Algebraic Connectivity of Degree Constrained Spanning Trees for FSO Networks

Hui Zhou[†], Alireza Babaei[‡], Shiwen Mao[†] and Prathima Agrawal[†]
[†]Department of ECE, Auburn University, Auburn, AL 36849, USA
[‡]Department of ECE, Virginia Tech, Blacksburg, VA 24061, USA
Email: hzz0016@auburn.edu, ababaei@vt.edu, smao@ieee.org, agrawpr@auburn.edu

***Abstract*—Free space optical (FSO) networking is an attractive technology with applications ranging from high capacity military communications to "Last-mile" broadband access solutions. Topology control is an important problem in FSO networks. The problem of building a spanning tree when number of transceivers on each base station is limited, in FSO networks, is NP-hard. What makes the problem even more challenging is to maximize the algebraic connectivity of the spanning tree. In this paper, we develop an initially configuring, or bootstrapping, algorithm which produces a degree constrained spanning tree with high algebraic connectivity and also high average edge weight, where the edge weight in the graph represents the FSO link reliability. We also develop a fast reconfiguration algorithm when one or more links fail in the FSO network. Our algorithms outperform alternative schemes in improving both algebraic connectivity and average edge weight. The robustness of the resulting topology of FSO networks is significantly improved.**

***Index Terms*—FSO network; bootstrapping; reconfiguration; algebraic connectivity; degree constrained spanning tree**

I. Introduction

Drawing increasing attention, free space optical networks are considered as an important alternative to the radio frequency (RF) systems as the backbone solution for future data-intensive networks. This appeal is primarily due to their high optical fiber-like data rates, resistance to electromagnetic interference, and security among other advantages. There are many research opportunities and also challenges in different aspects of FSO networks, such as pointing, acquisition and tracking (PAT), weather/environment issue, channel randomness, and topology control [1]. Because an FSO transmitter is highly directional, the performance of the FSO networks is dependent on the reliability of a line-of-sight (LOS) path. Atmospheric obstructions (e.g., due to the weather turbulence or flying objects) can degrade the link performance considerably. The network must, therefore, be capable of control and reconfiguration of its topology.

In topology control, bootstrapping and reconfiguration during operation are two important aspects, both of which are considered in this paper. In designing the bootstrapping algorithm for FSO networks, the difficulty lies in the distributed formation of topology with every node only knowing directly connected neighbor information. Moreover, there are only a limited number of transceivers in one base station, which makes this problem even more challenging; because now the problem becomes Degree Constrained Minimum Spanning Tree Problem (DCMST). The DCMST problem can be categorized as Euclidean Problem and non-Euclidean Problem, e.g., the random table. In Euclidean problems, the edge weights are said to obey the triangle inequality; but in non-Euclidean problems, the edge weights do not necessarily obey the triangle inequality [2]. Non-Euclidean problems were found to be far less tractable than Euclidean problems. The problem of building a spanning tree in FSO network is a random table problem. Moreover, even in non-weighted graphs, building Degree Constrained Spanning Tree (DCST) is still NP-hard.

After bootstrapping the FSO network, base stations can exchange information about their neighbors and hence a centralized algorithm for reconfiguration can be applied. A centralized algorithm will consume less computation time, which is desirable in link failure situations. If one or several links fail in FSO networks, the algorithm should reconfigure the topology quickly to avoid partitioning or degradation.

In this paper, the objective is to maximize the edge weight and the algebraic connectivity of spanning tree, which result in higher robustness in topology of FSO networks. In this paper, Edge weight represents FSO link reliability. Algebraic connectivity, as defined in spectral graph theory, is the second smallest eigenvalue of the Laplacian matrix L associated with a graph [3]. It provides a good measure to determine how well the spanning tree is connected [4]. We develop a topology control scheme when not only the number of transceivers on each base station is limited but also when these numbers are different on different base stations. Both of our bootstrapping and reconfiguring algorithms are shown to generate spanning tree with high average edge weight and algebraic connectivity. In addition, by exploiting the knowledge of topology history, our reconfiguring scheme requires much less computation time in case of link failure.

The remainder of this paper is organized as follows. In Section II we review some related works on topology control in FSO networks. We formulate the problem using 0-1 integer linear programming (ILP) in Section III. The topology control algorithms and analysis are presented in Section IV. Our simulation results and performance analysis are presented in Section V. Section VI concludes this paper.

II. Related Work

Several previous works have focused on topology control in FSO networks. Some papers consider bootstrapping FSO

978-1-4673-3122-7/13/$31.00 ©2013 IEEE

networks and others propose reconfiguration schemes. In [5], a bottom-up (BU) algorithm is proposed for bootstrapping FSO networks. BU is a fully distributed approximation algorithm, which constructs a spanning tree with maximal node degree at most one larger than that in the optimal solution. In the algorithm proposed in [5], node ID is assigned arbitrarily but used as an important factor in determining which neighbor to connect with. Another problem, as it is also stated in [5] but not solved, is synchronization problem.

In [6], authors propose a hybrid network prototype using free space optical link for data transmission only, and RF link primarily for control signals as well as serving as a backup for data transmission whenever FSO link is unavailable. This scheme of dynamic path reconfiguration is shown to have high performance in terms of reconfiguration time and end-to-end delay. In this paper, we consider pure FSO network, which means there is no backup RF systems.

A heuristic distributed fragment selection and merging (FSM) algorithm [3] is presented to solve bootstrapping and reconfiguring FSO networks. The author formulates the topology creation as building a k-degree constrained spanning tree which means all the nodes in a graph are subject to the same degree constraint k. The proposed approach in [3] requires high computational complexity because of calculating the algebraic connectivity in every round, which makes this algorithm not competitive for fast reconfiguration. In this paper, a more general case, that is different degree constraints on every single node, will be considered. Similar to [3], we also use algebraic connectivity in this paper as it provides a useful measure for connectivity of graphs.

The authors in [7] propose a centralized algorithm which aims to design mesh topology with strong connectivity and short diameter with degree bound. The closest neighbor algorithm proposed in this paper starts with building a DCMST using the primal method.

In order to construct a DCMST in polynomial time, many heuristic and exact algorithms have been proposed, such as Primal method, Dual method, simulated annealing, Lagrangean relaxation and branch and bound method [2]. Several of these algorithms can effectively provide sub-optimal solutions and generate a spanning tree with minimum weight sum. However, these algorithms do not work well in bootstrapping FSO networks, since a base station will only have knowledge of neighboring base station's information. They also are not good at reconfiguring the network which requires fast reaction to avoid partitioning or degradation.

III. Problem Statement

A. Model Description

We consider network model using FSO technology to have the following characteristics:

- FSO network only consists of base stations equipped with FSO transceivers;
- FSO channel is full duplex;
- FSO transceivers can transmit data over a randomly oriented sector φ of degree θ (θ typically is $2\pi/9$). An edge in topology means a pair of nodes can communicate with each other bi-directionally;
- FSO transceivers have Pointing, Acquisition, and Tracking (PAT) capabilities;
- FSO base stations can use topology discovery to find their neighborhood, and in the beginning, base stations can adjust their transmitting power, if necessary.

There are several different models for characterizing weather turbulence in FSO network, such as log-normal distribution for weak-to-moderate turbulence, gamma-gamma distribution for modeling moderate-to-strong turbulent channels and other models. We adopt the gamma-gamma model and therefore, focus on topology control under turbulent channel conditions. The gamma-gamma distribution of channel state h is given as [8]

$$f(h) = \frac{2(\alpha\beta)^{\frac{\alpha+\beta}{2}}}{\tilde{h}\,\Gamma(\alpha)\Gamma(\beta)}\left(\frac{h}{\tilde{h}}\right)^{\frac{\alpha+\beta}{2}-1} \mathrm{K}_{\alpha-\beta}\left(2\sqrt{\alpha\beta\frac{h}{\tilde{h}}}\right) \qquad (1)$$

where $\mathrm{K}_{\varphi}(\cdot)$ is the modified Bessel function of the second kind of order φ, $\Gamma(\cdot)$ is the Gamma function, and α and β are parameters which are related to distance, aperture diameter and atmospheric conditions. $\tilde{h}$ in this model is a scale parameter, which is related to geometric spread loss and path attenuation and can be calculated with the knowledge of transceiver parameters and atmospheric conditions.

Given a threshold of channel state h_0 , link reliability is defined as

$$\mathcal{L} = \int_{\mathrm{h}_0}^{\infty} \frac{2(\alpha\beta)^{\frac{\alpha+\beta}{2}}}{\tilde{\mathrm{h}}\,\Gamma(\alpha)\Gamma(\beta)}\left(\frac{\mathrm{h}}{\tilde{\mathrm{h}}}\right)^{\frac{\alpha+\beta}{2}-1} \mathrm{K}_{\alpha-\beta}\left(2\sqrt{\alpha\beta\frac{\mathrm{h}}{\tilde{\mathrm{h}}}}\right)\,\mathrm{dh}$$

$$= \frac{\alpha\beta}{\tilde{\mathrm{h}}\Gamma(\alpha)\Gamma(\beta)}\left(\alpha\beta\frac{\mathrm{h}_0}{\tilde{\mathrm{h}}}\right)^{\frac{\alpha-\beta}{2}} \mathrm{G}_{1,3}^{3,0}\left(\alpha\beta\left(\frac{\mathrm{h}_0}{\tilde{\mathrm{h}}}\right)\middle|\begin{matrix} -\frac{\alpha+\beta}{2} \\ 1-\frac{\alpha+\beta}{2}, \frac{\alpha-\beta}{2}, \frac{\beta-\alpha}{2}\end{matrix}\right) \qquad (2)$$

Meijer G-function which is a standard built-in function in most of mathematical software packages [8] can be used to solve this integral.

Link reliability is an important parameter to measure the channel conditions in FSO networks. Hence, we use it as edge weight in graph. Weight on the edge which is incident on node i and node j is defined as the link reliability $\mathcal{L}_{ij}$ if $\mathcal{L}_{ij}$ is larger than a threshold value and zero otherwise; edges with zero weight should not be included in graph in our problem.

$$w_{ij} = \begin{cases} \mathcal{L}_{\mathrm{ij}}, & if\ \mathcal{L}_{ij} \geq \mathcal{L}_{th} \\ 0, & otherwise \end{cases} \qquad (3)$$

B. Problem Formulation

In this paper, we address the degree constrained spanning tree problem given a feasibility undirected graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of V base stations or vertices, and $\mathcal{E}$ is the set of E links or edges between vertices, containing no self-loops or parallel edges. Every edge has been assigned positive weight

defined in Equation (3) in Part A. In degree constraint vector $\boldsymbol{D}_{V*1} = \{d_i\}$, every entry d_i indicates the maximum number of edges which is allowed to be incident on node i.

We formulate two separate problems for bootstrapping and reconfiguring FSO networks. With respect to the bootstrapping problem, we seek a spanning tree T subject to the degree bound which has maximum algebraic connectivity. For reconfiguring the FSO network when links fail, fast reaction is more desirable. Hence, we aim to minimize computation time and transceiver movement.

The problem of finding a spanning tree in bootstrapping an FSO network is formulated as follows.

Maximize $\lambda_2(\boldsymbol{L})$ (4)

Subject to

$$1 \leq \sum_{e_{ij} \in \delta(v)} x_{ij} \leq d_v \qquad \forall v \in \mathcal{V} \tag{5}$$

$$\mathbf{1}^T \cdot \boldsymbol{X} \cdot \mathbf{1} = 2(V-1) \tag{6}$$

$$x_{ij} \in \{0,1\} \tag{7}$$

In the first constraint, $\delta(v)$ is the set of edges incident to node v. $\boldsymbol{X}$ is the adjacency matrix of the spanning tree. If there is a link between node i and j, $x_{ij} = 1$; otherwise, $x_{ij} = 0$.

The problem of reconfiguring an FSO network is slightly different since a spanning tree has already been formed. Another spanning tree is needed because of link failure in the former spanning tree. The objective is to minimize the difference between these two trees (or maximize edges that do not need to move) that is to minimize the transceiver movement. The problem is formulated as

Maximize $\mathbf{1}^T \cdot (\boldsymbol{X_1} \oplus \boldsymbol{X_2}) \cdot \mathbf{1}$ (8)

Subject to

$$1 \leq \sum_{e_{ij} \in \delta(v)} x_{ij} \leq d_v \qquad \forall v \in \mathcal{V} \tag{9}$$

$$\mathbf{1}^T \cdot \boldsymbol{X_2} \cdot \mathbf{1} = 2(V-1) \tag{10}$$

$$\boldsymbol{X_1} \oplus \boldsymbol{A_1} = \boldsymbol{A_1} \tag{11}$$

$$\boldsymbol{X_2} \oplus \boldsymbol{A_2} = \boldsymbol{A_2} \tag{12}$$

$$x_{ij} \in \{0,1\} \tag{13}$$

Where, $\boldsymbol{X_1}$ is already given as an adjacency matrix of the old spanning tree and $\boldsymbol{X_2}$ is the adjacency matrix of the new spanning tree. Operator $\oplus$ in graph theory gives the element-by-element minimum value in the two matrices. For example, if $\boldsymbol{A} = (a_{ij})_{m*n}$ $\boldsymbol{B} = (b_{ij})_{m*n}$, and $\boldsymbol{C} = (c_{ij})_{m*n} = \boldsymbol{A} \oplus \boldsymbol{B}$, then $c_{ij} = \min\{a_{ij}, b_{ij}\}$. Adjacency matrices of the graph $\boldsymbol{A_1}$ before the link failure and matrix $\boldsymbol{A_2}$ after the link failure are known. Obviously, edges in spanning tree belong to set of edges in graph, which are formulated as Eq. (11) and (12).

Constructing a DCMST is proven to be an NP-hard problem for non-Euclidean graphs [2]. If the degree constraint d_v is 2 for all nodes, then the problem reduces to the Hamilton path problem which is NP-complete [5]. The problem of maximizing algebraic connectivity is also proven to be NP-hard [9]. Hence, we develop a distributed heuristic algorithm for creating a spanning tree in the first place, and also design a scheme to reconfigure topology with small operational time.

IV. Topology Control Algorithm

Bootstrapping and reconfiguring FSO networks are two equally important aspects of topology control. To satisfy their unique requirements, we address these two problems individually. However, in designing an algorithm for bootstrapping FSO networks, we keep in mind of making the reconfiguration easier.

A. Bootstrapping Algorithm

Bootstrapping algorithm in this paper include two phases, which are presented in Tables I and II. Before the execution of these algorithms, we assume that each node has already known its neighbors' information through topology discovery.

1) Phase 1: Construct Degree Constrained Spanning Tree

The first phase is to asynchronously construct a degree constrained spanning tree without violating degree constraint as fast as possible. At the beginning, every node in the tree is sleeping. Then, the first one node or a few nodes are awakened spontaneously; other nodes are awakened by their neighbors. Once a node is awake, it first reads neighbor information and forms a group with a unique identification. Any nodes that are connected are in the same group. After reading neighbor information, it sorts these neighbors according to their degree constraints. Neighbors with higher degree constraints are preferred when deciding which link to choose.

A node marks neighbor nodes which are outside its group and need to be connected as "VISIBLE"; especially, when this node's degree constraint is satisfied, which means it can be connected with more neighbors, its neighbor nodes are marked as "REACHEABLE" and itself is marked as "ACTIVE". For the links inside the group, some links are connected in the spanning tree and these links are marked as "ONTREE"; other links are not in the spanning tree, so they are marked as "INGROUP". A node will try to establish a link with its "REACHABLE" node until its degree constraint is no longer satisfied. But sometimes a group has some "VISIBLE" nodes which are isolated. Hence, we need to adjust the topology in the group to include these nodes in the graph. "ADJUST" procedure is similar to "IMPROVEMENT" in [5]. We add an edge, which is not in the tree and the addition will not violate any degree limits, to the spanning tree, which results in the formation of a cycle, and then we delete another link in the cycle. Now, the cycle is broken and a new spanning tree is formed. By repeating this procedure, we can adjust the degree distribution, which might lead to more "ACTIVE" nodes.

When two nodes in different groups establish a link between them, the group identification should update to a unique ID for the new merged group and a root is selected. When there is no more visible node for every group or node, which means all nodes are included in a spanning tree, this DCST algorithm ends with a degree constrained spanning tree.

The distributed algorithm for constructing a DCST is presented in Table I.

2) Phase 2: Maximize Algebraic Connectivity

The second phase is maximizing the algebraic connectivity of the degree constrained spanning tree. This phase only runs in the root node. This heuristic maximum algebraic connectivity algorithm will generate a final sub-optimal topology and the final result will be distributed to all nodes in the FSO network.

TABLE I. DCST FORMULATION ALGORITHM

```
1:Wait for awakening ;
2:Read neighbor information and  Set Group identification;
3:While((there is still visible node for the group)
4:   IF(node has no "REACHABLE" neighbor )
5:        Mark yourself as "NOTACTIVE";
6:   ELSE
7:        Mark yourself as "ACTIVE";
8:   END
9:   Mark links which is in the spanning tree inside the group as "ONTREE",
      and  all other potential links inside the group as "INGROUP";
10:  Sort neighbors according to degree limit;
11:  WHILE (ACTIVE)
12:       Mark neighbors with different group ID as "VISIBLE", and  mark
          neighbors as  "REACHABLE" if currently degree is smaller than d_v;
13 :      Awaken and connect "REACHABLE" node with largest degree;
14:       Mark this link as "ONTREE";
15:       IF(Degree constraint is not satisfied)
16:            Mark yourself as "NOTACTIVE";
17:       END
18:  END
19:  Send root node your status;
20:  WHILE(Selected as ROOT)
21:       IF (No "ACTIVE" nodes in group)
22:            Adjust a "INGROUP" link by adding it and deleting another
              link  which form a cycle with this link;
23:           Change node status accordingly;
24:           IF(Still no "ACTIVE" node in group)
25:                Randomly delete more "ONTREE" links and add the
                  same number of potential links to maintain the tree;
26:               Change node status accordingly;
27:           END
28:       END
29:  END
30: END
```

TABLE II. MAXIMUM ALGEBRAIC CONNECTIVITY(MAC) ALGORITHM

```
1:   WHILE(for every edge in the tree)
2:      Find a reducing edge with larger weight to exchange;
3:      IF (multiple choices exist)
4:         Choose a reducing edge which will give us a branch in which
              ∑_{e∈P(i,i)} 1/w(e) for every reducing node is the smallest;
5:         Break a tie by choosing an edge with largest w_ij (v_i − v_j)^2
6:      END
7:   END
```

We use the edge exchange algorithm which is presented in Table II to maximize algebraic connectivity. Reducing edge sets, which is similar to the concept in [5], will be also useful in the reconfiguration algorithm. Every edge will have a set of edges whose addition to the tree will lead to forming a cycle with this edge. This set of edges are called "reducing edge" to this edge, and nodes with reducing edge incident to are named "reducing node".

In the decision of which reducing edge to choose if multiple potential reducing edges exist, the motivation for our heuristic is as follows:

If an edge is added to a graph G, the algebraic connectivity λ_2 will increase. λ_2 is defined as the second smallest eigenvalue of the Laplacian matrix $\boldsymbol{L}$ and can be written as $\lambda_2(\boldsymbol{L}) = min_{a\perp 1, a\neq 0} \frac{<\boldsymbol{L}\cdot a, a>}{<a,a>}$. If $\boldsymbol{v}$ is a normalized eigenvector corresponding to λ_2, for any symmetric matrix $\mathbf{Y}$, $\lambda_2(\boldsymbol{L}+\boldsymbol{Y}) \leq \lambda_2(\boldsymbol{L}) + Tr(\boldsymbol{Y}\boldsymbol{v}\boldsymbol{v}^T)$. If λ_2 is isolated, the increase in λ_2 can be found as $\frac{\partial}{\partial x_l}\lambda_2 = \boldsymbol{v}^T \frac{\partial \boldsymbol{L}}{\partial x_l}\boldsymbol{v}$ where $\boldsymbol{x}$ is a Boolean vector to indicate whether an edge belongs to the tree or not [4]. By definition

$$\boldsymbol{L} = \sum_{l=1}^{l=m} x_l \cdot w_l \cdot \boldsymbol{a_l} \cdot \boldsymbol{a_l^T} \tag{14}$$

where m is the number of edges in current graph, w_l is the weight of edge, and $\boldsymbol{a_l}$ is the edge vector. The edge vector, for an edge connecting vertices i and j, is defined as $a_{li} = 1, a_{lj} = -1, and\ a_{lx} = 0\ for\ other\ vertices\ x$. Since

$$\frac{\partial \boldsymbol{L}}{\partial x_l} = w_l \cdot \boldsymbol{a_l} \cdot \boldsymbol{a_l^T} \tag{15}$$

We can find the partial derivative of λ_2 with respect to x as $\frac{\partial}{\partial x}\lambda_2 = w_{ij}(v_i - v_j)^2$, which means $w_{ij}(v_i - v_j)^2$ gives the first order approximation of increase in λ_2. To clarify, node i and j are incident to edge l and w_{ij} is the weight of edge.

The computation of eigenvector for every edge exchange is time consuming which is not desirable in FSO networks and drives us to find a faster method to choose the reducing edge.

According to Theorem 1 in [10], if bottleneck matrices for two branches satisfy the condition: $\boldsymbol{M} \ll \widetilde{\boldsymbol{M}}$, then, algebraic connectivity of two trees with these two branches satisfies $\lambda_2 \geq \widetilde{\lambda_2}$. Bottleneck matrix for a branch **B** at a vertex v is a $k * k$ matrix, where k is the number of vertices in the branch. The entry in position (i,j) is defined as $\sum_{e\in P(i,j)} 1/w(e)$, where $P(i,j)$ is the set of edges which are on both the path from i to v and from j to v.

Note that we can always find a branch **B** which includes the reducing edges and a vertex v which the branch is at. Smaller bottleneck matrix for the branch **B** will give us larger algebraic connectivity.

After careful inspection of the bottleneck matrix, we find that there is even no need to calculate the whole bottleneck matrix. We only need to compare the $\sum_{e\in P(i,i)} 1/w(e)$ for every reducing node, where $P(i,i)$ is the set of edges which are on the path from reducing node i to v. The reason is that if reducing node is not on the path from another node j, then the entry for node j will remain the same; otherwise the change of $\sum_{e\in P(i,j)} 1/w(e)$ will be the same with $\sum_{e\in P(i,i)} 1/w(e)$. Hence, to maximize the algebraic connectivity, we need to find out which reducing edge will give us a branch in which $\sum_{e\in P(i,i)} 1/w(e)$ for every reducing node is the smallest.

Proposition 1: The complexity of our bootstrapping algorithm is $\mathcal{O}(V^2 * E)$.

Proof: In the phase 1, the computational complexity of the DCST formation algorithm is $\mathcal{O}(V^2 * E)$. In the phase 2, the complexity of our Maximum Algebraic Connectivity algorithm is $\mathcal{O}(V^2 * E)$, because in the first place, finding reducing edges for the edge in tree takes $\mathcal{O}(V * E)$. If multiple choices of reducing edges exist, finding the best bottleneck matrix takes $\mathcal{O}(V^2)$ and computing eigenvector takes $\mathcal{O}(V)$. Hence the whole computational complexity is $\mathcal{O}(V^2 * E)$. □

B. Reconfiguration Algorithm

Reconfiguration algorithm only applies when one or more FSO links fail in the system. We introduce an algorithm with low computational complexity because in best case we can make use of the knowledge that we get from the bootstrapping process. In bootstrapping algorithm, we have already found out the set of reducing edges for every edge.

TABLE III. RECONFIGURATION ALGORITHM

```
IF(a link failure information received)
    Find a reducing edge with largest weight to exchange;
    IF (multiple choices exist)
        Choose a reducing edge which will give us a branch in which
            ∑_{e∈P(i,i)} 1/w(e) for every reducing node is the smallest;
        Break a tie by choosing an edge with largest w_ij (v_i − v_j)^2
    END
END
IF(multiple link failure information received simultaneously)
    IF(check if there is node failure)
        Bootstrapping FSO network;
    ELSE
        Find a reducing edge with largest weight to exchange for every failed
        edge;
    END
END
```

In the case of only one link failure, one backup link is chosen from the set of reducing edges to replace the failed or degraded link. In the case of two or more link failures, if these links are not all incident on one failed node, reducing edges are chosen separately to form the spanning tree. In case of node failure, bootstrapping algorithm is adopted and preferred to construct a reliable new spanning tree.

To decrease the communication time and transceiver movement, this algorithm can be a centralized algorithm. It can also be used in a distributed fashion but may cost more time. The algorithm is presented in Table III.

Proposition 2: The complexity of our reconfiguration algorithm is $\mathcal{O}(E * V^2)$.

Proof: It takes $\mathcal{O}(V^2 * E)$ to find the reducing edge for every failed link in the graph. Finding the best bottleneck matrix takes $\mathcal{O}(V^2)$ and computing eigenvector takes (V) . So the whole complexity is $\mathcal{O}(E * V^2)$. □

But in the best case, with the knowledge from the bootstrapping algorithm, we already know reducing edges for every edge in the spanning formed by bootstrapping algorithm. Hence, in this case, just communication of failure information and transceiver movement time is required.

V. SIMULATION AND RESULT

A. Simulation Setting

In this section, we present simulation of our algorithms and compare them with alternative spanning tree construction algorithms in FSO network. We assume that FSO nodes are stationary and link reliability threshold $\mathcal{L}_{th}$ is set to be 0.9 which is often required for a reliable communication.

We abstract FSO network as a grid which is 100*100 and nodes are generated in this grid randomly. Edge weights are uniformly distributed between 0.9 and 1. Degree constraint on every node is uniformly distributed between x and y. The simulation was implemented in C++.

We provide a comparison with two alternative algorithms: (i) bottom-up algorithm (BU) [5]; (ii) fragment selection and merging algorithm (FSM) [3]. We compare two aspects to measure how these schemes work in FSO network: (i) algebraic connectivity; (ii) average weight of the spanning tree. We generate 100 random graphs and use these algorithms to obtain a spanning tree from which algebraic connectivity and average link weight are calculated.

B. Performance Analysis

The simulation results demonstrate the advantages of our algorithms over other algorithms. Fig. 1 and Fig. 2 illustrate performance of our algorithm compared with BU and FSM with degree constraint $x = y = 5$.

In Fig. 1, we plot algebraic connectivity for random networks with size ranging from 20 nodes to 50 nodes. The value of each point is the average of 100 random graphs. We can see that as the number of nodes increases, the algebraic connectivity decreases, which is consistent with Corollary 1.1 in [10] which asserts that adding a vertex will decrease algebraic connectivity for the graph. As we can see from Fig. 1, our algorithm produces a spanning tree with higher algebraic connectivity. It achieves 28.76% normalized improvement of algebraic connectivity over FSM, and 66.4% normalized improvement over BU. In Fig. 2, we plot average edge weight of resulting trees. The edge weight represents the reliability of FSO links. Fig. 2 illustrates that our Maximum Algebraic Connectivity (MAC) algorithm generates more robust spanning trees with higher average link weight. Our algorithm achieves 1.36% normalized improvement on average link weight over FSM, and 4.14% normalized improvement over BU.

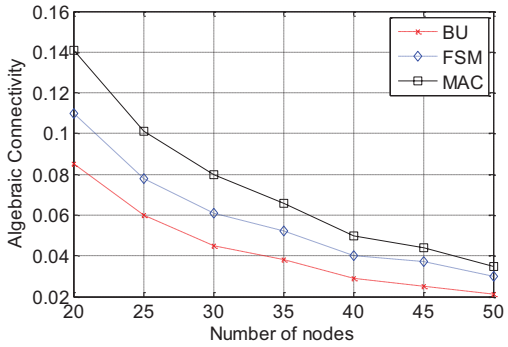

Figure 1. Algebraic connectivity of tree with degree constraint 5

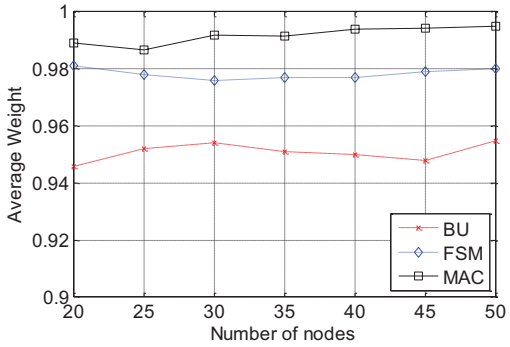

Figure 2. Average edge weight of tree with degree constraint 5

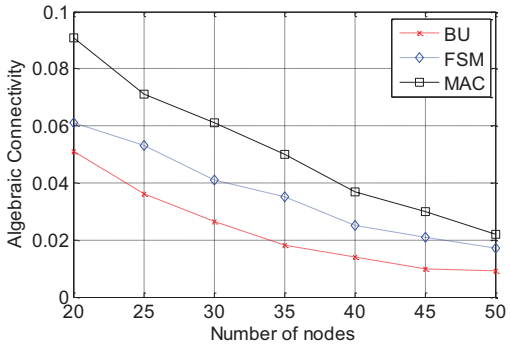

Figure 3. Algebraic connectivity of tree with degree constraint from 2 to 4

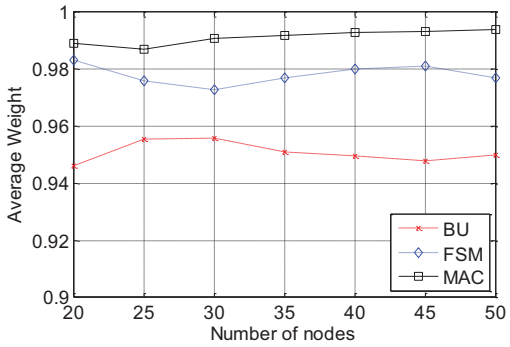

Figure 4. Average edge weight of tree with degree constraint from 2 to 4

Fig. 3 and Fig. 4 illustrate the performance of our algorithm compared with BU and FSM when $x = 2$, $y = 4$, which means degree constraint of each node is between 2 and 4. Nodes and links are generated in this grid randomly.

The simulation results show that our algorithm performs better than other alternative scheme in FSO network in both aspects: algebraic connectivity and average link weight. The BU algorithm does not take into account the edge weight which is the reason why it achieves the lowest average weight; FSM focuses on algebraic connectivity solely. However, in our algorithm, we consider both individual links' weight, which is also an important indicator of quality of FSO networks, and the algebraic connectivity of the robust spanning tree.

VI. Conclusion

In this paper, we studied the problem of building robust degree constrained spanning tree for FSO networks. Initial configuring, also called bootstrapping, and reconfiguration are two crucial parts of topology control in FSO networks; algorithms for bootstrapping and reconfiguring are both developed. The problem of building a degree constrained spanning tree with maximum algebraic connectivity and edge weight are formulated as 0-1 ILP problem and is proven to be NP-hard. For this NP-hard problem, we design fast bootstrapping and reconfiguration algorithms for FSO networks. Our simulation results show that the algorithms presented in this paper outperform other alternative algorithms for building spanning tree in FSO networks in terms of both algebraic connectivity and edge weight.

Acknowledgment

This work is supported in part by the U.S. National Science Foundation (NSF) under Grant CNS-1145446, and through the NSF Wireless Internet Center for Advanced Technology at Auburn University.

References

[1] J. Juarez, A. Dwivedi, A. Mammons Jr., S. Jones, V. Weerackody, and R. Nichols, "Free-space optical communications for next-generation military networks," *IEEE Commun. Mag.*, vol. 44, no. 11, pp. 46–51, Nov. 2006.

[2] J. Knowles and D. Corne, "A New Evolutionary Approach to the Degree-Constrained Minimum Spanning Tree Problem," *IEEE Trans. Evol. Comput.*, vol. 4, no. 2, pp. 125–134, July 2000.

[3] I. K. Son, S. Kim and S. Mao, "Building Robust Spanning Trees in Free Space Optical Networks," in *Proc. IEEE MILCOM'10*, San Jose, CA, Oct./Nov. 2010, pp.1397–1402.

[4] A. Ghosh and S. Boyd, "Growing well-connected graphs," in *Proc. IEEE CDC'06*, San Diego, CA, Dec. 2006, pp.6605–6611.

[5] F. Liu, U. Vishkin, and S. Milner, "Bootstrapping free-space optical networks," *IEEE J. Sel. Areas Commun*., vol. 24, no. 12, pp. 13–22, Dec. 2006.

[6] S. Gurumani, H. Moradi, H. H. Refai, P. G. LoPresti, and M. Atiquzzaman, "Dynamic Path Reconfiguration among Hybrid FSO/RF Nodes," in *Proc. IEEE GLOBECOM'08*, New Orleans, LA, Dec. 2008, pp. 1–5.

[7] P. C. Gurumohan and J. Hui, "Topology design for free space optical networks," in *Proc. IEEE ICCCN'03*, Dallas, TX, Oct. 2003, pp. 576–579.

[8] H.E. Nistazakis, E.A. Karagianni, A.D. Tsigopoulos, M.E. Fafalios, and G.S. Tombras, "Average Capacity of OpticalWireless Communication Systems Over Atmospheric Turbulence Channels," *J. Lightwave Techno.*, vol. 27, no. 8, pp. 974–979, Apr. 2009.

[9] D. Mosk-Aoyama, "Maximum algebraic connectivity augmentation is NP-hard," *Oper. Res. Lett.*, vol. 36, no. 6, pp. 677–679, Nov. 2008.

[10] S. Kirkland and M. Neumann, "Algebraic connectivity of weighted trees under perturbation," *Linear and Multilinear Algebra*, vol. 42, no. 3, pp. 187–203, Jun. 1996.