Flow Trees: A Lower Bound Computation Tool with Applications to Rearrangeable Multihop Lightwave Network Optimization

Bülent Yener and Terrance E. Boult*

Columbia University
Computer Science Department
Technical Report CUCS-006-94

Columbia University Department of Computer Science, NYC, NY 10027.
Phone: (212) 939-7022, Fax: (212) 666-0140, e-mail: yener@cs.columbia.edu

Abstract

This paper presents a new method for computing the lower bounds for multihop network design problems which is particularly well suited to optical networks.

More specifically, given N stations each with d transceivers and pairwise average traffic values of the stations, the method provides a lower bound for the combined problem of finding optimum (i) allocation of wavelengths to the stations to determine a configuration, and (ii) routing of the traffic on this configuration while minimizing *congestion* - defined as the maximum flow assigned on any link.

The lower bounds can be computed in time polynomial in the network size. Consequently, the results in this work yield a tool which can be used in (i) evaluating the quality of heuristic design algorithms, and (ii) determining a termination criteria during minimization.

The lower bound computation is based on first building *flow trees* to find a lower bound on the total flow, and then distributing the total flow over the links to minimize the congestion.

Keywords: Multihop lightwave networks, network optimization, lower bounds, algorithms.

---

*T. Boult is moving to Lehigh University Department of Electrical Engineering and Computer Science, Bethlehem, PA, tboult@eecs.Lehigh.edu. This work supported in part by NSF PYI award #IRI-90-57951 with industrial support from IBM and Texas Instruments.

1 Introduction

In recent years there has been an increasing interest in optimization of multihop lightwave network design [9, 7, 4, 3]. This is primarily motivated by the vast bandwidth of a fiber which can be exploited by employing concurrency with *Wavelength Division Multiplexing* (WDM) [2, 6, 8]. In this paper we address the combined problem of routing and design, in linear multihop lightwave networks, and present tools to compute a lower bound for this problem. Particularly, we consider linear multihop lightwave networks without channel sharing in which tuning a pair of transceivers to the same wavelength establishes a *logical* connection. Therefore, allocation of the wavelengths to the stations constructs a *logical topology* (i.e., configuration) which can be realized by broadcast-and-select property of WDM lightwave networks.

The choice of the configuration may depend on the traffic characteristics. However once the logical topology is determined, the performance of a routing algorithm is limited by that topology. Therefore, the routing and the configuration are mutually dependent. The problem of finding an optimal configuration and routing is NP-Hard (see [4]) thus the previous approaches are based on heuristics which hopefully yield near-optimal solutions. Performance criteria to determine a "good" configuration can be based on minimizing the maximum total flow (i.e., *congestion*), on any edge as suggested in [9, 7, 3, 5] or based on minimizing the propagation delay [1, 3]. Since the problem can only be solved approximately, it is natural to ask about the quality of the approximation. One way of answering this is to empirically compare various heuristics on particular sample problems. Another is to to compare the approximate solution, for every problem instance to a good lower bound, such as those developed in this paper. In addition, the lower bound techniques developed herein are computationally attractive and could therefore be used as part of a termination criterion for such optimization heuristics.

This paper addresses the development of lower bounds on the combined problem of finding (i) allocation of wavelengths to the stations (configuration), and (ii) routing of the traffic on this configuration while minimizing the value congestion. Given a pairwise (average) traffic matrix T, number of stations N, and d transceivers at each station, we denote an *instance* of this combined problem by the triplet (T, N, d).

In this paper we present a sequence of polynomial time algorithms which, given an instance of the problem (i.e., $(T, N.d)$), compute lower bounds on the congestion for that problem instance. The computed lower bounds can be used in assessing the performance of heuristic solutions in lightwave network design, as well as in deciding termination of the heuristics (i.e., stop the heuristic when its performance is close enough to the computed lower bound).

Although it is presented in the context of lightwave networks, the techniques presented in this work can be applied—as a general tool—for deriving lower bounds in network design problems which have flow-based objective functions.

This paper is organized as follows. In the following section, we define the problem. In section 3, we present a novel technique for computing instant-specific lower bounds. Section 4 explains how to implement the suggested technique to obtain an efficient tool. The work is concluded in Section 5.

2 Problem Definition

In this section we define the problem precisely, explaining the formulation and parameters. Given average pairwise traffic T of N stations each with d transceivers, we consider the problem of how (i) to find a topology (i.e., a configuration), and (ii) how to route the traffic in T on this configuration with minimum congestion. The average traffic for N stations, is represented by the matrix $T = [t_{s,t}]$ such that each entry $t_{s,t}$ is the expected amount of traffic to be sent from a source station s to a destination t. (For simplicity we assume that $t_{i,j}$ values are normalized to integers). A configuration is a representation of a (logical) network topology in which each node corresponds to a station and each edge to a logical (physical) connection. Precisely, a configuration is a digraph of N nodes, with no self-loops, such that each node has equal[1] in/out degree which is d. Given the traffic matrix T and a configuration, routing R is *an* assignment of the traffic $t_{s,t}$ to some directed path(s) in the configuration from source s to destination t, for all s and t. Note that a directed edge in the configuration may carry the traffic of various s, t pairs. Hence we define *congestion* $z_{i,j}(R, T)$ on a directed edge (i, j) as the accumulated amount of traffic carried on this edge for routing R of traffic T. Let $Z = max_{(i,j)}\{z_{i,j}(R, T)\}$ be the maximum value of congestion.

1. $Z \geq \sum_k f^k_{(i,j)}, \forall k, \forall\, pair(i, j)$
2. $\sum_i f^k_{(i,j)} - \sum_i f^k_{(j,i)} = t_{k,j}\ \forall\ k, j$ where $k \neq j$.
3. $0 \leq f^k_{(i,j)} \leq (\sum_u t_{k,u})x_{i,j}$
4. $\sum_j x_{i,j} = \sum_j x_{j,i} = d, \forall\ i, j$

[1] For simplicity, it is assumed that each station has exactly d transceivers. The work herein can be generalized in various ways to handle non-uniform degree constraints.

5. $x_{i,j} \in [0,1]$

6. $f^i_{(i,j)} \geq 0$

There are two variables of interest: the (integer) flow variable $f^k_{i,j}$, the 0-1 integer variable $x_{i,j}$ which is 1, if there is an edge from i to j, 0 otherwise. We note that directly solving the above mixed integer programming problem computationally too expensive [4] (shown via transformation from *minimum cut linear arrangement* problem [10]). Thus approximate solutions are sought by heuristics. In the next section, we construct lower bounds of this problem to evaluate the performance of heuristics.

3 Lower Bounds

Given an instance of the problem (i.e., T, N, d) in this section we present a new technique to compute a lower bound (instance-specific) on the congestion Z (i.e., the *maximum flow* on an edge). First note that an immediate lower bound on the congestion can be found by choosing a node for which the incoming or the outgoing traffic is the maximum and dividing that amount by the degree d [9]. Precisely, let LB_I denote the immediate lower bound:

$$LB_I = \frac{\max_{i,j}\{\sum t_{i,j}, \sum t_{j,i}\}}{d} \tag{1}$$

In general, this lower bound may not be very sharp since it does not take into account the routing of the traffic. A better bound may be possible if the total flow in the network is considered.

In this paper we develop algorithms yielding provable lower bounds on the total flow and use these to get better lower bounds on the congestion. The tool for computing a lower bound uses *minimum flow trees* for each commodity and generalize to other flow-based objective functions. Our approach has the following two steps: (1) finding a lower bound on the *total flow* carried in the network for a given traffic matrix (using minimum flow trees), and (2) finding a lower bound on congestion, by dividing the lower bound on the total flow (computed at step 1), by the total number of edges Nd. This two step approach was also originally proposed in [9] though no proven lower bounds are given.

The discussion of the technique starts with the simplest structure, a flow graph, then introduces minimum flow trees and finally constrained minimum flow trees.

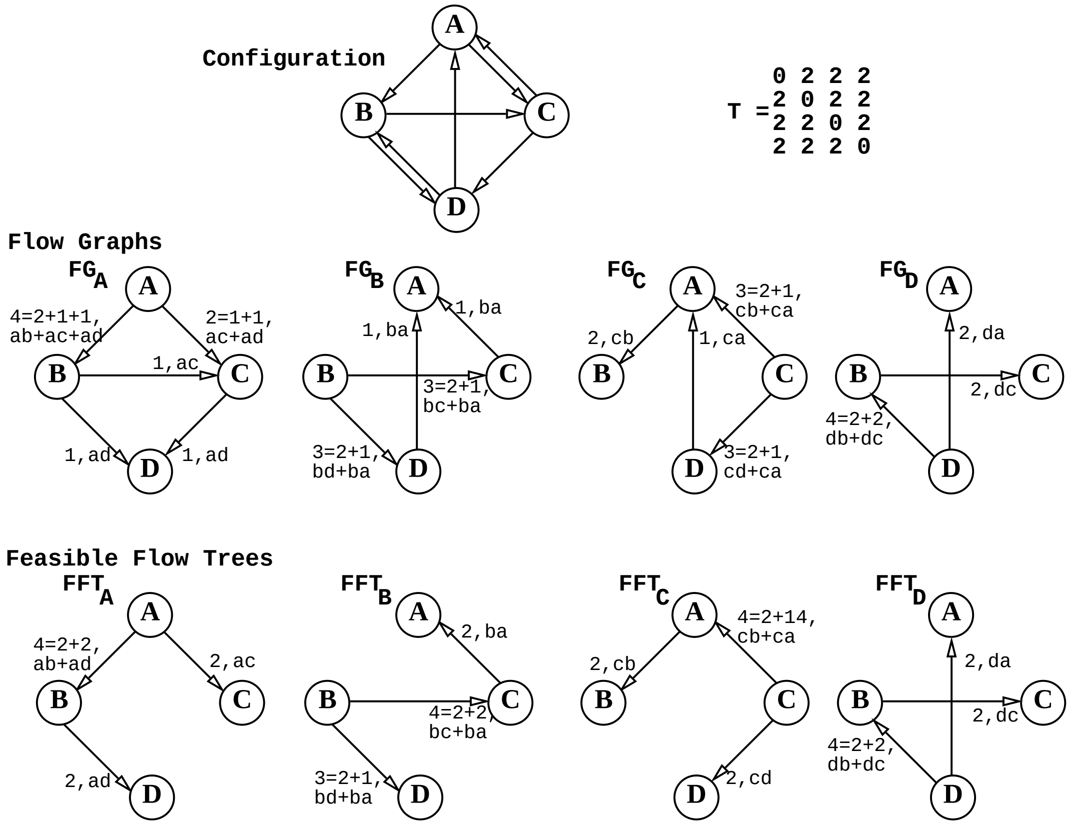

Figure 1: A feasible network (configuration) and a routing of the traffic on this configuration which induces the flow graphs. A set of feasible flow trees (FFTs) is also shown.

3.1 Flow Graphs and Flow Trees

Given a traffic matrix T, a configuration G, and a routing R of the traffic T on G, let FG_r denote a DAG called the *flow graph* for commodity r. FG_r is connected and can be constructed from an union of all the paths from the root node r to all others nodes, as specified by the routing R. Each arc (u, v) in FG_r is associated with a weight (i.e., its *cost*) which is the sum of the traffic routed on this edge from the root r.

For example consider Figure 1 in which each node has two units of traffic to be sent. A possible routing R of the traffic for each commodity i is shown on the corresponding flow graph FG_i. The weight on the edge (A, B) of the flow graph FG_A is 4 units is obtained as the sum of the traffic routed on this edge from the root A to the nodes B ($A \to B$ directly, 2 units), C ($A \to C$ via B, 1 unit), and D ($A \to D$ via B, 1 unit). One can easily check the total flow is 34 and the maximum congestion is 6 (e.g., edge BC).

Let FFT_r be a *feasible flow tree* rooted at node r which is obtained from a breath-first search of FG_r. Note that FFT_r is feasible since it uses a subset of the edges of the underlying (feasible) flow graph. However, the cost on the edges of a FFT_r may be different since the traffic is not split. Precisely, the cost on an edge (i, j) is the sum of $t_{r,j}$ and the traffic to all the nodes reachable from j. Figure 1 shows example $FTTs$ and the associated costs.

Let $level(j)$ of a node j be the length (number of hops) of the path from the root to j in the associated flow tree. Then we observe the following:

Observation 1 $Cost(FFT_r) = \sum_{(i,j) \in FFT_r} level(j) \times t_{r,j}$, *where (i,j) denotes a directed edge.*

Note that the flow tree FFT_i has an induced routing R such that all traffic flows from the root to each node of the tree. Let $Cost(G)$ be the total flow on a configuration G. Note that the distance in FFT_r from r to any other node j is no larger than the distance from r to j in FG_r. While splitting the traffic from r destined for node j (i.e., using multiple paths in FG_r) can reduce the congestion in FG_r, it cannot decrease the total flow. Moreover, if the paths used are not of the same length, such splitting will increase the total flow since traffic values will be propagated on more edges. For example, the total flow on the FFTs in Figure 1 is 33 and the maximum congestion is 6. From the above discussion we observe:

Observation 2

$$\begin{aligned} Cost(FG_r) &\geq Cost(FFT_r) \forall r \ and \\ Cost(G) &= \sum_r Cost(FG_r) \geq \sum_r Cost(FFT_r) \end{aligned}$$

This observations give a lower bound on the flow for a given traffic *and routing.* (Recall FG_r depends on the routing). What we desire, however, is a lower bound over all routings. But since we cannot get the minimal routing, we will obtain a lower bound by considering an easier problem: the one with a relaxed feasibility constraint. In particular, we allow each commodity node to choose *any edge* to route its traffic so as to minimize the flow needed for this routing. Each commodity is considered independently. A network (which will most likely be infeasible) can then be obtained from the union of the routes considering all the commodities. Informally, such a network would give a lower bound on the real total flow since we only relaxed the constraints.

Once such a network is obtained, the total amount of flow is distributed *equally* on the feasible number of links (i.e., Nd) in order to bound the congestion. This is not always a good bound since there are fewer constraints. For example, suppose d is much smaller than N and the infeasible

	a	b	c	d	e	f	g	h	$x_{1,\cdots,7}$
a	0	14	13	7	8	5	6	11	1
b	10	0	9	8	12	11	9	7	1
c	7	15	0	15	12	11	7	8	1
d	13	13	11	0	9	7	14	6	1
e	12	12	6	7	0	9	10	10	1
f	13	9	5	6	9	0	12	5	1
g	6	9	11	11	14	15	0	14	1
h	13	9	8	9	11	12	7	0	1
$x_{1,\cdots,7}$	1	1	1	1	1	1	1	1	0
For the sake of simplicity $x_{i,j} = 1$ if $i \neq j$ 0 otherwise									

Figure 2: A Sample (Average) Traffic Matrix

network, obtained from the union of the routes, is a complete graph. Then each (r, j) would only carry the traffic $t_{r,j}$ yielding a loose lower bound on the total flow. The more accurately we can determine the lower bound on the *total flow*, the tighter we can make the lower bound on *congestion*. In the next section we show how to construct (infeasible) networks with good lower bounds on the total flow by using trees with minimum total flow.

3.2 Minimum Flow Trees

Given the triplet (T, N, d), a **minimum flow tree** (MFT_r) is a *d-ary* directed, balanced and weighted spanning tree routed at each commodity r such that the nodes with larger $t_{r,j}$ values are closer to the root r. For example, in Figure 3 some of the minimum flow trees of the traffic matrix of Figure 2 are shown (the trees routed at the nodes x_i ($i = 1, 2, \cdots 7$) are omitted since any breadth first search tree is a minimum flow tree). Intuitively, the objective behind building minimum flow trees is to minimize the propagation of larger traffic values.

Each minimum flow tree defines a weighted shortest path routing (from the root to each node via tree links). Thus, the definition of cost in Observation 1 still applies: $Cost(MFT_r) = \sum_{(i,j)\in MFT_r} level(j) \times t_{r,j}$. For example in Figure 3 the tree MFT_a has cost $127 = 1(14 + 13) + 2(11 + 8 + 7 + 6) + 3(5 + 1 + 1 + 1 + 1 + 1 + 1 + 1)$, and the edge (b, h) has cost $17 = 11$ ($t_{a,h}$) + 6 ($t_{a,f}$) on MFT_a. Since the total cost of an edge (considering all the flow trees) is the total flow carried on this edge to route all the traffic in the network, the edge (b, h) has total cost 28 = 17 (on MFT_a) + 6 (on MFT_d) + 5 (on MFT_f).

We can use these minimum flow trees to derive a lower bound on the total flow associated with a traffic matrix T, and hence on the congestion on any network carrying that traffic. First let us derive a lower bound on the total flow:

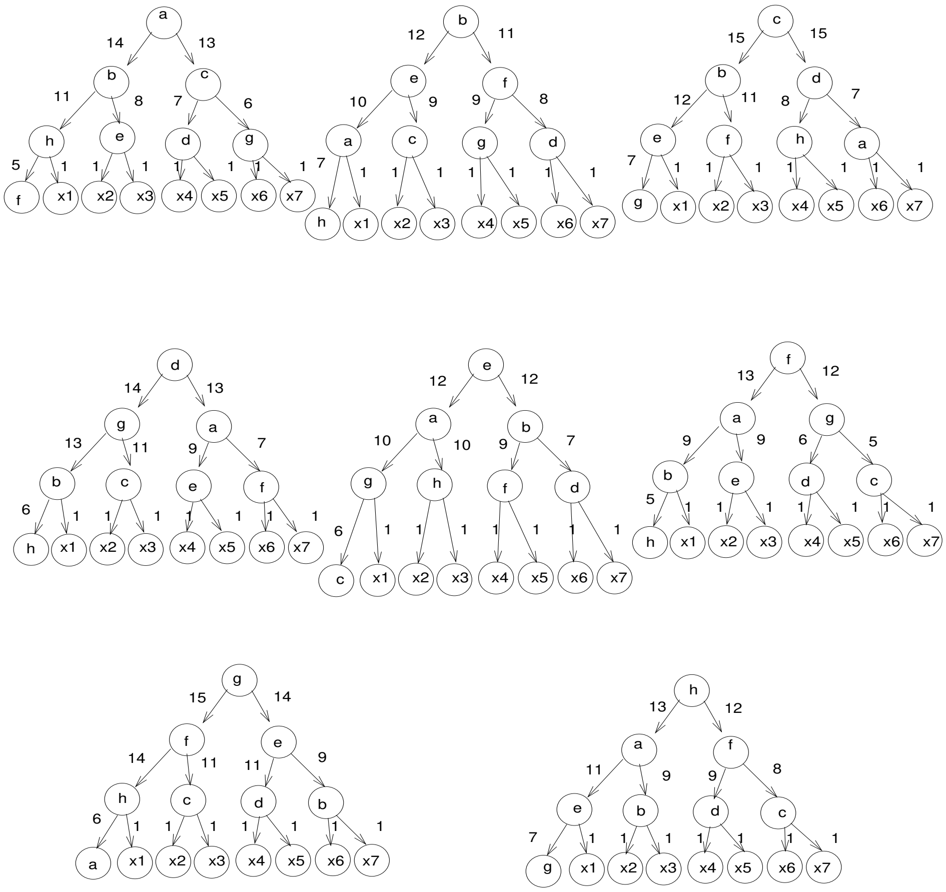

Figure 3: Some Minimum Flow Trees for the sample traffic matrix shown in figure 2. The trees rooted at $x_i, i = 1..7$ are all BFS trees.

Lemma 1 *Given a traffic matrix T the following is true for any feasible flow tree (FFT)*

$$\forall r, \quad Cost(MFT_r) \leq Cost(FFT_r)$$

Proof:

Without lost of generality, we consider commodity r. Let FFT'_r be a feasible flow tree which has minimum total cost (over all possible flow trees rooted at r).

Then it is sufficient to show:

$$\sum_{(i,j)\in MFT_r} level(j) \times t_{r,j} \leq \sum_{(i,j)\in FFT'_r} level(j) \times t_{r,j}$$

We show this by transforming FFT'_r into MFT_r without increasing its total cost (however, the cost may decrease). Since FFT_r' was defined to have minimum cost over all the feasible flow trees, the lemma follows.

Since the traffic values ($t_{r,j}$) are constant in both trees, the cost difference depends on the location of the nodes in the trees. We make the following two observations. First, note that any *permutation* of the nodes at the same level does not change the cost of a flow tree. Second, we note that if two nodes have the same traffic values, *swapping* their locations in a flow tree (FFT_r' or MFT_r) does not change the cost of the tree. Therefore, permutation operations and swapping operations preserve the cost and yield a cost equivalent set of flow trees. Consequently, only the use of these operations during the transformations of the FFT_r' into MFT_r is considered.

We compare (transform) the trees at each level k ($k = 0, 1, \cdots, \log_d N - 1$) of the minimum flow tree. We show by induction on level k that FFT_r' can be transformed to MFT_r without increasing the total flow, unless the promise that FFT_r' is a minimum cost feasible flow tree is contradicted. Let $level(i) \in MFT_r$ denote the level of node i on the tree rooted at r.

Basis: $k = 0$ trivial since only the root node resides in this level in both trees.

Hypothesis: suppose for $k = \log_d N - 2$ both of the trees are cost equivalent. Let j be a node at level $k+1$ of the MFT_r and suppose that its level in FFT_r' is $k\prime$. By induction we know $k\prime \geq k+1$ (note that $k\prime < k+1$ can be omitted since the nodes between these two levels must have the same amount of traffic from the root). If $k\prime$ is equal to $k+1$, then no transform is necessary so we only consider $k\prime > k+1$.

Given that both trees are *d-ary* there are two cases to consider. Case 1: there exists a node i which occupies the space in level $k+1$ of the FFT_r' where node j should be and case 2: the FFT_r' has one node less at level $k+1$ (i.e., the space required by j is vacant).

Case 1: consider the traffic $t_{r,i}$ to node i from the root.

By the induction hypothesis we know that $level(i) \in MFT_r \geq level(j) \in MFT_r$ thus, by definition of the minimum flow tree it follows that $t_{r,i} \leq t_{r,j}$.

Case 1.1: if $t_{r,i} = t_{r,j}$ then *swapping* of i and j preserves the cost, thus the FFT_r' can be transformed to MFT_r and induction holds for this case.

Case 1.2: if $t_{r,i} < t_{r,j}$ then let FFT_r'' be a feasible flow tree obtained by switching the location of only two nodes in FFT_r' namely the nodes i and j. Since the rest of the FFT_r'' and FFT_r' is identical any cost difference must be due to the change of these nodes. By Observation 1 we see the total cost on FFT_r'' is less than the total cost on FFT_r' which is a contradiction to the assumption that FFT_r' has the minimum total cost. Thus, by contradiction, this case of the induction holds.

Case 2 where FFT'_r has a "empty space" on level $k+1$.

This case also leads to a contradiction to the assumption that FFT'_r has the minimum total cost. Since one can decrease the cost of FFT'_r by moving any node closer to the root on the FFT'_r. Thus, at level $k+1$, FFT'_r and MFT_r must be cost equivalent (i.e., both trees are sorted according to the traffic values in an increasing order from root to level one). □

Combining the lemma with Observation 2, it follows that holds for any feasible configuration

$$Cost(\sum_r MTF_r) \leq Cost(\sum_r FG_r)$$

Recall that the minimum flow trees are constructed independently for each commodity so as to minimize the weighted distance from each root. Thus the number of edges in the (infeasible) network—obtained by an edge-union of the minimum flow trees—is at least that of in any feasible network and is often larger. Intuitively, the amount of total traffic on each edge is expected to be less. A lower bound on congestion is given in the following theorem.

Theorem 1 *For any (T, N, d) and routing R we have*

$$LB_0 = \frac{Cost(\sum_r MTF_r)}{Nd} \leq max_{(i,j)}\{z_{i,j}(R,T)\}$$

Proof: It is sufficient that equation of Lemma 1 holds for any feasible flow tree, since once a lower bound on the total flow is computed, the best one can do is to distribute (route) this total flow uniformly over Nd links to minimize the maximum flow on any edge.□

We note that LB_0 is mostly instructive, and will be super-seeded in the next section by a more complex lower bound which uses constrained minimum flow trees. However the bound derived in this section is cheaper to compute and hence may still be of some practical use.

3.3 Constrained Minimum Flow Trees

In the computation of LB_0, we did not enforce any constraint on the selection of the edges. If we knew an edge, say (i,j) occurred in the optimal solution, then we could "constrain" the flow trees by forcing them to "accommodate" this edge. Since the tree is limited to d-ary, the effect of the accommodated edge is to "push" some nodes farther away from the root and hence increase the total cost (i.e., the total flow) which consequently would improve the lower bound on the congestion. We note that to accommodate the edge does not require any flow assignment on this edge, but its place in the flow tree must not be used by another edge.

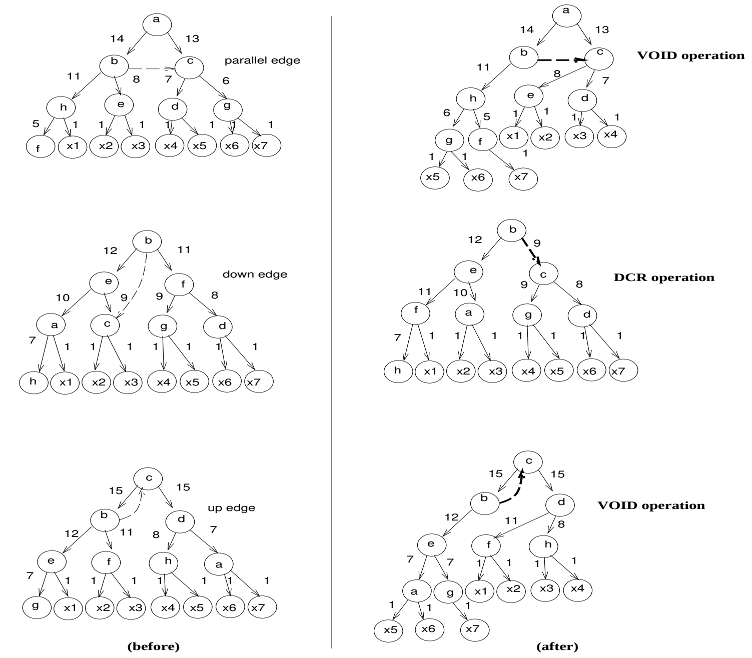

Figure 4: Possible accommodation of edge (b,c) into some Minimum Flow Trees (before and after)

A natural question is "Which edge should be chosen for accommodation in the trees?" It is tempting to use heuristics such as "use the edge with the largest point to point traffic." Unfortunately, the optimal solution may not actually use this edge, especially if there are many edges of nearly the same magnitude. However, some external constraints or priorities require an edge to occur in the solution. Otherwise, each possible edge must be considered as a candidate to be a part of the solution. Thus, we maintain our lower bound property by (i) trying all possible edges (note that there are $N(N-1)$ possible edges), and (ii) "accommodating" the edge on each minimum flow tree with minimum cost increase.

This leads us to the following definition: A **constrained flow tree** $CFT_r^{(i,j)}$ is a flow tree rooted at commodity node r, such that directed edge (i,j) is forced to be accommodated in building $CFT_r^{(i,j)}$ regardless of the $t_{r,j}$ value. If the edge would carry no flow, then it may be logically ignored, but its space in the d-ary tree cannot be filled by another edge.

A **constrained minimum flow tree** $CMFT_r^{(i,j)}$ is a constrained flow tree such that $Cost(CMFT_r^{(i,j)}) - Cost(MFT_r)$ is minimum. For example consider the minimum flow trees in Figure 3 and suppose the edge to be accommodated is (b,c). Some of the constrained minimum flow trees accommodating this edge are shown in Figure 4. The cost of the tree rooted at node a is $Cost(CMFT_a^{(b,c)}) = 1(14+13)+2(11+8+7)+3(6+5+1+1+1+1)+4(1+1+1) = 136$, which is higher than the cost of MFT_a. Therefore, we can obtain a lower bound on the total flow by considering all possible edges (i.e., all pairs of (i,j) for $i \neq j$), and building the associated constrained minimum flow trees.

Theorem 2 $min_{ij}\{\sum_r cost(CMFT_r^{(i,j)})\} \leq Cost(\sum_r FG_r)$

Proof:
From Lemma 1 and the definition of a constrained minimum flow tree it follows that $\sum_r CMFT_r^{(i,j)}$ is a lower bound on the total flow, if we know edge (i,j) is in the solution. We note that the final graph must have at least one edge and since we take the minimum overall (i,j), the theorem holds.□

Let (u,v) be the edge for which the total flow (computed over all the constrained flow trees) is minimum (over all possible edges), then we can obtain a better lower bound (LB_1) than LB_0 by distributing this total flow over Nd edges. Thus given an instance (T,N,d) of the problem

Theorem 3 *For any routing R of T*
$LB_1 = \frac{min_{ij}\{\sum_r cost(CMFT_r^{(i,j)})\}}{Nd} \geq LB_0$

Proof:
For each possible edge (i,j) ($\forall i \neq j$) construct a hypothetical graph $G_1^{(i,j)}$ which is obtained, over all commodities, by the union of the edges of the $CMFT_r^{(i,j)}$ trees. The total flow carried on this graph is the sum of the costs (flow) on its edges as before. Let $G_1^{(u,v)}$ be the graph induced by accommodating the edge (u,v) such that the total flow on $G_1^{(u,v)}$ is minimum over all hypothetical graphs $G_1^{(i,j)}$ that is

$$Cost(G_1^{(u,v)}) = min_{(i,j)}\{\sum_r cost(CMFT_r^{i,j})\}$$

In other words, flow carried on this graph is the least, among all networks required for routing of the traffic T with the given edge constraint. As in the computation of LB_0, we distribute the total flow on this graph equally over only the feasible number of the edges obtaining the lower bound LB_1.

Due to the definition of a minimum constrained flow tree

$\sum_r cost(MFT_r) \leq min_{ij}\{\sum_r cost(CMFT_r^{(i,j)})\}$ thus $LB_1 \geq LB_0$.□

4 An Algorithm to Compute the Lower Bounds

In this section we consider the computational complexity of finding the lower bounds LB_0 and LB_1. Since our input is the traffic matrix with $M = N^2$ entries (where N is the number nodes), we express our bounds in terms of M.

Computing LB_0 is quite straightforward: first we sort each row of the traffic matrix in time $O(M \log M)$ and then build a d-ary tree for each row. Therefore the total cost for computing LB_0 is $O(M \log M)$.

However, efficient computation of LB_1 is much less trivial and achieved by a tree maintenance algorithm as described below. Construction of the constrained minimum flow trees is based on defining a set of *accommodation* operations defined on the minimum flow trees. The $CMFT_r^{(i,j)}$ can be obtained from MFT_r by applying exactly one of the insertion operations. The accommodation operations are chosen in order to minimize the cost increase on the minimum flow trees.

Consider a directed edge (i, j) forced to be accommodated in the all the minimum flow trees. Depending on level(i) and level(j) in the MFT_r, the edge, (i, j) under consideration can be classified as of the three types:

1. If $level(i) < level(j)$ then (i, j) is a *down edge*,
2. If $level(i) > level(j)$ then (i, j) is an *up edge*,
3. If $level(i) = level(j)$ then (i, j) is a *parallel edge*.

Given a MFT_r, reserving space for edge (i, j) into a minimum flow tree is based on exactly one of the five operations shown in Figure 5.

Theorem 4 *Given T, N and d, algorithm compute_LB_1(T,N,d) constructs minimum flow trees and computes the lower bound LB_1.*

Proof: Termination of the algorithm is ensured since there are nested two loops with total $O(N^3)$ iterations.

The correctness of the algorithm is proven by showing that two invariants of the algorithm—*tree-property* and *cost-property*—are maintained after each accommodation operation of Figure 5. The tree-property requires that after each operation performed on a minimum flow tree

1. the same node remain as the root with degree exactly d,
2. there is a path from the root node to all the others,
3. degree of each node is at most d
4. no cycles exist

Given an edge (i, j) to be accommodated on the MFT_r, each operation checks if the root r is the same as j (except for DCR and VOID operations for which $r \neq j$ by definition of these operations). There is no cycle since VOID operations do not establish the edge (i, j), and the rest of the operations do not result multiple parents (i.e., no operation establishes any parallel or up edges). Given that there is no cycle, it is ensured that there is a path from the root to all others, since each node (except r) has a father. Since MFT_r is a *d-ary*, the degree of the non-leaf nodes is at most d after VOID operations, and remains the same as before for all other operations.

Therefore, performing an accommodation operation on a minimum flow tree maintains a flow tree (i.e., constrained flow tree) on which traffic from the root flows down to all the nodes.

To show that the constrained flow tree has minimum cost we prove that the cost increase on the underlying minimum flow tree is minimal after the accommodation operation (i.e., cost-property is maintained).

Let (i, j) be an edge to be accommodated in all the minimum flow trees. The set of accommodation operations (shown in Figure 5) is complete, since no other operation is possible without violating the tree-property. Note that each accommodation operation of the algorithm preserves a lower and/or upper *cost equivalent components* with the minimum flow tree MFT_r (see Figure 6). That is the the set of the edges in MFT_r such that their cost remains the same as before after the accommodation operation. Note that any other tree must also have *at least* the same cost in the *equivalent components* with the MFT_r (since the MFT_r is a sorted d-ary tree) in order to ensure *at most* the same cost with $CMFT_r^{(i,j)}$. Consequently we can limit our argument to the segment of the minimum flow tree that is modified by the accommodation operation.

In the modified segment, some nodes are pushed away from the root (i.e., their levels are incremented by one) and some nodes are moved closer to the root. The number of UM (upward migrating) and DM (downward migrating) nodes are determined directly by the chosen accommodation operation. The UM set contains the nodes with largest traffic to be moved up in the tree (i.e., the cost decrease is maximized). On the other hand the DM set contains the nodes with the smallest amount of traffic at each level involved in the insertion operation. Thus, the cost increase is minimized by carrying the minimal amount of traffic away from the root

The algorithm chooses the feasible operation for which the net cost increase is minimal among all the feasible operations to accommodate the edge (i, j) (i.e., the constrained flow tree rooted at r is the $CMFT_r^{(i,j)}$).

At step 4. of the algorithm an edge (among all possible edges) for which the sum of the flow on all the constrained minimum flow trees (accommodating this edge) is chosen. Thus according to the Theorem 3, the output of the algorithm is the lower bound LB_1. □

Time Complexity: It requires $O(\log N)$ time to find the nodes of interest in a MFT and the remaining operations depend on the amount of the tree affected which, in the worse case, is $O(N)$. Since we must do this for all M possible pairings edges and N nodes, we end up with a $O(M^2)$ algorithm for computing lower bound LB_1. Since the bound only needs to be computed once this computable lower bound can be considered a reasonable component of an upper bound algorithm, e.g., it can be used in a termination criterion or to evaluate heuristic quality.

5 Summary and Conclusions

In this paper we presented a useful tool to measure the performance of heuristics in the design of multihop lightwave networks with minimum congestion routing. Given an instance of the problem the lower bounds can be computed in polynomial time in the network size. Thus the tool can be integrated into the heuristics. Indeed, it was applied in [4, 5] for the analysis of heuristic algorithms for allocation/routing in WDM networks.

Particularly, we derived 3 different lower bounds, a "immediate one" LB_I which ignored flow, and two flow tree based ones LB_0, LB_1. We also showed that $LB_0 \leq LB_1$. However we note that it is possible to determine the cases where $LB_0 = LB_1$. For example if there is an edge which occurs on all the MFT, or if i is a leaf, or if i and father of j are at the same level then the equality holds. Thus it is possible to *test* these conditions and avoid performing expensive computation for LB_1.

Next, let us consider the relation between LB_1 and LB_I. Unfortunately $LB_1 \geq LB_I$ is not always true. For instance consider a traffic matrix in which one or two nodes have much larger incoming and outgoing number of messages than the rest. In this case the flow tree approach does not perform well since one or two CMFT trees will have much higher cost than the others and their cost can be distributed over the other edges. Therefore, in practice, we compare LB_1 and LB_I and take the maximum:

$$LB = MAX(LB_1, LB_I).$$

We note that the constrained flow tree technique could be extended to force the flow trees to include 2, 3 or any number of edges. The trade-off is that doing so would require computing the CMFT over all pairs, all triples, etc.

Although we assumed that each node had fix degree d it is straightforward to generalize it to varying degree. Furthermore, if some of the nodes have in/out degree of one this information can be directly used in the constrained flow tree technique. Similarly, if other constraints allow one to determine an edge which *must* occur in the solution, the CMFT approach can use this information to further increase the lower bound.

Acknowledgement

The first author would like to thank D. Bienstock and J.F. Labourdette for their valuable comments on the earlier versions of this work.

References

[1] A.Bannister, L.Fratta, and M.Gerla. Topological design of the wavelength-division optical network. *Proc. IEEE INFOCOM'90*, June 1990.

[2] A.S. Acampora and M.J.Karol. An overview of lightwave packet networks. *IEEE Network Magazine*, 3:29–41, January 1989.

[3] S. Banerjee, B. Mukherjee, and D. Sarkar. Heuristic algorithms for constructing near-optimal structures of linear multihop lightwave networks. *IEEE INFOCOM'92*, pages 671–680, 1991.

[4] B.Yener and T. Boult. Logical embeddings for minimum congestion routing in lightwave networks. *Technical Report CUCS-008-93, Computer Science Dept., Columbia University*, 1993.

[5] B.Yener and T. Boult. A study of upper and lower bounds of minimum congestion routing in lightwave networks. *Proc. IEEE INFOCOM'94*, 1994.

[6] C.A.Brackett. Dense wavelength division multiplexing networks: Principles and applications. *IEEE J. Sel. A.r in Commun.*, 8, August 1990.

[7] I.Chlamtac, A.Ganz, and G.Karmi. Transport optimization in broadband networks. *Proc. IEEE INFOCOM'91*, April 1991.

[8] IEEE. *Jour. Sel. Areas in Commun., Special Issue on Dense WDM Tech. for High Capacity and Multiple Access Communication Systems*, 8(6), August 1990.

[9] J.P.Labourdette and A.S.Acampora. Logically rearrangeable multihop lightwave networks. *IEEE Trans. Commun.*, 39, August 1991.

[10] M.R.Garey and D.S.Johnson. *Computers and Intractability - A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

Existing edge: NOP

If the edge already exists in this tree then nothing is needed to accommodate it.

Down edge with Parent Replacement: DPR

Let v be the father of j. If $i = r$ then this operation is not defined. Otherwise, define the set UM of *upward migrating nodes* as the leftmost nodes of the MFT_r between the level of i and v, (i.e., $|UM| = k = level(v) - level(i)$).

Operation: 1. replace the parent of j with i: $level(i) = level(v)$ 2. move up: $level(u) = level(u) - 1$, $\forall u \in UM$.

Cost of DPR: $T(DPR) = k \times t_{r,i} - \sum_{u \in UM} t_{r,u}$. That is the difference between the cost increase caused by moving i down thus carrying the flow $t_{r,i}$ k times more and the flow decrease caused my shortening the distance of the nodes in UM from the commodity node r.

Down edge with Child Replacement: DCR

Let v be a child of i. Define the set DM of *downward migrating nodes* as the rightmost nodes of the MFT_r between the level of j and v, (i.e., $|DM| = k = level(j) - level(v)$).

Operation: 1. replace the child by bringing j up at the same level with v: $level(j) = level(v)$ 2. move down: $level(u) = level(u) + 1$, $\forall u \in DM$.

Cost of DCR: $T(DCR) = \sum_{u \in DM} -(k \times t_{r,j})$

Parallel/Up edge with Void Replacement: VOID

Let m be the depth of the MFT_r and let $n = m - level(i) + 1$. Note that n is the height of the subtree rooted at j. Define the set DM of *downward migrating nodes* as the set of rightmost 2^k nodes at $level(i) + 1 + k$ where $k = 0, 1, \cdots, n-1$.

Operation: 1. $level(u) = level(u) + 1$ $\forall u \in DM$.

Cost of VOID: $T(VOID) = \sum_{u \in DM} t_{r,u}$

Parallel/Up edge with Parent Replacement: PUPR

Let v be the father of j. If $v = r$ or $j = r$, this operation is not defined else define the set DM of k *downward migrating nodes* as the set of rightmost nodes whose level is in between $level(i), \cdots, level(v)$, (k = level(i)- level(v)).

Operation: 1. bring i to the same level as the parent of j: $level(i) = level(v)$, 2. move the rightmost ones one level down $level(u) = level(u) + 1$, $\forall u \in DM$.

Cost of PUPR: $T(PUPR) = \sum_{u \in DM} t_{r,u} - (k \times t_{r,i})$

Parallel/Up edge with Child Replacement: PUCR

Let v be a child of i. If $j = r$ then it is not defined else define the set UM of k *upward migrating nodes* as the set of leftmost nodes whose level is in between $level(j), \cdots, level(v)$.

Operation: 1. $level(j) = level(v)$, 2. $level(u) = level(u) - 1$, $\forall u \in UM$.

Cost of PUCR: $T(PUCR) = k \times t_{r,j} - \sum_{u \in UM} t_{r,u}$

Figure 5: Set of possible operations defined on minimum flow trees to obtain constrained minimum flow trees from accommodating edge (i, j).

Algorithm Compute_$LB_1(T, N, d)$
input: Traffic matrix T, degree d, of N nodes
output: LB_1
begin
preprocessing: for all commodity r, build MFT_r
    begin: Do for all possible pairs (i, j)
        begin: Do for all commodity r
0.   if the edge is on the MFT_r skip the rest and consider next commodity
1.   determine the *direction* of the edge
2.   *If* it is a *Down* edge then compute and choose $min\{T(DPR), T(DCR)\}$
3.   *Else* compute and choose $min\{T(PUPR), T(PUCR), T(VOID)\}$.
        end
    end
4. Determine the edge (u, v) s.t $\sum_r Cost(CMFT_r^{u,v})$ is minimal over all edges
5. $LB_1 = \frac{\sum_r Cost(CMFT_r^{u,v})}{dN}$
end

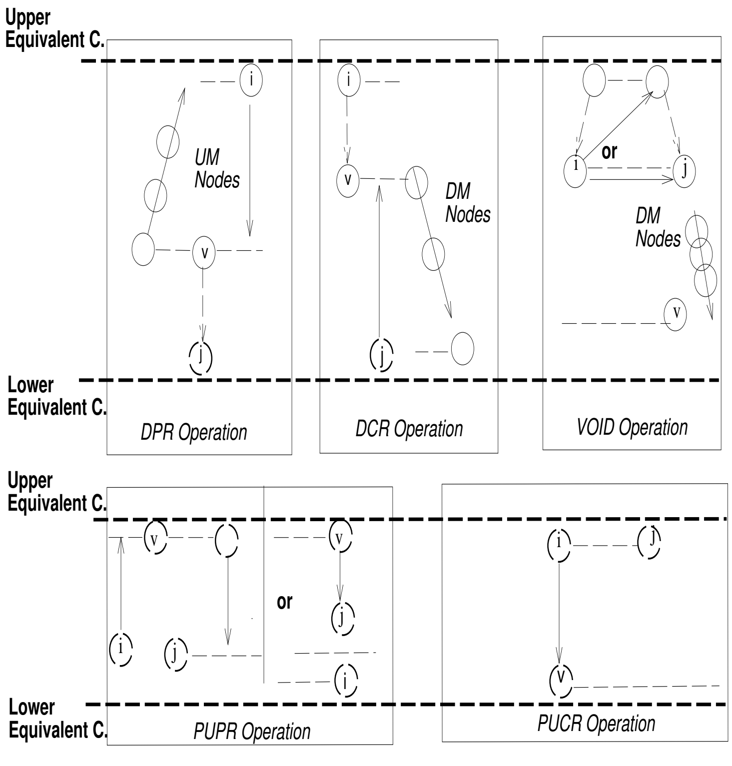

Figure 6: Edge Accommodation Operations