Static Routing and Wavelength Assignment for Multicast Advance Reservation in All-Optical Wavelength-Routed WDM Networks

Neal Charbonneau, *Member, IEEE*, and Vinod M. Vokkarane, *Senior Member, IEEE*

***Abstract*—In this paper, we investigate the static multicast advance reservation (MCAR) problem for all-optical wavelength-routed WDM networks. Under the advanced reservation traffic model, connection requests specify their start time to be some time in the future and also specify their holding times. We investigate the static MCAR problem where the set of advance reservation requests is known ahead of time. We prove the MCAR problem is NP-complete, formulate the problem mathematically as an integer linear program (ILP), and develop three efficient heuristics, *seqRWA*, *ISH*, and *SA*, to solve the problem for practical size networks. We also introduce a theoretical lower bound on the number of wavelengths required. To evaluate our heuristics, we first compare their performances to the ILP for small networks, and then simulate them over real-world, large-scale networks. We find the *SA* heuristic provides close to optimal results compared to the ILP for our smaller networks, and up to a 33% improvement over *seqRWA* and up to a 22% improvement over *ISH* on realistic networks. *SA* provides, on average, solutions 1.5–1.8 times the cost given by our conservative lower bound on large networks.**

***Index Terms*—Advance reservation, heuristics, integer linear program (ILP), multicast, NP-complete, routing and wavelength assignment (RWA), scheduled demands, simulated annealing.**

I. Introduction

OPTICAL wavelength-routed WDM [1] networks will play an important role in the future Internet to provide large bandwidth and services to next-generation applications. In WDM networks, each fiber is partitioned into a number of wavelengths, each capable of transmitting data. 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). When a connection request arrives at the network, the request must be routed over the physical topology and also assigned a wavelength. 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 a single-hop, or all-optical, WDM network, the signal is transmitted all-optically through the network, whereas in multihop networks the signal may undergo optical/electronic/optical (O/E/O) conversion at some intermediate nodes. In the absence of wavelength converters (which are expensive), a connection in a single-hop WDM network must use the same wavelength across all links. This is known as the *wavelength continuity constraint*. Also note that if a wavelength is used on a particular link, no other connection may use the same wavelength over the same link in either direction. We will refer to this as the *wavelength clash constraint*. Multihop networks can use different wavelengths on different links because the signal may undergo O/E/O conversion at some intermediate nodes, allowing it to be retransmitted on a wavelength different from the received wavelength. This conversion process can be expensive, however, both in terms of cost of equipment and due to the dependence of the conversion process on the connection line rate and modulation format. The disadvantage of single-hop networks is that, in the absence of regenerators, the signal noise accumulates from physical-layer impairments, such as crosstalk, ASE noise, and nonlinear impairments like four-wave-mixing, cross phase modulation, and stimulated Brillouin and Raman scattering. To counter this, impairment-aware routing can be used to ensure 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 [4]. In this paper, we consider single-hop networks. Impairment-awareness is an area of future work.

Traditionally, communication in a network is unicast, where a single source sends data to a single destination. In this paper, we consider multicast communication, where a single source communicates with multiple destinations simultaneously. Multicast applications are becoming more popular and will make up an important part of future Internet traffic. Examples of multicast applications are video conferencing, interactive distance learning, streaming media, distributed data processing, storage area networks, and e-Science. Because these applications require large amounts of bandwidth, it is important to support multicast at the optical layer. The benefits of supporting multicast at the optical layer have been discussed in [5]–[7]. A unicast request is supported by the use of lightpaths, as previously discussed. To efficiently support multicast requests, the network must create *light-trees* [6]. A light-tree is a generalization of a lightpath that starts at the source node of a multicast request and reaches all its destinations by possibly branching (splitting

Manuscript received September 11, 2010; revised March 23, 2011; accepted April 20, 2011; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor M. Reisslein. Date of publication November 24, 2011; date of current version February 15, 2012. This work was supported in part by the National Science Foundation (NSF) under Grant CNS-0626798 and the Department of Energy (DOE) COMMON Project under Grant DE-SC0004909. Portions of this paper have appeared in the IEEE Global Communications Conference (GLOBECOM), Miami, FL, December 6–10, 2010.

N. Charbonneau was with the Department of Computer and Information Science, University of Massachusetts, Dartmouth, MA 02747 USA. He is now with the MITRE Corporation, Bedford, MA 01730 USA (e-mail: ncharbonneau@ieee.org).

V. M. Vokkarane is with the Department of Computer and Information Science, University of Massachusetts, Dartmouth, MA 02747 USA (e-mail: vvokkarane@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2011.2175007

the signal) at intermediate nodes. The problem of finding the optimal route for a light-tree is equivalent to finding a Steiner tree, which is known to be NP-complete [8], 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 splitter-and-delivery (SaD)-based switches [9], [10]. These switches are known as multicast-capable OXCs (MC-OXCs).

Two traffic models are usually considered for wavelength-routed networks: static and dynamic [11]. A static traffic model gives all the traffic demands between source and destinations ahead of time. A traffic matrix is given, and the goal is typically to find an RWA that can meet all the demands and minimize overall cost (e.g., using the least number of transmitters/receivers). Dynamic traffic requests arrive one by one 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. There is a significant amount of work for the multicast problem with these types of traffic demands [12]–[20].

We can further classify the above traffic models as *immediate reservation* (IR) or *advance reservation* (AR) [21] requests. The data transmission of an IR demand starts immediately upon arrival of the request, and the holding time is typically unknown for dynamic traffic or assumed to be infinite for static traffic. AR demands, in contrast, typically specify a data transmission start time that is sometime in the future and also specify a finite holding time. Advance reservation is also referred to as scheduled demands [22], especially when considering static traffic.

In this paper, we consider static advance reservation of multicast demands, where the advance reservation demands are known ahead of time. While there is significant work for immediate reservation of multicast demands, there are few works dealing with advance reservation [23]–[25]. There are many examples where we know that a certain amount of bandwidth will be required by a particular application for some time in the future. Examples include reserving extra capacity for peak hours (logical topology reconfiguration), scheduled transfer of large data, real-time experiments in e-Science, scheduled video conferences or streaming media broadcast, and remote surgery. These applications of advance reservation overlap with the applications of multicast that we presented earlier. Some applications of multicast advance reservation include large-scale distributed or replicated data transfer, e-Science applications like remote experimentation, and streaming media distribution (e.g., IPTV). For example, Cisco is investing in telepresence, which is well suited for multicast advance reservation. Multicast advance reservation is also useful in Grid or service-oriented networks where applications directly request resources. Large-scale science experiments, like those at CERN [26], require reserving large amounts of bandwidth to transfer data sets to possibly multiple locations. The alternatives to using advance reservation requests are to either overprovision resources or to use dynamic immediate reservation connection requests when additional bandwidth is required. The first alternative wastes expensive resources, while the second approach has the possibility of the request being blocked due to insufficient resources. Advance reservation is beneficial to both the providers and users. Given advance reservation requests, the provider can better optimize resource usage (due to differences in the time the request arrives and the time the resources must be reserved) [27], which is especially true for multicast traffic. The user submitting the advance reservation request can receive better quality of service compared to dynamic immediate reservation requests.

Advance reservation has been proposed for nonoptical networks in the past (e.g., [28]), where the research focus was on dynamically arriving calls. It was first proposed for WDM optical networks for unicast traffic in [21]. The basic idea is that instead of a request reserving resources immediately after it is received, the request specifies some additional information about start times and durations that it requires. The authors of [21] defined three types of advance reservation requests: specified start time specified duration (STSD), specified start time unspecified duration (STUD), and unspecified start time specified duration (UTSD). As the names suggest, STSD requests specify both the start and duration of the request. A typical unicast request may be a tuple specifying a source and destination: (s, d), but an STSD advance reservation unicast request specifies a start time α and a duration ω: (s, d, α, ω). STUD specifies only the start time while the duration is variable (usually limited by some threshold). UTSD requests do not specify a start time at all. It is assumed these types of requests must be accommodated as soon as possible and have a fixed duration once scheduled.

Advance reservation requests can be either static or dynamic in nature. Static advance reservation means that all of the requests are known ahead of time, while dynamic advance reservation requests arrive according to some stochastic process. Kuri *et al.* [22], [29], [30] investigated the case of static STSD advance reservation requests. They were referred to as "foreseeable" lightpath demands or scheduled lightpath demands (SLDs). They presented a number of heuristics and also a branch and bound algorithm to find the optimal cost given a set of precomputed paths. Wang *et al.* [31] introduced a new type of advance reservation request that used a sliding window. Instead of a demand starting at some fixed time, it can start at anytime in a given window. This type of request has the form (s, d, l, r, τ), where l is the start of the window, r is the end, and τ is the duration of the request $(r - l \geq \tau)$. The request must be scheduled within the window $[l, r]$. The problem can be solved in two steps. First, schedule all the demands within their windows so that the temporal overlap is minimized. The result of this stage is the STSD advance reservation problem, which can then be solved independently. He *et al.* [32], [33] introduced a new type of advance reservation with both unspecified start times (start as soon as possible) and unspecified durations. Gagnaire *et al.* [34] were the first to propose algorithms for a mix of advance reservation and dynamic immediate reservation requests. Wallace *et al.* [35] proposed a new type of advance reservation based on STSD. Here, instead of specifying a start time, the demand specifies a requested service time (RST). The demand may still be scheduled after the RST. The goal now becomes to minimize the difference between the actual start time and the RST. They present a mixed-integer linear program as well as two meta-heuristics. The author in [36] presented some heuristics as well as lower bounds for the STSD advance

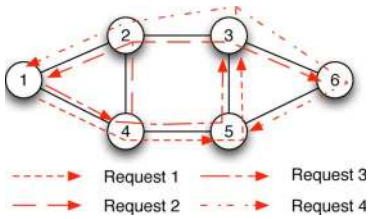


Fig. 1. Example of the MCAR problem. The requests are shown in Table I. The time independence allows requests to reuse resources. In this case, only a single wavelength is required.

TABLE I
EXAMPLE SET OF STATIC MULTICAST ADVANCE RESERVATION REQUESTS. THE OPTIMAL RWA IS SHOWN IN FIG. 1

Request	Source	Destinations	Start	End
1	1	{3,5}	12:00pm	3:30pm
2	4	{1,2,6}	2:30pm	6:00pm
3	1	{3,4}	4:00pm	9:00pm
4	3	{1,5}	7:00pm	10:00pm

reservation problem. Additional solution techniques are also presented in [37] for the sliding window model and in [38] for STSD.

The work mentioned so far focuses on a static set of advance reservation demands. There has also been work dealing with dynamic advance reservation demands, mostly from the grid community [39]–[43]. In addition to RWA, there is also work for survivability [44]–[46] and grooming [47].

The multicast static advance reservation problem was initially proposed in [48], where two heuristics were presented. We provide a more comprehensive evaluation of the problem. Our contributions are discussed at the end of this section.

Table I shows an example of a static set of multicast advance reservation requests. Each request specifies the source, the destination set, the start time, and the end time. The optimal RWA for this set is shown in Fig. 1. To accommodate these requests, only a single wavelength is used because we can take advantage of requests that are independent in time. For example, Requests 2 and 4 use some of the same links on the same wavelength because they are independent in time, and similarly with Requests 1 and 3. In traditional RWA, we can find independence in the spatial domain to reuse wavelengths. Advance reservation allows reuse of wavelengths due to spatial *or* temporal independence.

The contributions of this paper include a proof that the MCAR problem is NP-complete, a mathematical formulation of the problem as an integer linear program (ILP), three efficient heuristics to solve the problem in realistic amounts of times, and a theoretical lower bound. The paper is organized as follows. In Section II, we first define the problem formally and then state our traffic patterns and define the metrics we calculate. We also show that the MCAR problem is NP-complete. Section III presents the ILP formulation of the problem, while Section IV presents the three heuristics. Two simple lower bounds are derived in Section V. We evaluate our heuristics in Section VI and finally conclude the paper in Section VII.

II. MCAR PROBLEM DEFINITION

Given a network, $G = (V, E)$, a multicast advance reservation request is defined as $R = (s, D, \alpha, \omega)$, where $s \in V$ is the source node, $D \subseteq V - \{s\}$ is the set of destination nodes, α is the arrival time, and ω is the teardown time. Starting at the source node, we must find a light-tree (or lightpath) that reaches all destinations in D. The resources must be reserved at α and released at ω. This is a type of STSD [21] advance reservation request. We assume that each demand requires one wavelength and uses 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 MCAR problem is defined as follows.

Definition $\mathrm{MCAR}(G, \Delta)$*:* Given a network $G = (V, E)$ and a set, $\Delta = \{R_1, R_2, \ldots, R_n\}$, of multicast advance reservation requests, the solution must assign a route tree and 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 assumption is that these are handled in batch, so the computation occurs offline. This computation may occur daily, weekly, or monthly, depending on the service provider.

A. Network Assumptions

We assume single-hop or all-optical WDM networks. Once the signal enters the network, it is switched all-optically. We assume there is a single link between each node (i.e., link (i, j) is the same as (j, i)) and each wavelength can be used in only one direction. For example, if a wavelength is used on link (i, j) from node i to j, it cannot be used on the same link from node j to i. The light-trees are unidirectional. We assume wavelength converters are not available, so the wavelength continuity constraint applies, meaning the light-tree or lightpath must use the same wavelength over all links. We also assume that all switches are MC-OXCs that can split an incoming signal to any number of output ports.

We assume that the time domain is broken down into discrete timeslots. Each timeslot, T, represents some unit of real time (for example, 15 min). The request must start at the beginning of a timeslot, and its duration must be an integer multiple of the timeslot length. For example, a request may start at 1:15 p.m. and last until 6:00 p.m. (19 timeslots). The minimum duration of a request is assumed to be one timeslot, with a maximum duration of 24 h.

Given our assumption of a transparent optical network and the fact that the signals will be split at intermediate nodes, the physical-layer impairments, power, and the duration of a timeslot are important to consider. However, in this paper we are interested in defining the problem and presenting some initial solutions. In our future work, we will also consider the physical-layer impairments. We have previously done work in impairment-awareness, e.g., [49].

B. Time Correlation and Wavelength Reuse

We now define two metrics we use for evaluating our proposed solutions. For static advance reservation traffic, we can classify the time correlation of the request set. We denote the time correlation of request set, Δ, by $\tau(\Delta)$. If few requests

overlap with each other in the time domain, the set is said to be weakly correlated ($\tau(\Delta)$ is close to 0), while if there is a lot of overlap it is strongly correlated ($\tau(\Delta)$ is close to 1). A set of a weakly correlated requests should be able to obtain more temporal reuse, thus requiring fewer wavelengths, than a strongly correlated request set. For example, if all requests overlap with all other requests (as in static immediate reservation), then there is no chance for temporal reuse, and this is the strongest correlation ($\tau(\Delta) = 1$). The time correlation of a set, τ, can be defined precisely as follows. Let $c(r, \Delta)$ be the number of other requests r overlaps in time with in Δ (note if r_i overlaps with r_j, r_j overlaps with r_i). We have

$$\tau(\Delta) = \frac{\sum_{r \in \Delta} c(r, \Delta)}{|\Delta| * (|\Delta| - 1)}.$$

We use time correlation values of 0.1, 0.4, and 0.7 to represent weak, medium, and strong time correlation between requests. Related to time correlation, a reuse metric ψ can also be defined that will quantify the amount of temporal wavelength reuse our proposed algorithms are able to obtain. A wavelength on a link is *reused* if it has been used by two or more separate lightpath/light-trees at separate times. Let E be the set of links used in the network and W be the number of wavelengths required. Define $r_{e,w}$ as the number of requests that use wavelength w on link $e \in E$. Let w_e be the number of wavelengths used on link e over the entire simulation (this does not count reuse). First, we summarize the amount of wavelength reuse among all requests for a given link (r_e), then define the reuse metric

$$r_e = \sum_{i=0}^{W} r_{e,w} \quad \psi = \frac{\sum_{e \in E} \left(1 - \frac{w_e}{r_e}\right)}{|E|}.$$

When there is no reuse at all, i.e., each wavelength is only used once the entire time, then $\frac{w_e}{r_e} = 1$, and the metric ψ is 0. As the reuse increases, that ratio decreases. ψ will be between 0 and 1, with higher values indicating more reuse.

C. $\mathcal{NP}$-completeness

This section discusses the complexity of the problem. We show that the MCAR problem is NP-complete and therefore is unlikely to have an efficient polynomial-time algorithm. First, we define the MCAR problem as a decision problem.

Definition $k\text{MCAR}(G, \Delta, k)$*:* Given an undirected graph G and a set of multicast advance reservation requests Δ, is it possible to assign a light-tree to each multicast request using k wavelengths? Note the wavelength continuity and clash constraints apply. We assume that each node is an MC-OXC capable of splitting a signal to any output port and there is no transmitter or receiver limit at each node (this is what the optimization problem will minimize). We also assume that the source uses only a single wavelength to transmit data. These assumptions are in agreement with the assumptions of the optimization problem.

Theorem II.1: The kMCAR problem is NP-complete.

Proof: The kMCAR problem is in the complexity class NP because given a set of light-trees, it can easily be verified in polynomial time that the solution is valid and uses k wavelengths. One pass can validate the trees and the continuity constraint, which can be done in time proportional to the request set size, and another pass can then check all links to verify the clash constraint, which can be done in time proportional to the number of links in the network. To complete the proof, we will show that COLOR (CHROMATIC NUMBER, proven NP-complete in [8]) is polynomial-time-reducible to kMCAR. The COLOR problem takes as its input a graph G and an integer k and asks whether or not the nodes of the graph can be colored using only k colors in such a way that no two adjacent nodes have the same color.

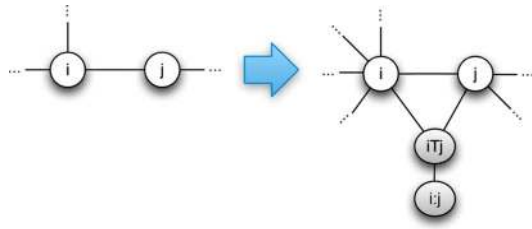


Fig. 2. Construction of new nodes in Theorem II.1. For each edge in G, three new links and two new nodes are added in G'.

Let $\langle G = (V, E), k \rangle$ be the input to an instance of COLOR. We construct a new graph, $G' = (V', E')$, by making the following modifications. First, $V' = V$ and $E' = E$. For each pair of adjacent nodes, $(i, j) \in E$, add a new node adjacent to both, forming a triangle. Call this new node iTj. Now add another node, Node $i : j$, connected only to iTj. This step is shown in Fig. 2. For each edge in G, this adds three new edges and two new nodes to G'. This new graph represents a network where each edge is a bidirectional link. We assume that the nodes labeled $*T*$ (where "*" refers to any value) simply transmit any received signal to the adjacent node labeled $*:*$ to which it is connected.

Now, we must define the multicast request set Δ. Let each node in G be a source node. This means there are $|V|$ requests in Δ. Each source node i will have its destination set be any of the new nodes named $*:i$ or $i:*$. In Fig. 2, both Nodes i and j would have $i : j$ in their destination sets. The destination set size for each source node is equal to the number of neighbors (adjacent nodes) that node has.

The last step is to assign start and end times for each request. This is done as follows. Choose an arbitrary start node, Node i, and assign $\alpha_i = 1$ and $\omega_i = 2$ to the request sourced at that node (we assume the intervals are inclusive, so $[1, 2]$ overlaps with $[2, 3]$). Repeat the following step until all requests have been assigned start and end times. Choose a node, Node j, whose request does not have a time assignment, but has at least one neighbor with a time assignment. Assign $\alpha_j = \min_n\{\omega_n\}$ and $\omega_j = \max_n\{\omega_n\} + 1$, where n represents each of the neighboring nodes (that already have time assignments). When all nodes have been assigned times, this assignment of start and end times ensures that each request overlaps in time with the requests of neighboring nodes. Each request in Δ can be defined as $R_i = (i, \{i : * \cup * : i\}, \alpha_i, \omega_i)$, where i, $i:*$, $*:i$. This construction can be done in polynomial time. An example of the entire construction is shown in Fig. 3. The resulting multicast request set, Δ, is shown in Table II.

To assign start and end times, we start with Node 1 by assigning $\alpha_1 = 1$ and $\omega_1 = 2$. Next, we choose Node 2. Node 2's start time is assigned the minimum end time of its neighbors. Node 1 is the only neighbor with an assigned time, so Node 2's

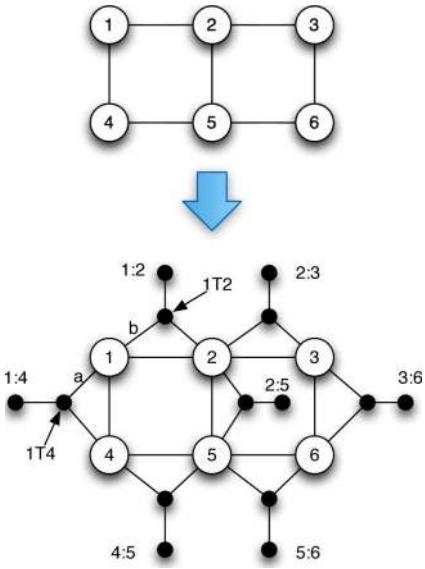


Fig. 3. Construction used in proof of Theorem II.1. The graph on top is G, and the graph underneath is G'.

TABLE II
MULTICAST REQUEST SET RESULTING FROM CONSTRUCTION IN THEOREM II.1

Request	Source	Destinations	Start	End
1	1	{1:2, 1:4}	1	2
2	2	{1:2, 2:3, 2:5}	2	3
3	3	{2:3, 3:5}	3	4
4	4	{1:4, 4:5}	2	3
5	5	{4:5, 2:5, 5:6}	3	6
6	6	{4:6, 3:6}	4	5

start time becomes 2. Node 2's end time is the maximum end time of its neighbors, plus one. Again, only Node 1 has been assigned times, so Node 2's end time is 3. Next, we choose Node 4. The assignment is the same as Node 2's. Next, we choose Nodes 3 and 6. The assignment works in a similar way, and the values are shown in the table. Lastly, Node 5 is assigned a time. The start time is the minimum end time of all its neighbors, which is from Node 2 (or 4), and it is equal to 3. The end time is the maximum end time of Node 5's neighbors plus one, which is from Node 6 and is equal to $5 + 1 = 6$. The order the nodes are chosen does not matter; the end result will be that neighboring nodes overlap in time. Our claim is that the graph G is colorable using k colors *if and only if* an RWA for Δ exists on G' using k wavelengths.

($\Longrightarrow$)*:* Assume G can be colored using k different colors. This means that no two adjacent nodes have the same color. Let the color of Node i be $c(i)$. The solution to the kMCAR problem is constructed as follows. Each source node in G' transmits on wavelength $\lambda_{c(i)}$ over the newly created links to each of its destinations by splitting the signal. For example, Node 1 in Fig. 3 would transmit its signal onto links a and b. Nodes $1T4$ and $1T2$ simply transmit any received signal to *1:4* and *1:2*, respectively. Clearly, for each source, all of its destinations are reached in this way. The wavelength continuity constraint is satisfied since each source transmits only on the color assigned to it. The wavelength clash constraint is satisfied because each source shares its destinations only with its neighbors and its neighbors must be transmitting on different wavelengths because they were assigned different colors in the graph coloring. Note the requests that share destinations overlap in time, therefore they *must* use different wavelengths. The number of wavelengths required is k since this is how many colors were needed to color the graph and each node simply transmits on its color. Therefore, this is a valid solution to kMCAR using k wavelengths.

($\Longleftarrow$)*:* Conversely, assume G' has a valid RWA of Δ using k wavelengths. Since the RWA is valid, each source reaches its destinations on a single wavelength. Since each source shares a destination with each of its neighbors, it must be transmitting on a different wavelength than each of its neighbors because they overlap in time, otherwise the wavelength clash constraint would be violated. Assign the wavelength color to be the color of the node. This results in a k coloring of G since k wavelengths were used in G'. ■

III. INTEGER LINEAR PROGRAM FORMULATION

We formulate the MCAR problem mathematically as an ILP. We use an ILP solver (CPLEX) to find an optimal solution. Note that this does not change the complexity of the problem. Solving an ILP is also NP-complete. The objective is to minimize the number of wavelengths used. Note that this is different from minimizing the total number of wavelengths on any link (congestion). The authors in [45] have presented an ILP for unicast protection for STSD requests with an objective to minimize congestion.

The parameters to the ILP are the following.

- We use i, j to denote links, m to denote the mth multicast request, and w to denote wavelengths.
  — s_m: The source node of request m.
  — D_m: The destination set of request m.
- T_{m_1,m_2}: 1 if requests m_1 and m_2 overlap in time, 0 otherwise.

The ILP will solve for the following variables.

- $y_{i,j}^{m,w}$: 1 if wavelength w is used on link (i, j) for multicast request m, 0 otherwise.
- U_i^m: order node i was added to the tree for multicast 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:

$$\text{minimize}: \quad \text{maxIndex}$$

Subject to:

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

$$y_{i,j}^{m_1,w} + y_{i,j}^{m_2,w} + y_{j,i}^{m_1,w} + y_{j,i}^{m_2,w} + T_{m_1,m_2} \leq 2 \quad \forall m_1, m_2 (> m_1) i, j, w \tag{2}$$

$$\sum_i \sum_w y_{i,j}^{m,w} = 1 \qquad \forall m, j \in D_m \tag{3}$$

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

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

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

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

$$\sum_j y_{j,i}^{m,w} - \sum_j y_{i,j}^{m,w} \leq 0 \qquad \forall m, w, i \neq D_m \tag{8}$$

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

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

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

Constraint (1) is used to keep track of the maximum wavelength index used. Constraint (2) is used to enforce the wavelength clash constraint. Two or more requests can use the same wavelength over the same link only if they are all independent. This constraint checks every pair of requests to ensure that if they share a wavelength on a link, they are independent in time ($T_{m1,m2} = 0$). Since we assume there is a single link between each node, both directions are considered in this constraint. Constraints (3)–(9) are used to build each route tree. Constraint (3) ensures all destinations in D are reached. 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 destination set that have an incoming wavelength must have at least one outgoing. 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 it.

IV. MCAR Heuristics

We have shown the MCAR problem is NP-complete, hence no efficient algorithm exists for it unless P = NP. We introduce three efficient heuristics in this section to solve the problem suboptimally.

Before discussing each individual heuristic, we describe the algorithm we use to create Steiner trees for routing individual multicast requests. As we mentioned previously, the Steiner tree problem is NP-complete, but many approximation algorithms exist. We use the heuristic called MPH presented in [50], which is a $2(1 - \frac{1}{|D|})$ approximation for Steiner trees. We include it here, shown in Algorithm 1, for completeness and to discuss its runtime.

The heuristic works by creating the tree incrementally. The function SP in line 4 finds the shortest path between two nodes. The heuristic starts by selecting the node in D with 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 all nodes in D are part of the tree. The outer loop executes a total of $|D|$ times. Each iteration of the loop, we must find the minimum-cost shortest path between a node on the tree and a node in D. To check all of these shortest paths, a loose upper bound for the number of comparisons is $O(|V||D|)$ since u_1 could come from any node in V and u_2 could come from any node in D. We will use $O(|D|^2|V|)$ as the runtime for MPH. This analysis assumes shortest paths are precomputed.

Algorithm 1: MPH Approximation for Steiner Trees Presented in [50]

```
1: T = (V', E') s.t. V' = {s}, E' = φ
2: copy = 0
3: while copy < |D| do
4:     path = min{SP(u1, u2)}   u1 ∈ V', u2 ∈ D − V'
5:     V' = V' ∪ {nodes in path}
6:     E' = E' ∪ {edges in path}
7:     copy = copy + 1
8: end while
9: T.cost = Σ_{i,j∈E'} c_{i,j}
```

Algorithm 2: Sequential RWA Heuristic

```
1: sort_time(Δ)
2: for all R ∈ Δ do
3:     generated(R) = gen_kAlt()
4: end for
5: for all R ∈ Δ do
6:     assigned = false
7:     for i = 0 to k do
8:         tree = generated(R)[i]
9:         if !newWL(tree) then
10:            firstFit(tree)
11:            assigned = true
12:            break
13:        end if
14:    end for
15:    if !assigned then
16:        firstFit(generated(R)[0])
17:    end if
18: end for
```

A. Sequential RWA (seqRWA)

The sequential RWA heuristic *seqRWA* shown in Algorithm 2 is a simple heuristic for MCAR. It generates k-alternate trees for each request in advance and uses these in the RWA process. The heuristic begins by sorting the requests according to start time. In this way, the heuristic treats the static set as if they arrive one by one. For each request, it loops through the k trees and tries to assign a wavelength according to the first-fit policy [11]. If assigning the wavelength results in no increase in overall wavelength count for the network (line 9), that tree and the corresponding wavelength are chosen as the light-tree and the network is updated to include that tree (line 10). Otherwise, it tries the remaining trees. If all trees result in an increase in number of wavelengths, the first tree is selected and a new wavelength is used (line 16). The heuristic maintains wavelength availability on each link for all timeslots. The same wavelength can be used on the same link by different requests if they do not overlap in time.

In the case of k-alternate routing for unicast requests, the routes are typically link-disjoint. It is difficult, if not impossible for some topologies, to generate k link-disjoint multicast trees. We use a simple algorithm to generate k-trees (line 3). For each request, the trees are generated using MPH. After generating a tree, the costs of the links that this tree uses are then incremented to discourage the remaining trees from using them. Once all k trees are generated in this way, the link costs are reset, and trees are generated for the next request.

1) Complexity: When describing the complexity of this and the next heuristic, we use the term $|D|$, which is the size of the multicast request. Each request may have a different request size, so $|D|$ will refer to the maximum multicast request set size of all requests in the runtime analysis since we are looking at upper bounds of runtimes. The cost of creating the k alternate trees dominates the runtime of this heuristic. For each of the $|\Delta|$ requests, we must create k trees. To create each tree, we use MPH and update the edge weights of the links used. This means that for each of the k trees, the shortest paths must be recomputed as well. The computation of a Steiner tree after the shortest paths are computed takes $O(VD^2)$ time.[1] We must compute shortest paths each time, which can be done in $O(V^3)$ by using the Floyd–Warshall algorithm, for example. The total runtime to generate the k alternate trees is then $O(\Delta k(V^3 + VD^2))$. The remainder of the heuristic is the for loop that assigns trees and wavelengths. This is executed $O(\Delta k)$ times. During each iteration, the heuristic checks the wavelength availability and assigns a wavelength. This can be done in time proportional to the size of the generated tree since it only involves checking the tree's links for availability. We can use $O(E)$ as an upper bound for this. The loop therefore takes $O(\Delta kE)$. The upper bound for the total runtime of the heuristic is therefore given by the first loop: $O(\Delta k(V^3 + VD^2))$.

B. Independent Set Heuristic (ISH)

The *seqRWA* heuristic, while simple, has several disadvantages. First, the trees are precomputed, so there is little flexibility in the routing of requests. Second, it does not do anything specifically to take advantage of the time independence between requests besides sorting them by time. The next heuristic, called ISH, shown in Algorithm 3, tries to eliminate these disadvantages. The basic idea is that when the heuristic is complete, it will have k sets of requests $\{\tau_i\}_{i=1}^{k}$, such that $\bigcup_{i=1}^{k} \tau_i = \Delta$, where all the requests in any given set are independent in space or in time, meaning they use disjoint resources either in the spatial domain or in the temporal domain. This means that we can use k wavelengths to route Δ since all of the requests in each set can use the same wavelength. The idea for creating independent sets has been explored in [36]–[38], but this heuristic is unique. We use a similar structure to [36] and also employ an interval graph to find additional time-independent requests.

The heuristic works as follows. The requests are sorted in nonincreasing order according to $|D|$. For the ith iteration of the outer loop, we create a new set τ_i and select the next unrouted request R. We then find the maximum number of unrouted requests that are independent in time with each other and the current request R. This is done by first finding all time independent requests of R and creating an interval graph of them (line 8). With the interval graph, we can find the maximum independent set (line 9), which results in the maximum number of requests independent with each other and R. The maximum independent set problem for general graphs is NP-complete [8], but for interval graphs there exists an $O(n \log n)$ algorithm [51]. These time-independent requests are now added to τ_i. This is how the heuristic attempts to take advantage of the time independence of requests. The time-independent requests can now be routed over the network using MPH (line 11). Note we do not implement any wavelength assignment. This is done at the end of the heuristic by simply assigning a unique wavelength to each independent set, τ_i.

The time-independent requests can be routed using the same links since the requests are not active at the same times. Given these requests, it may be possible to add additional requests to τ_i that are independent in space, i.e., we can find a route that does not overlap spatially with other requests. We perform another linear scan through the remaining requests. For each request not routed yet, R_2, we try to add it to τ_i by first removing any edges in G used by requests that overlap with R_2 (lines 17–18) in time in τ_i. R_2 may only overlap in time with few requests in τ_i. We only remove links from G that are used by those requests. If R_2 can still be routed, it is added to τ_i; otherwise it is left alone.

The heuristic continues until all requests have been routed. Once it is complete, the requests in τ_i are assigned wavelength i on their selected route trees. In addition to sorting by $|D|$, we tried sorting by starting times in nonincreasing order. We also tried performing the inner loop in reverse. We found the combination presented here performed the best.

Algorithm 3: Independent Set Heuristic (ISH)

```
 1: i = 1
 2: routed = {}
 3: sort(Δ)
 4: for all R ∈ Δ do
 5:     if R ∈ routed then
 6:         continue
 7:     end if
 8:     I = construct_interval(R, Δ)
 9:     time_indep = max_indep(I)
10:     τ_i = {R} ∪ time_indep
11:     route(τ_i, G)
12:     routed = routed ∪ τ_i
13:     for all R_2 ∈ Δ do
14:         if R_2 ∈ routed then
15:             continue
16:         end if
17:         G_i = delete_overlap(τ_i, G, R_2)
18:         if route(R_2, G_i) then
19:             τ_i = τ_i ∪ {R_2}
20:             routed = routed ∪ {R_2}
21:         end if
22:     end for
23:     i = i + 1
24: end for
```

[1]To make the O notations more readable, we do not show the set cardinality notation to represent the number of elements in the set, i.e., we will use V instead of $|V|$

1) Complexity: To analyze the complexity of this heuristic, we will assume only one request is added to the routed set in each iteration of the outer loop and one in each iteration of the inner loop in order to get an upper bound on runtime. For each iteration of the outer loop, we must construct an interval graph and find the maximum independent set. The interval graph can be created by a linear scan of the remaining requests: $O(\Delta)$. The independent set can then be calculated in $O(\Delta \log(\Delta))$. Assuming only one tree is found to be independent, it can be routed in $O(VD^2)$. The inner loop is also executed for each request, and it involves deleting overlapping links and attempting to route each of the new requests. Deleting the links can be done in $O(E)$ time, but once this is done, shortest paths must be recomputed in $O(V^3)$. Assuming only one request is routed in $O(VD^2)$, the total runtime of the inner loop is therefore $O(\Delta V^3 + VD^2)$. The total runtime is therefore $O(\Delta(\Delta \log(\Delta) + VD^2 + \Delta V^3 + VD^2))$, which can be simplified to $O(\Delta^2 V^3 + \Delta VD^2)$. This is a loose upper bound because during each iteration of the inner and outer loop, multiple requests may be added to the $routed$ set instead of just one, therefore reducing the number of iterations required. As we will see later, the runtime is comparable to that of *seqRWA*.

C. Simulated Annealing (SA)

In addition to the two simple heuristics discussed, we also implemented a meta-heuristic based on simulated annealing. The two heuristics presented so far build a single solution piece by piece. We can obtain better solutions if we explore a number of different solutions iteratively instead of stopping with a single solution. For example, given an initial solution, we can make small perturbations, reevaluate the cost, and move to solutions with lower cost. We can repeat this iteratively until no more improvements can be achieved. The problem with this approach is that we may get stuck in a local minima. Simulated annealing [52] is a technique that can be used to avoid this problem. It allows solutions to move "uphill" in a controlled manner so as to escape from local minima.

Simulated annealing is based on the physical process of annealing a solid. To put some material into a low-energy state, it is first heated up, then cooled slowly. At each temperature, a "thermal equilibrium" is reached. The energy of the solid depends on the energy of the particles making up the solid. A low-energy state is equivalent to finding some configuration of particles that results in the lowest energy. This can be thought of as a combinatorial optimization problem. We can therefore mimic the annealing process to create solutions to combinatorial optimization problems.

The basic requirements for simulated annealing are as follows. First, the configuration needs to be specified, which can be thought of the set of particles in the physical object. Next, we need an initial temperature and an annealing schedule, or how the temperature is decreased over time. At each given temperature, the current configuration is perturbed to create a new configuration. If the cost of the new configuration is lower than the current one, we accept it and set the current configuration to the new configuration that was generated. If the new configuration has higher cost, we accept it with some probability. This allows the process to escape from local minima. The probability of accepting a higher cost configuration is higher with high temperature and decreases as the temperature cools. This allows an aggressive random search in the beginning that slowly turns into a focused local search. Typically, the Metropolis Algorithm is used to determine whether or not to accept new higher cost configurations. The probability distribution used is the Boltzmann distribution and is given as $e^{-\Delta E/T}$, where ΔE is the change in energy between the two configurations and T is the current temperature. At each temperature, a number of perturbations is made to the configuration. The number should be large enough for equilibrium to be reached at the current temperature. Once this is done, the temperature is decreased according to the annealing schedule. A typical annealing schedule is $T_n = \alpha T_{n-1}$, where $0.9 \leq \alpha \leq 0.99$. The meta-heuristic ends when some stopping condition is met, like the temperature is below a threshold, a maximum number of iterations is reached, or the solution has not improved in a number of temperature decreases. Simulated annealing theoretically converges to the optimal solution, but it does so asymptotically [53], so convergence will not likely happen in realistic scenarios.

Algorithm 4: Simulated Annealing

```
1: currentConfig = sort(Δ)
2: currentEnergy = IS(currentConfig)
3: iterations = 0
4: repeat
5:     for i = 0 to iterPerTemp do
6:         newConfig = perturb(currentConfig)
7:         newEnergy = IS(newConfig)
8:         if newEnergy > currentEnergy then
9:             diff = newEnergy − currentEnergy
10:            if e^{−diff/kT} > rand[0, 1] then
11:                currentConfig = newConfig
12:                currentEnergy = newEnergy
13:            end if
14:        else
15:            currentConfig = newConfig
16:            currentEnergy = newEnergy
17:        end if
18:    end for
19:    currentTemp = α ∗ currentTemp
20: until currentTemp ≤ 0 || iterations > MAX_ITER
```

Our simulated annealing heuristic (*SA*), shown in Algorithm 4, uses the *ISH* heuristic to generate solutions given a configuration and explores different orderings of requests. Recall that *ISH* begins by sorting the requests in the order of $|D|$. The ordering of requests will affect the final solution. Ordering by destination set size is a simple approach, but likely not the best. There will exist some ordering of the requests that results in the lowest cost solution using *ISH*. Given that there are $|\Delta|!$ different orderings, it is not feasible to try all of them. This is the search space for our simulated annealing approach. A configuration is therefore a specific permutation of the requests. *SA* calls *ISH* as a subroutine, but in this case *ISH* does not sort the requests as shown in Algorithm 2 since *SA* is exploring different orderings itself.

Our *SA* heuristic implements the standard simulated annealing technique. It starts by sorting the requests according to $|D|$ like *ISH*. The $perturb$ function performs a random

swap of two requests in Δ to change the ordering and get a new configuration. We also use a constant k, shown in line 10, known as the *Boltzmann constant*. The Boltzmann constant can be used to help normalize the ratio for better probabilities. The *SA* heuristic runs until the temperature value close to zero or some maximum number of configurations is reached. The main parameters are the number of iterations per temperature, the total number of iterations, the Boltzmann constant (k), and α (for the annealing schedule). We discuss how we selected their values in Section VI. We should also note that even if the *SA* heuristic found the optimal solution (i.e., the optimal ordering), it does not necessarily mean it is the optimal solution to the entire MCAR problem. The *SA* heuristic simply improves upon the suboptimal solutions found by *ISH*.

1) Complexity: The complexity of the simulated annealing heuristic is directly related to the complexity of *ISH*. The perturbation takes constant time, as does checking for uphill moves and updating the temperature. The number of iterations is the parameter that has the highest effect on the runtime of *SA*. For each iteration, *ISH* is computed.

V. Lower Bounds

In this section, we define a lower bound on the number of wavelengths required for multicast advance reservation. Our derivations are based on the work in [36] and [54]. We derive two lower bounds, one based on comparing the logical nodal degree and physical nodal degree of the nodes, LB_1, and one based on congestion of the links, LB_2. Let p_i be the physical nodal degree of Node i. The in-degree and out-degree are the same for each node. Let $\ell_i^{t,t+1}$ be the number of incoming and outgoing lightpaths at Node i during time interval $[t, t+1]$

$$\mathrm{LB}_1 = \max_t \{ \max_i \left\{ \left\lceil \frac{\ell_i^{t,t+1}}{p_i} \right\rceil \right\} \}.$$

The average number of lightpaths routed over each physical edge of Node i is given by $\frac{\ell_i^{t,t+1}}{p_i}$. At least one of these edges must therefore have $\left\lceil \frac{\ell_i^{t,t+1}}{p_i} \right\rceil$ lightpaths routed over it (we take the ceiling since the number of lightpaths must be an integer). The lower bound simply finds the node that requires the most wavelengths in any time interval. This lower bound does not take the routing of requests into account, therefore we propose another lower bound that does.

The next lower bound, LB_2, is derived by taking the routing of trees into account, providing a lower bound on congestion over any link, which is in turn a lower bound for the number of wavelengths required. Let ℓ_r be the minimum number of links required to route request R.

Lemma V.1: $\ell_r = (\min_d \{\mathrm{SP}(s_r, d)\}, d \in D_r) + |D_r| - 1$, where SP is the number of links on the shortest path.

Proof: Consider any tree T for request R routed over the network $G = (V, E)$. T is sourced at $s_r \in V$ and reaches all destinations in $D_r \subseteq V - \{s_r\}$. Let $d_i \in D_r$ be the shortest-path node from s_r on T. Note this is the shortest path based on T, not necessarily the shortest path based on G. Since d_i is the shortest path node, no other nodes in D_r can be on the path, and therefore no other nodes are reached by this path. The remaining $|D| - 1$ nodes can be reached at best in one hop (using $|D| - 1$ links). Let the number of links on the T from s_r to d_i be $h(s_r, d_i, T)$. The number of links used by this tree is therefore at least $h(s_r, d_i, T) + |D| - 1$.

Let d_j be the node with shortest hop distance from s_r to any node in D_r over G with hop distance $\mathrm{SP}(s_r, d_j)$ (it is possible $d_i = d_j$). We must have $h(s_r, d_i, T) \geq \mathrm{SP}(s_r, d_j)$. The minimum number of links is therefore at least $\mathrm{SP}(s_r, d_j) + |D| - 1$. ■

Let $A_{t,t+1}$ be the set of active requests during time interval $[t, t+1]$. The lower bound can then be defined as

$$\mathrm{LB}_2 = \max_t \left\{ \left\lceil \frac{\sum_{r \in A_{t,t+1}} \ell_r}{|E|} \right\rceil \right\}.$$

This lower bound finds the time interval with the greatest congestion and uses this as a lower bound for the number of wavelengths required. For each interval, we find the total minimum number of links required by all of the requests in the interval. The average number of requests using each physical edge is this total divided by the number of edges (we assume each wavelength can be used in a single direction). This congestion is a lower bound on the number of wavelengths required. The final lower bound we use will be $\mathrm{LB} = \max\{\mathrm{LB}_1, \mathrm{LB}_2\}$.

VI. Performance Evaluation

In this section, we evaluate the performance of our heuristics for the MCAR problem. We first compare them to the optimal solution from the ILP on a small six-node network, and then compare them to each other on four realistic networks. We also discuss how the heuristics compared to our lower bounds.

The parameters for the heuristics are as follows. We use $k = 3$ for *seqRWA*. Values larger than 3 did not result in any better performance. For the simulated annealing parameters, we used 200 iterations per temperature, 15 000 iterations in total, $k = 6$ for the Boltzmann constant, and $\alpha = 0.9$ (the initial temperature is 1).

A. ILP Results

In order to get results for the ILP we have to run it over small networks due to its complexity. The network used for comparison of the ILP and heuristics is shown in Fig. 4. The ILP was solved with CPLEX 12.0. 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 ILPs. Each point on the graphs is the average of 20 individual runs. We ran the ILP for a maximum of 24 h each run. There were instances (the most was 3 of 20 for $\tau(\Delta) = 0.4, |\Delta| = 30$) where it hit the 24-h limit without finding the optimal solution. In this case, we just used the feasible solution. It is likely, given

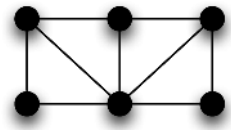


Fig. 4. Small mesh network used to get ILP results.

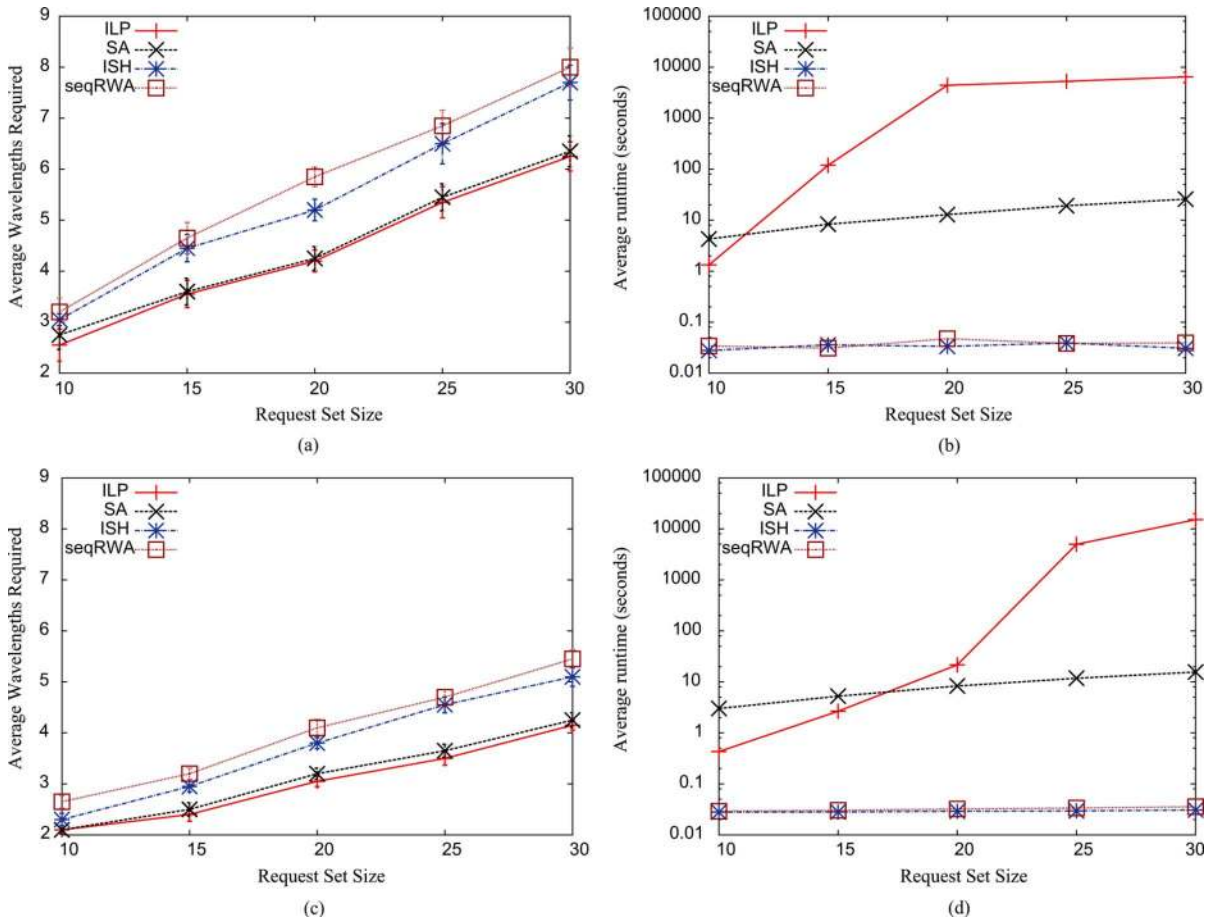


Fig. 5. Comparison of the optimal results provided by the ILP and results provided by the heuristics over the network shown in Fig. 4. Graphs (a) and (b) are results obtained with time correlation of 0.7, and graphs (c) and (d) are from a time correlation of 0.4. (a) Average wavelength usage ($\tau(\Delta) = 0.7$). (b) Average runtimes ($\tau(\Delta) = 0.7$). (c) Average wavelength usage ($\tau(\Delta) = 0.4$). (d) Average runtimes ($\tau(\Delta) = 0.4$).

the size of the problem, that this feasible solution is also the optimal solution (this feasible solution was never higher than the corresponding solution *SA* found).

We compare the ILP and the heuristics for different request set sizes and time correlations of 0.4 and 0.7. For each request set size, the number of multicast destinations for individual multicast requests is set to two. The source and destination nodes are chosen with a uniform random distribution. The results are shown in Fig. 5 with a 95% confidence interval for the average wavelengths required. We did not include the confidence interval in the graphs for average runtimes because the ILP runtimes varied greatly, leading to large intervals. The intervals for runtimes of the heuristics were small. Both time correlation values show similar results. We observe from Fig. 5(a) and (c) that the simulated annealing meta-heuristic is able to achieve optimal or close-to-optimal solutions for the given set sizes. We cannot, however, make the same conclusion about much larger set sizes or networks. While this is a good indication that the simulated annealing heuristic performs well, it does not mean that it always provides a solution so close to the optimal solution as the requests sizes get larger. Fig. 5(b) and (d) compares average runtimes for the different heuristics and the ILP (notice the log-scale for the y-axis). *seqRWA* and *ISH* have similar subsecond runtimes. Simulated annealing has runtimes in the tens of seconds that grow slowly as the request set size increases. The ILP runtime, as expected, grows rapidly. This shows that the ILP is not practical, given the runtimes for the small set sizes and small network.

B. Comparison of MCAR Heuristics

In this section, we use the realistic networks shown in Fig. 6 to evaluate our heuristics. The results are shown in Table III. The table lists the number of nodes, edges, and average nodal degree of each network. The results are shown for a request set size ($|\Delta|$) of 100 requests (results are similar for other sizes). Source and destination nodes were uniformly distributed, and the size of the destination set ($|D|$) was uniformly distributed between two and four. The results are grouped by time correlation value and heuristic. The w columns represent the average number of wavelengths required, and the ψ column is the average value of the reuse metric defined in Section II. The results are an average of 30 runs. We computed the 95% confidence intervals for all the results, but they are not included here since they are negligible. We ran the heuristics for set sizes of 30 and 60 with similar results, so they are not presented here.

For the ATT and NSFnet networks, we observe that in general across time correlations, ISH provides about a 9%–12% improvement in cost compared to *seqRWA*. SA provides about a

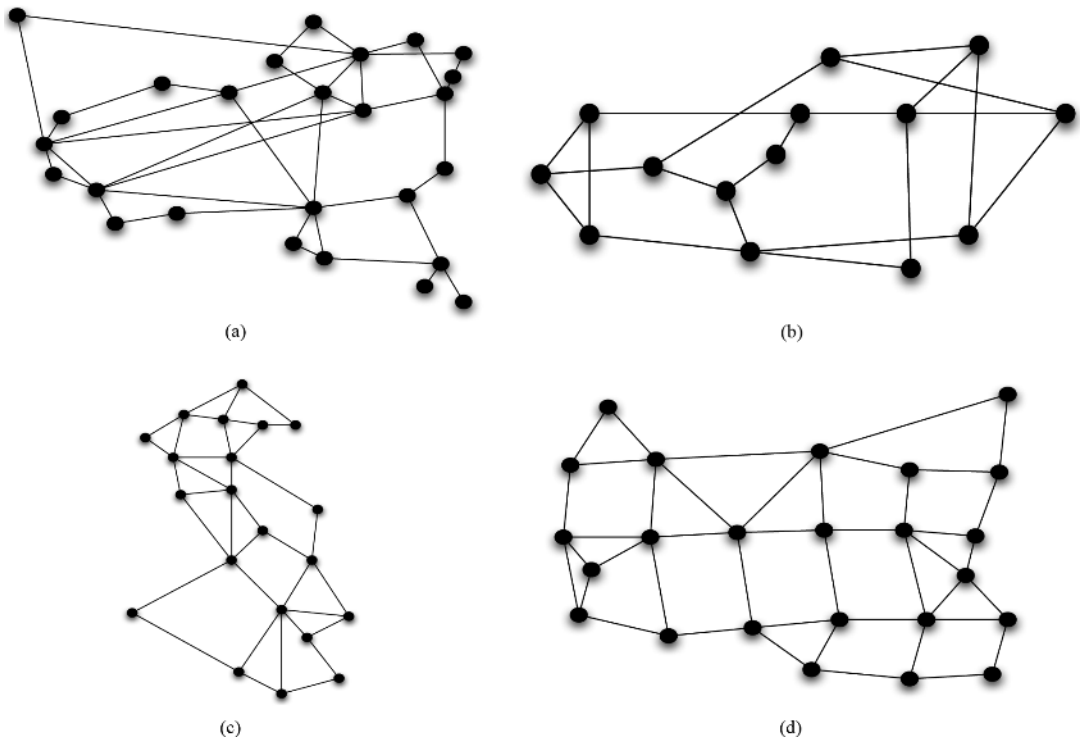


Fig. 6. WDM mesh networks used for evaluation of proposed heuristics. Link weights not shown for clarity. (a) Scaled AT&T network. (b) 14-node NSFnet. (c) Italian WDM network. (d) 24-node network.

TABLE III
COMPARISON OF MCAR HEURISTICS OVER THE TOPOLOGIES SHOWN IN FIG. 6. THE REQUEST SET SIZE ($|\Delta|$) IS 100

Network	V	E	δ	$\tau(\Delta) = 0.1$						$\tau(\Delta) = 0.4$						$\tau(\Delta) = 0.7$					
				SA		ISH		seqRWA		SA		ISH		seqRWA		SA		ISH		seqRWA	
				w	ψ	w	ψ	w	ψ	w	ψ	w	ψ	w	ψ	w	ψ	w	ψ	w	ψ
ATT	27	41	3.0	5.2	0.73	6.1	0.73	7.0	0.60	11.2	0.49	13.4	0.48	14.7	0.34	17.9	0.24	20.2	0.24	23.5	0.15
NSFnet	14	21	3.0	4.7	0.80	5.7	0.80	6.3	0.72	10.9	0.57	12.6	0.55	13.9	0.44	18.4	0.30	20.7	0.29	22.8	0.22
Italy	21	36	3.4	4.1	0.76	5.2	0.75	6.6	0.60	9.6	0.52	11.1	0.51	14.3	0.35	16.0	0.27	18.0	0.26	22.3	0.17
24-node	24	43	3.6	3.9	0.75	4.6	0.75	6.1	0.60	8.4	0.52	9.7	0.51	12.4	0.35	14.1	0.26	15.9	0.26	20.2	0.16

14% improvement over *ISH* and a 22%–24% improvement over *seqRWA*. For the Italian and 24-node networks, *ISH* is about 13%–15% better than *seqRWA*, SA is about 22% better than *ISH*, and *SA* is about 32% better than *seqRWA*.

We observe that the heuristics perform similarly on ATT and NSFnet. The heuristics achieve better results on the Italian and the 24-node network, which also have similar performance to one another. This can be explained by the average nodal degree of the networks. With a higher nodal degree, it is easier to find spatial reuse in the network. The heuristics provide better results with a higher nodal degree both in terms of absolute value and how they improved compared to each other.

From the reuse metric columns in Table III, we observe that the values are similar for *ISH* and *SA*, while *seqRWA* has a lower value. *ISH* and *SA* achieve similar amounts of temporal wavelength reuse. Even though they have similar reuse metric values, *SA* still improves the minimum cost compared to *ISH*. This seems to imply that *ISH* is able to take advantage of time independence and temporal reuse well, but *SA* is able to find better spatial reuse by permuting the ordering of requests. One area of future work may be to modify *ISH* to improve its ability to find spatial reuse. We could, for example, modify the inner loop to make more intelligent selections of requests instead of using a simple linear scan.

We will briefly describe the runtimes of the heuristics, which are not shown in the table. These heuristics were run on a machine with a Core2 Quad Core CPU at 2.66 GHz with 8 GB of RAM. First, since the runtimes are highly dependent on the network size ($|V|$), the runtimes are highest for ATT and lowest for NSFnet. The patterns are similar for each network, however. At low time correlation values, *ISH* runs faster than *seqRWA* even though *ISH* has higher complexity. This is because with low time correlation, *ISH* is able to find more time-independent requests in each iteration, significantly reducing the number of

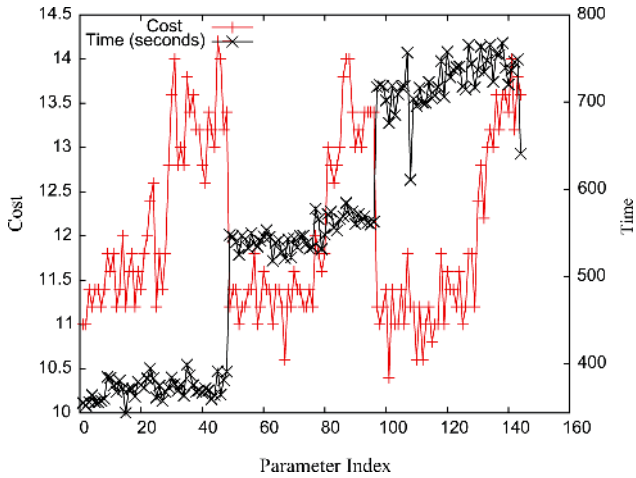


Fig. 7. Comparison of runtime and wavelengths required for different *SA* parameters. Each point on the x-axis represents a different combination of parameters.

iterations required, as discussed in the complexity analysis of *ISH*. For example, for the Italian network with $\tau(\Delta) = 0.1$, the runtime of *ISH* is 220 ms, while *seqRWA* is 330 ms. As the time correlation increases, so does *ISH*'s runtime. When $\tau(\Delta) = 0.7$, the runtime increases to 336 ms. In all cases, *seqRWA* and *ISH* achieve subsecond runtimes (a max of about 400 ms for *seqRWA* with the ATT network). The *SA* heuristic has much longer runtimes because it iterates *ISH* many times. Again, since *ISH* is used by *SA*, *SA* runs much faster for low time correlation values on the same network. For example, on the Italian network with $\tau(\Delta) = 0.1$, *SA* runs in 436 s, while with $\tau(\Delta) = 0.7$, it runs in 2665 s (44 min). The maximum runtime of *SA* is on the ATT network with $\tau(\Delta) = 0.7$, which is 4824 s (80 min). The runtimes between *ISH* and *seqRWA* are essentially negligible considering this computation occurs offline. It is therefore easy to choose *ISH* over *seqRWA* given the performance improvement. *SA*'s runtime is significantly longer than *ISH*, especially for high time correlation. Given that the advance reservation requests are typically made for periods of days or weeks, the runtime of *SA* is still reasonable. It is possible to use an algorithm that runs in 80 min, worst case, to schedule a batch of requests for the next 24 h.

C. Simulated Annealing Parameters

To find the best parameters for our simulated annealing metaheuristic, we ran simulations for different combinations of parameters. We used NSFnet in the analysis with a request set size of 60 requests and a time correlation of 0.7. The results of these experiments are shown in Fig. 7. Each point on the x-axis represents some combination of input parameters. We varied the α value, the Boltzmann constant (k), the number of iterations per temperature, and the total number of iterations. We varied α from 0.9 to 0.99, the Boltzmann constant from 4 to 10, the number of iterations per temperature from 100 to 300, and the total number of iterations from 10 000 to 20 000. These different combinations led to 108 different parameter settings. Each of these were run five times, and the average cost and runtimes were recorded as shown in the figure. The runtimes are grouped into three separate ranges; these correspond to different settings for the total number of iterations. We observe the cost also has two distinct ranges for each runtime range. These correspond to lower and higher values of the Boltzmann constant, with lower values providing better results. This parameter has the greatest impact on overall performance of *SA*.

We choose point 68 on the x-axis, which corresponds to the settings discussed earlier, since it has the second lowest average cost and the runtime is part of the medium group. In addition to these different parameter settings, we also tried the annealing schedule defined as $T_n = \frac{T_{\text{init}}}{1+\alpha T_{n-1}}$. This schedule resulted in worse results than the schedule we selected.

TABLE IV
OPTIMAL SOLUTION COMPARED TO OUR LOWER BOUNDS FOR THE NETWORK SHOWN IN FIG. 4

Requests	LB	ILP	SA	ILP/LB	SA/LB
$\|\Delta\| = 30,\ \tau(\Delta) = 0.7$	6.0	6.2	6.2	3.3%	3.3%
$\|\Delta\| = 30,\ \tau(\Delta) = 0.4$	3.8	4.2	4.27	10.5%	12.2%
$\|\Delta\| = 25,\ \tau(\Delta) = 0.7$	5.2	5.4	5.53	3.8%	6.4%
$\|\Delta\| = 25,\ \tau(\Delta) = 0.4$	3.27	3.4	3.6	4.1%	10.2%
$\|\Delta\| = 20,\ \tau(\Delta) = 0.7$	4.13	4.27	4.33	3.2%	4.8%
$\|\Delta\| = 20,\ \tau(\Delta) = 0.4$	2.8	3.07	3.2	9.5%	14.2%

D. Lower Bound Results

In this section, we discuss how our heuristics compared to our lower bounds. As we mentioned earlier, we defined two lower bounds and take the maximum of the two as the actual bound, LB. This lower bound is not the actual minimum number of wavelengths required, just a theoretical lower bound. We first compare our ILP to the lower bound to show the bound is reasonable, then compare our heuristics to the bound.

Table IV compares the ILP optimal solution and SA heuristic to the lower bound on the small six-node network shown in Fig. 4. The ILP and SA results are those shown in Fig. 5. The first three columns show the wavelengths required, while the last two show the percentage difference between the ILP and the LB and SA and the LB. We observe the lower bound is pretty accurate with a maximum of 10.5% difference from the optimal solution. The table also shows that the SA heuristic provides good solutions for this small network (as shown in Fig. 5 as well).

Lastly, we compare the lower bounds and heuristics over the networks shown in Fig. 6. The results are shown in Table V. The table is structured similar to Table III. We group the results by time correlation values. The *SA* column is the number of wavelengths required from the *SA* heuristic, *LB* is the lower bound, and *Diff.* is the percentage difference between the two. The table shows that percentage varies between 40% and 100%. Again, this is just a theoretical lower bound, not the actual minimum number of wavelengths required. As network sizes increase, the bound also becomes less accurate because the possible routing gets much more complicated. In the worst case, *SA* provides a solution that is two times the cost of the lower bound. Given that

TABLE V
COMPARISON OF SA HEURISTIC TO THE LOWER BOUND OVER THE TOPOLOGIES SHOWN IN FIG. 6. THE REQUEST SET SIZE ($|\Delta|$) IS 100

Network	V	E	δ	$\tau(\Delta) = 0.1$			$\tau(\Delta) = 0.4$			$\tau(\Delta) = 0.7$		
				SA	*LB*	*Diff.*	*SA*	*LB*	*Diff.*	*SA*	*LB*	*Diff.*
ATT	27	41	3.0	5.17	3.7	39.6%	11.2	7.93	41.2%	17.87	11.77	51.8%
NSFnet	14	21	3.0	4.73	3.1	52.7%	10.87	6.17	76.2%	18.4	10.33	78.1%
Italy	21	36	3.4	4.1	2.93	39.8%	9.63	4.87	97.9%	16.0	7.83	104.3%
24-node	24	43	3.6	3.87	2.3	68.1%	8.43	4.33	94.6%	14.13	7.03	100.9%

the lower bound is conservative, it is a good indication that *SA* performs well.

We present one final note about the bound computation. We found that with normal static demands for immediate arrival requests, the second lower bound typically provided higher values, meaning link congestion had a big impact on the minimum number of wavelengths. When time independence with advance reservation was introduced, we found that the first bound typically provided the higher value. The congestion is lowered due to the fact that there are fewer requests active at any time interval. This means that the average nodal degree of the network has significant impact on number of wavelengths required, which explains why, in Table III, Italy and 24-node networks have better results than the other two.

VII. CONCLUSION

In this paper, we have investigated the multicast advance reservation problem and showed that the problem is NP-complete. We formulated it as an ILP, presented three efficient heuristics to solve the problem, and derived lower bounds on the number of wavelengths required. Our simulated annealing heuristic provides close-to-optimal solutions when compared to the ILP for a small network. In realistic network, *SA* achieves up to a 22% improvement over *ISH* and a 33% improvement over *seqRWA*. The runtimes for the *SA* heuristic are significantly higher, but these computations are performed offline and at most daily, so the runtimes of *SA* are reasonable given this constraint. The *SA* provided, on average, solutions 1.5–1.8 times above our conservative theoretical lower bound for real-world networks.

Considering the existence of different approximation algorithms for the Steiner tree problem, one area of future work is investigating the possibility of creating an approximation algorithm for the MCAR problem. We will also investigate stricter lower bounds, possibly by exploring a more graph-theoretic approach to obtain bounds for congestion. Because we are dealing with transparent optical networks, another important area of future work is impairment awareness.

We will also consider an alternative formulation to the problem that is consistent with a batch scheduling approach. Rather than minimizing resources, we will maximize the number of accepted connections given a fixed number of resources. The ILP and heuristics can be adapted for this case.

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, Oct. 1995.

[3] 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, Jul. 1992.

[4] 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.

[5] R. Malli, X. Zhang, and C. Qiao, "Benefit of multicasting in all-optical networks," in *Proc. SPIE All-Opt. Netw.*, Nov. 1998, pp. 209–220.

[6] 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.

[7] G. N. Rouskas, "Optical layer multicast: rationale, building blocks, and challenges," *IEEE Netw.*, vol. 17, no. 1, pp. 60–65, Jan.–Feb. 2003.

[8] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computions: Proceedings of Symposium on the Complexity of Computer Computations*, ser. IBM Research Symposia, R. E. Miller and J. W. Thatcher, Eds. New York: Plenum, 1972, pp. 85–103.

[9] 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.

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

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

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

[13] 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.

[14] 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.

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

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

[17] 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, Oct. 2002.

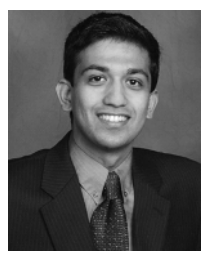
[18] B. Chen and J. Wang, "Efficient routing and wavelength assignment for multicast in WDM networks," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 1, pp. 97–109, Jan. 2002.

[19] 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.

[20] Y. Xin and G. N. Rouskas, "Multicast routing under optical layer constraints," in *Proc. IEEE INFOCOM*, Mar. 2004, vol. 4, pp. 2731–2742.
[21] J. Zheng and H. T. Mouftah, "Routing and wavelength assignment for advance reservation in wavelength-routed WDM optical networks," in *Proc. IEEE ICC*, 2002, vol. 5, pp. 2722–2726.
[22] J. Kuri, N. Puech, M. Gagnaire, E. Dotaro, and R. Douville, "Routing and wavelength assignment of scheduled lightpath demands," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 8, pp. 1231–1240, Oct. 2003.
[23] F. Arshad, S. R. Ramay, S. Tanwir, L. Battestilli, and S. M. H. Zaidi, "Advance reservation and dynamic scheduling of point to multipoint lightpaths," in *Proc. Int. Symp. High Capacity Opt. Netw. Enabling Technol.*, Nov. 2008, pp. 69–74.
[24] P. Pavarangkoon, A. Gunabhibal, C. Pomavalai, and R. Varakulsiripunth, "An efficient dynamic multicast routing algorithm with advance resource reservation awareness," in *Proc. Int. Conf. Adv. Commun. Technol.*, Oct. 2004, vol. 2, pp. 651–655.
[25] D. Chakraborty, C. Pornavalai, G. Chakraborty, and N. Shiratori, "Distributed routing for dynamic multicasting with advance resource reservation information," in *Proc. Int. Conf. Inf. Netw.*, 2001, pp. 603–610.
[26] "Worldwide LHC computing grid," Feb. 2010 [Online]. Available: http://lcg.web.cern.ch/lcg/
[27] J. Zheng, B. Zhang, and H. T. Mouftah, "Toward automated provisioning of advance reservation service in next-generation optical Internet," *IEEE Commun. Mag.*, vol. 44, no. 12, pp. 68–74, Dec. 2006.
[28] A. G. Greenberg, R. Srikant, and W. Whitt, "Resource sharing for book-ahead and instantaneous-request calls," *IEEE/ACM Trans. Netw.*, vol. 7, no. 1, pp. 10–22, Feb. 1999.
[29] J. Kuri, N. Puech, M. Gagnaire, and E. Dotaro, "Routing foreseeable lightpath demands using a tabu search meta-heuristic," in *Proc. IEEE GLOBECOM*, Nov. 2002, pp. 2803–2807.
[30] J. Kuri, N. Puech, and M. Gagnaire, "Diverse routing of scheduled lightpath demands in an optical transport network," in *Proc. Design Rel. Commun. Netw.*, Oct. 2003, pp. 69–76.
[31] B. Wang, T. Li, X. Luo, Y. Fan, and C. Xin, "On service provisioning under a scheduled traffic model in reconfigurable WDM optical networks," in *Proc. IEEE BroadNets*, Oct. 2005, vol. 1, pp. 13–22.
[32] E. He, X. Wang, and J. Leigh, "A flexible advance reservation model for multi-domain WDM optical networks," in *Proc. IEEE BroadNets*, Oct. 2006, pp. 1–10.
[33] E. He, X. Wang, V. Vishwanath, and J. Leigh, "AR-PIN/PDC: Flexible advance reservation of intradomain and interdomain lightpaths," in *Proc. IEEE GLOBECOM*, Dec. 2006, pp. 1–6.
[34] M. Gagnaire, M. Koubaa, and N. Puech, "Network dimensioning under scheduled and random lightpath demands in all-optical WDM networks," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 9, pp. 58–67, Dec. 2007.
[35] T. D. Wallace, A. Shami, and C. Assi, "Scheduling advance reservation requests for wavelength division multiplexed networks with static traffic demands," *IET Commun.*, vol. 2, no. 8, pp. 1023–1033, 2008.
[36] N. Skorin-Kapov, "Heuristic algorithms for the routing and wavelength assignment of scheduled lightpath demands in optical networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 2–15, Aug. 2006.
[37] C. V. Saradhi and M. Gurusamy, "Scheduling and routing of sliding scheduled lightpath demands in WDM optical networks," in *Proc. OFC*, Mar. 2007, pp. 1–3.
[38] C. V. Saradhi, J. C. Wei, M. Shujing, and M. Gurusamy, "Circular arc graph based algorithms for routing scheduled lightpath demands in WDM optical networks," in *Proc. IEEE BroadNets*, Oct. 2005, pp. 320–322.
[39] S. Naiksatam, S. Figueira, S. A. Chiappari, and N. Bhatnagar, "Analyzing the advance reservation of lightpaths in lambda-grids," in *Proc. IEEE Int. Symp. Cluster Comput. Grid*, May 2005, vol. 2, pp. 985–992.
[40] X. Yang, L. Shen, A. Todimala, B. Ramamurthy, and T. Lehman, "An efficient scheduling scheme for on-demand lightpath reservations in reconfigurable WDM optical networks," in *Proc. OFC*, Mar. 2006, p. 3.
[41] T. D. Wallace and A. Shami, "Connection management algorithm for advance lightpath reservation in WDM networks," in *Proc. IEEE BroadNets*, Sep. 2007, pp. 837–844.
[42] S. Tanwir, L. Battestilli, H. Perros, and G. Karmous-Edwards, "Dynamic scheduling of network resources with advance reservations in optical grids," *Int. J. Netw. Manage.*, vol. 18, no. 2, pp. 79–105, 2008.
[43] E. Escalona, S. Spadaro, J. Comellas, and G. Junyent, "Advance reservations for service-aware GMPLS-based optical networks," *Comput. Netw.*, vol. 52, no. 10, pp. 1938–1950, 2008.
[44] J. Kuri, N. Puech, and M. Gagnaire, "Diverse routing of scheduled lightpath demands in an optical transport network," in *Proc. Design Rel. Commun. Netw.*, Oct. 2003, pp. 69–76.
[45] A. Jaekel and Y. Chen, "Routing and wavelength assignment for prioritized demands under a scheduled traffic model," in *Proc. IEEE BroadNets*, Oct. 2006, pp. 1–7.
[46] A. Jaekel, "Lightpath scheduling and allocation under a flexible scheduled traffic model," in *Proc. IEEE GLOBECOM*, Dec. 2006, pp. 1–5.
[47] B. Wang, T. Li, X. Luo, and Y. Fan, "Traffic grooming under a sliding scheduled traffic model in WDM optical networks," presented at the IEEE BroadNets, 2004.
[48] B. Wang, Y. Fan, and X. Luo, "Multicast service provisioning under a scheduled traffic model in WDM optical networks," presented at the IASTED Int. Conf. Commun. Comput. Netw., 2005.
[49] 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.
[50] H. Takahashi and A. Matsuyama, "An approximate solution for the Steiner problem in graphs," *Math. Japonica*, vol. 24, no. 6, pp. 573–577, 1980.
[51] U. I. Gupta, D. T. Lee, and J. Y.-T. Leung, "Efficient algorithms for interval graphs and circular-arc graphs," *Networks*, vol. 12, no. 4, pp. 459–467, 1982.
[52] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[53] R. A. Rutenbar, "Simulated annealing algorithms: An overview," *IEEE Circuits Devices Mag.*, vol. 5, no. 1, pp. 19–26, Jan. 1989.
[54] R. Ramaswami and K. N. Sivarajan, "Design of logical topologies for wavelength-routed optical networks," *IEEE J. Sel. Areas Commun.*, vol. 14, no. 5, pp. 840–851, Jun. 1996.

Neal Charbonneau (S'08–M'10) received the B.S. and M.S. degrees in computer science from the University of Massachusetts, Dartmouth, in 2008 and 2010, respectively.

He is currently with the MITRE Corporation, Bedford, MA. 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, Mysore, India, in 1999, and the M.S. and Ph.D. degrees in computer science from the University of Texas at Dallas in 2001 and 2004, respectively.

He is an Associate Professor of computer and information science with the University of Massachusetts, Dartmouth. He is currently a Visiting Scientist with the Research Laboratory of Electronics (RLE), Massachusetts Institute of Technology (MIT), Cambridge. He is the coauthor of the book *Optical Burst Switched Networks* (Springer, 2005). His primary areas of research include design and analysis of architectures and protocols for optical and wireless networks.

Dr. Vokkarane currently serves on the Editorial Board of the IEEE COMMUNICATION LETTERS. He has been on the Technical Program Committee (TPC) of several IEEE conferences including INFOCOM, ICC, GLOBECOM, ICCCN, HSN, and ANTS, and served as TPC Co-Chair for the Optical Networks and Systems symposia at ICCCN 2007 and 2010, GLOBECOM 2011, and ICC 2012 and INFOCOM High-Speed Networks (HSN) Workshop 2011. He is a recipient of the Texas Telecommunication Engineering Consortium Fellowship for 2002–2003 and the University of Texas at Dallas Computer Science Dissertation of the Year Award for 2003–2004. He is a recipient of the Best Paper Award at the IEEE GLOBECOM 2005 Symposium on Optical Systems and Networks and the University of Massachusetts Dartmouth Scholar of the Year Award in 2011.