Routing and Wavelength Assignment of Static Manycast Demands Over All-Optical Wavelength-Routed WDM Networks

Neal Charbonneau and Vinod M. Vokkarane

***Abstract*—In this paper we present the static manycast routing and wavelength assignment (MA-RWA) problem along with heuristics and an integer linear program (ILP) to solve it. Manycast is a point-to-multipoint communication paradigm with applications in e-Science, Grid, and cloud computing. A manycast request specifies a candidate set of destinations, of which a subset must be reached. To solve MA-RWA, a light-tree must be assigned to each manycast request in a static set such that the number of wavelengths required is minimized. We present two heuristics, the shortest path heuristic (SPT) and the lambda path heuristic (LPH), a tabu search metaheuristic (TS), and an ILP formulation. We show that TS provides results close to the optimal solution (from the ILP) for small networks. We then show that TS provides a 10% improvement over LPH and a 30%–40% improvement over SPT for various realistic networks.**

***Index Terms*—Manycast; Multicast; Heuristics; ILP; Tabu search; WDM; Wavelength routing; RWA.**

I. Introduction

Optical wavelength-routed WDM [1] networks will be an important technology to provide high bandwidth and services to the next-generation Internet and applications. In WDM networks, each fiber is partitioned into a number of wavelengths that can each transmit data simultaneously. This allows each fiber to provide data transmission rates of terabits per second. An optical WDM network consists of fibers connected by switches, or optical cross connects (OXCs). In wavelength-routed optical networks, when a connection request arrives at the network, a resource reservation process must find a route over the physical topology and also assign a free wavelength to the incoming request before data transmission begins. This is known as the *routing and wavelength assignment* (RWA) problem [2]. The combination of a route and wavelength is known as a *lightpath* [3]. In single-hop or all-optical WDM networks, the signal is transmitted all-optically from the source node to the destination node. In the absence of wavelength converters (which are expensive), a lightpath in an all-optical WDM network must use the same wavelength on each link along the end-to-end path. This is known as the *wavelength continuity constraint*.

Three traffic models are usually used to describe requests over wavelength-routed networks: static, incremental, and dynamic [4]. A static traffic model gives all the traffic demands between source and destinations ahead of time. A traffic demand matrix is given and the goal is typically to find RWA for all the demands, such that it minimizes the overall cost (e.g., using the least number of transmitters/receivers). The connections are assumed to be held for long periods of time. Traffic demands are incremental if, in addition to the static demands, there are new demands arriving over time. Dynamic traffic requests arrive according to some stochastic process and they are also released after some finite amount of time. When dynamic traffic is considered, the number of transmitters and receivers is fixed and the goal is to minimize request blocking. A request is said to be blocked if there are not enough resources available to route it.

Traditionally, communication in a network is unicast, where a single source sends data to a single destination. In this work, we consider a communication paradigm called manycast [5–7]. Given a network, $G=(V,E)$, a unicast connection request can be defined as a two-tuple, (s,d), where $s \in V$ is the source node and $d \in V$ is the destination. We can define a manycast request as a three-tuple, (s,D_c,k), where $s \in V$ is the source, $D_c \subseteq V$ is the candidate destination set, and k

Manuscript received February 18, 2010; revised May 2, 2010; accepted May 16, 2010; published June 14, 2010 (Doc. ID 124021).

The authors are with the Department of Computer and Information Science, University of Massachusetts, Dartmouth, Massachusetts, USA (e-mail: vvokkarane@ieee.org).

Digital Object Identifier 10.1364/JOCN.2.000427

1943-0620/10/070427-14/$15.00

$\leq |D_c|$ is the number of nodes necessary to reach out of D_c. This means that the source node will send data, simultaneously, to some subset of size k of the candidate destination set. This is a generalization of the multicast communication paradigm. In multicast, we are given a source node and a set of destinations $D \subseteq V$. In the multicast problem, the source communicates with all of D, simultaneously. Given our definition of manycast, if we have $k = |D_c|$, then the manycast request becomes a multicast request. Since we can define multicast as a specific instance of manycast, we consider manycast a generalization of multicast. It is not, however, a replacement for multicast. We will motivate the use of manycast shortly.

Manycast and multicast are related in that they both require setting up a tree in the network instead of a path so that the source can communicate with multiple destinations. A unicast request is supported by the use of lightpaths, as previously discussed. To efficiently support manycast or multicast requests, the network must create *light-trees* [8]. A light-tree is a generalization of a lightpath that starts at the source node of a multicast or manycast request and reaches all of its destinations all-optically by possibly branching (splitting the signal) at intermediate nodes. The problem of finding the optimal route for a light-tree is equivalent to finding the minimal Steiner tree, which is known to be NP-complete [9], although efficient approximations exist. In order to support light-trees, the nodes in an optical network must be able to split an incoming signal to multiple output ports. This can be accomplished by using switches based on splitter and delivery (SaD) [10,11]. These switches are known as multicast-capable OXCs (MC-OXCs). Multicast-capable reconfigurable optical add–drop multiplexers (ROADMs) may also be used [12]. We discuss these switches in more detail in Section II.

The key difference between multicast and manycast is that in multicast, the destinations are specified ahead of time, whereas in manycast the destinations must be chosen (based on the state of the network, for example). In other words, in multicast, there is no flexibility in choosing which destinations to transmit data to, whereas in manycast we can choose which destinations to transmit to out of a larger candidate set. To solve the multicast problem, we have a single set of nodes that are used to find the optimal Steiner tree connecting the source and destination set. In manycast, we must choose a subset of k nodes; this means that there are a total of $\binom{|D_c|}{k}$ combinations of nodes to use in the creation of a Steiner tree. Of these $\binom{|D_c|}{k}$ different Steiner trees, one of them has the lowest cost. The manycast problem, though similar to multicast, requires a new set of solution techniques. As we will show later, turning a manycast request into a multicast request at the source results in poor performance.

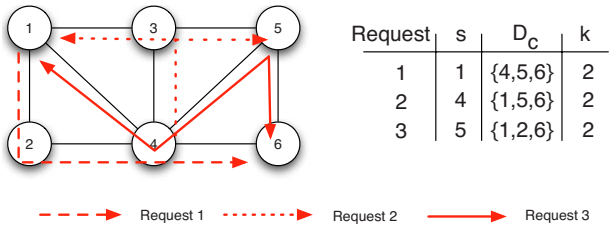
In this work we will consider the static manycast routing and wavelength assignment (MA-RWA) problem. In this problem we are given a set of manycast requests and for each request we must assign a light-tree. The objective is to minimize the number of wavelengths required to satisfy all the manycast requests. Manycast is a powerful communication framework that is important for next-generation applications [13]. Manycast is particularly useful in cloud/utility computing, Grid computing, and e-Science. In all of the above scenarios, we are typically dealing with large amounts of data that must be transferred. In these environments, a service provider may host a number of servers that provide the same service. For example, there may be a number of servers that can be used simultaneously for distributed data storage (and retrieval). There may be a number of servers that can process computational tasks in parallel. The client will want to use some subset of these available resources to execute the storage or computation task. The subset may correspond to the lowest cost servers or servers with the lowest latency. As a specific example, consider parallel content distribution for an e-Science scenario. e-Science experiments typically produce large amounts of data [14] that may then be stored at multiple locations for distributed backup or so that multiple research labs can then process it locally. We can use manycast to choose some subset of these locations (e.g., the lowest-cost storage clusters) to send this data in parallel along a light-tree set up by the network. This problem can also be solved using multicast instead of manycast, but there are several disadvantages. Instead of using manycast, where the network would choose the subset of destinations for us, we can use multicast by choosing k destinations at the source. The first disadvantage is from a user perspective. The nodes selected at the source may be heavily loaded or may provide a slower transfer rate than other nodes in the candidate set. If some/all of the preselected nodes are not available, then the request may be blocked, even though there are other nodes available from the candidate set. From the network operator perspective, manycast would allow the network to optimize its resources, for example, by load balancing. Manycast allows the freedom to choose different nodes depending on the state of the network, leading to better performance for user applications and better utilization for the network. Another option to providing a manycast service is to provide manycast at the application layer supported by unicast at the optical layer. The disadvantage to application-layer manycast is that using unicast to support point-to-multipoint communication at the optical layer wastes resources. The authors in [15] show the benefit

of supporting *multicast* directly at the optical layer instead of higher layers (IP in this case). Supporting multicast at the optical layer can significantly reduce the number of wavelengths required. These results are applicable to manycast as well, and likely even more so because the choice of nodes will also impact the solution. The higher layers may select nodes from the candidate set that result in suboptimal lightpaths at the optical layer. Supporting manycast at the optical layer allows the network to set up a light tree to the "best" nodes according to the network state, which minimizes the number of wavelengths required. There is much work that investigates multicast at the optical layer (e.g., [16–24]) because of the efficiency and lower resource utilization; likewise it is most cost-effective to support manycast at the optical layer.

This is the first paper, to our knowledge, that investigates manycast over wavelength-routed networks. Manycast is also known as quorumcast or the k-Steiner problem. It was proposed in [5,6]. Since then, a number of quorumcast routing algorithms have been proposed [5,25–27]. Finding a minimum-cost tree for a manycast request is NP-hard [28]. Manycast is also related to the k-MST problem [28] in that a ρ-approximation algorithm for k-MST leads to a 2ρ-approximation algorithm for manycast. These works focus on finding minimum-cost trees for an individual "request." In this work we have a set of static requests and we must also perform routing and wavelength assignment.

Manycast has also been proposed over optical burst-switched (OBS) networks [7,29–31]. The main challenge for manycast over OBS is providing reliability despite random contentions. These works focus on dynamic traffic and distributed routing algorithms or unicast routing algorithms to provide reliable manycast for OBS. These approaches typically do not set up a route tree for each request nor do they consider wavelength assignment (they assume full wavelength conversion is available). The authors in [32] propose an ILP (used as a basis for our ILP) and several heuristics for solving multiresource manycast in mesh networks. In manycast we may consider that each node provides a single resource, so to reach k resources we must reach k nodes. Multiresource manycast generalizes manycast by allowing nodes to provide more than a single resource. This work does not consider static requests or wavelength assignment. Recently, an anycast RWA algorithm was proposed for wavelength-routed networks [33]. Anycast is a specific instance of manycast where $k=1<m$.

An example of static manycast routing and wavelength can be seen in Fig. 1. The table on the right in the figure shows the static manycast request set. For each request, we must find a light-tree from the source to any two of the three destinations in the candidate set, D_c. The (optimal) RWA is shown over the six-node network on the left. The source of each request reaches exactly two of the destinations in the candidate set D_c. For example, consider request 2. It is sourced at node 4, uses link 4–3, then splits the signal at node 3, reaching nodes 1 and 5. Both of these nodes are in its candidate set and since $k=2$, this light-tree satisfies the manycast request. In this example, only a single wavelength is required to route all three requests over the network. We discussed the wavelength continuity constraint that specifies that a light-tree must use the same wavelength on all links. The *wavelength clash constraint* specifies that any given wavelength can be used at most once on each link. If two of the requests shared a link, then they would have to use two different wavelengths, but since no requests overlap on any given link, they can each use the same wavelength.



Request	s	D_c	k
1	1	{4,5,6}	2
2	4	{1,5,6}	2
3	5	{1,2,6}	2

Fig. 1. (Color online) Static manycast RWA example. The requests are given in the table on the right. This RWA requires only a single wavelength.

The contributions of this paper are the introduction of manycast RWA for wavelength-routed optical networks, an ILP formulation of the problem, and three heuristics to solve it. The paper is organized as follows. Section II presents the formal definition of the MA-RWA problem, and Section III presents our ILP formulation. We discuss our heuristics in Section IV. Our solution techniques are evaluated in Section V, and Section VI concludes the paper.

II. Problem Definition

Given a network, $G=(V,E)$, a manycast request is defined as $R=(s,D_c,k)$, where $s\in V$ is the source, $D_c\subseteq V-\{s\}$ is the candidate destination set, and $k\leq|D_c|$ is the number of nodes necessary to reach out of D_c. We must find a light-tree (combination of a route tree and lightpath) that starts at node s and reaches at least k nodes out of D_c. We assume that each manycast request requires one wavelength and it can use only one wavelength (i.e., a source node cannot transmit separate wavelengths to reach different destinations, it must use only a single wavelength to reach all destinations). The static MA-RWA problem can be defined as follows.

Definition *MA-RWA*(G,M): Given a network $G=(V,E)$ and a set of manycast requests, $M=\{R_1,R_2,\ldots,R_n\}$, the solution must assign a route tree and a wavelength to each request, R_i, in such a way that the number of wavelengths required is minimized while satisfying the wavelength continuity and wavelength clash constraints. Note, we are minimizing the network-wide wavelength count, not the maximum number of wavelengths on any link.

The MA-RWA problem can be computed offline because we are given the set of requests ahead of time. This is in contrast to the dynamic RWA problem where computations occur online as each individual request arrives.

A. Network Assumptions

We consider all-optical networks without wavelength converters, which implies that once the signal enters the network, it must use the same wavelength on all links (wavelength continuity constraint). We also assume that a wavelength can be used *at most* once in either direction on any link (wavelength clash constraint). We assume all nodes in the network are able to split an incoming signal to any number of output ports. As discussed previously, these types of switches are known as multicast-capable optical cross connects. We show the switch architecture of the MC-OXCs we use in Fig. 2. The splitter-and-delivery components of the MC-OXC are shown in Fig. 3, proposed in [10]. The SaD switch consists of N power splitters, N^2 2×1 optical gates that are used to reduce crosstalk, and N^2 2×1 photonic switches as shown in Fig. 3. These SaD switches allow an incoming signal to be split to any number of output ports.

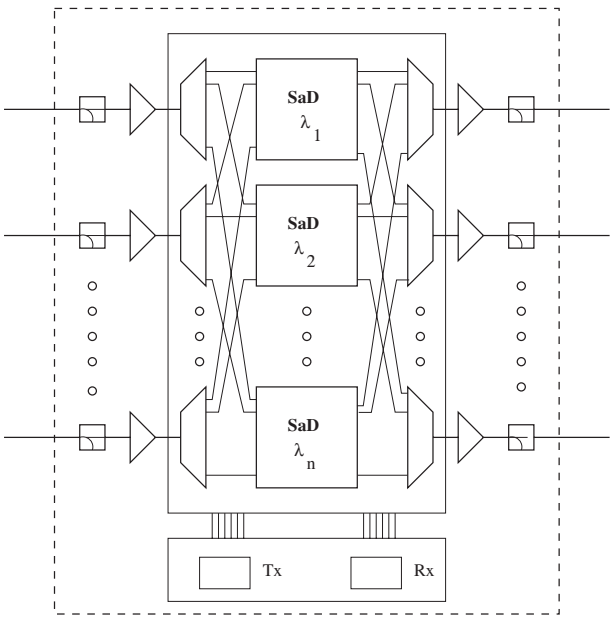


Fig. 2. MC-OXC based on splitter-and-delivery architecture.

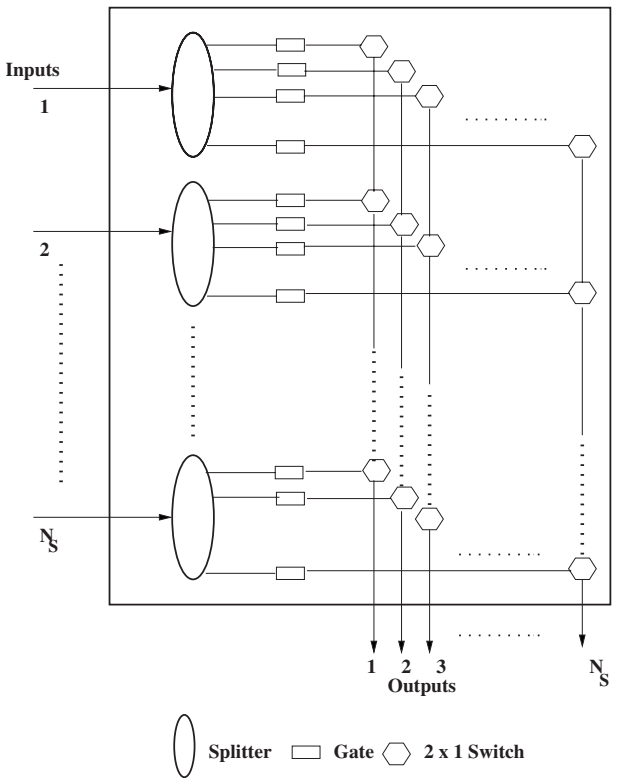


Fig. 3. An $N\times N$ SaD switch.

While we assume MC-OXCs in this paper, our algorithms are not limited to optical cross connects. ROADMs are expected to play an integral role in all-optical networks. Our algorithms work with either type of switching node. The feasibility of multicast-capable ROADMs (MC-ROADMs) was recently investigated in [12]. We refer the reader to this paper for details about the MC-ROADM architecture.

We do not consider physical-layer impairments in this work. In all-optical WDM systems noise accumulates from physical-layer impairments such as crosstalk, amplified spontaneous emission (ASE) noise, and nonlinear impairments such as four-wave mixing, cross-phase modulation, stimulated Brillouin scattering, and stimulated Raman scattering. To counter this, impairment-aware routing can be used to ensure that the signal-to-noise ratio is at acceptable levels when the signal reaches the destination. There has recently been significant work in impairment-aware routing [34]. While it is important to incorporate physical-layer impairments, especially since we split the optical signal, the primary objective of this paper is to define the static manycast RWA problem and to provide initial solutions. We therefore consider impairment-aware routing out of the scope of this work. We will discuss our future work in this area in Section VI.

B. Complexity

Routing and wavelength assignment problems to minimize the number of wavelengths are equivalent to the graph coloring problem and are therefore NP-hard [35]. It has also been shown recently that some types of RWA problems cannot be approximated within a constant factor [36] unless $P=NP$. Because of the complexity of the problem, we will focus on heuristic approaches to find suboptimal solutions.

III. ILP Formulation

We first formulate an ILP for static MA-RWA to find the optimal solution. The ILP is not practical for large networks, so we introduce heuristics in the following section. We can still use the ILP for smaller networks to compare the results of our heuristics to the optimal solutions. The objective is to minimize the number of wavelengths used. Note that this is different from minimizing the maximum number of wavelengths on any link. We use i,j to denote links, m to denote the mth manycast request, and w to denote wavelengths. The ILP will solve for the following variables:

- $y_{i,j}^{m,w}$: 1 if wavelength w is used on link (i,j) for manycast request m, 0 otherwise.
- U_i^m: order node i was added to the tree for manycast request m. This prevents loops.
- *maxIndex*: the largest wavelength index in use.
- $C^{m,w}$: 1 if wavelength w is used by request m, 0 otherwise.

Objective Function:

$$minimize: \quad maxIndex$$

Subject to:

$$maxIndex \geq C^{m,w} \times w \quad \forall\, m,w, \tag{1}$$

$$\sum_m y_{i,j}^{m,w} + y_{j,i}^{m,w} \leq 1 \quad \forall\, i,j,w, \tag{2}$$

$$\sum_i \sum_{j \in D_{mc}} \sum_w y_{i,j}^{m,w} \geq k_m \quad \forall\, m, \tag{3}$$

$$\sum_j \sum_w y_{s_m,j}^{m,w} \geq 1 \quad \forall\, m, \tag{4}$$

$$\sum_j \sum_w y_{j,s_m}^{m,w} = 0 \quad \forall\, m, \tag{5}$$

$$\sum_j \sum_w y_{j,i}^{m,w} \leq 1 \quad \forall\, m, i \neq s_m, \tag{6}$$

$$\sum_j y_{i,j}^{m,w} - |V| \sum_j y_{j,i}^{m,w} \leq 0 \quad \forall\, m,w,i \neq s_m, \tag{7}$$

$$\sum_j y_{j,i}^{m,w} - \sum_j y_{i,j}^{m,w} \leq 0 \quad \forall\, m,w,i \neq D_{mc}, \tag{8}$$

$$U_i^m - U_j^m + |V| y_{i,j}^{m,w} \leq |V| - 1 \quad \forall\, m,w,i,j, \tag{9}$$

$$\sum_w C^{m,w} = 1 \quad \forall\, m, \tag{10}$$

$$y_{i,j}^{m,w} + y_{j,i}^{m,w} \leq C^{m,w} \quad \forall\, m,w,i,j (j > i). \tag{11}$$

We will now describe each of the constraints. When we refer to wavelengths in the following explanation, we are referring to wavelengths on the light-tree that is constructed. For example, when we say that there may be no outgoing wavelengths at a node, we mean that when the light-tree is created, that particular node on the tree cannot have any outgoing wavelengths that are part of the tree. Constraint (1) is used to keep track of the maximum wavelength index used. Constraint (2) is used to enforce the wavelength clash constraint. Constraints (3)–(9) are used to build each route tree. Constraint (3) specifies that at least k destinations must be reached for each request. Constraint (4) specifies that there must be at least one outgoing wavelength at the source, and constraint (5) specifies that there can be no incoming wavelengths to the source. Constraint (6) specifies that all nodes (except the source) can have at most one incoming wavelength, while constraint (7) specifies that a node can have outgoing wavelengths only if it has incoming wavelengths. Constraint (8) specifies that nodes not in the candidate destination set that have an incoming wavelength must have at least one outgoing wavelength. Constraint (9) is used to prevent formation of routing loops. Constraints (10) and (11) are used to enforce the wavelength continuity constraint by ensuring each tree uses exactly one wavelength and all links on the tree use that wavelength.

Algorithm 1 Shortest Path Heuristic for Static MA-RWA

```
1: sort_desc(M)
2: for all m in M do
3:    D=kShortest(D_mc)
4:    T=(V',E') s.t. V'={s_m}, E'=φ
5:    path=min{SP(s_m,u)} u ∈ D
6:    Update(T,path)
7:    copy=1
8:    while copy<k do
9:       path=min{SP(u_1,u_2)} u_1 ∈ V', u_2 ∈ D−V'
10:      Update(T,path)
11:      copy=copy+1
12:   end while
13:   T.cost=Σ_{i,j∈E'} c_{i,j}
14:   FirstFit(G,T)
15: end for
```

IV. Heuristics

In this section we discuss two simple heuristics for solving static MA-RWA. We then discuss a tabu search meta-heuristic with longer run times but better results. Due to the complexity of the ILP, it is only feasible to solve the ILP formulation for small problem instances. We introduce the heuristics so that large instances of the problem can be solved. These heuristics provide suboptimal solutions but reasonable run times for realistic problem instances.

Typical solution techniques for routing and wavelength assignments either consider both problems, routing and assigning wavelengths, jointly or independently. In this work, our heuristics solve the problem jointly.

A. Shortest Path Heuristic (SPT)

The simplest manycast heuristic is called the shortest path heuristic, SPT, shown in Algorithm 1. When discussing the manycast request tuple for some request i, we denote it as (s_i, D_{ic}, k_i). First, the request set is sorted in decreasing size of k_i for request i. For each individual request, in sorted order, SPT chooses k out of D_c that are closest to the source according to the shortest paths. It then uses the minimum path heuristic (MPH) [37] to create a Steiner tree to these destinations. Once the tree is created, the wavelength is assigned using the first-fit policy [4], which assigns the lowest available wavelength index to the tree. SPT essentially treats the request as a multicast request since it only considers the closest k nodes instead of all possible nodes.

Line 3 finds the k closest nodes in D_c for request m and stores them in D. The remainder of the *for* loop is essentially the MPH heuristic. It creates a Steiner tree by building it incrementally. The function *SP* on line 5 finds the shortest path between two nodes. The heuristic starts by selecting the node in D that has the shortest path from s. It uses this path as the start of the tree. It then looks for the next node in D with the minimum-cost shortest path from any node on the partially built tree and adds that node to the tree using the shortest path. It continues adding nodes in this manner until k nodes are part of the tree. The function *Update* on line 6 in the pseudocode is used to add edges and nodes on the shortest path to the route tree T.

1) Complexity: Assuming shortest paths are precomputed in $O(|V|^3)$, the SPT heuristic runs in $O(kV|D_{mc}|)$ for each request. The k is from the loop and the $V \times D_{mc}$ is the time to scan for the minimum-value shortest path based on the nodes currently in the tree (maximum of $V-1$) and remaining nodes (maximum of $|D_{mc}|$). The $|D_{mc}|$ term is dependent on each request. Different requests will have different request sizes. To get an upper bound when considering the whole set, we will replace $|D_{mc}|$ with V. The heuristic is run once for $|M|$ requests, so the total run time is $O(|M|kV^2)$.

B. Lambda Path Heuristic (LPH)

We will now describe our next heuristic, the LPH heuristic. It begins in the same way as SPT, by sorting the static request set according to k of each request. With the requests sorted, LPH processes each request individually. To satisfy each individual request, we use a modified version of the improved path heuristic (IMP) [5] for quorumcast. We modify it to include wavelength assignment, an additional constraint to minimize wavelengths required, load balancing, and to also iterate over more Steiner trees. The heuristic is shown in Algorithm 2. Before describing how it works, we will define the functions that it uses. First, the *SP* function finds the shortest path between the two nodes specified as parameters. The *Update* function adds a path (edges and links) to the specified tree T. The *increasesWL* function determines whether assigning a wavelength to a tree, using first-fit, would require an increase in the wavelength count given the wavelengths currently used for the previous requests. Lastly, the *updateWeights* function is used for load balancing. It updates the weight of each link according to $\alpha+(1-\alpha)\times c/c_{max}$, where c is the current number of wavelengths on the link, c_{max} is the number of wavelengths on the most congested link, and $0 \leq \alpha \leq 1$. Depending on the value of α, this helps achieve a degree of load balancing in the network.

To satisfy a manycast request, there are $\binom{|D_c|}{k}$ possible combinations of nodes that can be used to create Steiner trees. This heuristic works by creating just $|D_c|$ Steiner trees, where each tree is created using the same MPH heuristic used by SPT. $|D_c|$ Steiner trees are created by forcing selection of the first node in line 7 when building the tree. During the first iteration, the shortest-path node is selected, in the next iteration the next shortest-path node is selected, and so on. Selecting a different start node each time makes it likely that a different tree will be created each iteration. Each time a node is selected to be the initial node it is added to D. An iteration terminates when $D=D_c$. This ensures that all nodes in D_c are in at least one Steiner tree and that multiple trees are generated. The goal of this is to find a good Steiner tree without having to try all $\binom{|D_c|}{k}$ combinations of nodes. After the generation of each tree, line 17 checks to see whether assigning a wavelength according to first-fit would result in an increase in wavelength count required. Once the trees have been generated, the heuristic chooses the minimum-cost tree that requires no increase in wavelength count using first-fit wavelength

Algorithm 2 Lambda Path Heuristic for Static MA-RWA

```
sort_desc(M)
for all m in M do
  D={ }
  allTrees=list( )
  while D_mc−D ≠ φ do
    T=(V′,E′) s.t. V′={s_m}, E′=φ
    path=min{SP(s_m,u)} u ∈ D_mc−D
    Update(T,path)
    D=∪(u)
    copy=1
    while copy<k do
      path=min{SP(u_1,u_2)} u_1 ∈ V′, u_2 ∈ D_mc−V′
      Update(T,path)
      copy=copy+1
    end while
    T.cost=Σ_{i,j∈E′} c_{i,j}
    T.newWL=increasesWL(G,T)
    allTrees.append(T)
  end while
  T=min(allTrees)
  FirstFit(G,T)
  updateWeights(α,1−α)
end for
```

assignment. If there is no such tree, just the minimum-cost tree is chosen. The cost is defined as the tree with the least number of links. The reasoning behind this cost function is that using smaller trees will leave more resources for future requests. After the tree is assigned the wavelength, the costs of the links are updated.

The main difference between SPT and LPH is that LPH considers *multiple* Steiner trees by including different nodes while SPT makes a *single* decision on which nodes to include.

1) Example: We will now provide an example of the LPH heuristic for a single manycast request. The request and network (NSFnet) are given in Fig. 4(a). The request, (0, {2,4,13}, 2), means that node 0 is the source; nodes 2, 4, and 13 are the candidate destinations; and two out of the three destinations must be reached. LPH will iterate three times ($|D_c|$), each time choosing a different starting node to form a tree. On the first iteration, node 2 is chosen since it is the shortest path distance from 0 ($0 \rightarrow 2$). Node 4 is added by concatenating ($2 \rightarrow 1 \rightarrow 3 \rightarrow 4$) to the tree. This results in the tree seen in Fig. 4(b), which covers nodes 2 and 4. In the next iteration, node 4 is chosen to start the tree ($0 \rightarrow 1 \rightarrow 3 \rightarrow 4$). A new branch can then be added to create a tree reaching node 2 ($1 \rightarrow 2$), as seen in Fig. 4(c). Last, node 13 is chosen first with path ($0 \rightarrow 7 \rightarrow 8 \rightarrow 13$). The tree can then be modified to branch at node 7, reaching node 4 ($7 \rightarrow 6 \rightarrow 4$), as seen in Fig. 4(d). The iterations are now complete since every node in D_c has been used as a starting node. The

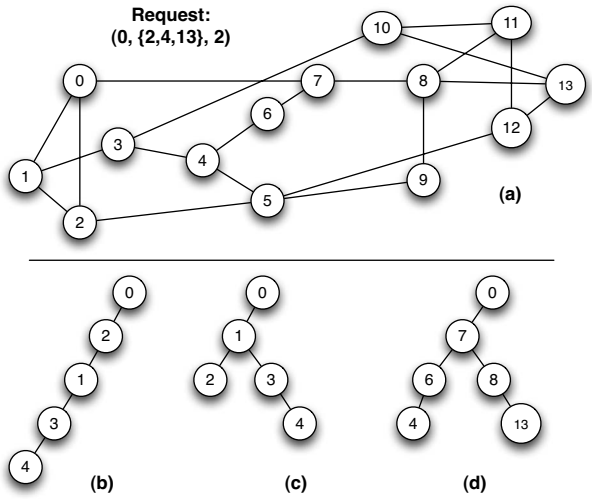


Fig. 4. LPH illustration with manycast request (0, {2,4,13}, 2). Given the network and request in (a), LPH generates the Steiner trees shown in (b)–(d).

heuristic is now able to choose the best tree. It will choose the lowest-cost tree that does not require an increment in the number of wavelengths used in the network. If this is not available, the lowest-cost tree will be chosen. The same heuristic is run for all requests.

2) Complexity: The heuristic runs in $O(kV|D_{mc}|^2 + |V|^3)$ time for each individual request. The reasoning is similar to the SPT heuristic except now there is an additional outer loop that runs D_{mc} times, hence the squared term. In addition to this, the link weights are modified at the end when the tree is selected, so the shortest paths must be recomputed. The heuristic is run once for each of the $|M|$ requests for a total run time of $O(|M|kV^3)$. Again we use V as an upper bound for $|D_{mc}|$, as in SPT.

C. Tabu Search (TS)

We now discuss our tabu search meta-heuristic. Tabu search is often used for combinatorial optimization problems [38]. It explores the solution space for a number of iterations or until some other stopping criterion is met. An initial solution is first generated either randomly or by using another heuristic. Given the current solution, a neighborhood set of solutions is generated by performing simple *moves* from the current solution. The best solution from the neighborhood is then chosen as the current solution and the process continues. To avoid getting trapped in local minima, a *tabu list* is maintained. The tabu list records moves that were used to generate selected solutions. These moves cannot be performed again as long as they are on the list. Tabu search meta-heuristics also use diversification and intensification steps. Diversification is typically used when the best solution has not im-

proved after a number of iterations. Diversification usually generates a new solution elsewhere in the search space and the tabu search restarts there. Intensification can be used to perform a more thorough search of the neighborhood of a good solution that has been found. Details of tabu search can be found in [38]. There is no proof of convergence to an optimal solution, but tabu search often works quite well in practice. Tabu search has been used to solve routing and wavelength assignment problems; see [39–41], among others.

The tabu search heuristic that we propose uses LPH. We will refer to our tabu search heuristic as TS. Given a set of manycast requests, LPH orders them and then iterates over the set assigning a route tree and wavelength to each request in order. The ordering of requests has a significant impact on the solution. There must exist some optimal ordering that requires the least number of wavelengths required. TS attempts to find this optimal ordering by searching different permutations of orderings.

Instead of defining the requests as a set, for purposes of TS we will define it as a sequence, since ordering is important. Given a sequence, LPH is used to assign route trees and wavelengths to each request in the order specified by the sequence. This ordering of requests and their route tree and wavelength assignments constitutes a solution. The cost of a solution is defined by the number of wavelengths required to satisfy all the requests. The objective is to find a solution that requires the minimum number of wavelengths.

We will describe TS by describing each of the individual parts: the initial solution, the neighborhood set, tabu list, diversification strategies, and intensification strategies.

Initial Solution: the initial solution is the sequence obtained by ordering the manycast request set according to the largest k_i value first (largest request first). Given this sequence, LPH is used to assign routes and wavelengths.

Neighborhood Set: to obtain the neighborhood set, we define the move operation that creates a neighbor given the current solution. The move operation simply swaps two requests, i and j, where $i \neq j$, in the sequence. If we performed all valid moves the neighborhood set size would be $|M| \times (|M|-1)/2$. Note, the move (i,j) is the same as (j,i). For large request sets, this neighborhood size is too large, so we perform a smaller number of random moves to generate the neighborhood. The number of moves we perform is proportionate to the actual neighborhood size. For example, we can specify that the neighborhood size be, say 6%, of the full neighborhood size. This generates random moves (without duplication) until a neighborhood size 6% of the full size is created. Once the neighborhood is generated, the least-cost neighbor is selected as the new current solution, subject to the tabu list.

Tabu List: The tabu list maintains a list of recently performed moves, where a move is the two indices, i and j, that were swapped. Note that $(i,j)=(j,i)$. A move cannot be chosen to create a new solution if it is already on the tabu list unless it meets an *aspiration criterion*. In this case, if the tabu move would result in a solution better than the current best solution, it is allowed. Items are kept on the tabu list for a specified number of iterations known as the *tabu tenure*. Once an item's tenure is up, it is removed from the list.

Diversification: This is typically used to explore other areas in the solution space. We use diversification if the solution has not improved after a number of iterations. Our diversification step randomly permutes the sequence of requests and removes all entries from the current tabu list.

Intensification: Instead of only recording the best and current solutions, TS records the top five solutions seen so far. These are then used for intensification while the TS is running. If there is no improvement after a number of iterations, diversification is used to generate a new solution. If, after a number of diversifications, we still have not improved our best solution, intensification is performed. During intensification, the entire neighborhood set is analyzed instead of just a random portion of it. When intensification is to be performed, the best solution is selected from the list of the top five. If this solution has already been "intensified," then the next solution is used, and so on. The intensification is also a depth-first search process. If a better solution is found upon intensification, that new solution is used for another intensification. Once no better solution is found, TS continues as normal with the best solution that was generated. If the list of best solutions contains multiple solutions with equal scores that have already been searched by intensification, only one is kept in the list and the remaining are discarded. Because they all have equal scores, we arbitrarily choose to keep the one with the lowest index in the list. This allows new solutions to be added to the list to be searched by the intensification process later.

We show the general flow of the algorithm in Fig. 5. The inputs to the algorithm are the number of iterations (controlling the first branch in the flow chart), the fraction of the neighborhood size to explore, the tabu tenure, the number of iterations before diversifications (controlling the "Diversify?" branch), and the number of diversifications before intensification (controlling the "Intensify?" branch). The blocks that generate partial or full neighborhoods use LPH after each move to generate the solution to add to the neighbor-

hood set. The blocks in the flow chart correspond to the steps we described above.

1) Justifications: The idea behind TS is for the diversification and normal iterations to find solutions that can be added to the list of best solutions. The list of best solutions may contain solutions from different areas of the search space due to diversification. The intensification step can then perform a depth-first search or a local search in those areas to improve the solution's score. This is the reason for using a list of the top five solutions instead of just the top solution. The list stores the best solutions from different areas of the search space that can be used during intensification.

In addition to the tabu search method described above, we tried another tabu search meta-heuristic that did not perform as well. The other tabu search started by generating k alternate trees for each manycast request. The tabu search would then search the solution space of different combinations of trees and used the largest-first graph coloring heuristic to assign wavelengths. The tabu search using the LPH heuristic performed better because it is more flexible. Using LPH allows load balancing in the network. It also allows selection of each tree to take into account the previously assigned wavelengths.

2) Example: We will describe an example of our move operation to generate a neighborhood set. Consider an initial set of manycast requests, $M=\{R_1,R_2,R_3,R_4\}$, where $R_i=(s_i,D_{ic},k_i)$. The requests are first sorted in descending order according to k_i. Note, this is equivalent to sorting according to D_{ic} because, as we discuss later, we set $k_i=\lceil D_{ic}/2\rceil$. Let the sorted sequence now be $M'=(R_2,R_1,R_4,R_3)$. Given this sequence, the LPH heuristic is run on the requests in order to generate a solution.

During the first iteration, some percentage of the neighborhood would be explored. The entire neighborhood consists of all possible combinations generated by swapping two elements. In this simple example, we can generate six solutions by swapping: (R_1,R_2), (R_1,R_3), (R_1,R_4), (R_2,R_3), (R_2,R_4), and (R_3,R_4). With large $|M|$ values, this is too large, so instead a series of random moves are generated. Let us assume that a 50% neighborhood size was specified. This may result in the moves (R_1,R_4), (R_2,R_3), and (R_2,R_4) being randomly generated. With these moves, the neighborhood set becomes $\{(R_4,R_2,R_3,R_1),(R_1,R_3,R_2,R_4),(R_1,R_4,R_3,R_2)\}$. Given this set, LPH is run on each sequence and the best one is chosen as the sequence to use in the next iteration (subject to the tabu list).

3) Complexity: In this subsection we discuss the complexity of TS in terms of the complexity of generating neighborhood solutions. The number of neighbors created during each iteration directly impacts the run time, since for each neighbor, LPH must be run. The LPH algorithm, runs in $O(|M|(k|V|^3))$ for each individual request. If the entire neighborhood space is searched each iteration, there are $O(|M|^2)$ neighbors generated, which would result in a run time of $O(|M|^3\times(k|V|^3))$ for each iteration. For large $|M|$, this is clearly not efficient, so we randomly generate small fractions of the full neighborhood set to keep the neighborhood space $O(|M|)$. Each iteration, therefore, takes $O(|M|^2\times(k|V|^3))$. TS is run for a number of iterations. We will see later in the paper that TS is significantly slower than LPH and SPT, but the results from TS are close to optimal results provided by the ILP (with much smaller run times than the ILP).

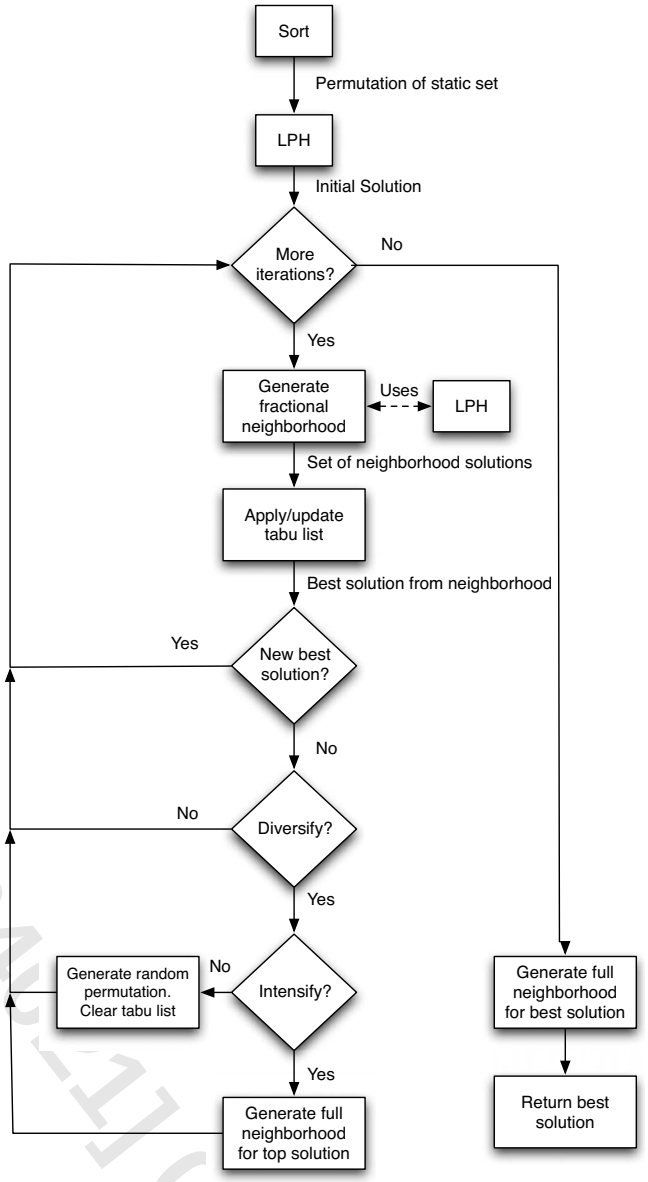


Fig. 5. Tabu search meta-heuristic flow chart.

V. Evaluation

We will evaluate the heuristics in two steps. First, we will compare the heuristics' results to the optimal

results provided by the ILP formulation. Because of the complexity of the ILP this is only possible for small request sizes. In addition to comparing the results with the ILP, we will also compare run times.

Next, we will compare only the heuristics on more realistic networks with larger request sizes. We ran extensive simulations on the AT&T network, the NSF network, the Italian WDM network, and a 24-node mesh network shown in Fig. 6. We use the link distances for calculating the average tree delay, but routing is done based on hop count.

A. ILP and Heuristics Comparison

This section compares the results of the heuristics with the optimal results of the ILP. Given the complexity of the ILP, we can only run it on small networks. We use the network shown in Fig. 7. We run the ILP and heuristics for request set sizes, $|M|$, of 10, 15, 20, and 25. For each request set size, we ran the simulations 20 times and plot the average value (along with the 95% confidence intervals for the number of wavelengths required). The source of each manycast request is uniformly distributed. The candidate destination size, (D_c), is either three or four (with equal probability) and $k=2$, for all requests. The α parameter to LPH is 0.8 (best value). We used CPLEX 12.0 to obtain results for the ILP. Both the ILP and heuristics were run on a machine with a 2.33 GHz Quad Core Xeon processor and 8 GB of RAM. The processor also has Hyper-Threading, so CPLEX was able to use eight threads while solving the ILP.

The number of wavelengths required by the heuristics and ILP is shown in Fig. 8(a). The figure shows that TS provides close to optimal results and significantly outperforms SPT.

We also plot the average run times of the different solution approaches in Fig. 8(b). We can observe the large increase in run times for the ILP as the request set gets large. Note the log scale of the y-axis. The LPH and SPT heuristics essentially finish instantly compared with the others. The run time for the ILP grows rapidly. With a request set size of 25 requests over a small 6-node network, the longest run time was over 30 h for the ILP compared with under 5 min for TS.

B. Heuristics

We will first present the results obtained for our heuristics, then go on to discuss how we selected the input parameters.

In Table I we compare TS, LPH, and SPT. The network characteristics are given in the table where V is the number of nodes, E is number of links, δ is average nodal degree, and τ is average delay per link (ms).

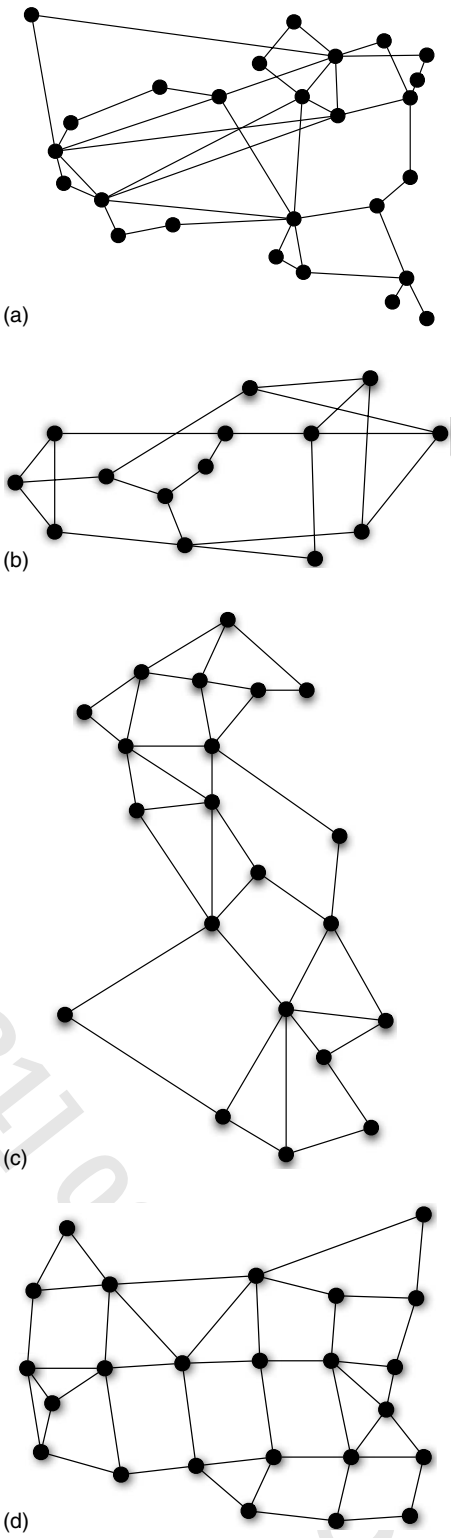


Fig. 6. Networks used for heuristic evaluation.

We generate a set of 150 requests. Other request set sizes provide similar patterns of results. The source node for each request is uniformly distributed over all nodes in the network. For each request m, the size of D_{mc} is uniformly distributed from $3,\ldots,D_{max}$ (a parameter representing the maximum candidate desti-

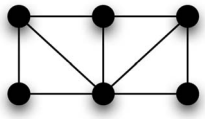


Fig. 7. Six-node network used for ILP evaluation.

nation size) and $k_m=[D_{mc}/2]$. The destination nodes are also uniformly distributed across the network for each request. We ran each heuristic with different maximum destination set sizes, D_{max}, and recorded the average number of wavelengths required, w_a, and average tree delay, d_a (ms). The average tree delay is defined as the average delay from the root to each destination node. The parameters for TS are as follows. There are 1000 iterations in total, the tabu tenure is 20, the fractional neighborhood search searches 6%, there are 25 iterations before diversification, and 2 diversifications before intensification (when no better solution has been found for these last two cases). For both LPH and TS $\alpha=0.8$ for load balancing. Most of these parameters were obtained empirically; this is discussed later in this section. Each data point is the average of 20 simulation runs. We calculated the confidence intervals but do not include them in the table. For TS and LPH, the confidence intervals were within 3% of the mean while they were slightly larger for SPT at around 5%, all with 95% confidence.

The table shows a significant decrease in the number of wavelengths required (w_a columns) between TS and SPT as well as a significant decrease between TS and LPH. TS reduces wavelengths required by between 30% and 40% compared with SPT. The greatest gains are in the Italian network while NSFnet has the smallest gains. TS also performed about 10% better than LPH.

Low delay is a requirement for many next-generation applications, so the heuristics must not significantly impact delay. Even though SPT results in a smaller average tree delay (d_a columns), the savings in wavelengths when using TS is significantly larger. The largest difference in delay between SPT and TS is around 1 ms. The average tree delays of TS and LPH were very similar, and TS was able to reduce wavelengths required by about 10% compared with LPH.

As expected, as the maximum destination set size increases, the number of wavelengths required also increases. The set size of 150 was chosen for demonstration purposes. We evaluated the heuristics on varying set sizes from 50 to 200 with similar results. We chose these four networks to represent realistic scenarios with varying nodal degrees. The networks represent backbone or long-haul networks for which a wavelength-routed WDM network is a good candidate.

The relative number of wavelengths required by TS and LPH are consistent across networks (e.g., NSFnet needs the most, followed by Italy, AT&T, and the 24-node network). This is a result of the characteristics of the networks. Networks with more nodes and higher nodal degrees will require fewer wavelengths because 1) the request size is the same, so 150 requests will require fewer wavelengths on networks with more nodes than networks with fewer nodes, and 2) a network with higher nodal degree has a better chance of finding alternate paths/trees for the requests, therefore reducing the maximum number of wavelengths required on a link.

We will now discuss how we selected the parameters for our heuristics. Two decisions that affect both TS and the LPH heuristic are the choice of α for load balancing and the choice to use link distance versus hop count for shortest path routing. We found that, with the exception of the AT&T network, using hop count instead of link distance had a negligible affect on average tree delay while reducing the number of wavelengths required. For the AT&T network, the delay was in some cases doubled when using hop count

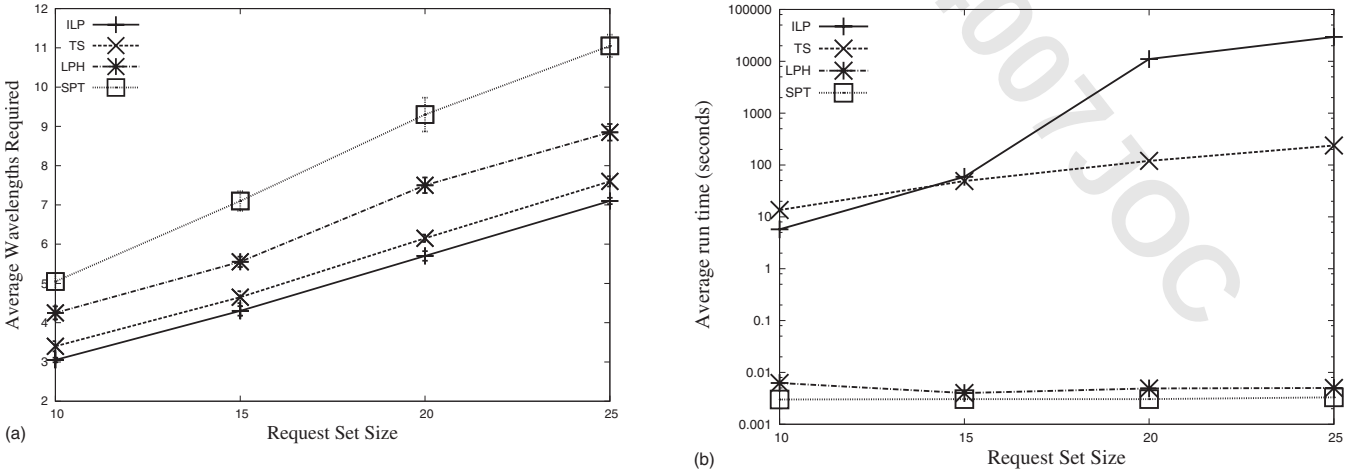


Fig. 8. Performance comparison of ILP and heuristics.

TABLE I
COMPARISON OF TS, LPH, AND SPT OVER DIFFERENT NETWORK TOPOLOGIES FOR A STATIC SET OF 150 MANYCAST REQUESTS

					$D_{max}=6$						$D_{max}=8$						$D_{max}=10$					
					TS		LPH		SPT		TS		LPH		SPT		TS		LPH		SPT	
Network	V	E	δ	τ	w_a	d_a	w_a	d_a	w_a	d_a	w_a	d_a	w_a	d_a	w_a	d_a	w_a	d_a	w_a	d_a	w_a	d_a
AT&T	27	41	3.0	3.4	28.8	11.7	32.5	11.6	43.9	11.2	30.2	11.3	34.5	11.3	47.1	10.8	31.6	11.4	35.8	11.4	47.6	10.7
NSFnet	14	21	3.0	4.3	32.9	9.6	36.7	9.4	46.4	8.6	35.8	9.6	39.6	9.5	49.6	8.4	39.3	9.8	43.0	9.6	55.1	8.4
Italy	21	36	3.4	0.6	29.4	1.7	33.3	1.7	47.0	1.5	31.9	1.8	35.9	1.7	50.8	1.6	33.3	1.8	37.5	1.7	51.5	1.5
24 node	24	43	3.6	3.9	25.5	11.6	28.6	11.4	37.7	10.6	26.8	11.5	30.5	11.3	40.0	10.3	28.4	11.5	32.2	11.4	42.3	10.2

instead of link distance. This trade-off of delay versus wavelengths required is something that must be considered for particular networks, but it seems in most cases the best choice is to perform shortest-path routing based on hop count. Shortest path based on hop count instead of link distance provides better performance because a shortest path according to link distance will likely result in paths with more hops, which results in more resource usage and therefore fewer wavelengths being available.

To perform load balancing, we introduce a parameter, α, where $0 \leq \alpha \leq 1$, as we discussed earlier when describing LPH. The load-balancing updates the weight of each link after a new tree and wavelength are assigned according to $\alpha+(1-\alpha)\times c/c_{max}$, where c is the current number of wavelengths on the link and c_{max} is the number of wavelengths on the most congested link. A smaller value of α puts more emphasis on load balancing when computing shortest paths. We found that if α is set too small, (e.g., 0.2), the load in the network is evenly distributed over most links, but this actually increases the number of wavelengths required. One explanation is that this forces trees to get larger in size in order to use fewer loaded links, which makes it harder to find a single wavelength for later trees. Higher values of α performed the best. As we mentioned previously, we used $\alpha=0.8$. This provided better distribution of load over the network than no load balancing while also decreasing the number of wavelengths required.

One disadvantage of using TS is that we must select a number of different parameters, each having different effects on the performance. To find the best combination of input parameters, we tried 81 different combinations of neighborhood fraction, tenure, diversification iterations, and intensification iterations over NSFnet with a request set size of 50. The number of iterations is fixed at 750 and $\alpha=0.8$. We tried fractions of 6%, 20%, and 50%; tenure values of 10, 20, and 30; diversification values of 15, 25, and 30; and intensification values of 2, 3, and 4. We chose the parameter combination with the best cost/run time trade-off based on the empirical results.

C. Discussion

Our performance evaluation has confirmed that while manycast and multicast are similar, it is important to develop new heuristics for manycast RWA problems. Our SPT heuristic essentially treats the manycast request as a multicast request by fixing the destination set at the source. Both LPH and TS perform much better than SPT.

While TS improves upon LPH in terms of the solution cost, TS has significantly higher run times than LPH. LPH run times are typically about a second. TS, on the other hand, can take over eight hours for a network like the AT&T network with many nodes. The TS implementation can be optimized to reduce run time, but nevertheless there is a large increase in run time for a small increase in performance. Since these computations occur offline, it may still be reasonable to allow for the longer run times to gain around a 10% improvement in cost. In any case, with realistic size networks it is not feasible to get optimal solutions unless significant computing power is available. For example, CPLEX did not find a solution after 3 days for a request set size of just 30 on a machine with 4 cores and Hyper-Threading.

VI. CONCLUSION

We have introduced the static MA-RWA problem and presented three heuristics along with an ILP to solve the problem. Our tabu search heuristic achieved between a 30% and 40% improvement over our simpler shortest-path heuristic and about a 10% improvement over LPH for realistic networks. The TS metaheuristic also produced results similar to the ILP for small networks.

We have several areas of future work for the manycast problem over wavelength-routed networks. One is the dynamic manycast problem. This is especially applicable to Grid networks and cloud computing applications where multiple resources are required (point-to-multipoint) [14,42]. In addition to new algo-

rithms, we could investigate extensions to our current and past work. We can use a modified LPH for each dynamically arriving request or we can also modify our distributed manycast algorithms from our work on dynamic manycast over OBS [43]. Multilayer optimization is also an interesting topic for manycast in the context of Grid networks. The selection of nodes may be based not only on the network state but also on the Grid resource utilization.

Another interesting area of future work is survivability of manycast requests. We can provide survivability for both link and node failures. We can protect against link failures through traditional techniques such as shared path protection, but may also be able to use the extra $|D_c|-k$ nodes from the candidate set to either handle a link *or* a node failure. Switching to a different node in the candidate set depends on the type of application.

We are currently working on incorporating physical-layer impairments into the heuristics (for static MA-RWA) in order to ensure that the signal can be received at the destinations given physical-layer impairments, such as ASE noise, crosstalk, and power loss. We have considered impairments in previous work for manycast over OBS networks [31,44]. In addition to this, we have done work with QoS in dynamic manycast over OBS networks, where one of the QoS parameters is signal quality [7]. This work can be extended for wavelength-routed networks. As an extension to the work presented in this paper, we can make LPH more intelligent when generating sets of route trees by taking into account the signal quality at the nodes. We are investigating different techniques, such as providing quality of transmission (QoT) *guarantee*, where we use traditional RWA but do not admit connections with poor signal quality, and QoT *awareness*, where the RWA algorithms consider physical impairments directly.

Acknowledgments

This work was supported in part by the National Science Foundation (NSF) under grant CNS-0626798. Portions of this paper have appeared in IEEE ANTS 2009 and IEEE ICC 2010.

References

[1] G. E. Keiser, "A review of WDM technology and applications," *Opt. Fiber Technol.*, vol. 5, no. 1, pp. 3–39, 1999.

[2] R. Ramaswami and K. N. Sivarajan, "Routing and wavelength assignment in all-optical networks," *IEEE/ACM Trans. Netw.*, vol. 3, no. 5, pp. 489–500, 1995.

[3] I. Chlamatac, A. Ganz, and G. Karmi, "Lightpath communications: an approach to high bandwidth optical WAN's," *IEEE/ACM Trans. Netw.*, vol. 40, no. 7, pp. 1171–1182, July 1992.

[4] H. Zang, J. P. Jue, and B. Mukherjee, "A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks," *Opt. Networks Mag.*, vol. 1, no. 1, pp. 47–60, Jan. 2000.

[5] S. Y. Cheung and A. Kumar, "Efficient quorumcast routing algorithms," in *Proc. IEEE INFOCOM*, 1994, pp. 840–847.

[6] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi, "Spanning trees short or small," in *Proc. ACM-SIAM Symp. on Discrete Algorithms*, 1994, pp. 546–555.

[7] B. G. Bathula and V. M. Vokkarane, "QoS-based manycasting over optical burst-switched (OBS) networks," *IEEE/ACM Trans. Netw.*, vol. 18, no. 1, pp. 271–283, Feb. 2010.

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

[9] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*. Springer, 1972, pp. 85–103.

[10] W. S. Hu and Q. J. Zeng, "Multicasting optical cross connects employing splitter-and-delivery switch," *IEEE Photon. Technol. Lett.*, vol. 10, pp. 970–972, 1998.

[11] J. Leuthold and C. H. Joyner, "Multimode interference couplers with tunable power splitting ratios," *J. Lightwave Technol.*, vol. 19, no. 5, pp. 700–707, May 2001.

[12] H. S. Chung, S. H. Chang, and K. Kim, "Experimental demonstration of layer-1 multicast for WDM networks using reconfigurable OADM," *Opt. Fiber Technol.*, vol. 15, no. 5–6, pp. 431–437, 2009.

[13] R. Jain, "Internet 3.0: ten problems with current Internet architecture and solutions for the next generation," in *Proc. IEEE MILCOMM*, 2006, pp. 1–9.

[14] "Large hadron collider (LHC) project," http://lhc.web.cern.ch/lhc.

[15] R. Malli, X. Zhang, and C. Qiao, "Benefits of multicasting in all-optical networks," in *Proc. All-Optical Networking*, 1998, pp. 209–220.

[16] G. Sahin and M. Azizoglu, "Multicast routing and wavelength assignment in wide-area networks," *Proc. SPIE*, vol. 3531, pp. 196–208, 1998.

[17] M. Ali and J. S. Deogun, "Power-efficient design of multicast wavelength-routed networks," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 10, pp. 1852–1862, Oct. 2000.

[18] J. Bermond, L. Gargano, S. Perennes, A. A. Rescigno, and U. Vaccaro, "Efficient collective communication in optical networks," *Theor. Comput. Sci.*, vol. 233, no. 1–2, pp. 165–189, 2000.

[19] X. Zhang, J. Y. Wei, and C. Qiao, "Constrained multicast routing in WDM networks with sparse light splitting," *J. Lightwave Technol.*, vol. 18, no. 12, pp. 1917–1927, Dec. 2000.

[20] L. H. Sahasrabuddhe and B. Mukherjee, "Multicast routing algorithms and protocols: a tutorial," *IEEE Network*, vol. 14, no. 1, pp. 90–102, Jan./Feb. 2000.

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

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

[23] S. Sankaranarayanan and S. Subramaniam, "Comprehensive performance modeling and analysis of multicasting in optical networks," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 9, pp. 1399–1413, Nov. 2003.

[24] Y. Xin and G. N. Rouskas, "Multicast routing under optical layer constraints," in *Proc. IEEE INFOCOM*, 2004, vol. 4, pp. 2731–2742.

[25] B. Du, J. Gu, D. H. K. Tsang, and W. Wang, "Quorumcast routing by multispace search," in *Proc. IEEE GLOBECOM*, Nov. 1996, vol. 2, pp. 1069–1073.

[26] C. P. Low, "Optimal quorumcast routing," in *Proc. IEEE*

GLOBECOM, 1998, vol. 5, pp. 3013–3016.
[27] B. Wang and J. Hou, "An efficient QoS routing algorithm for quorumcast communication," in *Proc. IEEE ICNP*, 2001, pp. 110–118.
[28] F. A. Chudak, T. Roughgarden, and D. P. Williamson, "Approximate k-MSTs and k-Steiner trees via the primal-dual method and Lagrangean relaxation," *Math. Program.*, vol. 100, no. 2, pp. 411–421, 2004.
[29] X. Huang, Q. She, V. M. Vokkarane, and J. P. Jue, "Manycasting over optical burst-switched networks," in *Proc. IEEE ICC*, 2007, pp. 2353–2358.
[30] Q. She, X. Huang, N. Kannasoot, Z. Qiong, and J. P. Jue, "Multiresource manycast over optical burst-switched networks," in *Proc. IEEE ICCCN*, Aug. 2007, pp. 222–227.
[31] B. G. Bathula, V. M. Vokkarane, R. R. C. Bikram, and S. Talabathula, "Impairment-aware manycast algorithms over optical burst-switched networks," in *Proc. IEEE ICCCN*, 2008.
[32] Q. She, N. Kannasoot, J. P. Jue, and Y.-C. Kim, "On finding minimum cost tree for multi-resource manycast in mesh networks," *Opt. Switching Networking*, vol. 6, no. 1, pp. 29–36, Jan. 2009.
[33] D. Din, "A hybrid method for solving ARWA problem on WDM network," *Comput. Commun.*, vol. 30, no. 2, pp. 385–395, 2007.
[34] S. Azodolmolky, M. Klinkowski, E. Marin, D. Careglio, J. S. Pareta, and I. Tomkos, "A survey on physical layer impairments aware routing and wavelength assignment algorithms in optical networks," *Comput. Netw.*, vol. 53, no. 7, pp. 926–944, 2009.
[35] J. P. Jue, "Lightpath establishment in wavelength-routed WDM optical networks," in *Optical Networks: Recent Advances*. Springer, 2001, pp. 99–122.
[36] M. Andrews and L. Zhang, "Complexity of wavelength assignment in optical network optimization," *IEEE/ACM Trans. Netw.*, vol. 17, no. 2, pp. 646–657, 2009.
[37] H. Takahashi and A. Matsuyama, "An approximate solution for the Steiner problem in graphs," *Math. Japonica*, vol. 24, no. 6, pp. 573–577, 1980.
[38] F. Glover and M. Laguna, *Tabu Search*. Kluwer Academic Publishers, 1997.
[39] J. Kuri, N. Puech, M. Gagnaire, and E. Dotaro, "Routing foreseeable lightpath demands using a tabu search metaheuristic," in *Proc. IEEE GLOBECOM*, 2002, vol. 3, pp. 2803–2807.
[40] Y. Wang, T.-H. Cheng, and M.-H. Lim, "A tabu search algorithm for static routing and wavelength assignment problem," *IEEE Commun. Lett.*, vol. 9, no. 9, pp. 841–843, Sept. 2005.
[41] C. Dzongang, P. Galinier, and S. Pierre, "A tabu search heuristic for the routing and wavelength assignment problem in optical networks," *IEEE Commun. Lett.*, vol. 9, no. 5, pp. 426–428, May 2005.
[42] "NP science network requirements," 2008, http://www.es.net/pub/esnet-doc/NP-Net-Req-Workshop-2008-Final-Report.pdf.
[43] V. M. Vokkarane and B. G. Bathula, "Manycast service in optical burst/packet switched (OBS/OPS) networks (Invited Paper)," in *ICST/ACM GridNets*, 2008, pp. 231–242
[44] B. G. Bathula, V. M. Vokkarane, and R. R. C. Bikram, "Impairment-aware manycasting over optical burst-switched networks," in *Proc. IEEE ICC*, 2008, pp. 5234–5238.

Neal Charbonneau (S'08) received the B.S. degree in computer science from the University of Massachusetts, Dartmouth, in 2008. He is currently pursuing his M.S. degree in computer science at the University of Massachusetts, Dartmouth. His interests include computer networks and software design and development.

Vinod M. Vokkarane (S'02-M'04-SM'09) received the B.E. degree with honors in computer science and engineering from the University of Mysore, India, in 1999; the M.S. degree in computer science from the University of Texas at Dallas in 2001; and the Ph.D. degree in computer science from the University of Texas at Dallas in 2004. He is a recipient of the Texas Telecommunication Engineering Consortium Fellowship 2002-03 and the University of Texas at Dallas Computer Science Dissertation of the Year Award 2003-04. Dr. Vokkarane is the co-author of the book *Optical Burst Switched Networks*, Springer, 2005. He is currently an Assistant Professor of Computer and Information Science at the University of Massachusetts, Dartmouth. His primary areas of research include design and analysis of architectures and protocols for optical and wireless networks.