Tabu Search Meta-Heuristic for Static Manycast Routing and Wavelength Assignment over Wavelength-Routed Optical WDM Networks

Neal Charbonneau and Vinod M. Vokkarane
Department of Computer and Information Science, University of Massachusetts, Dartmouth, MA
E-mail: u_ncharbonne@umassd.edu and vvokkarane@umassd.edu

***Abstract*— This paper presents a tabu search meta-heuristic to solve the static manycast routing and wavelength assignment problem (MA-RWA). The problem is to route a set of static manycast requests over a wavelength-routed WDM network such that the number of wavelengths required is minimized. We present the details of a tabu search meta-heuristic for this problem and compare it to another MA-RWA heuristic called lambda path heuristic (LPH) as well as a multicast RWA heuristic. The tabu search meta-heuristic shows a 10% improvement over LPH and a 30-40% improvement over the multicast heuristic for various realistic networks.** [1]

Keywords: manycast, multicast, tabu search, WDM, wavelength-routing, and RWA.

I. Introduction

Future Internet applications, such as IPTV, cloud storage/computation, video conferencing, and peer-to-peer (P2P), will require large amounts of bandwidth and support for point-to-multipoint communication. To support these applications, the next-generation Internet will be based on optical networks that can provide huge amounts of bandwidth. Manycast [1], [2] is a communication paradigm that can support the point-to-multipoint nature of future applications, in addition to supporting traditional communication paradigms. Manycast supports communication from a sender to any k out of m ($k \leq m$) candidate destinations where the candidate destination set, $|D_c| = m$, is a subset of nodes in the network. If we change the parameters of the manycast request, we can also perform unicast ($k = m = 1$), multicast ($k = m > 1$) and anycast ($k = 1 < m$). Manycast is a powerful communication framework that is important for next-generation applications [3]. Since the future Internet will based on optical networks, it is important to support manycast over wavelength-routed networks.

In this work we will consider the static MA-RWA problem. In this problem we are given a set of manycast requests and for each request we must assign a route tree (or light-tree [4]) and a wavelength. The objective is to minimize the number of wavelengths required to satisfy all the manycast requests. We can define a manycast request as (s, D_c, k) where s is the source, D_c is the candidate destination set, and k is the number of nodes necessary to reach out of D_c. This is related to the multicast problem, but is more general. In multicast, the destinations are specified ahead of time, in manycast the destinations must be chosen. To solve the multicast problem, a Steiner tree must be generated, which has been shown to be NP-hard [2]. Because the destinations must be chosen in manycast, there are $\binom{|D_c|}{k}$ combinations of nodes to use in the creation of a Steiner tree. Since manycast is a generalization of multicast, it is also NP-hard.

Supporting manycast over optical networks is important because supporting point-to-multipoint communication with unicast results in wasted resources at the optical layer [5]. Solving MA-RWA will help in dimensioning and enabling future networks to support new applications. For example, consider a grid network with replicated services. MA-RWA can choose the appropriate destinations for long-term requests that are known in advance. Also, in the case of IPTV content distribution, where resources required for a certain time period across a core network may be known ahead of time, MA-RWA can create light-trees to satisfy the resource requests.

The paper is organized as follows: Section II discusses related work, Section III gives a formal definition of the problem while Section IV presents our tabu search meta-heuristic. Section V presents a performance evaluation and Section VI concludes the paper.

II. Related Work

Quorumcast, which is a specific case of manycast where $k = \left\lceil \frac{|D_c|}{2} \right\rceil$, was proposed by [1], [2]. Since then, a number of quorumcast routing algorithms have been proposed [1], [6], [7], [8]. Manycast has also been proposed over optical burst-switched networks [9], [10], [11], [12]. The main challenge for manycast over OBS is providing reliability despite random contentions. These works focus on dynamic traffic and distributed routing algorithms or unicast routing algorithms to provide reliable manycast for OBS. These approaches typically do not setup a route tree for each request. The authors in [13] propose an ILP and several heuristics for solving multi-resource manycast in mesh networks. In manycast we may consider that each node provides a single resource, so to reach k resources we must reach k nodes. Multi-resource manycast generalizes manycast by allowing nodes to provide more than a single resource. Recently, an anycast RWA algorithm was proposed for wavelength-routed networks [14]. Anycast is a specific instance of manycast where $k = 1 < m$. To the best of our knowledge, this is the first time static manycast routing and wavelength assignment has been proposed.

Tabu search is a meta-heuristic often used for combinatorial optimization problems. It explores the solution space for a number of iterations or until some other stopping criteria is met. An initial solution is first generated either randomly or by using another heuristic. Given the current solution, a neighborhood set of solutions is generated by performing simple *moves* from the current solution. The best solution from the neighborhood is then chosen as the current solution and the process continues. In order to avoid getting trapped in

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

local minima, a *tabu list* is maintained. The tabu list records moves that were used to generated selected solutions. These moves cannot be performed again as long as they are on the list. Tabu search meta-heuristics also use diversification and intensification steps. Diversification is typically used when the best solution has not improved after a number of iterations. Diversification typically generates a new solution elsewhere in the search space and the tabu search begins again there. Intensification can be used to perform a more thorough search of the neighborhood of a good solution that has been found. Details of tabu search can be found in [15]. There is no proof of convergence to an optimal solution, but tabu search often works quite well in practice.

Tabu search has been used to solve routing and wavelength assignment problems, see [16], [17], [18], among others.

III. Problem Definition

The static MA-RWA problem can be defined as follows. We are given a network $G = (V, E)$ and a set of manycast requests $M = \{(s_1, D_{1c}, k_1), (s_2, D_{2c}, k_2), ..., (s_n, D_{nc}, k_n)\}$, where $s_i \in V \forall i$, $D_{ic} \subset V \forall i$, and $k_i \leq |D_{ic}| \forall i$. We must then find a route tree, or light-tree, and wavelength assignment for each manycast request in M such that the number of wavelengths required is minimized. Since the requests are known ahead of time, this is done offline. We use a single route tree for each request and assume each request needs one wavelength. We assume wavelength converters are not available so each tree must satisfy the wavelength continuity constraint. In other words, each tree must use the same wavelength on all links. The other constraint is that no two light-trees can use the same wavelength over the same link.

We assume all nodes in the network are able to split an incoming signal to any number of output ports. These types of switches are known as multicast-capable optical cross connects (MC-OXCs). These can be implemented with splitter-and-delivery (SaD) [19] or tunable SaD switches [20]. We also do not consider impairment or power-awareness in this paper. In a realistic scenarios, especially with splitters, the power and signal-to-noise ratio should be taken into account for routing in optical networks. We have investigated this problem in previous work for manycast over OBS [11], [12], but this is out of the scope of this paper and we will investigate impairment-awareness in the future.

IV. Tabu Search

The tabu search heuristic that we propose uses another heuristic that we have created called the lambda path heuristic (LPH). Given a set of manycast requests, LPH orders them and then iterates over the set assigning a route tree and wavelength to each request in order. The ordering of requests has a significant impact on the solution. There must exist some optimal ordering or orderings that produce the least number of wavelengths required. Our tabu search heuristic attempts to find this optimal ordering by searching different permutations of orderings.

Algorithm 1 Lambda Path Heuristic for static MA-RWA.

```
1:  sort_desc(M)
2:  for all m in M do
3:      D = {}
4:      allTrees = list()
5:      while D_mc − D ≠ φ do
6:          T = (V', E') s.t. V' = {s_m}, E' = φ
7:          path = min{SP(s_m, u)} u ∈ D_mc − D
8:          Update(T, path)
9:          D = ∪{u}
10:         copy = 1
11:         while copy < k do
12:             path = min{SP(u_1, u_2)} u_1 ∈ V', u_2 ∈ D_mc − V'
13:             Update(T, path)
14:             copy = copy + 1
15:         end while
16:         T.cost = Σ_{i,j∈E'} c_{i,j}
17:         T.newWL = increasesWL(G, T)
18:         allTrees.append(T)
19:     end while
20:     T = min(allTrees)
21:     FirstFit(G, T)
22:     updateWeights(α, 1 − α)
23: end for
```

A. Lambda Path Heuristic (LPH)

We will first briefly describe the LPH heuristic then go into detail about our tabu search. Once the static requests are placed in some order, LPH is run over the set. To satisfy each individual request, we use a modified version of the improved path heuristic (IMP) [1]. We modify it to include wavelength assignment, an additional constraint to minimize wavelengths required, load-balancing, and to also iterate over more Steiner trees. The heuristic is shown in Algorithm 1. Before describing how it works, we will define the functions that it uses. First, the SP function finds the shortest path between the two nodes specified as parameters. The $Update$ function adds a path to the specified tree T. The $increasesWL$ function determines if assigning a wavelength to a tree, using First-Fit, would require an increase in the wavelength count given the wavelengths currently used for the previous requests. Lastly, the $updateWeights$ function is used for load-balancing. It updates the weight of each link according to: $\alpha+(1-\alpha)*\frac{c}{c_{max}}$, where c is the current number of wavelengths on the link, c_{max} is number of wavelengths on the most congested link and $0 \leq \alpha \leq 1$. Depending on the value of α, this helps achieve a degree of load-balancing in the network.

To satisfy a manycast request, there are $\binom{|D_c|}{k}$ possible combinations of nodes that can be used to create Steiner trees. This heuristic works by creating just $|D_c|$ Steiner trees using the shortest-path heuristic proposed in [21]. The shortest-path heuristic creates a Steiner tree by building it incrementally. It adds each new node to the tree using to the shortest path from any node already on the tree. $|D_c|$ Steiner trees are

created by forcing selection of the first node in Line 7 when building the tree. During the first iteration, the shortest-path node is selected, in the next iteration the next shortest-path node is selected, and so on. Selecting a different start node each time makes it likely that a different tree will be created each iteration. Each time a node is selected to be the initial node it is added to D. Iteration terminates when $D = D_c$. This ensures that all nodes in D_c are in at least one Steiner tree and that multiple trees are generated. The goal of this is to find a good Steiner tree without having to try all $\binom{|D_c|}{k}$ combinations of nodes. After the generation of each tree, Line 17 checks to see if assigning a wavelength according to First-Fit would result in an increase in wavelength count required. Once the trees have been generated, the heuristic chooses the minimum cost tree that requires no increase in wavelength count using First-Fit wavelength assignment. If there is no such tree, just the minimum cost tree is chosen. The cost is defined as the tree with the least number of links. The reasoning behind this cost function is that using smaller trees will leave more resources for future requests. After the tree is assigned the wavelength, the costs of the links are updated.

B. Tabu Search (TS)

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

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

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

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

Tabu List: the tabu list maintains a list of recently performed moves, where a move is the two indices, i and j, that were

Algorithm 2 Tabu Search Meta-Heuristic.

```
1:  input : iterations, frac, tenure, diverse, intense
2:  sort_desc(M)
3:  current = LPH(M)
4:  best = current
5:  itWOImprov = 0; divWOImprov = 0
6:  tabuList = {}
7:  nSize = |M|*(|M|-1)/2
8:  fracNSize = nSize * frac
9:  while iterations-- > 0 do
10:   neighborhood = {}
11:   curNSize = 0
12:   while curNSize < fracNSize do
13:     move = genRandomMove()
14:     M' = swap(move, current)
15:     neighborhood.add(LPH(M'))
16:     curNSize++
17:   end while
18:   current = getBest(tabuList, tenure, neighborhood)
19:   if current.cost < best.cost then
20:     best = current
21:     itWOImprov = 0; divWOImprov = 0
22:   else
23:     itWOImprov++
24:   end if
25:   if itWOImprov < diverse then
26:     continue
27:   end if
28:   if divWOImprov < intense then
29:     current = LPH(genRandomPerm(M))
30:     itWOImprov = 0; divWOImprov++
31:     tabuList.clear
32:   else
33:     toIntensify = getTop()
34:     while TRUE do
35:       fullNeighborhood = {}
36:       for i = 0 to i = |M| do
37:         for j = i + 1 to j = |M| do
38:           move = (i, j)
39:           M' = swap(move, toIntensify)
40:           neighborhood.add(LPH(M'))
41:         end for
42:       end for
43:       toIntensify = neighborhood.best
44:       if toIntensify.cost > best then
45:         break
46:       else
47:         best = toIntensify
48:       end if
49:     end while
50:     itWOImprov = 0; divWOImprov = 0
51:     current = best
52:     updateTop()
53:   end if
54: end while
```

swapped. Note that $(i, j) = (j, i)$. A move cannot be chosen to create a new solution if it is already on the tabu list unless it meets an aspiration criteria. In this case, if the tabu move would result in a solution better than the current best solution, it is allowed.

Diversification: this is typically used to try to explore other areas in the solution space. We use diversification if the solution has not improved after a number of iterations. Our diversification step simply randomly permutes the sequence of requests. The tabu list is also cleared.

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

The detailed algorithm can be found in Algorithm 2. The inputs to the algorithm (*iterations, frac, tenure, diverse, intense*) are the number of iterations, the fraction of the neighborhood size to explore, the tabu tenure, the number of iterations before diversifications, and the number of diversifications before intensification. The number of iterations before diversification and intensification are incremented as long as no better solution is found. The function $getRandomMove$ on Line 13 generates random indices to swap. The same two are never chosen during an iteration. The $getBest$ function on Line 18 will choose the best solution based on the tabu list. If the solution in the neighborhood is on the tabu list, it is not selected unless it meets the aspiration criteria: being better than the best solution seen so far. The move is also added to the tabu list. The $getTop$ function on Line 33 will find a solution in the list of top solutions that has not yet gone through intensification. Lastly, the $updateTop$ function on Line 52 will update the list of top five solutions, removing any if necessary.

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

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

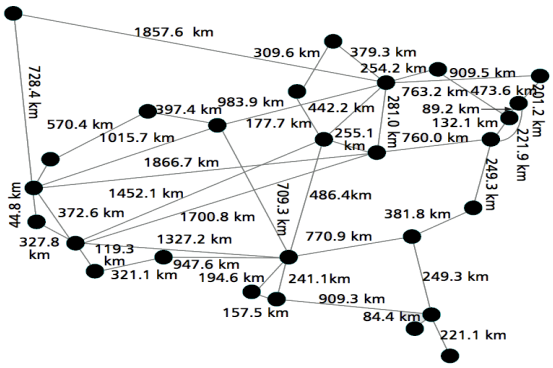
2) Complexity: In this section we discuss the complexity of our tabu search in terms of the complexity of generating neighborhood solutions. The number of neighbors created during each iteration directly impacts the runtime, since for each neighbor, LPH must be run. The LPH algorithm, once shortest paths have been computed, runs in $O(k|D_c|^2)$ for each individual request. After each tree is selected, the link weights are updated so the shortest paths must be computed again, which can be done in $O(|V|^3)$. Generating a single solution given a sequence therefore takes $O(|M| * (k|D_c|^2 + |V|^3))$. If the entire neighborhood space is searched each iteration, there are $O(|M|^2)$ neighbors generated, which would result in a runtime of $O(|M|^3 * (k|D_c|^2 + |V|^3))$ for each iteration. For large $|M|$, this is clearly not efficient, so we randomly generate small fractions of the full neighborhood set to keep the neighborhood space $O(|M|)$.

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

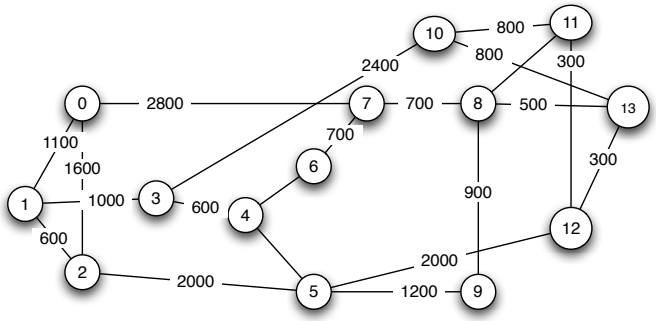
During the first iteration, some percentage of the neighborhood would be explored. The entire neighborhood consists of all possible combinations generated by swapping two elements. In this simple example, we can generate six solutions by swapping: (R_1, R_2), (R_1, R_3), (R_1, R_4), (R_2, R_3), (R_2,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

V. Evaluation

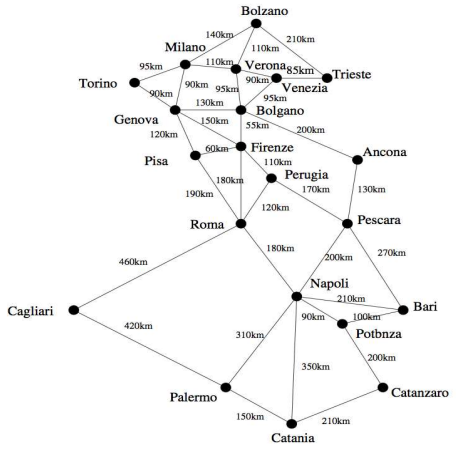
To evaluate our proposed tabu search meta-heuristic, we use it for static MA-RWA over four different network topologies. We ran extensive simulations on the AT&T network (scaled version), the NSF network, the Italian WDM network, and a 24-node mesh network shown in Fig. 1. We use the link distances for calculating the average tree delay, but routing



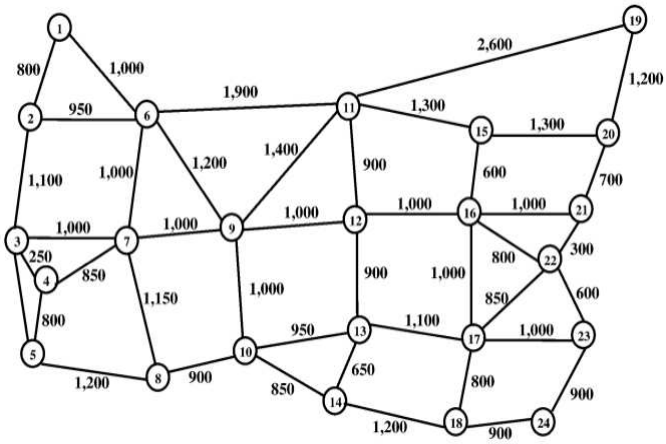
(a) Scaled AT&T network.



(b) 14 node NSFnet.



(c) Italian network.



(d) 24 node network.

Fig. 1. Networks used for heuristic evaluation.

is done based on hops. We generate a set of 150 requests for static MA-RWA. The source node for each request is uniformly distributed over all nodes in the network. For each request, the size of D_{ic} is uniformly distributed from $3, ..., D_c^m$ (a parameter representing the maximum candidate destination size) and $k = \left\lceil \frac{D_{ic}}{2} \right\rceil$. The destination nodes are also uniformly distributed across the network for each request.

To evaluate the effectiveness of our proposed tabu search, we compare it to a single run of LPH as well as a multicast heuristic called shortest path tree (SPT). For each request, SPT chooses k out of D_c that are closest to the source according to the shortest paths. It then uses the minimum path heuristic (MPH) [21] to create a multicast tree to these destinations. The main difference between SPT and LPH is that LPH considers multiple Steiner trees by including different nodes while SPT makes a single decision on which nodes to include. MPH is a 2-approximation for Steiner trees, so we use it as an upper-bound to show that improvements can be made by taking into account the ability to choose destinations dynamically.

In Table I we compare tabu search (TS), LPH, and SPT. The network characteristics are given in the table where V is the number of nodes, E is number of edges, δ is average nodal degree, and τ is average delay per edge (ms). We ran each heuristic with different maximum destination set sizes, D_c^m, and recorded the average number of wavelengths required, w_a, and average tree delay, d_a (ms), for 150 requests. The average tree delay is defined as the average delay from the root to each destination node. Referring to Algorithm 2, the parameters for the tabu search are as follows: $iterations = 1000$, $tenure = 20$, $frac = 0.06$, $diverse = 25$, and $intense = 2$. For both LPH and the tabu search $\alpha = 0.8$ for load-balancing. Most of these parameters were obtained empirically, this is discussed later in this section.

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

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

As expected, as the maximum destination set size decreases, the number of wavelengths required also decreases. The set size of 150 was chosen for demonstration purposes. We evaluated the heuristics on varying set sizes from 50 to 200 with similar results. We chose these four networks to represent realistic scenarios with varying nodal degrees. The

TABLE I

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

Netw.	V	E	δ	τ	$D_c^m = 10$						$D_c^m = 8$						$D_c^m = 6$					
					TS		LPH		SPT		TS		LPH		SPT		TS		LPH		SPT	
					w_a	d_a	w_a	d_a	w_a	d_a	w_a	d_a	w_a	d_a	w_a	d_a	w_a	d_a	w_a	d_a	w_a	d_a
ATT	27	41	3.0	3.4	31.7	11.5	35.8	11.7	47.2	10.7	30.7	11.1	33.7	11.1	46.1	10.7	29.1	11.6	32.9	11.7	44.6	11.3
NSFnet	14	21	3	4.3	39.2	9.9	43.6	11.4	55.7	8.4	34.9	9.6	39.0	9.3	48.3	8.3	32.6	9.6	36.7	9.4	45.6	8.5
Italy	21	36	3.4	0.6	33.0	1.7	37.2	1.7	52.1	1.5	32.8	1.7	36.9	1.7	53.5	1.5	29.3	1.7	33.3	1.7	48.9	1.5
24-node	24	43	3.6	3.9	28.3	11.3	31.8	11.4	42.7	10.2	27.0	11.5	30.1	11.2	40.2	10.2	25.7	11.6	29.0	11.5	38.1	10.7

networks represent backbone or long-haul networks for which wavelength-routed WDM network is a good candidate. We also calculated 95% confidence intervals for all results but did not include them here because they are very similar to the values presented.

A. Tabu Search parameters

Two decisions that affect both the tabu search and the LPH heuristic are the choice of α for load-balancing and the choice to use link distance versus hop count for shortest path routing. We found that, with the exception of the AT&T network, using hop count instead of link distance had a negligible affect on average tree delay while reducing the number of wavelengths required. For the AT&T network, the delay was in some cases doubled when using hop count instead of link distance. This tradeoff of delay versus wavelengths required is something that must be considered for particular networks, but it seems in most cases the best choice is to perform shortest path routing based on hop count.

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

The parameters specific to tabu search, the number of iterations, tabu list tenure, diversification iterations, and intensification iterations were obtained empirically. For all but the Italian WDM network, the best solution was found after 500 iterations half the time and earlier the other half. Typically clustering around 550-650 and 340-480. There were a number of times the best solutions were found higher in the 800 range. Due to this possibility, we set the maximum number to 1000.

VI. CONCLUSION

In this paper we presented a tabu search meta-heuristic for solving the static manycast routing and wavelength assignment problem. We compared tabu search with two other simpler heuristics and found that it decreases wavelengths required by between 30-40% compared to the multicast heuristic and about 10% compared to LPH. There is no significant difference in average tree delay for the different approaches.

Areas of future work include considering networks with sparse splitting, i.e., not all the nodes are equipped with MC-OXCs. We will also consider physical-layer impairments when creating route trees, such as power, optical signal to noise ratio, crosstalk, and nonlinearities.

REFERENCES

[1] S. Y. Cheung et. al., "Efficient quorumcast routing algorithms," in *Proceedings, IEEE INFOCOM*, June 1994, pp. 840–847.

[2] R. Ravi et. al., "Spanning trees short or small," in *Proceedings, ACM-SIAM symposium on Discrete algorithms*, 1994, pp. 546–555.

[3] R. Jain, "Internet 3.0: Ten problems with current Internet architecture and solutions for the next generation," in *Proceedings, IEEE MILCOMM*, Oct. 2006.

[4] 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.

[5] R. Malli et. al., "Benefits of multicasting in all-optical networks," in *Proceedings, All-Optical Networking*, 1998, pp. 209–220.

[6] B. Du et. al., "Quorumcast routing by multispace search," in *Proceedings, IEEE GLOBECOM*, Nov. 1996, vol. 2, pp. 1069–1073.

[7] C. P. Low, "Optimal quorumcast routing," in *Proceedings, IEEE GLOBECOM*, Nov. 1998, vol. 5, pp. 3013–3016.

[8] B. Wang et al., "An efficient QoS routing algorithm for quorumcast communication," in *Proceedings, IEEE ICNP*, 2001.

[9] X. Huang et. al., "Manycasting over optical burst-switched networks," in *Proceedings, IEEE ICC*, Jun. 2007.

[10] Q. She et. al., "Multi-resource manycast over optical burst-switched networks," in *Proceedings, IEEE ICCCN*, Aug. 2007, pp. 222–227.

[11] B. G. Bathula et. al., "Impairment-aware manycast algorithms over optical burst-switched networks," in *Proceedings, IEEE ICCCN*, 2008.

[12] B. G. Bathula et. al., "QoS-based manycasting over optical burst-switched (OBS) networks," *IEEE/ACM ToN*, May 2009.

[13] Q. She et. al., "On finding minimum cost tree for multi-resource manycast in mesh networks," *Elsevier Optical Switching and Networking (OSN)*, vol. 6, 2009.

[14] D. Din, "A hybrid method for solving ARWA problem on WDM network," *Computer Communications*, vol. 30, no. 2, pp. 385–395, 2007.

[15] F. Glover and M. Laguna, *Tabu Search*, Kluwer Academic Publishers, 1997.

[16] J. Kuri et. al., "Routing foreseeable lightpath demands using a tabu search meta-heuristic," in *Proceedings, IEEE GLOBECOM*, Nov. 2002, vol. 3, pp. 2803–2807.

[17] Y. Wang et. al., "A tabu search algorithm for static routing and wavelength assignment problem," *IEEE Communications Letters*, vol. 9, no. 9, pp. 841–843, Sep. 2005.

[18] C. Dzongang et. al, "A tabu search heuristic for the routing and wavelength assignment problem in optical networks," *IEEE Communications Letters*, vol. 9, no. 5, pp. 426–428, May 2005.

[19] W. S. Hu et. al., "Multicasting optical cross connects employing splitter-and-delivery switch," *IEEE Photon. Tech. Lett.*, vol. 10, pp. 970–972.

[20] J. Leuthold et. al., "Multimode interference couplers with tunable power splitting ratios," *IEEE/OSA JLT*, vol. 19, no. 5, pp. 700–707, May 2001.

[21] H. Takahashi and A. Matsuyama, "An approximate solution for the steiner problem in graphs," *Math. Japonica*, 1980.