This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2010 proceedings.

Multicast Advance Reservation RWA Heuristics in Wavelength-Routed Networks

Neal Charbonneau and Vinod M. Vokkarane
Department of Computer and Information Science, University of Massachusetts, Dartmouth, MA
Email: {u_ncharbonne, vvokkarane}@umassd.edu

***Abstract*—In this paper we investigate the static multicast advance reservation (MCAR) problem for all-optical wavelength-routed WDM networks. Advance reservation 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 develop two efficient heuristics, *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 run simulations over real-world, large scale networks and compare them to our lower bound. We find the *SA* heuristic provides up to a 21% improvement over *ISH* (14% on average) on realistic networks. *SA* provides, on average, solutions 1.5-1.8x times the cost given by our conservative lower bound on large networks.** *

***Index Terms*—advance reservation, scheduled demands, RWA, multicast, heuristics, simulated annealing**

I. Introduction

Optical wavelength-routed WDM 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. Each fiber has a bandwidth 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. The combination of a route and wavelength is known as a *lightpath*. In a single-hop, or all-optical, WDM system, the signal is transmitted all-optically through the network. In the absence of wavelength converters (which are expensive), a connection in a all-optical WDM network must use the same wavelength across all links. This is known as the *wavelength continuity constraint*. We also consider the *wavelength clash constraint*, which means that if a wavelength is used on a particular link, no other connection may use the same wavelength over the same link. The disadvantage of all-optical networks is that, in the absence of regenerators, the signal noise accumulates from physical layer impairments (e.g. cross-talk, ASE noise, four-wave-mixing, etc.) Impairment-aware routing can be used to deal with this issue, but we leave that to future work.

Traditionally, communication in a network is unicast, where a single source sends data to a single destination. In this work, 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. Example of multicast applications are video conferencing, interactive distance learning, streaming media, distributed data processing, storage area networks, e-Science, etc. 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 [1], [2], [3]. A unicast request is supported by the use of lightpaths, as previously discussed. To efficiently support multicast requests, the network must create *light-trees* [2]. 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 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 [4], 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 [5]. These switches are known as multicast capable OXCs (MC-OXC).

Two traffic models are usually considered for wavelength-routed networks: static and dynamic [6]. 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). The connections are assumed to be held for long periods of time. 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, among others [7], [8], [9]. Given these traffic models, the requests can be called *immediate reservation* (IR) requests. The resources must be allocated at the time the request arrives.

In this paper we consider a request type called *advance reservation* (AR) [10] requests, or scheduled lightpath demands [11]. An advance reservation request specifies that the data transmission starts some time in the future. Note, advance reservation requests can be either static or dynamic in nature. In this work we consider static advance reservation, where the advance reservation demands are known ahead of time. There are many instances where it is known bandwidth will be required 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

*This work was supported in part by the National Science Foundation (NSF) under grant CNS-0626798.

978-1-4244-5638-3/10/$26.00 ©2010 IEEE

reservation include distance learning, where at scheduled times HD video must be streamed to different sites, video conferencing, 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 [12], require reserving large amounts of bandwidth to transfer data sets to possibly multiple locations. The alternatives to using advance reservation requests are to either over-provision 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) [13], 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 non-optical networks in the past, (e.g. [14]), where the research focus was on was dynamically arriving calls. It was first proposed for WDM optical networks for unicast traffic in [10]. The authors defined three types of advance reservation. The most popular type used in the literature is *specified start time and specified duration* (STSD). As the names suggest, STSD requests specify both the start and duration of the request. A typical unicast request may be a two-tuple specifying a source and destination: (s, d), but a STSD advance reservation unicast request specifies a start time, α, and a duration, ω as well: (s, d, α, ω). This is the type of advance reservation request we consider in this paper. Kuri *et al.* [11] investigated the case of static STSD advance reservation unicast requests. They were referred to as scheduled lightpath demands (SLDs). Wang *et al.* [15] 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. Gagnaire *et al.* [16] were the first to propose algorithms for a mix of advance reservation and dynamic immediate reservation requests. Wallace *et al.* [17] proposed a new type of advance reservation based on STSD. Here, instead of specifying a start time, the request specifies a requested service time (RST) and the difference between the actual start time and RST must be minimized. The author in [18] presented some heuristics as well as lower bounds for the STSD advance reservation problem. Additional heuristics are also presented in [19] for the sliding window model and in [20] for STSD. The static multicast STSD problem was initially investigated in [21] where they presented two heuristics. They allow the requests' start times to be re-negotiated if the request cannot be accommodated. The work mentioned so far focuses on static advance reservation demands. There has also been work dealing with dynamic advance reservation demands, mostly from the grid community. We do not discuss this work here.

The multicast static advance reservation problem was initially proposed in [21]. We provide a more comprehensive evaluation of the problem. We propose two novel heuristics, a new reuse metric, and derive a lower bound for the number of wavelengths required. The lower bound gives us more confidence in the performance of our heuristics. We also formally show the problem is NP-complete and formulate it as an integer linear program in our technical report [22]. The paper is organized as follows. Section II defines the problem formally, our metrics, and traffic patterns. Section III presents the two heuristics. Two simple lower bounds are derived in Section IV. We evaluate our heuristics in Section V and conclude in Section VI.

II. 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 STSD [10] advance reservation request. We assume that each request requires one wavelength and uses only one wavelength (i.e. a source node cannot transmit separate wavelengths to reach different destinations). The static MCAR problem is defined as follows.

Definition $MCAR(G, \Delta)$: Given a network $G = (V, E)$ and a set, $\Delta = \{R_1, R_2, ..., R_n\}$, 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 the service provider.

RWA algorithms for IR requests try to take advantage of spatial wavelength reuse, meaning they try to reuse the same wavelength on different links in the network to help minimize the number of wavelengths required (either to minimize cost or blocking of future requests, depending on the traffic type). AR requests can take advantage of *temporal* wavelength reuse as well. If two requests do not overlap in time, then they may use the same wavelength on the same links.

A. Network Assumptions

We are assuming all-optical WDM networks. We assume there is a single link between each node and each wavelength can be used in only one direction (wavelength clash constraint). The light-trees are unidirectional. Once the signal enters the network, it is switched all optically. We assume wavelength converters are not available, so the wavelength

978-1-4244-5638-3/10/$26.00 ©2010 IEEE

continuity constraint applies, meaning the light-tree or light-path 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 time slots. Each time slot, T, represents some unit of real time (for example, 15 minutes). The request must start at the beginning of a time slot and its duration must be a multiple of the time slot length.

We do not consider physical layer impairments. Our goal is to define the problem and present initial solutions.

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 mean low, 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 $r_{e,w}$ to be 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:

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

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 is 0. As the reuse increases, that ratio decreases. ψ will be between 0 and 1, with higher values indicating more reuse.

III. Heuristics

We have proved the MCAR problem is NP-complete, so no efficient algorithm exists for it unless P=NP (the proof is omitted here due to space). We introduce two efficient heuristics in this section to solve the problem sub-optimally. We have also formulated the problem as an integer linear program, which can be used to find the optimal solution. Solving the ILP is not practical for large-scale networks, however, so we must use heuristics. For details of the NP-complete proof and the ILP see our technical report [22].

Before discussing each individual heuristic, we describe the algorithm we use to create Steiner trees for routing the 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 [23], which is a $2(1 - \frac{1}{|D|})$ approximation for Steiner trees.

The heuristic works by creating the tree incrementally. 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 on the tree. The runtime is $O(|D|^2|V|)$ (with shortest paths precomputed).

A. Independent Set Heuristic (ISH)

Our first heuristic is called the Independent Set Heuristic (ISH), shown in Algorithm 1. 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 [18], [19], [20], but this heuristic is unique. We use a similar structure to [18], but we use an interval graph to find more time independence.

The heuristic works as follows. The requests are sorted in descending order by $|D|$. For the i^{th} iteration of the outer loop we create a new set, τ_i, and select the next un-routed request, R. We then find the maximum number of un-routed 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). The request are independent in time with R, but may not be independent in time with each other. 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 [4], but for interval graphs there exists an $O(n\log n)$ algorithm [24]. 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 do 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 all be routed while sharing links. 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.

Algorithm 1 Independent Set Heuristic

```
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
```

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 time and tried running the inner loop in reverse. The combination presented here performed the best.

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 $O(VD^2)$ (using MPH). 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 $O(\Delta^2 V^3 + \Delta VD^2)$.

B. Simulated Annealing (SA)

In addition to *ISH*, we also implemented a meta-heuristic based on simulated annealing. Due to space limitations, we do not discuss simulated annealing here (see [25] for details).

Our simulated annealing heuristic (*SA*) 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 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* makes calls to *ISH*, but here *ISH* does not sort the requests as shown in Algorithm 1 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*. It performs a random swap of two requests in Δ to change the ordering and get a new configuration for iteration. The *SA* heuristic runs until the temperature is a value close to zero or some maximum number of configurations are 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 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 sub-optimal solutions found by *ISH*.

1) Complexity: Each iteration of *SA* runs *ISH* so the complexity of the simulated annealing heuristic is directly related to the complexity of *ISH* and the number of iterations.

IV. Lower Bounds

In this section we define a lower bound on the number of wavelengths required for static multicast advance reservation. Our derivations are based on the work in [18], [26]. 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].

$$LB_1 = max_t\{max_i\left\{\left\lceil \frac{\ell_i^{t,t+1}}{p_i} \right\rceil\right\}\}.$$

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 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, so 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.

[†]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 $|V|$

Lemma IV.1 $\ell_r = (min_d\{SP(s_r, d)\}, d \in D_r) + |D_r| - 1$, *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 $SP(s_r, d_j)$ (it is possible $d_i = d_j$). We must have $h(s_r, d_i, T) \geq SP(s_r, d_j)$. The minimum number of links is therefore at least $SP(s_r, d_j)+|D|-1$. ■

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:

$$LB_2 = max_t \left\{ \left\lceil \frac{\sum_{r \in A_{t,t+1}} \ell_r}{|E|} \right\rceil \right\}.$$

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 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 number of wavelengths required. The final lower bound we use will be $LB = max\{LB_1, LB_2\}$.

V. Performance Evaluation

In this section we evaluate the performance of our heuristics. We compare them to each other and to our lower bounds. We use the following simulated annealing parameters: 200 iterations per temperature, 15,000 iterations in total, $k = 6$ for the Boltzmann constant, and $\alpha = 0.9$ (the initial temperature is 1). These parameters were obtained empirically by running a number of simulations over NSFnet with 108 different combinations of parameters.

A. Comparison of Heuristics

In this section, we use four realistic networks to evaluate our heuristics. We describe them in the Table I along with the results (images omitted due to space). 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. We computed the 95% confidence intervals for all the results but they are not included here since they are negligible.

TABLE I
Comparison of MCAR heuristics. Request set size ($|\Delta|$) is 100.

Network	V	E	δ	$\tau(\Delta) = 0.1$				$\tau(\Delta) = 0.7$			
				SA		ISH		SA		ISH	
				w	ψ	w	ψ	w	ψ	w	ψ
ATT	27	41	3.0	5.2	0.73	6.1	0.73	17.9	0.24	20.2	0.24
NSFnet	14	21	3.0	4.7	0.80	5.7	0.80	18.4	0.30	20.7	0.29
Italy	21	36	3.4	4.1	0.76	5.2	0.75	16.0	0.27	18.0	0.26
24-node	24	43	3.6	3.9	0.75	4.6	0.75	14.1	0.26	15.9	0.26

Notice the large difference in the number of wavelengths required for the different time correlation values. With lower time correlation, there is a much larger amount of temporal wavelength reuse. We can see that on average across time correlations, SA provides about a 14% improvement over *ISH*, with a maximum for 21% improvement for the Italian network.

We can see 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 Table I, we see that the values of the reuse metric are similar for *ISH* and *SA*. *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 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 2.66GHz with 8GB 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* has lower runtimes than high time correlation. This is because with low time correlation, *ISH* is able to find more time independent requests in each iteration, significantly reducing the number of 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 220ms. As the time correlation increases, so does *ISH*'s runtime. When $\tau(\Delta) = 0.7$, the runtime increases to 336ms. In all cases, *ISH* achieve sub-second runtimes. The *SA* heuristic has much longer runtimes because it iterates *ISH* many times. Again, since *ISH* is used by *SA*, *SA* runs much

TABLE II
COMPARISON OF SA HEURISTIC WITH THE LOWER BOUND. ($|\Delta|$) IS 100.

Network	V	E	δ	$\tau(\Delta) = 0.1$			$\tau(\Delta) = 0.7$		
				SA	LB	$Diff.$	SA	LB	$Diff.$
ATT	27	41	3.0	5.2	3.7	39.6%	17.9	11.8	51.8%
NSFnet	14	21	3.0	4.7	3.1	52.7%	18.4	10.3	78.1%
Italy	21	36	3.4	4.1	2.9	39.8%	16.0	7.8	104.3%
24-node	24	43	3.6	3.9	2.3	68.1%	14.1	7.0	100.9%

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 seconds, while with $\tau(\Delta) = 0.7$, it runs in 2,665 seconds (44 minutes). The maximum runtime of *SA* is on ATT network with $\tau(\Delta) = 0.7$, which is 4824 seconds (80 minutes). Given that the advance reservation requests are typically made for periods of days or weeks, the runtime of *SA* is still reasonable.

B. Lower Bound Results

In this section we discuss how *SA* compared to our lower bound. As we mentioned earlier, we defined two lower bounds and take the maximum as the actual bound, LB.

The results are shown in Table II (structured similarly to Table I). 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 worst case, *SA* provides a solution that is two times the cost of the lower bound. Given that the lower bound is conservative, it is a good indication that *SA* performs well.

We found that with normal static immediate reservation demands, 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 higher impact on number of wavelengths required, which explains why, in Table I, Italy and 24-node networks have better results than the other two. This also suggests a direction for a better meta-heuristic. We can focus on minimizing the maximum logical to physical degree of any node to try to reduce the number of wavelengths required.

VI. CONCLUSION

We have proposed two efficient heuristics to solve the static multicast advance reservation problem along with lower bounds on the number of wavelengths required. In realistic networks, *SA* achieves up to a 21% improvement over *ISH*, with an average of 14% improvement. The runtimes for *SA* are much higher, but these computations are performed offline and at most daily, making the the runtimes of *SA* reasonable. The *SA* provided, on average, solutions 1.5-1.8x above our conservative theoretical lower bound.

REFERENCES

[1] R. Malli *et al.*, "Benefit of multicasting in all-optical networks," in *Proceedings, SPIE All-Optical Networks*, Nov 1998, pp. 209–220.

[2] L. H. Sahasrabuddhe *et al.*, "Light-trees: optical multicasting for improved performance in wavelength-routed networks," *IEEE Communications Magazine*, vol. 37, no. 2, pp. 67–73, Feb 1999.

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

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

[5] W. S. Hu *et al.*, "Multicasting optical cross connects employing splitter-and-delivery switch," *IEEE Photonics Technology Letters*, vol. 10, pp. 970–972, 1998.

[6] H. Zang *et al.*, "A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks," *SPIE Optical Networks Magazine*, vol. 1, pp. 47–60, 2000.

[7] G. Sahin *et al.*, "Multicast routing and wavelength assignment in wide-area networks," in *Proceedings, SPIE All-Optical Networks*, vol. 3531, 1998, pp. 196–208.

[8] S. Sankaranarayanan *et al.*, "Comprehensive performance modeling and analysis of multicasting in optical networks," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 9, pp. 1399–1413, Nov. 2003.

[9] Y. Xin *et al.*, "Multicast routing under optical layer constraints," in *Proceedings, IEEE INFOCOM*, vol. 4, Mar. 2004, pp. 2731–2742 vol.4.

[10] J. Zheng *et al.*, "Routing and wavelength assignment for advance reservation in wavelength-routed WDM optical networks," in *Proceedings, IEEE ICC*, vol. 5, 2002, pp. 2722–2726.

[11] J. Kuri *et al.*, "Routing and wavelength assignment of scheduled lightpath demands," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 8, pp. 1231–1240, Oct. 2003.

[12] LHC Computing Grid. [Online]. Available: http://lcg.web.cern.ch/lcg/

[13] J. Zheng *et al.*, "Toward automated provisioning of advance reservation service in next-generation optical Internet," *IEEE Communications Magazine*, vol. 44, no. 12, pp. 68–74, Dec. 2006.

[14] A. G. Greenberg *et al.*, "Resource sharing for book-ahead and instantaneous-request calls," *IEEE/ACM Transactions on Networking*, vol. 7, no. 1, pp. 10–22, 1999.

[15] B. Wang *et al.*, "On service provisioning under a scheduled traffic model in reconfigurable WDM optical networks," in *Proceedings, IEEE BroadNets*, vol. 1, Oct. 2005, pp. 13–22.

[16] M. Gagnaire *et al.*, "Network dimensioning under scheduled and random lightpath demands in all-optical WDM networks," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 58–67, Dec. 2007.

[17] T. D. Wallace *et al.*, "Scheduling advance reservation requests for wavelength division multiplexed networks with static traffic demands," *IET Communications*, vol. 2, no. 8, pp. 1023 –1033, Sept. 2008.

[18] N. Skorin-Kapov, "Heuristic algorithms for the routing and wavelength assignment of scheduled lightpath demands in optical networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 2–15, Aug. 2006.

[19] C. V. Saradhi *et al.*, "Scheduling and routing of sliding scheduled lightpath demands in WDM optical networks," in *Proceedings, Optical Fiber Communication Conference (OFC)*, Mar. 2007, pp. 1–3.

[20] ——, "Circular arc graph based algorithms for routing scheduled lightpath demands in WDM optical networks," in *Proceedings, IEEE BroadNets*, vol. 1, Oct. 2005, pp. 320–322.

[21] B. Wang *et al.*, "Multicast service provisioning under a scheduled traffic model in WDM optical networks," in *IASTED International Conference on Communications and Computer Networks*, 2005.

[22] N. Charbonneau *et al.*, "Static routing and wavelength assignment for multicast advance reservation in all-optical wavelength-routed WDM networks," *UMass Dartmouth Tech Report, UMASSD-CIS-TR-2009006*, Dec. 2009. [Online]. Available: http://www.umassd.edu/engineering/cis/reports/report.cfm?r=31

[23] H. Takahashi *et al.*, "An approximate solution for the steiner problem in graphs," *Math. Japonica*, 1980.

[24] U. I. Gupta *et al.*, "Efficient algorithms for interval graphs and circular-arc graphs," *Networks*, vol. 12, no. 4, pp. 459–467, 1982.

[25] S. Kirkpatrick *et al.*, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[26] R. Ramaswami *et al.*, "Design of logical topologies for wavelength-routed optical networks," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 5, pp. 840–851, Jun 1996.

978-1-4244-5638-3/10/$26.00 ©2010 IEEE