A Multistart Randomized Greedy Algorithm for Traffic Grooming on Mesh Logical Topologies

Mauro Brunato[*][†] Roberto Battiti[*][†]

Abstract

In this paper we consider a logical topology design problem on DWDM optical networks where the traffic is quantized at sub-wavelength resolution and the critical factor to determine the fitness of a solution is the number of lightpaths required, that is proportional to the number of hardware modules to handle the traffic.

The problem has been shown to be NP-hard. We review some of the previous work in the field, examine a number of regular but unsatisfactory solutions and describe a greedy-based iterated heuristic belonging to the GRASP family to minimize the number of lightpaths.

At the end we present some experimental results that compare our GRASP heuristic with other greedy-based methods and with regular topologies.

Keywords — Logical topology design, Traffic grooming, Optimization algorithms, GRASP

1 Introduction

Significant advances in optical communications technology are leading to the creation of optical Wide Area Networks (WANs) with large link capacity. With the introduction of the Dense Wavelength Division Multiplexing (DWDM) technology, the whole bandwidth of every fiber is potentially available for transmission, as many independent signals can be accommodated on the same physical link by using densely packed adjacent wavelengths.

The link between two nodes in a WAN is composed of a thick bundle of fibers, usually in the hundreds; commercial systems allowing a few hundreds of DWDM wavelength carriers per fiber have already become available, and each carrier has a data transmission speed of various Gigabits per second. On the other hand, every single wavelength needs fairly expensive hardware equipment to be exploited (a tunable laser, a photodetector, fast electronics).

However, many low-bandwidth applications do not require a whole wavelength carrier; to allow a better traffic optimization link traffic should be quantized below the wavelength level by using Time Division Multiplexing (TDM). Packing many traffic units into less wavelengths is an effective way to reduce the number of hardware components, hence to cut off the overall cost.

As a consequence, technological concerns shift from improving the amount of bandwidth to the reduction of hardware cost through careful resource optimization.

Another advantage of the DWDM technology is that it allows a limited amount of static routing without electronic conversion. Upon arriving into a node, the signal travelling through a fiber is split into its composing wavelengths (see Fig. 1). Every wavelength can be treated in a different way: some of them are converted into electronic format for local treatment or for traditional routing, some other wavelengths are relayed through an internal fiber directly to an output fiber for retransmission. It is possible to set up a so-called *lightpath*, i.e. a direct connection between nonadjacent nodes acting as a logical one-hop link, where all intermediate nodes are transparent to it. Transparency of intermediate nodes is crucial in order to abstract the actual data flow pattern, called the *logical* or *virtual* topology and depending on lightpaths, from the *physical* fiber connection layer [6].

[*]Università di Trento — Dipartimento di Matematica, via Sommarive 14 — I-38050 Pantè di Povo (TN) — ITALY
[†]Email: `brunato|battiti@science.unitn.it`

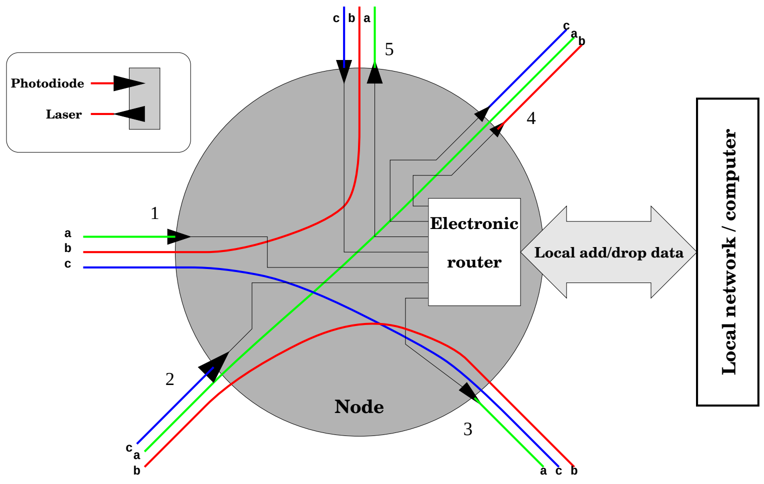

Figure 1: A network node

Traffic handling techniques including multiplexing and conversions of signals are collectively known as *grooming* techniques [1]. In particular, in the following sections the term *grooming* will refer to the act of merging many low-bandwidth traffic connections into a single wavelength by way of TDM.

Most studies about grooming techniques are focused onto the current optical infrastructure based on SONET/SDH rings. Many problems are addressed in this context. Among them, the above cited cost reduction problem has been introduced in [9, 8, 10] and a relevant research effort is still being spent on the subject with the proposal and evaluation of heuristic methods [14, 18, 3, 5, 2, 19, 4] under various assumptions about technological device availability, ring flexibility and traffic requirements. Grooming methods for general topologies have been discussed in [11], where the problem is treated as an Integer Linear Program and solved as a multicommodity flow problem for limited dimensions of the instances.

The focus of this work is to study the efficacy of greedy and iterated greedy algorithms in a simplified context. Therefore we shall only consider logical topology design, with no concern about its implementation on the physical layer. In other words, we assume that mapping a logical topology onto the physical layer is always feasible as in [11]. In a practical application this can be the case if abundant fibers have been installed, which is the case of many wide and metropolitan area networks. In a future extension of this work we will present problem formulations where the actual realization of the logical topology on a given physical topology is also considered.

The problem of logical topology design originates from the introduction of ATM networks and was recently boosted by introduction of an optical layer in digital packet-switched networks [7, 15] and, later, by the eventual introduction of all-optical systems [13, 12] and by the strong need of a specific IP-aware optical infrastructure [17].

The proposed optimization technique, is based on GRASP (Greedy Randomized Adaptive Search Procedure). GRASP is a metaheuristic for combinatorial optimization. GRASP usually is implemented as a multistart procedure, where each iteration is made up of a construction phase, where a randomized greedy solution is constructed, and a local search phase which starts at the constructed solution and applies iterative improvement until a locally optimal solution is found [16].

In Section 2 we state the problem that we to consider; in Section 3 we provide a few simple theoretical results such as lower bounds and values for regular topologies. Next, in Section 4 we introduce and discuss some optimization heuristics and we analyze some experimental results in Section 5.

2 Traffic grooming in logical topologies

Suppose we have a network with N nodes. We are not concerned with the physical topology of the network, although we assume that it is *connected*, i.e. it is always possible to establish a communication path between any

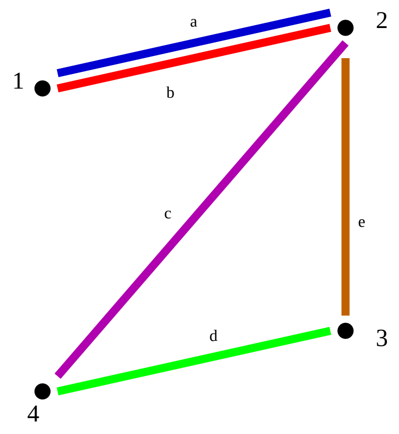

Figure 2: A logical topology over 4 nodes

two nodes.

We are also given a static *traffic pattern* in the form of a matrix (T_{ij}) whose entry T_{ij} contains the number of traffic units required for communications from node i to node j. A *traffic unit* is the unit of bandwidth considered, and all communication requirements are given as multiples of it.

We want to set up a *logical topology*, i.e. a set of lightpaths between nodes, in order to satisfy all traffic requirements, considering that every lightpath can carry at most c time-multiplexed traffic units.

Given a lightpath P from node i to node j , we define node i as its *source*, $i = \text{src}(P)$, and the node j as its *destination*, $j = \text{dest}(P)$. Many different lightpaths may have the same source and destination, to fulfill traffic requirements. A *chain of lightpaths* from node i to node j is a simple walk in the lightpath graph, i.e. a sequence of lightpaths $\mathcal{P} = (P_1, \dots, P_n)$ such that:

- the first lightpath originates from node i: $\text{src}(P_1) = i$,
- the last lightpath ends into node j: $\text{dest}(P_n) = j$, and
- two subsequent lightpaths join at the same node:

$$\forall k = 1, \dots, n-1 \quad \text{dest}(P_k) = \text{src}(P_{k+1}).$$

A traffic unit from node i to node j can be routed directly through a lightpath established from i to j, or it can be routed through a concatenation of lightpaths from i to j. For example, if $N = 4$ we could design the topology in Fig. 2, where two lightpaths are established through nodes 1 and 2. Traffic between nodes 1 and 4 may be routed through a concatenation of lightpaths, while traffic from 2 to 4 may be routed in many different ways.

Note that we are not concerned with the actual implementation of the lightpaths, i.e. the mapping of the logical topology onto the physical links. For example, the actual configuration could be that shown in Fig. 3; however, we suppose that the physical link capacity is enough to carry any reasonable logical topology and that the intermediate nodes traversed by a lightpath in the physical layer are completely transparent to the lightpath.

Given nodes i and j, the k-th traffic unit from i to j ($k = 1, \dots, T_{ij}$) shall be routed through a chain of lightpaths $\mathcal{P}_{ijk}$, made of one or more lightpaths, originating from i and ending in j. A *routing assignment* $(\mathcal{P}_{ijk})_{i,j=1,\dots,N;k=1,\dots,T_{ij}}$ is *suitable* if no lightpath appears more than c times.

Our objective is to minimize the number of lightpaths needed by the logical topology: this is equivalent to minimizing the number of critical electronic components, which is our main goal. The problem can be stated as follows.

> LOGICAL GROOMING PROBLEM — given a set of N nodes, a traffic pattern (T_{ij}) and a lightpath capacity c, find a suitable logical topology and a routing assignment for each traffic unit such that the number of lightpaths is minimum.

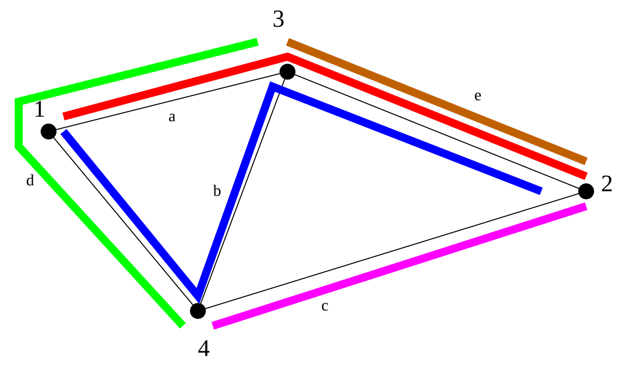

Figure 3: A mapping of the logical topology onto the physical layer

3 Exact results

In the following section we derive a lower bound for the number of lightpaths in any problem instance. We also consider some regular topologies and derive exact results for the number of lightpaths required in these cases. However, none of these solutions is near-optimal, and in Sec. 4 we shall introduce heuristics to obtain more satisfactory topologies.

3.1 Lower bound

The overall traffic carried by the network is, trivially,

$$\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}.$$

If every lightpath could be completely filled with c traffic units, and no traffic needed to be routed through more than 1 lightpath, then the number of lightpaths needed would be

$$\mathrm{LB} = \left\lceil \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}}{c} \right\rceil .$$

This number represents a lower bound for the number of lightpaths. In fact, if less than LB lightpaths were used then at least one traffic unit could not fit.

In case of uniform traffic $T_{ij} = T$, $i \neq j$, this means that the least number of wavelengths is

$$\mathrm{LB}_{\text{uniform}} = \left\lceil \frac{TN(N-1)}{c} \right\rceil ,$$

which results in a lower bound of 21 lightpaths for $T = 3$, $N = 8$ and $c = 8$.

3.2 Simple cases

3.2.1 Complete topology

One lightpath per couple of nodes, hop distance is 1 for every connection. Number of lightpaths for connection (i, j) is

$$L_{ij} = \left\lceil \frac{T_{ij}}{c} \right\rceil .$$

The total number of lightpaths is

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\left\lceil\frac{T_{ij}}{c}\right\rceil.$$

For instance, if $T_{ij}=3$ for $i\neq j$, $c=8$ and $N=8$, the total number of lightpaths is $7\times 8=56$.

3.2.2 Star topology

All lightpaths having a special node (e.g. node 1) as an endpoint. Edge $(i,1)$, $i=2,\dots,N$, carries all traffic outgoing from node i, and thus requires

$$\left\lceil\frac{\sum_{j=1}^{N}T_{ij}}{c}\right\rceil$$

lightpaths. Incoming traffic to node i is carried by edge $(1,i)$, requiring

$$\left\lceil\frac{\sum_{j=1}^{N}T_{ji}}{c}\right\rceil$$

lightpaths. In the case of uniform traffic T, edge $(i,1)$ carries traffic $T(N-1)$, thus requiring $L_{i1}=\lceil T(N-1)/c\rceil$ outgoing lightpaths. For the same reason, each of the $N-1$ peripheral nodes requires $L_{1i}=L_{i1}$ incoming lightpaths, so the overall number of lightpaths is

$$2\left\lceil\frac{T(N-1)}{c}\right\rceil(N-1),$$

which leads to 42 lightpaths for 8 nodes and traffic equal to 3 units per node pair.

3.2.3 Unidirectional ring topology

In this case node i, $i=1,\dots,N-1$, is connected to node $i+1$, while node N is connected to node 1. Let us label the nodes with their source edge labels; the overall traffic carried by edge i, $i=1,\dots,N$, is

$$C_i=\sum_{h=1}^{i}\left(\sum_{k=i+1}^{N}T_{hk}+\sum_{k=1}^{i-1}T_{hk}\right)+\sum_{h=i+2}\sum_{k=i+1}^{h-1}T_{hk}.$$

So the number of required lightpaths is

$$L_{ij}=\begin{cases}\left\lceil\frac{C_i}{c}\right\rceil & \text{if } j=(i \mod N)+1\\ 0 & \text{otherwise.}\end{cases}$$

In the case with uniform traffic $T_{ij}=T$, $i\neq j$, every node $(i,i+1)$ is crossed by a path of length 1 (the communications from node i to node $i+1$), two paths of length 2 and so on, up to $N-1$ paths of length $N-1$. So the total load of an edge is

$$T\cdot(1+2+\cdots+(N-1))=T\frac{N(N-1)}{2}.$$

Thus the complete number of lightpaths required is

$$L_{ij}=\begin{cases}\left\lceil\frac{T\frac{N(N-1)}{2}}{c}\right\rceil & \text{if } j=(i \mod N)+1\\ 0 & \text{otherwise.}\end{cases}$$

```
function assignPath (node i, node j) returns lightpathChain
    find shortest existing lightpath chain LPC
        from i to j with non-null spare load
    if not found LPC
        P ← new lightpath from i to j
        LPC = single hop chain (P)
    return LPC
```

Figure 4: the basic Greedy move

```
function setGreedyTopology
    erase all lightpaths and all routing information
    for each couple of nodes (i, j) in random order
        for each traffic unit $t \in \{1, \ldots, T[i][j]\}$
            LPC ← assignPath (i, j)
            route traffic unit t from node i to node j through LPC
            for each lightpath P in LPC
                increase load in P by 1
```

Figure 5: the Greedy topology algorithm

There are exactly N edges on the ring, so the total number of lightpaths is

$$N \left\lceil \frac{T\frac{N(N-1)}{2}}{c} \right\rceil,$$

accounting for 88 lightpaths if $T = 3$, $N = 8$ and $c = 8$.

4 Greedy and Iterated Greedy schemes

In the following, every lightpath is associated to a *load*, an integer value reporting the number of traffic unit routed to that lightpath. The *spare load* of a lightpath is the capacity c minus its current load, and accounts for the number of traffic units that can be added to the lightpath before saturating it. The *spare load* of a lightpath chain is the smallest spare load of its component lightpaths.

The basic component of our greedy scheme is function *assignPath*, shown in Fig. 4, which tries to allocate a chain of lightpaths to a traffic unit going from node *i* to node *j*. The function executes a breadth-first search from node *i* through the logical topology graph that is under construction until it gets to node *j*. Only lightpaths having spare capacity of at least 1 are considered (lines 2-3). If no lightpath chain can be determined (line 4), a new lightpath is created from node *i* to node *j* (line 5) and the lightpath chain is instantiated to that single path (line 6).

The overall greedy algorithm, which is shown in Fig. 5, begins by resetting all load and routing information (line 2) in order to start the assignment from scratch. Then for each couple of nodes and each traffic unit it invokes function *assignPath* (line 5) to retrieve a lightpath chain suitable for routing that traffic unit. Then it sets all routing information needed to use the lightpath chain (line 6) and it increases the load of every lightpath in it by 1 (lines 7-8).

The basic idea of our greedy scheme is that, whenever we need to allocate a route for a traffic unit, we want to use residual traffic capacity if possible (so that no new lightpaths must be created), otherwise we prefer a new lightpath with many spare units of traffic in the hope that future traffic allocations will use it. Of course this heuristic is not optimal. This can be seen experimentally (Sec. 5) and theoretically, as the NP-hard multicommodity flow problem can be reduced to this problem [11]. In particular, earlier assignments cannot take advantage of lightpaths

```
function GRASPIteration
  for each couple of nodes $(i, j)$ in random order
    for each traffic unit $t \in \{1, \dots, T[i][j]\}$
      $LPC \leftarrow$ routing of traffic unit $t$ from node $i$ to node $j$
      for each lightpath $P$ in $LPC$
        decrease load of $P$ by 1
        if load of $P$ is null
          delete lightpath $P$
      forget routing of traffic unit $t$ from node $i$ to node $j$
    for each traffic unit $t \in \{1, \dots, T[i][j]\}$
      $LPC \leftarrow assignPath\ (i, j)$
      route traffic unit $t$ from node $i$ to node $j$ through $LPC$
      for each lightpath $P$ in $LPC$
        increase load in $P$ by 1
```

Figure 6: the GRASP iteration algorithm

that have been created only later (this is the main problem with all greedy schemes), so some lightpaths are eventually underused.

To allow older assignments to take advantage of all lightpaths that have been created, we can repeatedly apply the same greedy scheme without restarting from scratch each time; as we can see in Fig. 6, for each couple of nodes we forget everything about their routing assignment while keeping all other lightpaths (lines 4-9), and rebuild it just as we did in the greedy scheme (lines 11-14). This time, however, all other couples of nodes mintain their current assignment, so many lightpaths are available when a suitable route mst be found from *i* to *j*.

By implementing the greedy scheme, followed by repeated application of the *GRASPIteration* function, we obtain an iterated optimization scheme similar to the GRASP technique [16]. In our case the procedure starts in a feasible state obtained by a greedy function. This function's randomness lies in the order the nodes are considered. Also the iteration scheme has some amount of randomness because of node ordering.

5 Experimental results

We have implemented the greedy and GRASP algorithms as discussed in Sec. 4 in C++, and we applied them to various types of traffic. In particular, we chose to consider a *uniform* pattern where traffic between each pair of nodes is constantly equal to 3 or 5, and a *server-type* traffic where all node pairs have a unit traffic requirement unless the source node is one of three designated servers,in which case the required traffic is 10 units. In the first case the traffic matrix is uniform; in the second case, all entries are equal to 1 with the exception of three rows, equal to 10. In both cases the diagonal is null.

$$T_{\text{uniform}} = \begin{pmatrix} 0 & 3 & 3 & \dots & 3 \\ 3 & 0 & 3 & \dots & 3 \\ & \vdots & & \ddots & \vdots \\ 3 & 3 & 3 & \dots & 0 \end{pmatrix} \qquad T_{\text{server}} = \begin{pmatrix} 0 & 1 & 1 & \dots & 1 \\ 1 & 0 & 1 & \dots & 1 \\ & \vdots & & & \vdots \\ 10 & 10 & 10 & \ddots & 10 \\ & \vdots & & & \vdots \\ 1 & 1 & 1 & \dots & 0 \end{pmatrix}$$

As a possible candidate for an alternate startup function we also implemented a random assignment function such that for every couple of nodes and every traffic unit a random lightpath chain was determined.

In Figures 7 and 8 we show the comparison among the greedy and random setup schemes, the two corresponding GRASP techniques (random- and greedy-initiated) and the regular star topology discussed in Sec. 3. Each value shown is the average of five runs; the iterated schemes were run 10000 times. The x axis represents the

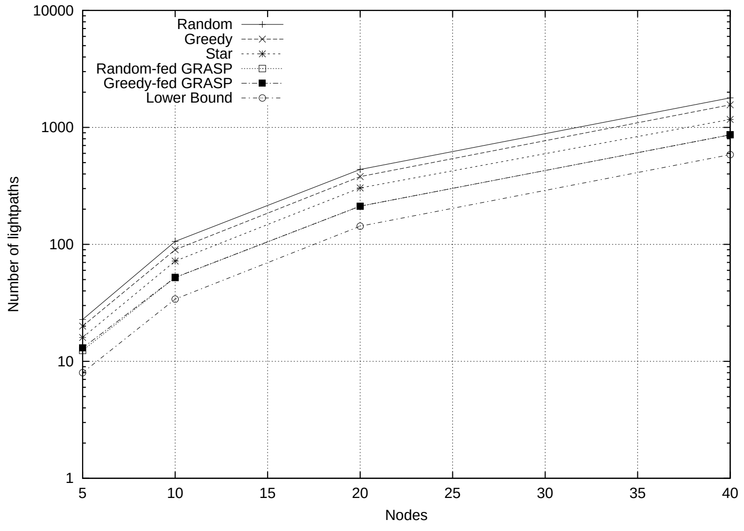

Figure 7: Uniform traffic, 3 traffic units per node pair, symmetric routing

size of the network, while the y axis indicates the number of lightpaths that were established by the algorithm. In this case, the algorithms had another constraint: as traffic is symmetric, we also keep a symmetric routing scheme by iterating through node couples (i, j) such that $i < j$. In fact, in the case of symmetric traffic we could use full-duplex optical links [11].

One run of 10000 GRASP iterations on a problem instance with uniform traffic equal to 5 and 20 nodes took 121 seconds on a Pentium III Windows machine with a GNU compiler and the Cygwin Unix API emulation libraries. However, we have no evidence that such a large number of iterations is necessary: in fact, the minimum was always reached in the first 100 iterations. Further investigation will allow us to identify a suitable number of iterations for each case.

In both cases (traffic 3 and 5), the greedy algorithm performs better than the random scheme. However, when the GRASP scheme is used the initial assignment is not important and the two curves overlap almost perfectly. In particular, when the traffic is equal to 5 the greedy-initiated GRASP scheme improves the solution by 10% in the 5-node case, up to 23% in larger graphs. Even though the star topology allows a lower number of lightpaths when compared to the greedy and random schemes, we have verified that, when used to initialize the GRASP algorithm, it does not lead to better results. Moreover, it has no randomness in it, so it is not suitable as a starting point for our search procedure.

In the case of server traffic, the symmetry constraint was not applied, and the results are shown in Fig. 9. In this case, too, the greedy scheme outperforms random assignment, but this is ininfluent when the GRASP algorithm is applied.

Conclusions

We have considered a logical topology design problem where the classical routing problem is complemented with sub-wavelength resolution of traffic. As far as we know, this problem was considered for mesh networks only in [11], while the two separate problems have been vastly analyzed in the literature (see Sec. 1).

We have analyzed a few regular topologies and we have defined an iterated greedy scheme. Experimental analysis has shown a greater improvement over simple greedy, random and regular topologies when applied to regular traffic schemes and simple server traffic models.

At present, the complete network is revised at every step of our GRASP approach. Future work will be dedicated to experimenting Local Search techniques to allow a smoother transition of the network logical topology towards

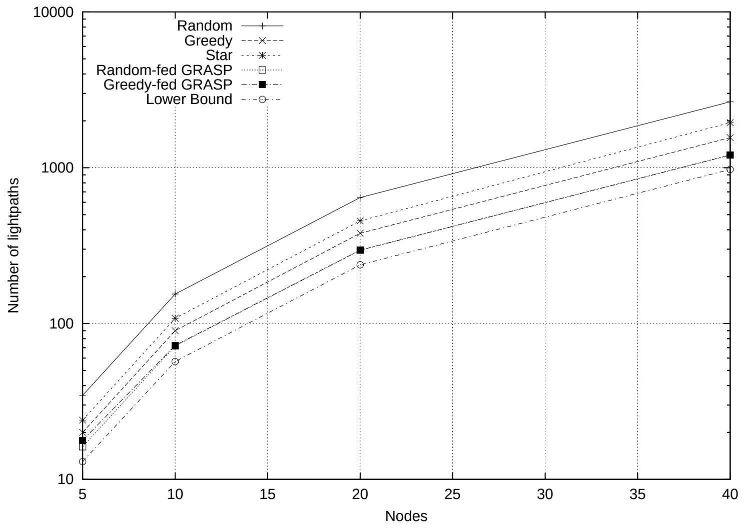

Figure 8: Uniform traffic, 5 traffic units per node pair, symmetric routing

the minimum, and possibly the ability to serve dynamically chaning traffic patterns while limiting the amount of lightpath reorganization required at every step.

References

[1] Richard Barr and Raymond A. Patterson. Grooming telecommunications networks. *Optical Networks Magazine*, 2(3), 2001.

[2] Roberto Battiti and Mauro Brunato. Reactive search for traffic grooming in WDM networks. In S. Palazzo, editor, *Proceedings of IWDC2001*, Lecture Notes in Computer Science. Springer-Verlag, September 2001.

[3] Randall Berry and Eytan H. Modiano. Reducing electronic multiplexing costs in SONET/WDM rings with dynamically changing traffic. *IEEE Journal on selected areas in communications*, 18(10):1961–1971, October 2000.

[4] Gruia Călinescu and Peng-Jun Wan. Traffic partition in WDM/SONET rings to minimize SONET ADMs. In *IWST 2001*, 2001, to appear.

[5] Angela L. Chiu and Eytan H. Modiano. Traffic grooming algorithms for reducing electronic multiplexing costs in WDM ring networks. *IEEE Journal of Lightwave Technology*, 18(1):2–12, January 2000.

[6] I. Chlamtac, A. Ganz, and G. Karmi. Lightpath communications: A novel approach to high bandwidth optical WANs. *IEEE Transactions on Communications*, 40(7):1171–1182, 1992.

[7] Aura Ganz and Xudong Wang. Efficient algorithm for virtual topology design in multihop lightwave networks. *IEEE/ACM Transactions on Networking*, 2(3):217–225, June 1994.

[8] Gerstel, Lin, and Sasaki. Wavelength assignment in a WDM ring to minimize the cost of embedded SONET rings. In *INFOCOM1998*, 1998.

[9] Gerstel and Ramaswami. Cost effective grooming in wdm rings. In *INFOCOM1998*, 1998.

[10] Ornan Gerstel, Rajiv Ramaswami, and Galen H. Sasaki. Cost-effective traffic grooming in WDM rings. *IEEE/ACM Transactions on Networking*, 8(5):618–630, October 2000.

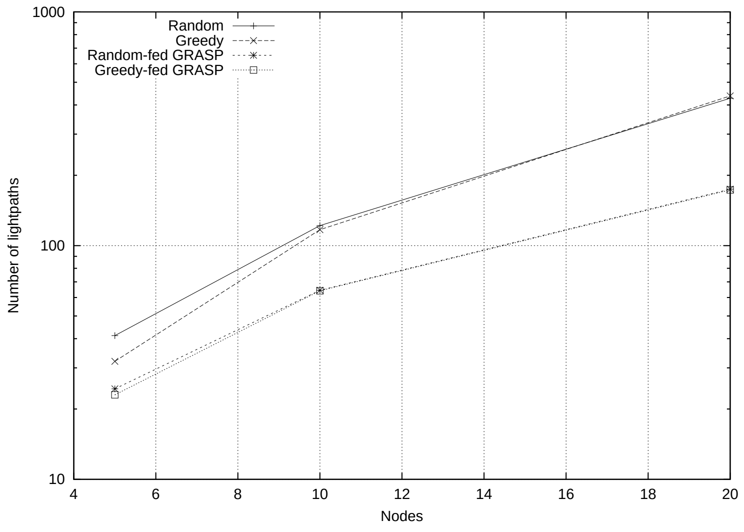

Figure 9: Server traffic

[11] V. R. Konda and T. Y. Chow. Algorithm for traffic grooming in optical networks to minimize the number of transceivers. In *IEEE HPSR2001*, 2001.

[12] Rajesh M. Krishnaswami and Kumar N Sivarajan. Design of logical topologies: A linear formulation for wavelength routers with no wavelength changers. *IEEE/ACM Transactions on Networking*, 9(2):186–198, April 2001.

[13] Emilio Leonardi, Marco Mellia, and Marco Ajmone Marsan. Algortihms for the topology design in WDM all-optical networks. *Optical Networks Magazine*, 1(1):35–46, January 2000.

[14] Liwu Liu, Xiangyang Li, Peng-Jun Wan, and Ophir Frieder. Wavelength assignment in WDM rings to minimize SONET ADMs. In *INFOCOM 2000*, pages 1020–1025, 2000.

[15] Rajiv Ramaswami and Kumar N. Sivarajan. Design of logical topologies for wavelength-routed optical networks. In *INFOCOM1995*, 1995.

[16] Mauricio G. C. Resende and Celso C. Ribeiro. Greedy randomized adaptive search procedures. In F. Glover and G. Kochenberger, editors, *State-of-the-Art Handbook in Metaheuristics*. Kluwer Academic Publisher, 2001. To appear.

[17] D. A. Schupke and D. Sellier. Lightpath configuration of transparent and static WDM networks for IP traffic. In *ICC2001*, 2001.

[18] Peng-Jun Wan, Gruia Călinescu, Liwu Liu, and Ophir Frieder. Grooming of arbitrary traffic in SONET/WDM BLSRs. *IEEE Journal on Selected Areas in Communications*, 18(10):1995–2003, October 2000.

[19] Xijun Zhang and Chunming Qiao. An effective and comprehensive approach for traffic grooming and wavelength assignment in SONET/WDM rings. *IEEE/ACM Transactions on Networking*, 8(5):608–617, October 2000.