Topology Control and Routing over Wireless Optical Backbone Networks*

Abhishek Kashyap*, Samir Khuller† and Mark Shayman*
*Institute for Systems Research
Department of Electrical and Computer Engineering
University of Maryland, College Park
Email: {kashyap, shayman}@glue.umd.edu
†Department of Computer Science
University of Maryland, College Park
Email: samir@cs.umd.edu

***Abstract*— The problem of topology control and routing over wireless optical backbone networks is addressed. The input to the problem is a potential topology and a traffic profile. The constraints are that of limited interfaces at each node and the limited bandwidth of the links, and the objective is of maximizing the throughput. The problem turns out to be NP-Hard, and we propose some heuristics based on graph modelling and multi-commodity flow algorithms to solve the problem. We also enhance the heuristics to try and provide fairness to the ingress-egress pairs in terms of how much traffic we route for each of them.**

I. Introduction

We consider the problem of topology control in wireless optical backbone networks, taking the estimated traffic into consideration. Our aim is to maximize the throughput over the topology for a given estimate of the traffic. We prove the problem to be NP-Hard, and then propose heuristics to achieve the goal. We also propose a scheme to provide fairness to the traffic routed between different ingress-egress pairs.

We consider wireless optical links because of their attractive characteristics which make them more suitable for backbone networks (compared to using RF and wireline optical links). Free-space optics technology is expected to deliver unprecedented bandwidth, massive carrier reuse, ultra-low inter-channel interference, low power consumption, and cost savings where electrical wires and optical fibers are too expensive to deploy and maintain [1]. A key distinguishing feature of wireless optical networks is that the links are point-to-point rather than broadcast. Also, it has wide applicability from long range satellite to indoor wireless communications.

The problem of topology control for wireless optical networks is different from that in wireless RF (radio frequency) networks since the links are point-to-point as opposed to broadcast. In wireless optical networks, each node has a limited number of transceivers, and hence can establish links with only a limited number of nodes within its transmission range. Thus, topology control is concerned with determining the neighbors with which to establish the limited number of possible links. In wireless networks, most research for topology control so far has focused on RF networks (see e.g., [10], [11], [16], [17], [18], [19], [20]). In RF-wireless networks with isotropic antennas, topology control is closely related to power control. Power is controlled to reduce the transmission range to conserve power and decrease interference while providing adequate connectivity.

There are important differences between topology control for reconfigurable wireline optical networks and topology control for wireless optical networks. In the wireline case, transmission range (lightpath length) is not a major issue. Furthermore, if the optical layer has sufficient resources so the routing and wavelength assignment problem is always solvable, then whenever a source and destination both have available interfaces, a direct connection (one logical hop) can be established. In contrast, in the wireless case, unless the destination is within the transmission range of the source, a multihop connection is required. For these reasons, the many published results on logical topology design for wireline optical networks, [12], [13], [14], [15], are not directly applicable to free-space optical networks.

There has been recent work on topology control in wireless optical networks ([4], [5]): [5] does not take traffic into consideration, while [4] considers only ring topologies. The work presented in [6] proposes a framework and some heuristics for topology control and routing which take potential traffic into consideration, and we compare our algorithms with those heuristics. The algorithms presented in this paper have a lower time complexity than the algorithms proposed in [6]. We also extend the algorithms proposed in [6] to allow for splitting the traffic as a commodity using multi-commodity flow formulation.

The heuristics we propose take as input a potential topology and the estimated aggregate traffic between the ingress-egress pairs (which we call the traffic profile). Each node is assumed to have a constraint on the number of interfaces it can have, and hence on the number of actual links that can be created from among the potential links. The decision of selecting the links to be formed is made on the basis of evaluating the total bandwidth guarantees (which we call throughput) we

*This research was partially supported by AFOSR under grant F496200210217.

can give for the given traffic profile. The traffic profile can be measured over some previous operation of the network or can be had from service level agreements (SLAs). We use matching theory, [9] along with the formulation of routing as a multi-commodity flow problem to calculate the topology and bandwidth reservations for the ingress-egress pairs. The bandwidth reservations calculated can be used for routing and admission control when the network is formed. The advantages of computing the bandwidth reservations offline and using that information for routing and exercising admission control can be found in [3].

The paper is organized as follows: Section 2 describes the network model and gives the problem statement. Section 3 explains the policies we compare the algorithms with. Section 4 explains our algorithm. Section 5 gives the extension to achieve fairness. Section 6 discusses the computational complexity and presents the simulation results.

II. Network Model and Problem Definition

We model the network as a graph $G = (V, E)$, where V is the set of nodes and E is the set of potential links between them. We consider wireless backbone networks in which each wireless node is equipped with point-to-point wireless optical interfaces. By the term 'node' we implicitly mean "backbone node". Each node has the capability to perform routing. We assume that it does not move very frequently. We also assume that wireless links can be set up in any direction with all the nodes within the transmission range. We do not consider the possibility of optical beam obscuration, but this assumption is not essential. Since the transmission distance is related to the power level of the node, the power level and thus the transmission range of each node can be different. The wireless links are unidirectional. If there is a pair of unidirectional links between two nodes, the link capacities may differ. The number of transmitters and receivers at each node is limited (which we call an interface constraint), thereby restricting the number of nodes to which it can connect.

We have a traffic profile, which consists of the aggregate traffic demands between the sources and destinations. The traffic demand from node x to node y can be different from the traffic demand from node y to node x.

The problem we address is to form a subgraph $G' = (V, E')$, such that the interface constraints are satisfied for all nodes in the set V (i.e., the degree of each node is bounded by the number of available interfaces), and we maximize the throughput considering the traffic profile. The algorithm forms this subgraph, which we call topology control, and comes up with routes and bandwidth reservations for the ingress-egress pairs given in the traffic profile. The server should recompute the topology, routes and bandwidth reservations whenever either the traffic profile or the (backbone) node locations change significantly. We do not anticipate that this would be done more often than hourly. The nodes then use this information to perform routing and traffic engineering on incoming flows.

We provide a proof that the problem of topology control is NP-Hard: Consider a small amount of traffic between each pair of nodes in the network (small enough not to violate any capacity constraints). The problem of maximizing the throughput reduces to finding a connected subgraph here. We can remove the extra edges and the problem reduces to finding a degree-constrained spanning tree, which is a known NP-Hard problem. A special case is where we have a degree constraint of 1 incoming and 1 outgoing edge on each node. In this case, the problem reduces to finding a hamiltonian cycle, which is known to be NP-Complete [7].

A. Routing as Multi-commodity flow problem

We set up the problem of routing a given traffic profile over a computed topology for maximizing the throughput as a linear multi-commodity flow problem [3]. Let there be M commodities (profile entries, the value of each entry is $profile(i)$), N nodes and L links in the network. We add a dummy link (infinite cost, infinite capacity) between the source and destination of each commodity to achieve feasibility (thus, there are M such links). Let $x_i(l)$ be the amount of commodity i routed through link l. Let $cost(l)$ represent the cost of each link, which is 1 for an actual link for our objective of maximizing the throughput. Let the set of incoming and outgoing links at node j be denoted by in_j and out_j respectively. Let $source_i$ and $dest_i$ represent the source and destination of profile i. Equation 1 achieves the objective of maximizing the throughput as the algorithm tries to route on the actual links due to large cost of the dummy links. Equation 2 represents the bandwidth constraints. Equations 3 and 4 represent the flow conservation laws.

$$minimize \quad \sum_{l=1}^{L+M} (cost(l) \sum_{i=1}^{M} x_i(l)) \tag{1}$$

$$\sum_{i=1}^{M} x_i(l) \leq capacity(l) \quad \forall l \in \{1,..,L\} \tag{2}$$

$$\sum_{l \in in_j} x_i(l) \quad = \sum_{l \in out_j} x_i(l)$$
$$\forall j \in \{1,..,N\} - \{source_i, dest_i\}, \forall i \in \{1,..,M\} \tag{3}$$

$$\sum_{l \in out_j} x_i(l) \quad - \sum_{l \in in_j} x_i(l) = profile(i),$$
$$j = source_i, \forall i \in \{1,..,M\} \tag{4}$$

III. Rollout Algorithms

We present a brief description of some rollout algorithms and their heuristic proposed in [6]. The heuristic takes a traffic profile and sorts it in decreasing order of demands. For each entry of the profile in that order, it finds a single constrained shortest path between the source and the destination, forms that path (finalizes the links of that path) in the topology (if one exists which can accommodate all the demand of this entry), deletes the links violating the interface constraints in this partial topology, and updates the capacity of the links of this shortest path by decreasing by the value of this profile entry.

A. *Route Rollout*

This technique is an improvement over the heuristic, and is guaranteed to work at least as good as the heuristic in terms of the objective function (throughput). It starts with a sorted traffic profile (in decreasing order). At stage i, it finds K constrained shortest paths for profile entry i, and starting with each of those paths (i.e., temporarily finalizing the links in that path), it forms the complete topology using the heuristic. The path which gives the maximum throughput (total demand routed) is chosen (i.e., links finalized, potential links violating interface constraints deleted and residual bandwidths updated to reflect the demand routed), and the algorithm advances to the next stage.

B. *Index Rollout*

This technique decides the order in which the profile entries should be routed to get maximum throughput. At each stage, it uses the heuristic to form the complete topology starting from each of the unrouted profile entries (and sorting the remaining unrouted entries in decreasing order), and calculates the throughput in each case. It selects the profile entry which gives the maximum throughput, and finalizes the links for its shortest path, along with updating the residual bandwidths and deleting links violating the interface constraints.

C. *Integrated Rollout*

This rollout takes into consideration both the index and the routes. Unlike index rollout, at each stage, it finds K constrained shortest paths for each of the unrouted entries, and for each case, it uses the heuristic to route the rest of the unrouted profile entries. The profile entry (and the corresponding path) giving the maximum throughput is chosen and those links finalized, and the algorithm advances to next stage.

In all the above cases, we are routing the whole profile entry over a single path (and not routing it at all if we are not able to route it all over one path). As an improvement, we can split the traffic linearly by applying the multi-commodity flow algorithm over the topology computed by these (as explained in Section 2) and get a better throughput.

IV. Application of Matching Theory

Here we present a new algorithm to form the topology and route the traffic profiles. The algorithm is outlined below and explained in the following subsections.

1) Weight the links in the initial graph to favor the links which are expected to have a higher traffic flow.
2) Map the network to one which can be given as an input to a maximum weight matching problem, [9] and solve the matching problem to get a maximum weight subgraph which satisfies the interface constraints.
3) Solve the multi-commodity flow problem over this topology.
4) Modify the topology sequentially and solve the multi-commodity flow problem again each time, and finally, keep the topology which gives the maximum throughput.

A. *Giving Initial Weight to Edges*

The way we map the problem to maximum weight matching problem, if we give the same weight to all the links in the network, the output topology will have the maximum number of links while satisfying the degree constraints (as we try to maximize the weight during maximum weight matching, so same weight to all edges would result in maximizing the number of edges). We call the algorithm using this policy as *Uniform Weighted Matching (UWM)*. As our objective is maximizing the throughput, so it is better to give extra weight to edges which are expected to carry more traffic. We call the algorithm using this policy as *Traffic Weighted Matching (TWM)*, and explain it below.

- Give a weight of 1 to all edges.
- Find K shortest paths (maximum) of same length for each traffic profile (over the potential topology).
- Each time a link comes in a path, add to its weight $profile(i)/numberSP$, where $profile(i)$ is the value of the profile entry, and $numberSP$ the number of shortest paths we found for that profile entry.

Another possibility is to give weights according to the number of paths a link comes on irrespective of the amount of traffic. In this case, we add 1 to the weight of a link in step 3 above. We called the algorithm using this strategy as *Flow Weighted Matching*. We work with shortest paths as the multi-commodity flow formulation will prefer shorter paths for a commodity because it minimizes the cost function of Equation 1 (to maximize the throughput).

B. *Mapping to Maximum Weight Matching*

Given a graph $G = (V, E)$ (which corresponds to the potential network) with edge weights as explained in subsection A, we form a graph $G' = (V', E')$ and give it as an input to the maximum weight matching problem. Let the number of transmit and receive interfaces (i.e., input and output degree constraints) at each vertex be Δ. Figure 1 shows an example potential topology with $\Delta = 2$. The steps of mapping are explained below.

- For each vertex, we form 2Δ vertices in graph G' (Δ of them correspond to the transmit interfaces, and Δ to receive interfaces).
- For each edge between two vertices in G, we form two vertices in G', and add an edge between them (of weight 0), as well as between one of them and the transmit interface vertices (all Δ of them) of the vertex it was going out of (with weight equal to the weight of the corresponding edge in G). Also add edges between the other vertex (in G') of this edge (from G) and the receive interface vertices of the vertex it was incident on (with weight equal to the weight of the corresponding edge

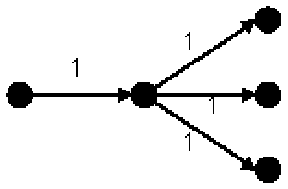

Fig. 1. Potential Topology, $\Delta = 2$

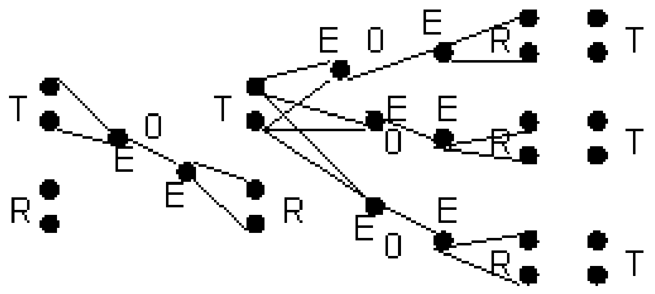

Fig. 2. Modified graph from potential topology

in G). The resulting graph for our example is shown in Figure 2 (the edges shown without weights have a weight of 1 here). T, R and E refer to the vertices corresponding to transmit interfaces, receive interfaces and edges in G respectively.

- Add a clique with $2\Delta N$ vertices to the graph G', with each edge in the clique having a weight 0. Add 0 weight edges between each of these vertices and each of the vertices in G' which correspond to the transmit and receive interfaces of the vertices in graph G.
- Sum the weight of all the edges in G' and add that to the weight of all edges in G'. We do this to make sure we get a perfect matching using a maximum weight matching algorithm.

We solve the maximum weight matching problem on G', and deduce the resulting topology based on the result we get. We show the rules for mapping from the output of matching to the graph representing the topology: Figure 3 shows the scenario (a subgraph from the output of matching algorithm) when two vertices will have an edge between them in the final topology. Figure 4 shows the case when two vertices which had an edge between them in G will not have the edge in the resulting topology (as they are not connected to the edge

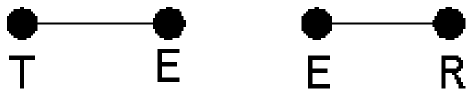

Fig. 3. Vertices are connected

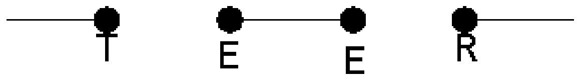

Fig. 4. Vertices are not connected

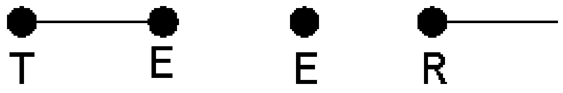

Fig. 5. Connection between vertices undecidable

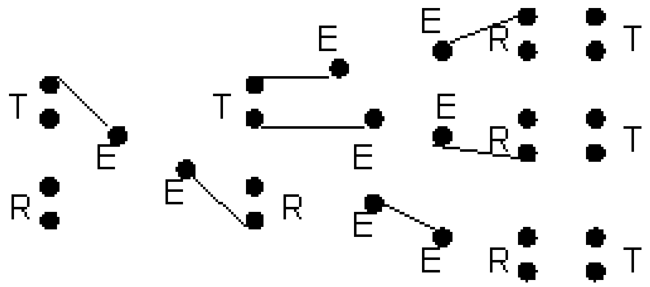

Fig. 6. Result of Matching Algorithm

vertices corresponding to the edge between them). We detect such subgraphs in the output graph from matching algorithm (the output graph will be composed of such subgraphs only), and use these rules to deduce the resulting topology.

We added a clique to the graph G' to avoid the scenario in which we cannot deduce the topology from the result of the matching algorithm. Figure 5 shows the case when this has happened between two vertices. In this case, one of the vertices connects to the edge vertex in G', while the other vertex does not connect with the corresponding edge vertex. This can be avoided by having perfect matching, and for achieving that we add a clique of size $2\Delta N$ and connect each of them to all the vertices in G' which correspond to transmit and receive interfaces of vertices in G. We accommodate the worst case in which all the vertices will be disconnected in the final topology.

Figure 6 shows the output (minus the clique vertices and corresponding edges) of the matching algorithm and Figure 7 shows the final topology we get for the example network of Figure 1.

C. Topology Change Strategy

We solve the multi-commodity flow problem (as explained in Section 2) on the topology we get from the algorithm explained in Section 4.B. We sequentially change this topology and solve the multi-commodity flow problem on the resulting topologies to get an improvement in the throughput. The algorithm for changing the topology is as explained below:

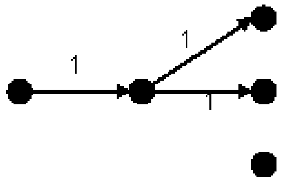

Fig. 7. Final Topology

- Make a list of the profile entries for which we could route less than $x\%$ of the demand in decreasing order of demand.
- For the first entry in this list, find (maximum) K shortest paths each in the potential topology and the current topology. Form the first path which is present in the potential topology, but not in the current topology by deleting the least loaded links (with traffic as given by the result of multi-commodity flow problem) at each of the interface-starved nodes in the current topology on this path. If all the paths are the same then repeat this step for the next entry in the list.
- Solve the multi-commodity flow problem for this changed topology. If the throughput is more than the throughput in the current topology then change the current topology to this topology and update the list by deleting the entries for which we have routed more than $x\%$ on this topology. If the throughput is less than before, then let the current topology remain the same and start with step 2 for the next entry in the list.

We finally keep the topology we have at the end of this procedure.

V. Incorporating Fairness

In all the algorithms we have seen so far, we do not consider fairness i.e., trying to route at least a certain fraction of each profile entry while maximizing the throughput. We address this problem by making changes to the multi-commodity flow formulation. We use two different weighting strategies for comparison:

- *Fairness1*: Use *Traffic Weighted Matching*.
- *Fairness2*: Use *Flow Weighted Matching*. This strategy is expected to be more fair than *Fairness1*.

A. Multi-commodity Flow Formulation

The primary change from the formulation explained before is the addition of the fairness constraint of Equation 5. Here, we sum the outgoing and incoming flows only over the actual links (and not over the (infinite capacity, infinite cost) links we added between the source and destination, as the traffic flowing over them is what we could not route). We also change the flow conservation law at intermediate nodes (of Equation 3) to include only the actual links adjacent to those nodes (Equation 6). Here, ain_j denotes the actual incoming links at node j, and $aout_j$ denotes the actual outgoing links at j. This is done to make sure the traffic for a profile entry is not routed through the extra links between some nodes other than the source and destination of this profile entry (otherwise that will happen as the algorithm tries to force the condition of Equation 5).

$$\sum_{l\in aout_j} x_i(l) - \sum_{l\in ain_j} x_i(l) \geq y * profile(i),$$
$$j = source_i, \forall i \in \{1,..,M\} \quad (5)$$

$$\sum_{l\in ain_j} x_i(l) = \sum_{l\in aout_j} x_i(l),$$
$$\forall j \in \{1,..,N\} - \{source_i, dest_i\}, \forall i \in \{1,..,M\} \quad (6)$$

This formulation will give an infeasible result if it is not able to route at least fraction y of each profile entry. We follow the following procedure to get the routes (and reservations) for the topology we get from the matching algorithm:

- Start at $y = 0.95$ and solve the changed multi-commodity flow problem.
- If it returns a feasible result, keep the current topology and these routes (and reservations) and exit. Else, decrease y by 0.05, and solve the problem again.

If we get the answer at $y = 0$, then the answer is the same as what we get at the first step of the algorithm explained in Section 4.

VI. Simulation and Computational Analysis

A. Computational Complexity

The proposed algorithms take $O(N^4 logN)$ time, with N being the number of nodes in the network. This complexity is much lower than that for route rollout ($O(N^6)$) and index and integrated rollout algorithms ($O(N^8)$). The heuristic used by these rollout algorithms takes $O(N^4)$ time.

B. Simulation Results

The network used for simulations was assumed to have the following parameters:

- Number of nodes in the network = 20. This represents a reasonably sized backbone network.
- Nodes are uniformly distributed, with each node having an average of 6.5 potential neighbors.
- The transmission range of all nodes is assumed to be the same.
- Capacity of each link = 100 in each direction.
- Number of source-destination pairs = 160, chosen randomly.
- Aggregate traffic between each pair: Uniformly distributed between 1 and 40 units.
- Number of receive interfaces at each node = 3.
- Number of transmit interfaces at each node = 3.
- Threshold, x, for including a profile in the sequential topology change list = 20%.
- Number of shortest paths considered in Route Rollout and Integrated Rollout = 4.
- Number of shortest paths considered for initial weighting and sequential topology change, K = 3.
- Weight of each link for constrained shortest path computation = 1, thus making the shortest path as the constrained min-hop path.

The simulation was run on a different random network and random profile 10 times and in each simulation, the network topology was formed starting with these parameters. The simulations were run for the heuristic and rollout algorithms described in [6], both with and without following it with

TABLE I

AVERAGE THROUGHPUT FOR ROLLOUT ALGORITHMS

Heuristic	Route Rollout	Index Rollout	Integrated Rollout
0.5748	0.6313	0.6732	0.6892

TABLE II

AVERAGE THROUGHPUT FOR ROLLOUT ALGORITHMS WITH TRAFFIC SPLITTING

Heuristic	Route Rollout	Index Rollout	Integrated Rollout
0.7090	0.7223	0.7335	0.7550

solving multi-commodity flow problem for routing. Tables I and II show the average fractional throughput for the rollout algorithms and their heuristic for the case where we do not split the traffic and the case where we split the traffic as a commodity respectively. As can be seen, splitting the traffic increases the throughput considerably.

Table III shows the average throughput for the proposed algorithms. The matching algorithm used was an implementation of H. Gabow's N-cubed weighted matching algorithm [21]. As we can see, the algorithm with *Traffic Weighted Matching (TWM)* works the best, followed by *Flow Weighted Matching (FWM)*, *Uniform Weighted Matching (UWM)* and the fairness schemes. *Traffic Weighted Matching* works 8.88% better than the heuristic for rollout and 2.25% better than the integrated rollout (with the proposed traffic splitting strategy). Table IV shows the average throughput we get in the proposed algorithms without using the sequential topology change strategy. As can be seen by comparing Tables III and IV, the improvement in throughput is the maximum in *Uniform Weighted Matching* followed by *Flow Weighted Matching* and *Traffic Weighted Matching*. This, along with the results of Table III, shows that *Traffic Weighted Matching* works the best among these three strategies.

The metric we use to evaluate fairness is the minimum of the fraction of the demand routed for each profile entry. Table V shows the average of this parameter over the simulations. *Fairness2* is more fair (0.53) than *Fairness1* (0.425) as expected, though at the cost of throughput, as can be seen from Table III.

TABLE III

AVERAGE THROUGHPUT FOR MATCHING HEURISTICS

UWM	FWM	TWM	Fairness1	Fairness2
0.7178	0.7606	0.7720	0.7080	0.6719

TABLE IV

AVERAGE THROUGHPUT FOR MATCHING HEURISTICS WITHOUT SEQUENTIAL TOPOLOGY CHANGE

UWM	FWM	TWM
0.6650	0.7294	0.7535

TABLE V

AVERAGE MINIMUM ROUTED FOR FAIRNESS SCHEMES

Fairness1	Fairness2
0.425	0.53

This metric turns out to be 0 for all other algorithms mentioned in this paper.

REFERENCES

[1] N. A. Riza, "Reconfigurable Optical Wireless", *LEOS '99* IEEE , vol.1 , pp.70-71, 8-11 Nov. 1999.

[2] Z. Yaqoob, N. A. Riza, "Smart Free-Space Optical Interconnects and Communication Links using Agile WDM Transmitters", *2001 Digest of the LEOS Summer Topical Meetings* , 30 July-1 Aug. 2001.

[3] S. Suri, M. Waldvogel, D. Bauer, P. R. Warkhede, "Profile-Based Routing and Traffic Engineering", *Computer Communications*, vol. 24(4), pp. 351-365, March 2003.

[4] A. Desai, "Dynamic Topology Control of Free Space Optical Networks", M.S. Thesis, Department of Electrical Engineering, University of Maryland, 2003.

[5] P. C. Gurumohan, J. Hui, "Topology Design for Free Space Optical Networks", *ICCCN 2003*.

[6] Abhishek Kashyap, Kwangil Lee, Mark Shayman, "Rollout Algorithms for Integrated Topology Control and Routing in Wireless Optical Backbone Networks", Technical Report, Institute for Systems Research, University of Maryland, 2003.

[7] M. Garey and D. Johnson, "Computers and Intractability: A Guide to the theory of NP-Completeness", Freeman and Company, 1979.

[8] D. P. Bertsekas, "Dynamic Programming and Optimal Control", vol. 1, Athena Scientific, 2000.

[9] L. Lovász and M. D. Plummer, "Matching Theory", North-Holland, 1986.

[10] R. Ramanathan and R. Rosales-Hain, "Topology Control of Multihop Wireless Networks using Transmit Power Adjustment", *IEEE Infocom 2000*, vol. 2, pp. 404-413, 26-30 March 2000.

[11] Z. Huang, C-C. Shen, C. Srisathapornphat and C. Jaikaeo, "Topology Control for Ad Hoc Networks with Directional Antennas", *ICCCN 2002*, pp. 16-21, Miami, Florida, October 2002.

[12] D. Banerjee, B. Mukherjee, "Wavelength-routed Optical Networks: Linear Formulation, Resource Budgeting Tradeoffs and a Reconfiguration Study", *IEEE/ACM Trans. Networking*, vol. 8, pp. 598-607, Oct. 2000.

[13] K. H. Liu, C. Liu, J. L. Pastor, A. Roy, J. Y. Wei, "Performance and Testbed Study of Topology Reconfiguration in IP over WDM", *IEEE Transactions on Communications*, in press, 2002.

[14] E. Leonardi, M. Mellia, M. A. Marsan, "Algorithms for the Logical Topology Design in WDM All-Optical Networks", *Optical Networks Magazine*, pp. 35- 46, Jan. 2000.

[15] R. Ramaswami, K. N. Sivarajan, "Design of Logical Topologies for Wavelength-Routed optical Networks", *IEEE JSAC*, pp. 840-851, Jun. 1996.

[16] L. Li, J. Halpern, P. Bahl, Y-M. Wang and R. Wattenhofer, "Analysis of a cone-based distributed topology control algorithm for wireless multi-hop networks", *ACM Symposium on Principles of Distributed Computing*, 2001.

[17] V. Rodoplu and T. Meng, "Minimum Energy Mobile Wireless Networks", *IEEE International Conference on Communication*, vol. 3, pp. 1633-1639, 7-11 June 1998.

[18] R. Wattenhofer, L. Li, P. Bahl and Y-M. Wang, "Distributed Topology Control for Wireless Ad-hoc Networks", *IEEE Infocom 2001*, pp. 1388-1397.

[19] S. Ruhrup, C. Schindelhauer, K. Volbert and M. Grunewald, "Performance of Distributed Algorithms for Topology Control in Wireless Networks", *International Parallel and Distributed Processing Symposium*, 2003.

[20] L. Bao and J. J. Garcia-Luna-Aceves, "Topology Management in Ad Hoc Networks", *ACM Mobihoc*, pp. 129-140, Annapolis, Maryland, June 2003.

[21] H. Gabow, "Implementation of Algorithms for Maximum Matching on Nonbipartite Graphs", Ph.D. thesis, Stanford University, 1973.