Logical Embeddings for Minimum Congestion Routing in Lightwave Networks

Bülent Yener and Terrance E. Boult

Columbia University Department of Computer Science, NYC, NY 10027.

Technical Report CUCS-008-93

Abstract

The problem considered in this paper is motivated by the independence between logical and physical topology in Wavelength Division Multiplexing (WDM) based local and metropolitan lightwave networks.

This paper suggests logical embeddings of digraphs into multihop lightwave networks to maximize the throughput under nonuniform traffic conditions.

Defining *congestion* as the maximum flow carried on any link, two perturbation heuristics are presented to find a *good logical* embedding on which the routing problem is solved with minimum congestion.

A constructive proof for a lower bound of the problem is given, and obtaining an optimal solution for integral routing is shown to be NP-Complete.

The performance of the heuristics is empirically analyzed on various traffic models. Simulation results show that our heuristics perform, on the average, 20% from a computed lower bound. Since this lower bound is not quite tight, we suspect that the actual performance is better. In addition, we show that $5\% - 20\%$ performance improvements can be obtained over the previous work.

1 Introduction

The vast optical bandwidth of a fiber, confronted with the electro-optic bottleneck, motivates the effort on employing concurrency and parallelism in optical networks, as well as new architectures.

The *Wavelength Division Multiplexing* technique [1, 2, 9, 17, 18], realized by tunable optical transceivers, has been proposed as a way of constructing multichannel, multihop lightwave networks. A station in such a network has electronic and optical components. The optical components are transmitters and receivers used to tap into the optical medium by tuning to a special wavelength, whereas the electronic components constitute the rest to control the operation of the switch.

We consider networks whose underlying physical topology is based on sharing a single fiber (e.g., a linear bus, a tree or a star topology). In this case, potentially all the stations can reach the medium by tapping into it.

Pairing a transmitter on one node with a receiver on another one (by tuning them to the same wavelength) establishes a logical connection between these two stations. Therefore, allocation of the wavelengths to the stations constructs a logical embedding into the network, independent from the physical topology. Although, sharing is possible, we consider unique end-to-end assignments of the wavelengths (no channel sharing). The logical embeddings are constrained by some design parameters, such as the number of available transceivers or wavelengths. We refer to [5], [6] for a recent survey on WDM based architectures.

In this paper, we propose logical embeddings of directed graphs called *configurations* for achieving high throughput under non-uniform traffic conditions. Candidate configurations are generated by *perturbation heuristics* and the routing problem is solved on these configurations by Linear Programming. The routing problem is formulated as a Multi Commodity Flow (MCF) problem with the objective function to minimize the maximum flow (congestion) on an edge.

This paper organized as follows: In the following section, we define the problem and state our approach more formally. In section 3, the algorithmic issues involved with the perturbation process are addressed. In section 4 and 5, we present our algorithms and their computational performance analysis, respectively. A constructive proof for a lower bound of the problem by using special spanning trees and some complexity results on the problem are presented in sections 6, 7 respectively.

2 Problem Definition and Approach

Given a traffic matrix $T = [t_{s,t}]$ for N stations, such that an entry $t_{s,t} \in \Re^+$ is the amount of average traffic to be sent from a source station s to a destination t, let integer τ_i denote the number of incoming and outgoing links for each station i.

Define a configuration CONF as a strongly connected digraph without self-loops such that each node has in/out degree exactly d. A configuration is a logical representation of a network topology in which each node corresponds to a station and each edge to a logical/physical connection.

Given a traffic matrix T and a configuration CONF, define routing R as an assignment of the traffic $t_{s,t}$ to directed path(s) on CONF, from source s to sink t, for all s, t pair.

Define *congestion* $z_{i,j}$ on a directed edge $(i, j) \in CONF$ as the total amount of traffic carried on this edge for routing the traffic in T. Let $Z = max_{(i,j)}\{z_{i,j}\}$

For a given T and d, we are interested in finding a configuration CONF and a routing R to minimize Z while satisfying the traffic T.

Note that this problem has two parts:

1. how to find a good embedding,

2. how to find a good routing algorithm

A unified solution can be obtained by solving the following mixed integer programming problem by defining a unique commodity for a station k such that $\sum_j t_{k,j} > 0$. Denote by $f^k_{(u,v)}$ the flow on edge (u,v) of commodity k. Then, *Min* Z such that

1. $Z \geq \sum_k f^k_{(u,v)}$, $\forall k$, $\forall pair(u,v)$

2. $\sum_i f^k_{(i,j)} - \sum_i f^k_{(j,i)} = t_{k,j}$ $\forall k, j$ where $k \neq j$.

3. $0 \leq f^k_{(u,v)} \leq (\sum_j t_{k,j}) x_{u,v}$

4. $\sum_j x_{i,j} = \sum_j x_{j,i} = \tau_i$, $\forall i, j$

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

6. $0 \leq f^i_{(u,v)}$

There are two variables of interest: the 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 show in section 7 that for an integral traffic matrix T, finding a configuration CONF and an integral routing of the traffic with minimum congestion on this CONF is NP-complete. The conjecture is that the problem is still NP hard, with the integrality relaxation on the routing problem, (due to remaining integrality constraint on the edge variables).

Therefore, we introduce two *perturbation heuristics* [33] for approximate solutions based on *Variable Depth Local Search* [20], and *Simulated Annealing* techniques [19], respectively. Each heuristic performs a greedy search in an exponentially large configuration space (the set of all CONFs). The direction of the search is guided by the value of a cost function, which is the amount of maximum congestion. Value of congestion is computed by solving the routing problem, formulated as a *multicommodity flow* problem, with Linear Programming (LP) [1] on the current configuration.

---

[1] **The Subroutine-LP:** Given a configuration CONF, the traffic matrix T and the degree bound d, we formulate the problem of finding minimum congestion routing as an instance of *Multicommodity flow problem* (MCF). In this formulation, each commodity corresponds to a source node leading to polynominal number of rows and columns in the Simplex Tableau. Precisely, define as a commodity a station k such that $\sum_j t_{k,j} > 0$. Denote by $f^k_{(u,v)}$ the flow on edge (u,v) of commodity k. Then, *Min* Z such that

1. $Z \geq \sum_k f^k_{(u,v)}$, $\forall$ commodity k and $\forall$ edge (u,v)

2. $\sum_u f^k_{(u,v)} - \sum_u f^k_{(v,u)} = t_{k,v}$, $\forall k, v$, where $k \neq v$, $f_{(u,v)} \geq 0$.

The output of the LP solution is the minimized maximum congestion Z and the flow assignment $f^k_{(u,v)}$ for all the edges (u,v) and commodities k.

Although, we relaxed the integrality constraint for the routing problem, note here that if the entries of the traffic matrix are large real numbers, a *good* rounding of the flow [31],[30] to obtain an integer solution to routing problem does not substantially change the value of the congestion. Therefore, LP solution approximates to the optimal integral routing as the traffic matrix consists of larger numbers.

Starting with an initial configuration, the heuristics perform the following three basic steps.

1. Ask Subroutine-LP for the value of maximum congestion for the current configuration.
2. Determine the direction of the search.
3. Modify the current configuration to obtain a new one.

Termination of the heuristics can be realized either externally by an input parameter, or internally when no more decrease is achieved on the value of the cost function.

Choice of a starting configuration is important for finding a local optimum solution. Starting points can be generated randomly as well as with greedy algorithms. We give an empirical comparison of various starting points in section 5.

A candidate configuration, for which the routing problem can be solved with less congestion, is accepted and the search of the configuration space continues from that point. An accepted CONF is modified to obtain a new candidate which is called *valid*, if it maintains the CONF property. Maintenance of CONF property is realized, with a validity test prior to the perturbation operation which determines whether or not the CONF property would be destroyed after the operation.

The modification of the current configuration is achieved by perturbation operation which is based on a *n-change* operation. A set of edges (with cardinality n) is *randomly* chosen to be replaced by a new set of edges (with the same cardinality) to obtain a new configuration. In other words, one of the configurations that can be obtained from the current one by n-changes, is chosen randomly.

Choice of perturbation determines the direction and the step size of the search. For instance, big changes cause big jumps in the search space, therefore if the space has narrow valleys a nearby local optimum could be missed. On the other hand, small step size may require longer search process to find a local optimum.

In the following section, we introduce two different perturbation operations and then compare their performance in section 5.

3 Perturbation and Maintenance of CONF property

In this section we present two perturbation operations: *node* and *edge* perturbation and show how to test the *validity* in advance.

Node perturbation case (4-change): Two distinct nodes are randomly chosen among N nodes and their *out-going* edges are exchanged as follows:

1. Choose randomly with probability $\frac{1}{N} \times \frac{1}{N-1}$ a distinct pair of nodes u, v from the configuration see figure 1 2.

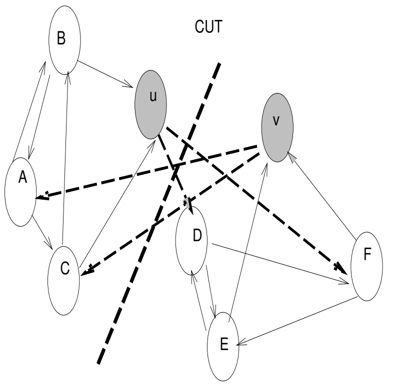

Figure 1: Before Node perturbation

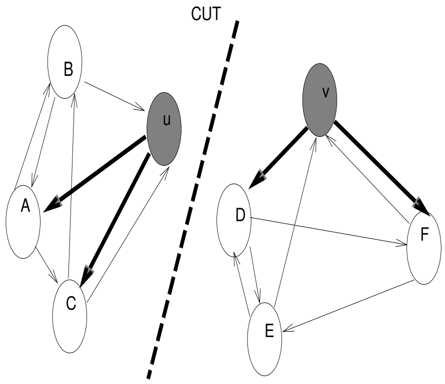

Figure 2: After Node perturbation (pair u,v is an invalid one)

2. If u, v is a *valid* pair then swap the out-going edges of u and v (see figure 2).

3. Else keep u and choose a new v randomly and go to step 2.

A pair of rows is a *valid* one, if the swap of their out-going edges does not destroy the CONF property of the underlying graph. Note that the degree requirement is always satisfied, thus we only need to ensure the strong connectivity and no self-loop requirement of a configuration on the perturbed graph.

If the connectivity property is destroyed, then the routing problem will have an infeasible solution, assuming $t_{i,j} \neq 0, \forall i, j$. Thus, under heavy traffic conditions, a feasible LP solution ensures the property. However, using LP to check this property is computationally expensive and it is not suitable for sparse traffic matrices. Thus, we introduce a validity test to determine whether or not a pair of nodes is *valid* before the routing problem is attempted to be solved. The idea is based on building directed spanning trees T_u, T_v rooted at the pivot nodes u and v without including any of the outgoing edges the other pivot node, respectively.

Procedure Node-Test(u,v)

1. If: the edge (u, v) or (v, u) exists then stop and print *invalid*

2. Else: rooted at u and v start building 2 spanning trees T_u, T_v

   such that v and u must be a leaf in the trees, respectively

3. if T_u and T_v can be build then u, v is a *valid* pair for node perturbation

4. else u, v is an *invalid* pair.

Let us analyze the correctness of this algorithm. Given a CONF, if there is an edge between u and v then due to the perturbation a self loop will be established, thus the CONF property will be destroyed.

Let's consider only the pivot node u without lost of generality. Let T'_u be the subtree rooted at u which contains the nodes reachable from u without visiting the node v. If $T'_u + v = T_u$ (i.e., induces a spanning tree) then perturbation will swap the root u with v. Thus, all the nodes will be reachable from v after the perturbation. (Similarly, a perturbation on the spanning tree rooted at v will result u to be the new root on T_v). Since the incoming edges of the pivot nodes remain the same they will be reachable from all nodes thus preserve the CONF property.

For sufficiency, we show that these trees can not be constructed only if the perturbed graph is not strongly connected. Suppose that T_u and T_v does not induce a spanning tree on the original graph and the perturbed graph is strongly connected. Then there exists cuts $N_u = (T'_u, T_u \setminus T'_u)$ and $N_v = (T'_v, T_v \setminus T'_v)$ on the original graph which contains the outgoing edges of v, and u, respectively (since T_u and T_v are not spanning trees). Furthermore, since neither of these trees can be built, and CONF is *d_regular* then, there is a cut $N_{u,v} = (T'_u, T'_v)$ which must contain only the outgoing edges of u and v. Thus removing the edges from this cut (the perturbation step) must disconnect the original graph to result an invalid perturbed graph, leading to a contradiction. (For example in the figure 1, $T'_u = \{u, d, f, e\}$ and $T'_v = \{v, a, c, b\}$.)

An invalid pair is rejected and another random selection is made. Since one of the pivot nodes is kept fixed a valid pair can be found with linear number of tries.

Edge perturbation case (2-change): An edge is chosen randomly among dN edges and replaced by a new one as follows (see Figure 3).

1. choose an edge (u, v) to be replaced with probability $\frac{1}{dN}$
2. choose a new edge $(u, w) \notin CONF$ between u and a randomly chosen station $w \neq v$ with probability $\frac{1}{N-2}$
3. chose one of the d incoming edges of w with probability $\frac{1}{d}$. Say (x, w).
4. If edges (u, w) and (x, v) are *valid* ones then establish edges (u, w) and (x, v) and delete edges (u, v) and (x, w) see figure 4.
5. Else go to 2.

A similar test is given below to determine the effect of an edge perturbation operation in advance. Let (u, v) be the edge chosen to be replaced and let w, x be the other two nodes involved into this replacement. However, this time we build the spanning trees T_w and T_v, rooted at w and v, respectively and require that u in T_w and x in T_v to be a leaf. If both of the trees can be build then the replacement of edges (u, v) and (x, w) with (u, w) and (x, v) is a valid one.

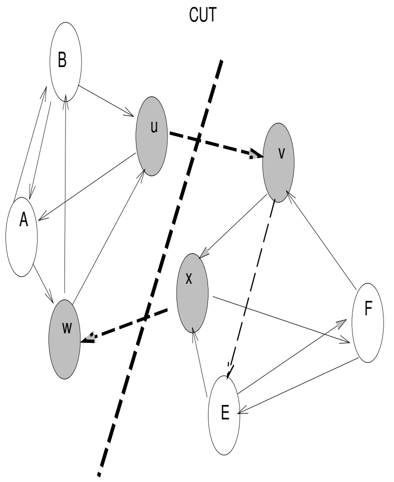

Figure 3: Before Edge perturbation

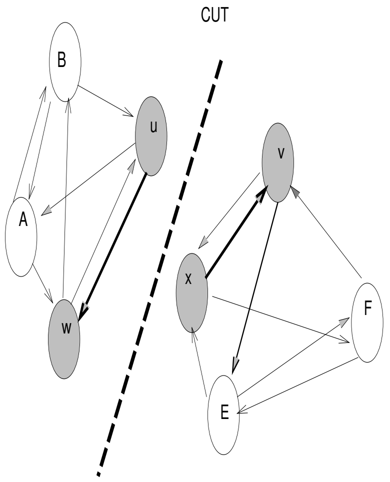

Figure 4: After Edge perturbation (invalid choice)

4 Perturbation Heuristics

In this section, we present two heuristics based on randomized perturbation of the current configuration. The first heuristic (STO-Based) has a stochastic decision process, the second one (VDS-Based) has backtracking to avoid the local minima problem.

4.1 STO-Based Heuristics

This approach is based on simulated annealing, in which the direction of the search and generation of a new candidate configuration is realized by probabilistic processes.

The algorithm, detailed in figure 5, takes a traffic matrix T of N stations, a degree constraint r, and a control scheme C for the search process, and outputs the best configuration it finds.

The input control scheme C contains two parameters to control the search process: the *Length* and the *Depth*. The *Length* parameter is used to control the length of the search, whereas the *Depth* variable determines the diameter of the local search at a specific length. The control scheme determines the performance of this heuristic with a trade off between increasing time complexity and decreasing congestion. As explained in Section 5, we studied the convergence of the algorithms by plotting the minimum congestion as a function iteration, to learn about a good control scheme. In step 2 of the algorithm, the value of the congestion on the candidate configuration is compared to the congestion for the last accepted solution. If the candidate solution has less cost (congestion) then it becomes the last accepted one. However, to avoid the local minima problem, a bad configuration may be accepted with a probability $e^{\Delta Z/L'}$, where ΔZ is the congestion difference between the current and candidate configurations at length L'.

In Step 3, one of the two modes of perturbation is performed to modify the current configuration to obtain a new candidate. Steps 4,5 are controlled by the control scheme C.

Although we control the depth and the length of the search process externally, it can also be done internally by monitoring the change at the value of the cost function. The search process would be terminated when a local minima is found (i.e. the decrease of the cost is zero).

Algorithm *STO-Based Heuristic* (T, r, C)
begin
  Start with an arbitrary configuration Candidate-CONF
  Set *Min-CONF*,*Last-CONF* to *Candidate-CONF*
  begin length: Do L' = 1 to Length L
    begin depth: Do D' = 1 to a Depth $D(L')$, at each length L'
1. *Route the traffic on the Candidate-CONF*
   formulate the problem as a MCF problem and
   ask Subroutine-LP to compute the maximum congestion
2. *Make a decision for the direction of the search*
   If $Z_{Candidate-CONF} \leq Z_{Last-CONF}$
   Or with probability $e^{\Delta Z/L'}$ Then
   Set the *Last-CONF* to *Candidate-CONF*
   If $Z_{Candidate-CONF} \leq Z_{Min-CONF}$ Then
   Set the *Min-CONF* to *Candidate-CONF*
3. *Generate a new candidate configuration*
   If pivot is an *edge* then do *Edge-Perturbation*
   Else do *Node-Perturbation*
4. $D' \leftarrow D' + 1$
    end
5. $L' \leftarrow L' + 1$
  end
end

Figure 5: STO-Based Heuristic algorithm

Note here that, although we use the Metropolis Criteria as in the SA technique for avoiding the local minima problem, our approach is different since we maintain an overall best configuration during the execution of the algorithm. Thus, unlike SA, our algorithm does not forget the past -a good solution encountered in the earlier steps of an execution of the algorithm will not be lost.

Algorithm *VDS-Based Heuristic* $(N, T, r, l\ )$
begin
Start with an arbitrary configuration Candidate-CONF
Set the *Min-CONF* to *Candidate-CONF*
1. **begin** Do for a Length l of the search
1.1 $k \leftarrow DEPTH$
begin Do for a finite Depth k
2. *Route the traffic on the Candidate-CONF*
formulate the problem as a MCF problem and
get the cost of this configuration
3. *Make a decision for the direction of the search*
3.1 *If* $Z_{Candidate-CONF} \leq Z_{Min-CONF}$ Then
Set the *Min-CONF* to *Candidate-CONF*
3.2 *Else if* $k < l$ Increment the depth $k \leftarrow k + 1$
4. *Generate a new candidate configuration*
If pivot is an *edge* Then do *Edge-Perturbation*
Else do *Node-Perturbation*
provided that new edge(s) was not replaced previously
end
5. Set the *Candidate-CONF* to *Min-CONF*
end
end

Figure 6: VDS-based heuristic algorithm

4.2 VDS-Based Heuristics

In this class of heuristics, we modify a local search technique called the *Variable Depth Local Search* to find a local optimum solution which has a high probability of also being the global optimum. Suggested by Kerninghan and Lin [20], the technique is based on the idea of replacing the search for one modification by a search for a sequence of modifications, using a cost function to direct the search. Our algorithm modifies this technique by enhancing it with backtracking and maintaining an overall best solution.

Define a *neighborhood* $N_k(CONF_i)$ as all the possible configurations that can be obtained from the last configuration $CONF_i$ by k operations, for a *good* k called the *depth* of the search.

The algorithm, presented in figure 6, inputs a traffic matrix T of N stations, a degree constraint r and length l, and outputs the best configuration it finds. Depth k is the diameter of the neighborhood for which the decrease on the congestion is maximized by the search process. The length l is the number of the steps for which no more decrease in the cost function can be obtained. We assume that $l > k$, and input the length l in our algorithm (however, it is possible to control the length dynamically by examining the change in the cost function).

We have a conjecture that the diameter (longest-shortest-path) of a configuration has a bound as a function of N and r. For instance if $r = 1$ then logical network is a circuit with a diameter

$N-1$ and for $r=2$ the diameter can not exceed $N/2$. Therefore, the maximum number of hops on the routing is also bounded. Although this conjecture needs to be proven for arbitrary N and r, we implemented the above heuristic by setting the constant $DEPTH = N/2$ for $r=2$.

The depth variable k is set to the constant DEPTH for each length of the search (step 1.1). However, note that the depth of the search is incremented each time upon backtracking to the same configuration (step 3.3) during the same length. Once an edge is removed from a configuration it can not re-enter during the same length $l' \leq l$ of the search process. At the start of each length l' all the edges that are not in the configuration become again candidates to be inserted.

During the execution of the algorithm an overall best solution is maintained (step 3.2) and at each length, a *backtracking* to the overall best solution found so far is performed (step 5.). We note also that backtracking does not cause the *cycling* (i.e. getting stuck by keep on backtracking to the same point without exhausting the other paths) , since the edges are chosen without replacement and the depth is incremented. The local search process in this heuristic can also be enriched by defining a set of *must-in* and *must-out* edges to limit the number of operations.

As a final remark, the <u>initial solution</u> and definition of <u>neighborhood</u> are important in the performance of this heuristic. A good neighborhood can relax the requirement for a good initial solution. On the other hand, using many different randomized initial configurations increases the performance of the algorithm as verified by the simulation results given in the next section.

5 Computational Results

Our simulation efforts [2] focus on two issues. The first is to analyze and compare the performance of our heuristics. The second is to learn about the nature of this problem by using some statistical and algebraic tools. We used the same illustrative traffic matrices (the matrices are scaled up to have integrality, see Appendix) of Labourdette and Acampora [25],[26] for the networks of 8 nodes and in- and out- degree 2, as well as 10 randomly generated nonuniform traffic models to test and compare our algorithms. For each traffic matrix, we determined randomly 29 different starting points as well as a "greedy" one, to initiate our algorithms.

In addition, we implemented an algorithm to compute a lower bound of the problem to compare our results. A summary of the comparative results is given in Table: 1.

	Previous Results (JFL&AA91)		Perturbation Heuristics							
			Greedy Start				Random Start			
			*STO-Based**		*VDS-Based**		*STO-Based**		*VDS-Based**	
Traffic:	min	% LB	min	% LB	min	%LB	min	% LB	min	%LB
Uniform	80	23.3%	66.6	3%	66.6	3%	66.6	3%	66.6	3%
Quasi-uni2	75.8	29.2%	69.2	18%	67.6	15%	68.0	16%	66.5	13%
Ring	131.7	25%	133.5	27%	133.8	27.4%	129.	22%	127.	20.9%
Quasi-uni1	64.2	10.7%	-	-	-	-	60.8	5.2%	61.1	5.7%
Disconn.	312.0	35.20%	-	-	-	-	278.0	20%	279.0	21%
Central	335.0	0%	-	-	-	-	335.0	0%	335.0	0%

Table 1: Comparison to Previous Results and to Lower Bound

In this table, performance analysis is based on the value of minimum congestion and its deviation (percentage) from the computed lower bound. Under the "Previous Results" column, we introduce the output of [25], [26] for various traffic matrices. For a fair comparison, we determined same heuristic starting point, based on maximizing the 1-hop traffic, by solving the *transportation problem*. The numbers under "Greedy Start" summarize the performance of our heuristics using *edge perturbation*, from the same starting point as [25], [26]. Traffic matrices with a dash under this column are the ones for which the LP solution to the transportation problem results a starting configuration on which LP solution to the routing problem has an infeasible solution (see Section 3). Therefore, no comparison from the same starting point is possible for those traffic matrices. The numbers under the "Random Start" column are obtained by starting the heuristics from 29 random configurations and choosing the minimum value of the congestion.

From the comparisons we make the following remarks for this problem: First, there is no clear winner between the heuristics based on Variable Depth Local Search technique with backtracking, and the one based on modified Simulated Annealing. For some traffic models the former performs better; for some others the later does (see also table 4). The average performance of the heuristics is approximately 20% of the optimum for various traffic matrices (including the random traffic models).

Second, the greedy starting point does not necessarily give the best result, since some random initial configurations lead to a better local minimum. However, considering an average of over 30

[2] The computational results in this section were obtained on the SUN630 machines using *C* and *Unix*.

different starting points (see Table 2. mean minimum congestion values), the greedy approach seems reasonable.

Third, the perturbation heuristics demonstrated performance improvements of 5% to 20% over the previous results reported in [25], [26] on the same traffic models. The reported results are based on starting with a "greedy" topology and modifying it with branch-exchange operations. The routing problem is solved by using the Flow Deviation method [12]. We suspect that the choice of starting topology is one of the reasons that our algorithms outperforms the previous results. Another reason for the performance difference is the step size in the search process. The edge perturbation mode is 2-change operation whereas the branch-exchange operations on digraphs are quite tricky. Finally, Linear Programming for solving the Routing Problem may be a better choice (in contrast to the Flow Deviation method which is suggested in [12] for non-linear, *unconstraint* flow problems).

In order to study the sensitivity of our heuristics to different *starting topologies*, we generated random [3] starting points and computed the standard deviation of minimum congestion for each perturbation mode by gathering information from 30 different starting points. In the Table 2 and 3 below, we studied the contribution of LP optimization to the performance and compared edge and node perturbation models for VDS-Based and STO-Based heuristics, respectively.

	Mean Congestion				Stdt Dev. of Min. Cong.	
Traffic: Model	LP Phas1	LP Phas2	Min Edge P.	Min Node P.	Min Edge P.	Min Node P.
Uniform	86.37	81.27	66.79	69.43	0.43	1.24
Quasi-uni2	89.57	84.29	69.29	71.64	0.77	1.82
Ring	199.67	187.64	135.29	141.43	2.48	8.21
Quasi-uni1	80.86	75.91	62.61	64.85	1.27	0.91
Disconn.	405.10	387.13	284.83	299.75	4.89	47.23
Central.	359.53	338.23	335.00	335.00	0.00	0.00
Uniform2	86.24	81.35	67.01	70.59	0.67	6.99

Table 2: Performance Comparison of *Node Perturbation* and *Edge Perturbation* in STO-Based Heuristic

The numbers under the "Mean Congestion" column of these tables show the average value of congestion for the initial feasible solution to the routing problem ("LP Phas1"), after the LP optimization ("LP Phas2"), and for the final output of our algorithms ("Min Edge P." and "Min Node P." for both node and edge perturbation).

We have the following remarks from these tables: First, note that the congestion decrease from Phase 1 to Phase 2 is much less than the decrease obtained after Phase 2 by the perturbation algorithms. Hence, the performance of the algorithms does not heavily depend on the LP optimization.

[3]In the random generation, a configuration is represented as a 2-D array such that an entry $c_{i,j}$ in the array has 1, if there is a directed edge from node i to j, otherwise its value is 0 (also $c_{i,i} = 0 \forall i$). The entries with value 1 are chosen randomly provided that each row and column has exactly d number of 1's. The connectivity requirement for the CONF property, on a random configuration, is ensured by having an initial feasible solution to the routing problem.

Second, we confirmed the starting point sensitivity of our heuristics. For instance, the standard deviation of the minimum congestion obtained from different starting points is approximately 9.15 for *disconnected*, whereas it is 1.33 for *quasi-xn1* traffic matrix. Intuitively, high standard deviation indicates that depending on the starting configuration, there are different local optimas.

Third, a comparison of the perturbation models shows that in both algorithms (VDS- and STO-

	Mean Congestion				Stdt Dev. of Min. Cong.	
Traffic:	LP Phas1	LP Phas2	Min Edge P.	Min Node P.	Min Edge P.	Min Node P.
Uniform	86.37	81.27	67.53	70.58	1.26	1.90
Quasi-uni2	89.57	84.29	69.29	72.56	1.33	1.65
Ring	199.67	187.64	135.44	153.37	4.63	7.03
Quasi-uni1	80.86	75.91	62.83	66.10	1.30	1.57
Disconn.	405.10	387.13	292.34	309.3	9.15	20.19
Central.	359.53	338.23	335.00	335.00	0.00	0.00
Uniform2	86.24	81.35	67.18	70.88	1.11	1.96

Table 3: Performance Comparison of *Node Perturbation* and *Edge Perturbation* in VDS-Based Heuristic

based) the edge perturbation demonstrated better results as indicated in the average minimum congestion columns. This is expected since node perturbation in general effects almost twice many edges as edge perturbation, therefore causing bigger jumps in the search space, as indicated by standard deviation of the best solutions in the tables, and possibly missing a local optima. Therefore, we confirm that size and scope determines the performance of our heuristics. To study the problem further, we also generated random *traffic matrices*, in which each i, j pair decides (with 0.5 probability) whether to have a communication between them or not. For each pair that decided to communicate a random number is generated and mapped to a specific range to determine the amount of the traffic between them. We run our algorithms on these traffic models starting from 30 different configurations and compared the performances over the lower bound. The high standard deviations on the minimum congestion values indicate that a *large* random starting points needed to capture better local optimal.

The convergence of the algorithms are studied emprically. We obtained the value of the minimum congestion at every k steps of an execution of each heuristics to determine the number of iterations to reach the best solution. The plotting of congestion as a function of iteration for *ring* and *quasi-uni2* traffic matrices is shown in figures 7 and 8. These plottings are based on the statistics gathered from the execution of the heuristics on 30 different starting points. The top and bottom *dash lines* indicates the maximum and minimum congestion at that specific iteration. The middle dash line shows the mean, and the vertical bars indicates the standard deviation. The flat part at the tail of the plots indicates that the value of the cost function remains constant such that consecutive iteration of the algorithm does not improve the solution. This information is important to determine the length of the search process.

Traffic	*STO-Based*	*VDS-Based*
random1	25%	25%
random2	19%	19%
random3	21%	20%
random4	25%	26%
random5	23%	24%
random6	22%	23%
random7	25%	23%
random8	22%	21%
random9	15%	16%
random10	17%	17%
Uniform	3%	3%
Quasi-uni2	16%	13%
Ring	22%	20.9%
Quasi-uni1	5.2%	5.7%
Disconn.	20%	21%
Central	0%	0%

Table 4: Performance comparison of the heuristics over the lower bound for the random traffic

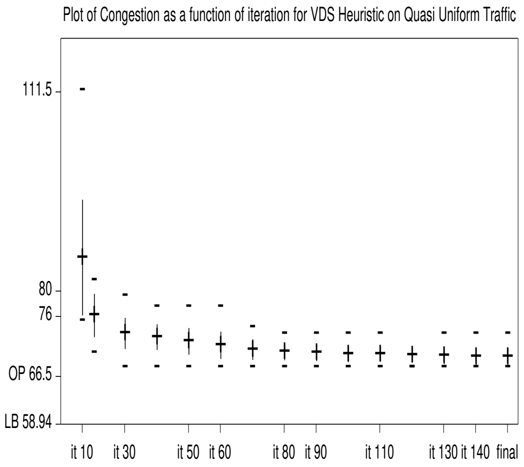

Figure 7: VDS-Based on Quasi-uni2

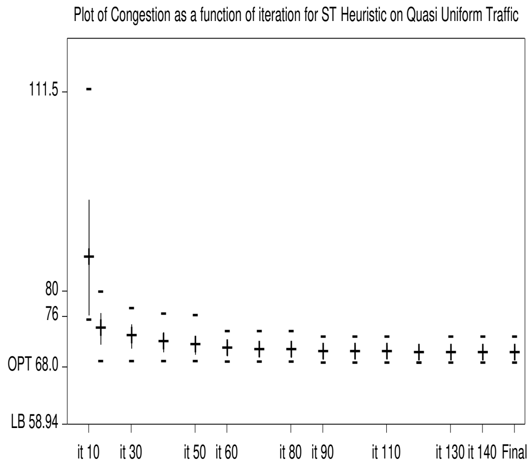

Figure 8: STO-Based on Quasi-uni2 Traffic

6 Lower Bound

In this section, we examine the lower bound of the problem. Let traffic matrix T, and degree constraint d be any input to the problem (such that $in\_degree = out\_degree = d$ at each node). A trivial 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 [25]. Precisely, let LB_t denote the trivial lower bound then

$$LB_t = \frac{max_{i,j}\{\sum t_{i,j}, \sum t_{j,i}\}}{d} \qquad (1)$$

Figure 9: VDS-Based on Ring

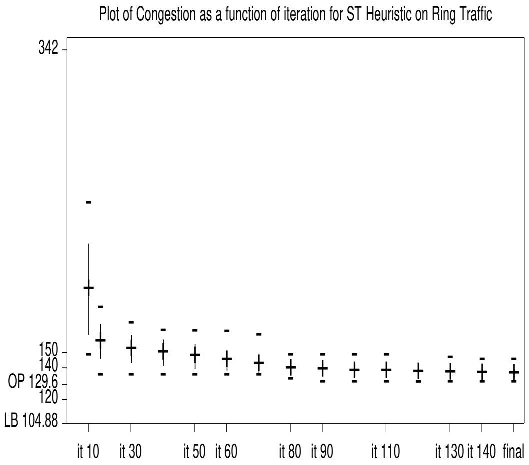

Figure 10: STO-Based on Ring

Clearly this lower bound is not a good one since it does not take into account the routing of the traffic. Following we show how to improve this trivial lower bound by using spanning trees and concentrating on a single commodity at a time.

Denote by r the commodity originated from the node r (recall that a commodity associated with node r, if $\sum_j t_{r,j} > 0$). Consider an $\geq$ ordering T_r of the amount of traffic $t_{r,j}$ from r to all other sink js. Let identification number $id(j) = [1, \cdots, N-1]$ of the node j be the sequence number (index) of the element $t_{r,j} \in T_r$.

Definition 1 *1. A flow tree FT_r for a commodity r is a directed d-ary spanning tree, rooted at the commodity node r.*

2. Cost of a flow tree is the total amount of flow carried on it to route the traffic in T_r.

Let $level(j)$ of a node j be the length (number of hops) of the path from the root to j in a flow tree.

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

Definition 2 *A minimum total flow tree MFT_r for a commodity r is a flow tree rooted commodity node r, such that $Cost(MFT_r)$ is the minimum among all the FT_rs*

Fact 2 *1. A flow tree FT_r is a minimum total flow tree if d^i next elements of the set T_r are at the level i of the tree, for $i = 1 \cdots logN$.*

2. Any permutation of the nodes at the same level of MFT_r has equivalent cost and does not change the cost of the tree.

Definition 3 *1. A constrained flow tree $CFT_r^{i,j}$ is a flow tree rooted commodity node r, such that directed edge (i,j) is forced to be in $CFT_r^{i,j}$ regardless of the $t_{r,j}$ value.*

2. A minimum flow constraint tree $MCFT_r^{i,j}$ is a constrained flow tree such that $Cost(MCFT_r^{i,j}) - Cost(MFT_r)$ is minimum.

Denote by Z the maximum amount of congestion such that $Z = max_{i,j}\{\sum_r f^r_{i,j}\}$ (i.e: the maximum flow assignment on edge (i,j))

Claim 1 *$Z \geq \frac{\sum_{i,j}\sum_r f^r_{i,j}}{Nd}$ gives a lower bound on the congestion.*

Proof: $Nd(max_{i,j}\{\sum_r f^r_{i,j}\}) \geq \sum_{i,j}\sum_r f^r_{i,j}.\Box.$

Depending on the relaxation on the r.h.s. of the inequality, the accuracy of the lower bound is determined.

Theorem 1 *Given a traffic matrix T and degree constraint d*

1. $Z \geq \frac{\sum_r cost(MFT_r)}{Nd} = LB_0, \forall r$

2. $Z \geq \frac{min_{ij}\{\sum_r cost(MCFT^{i,j}_r)\}}{Nd} = LB_1, \forall(i,j), \forall r$

3. $LB_0 \leq LB_1$

Proof

1. Consider a graph G_0 which is obtained by the union of the edges of the MFT_r trees for all commodity r. Due to the definition 1, total amount of flow on an edge (i,j) is $\sum_r f^r_{i,j} = \sum_{r, if(i,j)\in MFT_r} level(j) \times t_{r,j}$. Thus the total flow carried on the graph G_0 is $\sum_r \sum_{(i,j)} f^r_{i,j}$. Since there are only Nd edges available, the congestion is bounded by $\frac{\sum_r \sum_{(i,j)} f^r_{i,j}}{Nd} = LB_0$.

2. Suppose (i,j) is a directed edge in the optimal solution. Then it must occur in all the flow trees. Let graph $G^{i,j}_1$ obtained, over all commodities, by the union of the edges of the $MCFT^{i,j}_r$ trees. The total flow carried on this graph can be computed as before. Let's generate all such graphs by fixing each edge in turn and let $G^{u,v}_1$ be the one on which the total flow is minimum. In other words, flow carried on this graph is the least required for routing of the traffic T. Thus averaging the flow on this graph by the feasible number of the edges gives a lower bound LB_1.

3. Due to the definition of a minimum constrained flow tree

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

Computing LB_0 is quite straightforward since it involves building MFT_r for all commodity in time complexity $O(N^2)$. However computation of LB_1 is much less trivial, following we introduce an algorithm for this computation.

Consider a directed edge (i,j) to be forced into the optimal solution. We must insert this edge in all the MFT_r, yet ensure that the modified trees are $MCFT^{i,j}_r$. Depending on the level(i) and level(j) in the MFT_r, an edge can be forced to occur one of the three types in the $MFT^{i,j}_r$:

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

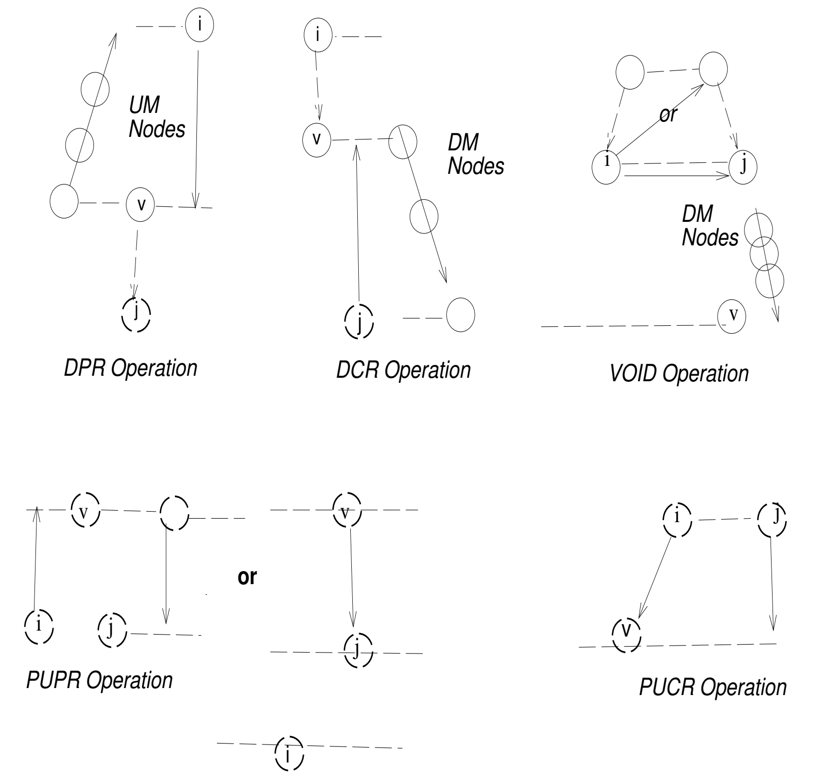

Figure 11: Edge Insertion Operations

Given a MFT_r, insertion of edge (i, j) into a minimum flow tree is based on exactly one of the following five operations (see figure: 11).

Down edge with Parent Replacement: DPR

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

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 the son of i, and define the set DM of k *downward migrating nodes* which are the rightmost nodes of the MFT_r between the level of j and v, ($k = level(j) - level(v)$).

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 v the rightmost leaf of the MFT_r. If MFT_r is full (i.e., there are exactly d^m leaves where m is the maximum depth) then let $k = level(v) - level(i) + 1$ else let $k = level(v) - level(i)$. Define the set DM of k *downward migrating nodes* which are the set of rightmost nodes (not including j) whose level is in between $level(i) + 1, \cdots, level(i) + k$.

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

What follows is an algorithm to construct the constraint flow trees and compute LB_1.

procedure *compute_LB*1(T, d)
input: Traffic matrix T, degree d
output: LB_1
begin
preprocessing: for all commodity r, build MFT_r
  begin: Do for all edge (i, j)
    begin: Do for all commodity r
      determine the *direction* of the edge
      If it is a *Down* edge then compute and choose $min\{T(DPR), T(DCR)\}$
      Else compute and choose $min\{T(PUPR), T(PUCR), T(VOID)\}$.
    end
  end
  Determine the edge (u, v) s.t $\sum_r Cost(MCFT_r^{u,v})$ is minimum
  $LB_1 = \frac{\sum_r Cost(MCFT_r^{u,v})}{dN}$
end

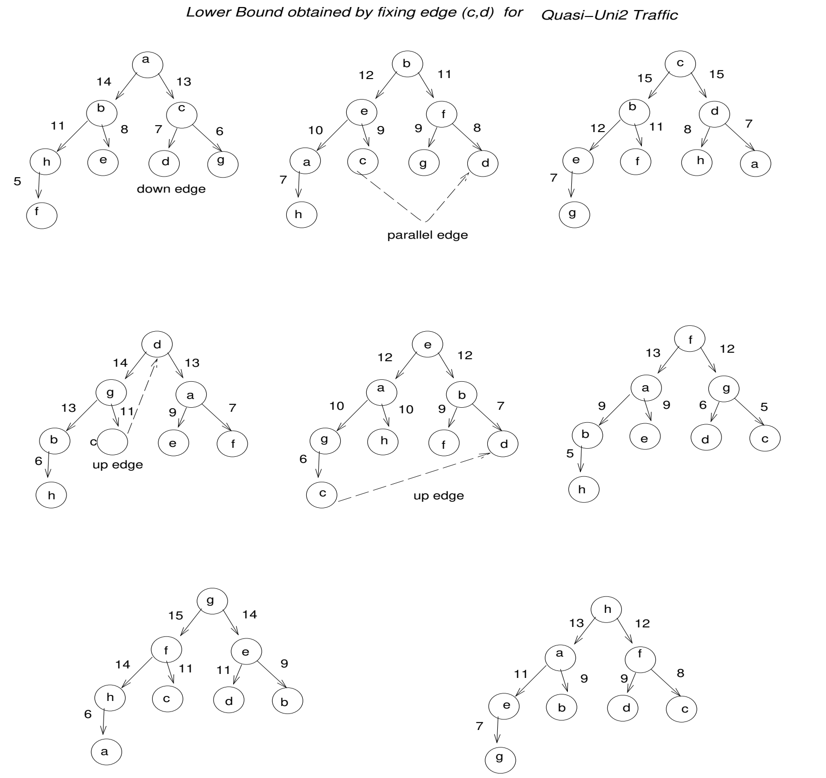

Figure 12: Computation of LB_1 for Quasi-uni2 Traffic Matrix

We give the following definitions to simplify the correctness proof of the algorithm.

Given a flow tree FT_r, denote by x and y minimum and maximum identification number (id) of the nodes at level k of the flow tree FT_r, respectively. Let z be the maximum id number in FT_r. Similarly, denote by x', y', z', the corresponding indices in MFT_r.

Definition 4 *A flow tree FT_r has an upper cost equivalent component UC with MFT_r at level k if:*

$$\sum_{id(j)=1}^{id(j)=x} t_{rj} \times level(j)_{j \in FT_r} = \sum_{id(j)=1}^{id(j)=x'} t_{rj} \times level(j)_{j \in MFT_r}$$

A flow tree FT_r has an lower cost equivalent component LC with MFT_r at level l if:

$$\sum_{id(j)=y}^{id(j)=z} t_{rj} \times level(j)_{j \in FT_r} = \sum_{id(j)=y'}^{id(j)=z'} t_{rj} \times level(j)_{j \in MFT_r}$$

Theorem 2 *Given T and d , procedure compute_LB1(T,d) constructs minimum flow trees and computes the lower bound LB_1.*

Proof: We analyze three cases determined by the direction of a forced edge (i, j) and prove by contradiction that another constrained flow tree not coexist with less cost than the output $MCFT$ of the algorithm.

First, note that each operation in the algorithm preserves a lower and/or upper cost equivalent component as follows (omitting the subscripts from the notation since it is sufficient to consider one commodity):

DPR : UC at $level(i) - 1$ and LC at $level(j)$

DCR : UC at $level(i)$ and LC at $level(j) + 1$

$VOID$: UC at $level(i)$

$PUPR$: UC at $level(j) - 2$ and LC at $level(i) + 1$

$PUCR$: UC at $level(j) - 1$ and LC at $level(i) + 2$

Therefore we can limit our argument (on the minimum cost property of $MCFT$) to the nonequivalent zone on the $MCFT$ tree.

We note that, the only difference between the MFT and the output MCFT is due to (exactly) one of the operations. Insertion of an edge (i, j) can be a replacement of the parent of the j with i, or it can be a replacement of one of the children of i or it can be a void edge meaning that neither i nor j is moved.

Since the number of the migrating nodes is exactly determined by each one of the operation, the choice of nodes into the UM and DM sets determines the net cost difference from the minimum flow tree. The UM sets contain the nodes with largest traffic to be moved up in the tree. Thus 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 minimum. Therefore, any other tree that has different UM and DM sets can not have *less* cost than $MFCT$. Since, there does not exists any other constraint flow tree with less cost with respect to the non-cost equivalent part of the MFCT, there is no tree with total cost less than MFCT unless MFT does not have minimum total cost. This is true since any other constrained tree must also have *at least* the same equivalent components to ensure *at most* the same cost with $MFCT$. □

Note here that by fixing two or more edges and building constraint flow trees, the lower bound gets higher, however the computation would get quite expensive and complicated. An alternative approach would be to apply *cutting plane algorithms* to obtain an integer linear programming solution to the problem. These techniques are out of the scope of this paper.

On the other hand, $LB_1 \geq LB_t$ 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. Then our spanning tree approach does not perform well since one or two MCFT trees will have much higher cost than the others and their cost will be distributed over the other edges. Therefore, in our simulations, we compare LB_1 and LB_t and take the maximum.

7 Complexity Results

In this section, we study the complexity of the unified approach to network design and minimum congestion routing problem.

We are interested in determining the complexity of choosing a configuration CONF, on which minimum congestion routing problem is solved optimally. Following we prove that for a given integral traffic matrix, finding a configuration and an integer routing [4] of the messages on this configuration with minimum congestion is NP-complete.

ICONF problem:

Instance: N processors, a traffic matrix $T = \{t_{i,j}\}$, such that for all processor pairs i, j:

$t_{i,j} = 0$, if $i = j$; $t_{i,j} \in Z^+$ otherwise;

and integers $r, B \in Z^+$.

Question: is there a configuration CONF and a routing R of the traffic T on this configuration, such that congestion on any edge is at most B and *in* and *out* links of each processor is exactly r. We show that ICONF is NP-complete.

Theorem 3 *ICONF is NP-complete.*

We note that as r gets closer to the N the complexity of the problem changes. For instance for a complete graph it is trivial to decide. Therefore we assume that $r << N$ and consider the case $r = 1$.

To show ICONF is NP-complete, we transform *minimum cut linear arrangement (MCLA)* [14][15] problem to ICONF.

MCLA problem:

Instance: Graph $G = (V, E)$, $K \in Z^+$

Question: Is there a one-to-one function $f : V \to \{1, 2, \cdots, |V|\}$ such that for all i, $1 < i < |V|$

$|\{(u, v) \in E : f(u) \leq i < f(v)\}| \leq K$.

Before we prove the theorem let's introduce some notation. Denote by $u_1^i, u_2^i, \cdots, u_N^i$ a linear arrangement of the vertices of G. Intuitively, u_k^i denotes the index k of the node u^i assigned by the function f for a particular cut. Given a linear arrangement, define a *cut* (u_x^i, u_{x+1}^i), such that $u_z^i \in S$, $\forall z \leq x$, and $u_z^i \in V - S$, $\forall z \geq x + 1$. Denote by $k(x, x + 1)$ the size of the cut between the vertices u_x^i and u_{x+1}^i. Thus $k(x, x + 1)$ is the number of edges that goes from all the nodes in the set S to the $V - S$ in this linear arrangement.

[4] Given a traffic matrix T and a configuration CONF, define routing R as a function which takes each nonzero entry s, t of T and maps it to a set of edges $(i, j) \in CONF$ such that the edges in the set induces a directed path from source s to sink t.

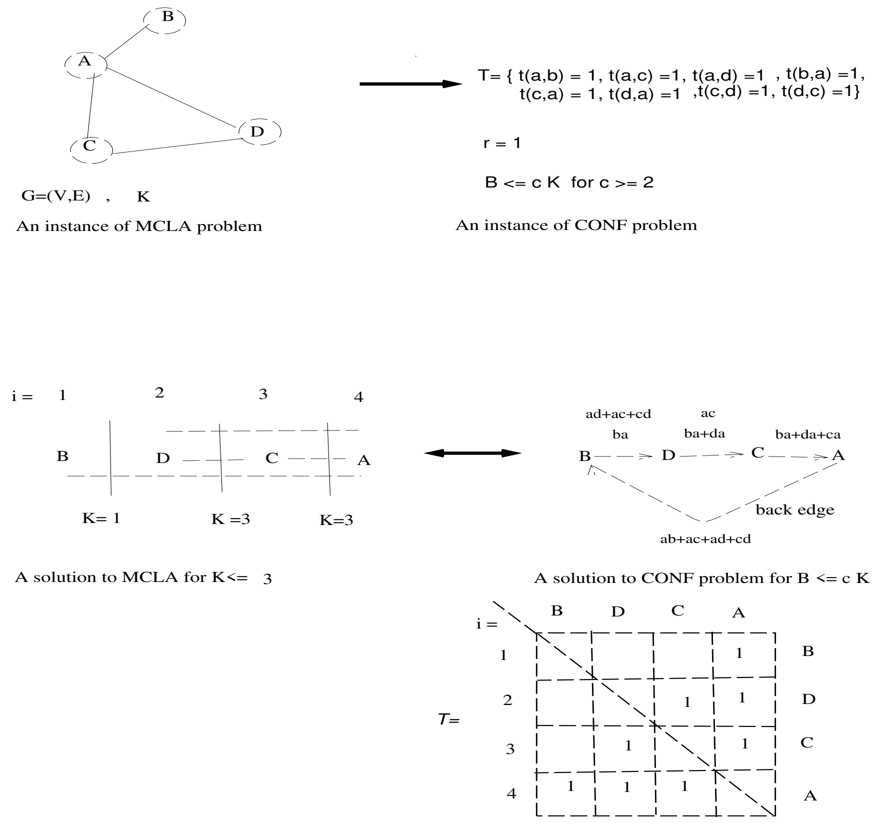

Figure 13: MCLA and CONF problems

ICONF is in NP since a nondeterministic Turing Machine can guess a configuration CONF and a routing R of the traffic on it, and the configuration properties (i.e: degree requirement, no-self looping, strong connectivity) and the congestion bound B can be checked in polynominal time.

Let $P1 : G = (V, E), K$ be any instance of MCLA problem, where $|V| = N$ (see figure: 13). We give the following construction of an instance $P2$ of ICONF problem with a traffic matrix T and integers r, B such that there exist a configuration CONF and a routing R of the traffic T with congestion at most B, if and only if there is a minimum cut linear arrangement of size K on G in problem $P1$.

<u>Construction:</u>

1. We construct a traffic matrix T such that $t_{i,j} = t_{j,i} = 1$, if edge $(i, j) \in E$, otherwise $t_{i,j} = t_{j,i} = 0$.
2. $B \leq 2K$

Clearly this construction can be done in polynominal time.

Before we present the NP-C proof let's note some properties implied by this construction. First, the traffic matrix T is a symmetric one with entries 0 or 1 indicating whether there is a connection request or not from i to j. As a result, we can redefine the congestion on an edge as the total number of occurrences of this edge in all the source-sink paths to satisfy the traffic matrix.

Second, note that a graph with CONF property and degree $r = 1$ must be a Hamiltonian Circuit. The permutation of the stations on the circuit will determine the value of the congestion since the routing is trivial. Thus due to our construction, choosing a configuration CONF will also determine the routing R. In other words we have a unique routing once the configuration is determined.

We restate our claim that there is a configuration CONF and routing R of T such that an edge is visited at most B messages (each belongs to a distinct source-sink pairs), iff G has a linear arrangement of size K. Now we proceed with the proof:

Proof

$\Rightarrow$: Suppose there exists an algorithm that solves the ICONF problem, we show that given an answer to $P2 \in ICONF$, we obtain an answer to $P1 \in MCLA$.

A *yes* answer to the P2 ensures the existence of a configuration and the routing of the traffic within the congestion bound. Thus, let CONF be the configuration and R be the routing of the traffic in T with congestion at most B. Answer to $P1 \in MCLA$ must also be *yes* as explained below.

Since the CONF is a directed cycle, there are exactly N linearization of this cycle by breaking it between any two nodes to obtain a single directed path. A numbering of the nodes from left to right on a path gives a valid linear arrangement (therefore, given a CONF there are exactly N possible linear arrangements).

Consider a linearization of the CONF such that the nodes in this path have the same index numbers as in the traffic Matrix T. Then, the routes that can be realized on this directed path belong to the traffic from the upper triangular part of T.

Any edge in this path will be visited by the traffic from a source node to a destination, iff the index of the (i) source node in T is less than and (ii) the index of the destination node is greater than the indices of the end points of this edge.

In other words, the total number of messages that can go through the edge (v, v') in the path is the sum of the entries in the traffic matrix whose row index is less than the index of u and column index greater or equal to v.

Formally, let x and $x+1$ be the indices of the end points v and v', respectively, then congestion due to the upper triangular part of T, on the edge (v, v') is

$$\sum_{y=1}^{x} \sum_{z=x+1}^{N} t_{u_y^i, u_z^i}$$

Without lost of generality, let edge (v, v') be the edge on this linearization of the CONF such that this sum is the maximum. Since congestion is the number of messages passing through an edge and a message in P2 corresponds to an edge in P1, the maximum number of edges in any cut is bounded by this quantity (see the example in Figure 13).

Therefore, knowing B which is the value of maximum congestion on an edge due to upper and lower triangular part of the traffic matrix, we can decide whether or not this linear arrangement satisfies the cut size constraint by checking $2K \geq B$.

$\Leftarrow$: Conversely, a *yes* answer to the problem $P1$ ensures the existence of a linear arrangement of the vertices. Thus, let $u_1^i, u_2^i, \cdots, u_N^i$ be a linear arrangement of the vertices of G such that the cut between any two vertices u_x^i and u_y^i has at most K edges from the edge set E.

In order to prove that $P2 \in ICONF$ must also have a *yes* answer, first we show that a *yes* answer to P1 ensures the existence of a CONF. Second we argue that a *yes* to P1 also ensures a routing of the traffic T with the congestion B.

By assuming a directed edge from u_x^i to u_{x+1}^i for each index $x = 1, \cdots, N-1$ and then a directed edge (call this the back-edge) from node u_N^i to node u_1^i we obtain a unique cycle. The induced graph is a configuration, since it is strongly connected, its in/out degree is uniform, and there are no self-loops. Thus we have a valid CONF from the linear arrangement.

Let's consider the routing on this CONF. Note that by definition, there is an edge in the linear arrangement from a node u_x^i to u_y^i iff $x < y$. Note also that an edge in $P1$ corresponds to an entry in the traffic matrix in $P2$. Therefore the indices of the vertices in the linear arrangement give the indices of the nodes in the traffic matrix T. In other words j'th vertex in the linear arrangement (assuming the numbering from left to right and starting from 1) is in the j'th row in the traffic matrix.

Therefore the cuts on the linear arrangement give the routes for the traffic in the upper triangular part of the matrix T with congestion at most K (see the example in Figure 13). In other words by looking at the cuts we can trace the source-sink paths. For example if edge (i,j) occurs in the cuts between (i,a), (a,b), (b,c) and (c,j), then the route for traffic $t_{i,j}$ is $i \rightarrow a \rightarrow b \rightarrow c \rightarrow j$.

Consider any index x in the linear arrangement, the congestion on the edge (u_x^i, u_{x+1}^i) to carry the traffic from upper triangular part of T is

$$k(x, x+1) = \sum_{y=1}^{x} \sum_{z=x+1}^{N} t_{u_y^i, u_z^i}$$

However, to satisfy the lower triangular part of the traffic matrix the back-edge must carry all the traffic from this part. Thus the congestion on the back-edge is $\sum_{y=1}^{N-1} \sum_{z=y+1}^{N} t_{y,z}$. Due to the symmetry, this is equal to $\frac{1}{2} \sum_{y=1}^{N} \sum_{z=1}^{N} t_{y,z}$. Thus, the contribution of the traffic from the lower triangular part of the matrix, on the cut between the nodes with indices $x, x+1$ can be written as

$$b(x, x+1) = \sum_{y=N}^{x+2} \sum_{z=x+1}^{y-1} t_{u_z^i, u_y^i}$$

$$b(x, x+1) = \sum_{y=x+2}^{N} \sum_{z=x+1}^{y-1} t_{u_y^i, u_z^i} \tag{2}$$

Thus total congestion on any cut between the nodes with index x and $x+1$ is

$$k(x, x+1) + b(x, x+1) \tag{3}$$

Because of the symmetry, and 0-1 property of the traffic matrix, the total of $t_{u_y^i, u_z^i}$ entries of equations 1 and 2 are always a subset of the lower/upper triangular part of T.

Therefore,

$$k(x, x+1) + b(x, x+1) \leq \sum_{y=1}^{N-1} \sum_{z=y+1}^{N} t_{y,z} \leq B \tag{4}$$

Note that $b(x, x+1)$ can be at most K, and K is bounded by $max_x\{k(x, x+1)\}$ for all indices x between 1 and N-1. Thus, having the answer to P1 with K, we can decide whether or not there is a solution to P2 with B by checking $B \leq 2K$. $\Box$

8 Conclusion

The problem of finding a best embedding for minimum congestion routing is a hard problem thus, we introduced two heuristics for approximate solutions. We compared our heuristics against to each other and conclude that their performance is quite the same. We also computed a lower bound of the problem observed that our performance results are in average (including the random traffic matrices) 20% this lower bound. Since our lower bound is not a tight one, we suspect that the performance results are actually better. Comparison to the previous results, on the same data set, shows that our heuristics results %5 to %20 decrease on the previous congestion values. We also examined the sensitivity of our algorithms to different starting points and obtained various statistical data by initiating the heuristics from 30 different configurations.

Acknowledgments

I would like to thank to Daniel Bienstock for his guidance on the optimization issues, to Moti Yung, Pino Italiano, J.P. Labourdette, Matthew Franklin, for their remarks and comments on the earlier versions of this paper.

References

[1] A.S.Acampora, "A Multichannel, Multihop Local Lightwave Network," *Proc. IEEE GLOBCOM'87*, November 1987.

[2] A.S. Acampora and M.J.Karol, "An Overview of Lightwave Packet Networks," *IEEE Network Magazine*, Vol., 3, pp 29-41, January 1989.

[3] A.Bannister,L.Fratta, and M.Gerla, "Topological Design of the Wavelength-Division Optical Network", *Proc. IEEE INFOCOM'90*, June 1990.

[4] A.Bannister, M.Gerla, "Design of the Wavelength-Division Optical Network", *Proc. Intern, Conf. on Commun.*, April 1990.

[5] Biswanath Mukherjee, "WDM-Based Local Lightwave Networks Part I: Multihop Systems," *IEEE Network Magazine*, May 1992.

[6] Biswanath Mukherjee, "WDM-Based Local Lightwave Networks Part II: Multihop Systems," *IEEE Network Magazine*, July 1992.

[7] Y.Ben-Asher, D.Peleg, R.Ramaswami, A.Schuster, "The Power of Reconfiguration," *Manuscript*, May 1991.

[8] K. Bala, T.E.Stern, and K.Bala, "Algorithms for Routing within a Linear Lightwave Network," *Proc. INFOCOM'91*, Bal Harbour, Florida, April 1991.

[9] C.A.Brackett, "Dense Wavelength Division Multiplexing Networks: Principles and Applications," *IEEE J. Sel. A.r in Commun.*, Vol. 8, August 1990.

[10] I.Chlamtac, A.Ganz, and G.Karmi, "Lightnet: Lightpath based solution fro wide bandwidth WANs," *Proc. IEEE INFOCOM'90*, San-Francisco, California, June 1990

[11] I.Chlamtac, A.Ganz, and G.Karmi, "Transport Optimization in Broadband Networks," *Proc. IEEE INFOCOM'91*, Bal Harbour, Florida, April 1991.

[12] L.Fratta, M.Gerla, and L.Kleinrock, "The Flow Deviation Method: An Approach to Store-and-Forward Communication Network Design," *Networks,* Vol. 3, 1973.

[13] M.Gerla. H.Frank, J.Eckl, " A Cut Saturation Algorithm for Topological Design of Packet Switched Communication Networks," *Proc. NTC.*, December 1974.

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

[15] F. Gavril, "Some NP-Complete Problems on Graphs," *Proc. 11'th Conf. on Information Sciences and Systems*, Johns Hopkins University, Baltimore, MD. 1977.

[16] M.G.Hluchyj, M.J.Karol, "Shuffle Net: An Application of Generalized Perfect Shuffles to Multihop Lightwave Networks," *Proc. IEEE INFOCOM'88*, March 1988.

[17] *IEEE J. Sel. Areas in Commun., Special Issue on Fiber Optics for Local Communications*, Vol., 3, November 1985.

[18] *IEEE J. Sel. Areas in Commun., Special Issue on Dense WDM Tech. for High Capacity and Multiple Access Communication Systems*, Vol., 8 No. 6 August 1990.

[19] S.Kirkpatrick,C.D.Gelatt,M.P.Vecchi, "Optimization by Simulated Annealing," *Science*, Vol. 220, 1983.

[20] B.W.Kerninghan, S.Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," BSTJ, Vol. 49, no.2, February 1970

[21] K.Sivarajan, R.Ramaswami, "Multihop Lightwave Networks Based on De Bruijn Graphs," submitted to *IEEE Trans. on Commun.*

[22] K.Sivarajan, R.Ramaswami, "A Packet-Switched Multihop Lightwave Network Using Subcarrier and Wavelength Division Multiplexing," submitted to *IEEE Trans. on Commun.*

[23] K.Sivarajan, R.Ramaswami, "Multihop Lightwave Networks Based on De Bruijn Graphs," *Proc. IEEE INFOCOM'91*,

[24] J.P.Labourdette, A.S.Acampora, " Wavelength Agility in Multihop Lightwave Networks," *Proc. IEEE INFOCOM'90*, June 1990.

[25] J.P.Labourdette, A.S.Acampora, "Logically Rearrangeable Multihop Lightwave Networks," *IEEE Trans. Commun.*, Vol. 39, August 1991.

[26] J.P.Labourdette, "Rearrangeability Techniques for Multihop Lightwave networks and Application to Distributed ATM Switching Systems," *Ph.D. Thesis*, Columbia University CTR Dept. of E.E., 1991.

[27] N.F.Maxemchuk, " Regular Mesh Topologies in Local and Metropolitan Area Networks," *AT&T Tech. Jrn.*, Vol. 64, September 1989.

[28] N.F.Maxemchuk, "Routing in the Manhattan Street Network," *IEEE Trans. Commun.*, Vol. COM-35, May 1987.

[29] Y.Ofek, M.Yung, "Principles fro High Speed Network Control: Lossless-nes and deadlock-freeness, self-routing and a single buffer per link," *9'th Ann. ACM Symp. on Princ. Distr. Computing* 1990.

[30] P.Raghavan, "Probabilistic Construction of Deterministic Algorithms: Approximating Packing Integer Programs," *Jrnl. Comp. and Sys. Sciences*, Vol. 37 1988

[31] P.Raghavan,C.Thomson, "Provably Good Routing in Graphs: Regular Arrays," *ACM STOC*, 1985.

[32] T.E.Stern, " Linear Lightwave Networks," Tech. Rep. No: 184-90-14, CTR, Columbia University, New York, 1990

[33] A.S.Tanenbaum, “Computer Networks,” Prentice-Hall, Inc. Englewood Cliffs, N.J., 81

9 Appendix: The Sample Traffic Matrices

Uniform Traffic Matrix							
0.00	10.0	10.0	10.0	10.0	10.0	10.0	10.0
10.0	0.00	10.0	10.0	10.0	10.0	10.0	10.0
10.0	10.0	0.00	10.0	10.0	10.0	10.0	10.0
10.0	10.0	10.0	0.00	10.0	10.0	10.0	10.0
10.0	10.0	10.0	10.0	0.00	10.0	10.0	10.0
10.0	10.0	10.0	10.0	10.0	0.00	10.0	10.0
10.0	10.0	10.0	10.0	10.0	10.0	0.00	10.0
10.0	10.0	10.0	10.0	10.0	10.0	10.0	0.00

Quasi-Uni2 Traffic Matrix							
0.00	14.0	13.0	7.00	8.00	5.00	6.00	11.0
10.0	0.00	9.00	8.00	12.0	11.0	9.00	7.00
7.00	15.0	0.00	15.0	12.0	11.0	7.00	8.00
13.0	13.0	11.0	0.00	9.00	7.00	14.0	6.00
12.0	12.0	6.00	7.00	0.00	9.00	10.0	10.0
13.0	9.00	5.00	6.00	9.00	0.00	12.0	5.00
6.00	9.00	11.0	11.0	14.0	15.0	0.00	14.0
13.0	9.00	8.00	9.00	11.0	12.0	7.00	0.00

Ring Traffic Matrix							
0.00	100.	9.00	10.0	8.00	10.0	11.0	8.00
9.00	0.00	110.	9.00	10.0	10.0	8.00	10.0
11.0	9.00	0.00	80.0	8.00	9.00	10.0	9.00
11.0	10.0	10.0	0.00	90.0	9.00	9.00	10.0
8.00	10.0	8.00	9.00	0.00	100.	11.0	9.00
9.00	9.00	10.0	10.0	9.00	0.00	120.	11.0
8.00	11.0	11.0	10.0	8.00	9.00	0.00	90.0
110.	9.00	8.00	7.00	10.0	11.0	8.00	0.00

Quasi-Uni1 Traffic Matrix							
0.00	11.0	10.0	9.00	9.00	10.0	8.00	7.00
8.00	0.00	7.00	10.0	11.0	10.0	9.00	9.00
9.00	10.0	0.00	11.0	12.0	8.00	9.00	8.00
11.0	11.0	10.0	0.00	7.00	8.00	9.00	9.00
9.00	10.0	11.0	7.00	0.00	8.00	8.00	10.0
10.0	8.00	8.00	9.00	10.0	0.00	10.0	10.0
11.0	10.0	9.00	9.00	8.00	7.00	0.00	9.00
8.00	8.00	10.0	10.0	11.0	9.00	9.00	0.00

Disconnected Traffic Matrix							
0.00	100.	110.	90.0	9.00	8.00	10.0	10.0
90.0	0.00	110.	90.0	10.0	8.00	10.0	9.00
100.	120.	0.00	80.0	9.00	9.00	11.0	10.00
80.0	90.0	100.	0.00	11.0	10.0	8.00	7.00
7.00	8.00	11.0	10.0	0.00	100.	110.	80.0
12.0	8.00	9.00	9.00	90.0	0.00	90.0	80.0
8.00	11.0	10.0	11.0	100.	110.	0.00	110.
9.00	11.0	10.0	10.0	110.	80.0	90.0	0.00

Centralized Traffic Matrix							
0.00	100.	80.0	70.0	110.	90.0	110.	100.
120.	0.00	8.00	10.0	9.00	10.0	10.0	9.00
80.0	11.0	0.00	7.00	9.00	10.0	11.0	10.0
90.0	10.0	11.0	0.00	8.00	8.00	7.00	9.00
100.	9.00	10.0	9.00	0.00	9.00	10.0	11.0
90.0	11.0	8.00	11.0	11.0	0.00	10.0	9.00
110.	8.00	9.00	9.00	10.0	10.0	0.00	7.00
80.0	7.00	10.0	11.0	8.00	8.00	9.00	0.00