Relay Placement for Minimizing Congestion in Wireless Backbone Networks*

Abhishek Kashyap, Fangting Sun, Mark Shayman
Department of Electrical and Computer Engineering, University of Maryland, College Park MD 20742
Email: {kashyap, ftsun, shayman}@glue.umd.edu

***Abstract*— Wireless optical networks are being increasingly used in the backbone of hierarchical ad hoc networks. We consider the problem of minimizing the congestion in wireless optical (FSO) backbone networks by placing controllable relay nodes. We propose algorithms for placement of relays in the network under node interface constraints. The interfaces at each backbone node are limited, thus limiting the number of neighbors a node can have. We come up with algorithms to formulate the problem as a constrained knapsack problem, and propose algorithms to solve it. We use the mathematical technique of rollout to achieve better performance than the heuristics. We show by simulations that our algorithms significantly outperform some greedy algorithms, and a small number of relay nodes (when placed using our algorithms) can lead to a significant reduction in the congestion in the network.**

Keywords: FSO Networks, Congestion Control, Topology Control, Rollout, Multi-Commodity Flow, Constrained Knapsack.

I. Introduction

Hierarchical ad hoc networks have been proposed to provide scalability to ad hoc networks [1], [2], [3]. A hierarchical ad hoc network consists of mobile nodes grouped into clusters with some nodes (or a single node) designated as cluster-heads for each cluster. Such a network consists of two layers, one for intra-cluster communication, and one for inter-cluster communication through cluster-heads (which form a backbone network). Typical deployment of hierarchical networks are battlefield networks and extension of MANs (*last mile*). Wireless optical links are gaining a lot of attention for use in ad hoc backbone networks due to their attractive characteristics. Free-space optics (FSO) technology provides 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 [4]. These characteristics make them more suitable for wireless backbone networks than RF and wireline optical links, as RF links do not offer the high bandwidth required for backbone networks, while wireline optical links cannot be setup quickly and thus cannot be used in a mobile network.

In a wireless backbone network with mobile nodes, the topology of the network needs to be reconfigured as nodes move, to maintain the network performance at a desired level. FSO links offer the flexibility of fast tracking and setup of links [5]. Thus, the topology can be modified quickly to suit the current backbone node locations. Also, in a backbone network designed according to an estimated traffic profile (aggregate traffic), the traffic profile may change over time. Thus, it is necessary to adapt the network topology to achieve the desired performance for the modified traffic profile. This is known as topology control. There has been recent work on topology control in wireless optical networks [6], [7], [8], [9], [10]. The papers address the problem of topology control and routing for a given traffic profile. They consider the problem of changing the topology of the whole network to maximize the throughput or minimize congestion for a given traffic profile. The objective of finding a minimum congestion and minimum physical layer cost network (defined as total BER in the network) is considered in [9].

We assume we have no control over the backbone nodes. In such a network, even if the topology design is optimal initially, the performance is expected to degrade over time as nodes move and traffic patterns change. There are transmission range and interface constraints on the backbone nodes. A node can connect only to nodes within its transmission range. The interface constraint is due to the limited number of transmit and receive interfaces the nodes have, which limits the number of links each node can have. Thus, desired links cannot always be established. We propose the use of relays to counter transmission range constraints and propose algorithms to position them in the network to improve the network performance. They are used for forming additional links between backbone nodes. The relays are added such that they form links with backbone nodes without violating the interface constraints. We can change their placement and links to do topology control and improve the network performance. We measure the network performance in terms of the maximum link load (which we call congestion) in the network for the current traffic profile.

We model the problem as a constrained knapsack problem [11]. A constrained knapsack problem is to pick a subset of items to maximize the total profit while obeying capacity constraints, and constraints on set of items which can be picked together. The profit and weight values are defined for each item, and capacity is a bound on the total weight. We define a set of items for our problem, propose algorithms to calculate the item profits, and propose greedy heuristics to find a maximum profit solution. We improve the heuristics considerably by using the technique of rollout [12] on the heuristics.

*This research was partially supported by AFOSR under grant F496200210217 and NSF under grant CNS-0435206.

The rollout algorithms are mathematically guaranteed to work better than the heuristics. The simulation results show that a significant reduction in congestion can be achieved by placing a small number of relay nodes using our algorithms.

The paper is organized as follows: Section 2 presents the network model and problem statement. Section 3 describes the framework and algorithms for placement of relay nodes. Section 4 describes the proposed rollout algorithms. Section 5 gives the simulation results and Section 6 concludes the paper.

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 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 nodes within transmission range. We assume the transmission range of all nodes to be the same. The wireless links are unidirectional. 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. Two nodes can connect only if there is a free transmit interface at the tail node and a free receive interface at the head node. We assume the topology is given, and it is connected.

We assume we are given an estimate of the aggregate traffic between the backbone nodes. We call the estimated traffic matrix as the traffic profile, where each *profile entry* is a traffic demand between a source-destination pair. We are given K relay nodes (with same transmission range and interface constraints as the backbone nodes), which we can position anywhere in the network. We address the problem of locating the relays in the given topology to minimize the maximum link load (which we call congestion) in the network for the given traffic profile. Let the graph formed by adding relays (and deciding which additional links to form) to the network $G = (V, E)$ be denoted by $G' = (V', E'), V \subseteq V', E \subseteq E'$. We do not allow breaking any existing edges of G. Let the location of each relay node $r_k, k \in \{1, .., K\}$ be denoted by (X_k, Y_k). Let the routing being used be denoted by f, and the congestion for the routing f on the graph computed by adding the relays to G be denoted by $\sigma_{(G',f)}$. The routing f in our model is multi-path routing, and it specifies the flow on each link for each profile entry. The problem we solve is to find the locations of the relays to achieve the objective stated in Equation 1, subject to the interface constraints. Note that the problem also includes connecting the relay nodes with the appropriate backbone nodes (to get G') once the relay nodes are placed in the network. Also, the routing output is integrated with the placement of nodes as the routing changes as G' changes.

$$\min_{(X_k,Y_k),\forall k} \sigma_{(G',f)} \tag{1}$$

Periodically, as the positions of the backbone nodes change, new positions for the relays are calculated, and they are moved to their new locations. Their current locations are not taken into account while calculating the new positions.

III. Framework and Algorithms

Since nodes cannot connect to nodes outside the transmission range, we use relays to form additional links between backbone nodes outside each other's range. More than one relay may be required to form an additional link between two backbone nodes, depending on the distance between them. We restrict the placement of the relays to the lines joining the backbone nodes, so minimum number of relays is needed to form each link. As the number of interfaces is scarce, we allow the relays only to form a link between the two backbone nodes for which they are added. We do not allow the relays to form links with other backbone nodes and relays in their transmission range. We use new relays every time we add a link between two backbone nodes, i.e., we do not allow relays already connecting two backbone nodes to form additional links. Thus, the resulting topology can be modelled to have the same vertices as G (backbone nodes), and additional edges (formed by using relays). The resulting graph is denoted by $G' = (V, E')$, $E \subseteq E'$.

We assume the initial topology is given. The backbone nodes have limited interfaces. Thus, the number of free interfaces in the network is very small and the pairs of nodes between which we can form additional links is much smaller than N^2, N being the number of backbone nodes. We enumerate the pairs of nodes between which we can form additional links. We call this list as the *candidate list* $\mathcal{L}$. The tail node of a link in the candidate list should have a free transmit interface, and the head node should have a free receive interface. We denote each link l by a triplet: $\{t, h, c\}$, t represents the tail node of the link, h represents head node of the link and c represents the minimum number of relays needed to form the link. The cost c can be calculated as in Equation 2, where TR is the transmission range of each node and $|l|$ is the length of link l.

$$c = \lceil \frac{|l|}{TR} \rceil - 1 \tag{2}$$

The problem reduces to finding a set of links from the candidate list such that adding those links minimizes the congestion in the network, and the total cost of the set is within K, the given number of relays. The set of links should be chosen such that adding it does not violate the interface constraints. If we could assign a *profit* to each pair in the set $\mathcal{L}$, which represents the reduction in congestion by adding that link, the problem would reduce to a constrained knapsack problem. The items to be packed are the links in $\mathcal{L}$, and the capacity of the knapsack is the number of relays K. The constraints of this problem are the number of free interfaces at backbone nodes. In this knapsack problem, depending on the number of free interfaces at nodes, there is a restriction on the links which can be co-formed. As an example, if a node has

TABLE I
NOTATION

Symbol	Definition
K	Number of relay nodes
L	Number of links in the network
N	Number of backbone nodes
M	Number of traffic profile entries
TI_n	Number of free transmit interfaces at node n
RI_n	Number of free receive interfaces at node n
b_i	Traffic demand for profile entry i
x_i^l	Demand of profile entry i routed on link l
σ	Maximum link utilization
s_i, d_i	Source and destination nodes for profile entry i
O_n, I_n	Set of outgoing and incoming edges at node n

two free transmit interfaces, and it is a tail node for four pairs in the candidate list, only two of those pairs may be present in the final solution. An easier version of this problem [11], in which constraints are only on pairs of items, is NP-Hard [13].

The reduction in the congestion in the network by adding each link in $\mathcal{L}$ depends on the other links that are added to the network as the routing changes depending on the combination of links being added to the network. Thus, there is no optimal way to assign independent profit values to the links to formulate the problem as a constrained knapsack problem. Later in this section, we propose heuristics which model the problem as constrained knapsack by assigning heuristically calculated profit values.

We start by describing the routing we use: the routing minimizes our performance measure of a topology, i.e., the congestion in the network.

A. Minimum Congestion Routing on a Fixed Topology

The objective of the algorithm for finding relay locations is to minimize the congestion in the network. Thus, we use a routing algorithm that minimizes the congestion while routing the whole traffic profile on the network.

We formulate the routing problem as a multi-commodity flow (MCF) problem [14], treating each profile entry as a commodity, which can be split over multiple paths. The notations are given in Table I, and explained below. Let there be M commodities (the value of commodity i is the profile entry demand b_i), N nodes and L links in the network. Let x_i^l be the amount of commodity i routed through link l. Let the set of outgoing and incoming links at node j be denoted by O_j and I_j respectively. Let s_i and d_i represent the source and destination of profile entry i. Equation 3a achieves the objective of minimizing the congestion value (σ), along with the constraints of Equation 3b. Equation 3b ensures that total traffic on any link does not exceed σ. Equations 3c, 3d and 3e represent the flow conservation laws at transit nodes, source node and destination node respectively for each commodity. Equation 3f gives the bounds on the variables. As we try to minimize σ, σ will take the maximum value of link load in the network (due to constraints of Equation 3b).

$$\text{minimize} \quad \sigma \tag{3a}$$

$$\text{s.t.} \sum_{i=1}^{M} x_i^l \leq \sigma, \quad \forall l \in \{1,..,L\} \tag{3b}$$

$$\sum_{l \in I_j} x_i^l \quad = \sum_{l \in O_j} x_i^l,$$
$$\forall j \in \{1,..,N\} - \{s_i, d_i\}, \forall i \in \{1,..,M\} \tag{3c}$$

$$\sum_{l \in O_j} x_i^l - \sum_{l \in I_j} x_i^l = b_i, j = s_i, \forall i \in \{1,..,M\} \tag{3d}$$

$$\sum_{l \in O_j} x_i^l \ = 0, j = d_i, \forall i \in \{1,..,M\} \tag{3e}$$

$$\sigma \geq 0, \ 0 \leq x_i^l \leq b_i \tag{3f}$$

B. Greedy Algorithm

We start with a natural greedy algorithm for placing the relays in the network. The algorithm attempts to form the links in increasing order of cost (number of relays needed) without violating the interface constraints (TI_n, RI_n being the number of free transmit and receive interfaces at node n) while free relays are available. This algorithm maximizes the number of additional links formed in the network. The algorithm is as follows:

Algorithm 1 Greedy Algorithm

1: $G'(V, E') = G(V, E)$
2: Sort the candidate list $\mathcal{L}$ by increasing cost
3: Set number of free relays $R = K$
4: For all links $l = \{t, h, c\} \in \mathcal{L}$:
- if $R \geq c$, $TI_t > 0$, $RI_h > 0$
  - $E' = E' \bigcup \{t, h\}$
  - $TI_t = TI_t - 1$
  - $RI_h = RI_h - 1$
  - $R = R - c$

5: Output G'

This heuristic does not take the traffic profile into account. The traffic profile affects the importance of the links in the candidate list as we are minimizing the congestion value for a given traffic profile. In the next subsection, we propose a heuristic which takes the traffic profile into account.

C. Traffic Based Greedy Algorithm

Traffic Based Greedy Algorithm (TBGA) assigns *profit* values to links in the candidate list $\mathcal{L}$ based on the traffic profile. Then it sorts $\mathcal{L}$ in decreasing order of $profit/cost$ and forms the links in that order. For each link $l = \{t, h, c\} \in \mathcal{L}$, $cost$ is the number of relays required to form a link (c), and $profit$ represents the total traffic entering the tail node t that is destined for the head node h. The traffic from node t to node h is calculated using flow decomposition [15] for all traffic demands, which gives a set of paths (and corresponding flow

values) for each traffic demand. This decomposition is non-unique, but any solution gives the correct value of traffic going from node t to node h. Adding a link between the tail and head nodes of the link l is expected to divert a significant part of this traffic through the added link, thus we assign this as the profit. Once the profit values have been calculated, the algorithm is the same as a common greedy heuristic for constrained knapsack problems [11]. TBGA is given in Algorithm 2.

Algorithm 2 Traffic Based Greedy Algorithm (TBGA)

1: $G'(V, E') = G(V, E)$
2: Solve the MCF of Equation 3 on the initial topology G to get a routing f
3: For each link $l = \{t, h, c\} \in \mathcal{L}$:
   1) Perform flow decomposition [15] to get paths for each profile entry. Let the set of paths for profile entry i be P_i, and let x_i^p denote the demand routed on path $p \in P_i$.
   2) Find the paths (for all profile entries) containing t and h in that order. Denote the set of paths as P^l.
   3) Set profit (r_l) of the link as the total traffic flowing through the paths in P^l (Equation 4). Here, $\mathcal{I}_{\{E\}}$ is one if event E is true, zero otherwise.

$$r_l = \sum_{i=1}^{M} \sum_{p=1}^{|P_i|} x_i^p \mathcal{I}_{\{p \in P^l\}} \tag{4}$$

4: Sort the candidate list by decreasing r/c ratio
5: Set number of free relays $R = K$
6: For all links $l = \{t, h, c\} \in \mathcal{L}$:
   - if $R \geq c$, $TI_t > 0$, $RI_h > 0$
     - $E' = E' \bigcup \{t, h\}$
     - $TI_t = TI_t - 1$
     - $RI_h = RI_h - 1$
     - $R = R - c$
7: Output G'

D. Extended Traffic Based Greedy Algorithm

We extend TBGA by changing the scheme of assigning the profit to each link. The profit is assigned according to the reduction in congestion achieved by forming the links in the topology independently. The procedure is as explained below:

1: Solve the MCF of Equation 3 on the initial topology G. Let the optimal congestion value returned be σ.
2: For each link $l = \{t, h, c\} \in \mathcal{L}$:
   - $G''(V, E'') = G(V, E)$
   - $E'' = E \bigcup \{t, h\}$
   - Solve the MCF on G''. Let the optimal congestion value returned be σ'.
   - Assign profit r_l as $\sigma - \sigma'$

ETBGA is computationally more expensive than TBGA, but gives better results as the profit values used are more accurate. The step of calculating profit for each link takes $O(MNE)$ time in TBGA, while it takes $O((MN + E)^{3.5})$ in ETBGA, as the MCF needs to be solved in ETBGA for each profit calculation, which is a linear program (LP) (we use interior point methods for solving the LP [16]). Here, N is the number of vertices, E is the number of edges and M is the number of profile entries in the network. Thus, the worst case time complexity of ETBGA is much worse than TBGA. In practice however, the LP solving algorithms are much faster than the worst case bound suggests. Thus, ETBGA is slower than TBGA, but it has a reasonable running time in practice for networks of moderate size. As the simulations will show, ETBGA performs better than TBGA.

IV. Rollout Based Algorithms

We propose another set of heuristics that use the rollout [12] technique to extend the heuristics proposed in the previous section. We start by explaining the basic rollout algorithm, followed by the application of the technique to our problem.

A. Basic Rollout Algorithm

Rollout is a general method for obtaining an improved policy for a Markov decision process starting with a base heuristic policy [12]. The rollout policy is a one step look-ahead policy, with the optimal cost-to-go approximated by the cost-to-go of the base policy. We use the specialization of rollout to discrete multistage deterministic optimization problems. Consider the problem of maximizing $G(u)$ over a finite set of feasible solutions U. Suppose each solution u consists of N components $u = (u_1, \cdots, u_N)$. We can think of the process of solving this problem as a multistage decision problem in which we choose one component of the solution at a time. Suppose that we have a heuristic algorithm, the so-called "base heuristic", that given a partial solution $(u_1, \cdots, u_n), (n < N)$, extends it to a complete solution $(u_1, \cdots, u_N)$. Let $H(u_1, .., u_n) = G(u_1, \cdots, u_N)$. In other words, the value of H on the partial solution is the value of G on the full solution resulting from application of the base heuristic. The rollout algorithm R takes a partial solution $(u_1, \cdots, u_{n-1})$ and extends it by one component to $R(u_1, \cdots, u_{n-1}) = (u_1, \cdots, u_n)$ where u_n is chosen to maximize $H(u_1, \cdots, u_n)$. Thus, the rollout algorithm considers all admissible choices for the next component of the solution and chooses the one that leads to the largest value of the objective function if the remaining components are selected according to the base heuristic.

It can be shown that under reasonable conditions, the rollout algorithm will produce a solution whose value is at least as great as the solution produced by the base heuristic. Note that the heuristic may be a greedy algorithm, but the rollout algorithms are not greedy as they make a decision based on the final expected value of the objective function, and not the increment to the value of the objective function at that decision step.

1) Index Rollout Algorithm: Index rollout seeks to optimize the order in which links in the candidate list are formed, as the order changes the number of free interfaces and the number

of relays left. We use the rollout on all the heuristics we proposed in the last section. When using TBGA and ETBGA, we calculate the $profit$ values for all links in the candidate list $\mathcal{L}$ only initially (before the start of index rollout), and not each time the heuristic is used. We use the term (base) heuristic to refer to all heuristics in this section. The index rollout algorithm works as follows: In the first step, the rollout algorithm forms a link $l_1 \in \mathcal{L}$ determined by the requirement that it minimizes the congestion when the base heuristic is used to complete the topology starting with l_1. The congestion value is computed for the topology output of the base heuristic using the MCF formulation of Equation 3.

Now, suppose that the links $(l_1, \cdots, l_{n-1})$ have been formed in this order by the rollout algorithm. In the next step, the rollout algorithm forms the link l_n determined by the requirement that it minimize the congestion when the base heuristic is used to complete the topology starting with $(l_1, \cdots, l_n)$. After forming a link at each step of index rollout, the number of free interfaces and relays is updated. At any decision step, a link that cannot be formed due to lack of interfaces at tail and head nodes or due to insufficient relays is discarded from the candidate list.

At the first step, the congestion for rollout is the same as that for the heuristic as we construct the whole topology according to the heuristic. The rollout algorithm works at least as well as the heuristic, as at each decision step, it always has the choice of going according to the heuristic which gives the congestion that was calculated at the previous step. Thus, the rollout performs at least as well as the heuristic in terms of the objective function (congestion).

The rollout runs the base heuristics $O(|\mathcal{L}|^2)$ times (except profit calculations, which are done once), where $|\mathcal{L}|$ is the number of links in the candidate list $\mathcal{L}$. Thus, the time taken by the rollout algorithms is $O(|\mathcal{L}|^2)$ times the base heuristics. The increase in computation time is justified by the improvement in results, as we will demonstrate by simulations. We do not give the worst case time complexity of any algorithm as it is dominated by the time complexity of LP solvers, which is $O(N^{10.5})$ for the MCF in the worst case. In practice, LP solvers like CPLEX [17] are very fast and thus the worst case time complexity is misleading.

V. Simulation Results

We generate a random network and a random traffic profile with the following parameters.

- Size of the network = 3km x 3km.
- Number of backbone nodes in the network = 20.
- Location of each node: Chosen uniformly randomly
- Transmission range of each node = 1km.
- Number of transmit interfaces at each node = 5 (same number of receive interfaces)
- Demand between each source-destination pair: Uniform random between 0 and 1

As the network is a wireless backbone network, a 20 node network is considerably big, considering the applications of FSO backbone networks. The results presented here should be the same if the number of backbone nodes, relays and network area are increased proportionally. We use one of the matching-based algorithms of [6] to generate the initial topology of the network. The algorithm does not take the traffic profile into account, and outputs a topology which maximizes the number of links in the network. As the number of links is maximized, the topology is expected to work well with most traffic profiles. Also, maximizing the number of links is expected to yield a routing with less congestion. We consider only the simulations where the initial topology in connected. We used CPLEX [17] to solve the multi-commodity flow linear program.

A. Evaluation for Varying Number of Relays

We first evaluate the performance of the algorithms for varying number of relays. The number of relays is varied from one to five. The simulations are performed 10 times, and the values are averaged over the 10 runs (where initial topology was connected). The number of profile entries is varied from 10 to 50. They are selected uniformly randomly from all possible source-destination pairs. We average the performance over the number of profiles as well (which represents the load in the network). Table II shows the average normalized congestion for different algorithms, averaged over all the simulations and over different number of profile entries (10-50). The normalization is done with respect to the congestion value in the initial topology for the given traffic profile for each run. Thus, the values represent the reduction in congestion achieved by each of the relay placement algorithms. TBGA works much better than the Greedy Algorithm, and ETBGA works much better than TBGA. The index rollout (which we also call rollout) gives a considerable improvement in performance for all the heuristics, with the rollout with ETBGA working the best, followed by rollout with TBGA and rollout with Greedy Algorithm. The rollout algorithms have a similar performance as the number of relays required is increased to five, as then the base heuristic used does not matter much because the number of relays is large. Also, using just one relay can bring the congestion down to 85% of the initial value, using two nodes can bring it down to 70% for rollout with TBGA, and to 60% if we use four relays. The decrease in congestion is much faster with the addition of relays initially, and then the decrease becomes slower. Thus, only a few relays are sufficient to reduce the congestion considerably.

Fig. 1 shows the variation of average normalized congestion values with respect to the number of relays when the number of traffic profile entries is fixed at 40. As in the average results presented before, the decrease in congestion values is fast initially, and slows down as the number of relays is increased. Also, all the rollout algorithms perform much better the corresponding base heuristics.

B. Evaluation for Varying Traffic Load

We now evaluate the algorithms under varying load conditions. The load is varied by varying the number of traffic profile entries (source-destination pairs) from 10 to 50. The number of relays is fixed at four. Fig. 2 shows the congestion

TABLE II
AVERAGE NORMALIZED CONGESTION

Relays	Greedy	Rollout Greedy	TBGA	Rollout TBGA	ETBGA	Rollout ETBGA
1	0.9985	0.8470	0.9428	0.8470	0.8470	0.8470
2	0.9316	0.7110	0.8979	0.7070	0.7585	0.6791
3	0.8860	0.6522	0.8469	0.6775	0.7262	0.6357
4	0.8292	0.6563	0.8207	0.6201	0.7291	0.5973
5	0.8143	0.6024	0.7682	0.5907	0.6405	0.5782

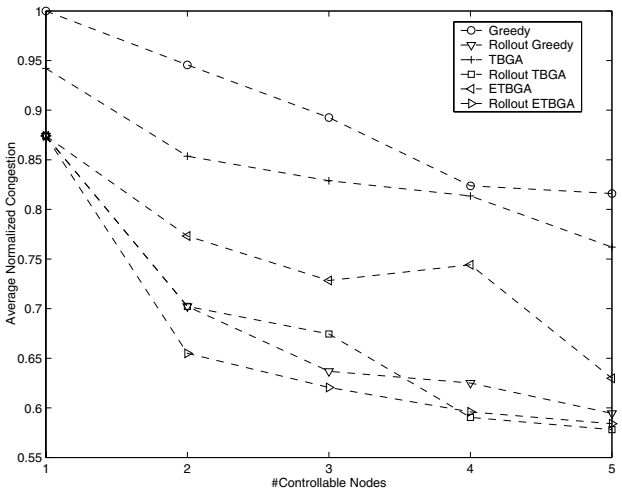

Fig. 1. Normalized congestion vs. number of relays for 40 profile entries

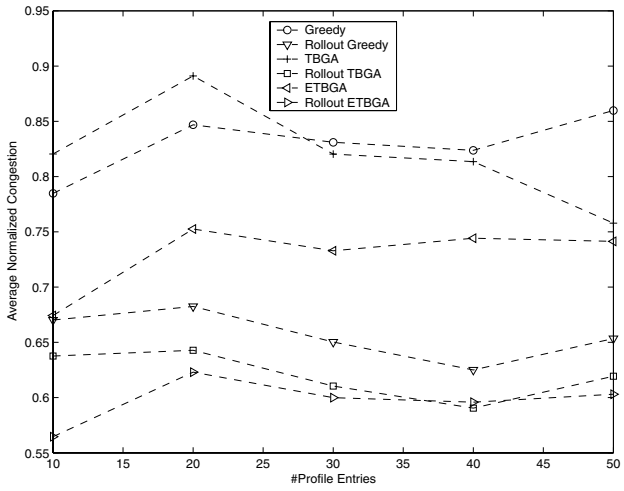

Fig. 2. Normalized congestion vs. number of profile entries for four relays

values for different algorithms normalized by the congestion on initial topology. The values shown are the average over 10 runs. There is no apparent relationship with the number of profile entries, which is a measure of the load on the network. Thus, the fractional reduction in the congestion achieved with respect to the case of no relays is similar for all load conditions considered.

VI. CONCLUSION

The problem of minimizing congestion in a backbone network by using relays has been considered. The relay placement problem is formulated as a constrained knapsack problem, and algorithms are proposed to compute the knapsack item profit values and compute the solution to the knapsack problem. We use the technique of rollout to improve the performance. The simulations show that there is a significant drop in congestion values by placing a small number of relays using our algorithms. The rollout algorithms can be used to obtain good solutions for constrained knapsack problems as well.

REFERENCES

[1] E. Perkins, *Ad Hoc Networking*. Addison-Wesley, 2001.

[2] S. Banerjee and S. Khuller, "A clustering scheme for heirarchical control in multi-hop wireless networks," *IEEE INFOCOM*, 2001.

[3] K. Xu, X. Hong, and M. Gerla, "An ad hoc network with mobile backbones," *IEEE ICC*, 2002.

[4] N. A. Riza, "Reconfigurable optical wireless," *IEEE LEOS*, vol. 1, pp. 70–71, 1999.

[5] T.-H. Ho, S. D. Milner, and C. C. Davis, "Fully optical real-time pointing, acquisition, and tracking system for free space optical link," *SPIE, Free-Space Laser Communication Technologies XVII, G. Stephen Mecherle, Ed.*, vol. 5712, pp. 81–92, 2005.

[6] A. Kashyap, S. Khuller, and M. Shayman, "Topology control and routing over wireless optical backbone networks," *Conference on Information Sciences and Systems*, 2004.

[7] A. Kashyap, M. Kalantari, K. Lee, and M. Shayman, "Rollout algorithms for topology control and routing of unsplittable flows in wireless optical backbone networks," *Conference on Information Sciences and Systems*, 2005.

[8] M. Kalantari, A. Kashyap, K. Lee, and M. Shayman, "Network topology control and routing under interface constraints by link evaluation," *Conference on Information Sciences and Systems*, 2005.

[9] J. Zhuang, M. J. Casey, S. D. Milner, S. A. Gabriel, and G. Baecher, "Multi-objective optimization techniques in topology control of free space optical networks," *IEEE MILCOM*, 2004.

[10] A. Desai and S. Milner, "Autonomous reconfiguration in free-space optical sensor networks," *IEEE JSAC Optical Communications and Networking Series*, vol. 23, no. 8, pp. 1556–1563, 2005.

[11] T. Yamada and S. Kataoka, "Heuristic and exact algorithms for the disjunctively constrained knapsack problem," *Information Processing Society of Japan Journal*, vol. 43, no. 9, pp. 2864–2870, 2002.

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

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

[14] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. The MIT Press, 2001.

[15] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Application*. Prentice-Hall, 1993.

[16] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2003.

[17] CPLEX, *http://www.cplex.com*.