [18], [19], [20], [21] and the references therein for a detailed review of works in this area. These works have considered both one-hop and multi-hop networks and have used relaxations of integer linear programming formulations and sophisticated approximation techniques for multicommodity flow problems.

In a recent paper [6], Datta *et al.* studied the problem (to be called the **OLET** problem) of establishing a set of disjoint lightpaths on a tree topology using a single wavelength to maximize the total one-hop traffic demand that is supported by the chosen set of lightpaths. The authors presented a dynamic programming algorithm (called *optimal tree algorithm*) which optimally solves this problem in $O(n^4+n\mathcal{D}^3)$ time, where n is the number of nodes in the network and $\mathcal{D}$ is the maximum node degree. They then presented a heuristic algorithm for the RWA problem on a general graph, using the optimal tree algorithm as a subroutine. Specifically, the heuristic loops over the wavelengths of the network. For each wavelength, the wavelength plane is decomposed into many link-disjoint trees. The optimal tree algorithm is applied to the trees in non-decreasing size of the trees, establishing lightpaths to support one-hop traffic demands that have not yet been supported. They used simulation results to illustrate the advantages of this new heuristic algorithm over an existing heuristic algorithm [20] for the RWA problem.

In this paper, we study the **OLET** problem and present an $O(n^2+n\mathcal{D}^2)$ time algorithm, thereby improving the algorithm of [6] by a factor of n. Our improved algorithm is achieved by carefully reusing computed partial information, both in terms of reducing the time required for computing various bookkeeping information and in terms of solving a set of *closely related* maximum matching problems in asymptotically faster time than solving them independently, making use of the result of Cunningham and Marsh [5].

We wish to point out that Garg *et al.* also presented a

Manuscript received July 27, 2005; revised May 15, 2006. The research of G. Xue, W. Zhang, and J. Tang was supported in part by NSF grants ANI-0312635 and CCF-0431167. The research of K. Thulasiraman was supported in part by NSF grant ANI-0312435.

G. Xue, W. Zhang, and J. Tang are with the Department of Computer Science and Engineering, Arizona State University, Tempe, AZ 85287-8809 (email: {xue, weiyi.zhang, jian.tang}@asu.edu).

K. Thulasiraman is Professor and Hitachi Chair in the School of Computer Science at the University of Oklahoma, Norman, OK 73019 (email: thulasi@ou.edu).

Digital Object Identifier 10.1109/JSAC-OCN.2006.22605.

TABLE I
TRAFFIC DEMANDS.

	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
v_0	0	7	17	21	7	1	25	10	21	1
v_1	7	0	10	1	6	7	8	1	1	31
v_2	17	10	0	4	1	6	25	22	25	37
v_3	21	1	4	0	13	17	6	13	15	1
v_4	7	6	1	13	0	10	17	1	13	8
v_5	1	7	6	17	10	0	1	9	21	19
v_6	25	8	25	6	17	1	0	10	9	1
v_7	10	1	22	13	1	9	10	0	13	1
v_8	21	1	25	15	13	21	9	13	0	10
v_9	1	31	37	1	8	19	1	1	10	0

method to solve a set of closely related maximum matching problems in asymptotically faster time than solving them independently [9], *within the context of finding maximum integral flow on a tree network with unit edge capacities*. It is interesting to note that finding maximum integral flow on a tree network with unit edge capacities is equivalent to *a special case* of the **OLET** problem which is studied in [6] and the current paper, *where the traffic demand takes binary values*, rather than arbitrary nonnegative integer values. Related work can also be found in [3], [4], where Costa *et al.* generalized the result of [9] to the case where the edge capacities can be arbitrary positive integers, but under the restriction that the tree can be oriented into a rooted tree such that for each source-destination pair, there is a directed path from the source to the destination in this rooted tree. This problem is different from the **OLET** problem that we are studying.

The rest of this paper is organized as follows. In Section II, we present notations and definitions. In Section III, we present an $O(n^2 + n\mathcal{D}^3)$ time algorithm. In Section IV, we discuss how to solve $|V| + 1$ related maximum matching problems in $O(|V|^3)$ time, which results in an $O(n^2 + n\mathcal{D}^2)$ time implementation of the algorithm presented in Section III. We conclude the paper in Section V.

II. DEFINITIONS AND NOTATIONS

We will use T to denote a tree network with n nodes and $n-1$ edges. For two nodes u and v in T, we will use $\pi(u, v)$ to denote the unique path in T which connects u and v. For a path π in T, we will use $s(\pi)$ and $t(\pi)$ to denote the two end nodes of the path. We use *nodes* and *vertices* interchangeably, as well as *links* and *edges*.

We study the same problem defined by Datta *et al.* [6], i.e., establishing a set of link-disjoint lightpaths on a tree topology using a single wavelength to maximize the total one-hop traffic demand supported by the chosen set of lightpaths. Since the problem concentrates on one wavelength at a time, it can be clearly formulated as follows without the notion of WDM and wavelengths.

Definition 2.1: [Optimal Lightpath Establishment on Tree (**OLET**)] We are given a tree network $T = (V, E)$, where V is the set of n nodes and E is the set of $n-1$ links. Between each pair of nodes $u, v \in V$, there is a given symmetric **traffic demand** $w(u, v) = w(v, u) \geq 0$ (we assume $w(u, u) = 0$ by convention). For a path π in T, the **path gain** of π, denoted by $g(\pi)$, is $g(\pi) = w(s(\pi), t(\pi))$. The **OLET** problem seeks a set of link-disjoint paths $\pi_1, \pi_2, \ldots, \pi_L$ in T to maximize the total gain, i.e., we want to find a set of link-disjoint paths $\pi_1, \pi_2, \ldots, \pi_L$ in T so that $\sum_{k=1}^{L} g(\pi_k)$ is maximized. We use $\mathcal{G}(T)$ to denote this maximum value and call it the **gain** of tree T. Any set of link-disjoint paths in T achieving a total gain of $\mathcal{G}(T)$ is called an **optimal set of link-disjoint paths in** T.

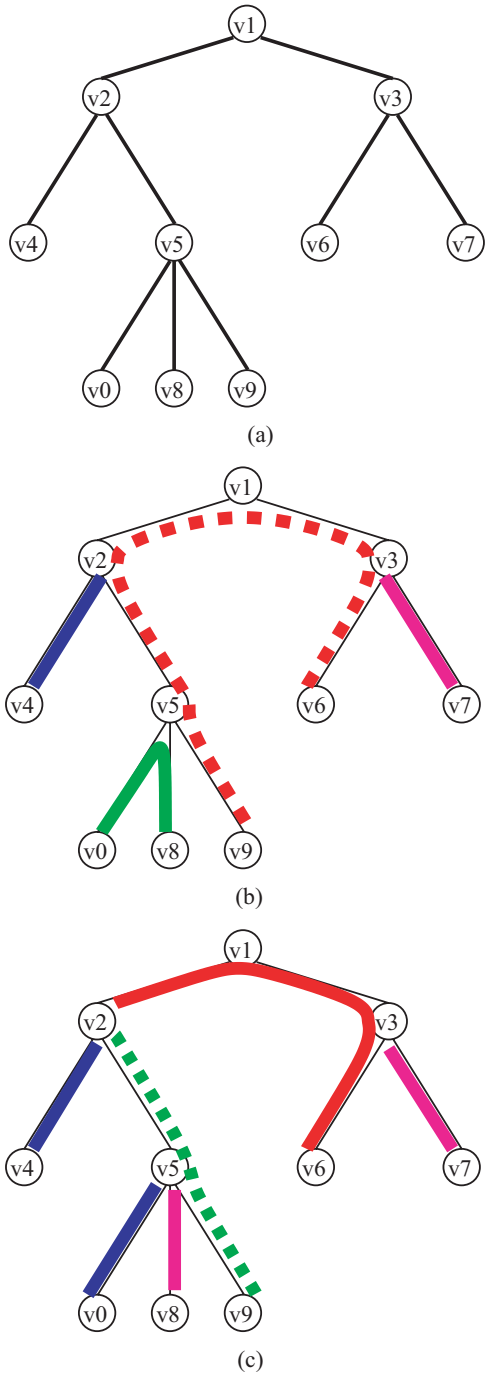


Fig. 1. (a) a tree network; (b) a set of link-disjoint paths; (c) an optimal set of link-disjoint paths.

We use the simple tree network shown in Figure 1(a) and the demand matrix shown in Table I to illustrate the concepts.

The entries in Table I represent the units of traffic demand requested between each node pair (for example, the number of OC-1s), where it is assumed that one wavelength is sufficient to carry all of the traffic between any given node pair. The tree in Figure 1(a) has 10 nodes and 9 links. Figure 1(b) shows four link-disjoint paths $\pi_1 = (v_0, v_5, v_8)$, $\pi_2 = (v_2, v_4)$, $\pi_3 = (v_3, v_7)$, $\pi_4 = (v_6, v_3, v_1, v_2, v_5, v_9)$. The path gains of the four paths are $g(\pi_1) = w(v_0, v_8) = 21$, $g(\pi_2) = w(v_2, v_4) = 1$, $g(\pi_3) = w(v_3, v_7) = 13$, $g(\pi_4) = w(v_6, v_9) = 1$, respectively. Therefore the total gain of these four link-disjoint paths is 36. Figure 1(c) shows six link-disjoint paths achieving a total gain of 98.

Note that a path π is also a tree. However, $g(\pi)$ (the path gain of π, where π is viewed as a path) and $\mathcal{G}(\pi)$ (the gain of π, where π is viewed as a tree) are different in general. For example, for $\pi_4 = (v_6, v_3, v_1, v_2, v_5, v_9)$ in Figure 1(b), we have $g(\pi_4) = w(v_6, v_9) = 1$, but $\mathcal{G}(\pi_4) = w(v_6, v_2) + w(v_2, v_9) = 62$.

Note that the concept of *the gain of a tree* T generalizes naturally to *the gain of a forest* F, which is the maximum total gain achievable by a set of link-disjoint paths in F. We will use this generalized concept to simplify notations.

Although the tree T is an undirected graph by definition, we can treat it as a *rooted tree* by selecting any of the tree nodes as the root of the tree. For example, the tree in Figure 1(a) can be viewed as having root at node v_1. There may be different ways to turn an undirected tree into a rooted tree. However, once the root node is chosen, the corresponding rooted tree can be uniquely identified. We point out that the choice of the rooted tree is an algorithmic technique, which does not affect the computed solution of the problem. Throughout this paper, we will assume that T is a rooted tree with root node r. Every node u other than the root node has a unique *parent node*, denoted by $p(u)$. Every non-leaf node u in T has $K_u \geq 1$ *child nodes*, denoted by $uv_1, uv_2, \ldots, uv_{K_u}$.

Definition 2.2: Let u be any node in T. We use $T(u)$ to denote the subtree rooted at node u. $T(u)$ defines another instance of the **OLET** problem. We use $\mathcal{G}(u) = \mathcal{G}(T(u))$ to denote the gain of $T(u)$.

We use the tree network shown in Figure 1(a) with demand matrix shown in Table I to illustrate the concept of $\mathcal{G}(u)$. For each leaf node u, we have $\mathcal{G}(u) = 0$. Therefore we have $\mathcal{G}(v_4) = \mathcal{G}(v_0) = \mathcal{G}(v_8) = \mathcal{G}(v_9) = \mathcal{G}(v_6) = \mathcal{G}(v_7) = 0$. It is more involved to compute $\mathcal{G}(u)$ for a non-leaf node u. To compute $\mathcal{G}(v_3)$, we note that $T(v_3)$ is a tree with three nodes v_3, v_6, v_7, and two edges (v_3, v_6) and (v_3, v_7). For this subtree, we could either establish one path (v_6, v_3, v_7) with a gain of $w(v_6, v_7) = 10$, or two link-disjoint paths (v_3, v_6) and (v_3, v_7) with a total gain of $w(v_3, v_6) + w(v_3, v_7) = 19$. Therefore we conclude that $\mathcal{G}(v_3) = 19$. To compute $\mathcal{G}(v_5)$, we note that we have the following four sets of link-disjoint paths: $\{(v_0, v_5, v_8), (v_5, v_9)\}$, $\{(v_0, v_5, v_9), (v_5, v_8)\}$, $\{(v_8, v_5, v_9), (v_5, v_0)\}$, $\{(v_5, v_0), (v_5, v_8), (v_5, v_9)\}$, with the last set achieving the maximum total gain of $\mathcal{G}(v_5) = 41$.

Definition 2.3: For any two nodes $u, v \in T$, we use $\pi(u, v)$ to denote the unique (undirected) path connecting u and v. Let u be any node in T and α be any node in $T(u)$. We use $T(u)/\pi(u, \alpha)$ to denote the forest obtained by removing all links on the path $\pi(u, \alpha)$ from tree $T(u)$. We use $\mathcal{P}(u, \alpha)$ to denote the gain of $T(u)/\pi(u, \alpha)$. It is clear that $\mathcal{P}(u, u) = \mathcal{G}(u)$. Let x be a child node of u. We use $T(u)/T(x)$ to denote the set of links that belong to $T(u)$, but not $T(x)$.

Note that when $\alpha \neq u$, $\mathcal{P}(u, \alpha) + w(u, \alpha)$ is in fact the maximum gain achievable by a set of link-disjoint paths in $T(u)$, subject to the condition that $\pi(u, \alpha)$ is one of the link-disjoint paths.

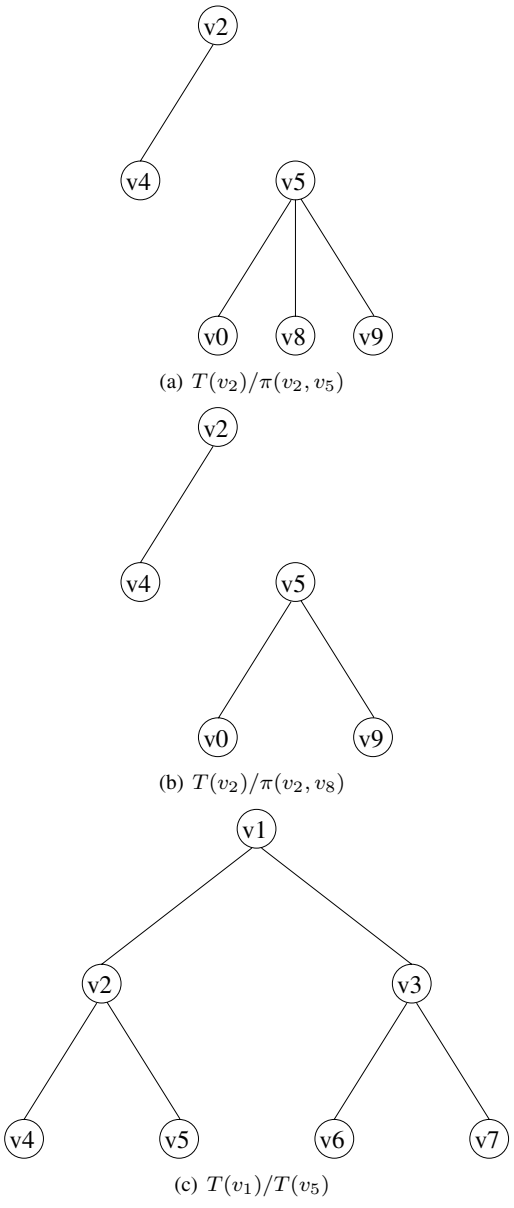


Fig. 2. Illustration of the concepts defined in Definition 2.3.

Figure 2 illustrates the concepts defined in Definition 2.3, using the tree network shown in Figure 1(a) and the demand matrix in Table I. Suppose $u = v_2$ and $\alpha = v_5$, $T(u)/\pi(u, \alpha) = T(v_2)/\pi(v_2, v_5)$ contains the links shown in Figure 2(a), i.e., all links in $T(v_2)$, except those on the path $\pi(v_2, v_5)$. Since $\mathcal{G}(v_5) = 41$ and $w(v_2, v_4) = 1$, we observe that $\mathcal{P}(v_2, v_5) = 42$. Suppose $u = v_2$ and $\alpha = v_8$, $T(u)/\pi(u, \alpha) = T(v_2)/\pi(v_2, v_8)$ contains the links shown in Figure 2(b), i.e., all links in $T(v_2)$, except those on the path $\pi(v_2, v_8)$. Since the maximum gain with the three links $(v_2, v_4), (v_5, v_0)$ and (v_5, v_9) is 21 (using the link-

disjoint paths $(v_2, v_4), (v_5, v_0)$ and (v_5, v_9)), we observe that $\mathcal{P}(v_2, v_8) = 21$. Suppose $u = v_1$ and $x = v_5$, $T(u)/T(x) = T(v_1)/T(v_5)$ contains the links shown in Figure 2(c), i.e., all links on $T(v_1)$, except those on $T(v_5)$.

The notion of $\mathcal{P}(u, \alpha)$ introduced in the above definition enables one to *save* and *reuse* computed information while performing the bottom-up algorithm of the next section. This leads to $O(n^2+n\mathcal{D}^2)$ computation of the required information. In contrast, the algorithm of [6] requires $O(n^4)$ time for this computation. We will revisit this point in Section III after the proof of Theorem 3.2.

III. An $O(n^2 + n\mathcal{D}^3)$ Time Algorithm for **OLET**

In this section, we present an improved algorithm for the **OLET** problem. The algorithm first performs a *bottom-up* computation of various information, including $\mathcal{G}(T)$, and stores the computed information at the nodes of the tree. It then performs a *top-down* computation to establish the optimal set of link-disjoint paths achieving the maximum gain. The bottom-up computation is carried out by Algorithm 1 (**OLET-UP**) while the top-down computation is carried out by Algorithm 2 (**OLET-DOWN**). For ease of understanding, nodes are colored either *black* or *white* within Algorithm 1. Similarly, links are colored either *black* or *white* within Algorithm 2.

The main idea behind Algorithm 1 is as follows. We compute the gain $\mathcal{G}(r)$ of the given tree T recursively by computing the gains of subtrees of T. Suppose we wish to compute the gain of the subtree $T(u)$ rooted at node u. We first note that the set of link-disjoint paths that achieve the gain of $T(u)$ must contain a set of paths that together include the links connecting u to its child nodes. These paths form a subtree T' of $T(u)$. Removing the links of T' from $T(u)$ will result in one or more subtrees of $T(u)$ that form a forest T''. The gain of $T(u)$ is the sum of the gain of T' and the gain of T''. As an example, for the tree rooted at v_1 in Figure 1(c), T' is the subtree formed by the links $(v_1, v_2), (v_1, v_3)$ and (v_3, v_6). So the gain of T' is $w(v_2, v_6) = 25$. T'' is the forest formed by the links (v_3, v_7), (v_2, v_4), (v_2, v_5), (v_5, v_0), (v_5, v_8), and (v_5, v_9). The gain of this forest is 73. So the gain of $T(u)$ is 98. In Algorithm 1, the links of the maximum weighted matching found in Step_5 provide information on finding the tree T'. The information required to compute the gain of the forest is computed in Step_6.

We now proceed to give a step by step discussion of Algorithm 1.

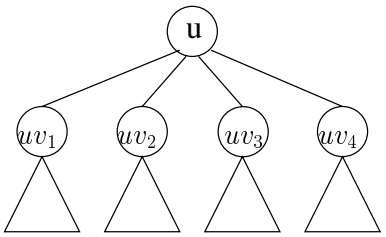
Throughout the execution of Algorithm 1, a node v has color *black* if and only if the gain of $T(v)$ has been computed. For each leaf node $v \in T$, the gain of $T(v)$ is set to $\mathcal{G}(v) := 0$, the gain of $T(v)/T(v)$ is set to $\mathcal{P}(v, v) := 0$, because $\mathcal{P}(v, v)$ is equal to $\mathcal{G}(v)$. We mark each leaf node of T *black* and mark each non-leaf node of T *white*. These are accomplished in Step_1 of the algorithm.

If all tree nodes are marked *black*, we have computed the gain of T and Algorithm 1 stops. Otherwise, there must exist a *white* node u such that all of its child nodes are already marked *black*. For such a *white* node u, we use K_u to denote its number of child nodes and use $uv_1, uv_2, \ldots, uv_{K_u}$ to denote its child nodes. These actions are carried out in Step_2 of the algorithm.

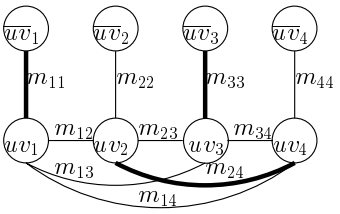
In Step_3 of the algorithm, for each child node uv_k of u, we compute *the maximum total gain that can be achieved by a set of link-disjoint paths in* $T(uv_k) \cup (u, uv_k)$, and use m_{kk} to denote this value. Let $k \in \{1, 2, \ldots, K_u\}$ be chosen. For each node $\gamma \in T(uv_k)$, we use $\phi(u, \gamma)$ to denote the sum of the path gain of $\pi(u, \gamma)$ and the gain of the forest $T(uv_k)/\pi(uv_k, \gamma)$, i.e., $\phi(u, \gamma) = g(\pi(u, \gamma)) + \mathcal{G}(T(uv_k)/\pi(uv_k, \gamma)) = w(u, \gamma) + \mathcal{P}(uv_k, \gamma)$. Note that $\phi(u, \gamma)$ is the maximum total gain that can be achieved by a set of link-disjoint paths in $T(uv_k) \cup (u, uv_k)$, *subject to the constraint that* $\pi(u, \gamma)$ *is one of the link-disjoint paths*. Since we wish to maximize this gain, we have $m_{kk} = \max\{\phi(u, \gamma) | \gamma \in T(uv_k)\}$. We use $\gamma(u, uv_k)$ to denote a node in $T(uv_k)$ such that $\phi(u, \gamma(u, uv_k)) = m_{kk}$. The nodes $\gamma(u, uv_k), k = 1, \ldots, K_u$, will be used by Algorithm 2 to set up an optimal set of link-disjoint paths.

In Step_4 of the algorithm, for each pair of child nodes uv_i and uv_j of u, we compute *the maximum total gain that can be achieved by a set of link-disjoint paths in* $T(uv_i) \cup (uv_i, u) \cup (u, uv_j) \cup T(uv_j)$, *under the constraint that the set of paths contains a path connecting a node in* $T(uv_i)$ *with a node in* $T(uv_j)$, and use m_{ij} to denote this value. Let i and j be chosen such that $1 \le i < j \le K_u$. For each pair of nodes α and β such that $\alpha \in T(uv_i)$ and $\beta \in T(uv_j)$, we use $\psi(\alpha, \beta)$ to denote the sum of the path gain of $\pi(\alpha, \beta)$, the gain of the forest $T(uv_i)/\pi(uv_i, \alpha)$, and the gain of the forest $T(uv_j)/\pi(uv_j, \beta)$, i.e., $\psi(\alpha, \beta) = g(\pi(\alpha, \beta)) + \mathcal{G}(T(uv_i)/\pi(uv_i, \alpha)) + \mathcal{G}(T(uv_j)/\pi(uv_j, \beta)) = w(\alpha, \beta) + \mathcal{P}(uv_i, \alpha) + \mathcal{P}(uv_j, \beta)$. Clearly, $m_{ij} = \max\{\psi(\alpha, \beta) | \alpha \in T(uv_i), \beta \in T(uv_j)\}$. We use $\alpha(uv_i, uv_j)$ and $\beta(uv_i, uv_j)$ to denote a pair of nodes where $\alpha(uv_i, uv_j) \in T(uv_i)$ and $\beta(uv_i, uv_j) \in T(uv_j)$ such that $\psi(\alpha(uv_i, uv_j), \beta(uv_i, uv_j)) = m_{ij}$. The nodes $\alpha(uv_i, uv_j), \beta(uv_i, uv_j), 1 \le i < j \le K_u$, will be used by Algorithm 2 to set up an optimal set of link-disjoint paths.

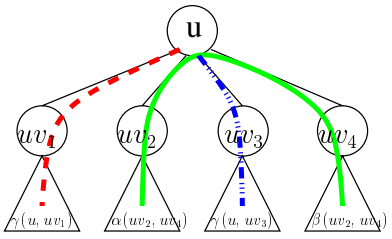
In Step_5 of the algorithm, we compute the gain $\mathcal{G}(u)$ of tree $T(u)$. In order to compute an optimal set of link-disjoint paths in $T(u)$, we need to decide how each of the K_u links $(u, uv_1), \ldots, (u, uv_{K_u})$ are used by the link-disjoint paths. Note that a link (u, uv_i) can be used by one of the link-disjoint paths in one of the following three ways. (1) (u, uv_i) is used by a path $\pi(u, \gamma(u, uv_i))$ connecting nodes u and $\gamma(u, uv_i) \in T(uv_i)$; (2) (u, uv_i) is used together with another link (u, uv_j) $(i < j)$ by a path connecting nodes $\alpha(uv_i, uv_j) \in T(uv_i)$ and $\beta(uv_i, uv_j) \in T(uv_j)$; (3) (u, uv_i) is used together with another link (u, uv_j) $(i > j)$ by a path connecting nodes $\alpha(uv_j, uv_i) \in T(uv_j)$ and $\beta(uv_j, uv_i) \in T(uv_i)$. For this purpose, we solve a maximum weighted matching problem [7], [13] on a graph G_u with $2K_u$ nodes and $\frac{K_u(K_u-1)}{2} + K_u$ links. To handle demand with zero value, we assume in this paper that the maximum weighted matching is one which not only has the maximum sum of edge weight, but also has the maximum number of edges among all matchings with the maximum sum of edge weight. This assumption does not lose any generality, as for each edge e in the graph, we can replace the edge weight $w(e)$ by a 2-tuple $(w(e), 1)$ and use lexicographical order when comparing weights. The graph G_u can be constructed in the following way: First, build a complete graph on the vertices $\{uv_k | 1 \le k \le K_u\}$,



(a) a *white* node u with four *black* child nodes



(b) graph G_u and its maximum weighted matching



(c) top level disjoint paths derived from the matching

Fig. 3. Illustration of the concepts used in Step_5 of Algorithm 1.

where the edge (uv_i, uv_j) has weight $w_u(uv_i, uv_j) = m_{ij}$ for $1 \leq i < j \leq K_u$. Then for each $k = 1, 2, \ldots, K_u$, add a vertex $\overline{uv}_k$ and and an edge $(uv_k, \overline{uv}_k)$ with weight m_{kk}. $\mathcal{G}(u)$, the gain of $T(u)$, is equal to the value of the maximum weighted matching of G_u so constructed. Once a maximum weighted matching $\mathcal{M}_u$ of G_u is computed, we not only know the value of $\mathcal{G}(u)$, but also know that there exists a set $\mathcal{S}_u$ of link-disjoint paths in $T(u)$ whose total gain is $\mathcal{G}(u)$ and that (1) if edge $(uv_i, \overline{uv}_i)$ in G_u is an edge in the matching $\mathcal{M}_u$, then the edge (u, uv_i) in $T(u)$ is used by path $\pi(u, \gamma(u, uv_i))$ in $\mathcal{S}_u$; (2) if edge (uv_i, uv_j) (with $1 \leq i < j \leq K_u$) in G_u is an edge in the matching $\mathcal{M}_u$, then edges (u, uv_i) and (u, uv_j) in $T(u)$ are used together by path $\pi(\alpha(uv_i, uv_j), \beta(uv_i, uv_j))$ in $\mathcal{S}_u$. In addition $\mathcal{S}_u$ may contain other paths in $T(uv_k)$, $1 \leq k \leq K_u$. These paths will be determined by Algorithm 2.

We use Figure 3 to illustrate the concepts used in Step_5. Figure 3(a) shows a *white* node u that has four (*black*) child nodes uv_1, uv_2, uv_3, uv_4. Figure 3(b) shows the graph G_u. Suppose the computed maximum weighted matching $\mathcal{M}_u$ contains the three edges $\{(uv_1, \overline{uv}_1), (uv_3, \overline{uv}_3), (uv_2, uv_4)\}$, as shown in thick dark edges in Figure 1(b). Then we know that $\mathcal{G}(u) = m_{11} + m_{33} + m_{24}$. Furthermore, we know that there exists an optimal set $\mathcal{S}_u$ of link-disjoint paths in $T(u)$ such that $\mathcal{S}_u$ contains (1) the path $\pi(u, \gamma(u, uv_1))$ (because $(uv_1, \overline{uv}_1) \in \mathcal{M}_u$); (2) the path $\pi(u, \gamma(u, uv_3))$ (because $(uv_3, \overline{uv}_3) \in \mathcal{M}_u$); and (3) the path $\pi(\alpha(uv_2, uv_4), \beta(uv_2, uv_4))$ (because $(uv_2, uv_4) \in \mathcal{M}_u$); together with (a) an optimal set of link-disjoint paths in $T(uv_1)/\pi(uv_1, \gamma(u, uv_1))$; (b) an optimal set of link-disjoint paths in $T(uv_2)/\pi(uv_2,\ \alpha(uv_2, uv_4))$; (c) an optimal set of link-disjoint paths in $T(uv_3)/\pi(uv_3, \gamma(u, uv_3))$; and (d) an optimal set of link-disjoint paths in $T(uv_4)/\pi(uv_4,\ \beta(uv_2, uv_4))$. This is illustrated in Figure 3(c).

While computing m_{kk} in Step_3, we need the value of $\mathcal{P}(uv_k, \gamma)$ for each $\gamma \in T(uv_k)$ (we also need these values while computing m_{ij} for $1 \leq i < j \leq K_u$ in Step_4). This means that if u is not the root node, then we need to calculate $\mathcal{P}(u, \alpha)$ for each $\alpha \in T(u)$. We compute these values in Step_6. Recall that $\mathcal{P}(u, \alpha)$ is the gain of the forest $T(u)/\pi(u, \alpha)$. Therefore $w(u, \alpha) + \mathcal{P}(u, \alpha)$ is in fact *the maximum total gain achievable by a set of link-disjoint paths in $T(u)$, subject to the constraint that $\pi(u, \alpha)$ is one of the paths in the set*. If $\pi(u, \alpha)$ contains the node uv_k, then in the set of link-disjoint paths that achieve the gain of $T(u)/\pi(u, \alpha)$, there will be no path using the link (u, uv_k), so this gain is equal to the weight of a maximum weighted matching $\mathcal{M}_{u,uv_k}$ of the graph G_{u,uv_k} obtained by deleting the nodes uv_k and $\overline{uv}_k$ from G_u. This computation is given in line 4 of Step_6. This step involves computing K_u maximum weighted matchings and can be accomplished in $O(K_u^4)$ time using the algorithms of [7], [13].

A complete execution of the algorithm on the example defined by the tree in Figure 1(a) and the demand matrix in Table I is illustrated in Appendix . The computation leads to $\mathcal{G}(v_1) = 98$.

Algorithm 1 OLET-UP(T, r, w), Part I:

Step_1 for each leaf node $v \in T$ do
  $\mathcal{G}(v) := 0; \mathcal{P}(v, v) := 0$; $color(v) := black$;
endfor
for each non-leaf node $v \in T$ do
  $color(v) := white$;
endfor

Step_2 if all tree nodes are marked *black* stop; Otherwise let u be a *white* tree node, with all child nodes already marked *black*. Let $uv_1, uv_2, \ldots, uv_{K_u}$ be the K_u child nodes of node u.

Step_3 for $k = 1, 2, \ldots, K_u$ do
  for each node $\gamma \in T(uv_k)$ do
    $\phi(u, \gamma) := w(u, \gamma) + \mathcal{P}(uv_k, \gamma)$;
  endfor
  $m_{kk} := \max\{\phi(u, \gamma) | \gamma \in T(uv_k)\}$;
  Let $\gamma(u, uv_k) \in T(uv_k)$ be a node such that $\phi(u, \gamma(u, uv_k)) = m_{kk}$.
endfor

Step_4 for each pair of indices $i, j \in \{1, 2, \ldots, K_u\}$ such that $1 \leq i < j \leq K_u$ do
  for each pair of $\alpha \in T(uv_i)$ and $\beta \in T(uv_j)$ do
    $\psi(\alpha, \beta) := w(\alpha, \beta) + \mathcal{P}(uv_i, \alpha) + \mathcal{P}(uv_j, \beta)$;
  endfor
  $m_{ij} := \max\{\psi(\alpha, \beta) | \alpha \in T(uv_i), \beta \in T(uv_j)\}$;
  $m_{ji} := m_{ij}$;
  Let $\alpha(uv_i, uv_j) \in T(uv_i)$ and $\beta(uv_i, uv_j) \in T(uv_j)$ be two nodes such that $\psi(\alpha(uv_i, uv_j), \beta(uv_i, uv_j)) = m_{ij}$.
endfor

Algorithm 1 OLET-UP(T, r, w), Part II:

Step_5 Construct an auxiliary undirected link weighted graph $G_u = (V_u, E_u, w_u)$, where the set of nodes is $V_u = \{uv_1, uv_2, \ldots, uv_{K_u}, \overline{uv}_1, \overline{uv}_2, \ldots, \overline{uv}_{K_u}\}$. For each pair of indices $1 \le i < j \le K_u$, E_u contains a link (uv_i, uv_j) with weight $w_u(uv_i, uv_j) = m_{ij}$. For each index $1 \le k \le K_u$, E_u also contains a link $(\overline{uv}_k, uv_k)$ with weight $w_u(\overline{uv}_k, uv_k) = m_{kk}$. Compute a maximum weighted matching $\mathcal{M}_u$ of G_u. Set $\mathcal{G}(u) := \mathcal{P}(u,u) := w_u(\mathcal{M}_u)$, the weight of $\mathcal{M}_u$. Let $\widehat{\mathcal{M}_u}$ be a set of node pairs (stored at node u) such that
  a. $(uv_i, uv_j) \in \widehat{\mathcal{M}_u}$ if and only if $(uv_i, uv_j) \in \mathcal{M}_u$;
  b. $(u, uv_j) \in \widehat{\mathcal{M}_u}$ if and only if $(\overline{uv}_j, uv_j) \in \mathcal{M}_u$.
if u is the root of T then $color(u) := black$; stop; endif

Step_6 for each $k = 1, 2, ..., K_u$ do
  Let G_{u,uv_k} be the subgraph of G_u with nodes uv_k and $\overline{uv}_k$ removed.
  Compute a maximum weighted matching $\mathcal{M}_{u,uv_k}$ of G_{u,uv_k}.
  for each node $\alpha \in T(uv_k)$ do $\mathcal{P}(u,\alpha) := \mathcal{P}(uv_k, \alpha) + w_u(\mathcal{M}_{u,uv_k})$; endfor
  Let $\widehat{\mathcal{M}_{u,uv_k}}$ be the set of node pairs (stored at node uv_k) such that
    a. $(uv_i, uv_j) \in \widehat{\mathcal{M}_{u,uv_k}}$ if and only if $(uv_i, uv_j) \in \mathcal{M}_{u,uv_k}$;
    b. $(u, uv_j) \in \widehat{\mathcal{M}_{u,uv_k}}$ if and only if $(\overline{uv}_j, uv_j) \in \mathcal{M}_{u,uv_k}$.
  endfor
$color(u) := black$; goto Step_2.

Theorem 3.1: At the time a node u is marked *black* during the execution of Algorithm 1, we have the following facts.

1) For every node $\alpha \in T(u)$, we have $color(\alpha) = black$.
2) $\mathcal{G}(u)$ is the gain of $T(u)$. In addition, the gain $\mathcal{G}(u)$ of $T(u)$ can be achieved by a set $\mathcal{S}_u$ of link-disjoint paths in $T(u)$, with the use of the links connecting node u and its child nodes $\{uv_1, uv_2, \ldots, uv_{K_u}\}$ decided by the maximum matching $\mathcal{M}_u$: if $\mathcal{M}_u$ contains a node pair (uv_i, uv_j) with $i < j$, then $\mathcal{S}_u$ contains path $\pi(\alpha(uv_i, uv_j), \beta(uv_i, uv_j))$ which uses both (u, uv_i) and (u, uv_j); if $\mathcal{M}_u$ contains a node pair $(uv_i, \overline{uv}_i)$, then $\mathcal{S}_u$ contains path $\pi(u, \gamma(u, uv_i))$ which uses link (u, uv_i).
3) For any node $\alpha \in T(u)$, $\mathcal{P}(u,\alpha)$ is the gain of the forest $T(u)/\pi(u,\alpha)$. In addition, if $\pi(u,\alpha)$ contains a child node uv_k of u, the gain $\mathcal{P}(u,\alpha)$ of $T(u)/\pi(u,\alpha)$ can be achieved by a set $\mathcal{S}_{u,uv_k}$ of link-disjoint paths in $T(u)/\pi(u,\alpha)$, with the use of the links connecting node u and its child nodes $\{uv_1, \ldots, uv_{K_u}\} \setminus \{uv_k\}$ decided by the maximum matching $\mathcal{M}_{u,uv_k}$: if $\mathcal{M}_{u,uv_k}$ contains a node pair (uv_i, uv_j) with $i < j$, then $\mathcal{S}_{u,uv_k}$ contains path $\pi(\alpha(uv_i, uv_j), \beta(uv_i, uv_j))$ which uses both (u, uv_i) and (u, uv_j); if $\mathcal{M}_{u,uv_k}$ contains a node pair $(uv_i, \overline{uv}_i)$, then $\mathcal{S}_{u,uv_k}$ contains path $\pi(u, \gamma(u, uv_i))$ which uses link (u, uv_i).

PROOF. All facts are true when only the leaf nodes are colored *black*. We will prove that when a node u is colored *black* during the execution of the algorithm, all three facts are preserved.

Fact 1 is preserved because the node u chosen in Step_2 has all its child nodes marked *black*.

In oder to compute the gain $\mathcal{G}(u)$ of $T(u)$, we construct an auxiliary graph G_u and compute a maximum weighted matching of G_u. Let $\mathcal{S}_u$ be a set of link-disjoint paths in $T(u)$ that achieves the gain of $T(u)$. Each link $(u, uv_i) \in T(u)$ can be used in one of the following two ways in $\mathcal{S}_u$.

case-1: (u, uv_i) is used, *together with another link* (u, uv_j) where $i < j$, on a path $\pi(\alpha, \beta) \in \mathcal{S}_u$ with end nodes $\alpha \in T(uv_i)$ and $\beta \in T(uv_j)$;

case-2: (u, uv_i) is used on a path $\pi(u, \gamma) \in \mathcal{S}_u$ where $\gamma \in T(uv_i)$.

We note that when case-1 happens, we can assume $\alpha = \alpha(uv_i, uv_j)$ and $\beta = \beta(uv_i, uv_j)$, for otherwise, we can replace α by $\alpha(uv_i, uv_j)$ and β by $\beta(uv_i, uv_j)$ *without reducing the gain*. Also, m_{ij} computed in Step_4 is the summation of path gain of $\pi(\alpha(uv_i, uv_j), \beta(uv_i, uv_j)$ with the gain of $T(uv_i)/\pi(uv_i, \alpha(uv_i, uv_j))$ and the gain of $T(uv_j)/\pi(uv_j, \beta(uv_i, uv_j))$. Similarly, when case-2 happens, we can assume $\gamma = \gamma(u, uv_i)$. Also, m_{ii} computed in Step_3 is the summation of the path gain of $\pi(u, \gamma(u, uv_i))$ with the gain of $T(uv_i)/\pi(uv_i, \gamma(u, uv_i))$. Therefore Fact 2 is preserved when node u is colored *black*.

Now we prove the preservation of Fact 3. Note that we want to compute the gain of the forest $T(u)/\pi(u,\alpha)$ for some node $\alpha \in T(u)$. If $\alpha = u$, $\pi(u,\alpha)$ becomes the empty path, which implies $T(u)/\pi(u,\alpha) = T(u)$, a case handled by Fact 2. So we assume that $\alpha \ne u$. Therefore $\alpha \in T(uv_k)$ for some $k \in \{1, 2, \ldots, K_u\}$. In this case, the gain of $T(u)/\pi(u,\alpha)$ is the sum of the gain of $T(u)/T(uv_k)$ with the gain of $T(uv_k)/\pi(uv_k, \alpha)$. Note that the gain of $T(uv_k)/\pi(uv_k, \alpha)$ has been previously computed (when uv_k is marked *black*) and is stored in $\mathcal{P}(uv_k, \alpha)$. The gain of $T(u)/T(uv_k)$ is computed by a maximum weighted matching of the graph G_{u,uv_k}, using an argument similar to that used for the computation of the gain of $T(u)$. Therefore Fact 3 is also preserved when u is marked *black*. □

Theorem 3.2: Let n be the number of nodes in T and $\mathcal{D}$ be the maximum node degree of T. Algorithm 1 has a worst-case running time of $O(n^2 + n\mathcal{D}^3)$.

PROOF. Step_1 takes $O(n)$ time. Each execution of Step_2 takes $O(1)$ time, as we can keep the nodes with no *white* child nodes in a FIFO queue. This step is executed $O(n)$ times. Therefore the total time required by all executions of Step_2 is $O(n)$.

Each execution of Step_3 takes $O(n)$ time as we spend $O(1)$ time for each descendant γ of node u (node γ loops over all nodes in $T(uv_k)$, k loops over $\{1, 2, \ldots, K_u\}$). This step is executed $O(n)$ times. Therefore the total time required by all executions of Step_3 is $O(n^2)$.

In Step_4 associated with node u, we have to deal with K_u subtrees $T(uv_k)$ for $k = 1, 2, \ldots, K_u$. For each pair of subtrees $T(uv_i)$ and $T(uv_j)$, we need to spend $O(1)$ time for each pair of nodes $\alpha \in T(uv_i)$ and $\beta \in T(uv_j)$. We will not try to bound the time required by each execution of Step_4.

Rather, we will bound the time required by *all* executions of Step_4. Note that the same pair of nodes α and β will be encountered (where we compute $\psi(\alpha,\beta)$) in executions of Step_4 in which the smallest common ancestor of α and β is u. Since there are $O(n^2)$ node pairs, all executions of Step_4 requires $O(n^2)$ time. Therefore the total time required by Step_1-Step_4 is $O(n^2)$.

In the rest of the proof, we will analyze the time required by all executions of Step_5 and Step_6. In Step_5 (at node u), we need to compute a maximum weighted matching of a graph with $2K_u$ nodes and $\frac{K_u(K_u+1)}{2}$ links. In Step_6 (at node u), we need to compute K_u maximum weighted matchings each for a graph with $2K_u-2$ nodes and $\frac{K_u(K_u-1)}{2}$ links. So the time complexity of Step_5 is dominated by that of Step_6. It is well-known that the maximum weighted matching for a graph with $|V|$ nodes can be computed in $O(|V|^3)$ time [7], [13]. Therefore the matching computations at node u requires $O(K_u^4)$ time, which is bounded by $O(K_u \times \mathcal{D}^3)$ where $\mathcal{D}$ is the maximum node degree. Since $\sum_{u\in T} K_u \le 2n$, the complete execution of Step_5 and Step_6 is bounded by $O(n\mathcal{D}^3)$. To summarize, we have shown that the worst-case running time of Algorithm 1 is $O(n^2+n\mathcal{D}^3)$. □

We wish to point out that Algorithm 1 follows the same principle as used in [6] (and in [9], for solving a special case of **OLET** *where each traffic demand is either* 0 *or* 1*). In the algorithm of [6], at each node u, the entries m_{ij} for $1 \le i,j \le K_u$ are computed afresh, in $O(n^3)$ time. This leads to $O(n^4)$ time computation when u loops over all nodes in the tree. In our algorithm, we save useful partial information $\mathcal{P}(u,\alpha)$ for $\alpha \in T(u)$ at node u whenever it is computed. This leads to $O(n^2+n\mathcal{D}^3)$ time computation when u loops over all nodes in the tree, if Gabow's maximum matching algorithm [7] is used. In Section IV, we will show that the time complexity can be improved to $O(n^2+n\mathcal{D}^2)$, using the techniques of Cunningham and Marsh [5] for solving a set of closely related matching problems.*

Once we finish the bottom-up computation, $\mathcal{G}(r)$ contains the gain of T. However, an optimal set of link-disjoint paths achieving this gain is not available yet. Therefore we apply Algorithm 2 to compute an optimal set $\mathcal{PATH}$ of link-disjoint paths in a top-down process. This time, instead of coloring the nodes, we color the links–a link is colored *black* only if a path using that link has been added to $\mathcal{PATH}$.

In Step_1, we initialize $\mathcal{PATH}$ to the empty set $\emptyset$ and color all links to *white*. In Step_2, we have a non-leaf node u such that all links on the path from u to the root are *black* and all the links in $T(u)$ are *white*. For each node pair $(u,uv_k) \in \widehat{\mathcal{M}_u}$ (which is equivalent to $(uv_k,\overline{uv}_k) \in \mathcal{M}_u$), we add the path $\pi(u,\gamma(u,uv_k))$ to $\mathcal{PATH}$ and color all links on the path to *black*. For each node pair $(uv_i,uv_j) \in \widehat{\mathcal{M}_u}$ with $i<j$ (which is equivalent to $(uv_i,uv_j) \in \mathcal{M}_u$), we add the path $\pi(\alpha(uv_i,uv_j),\beta(uv_i,uv_j))$ to $\mathcal{PATH}$ and color all links on the path to *black*. Step_3 checks the stopping condition and decides whether to goto Step_2 or to goto Step_4. When control reaches Step_4, we have a node u such that (1) all links on the path from u to the root of T are *black*; (2) a link connecting u to one of its K_u children is *white*; (3) a link connecting u to some other child of u is *black*. We claim that **there is exactly one *black* link connecting u to one of its child nodes** at this time. We note that we have not performed Step_2 or Step_4 at node u yet. Therefore the only reason for a link (u,uv_k) to be *black* is because we have added to $\mathcal{PATH}$ a path using link (u,uv_k) (and the link $(p(u),u)$) while performing a Step_2 or a Step_4 at one of node u's ancestors. Since link $(p(u),u)$ can only be used by one of the link-disjoint paths, no other child of u can be reached from the root of T by a path of *black* links at this time. Here we

Algorithm 2 OLET-DOWN(T,r,w)

Step_1 for each link $(u,v) \in T$ do
  $color(u,v) := white$;
 endfor
 $\mathcal{PATH} = \emptyset$; Set $u := r$, the root node of tree T.
Step_2 for each node pair $(u,uv_k) \in \widehat{\mathcal{M}_u}$ do
  $\mathcal{PATH} := \mathcal{PATH} \cup \pi(u,\gamma(u,uv_k))$;
  for each link e on $\pi(u,\gamma(u,uv_k))$ do
   $color(e) := black$
  endfor
 endfor
 for each node pair $(uv_i,uv_j) \in \widehat{\mathcal{M}_u}$ do
  // {without loss of generality, assuming $i<j$}
  $\mathcal{PATH} := \mathcal{PATH} \cup \pi(\alpha(uv_i,uv_j),\beta(uv_i,uv_j))$;
  for each link e on $\pi(\alpha(uv_i,uv_j),\beta(uv_i,uv_j))$ do
   $color(e) := black$
  endfor
 endfor
Step_3 if all tree links are marked $black$, stop; Otherwise let u be a non-leaf tree node such that
  a. The link $(p(u),u)$ connecting u and its parent node $p(u)$ in T is $black$;
  b. There is a child v of u such that $color(u,v)$ is $white$.
 Let $uv_1,\ldots,uv_{K_u}$ be the K_u child nodes of node u.
 if $color(u,uv_k) = white$ for all $k=1,\ldots,K_u$ then
  goto Step_2
 else
  goto Step_4
 endif
Step_4 Here we must have $K_u \ge 2$ and exactly one of the K_u links $(u,uv_1),(u,uv_2),\ldots,(u,uv_{K_u})$ has $black$ color. Let this $black$ link be (u,uv_k), where $k \in \{1,2,\ldots,K_u\}$.
 for each node pair $(u,uv_j) \in \widehat{\mathcal{M}_{u,uv_k}}$ do
  $\mathcal{PATH} := \mathcal{PATH} \cup \pi(u,\gamma(u,uv_j))$;
  for each link e on the u–$\gamma(u,uv_j)$ path do
   $color(e) := black$
  endfor
 endfor
 for each node pair $(uv_i,uv_j) \in \widehat{\mathcal{M}_{u,uv_k}}$ do
  // {without loss of generality, assuming $i<j$}
  $\mathcal{PATH} := \mathcal{PATH} \cup \pi(\alpha(uv_i,uv_j),\beta(uv_i,uv_j))$;
  for each link e on $\pi(\alpha(uv_i,uv_j),\beta(uv_i,uv_j))$ do
   $color(e) := black$
  endfor
 endfor

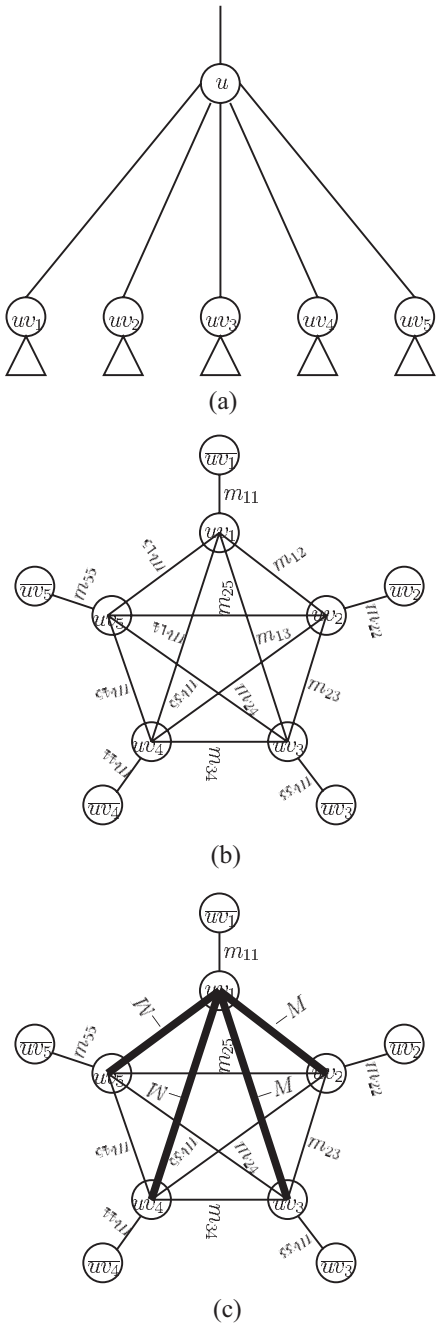


Fig. 4. (a) node u and its 5 child nodes; (b) the graph G_u, or the graph $\overline{G_u}$ without showing the zero-weight edges; (c) the graph $\overline{G_{u,uv_1}}$ without showing the zero-weight edges.

add paths to $\mathcal{PATH}$ in a similar manner as in Step_2, except that we use the matchings $\mathcal{M}_{u,uv_k}$ instead of the matching $\mathcal{M}_u$.

In the description of Algorithm 2, we have used a set $\mathcal{PATH}$ to contain the optimal set of link-disjoint paths achieving the gain of tree T. Note that the paths are added to $\mathcal{PATH}$ in Step_2 and Step_4 of the algorithm. In practice, at the time a path is added to $\mathcal{PATH}$, one can establish the corresponding lightpath in the network.

A complete execution of the algorithm on the example defined by the tree in Figure 1(a) and the demand matrix in Table I is illustrated in Appendix . The computed optimal set link-disjoint paths to this example of **OLET** is illustrated in Figure 1(c).

Theorem 3.3: At the end of Algorithm 1, $\mathcal{G}(r)$ is the gain of T, where r is the root of T. Algorithm 2 correctly constructs a set $\mathcal{S}$ of link-disjoint paths in T in $O(n)$ time.

PROOF. It follows from Fact 2 of Theorem 3.1 that $\mathcal{G}(r)$ is the gain of T. It follows from Fact 2 and Fact 3 of Theorem 3.1 that Algorithm 2 correctly constructs the set of link-disjoint paths $\mathcal{PATH}$ which achieves the gain of T. Since Algorithm 2 traverses the tree T and visits each tree link no more than twice, it takes $O(n)$ time to complete. □

IV. An $O(n^2 + n\mathcal{D}^2)$ Time Implementation

In the previous section, we have shown that the running time of our algorithm is bounded by $O(n^2 + n\mathcal{D}^3)$, where n is the number of nodes in the tree network and $\mathcal{D}$ is the maximum node degree. The term $n\mathcal{D}^3$ is due to the fact that at each node u, we need to solve $K_u + 1$ maximum weighted matching problems. As we pointed out, this can be accomplished in $O(K_u^4)$ time by straightforward applications of the matching algorithms in [7], [13].

We note that for each $k = 1, 2, \ldots, K_u$, the graph G_{u,uv_k} is obtained from the graph G_u by deleting the nodes uv_k and $\overline{uv}_k$. A natural question to ask is the following: *Can we compute a maximum weighted matching of G_{u,uv_k} in $O(K_u^2)$ additional time, after spending $O(K_u^3)$ time to compute a maximum weighted matching for G_u?* The answer is *YES* due to work by Cunningham and Marsh [5].

In [5] Cunningham and Marsh proved the following:

Lemma 4.1 ([5]): Let $G = G(V, E, \omega)$ be an edge weighted undirected graph, where V is the set of vertices, E is the set of edges, and $\omega(e)$ is the weight of edge e for each $e \in E$. Let U be any given subset of V. Let $\omega'(\cdot)$ be a new edge weighting function such that $\omega'(e) \neq \omega(e)$ implies that at least one end node of edge e belongs to U. Assume that a maximum perfect matching for G is already computed. Then a maximum perfect matching for $G' = G(V, E, \omega')$ can be computed in $O(|U| \times |V|^2)$ additional time. □

In the following, we discuss how Lemma 4.1 can be used to improve the running time of Algorithm 1. Suppose that we are executing Step_5 and Step_6 of Algorithm 1 at a node u. Let $uv_1, uv_2, \ldots, uv_{K_u}$ be the child nodes of u (see Figure 4(a) for the case of $K_u = 5$). The graph $G_u = G_u(V_u, E_u, w_u)$ is constructed with vertex set V_u, edge set E_u, and edge weighting function $w_u(\cdot)$, where V_u contains $2K_u$ vertices $\{uv_1, \ldots, uv_{K_u}, \overline{uv}_1, \ldots, \overline{uv}_{K_u}\}$, E_u contains edges in the form $(\overline{uv}_i, uv_i)$ with weight $m_{ii} = \phi(u, \gamma(u, uv_i))$, and edges in the form (uv_i, uv_j) with weight $m_{ij} = \psi(\alpha(uv_i, uv_j), \beta(uv_i, uv_j))$ (see Figure 4(b) for the case of $K_u = 5$).

Let $\overline{G_u}$ be the complete graph on V_u obtained from G_u by adding a zero-weight edge (x, y) for any pair of vertices $x, y \in V_u$ such that $(x, y) \notin E_u$. It is obvious that $\overline{G_u}$ contains a maximum perfect matching and that each maximum perfect matching of $\overline{G_u}$ corresponds to a maximum weighted matching of G_u.

Let $k \in \{1, 2, \ldots, K_u\}$. We need to compute a maximum weighted matching for G_{u,uv_k}, where G_{u,uv_k} is obtained from G_u by deleting the vertices uv_k and $\overline{uv_k}$.

Let M be a very big number, e.g., $M = 1 + \sum_{1 \le i < j \le n} w(i,j)$. Let $\overline{G_{u,uv_k}}$ be the graph obtained from $\overline{G_u}$ by modifying some of the edge weights in the following way: For each edge in the form (uv_k, uv_j), change the weight from m_{kj} to $-M$ (see Figure 4(c) for the case of $K_u = 5$ and $k = 1$ where zero-weight edges are not shown).

Due to the choice of the constant M, no maximum perfect matching of $\overline{G_{u,uv_k}}$ can contain any edge with weight $-M$. Therefore a maximum perfect matching of $\overline{G_{u,uv_k}}$ corresponds to a maximum weighted matching of G_{u,uv_k}.

It follows from Lemma 4.1, a maximum perfect matching of $\overline{G_{u,uv_k}}$ can be computed in $O(K_u^2)$ additional time, provided that a maximum perfect matching for $\overline{G_u}$ has been pre-computed. As a result, the maximum weighted matching of G_u, together with the maximum matchings for $G_{u,uv_k}, k = 1, 2, \ldots, K_u$, can be computed using the algorithm of [5] in $O(K_u^3)$ time. Therefore we have proved the following.

Corollary 4.1: Algorithms **OLET-UP** and **OLET-DOWN** can be implemented in $O(n^2 + n\mathcal{D}^2)$ time, if the Cunningham and Marsh algorithm is used to compute the maximum matching for G_u and $G_{u,uv_k}, k = 1, 2, \ldots, K_u$. □

V. Conclusions

In this paper, we have presented a faster algorithm for optimal lightpath establishment on a tree network. Let n and $\mathcal{D}$ be the number of nodes in the network and the maximum node degree, respectively. Our algorithm requires $O(n^2 + n\mathcal{D}^2)$ time, improving the previous best $O(n^4 + n\mathcal{D}^3)$ time algorithm [6] by a factor of n. The improved running time is achieved by making careful use of computed information, both in terms of partial path information and in terms of maximum weighted matchings of closely related graphs. As in the case of [6], our algorithm can be used as a subroutine in a heuristic for the general RWA problem. When used in such applications, our algorithm will establish exactly the same set of lightpaths as the algorithm of [6], but at a faster speed. The implementation of our algorithm is straightforward, keeping in mind that the maximum matchings of closely related graphs are computed using the method of Cunningham and Marsh [5], rather than using straightforward applications of the method of Gabow [7], [13]. Algorithm 1 can be carried out with the aid of an FIFO queue. Algorithm 2 can be carried out by a pre-order traversal of the tree T.

Acknowledgment

We would like to thank Professor William H. Cunningham for sharing with us his result on fast solutions of closely related maximum matching problems, *i.e.*, Lemma 4.1, which forms the foundation of Section IV, as well as for bringing to our attention references [5] and [9]. Thanks also go to Professors Charles J. Colbourn and Goran Konjevod for helpful discussions. The authors wish to thank the associate editor and the anonymous reviewers whose comments on earlier versions of this paper have helped to significantly improve the presentation of this paper.

Appendix

This appendix is provided as an aid to the readers to ease the understanding of Algorithm 1, using the example given in Section II. To save space, we ignore the values computed at the leaf nodes.

At node $\mathbf{v_5}$ ($u = v_5, K_u = 3, uv_1 = v_0, uv_2 = v_8, uv_3 = v_9$), we perform the following computations:

Step_3: $\phi(v_5, v_0) = w(v_5, v_0) + \mathcal{P}(v_0, v_0) = 1 + 0 = 1$;
$m_{11} = 1$; $\gamma(v_5, v_0) = v_0$;
$\phi(v_5, v_8) = w(v_5, v_8) + \mathcal{P}(v_8, v_8) = 21 + 0 = 21$;
$m_{22} = 21$; $\gamma(v_5, v_8) = v_8$;
$\phi(v_5, v_9) = w(v_5, v_9) + \mathcal{P}(v_9, v_9) = 19 + 0 = 19$;
$m_{33} = 19$; $\gamma(v_5, v_9) = v_9$;

Step_4: $\psi(v_0, v_8) = w(v_0, v_8) + \mathcal{P}(v_0, v_0) + \mathcal{P}(v_8, v_8) = 21 + 0 + 0 = 21$;
$m_{12} = 21$; $\alpha(v_0, v_8) = v_0$; $\beta(v_0, v_8) = v_8$;
$\psi(v_0, v_9) = w(v_0, v_9) + \mathcal{P}(v_0, v_0) + \mathcal{P}(v_9, v_9) = 1 + 0 + 0 = 1$;
$m_{13} = 1$; $\alpha(v_0, v_9) = v_0$; $\beta(v_0, v_9) = v_9$;
$\psi(v_8, v_9) = w(v_8, v_9) + \mathcal{P}(v_8, v_8) + \mathcal{P}(v_9, v_9) = 10 + 0 + 0 = 10$;
$m_{23} = 10$; $\alpha(v_8, v_9) = v_8$; $\beta(v_8, v_9) = v_9$;

Step_5: There is a unique maximum matching, with weight equal to $\mathcal{G}(v_5) = \mathcal{P}(v_5, v_5) = w_{v_5}(\mathcal{M}_{v_5}) = 41$. This matching leads to $\widehat{\mathcal{M}_{v_5}} = \{(v_5, v_0), (v_5, v_8), (v_5, v_9)\}$.

Step_6: For $k = 1$, we have $\mathcal{P}(v_5, v_0) = \mathcal{P}(v_0, v_0) + w_{v_5}(\mathcal{M}_{v_5, v_0}) = 0 + 40 = 40$;
$\widehat{\mathcal{M}_{v_5, v_0}} = \{(v_5, v_8), (v_5, v_9)\}$.
For $k = 2$, we have $\mathcal{P}(v_5, v_8) = \mathcal{P}(v_8, v_8) + w_{v_5}(\mathcal{M}_{v_5, v_8}) = 0 + 20 = 20$;
$\widehat{\mathcal{M}_{v_5, v_8}} = \{(v_5, v_0), (v_5, v_9)\}$.
For $k = 3$, we have $\mathcal{P}(v_5, v_9) = \mathcal{P}(v_9, v_9) + w_{v_5}(\mathcal{M}_{v_5, v_9}) = 0 + 22 = 22$;
$\widehat{\mathcal{M}_{v_5, v_9}} = \{(v_5, v_0), (v_5, v_8)\}$.

At node $\mathbf{v_2}$ ($u = v_2$, $K_u = 2$, $uv_1 = v_4, uv_2 = v_5$), we perform the following computations:

Step_3: $\phi(v_2, v_4) = w(v_2, v_4) + \mathcal{P}(v_4, v_4) = 1 + 0 = 1$;
$m_{11} = 1$; $\gamma(v_2, v_4) = v_4$;
$\phi(v_2, v_5) = w(v_2, v_5) + \mathcal{P}(v_5, v_5) = 6 + 41 = 47$;
$\phi(v_2, v_0) = w(v_2, v_0) + \mathcal{P}(v_5, v_0) = 17 + 40 = 57$;
$\phi(v_2, v_8) = w(v_2, v_8) + \mathcal{P}(v_5, v_8) = 25 + 20 = 45$;
$\phi(v_2, v_9) = w(v_2, v_9) + \mathcal{P}(v_5, v_9) = 37 + 22 = 59$;
$m_{22} = 59$; $\gamma(v_2, v_5) = v_9$;

Step_4: $\psi(v_4, v_5) = w(v_4, v_5) + \mathcal{P}(v_4, v_4) + \mathcal{P}(v_5, v_5) = 10 + 0 + 41 = 51$;
$\psi(v_4, v_0) = w(v_4, v_0) + \mathcal{P}(v_4, v_4) + \mathcal{P}(v_5, v_0) = 7 + 0 + 40 = 47$;
$\psi(v_4, v_8) = w(v_4, v_8) + \mathcal{P}(v_4, v_4) + \mathcal{P}(v_5, v_8) = 13 + 0 + 20 = 33$;
$\psi(v_4, v_9) = w(v_4, v_9) + \mathcal{P}(v_4, v_4) + \mathcal{P}(v_5, v_9) = 8 + 0 + 22 = 30$;
$m_{12} = 51$; $\alpha(v_4, v_5) = v_4$; $\beta(v_4, v_5) = v_5$;

Step_5: There is a unique maximum matching, with weight equal to $\mathcal{G}(v_2) = \mathcal{P}(v_2, v_2) = w_{v_2}(\mathcal{M}_{v_2}) = 60$. It leads to $\widehat{\mathcal{M}_{v_2}} = \{(v_2, v_4), (v_2, v_5)\}$.

Step_6: For $k = 1$, we have $\mathcal{P}(v_2, v_4) = \mathcal{P}(v_4, v_4) + w_{v_2}(\mathcal{M}_{v_2, v_4}) = 0 + 59 = 59$;

$\widehat{\mathcal{M}_{v_2,v_4}} = \{(v_2, v_5)\}$.
For $k = 2$, we have $\mathcal{P}(v_2, v_5) = \mathcal{P}(v_5, v_5) + w_{v_2}(\mathcal{M}_{v_2,v_5}) = 41 + 1 = 42$;
$\mathcal{P}(v_2, v_0) = \mathcal{P}(v_5, v_0) + w_{v_2}(\mathcal{M}_{v_2,v_5}) = 40 + 1 = 41$;
$\mathcal{P}(v_2, v_8) = \mathcal{P}(v_5, v_8) + w_{v_2}(\mathcal{M}_{v_2,v_5}) = 20 + 1 = 21$;
$\mathcal{P}(v_2, v_9) = \mathcal{P}(v_5, v_9) + w_{v_2}(\mathcal{M}_{v_2,v_5}) = 22 + 1 = 23$;
$\widehat{\mathcal{M}_{v_2,v_5}} = \{(v_2, v_4)\}$.

At node $\mathbf{v_3}$ ($u = v_3$, $K_u = 2$, $uv_1 = v_6, uv_2 = v_7$), we perform the following computations:

Step_3: $\phi(v_3, v_6) = w(v_3, v_6) + \mathcal{P}(v_6, v_6) = 6 + 0 = 6$;
$m_{11} = 6$; $\gamma(v_3, v_6) = v_6$;
$\phi(v_3, v_7) = w(v_3, v_7) + \mathcal{P}(v_7, v_7) = 13 + 0 = 13$;
$m_{22} = 13$; $\gamma(v_3, v_7) = v_7$;

Step_4: $\psi(v_6, v_7) = w(v_6, v_7) + \mathcal{P}(v_6, v_6) + \mathcal{P}(v_7, v_7) = 10 + 0 + 0 = 10$;
$m_{12} = 10$; $\alpha(v_6, v_7) = v_6$; $\beta(v_6, v_7) = v_7$;

Step_5: There is a unique maximum matching, with weight equal to $\mathcal{G}(v_3) = \mathcal{P}(v_3, v_3) = w_{v_3}(\mathcal{M}_{v_3}) = 19$. It leads to $\widehat{\mathcal{M}_{v_3}} = \{(v_3, v_6), (v_3, v_7)\}$.

Step_6: For $k = 1$, we have $\mathcal{P}(v_3, v_6) = \mathcal{P}(v_6, v_6) + w_{v_3}(\mathcal{M}_{v_3,v_6}) = 0 + 13 = 13$;
$\widehat{\mathcal{M}_{v_3,v_6}} = \{(v_3, v_7)\}$.
For $k = 2$, we have $\mathcal{P}(v_3, v_7) = \mathcal{P}(v_7, v_7) + w_{v_3}(\mathcal{M}_{v_3,v_7}) = 0 + 6 = 6$;
$\widehat{\mathcal{M}_{v_3,v_7}} = \{(v_3, v_6)\}$.

At node $\mathbf{v_1}$ ($u = v_1$, $K_u = 2$, $uv_1 = v_2, uv_2 = v_3$), we perform the following computations:

Step_3: $\phi(v_1, v_2) = w(v_1, v_2) + \mathcal{P}(v_2, v_2) = 10 + 60 = 70$;
$\phi(v_1, v_0) = w(v_1, v_0) + \mathcal{P}(v_2, v_0) = 7 + 41 = 48$;
$\phi(v_1, v_4) = w(v_1, v_4) + \mathcal{P}(v_2, v_4) = 6 + 59 = 65$;
$\phi(v_1, v_5) = w(v_1, v_5) + \mathcal{P}(v_2, v_5) = 7 + 42 = 49$;
$\phi(v_1, v_8) = w(v_1, v_8) + \mathcal{P}(v_2, v_8) = 1 + 21 = 22$;
$\phi(v_1, v_9) = w(v_1, v_9) + \mathcal{P}(v_2, v_9) = 31 + 23 = 54$;
$m_{11} = 70$; $\gamma(v_1, v_2) = v_2$;
$\phi(v_1, v_3) = w(v_1, v_3) + \mathcal{P}(v_3, v_3) = 1 + 19 = 20$;
$\phi(v_1, v_6) = w(v_1, v_6) + \mathcal{P}(v_3, v_6) = 8 + 13 = 21$;
$\phi(v_1, v_7) = w(v_1, v_7) + \mathcal{P}(v_3, v_7) = 1 + 6 = 7$;
$m_{22} = 21$; $\gamma(v_1, v_3) = v_6$;

Step_4: $\psi(v_2, v_3) = w(v_2, v_3) + \mathcal{P}(v_2, v_2) + \mathcal{P}(v_3, v_3) = 4 + 60 + 19 = 83$;
$\psi(v_2, v_6) = w(v_2, v_6) + \mathcal{P}(v_2, v_2) + \mathcal{P}(v_3, v_6) = 25 + 60 + 13 = 98$;
$\psi(v_2, v_7) = w(v_2, v_7) + \mathcal{P}(v_2, v_2) + \mathcal{P}(v_3, v_7) = 22 + 60 + 6 = 88$;
$\psi(v_0, v_3) = w(v_0, v_3) + \mathcal{P}(v_2, v_0) + \mathcal{P}(v_3, v_3) = 21 + 41 + 19 = 81$;
$\psi(v_0, v_6) = w(v_0, v_6) + \mathcal{P}(v_2, v_0) + \mathcal{P}(v_3, v_6) = 25 + 41 + 13 = 79$;
$\psi(v_0, v_7) = w(v_0, v_7) + \mathcal{P}(v_2, v_0) + \mathcal{P}(v_3, v_7) = 10 + 41 + 6 = 57$;
$\psi(v_4, v_3) = w(v_4, v_3) + \mathcal{P}(v_2, v_4) + \mathcal{P}(v_3, v_3) = 13 + 59 + 19 = 91$;
$\psi(v_4, v_6) = w(v_4, v_6) + \mathcal{P}(v_2, v_4) + \mathcal{P}(v_3, v_6) = 17 + 59 + 13 = 89$;
$\psi(v_4, v_7) = w(v_4, v_7) + \mathcal{P}(v_2, v_4) + \mathcal{P}(v_3, v_7) = 1 + 59 + 6 = 66$;
$\psi(v_5, v_3) = w(v_5, v_3) + \mathcal{P}(v_2, v_5) + \mathcal{P}(v_3, v_3) = 17 + 42 + 19 = 78$;
$\psi(v_5, v_6) = w(v_5, v_6) + \mathcal{P}(v_2, v_5) + \mathcal{P}(v_3, v_6) = 1 + 42 + 13 = 56$;
$\psi(v_5, v_7) = w(v_5, v_7) + \mathcal{P}(v_2, v_5) + \mathcal{P}(v_3, v_7) = 9 + 42 + 6 = 57$;
$\psi(v_8, v_3) = w(v_8, v_3) + \mathcal{P}(v_2, v_8) + \mathcal{P}(v_3, v_3) = 15 + 21 + 19 = 55$;
$\psi(v_8, v_6) = w(v_8, v_6) + \mathcal{P}(v_2, v_8) + \mathcal{P}(v_3, v_6) = 9 + 21 + 13 = 43$;
$\psi(v_8, v_7) = w(v_8, v_7) + \mathcal{P}(v_2, v_8) + \mathcal{P}(v_3, v_7) = 13 + 21 + 6 = 40$;
$\psi(v_9, v_3) = w(v_9, v_3) + \mathcal{P}(v_2, v_9) + \mathcal{P}(v_3, v_3) = 1 + 23 + 19 = 43$;
$\psi(v_9, v_6) = w(v_9, v_6) + \mathcal{P}(v_2, v_9) + \mathcal{P}(v_3, v_6) = 1 + 23 + 13 = 37$;
$\psi(v_9, v_7) = w(v_9, v_7) + \mathcal{P}(v_2, v_9) + \mathcal{P}(v_3, v_7) = 1 + 23 + 6 = 30$;
$m_{12} = 98$; $\alpha(v_2, v_3) = v_2$; $\beta(v_2, v_3) = v_6$;

Step_5: There is a unique maximum matching, with weight equal to $\mathcal{G}(v_1) = \mathcal{P}(v_1, v_1) = w_{v_1}(\mathcal{M}_{v_1}) = 98$. It leads to $\widehat{\mathcal{M}_{v_1}} = \{(v_2, v_3)\}$.

Step_6: v_1 is the root node. So we do not perform Step_6.

At this time, we know that the gain of the tree network T in Figure 1(a) is $\mathcal{G}(v_1) = 98$.

This appendix is provided as an aid to the readers to ease the understanding of Algorithm 2, using the example given in Section II. We assume that we have finished the execution of Algorithm 1. We initialize $\mathcal{PATH} = \emptyset$ and traverse the tree T, starting from the root node v_1.

At node v_1, we find that all links from v_1 to its child nodes are *white* (not used by any path yet). So we go to Step_2. We find that $\widehat{\mathcal{M}_{v_1}} = \{(v_2, v_3)\}$. To establish the path corresponding to the match $(v_2, v_3) \in \widehat{\mathcal{M}_{v_1}}$, we note that $\alpha(v_2, v_3) = v_2$ and $\beta(v_2, v_3) = v_6$. Therefore we establish a path connecting v_2 and v_6. This results in $\mathcal{PATH} = \{\pi(v_2, v_6)\}$. We also color all the links on this path to *black*. Next we visit node v_2 in the tree.

At node v_2, we find that all links from v_2 to its child nodes are *white* (not used by any path yet). So we go to Step_2. We find that $\widehat{\mathcal{M}_{v_2}} = \{(v_2, v_4), (v_2, v_5)\}$. So we establish a path connecting v_2 and $\gamma(v_2, v_4)$ (which is v_4) and a path connecting v_2 and $\gamma(v_2, v_5)$ (which is v_9). We color all the links on these two paths to *black*. This results in $\mathcal{PATH} = \{\pi(v_2, v_6), \pi(v_2, v_4), \pi(v_2, v_9)\}$. Next we visit node v_5 in the tree.

At node v_5, we find that the link from v_5 to its child node v_9 is *black* (already used by a path). So we go to Step_4. We find that $\widehat{\mathcal{M}_{v_5,v_9}} = \{(v_5, v_0), (v_5, v_8)\}$. For the match (v_5, v_0), we establish a path connecting v_5 and $\gamma(v_5, v_0)$ (which is v_0). For the match (v_5, v_8), we establish a path connecting v_5 and $\gamma(v_5, v_8)$ (which is v_8). We color all the links on these two paths *black*. This results in $\mathcal{PATH} = \{\pi(v_2, v_6), \pi(v_2, v_4), \pi(v_2, v_9), \pi(v_5, v_0), \pi(v_5, v_8)\}$. Next we visit node v_3 in the tree.

At node v_3, we find that the link from v_3 to its child node v_6 is *black* (already used by a path). So we go to Step_4. We

find that $\widehat{\mathcal{M}_{v_3,v_6}} = \{(v_3, v_7)\}$. For the match (v_3, v_7), we establish a path connecting v_3 and $\gamma(v_3, v_7)$ (which is v_7). We color all the links on this path *black*. This results in $\mathcal{PATH} = \{\pi(v_2, v_6), \pi(v_2, v_4), \pi(v_2, v_9), \pi(v_5, v_0), \pi(v_5, v_8), \pi(v_3, v_7)\}$. These set of six link-disjoint paths has a gain of 98.

The computed optimal solution to this example of **OLET** is illustrated in Figure 1(c).

REFERENCES

[1] D. Banerjee and B. Mukherjee, "A practical approach for routing and wavelength assignment in large wavelength-routed optical networks," *IEEE J. Sel. Areas Commun.*, vol. 14, no. 5, pp. 903–908, June 1996.

[2] I. Chlamtac, A. Ganz, and G. Karmi, "Lightpath communications: an approach to high bandwidth optical WANs," *IEEE Trans. Commun.*, vol. 40, no. 7, pp. 1171–1182, July 1992.

[3] M. C. Costa, L. Létocart, and F. Roupin, "A greedy algorithm for multicut and integral multiflow in rooted trees," *Operations Research Lett.*, vol. 31, pp. 21–27, 2003.

[4] M. C. Costa, L. Létocart, and F. Roupin, "Minimal multicut and maximal integer multiflow: a survey," *European J. Operational Research*, vol. 162, pp. 55–69, 2005.

[5] W. H. Cunningham and A. B. Marsh, III, "A primal algorithm for optimum matching," *Mathematical Programming Study 8*, pp. 50–72, 1978.

[6] R. Datta, B. Mitra, S. Ghose, and I. Sengupta, "An algorithm for optimal assignment of a wavelength in a tree topology and its applications in WDM networks," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 9, pp. 1589–1600, Nov. 2004.

[7] H. N. Gabow, "Implementation of algorithms for maximum matching and nonbipartite graphs," Ph.D. thesis, Stanford University, 1974.

[8] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[9] N. Garg, V. V. Vazirani, and M. Yannakakis, "Primal-dual approximation algorithms for integral flow and multicut in trees," *Algorithmica*, vol. 18, pp. 3–20, 1997.

[10] P. E. Green, Jr., *Fiber Optic Networks*. Prentice-Hall, Inc., 1993.

[11] X.-H. Jia, D.-Z. Du, X.-D. Hu, M.-K. Lee, and J. Gu, "Optimization of wavelength assignment for QoS multicast in WDM networks," *IEEE Trans. Commun.*, vol. 49, no. 2, pp. 341–350, Feb. 2001.

[12] J. M. Kleinberg, "Approximation algorithms for disjoint paths problems," Ph.D. dissertation, Department of Electrical and Computer Engineering, MIT, 1998.

[13] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston, New York, 1976.

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

[15] B. Mukherjee, "WDM-based local lightwave networks-part I: single-hop systems," *IEEE Network Mag.*, vol. 6, no. 3, pp. 12–27, May 1992.

[16] B. Mukherjee, "WDM-based local lightwave networks-part II: multi-hop systems," *IEEE Network Mag.*, vol. 6, no. 4, pp. 20–32, July 1992.

[17] B. Mukherjee, *Optical Communication Networks*. McGraw Hill, 1997.

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

[19] M. Saad and Z.-Q. Luo, "On the routing and wavelength assignment in multifiber WDM networks," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 9, pp. 1708–1717, Nov. 2004.

[20] Z. S. Zhang and A. S. Acampora, "A heuristic wavelength assignment algorithm for multihop WDM networks with wavelength routing and wavelength re-use," *IEEE/ACM Trans. Networking*, vol. 3, no. 3, pp. 281–288, June 1995.

[21] Y. Zhang, O. Yang, and H. Liu, "A Lagrangian relaxation and subgradient framework for the routing and wavelength assignment problem in WDM networks," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 9, pp. 1752–1765, Nov. 2004.

Guoliang (Larry) Xue (SM'99) received the BS degree (1981) in mathematics and the MS degree (1984) in operations research from Qufu Teachers University, Qufu, China, and the Ph.D. degree (1991) in computer science from the University of Minnesota, Minneapolis, USA. He is a Full Professor in the Department of Computer Science and Engineering at Arizona State University. He has held previous positions at Qufu Teachers University (Lecturer, 1984-1987), the Army High Performance Computing Research Center (Postdoctoral Research Fellow, 1991-1993), and the University of Vermont (Assistant Professor, 1993-1999; Associate Professor, 1999-2001). His research interests include efficient algorithms for optimization problems in networking, with applications to fault tolerance, robustness, and privacy issues in networks ranging from WDM optical networks to wireless ad hoc and sensor networks.

He has published over 140 papers in these areas, with many papers appearing in prestigious conferences and journals such as ACM Mobihoc, IEEE Infocom, SIAM/ACM SODA, *IEEE/ACM Transactions on Networking*, *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Communications*, *IEEE Transactions on Computers*, *SIAM Journal on Computing*, and *SIAM Journal on Optimization*. His research has been continuously supported by federal agencies, including NSF and ARO. He received the Graduate School Doctoral Dissertation Fellowship from the University of Minnesota in 1990, a Third Prize from the Ministry of Education of P.R. China in 1991, an NSF Research Initiation Award in 1994, and an NSF-ITR Award in 2003.

He is an Editor of *Computer Networks* (COMNET), an Editor of *IEEE Network*, and an Associate Editor of the *Journal of Global Optimization*. He has served on the executive/program committees of many IEEE conferences, including Infocom, Secon, ICC, Globecom, and QShine. He served as the General Chair of the IEEE International Performance, Computing, and Communications Conference in 2005, and will serve as a TPC co-chair of the IEEE Globecom'2006 Symposium on Wireless Ad Hoc and Sensor Networks, as well as a TPC co-chair of the IEEE ICC'2007 Symposium on Wireless Ad Hoc and Sensor Networks. He also serves on many NSF grant panels and is a reviewer for NSERC (Canada).

Weiyi Zhang (S'02) received the B.E. and M.E. degrees from Southeast University, China, in 1999 and 2002 respectively. Currently he is a Ph.D student in the Department of Computer Science and Engineering at Arizona State University. His research interests include reliable communication in networking, protection and restoration in WDM networks, and QoS provisioning in communication networks.

Jian Tang received the B.E. degree (1998) and M.E. degree (2001) from Beijing University of Posts and Telecommunications, and the Ph.D degree in Computer Science from Arizona State University (2006). His research interest is in the area of networking with emphases on routing, scheduling, cross-layer design and QoS provisioning in communication networks.

Krishnaiyan Thulasiraman (F'90) received the Bachelor's degree (1963) and Master's degree (1965) in electrical engineering from the University of Madras, India, and the Ph.D degree (1968) in electrical engineering from IIT, Madras, India. He holds the Hitachi Chair and is Professor in the School of Computer Science at the University of Oklahoma, Norman, where he has been since 1994. Prior to joining the University of Oklahoma, Thulasiraman was professor (1981-1994) and chair (1993-1994) of the ECE Department in Concordia University, Montreal. He was on the faculty in the EE and CS departments of the IITM during 1965-1981.

Dr. Thulasiraman's research interests have been in graph theory, combinatorial optimization, algorithms and applications in a variety of areas in CS and EE: electrical networks, VLSI physical design, systems level testing, communication protocol testing, parallel/distributed computing, telecommunication network planning, fault tolerance in optical networks, interconnection

networks etc. He has published more than 100 papers in archival journals, coauthored with M. N. S. Swamy two text books, *Graphs, Networks, and Algorithms* (1981) and *Graphs: Theory and Algorithms* (1992), both published by Wiley Inter-Science, and authored two chapters in the *Handbook of Circuits and Filters* (CRC and IEEE, 1995) and a chapter on "Graphs and Vector Spaces" for the *Handbook of Graph Theory and Applications* (CRC Press,2003).

Dr. Thulasiraman has received several awards and honors: Endowed Gopalakrishnan Chair Professorship in CS at IIT, Madras (Summer 2005), elected member of the European Academy of Sciences (2002), IEEE CAS Society Golden Jubilee Medal (1999), Fellow of the IEEE (1990) and Senior Research Fellowship of the Japan Society for Promotion of Science (1988). He has held visiting positions at the Tokyo Institute of Technology, University of Karlsruhe, University of Illinois at Urbana-Champaign, and Chuo University, Tokyo.

Dr. Thulasiraman has been Vice President (Administration) of the IEEE CAS Society (1998, 1999), Technical Program Chair of ISCAS (1993, 1999), Deputy Editor-in-Chief of the *IEEE Transactions on Circuits and Systems I* (2004-2005), Co-Guest Editor of a special issue on "Computational Graph Theory: Algorithms and Applications" (*IEEE Transactions on CAS*, March 1988), Associate Editor of the *IEEE Transactions on CAS* (1989-91, 1999-2001), and Founding Regional Editor of the *Journal of Circuits, Systems, and Computers*, and an editor of the AKCE *International Journal of Graphs and Combinatorics*. Recently, he founded the Technical Committee on "Graph theory and Computing" of the IEEE CAS Society.