Robust Routing with Unknown Traffic Matrices

Vahid Tabatabaee, Abhishek Kashyap, Bobby Bhattacharjee, Richard J. La, Mark A. Shayman
University of Maryland, College Park, MD 20742, USA
Email: {vahid@cs, kashyap@eng, bobby@cs, hyongla@eng, shayman@eng}.umd.edu

***Abstract*— In this paper, we present an algorithm for intra-domain traffic engineering. We assume that the traffic matrix, which specifies traffic load between every source-destination pair in the network, is unknown and varies with time, but that always lies inside an explicitly defined region. Our goal is to compute a *fixed* robust routing with best worst case performance for all traffic matrices inside the bounding region.**

We formulate this problem as a semi-infinite programming problem. Then, we focus on a special case with practical merits, where (1) the traffic matrix region is assumed to be a polytope specified by a finite set of linear inequalities, and (2) our objective is to find the routing that minimizes the maximum link utilization. Under these assumptions, the problem can be formulated as a polynomial size linear programming (LP) problem with finite number of constraints. We further consider two specific set of constraints for the traffic matrix region. The first set is based on the hose model and limits the total traffic rate of network Point of Presence (PoP) nodes. The second set is based on the pipe model and limits the traffic between source-destination pairs. We study the effectiveness of each set of constraints using extensive simulations.

Using simulation results on Rocketfuel topologies, we study and discuss effectiveness and characteristics of the proposed algorithm for real world network topologies. Simulation results show that robust routing is promising, and the number of paths required is limited and manageable. They also show the combination of hose and pipe model constraints can further enhance the performance.

I. INTRODUCTION

We present a traffic engineering algorithm for cases when the precise traffic load is not known. This is contrary to common approaches for traffic engineering [7], [11], which assume that the traffic load in the network is known or can be measured. Relying on precise knowledge of the traffic matrix enables nice provably optimal solutions [19]. However, in practice, traffic demands between nodes change continuously. With the increasing popularity of higher bandwidth applications (e.g., file swarming, online- and offline-video distribution), traffic patterns are more volatile even in the aggregate. Further, multi-homed customers cause abrupt changes in aggregate traffic by shifting traffic between networks [21]. Hence, traffic engineering schemes that require precise knowledge of the current traffic matrix must rely on on-line monitoring [18], and update their routes as traffic changes. These distributed load-based updates can lead to complexity and even network instability problems [14].

The alternative is to use a *fixed* routing that does not adapt to the traffic changes. In fact, most deployed networks fall into this category, and the (fixed) routing usually optimizes a static metric, such as hop count. In this paper, we describe an algorithm (called *robust routing*) that produces a single routing that does not change over time. In our solution, we incorporate readily available unchanging information about the traffic as follows: Even though instantaneous traffic is variable and demands unpredictable, certain information, e.g. maximum possible demand between two nodes or specific link capacities, is available and unchanging. Indeed, this information was the basis of the "hose" model [6]. In our solution, we assume that while the current (time-varying) traffic matrix is unavailable, the network provider is able to supply an *upper* bound on the traffic rate between each source-destination pair and total outgoing (incoming) traffic rate of a source (destination) node. This *feasible region* can be derived (loosely) using link/gateway capacities; finer grained information, which will result in better performance, can be derived using traffic history, existing SLAs, etc.

Instead of tailoring the routing exactly to a given traffic matrix, we find a single routing that works "reasonably" well for all traffic matrices within a feasible region. In particular, for a given cost function (say link utilization), our solution finds the routing that minimizes maximum cost over all traffic matrices inside the feasible region. Finding a single feasible "good" routing given some demand information is an active area of research, with *Oblivious Routing* [3], [4] being perhaps the best known example. Oblivious routing [4] finds the single path assignment with best *relative* performance compared to the best performance possible for any arbitrary traffic matrix. This is a powerful result; however, oblivious routing does not provide absolute performance guarantees, which are, arguably, more important in a production network. It is instructive to explore this difference between oblivious routing and our work with an example: Consider a traffic matrix region consisting of two matrices d_1 and d_2, and let $u_{\max}(R, d)$ be the maximum utilization when routing R is used with demand matrix d. Suppose that the optimal routing for d_1 results in utilization $u^*_{\max}(d_1) = 0.9$ and for d_2, $u^*_{\max}(d_2) = 0.4$. There are two candidate routings R_1 and R_2 with following performances: $u_{\max}(R_1, d_1) = 0.95$, $u_{\max}(R_1, d_2) = 0.7$, $u_{\max}(R_2, d_1) = 1.1$, $u_{\max}(R_2, d_2) = 0.6$. The oblivious ratio for R_1 is 1.75 (0.7/ 0.4) and for R_2 it is 1.5 (0.6/0.4). In this case, oblivious routing will select R_2, whereas R_1 is clearly preferable in a real network, since using R_1 allows us to admit demand d_1 without loss. We discuss oblivious routing and other newer related work in detail in Section III.

Solution Overview: We assume a *source-based multi-path* routing model in this paper. To describe the routing completely, for each source-destination pair, we have to specify each source-destination path, and a fraction of traffic that is sent through each path. General multi-path robust routing can be formulated as a semi-infinite programming problem. We focus on a special case where cost function is maximum link utilization and the traffic feasible region is a polytope specified by a finite set of linear constraints. Our linear constraint set derives directly from the well-understood hose and pipe model constraints.

Instead of the usual link-flow-based solution approach [16], we introduce a path-flow-based formulation. This path-based approach enables us to explicitly control some characteristics of the paths that are used for routing. For instance, we can limit maximum hops of a path, force paths to visit (or not to visit) certain links or node, or use only disjoint paths. We can also find the optimal load distribution among a given set of paths. It also gives us a simple way to augment the original path set with new paths to cope with partial network failures.

We consider both hose and pipe model constraints in our formulation. Using simulation results, we show that insertion of pipe model constraints can further improve the performance, or at least reduce sensitivity of the performance to routing parameters.

Our contributions in this paper are as follows:

1) We develop a path-flow-based polynomial sized LP and an iterative simple column generation algorithm for robust routing.
2) We consider both the hose and pipe model constraints for the traffic feasible region and study the added value of pipe model constraints.
3) We provide a framework for considering solutions that are robust to both traffic variations and failures in the network, and evaluate our solution using detailed simulations.

Roadmap: The rest of the paper is structured as follows. In section II, we describe the notation and traffic models. In section III, we review the related work. In Section IV, we first introduce a general formulation for the robust routing problem. For the linear case, with both hose and pipe model constraints, we convert the problem to a finite size LP. In Section V, using a column generation scheme, we reformulate the problem as a path-flow-based network flow problem. Based on the new formulation, we provide an iterative algorithm that updates the path set in each step and ultimately converges to the optimal solution. In Section VI, we explain how to take advantage of the path-based formulation in order to find alternate paths to cope with link failures. In Section VII, we study performance and characteristics of the algorithm on multiple tier-1 ISP topologies from Rocketfuel project. In particular, we study effectiveness and sensitivity of the solution to (1) the pipe model constraints and (2) link failures. Finally, we conclude and propose future work in Section VIII.

II. Assumptions and Notation

We consider a network $G = (V, E)$ with node set V and directed link set E. The network graph has N nodes and M links. The capacity of link l is c_l. $I(i)$ is the set of incoming links to node i and $O(i)$ is the set of outgoing links from node i. The traffic matrix element d_{ij} specifies traffic rate that network should transfer between source node i and destination node j. Set of source-destination node pairs is denoted by H and S is the cardinality of H. We assume that the traffic matrix is variable and unpredictable, however it always stays inside a region D. We specifically focus on two type of linear constraints with practical merits to characterize the traffic region:

Pipe Model Constraints: These constraints specify an upper bound $\omega_{ij} > 0$ for every entry of the traffic matrix d_{ij}, such that:

$$d_{ij} \quad \leq \quad \omega_{ij} \quad i,j = 1,\cdots,N. \tag{1}$$

The pipe model constraints can be derived from traffic profiles, service level agreements, or traffic policing mechanisms.

Hose Model Constraints: These constraints specify an upper bound η_i for total traffic emanating from a source node i, and an upper bound λ_j for total traffic being sent to a destination node j:

$$\begin{aligned} \sum_{k=1}^{N} d_{ik} \quad &\leq \quad \eta_i \quad i = 1,\cdots,N, \\ \sum_{k=1}^{N} d_{kj} \quad &\leq \quad \lambda_j \quad j = 1,\cdots,N. \end{aligned} \tag{2}$$

Hose model constraints can be derived from physical characteristics of networks such as router capacity, and total capacity of outgoing and incoming links of a node.

Our performance metric is maximum link utilization, and our goal is to find a routing that minimizes maximum link utilization for all traffic matrices in the feasible region D.

For each source-destination pair (i, j), a routing f can be defined by a set of *unit* link flow variables $f_{ij}(l)$, that specifies fraction of traffic that passes through link l. The flow variables specify a valid routing if they satisfy the flow conservation constraints:

$$\begin{aligned} \sum_{l\in O(k)} f_{ij}(l) - \sum_{l\in I(k)} f_{ij}(l) \quad &= \quad 0 \quad k \neq i,j \\ \sum_{l\in O(k)} f_{ij}(l) - \sum_{l\in I(k)} f_{ij}(l) \quad &= \quad 1 \quad k = i \end{aligned} \tag{3}$$

The set of all feasible routings which satisfy (3) for all pairs (i, j) is denoted by F. We can decompose flow variables into a set of paths. Alternatively, we can define a routing by specifying P_{ij}, a set of non-cyclic paths between each source-destination pair (i, j), and traffic rate x_p for every path p in P_{ij}. The *path based* formulation provides a valid routing if,

$$\sum_{p\in P_{ij}} x_p \quad = \quad 1 \;\; \forall \; (i,j) \in H. \tag{4}$$

Number of constraints in a path based formulation is far less than the link based formulation, however we have to introduce one variable per path per source-destination pair. We use a column generation method to avoid introducing all paths explicitly in our optimization problem. The path based formulation also enables us to explicitly control some other characteristics of the routing such as number of paths, number of hops per path and number of additional paths to cope with link failures. We will explain these characteristics in more detail later.

Utilization of link l, $u_l(f, d)$, is a function of routing f and traffic matrix d as follows,

$$u_l(f,d) = \sum_{(i,j)\in H} \frac{f_{ij}(l)d_{ij}}{c(l)}. \tag{5}$$

TABLE I
OVERVIEW OF RELATED WORK

Algorithm	Number of Traffic Matrices	Traffic Constraints	Size of LP	Formulation
Zhang et al. [25]	Finite	N/A	Finite	Link-based
Erlebach and Ruegg [8]	Infinite	Hose	Infinite	Link-based
Ben-Ameur and Kerivin [5]	Infinite	Hose and pipe	Infinite	Path-based
Kodialam et al. [16]	Infinite	Hose	Finite	Link-based
Azar et al. [4]	Infinite	No constraint	Infinite	Link-based
Applegate and Cohen [3]	Infinite	Pipe	Finite	Link-based
This paper	Infinite	Hose and Pipe	Finite	Path-based

III. RELATED WORK

Related problems on fixed robust routing for changing and unpredictable traffic matrices have been considered before. Zhang et al. [25] consider a finite number of traffic matrices and find a routing that provides good average and worst case performance. The hose model for resource management in virtual private networks (VPN) is introduced in [6], where single path and tree routing between the VPN endpoints are considered. Erlebach and Ruegg [8] consider multi-path routing for bandwidth reservation in the hose model using link-flow-based formulation. However, the proposed algorithm is based on solving an LP with infinite number of constraints. They use ellipsoid method with a finite size LP as separation oracle. Even though ellipsoid method is proved to have polynomial complexity, it is often too slow for practical purposes. The authors also propose an algorithm using a cutting-plane approach. The running time of this algorithm can be exponential in the worst case.

Ben-Ameur and Kerivin [5] also consider routing for a set of traffic matrices specified by linear constraints. In order to simplify the solution, they have used a conservative cost function. The cost function is a linear combination of maximum utilization of all links. Note that each link attains maximum utilization for a different traffic matrix, hence the links will not have their maximum utilization simultaneously in the network. To find the routing, they use an iterative approach to solve an LP with infinite number of constraints using a row generation procedure similar to [8]. Prasanna et al. [22] also consider traffic engineering and routing for traffic matrix regions specified by linear constraints. However, their objective is to find upper and lower bounds for the performance of single path *optimal* routing for traffic demands inside the feasible region.

Kodialam et al. [15] and Zhang-Shen and McKeown [26] propose a two phase routing scheme to make the maximum traffic rate between every two nodes in the network predictable and independent of traffic variations. In the first phase, a predetermined fraction α_j of the incoming traffic at any node i is sent to node j independent of the final destination of the traffic. In the second phase each node routes (relays) packets that it has received in phase 1 to their final destinations. Kodialam et al. [16] is perhaps the most relevant paper to the work presented in this paper. The authors propose polynomial size LPs to minimize maximum link utilization of two phase and direct path routing with hose model traffic constraints. Our solution is also a polynomial size LP to minimize maximum link utilization. However, we consider both hose model and pipe model traffic constraints, and therefore provide a framework to study how effective pipe model constraints are in presence of the hose model constraints. We also introduce a path-based formulation of the LP, which enables us to explicitly control some characteristics of the paths that are selected. The path-based formulation also gives us an appropriate framework to provision additional paths that we need to cope with failures in the network. Note that the path-based formulation can be applied to the two stage routing model in [16], to provide more control over characteristics of the adopted paths.

Azar et al. [4] introduce the concept of oblivious routing. Their performance metric for a routing f is *relative* and it does not give any guarantee about the absolute performance of the selected routing. For a routing f, the oblivious ratio is maximum ratio (over all possible traffic matrices) of maximum link utilization of routing f over maximum link utilization of the optimal routing for traffic matrix d. Therefore, each routing is compared to the best possible routing for each traffic matrix and the oblivious ratio represents its worst relative performance. The oblivious routing is the routing with best oblivious ratio, which means it has the best relative performance. Applegate and Cohen [3] introduce a polynomial size LP to find the oblivious routing. We use the same approach to find a polynomial size LP for robust routing. Oblivious routing is originally defined for unconstrained set of traffic matrices. It is interesting to note that *if for any traffic matrix d there is a traffic matrix $d' \in D$ such that $d = \alpha d'$ for some $\alpha > 0$, then the oblivious routing over the region D is the same as the oblivious routing over entire traffic matrix space.* Many practical traffic matrix constraints, including hose and pipe model constraints, fall into this category. Hence, the oblivious routing can not take advantage of these types of constraints.

IV. LINK-FLOW-BASED FORMULATION

Our goal is to find routing f that minimizes the maximum cost over all traffic matrices in the region D. The cost function, $cost(f,d) : F \times D \to \mathbf{R}^+$ is a non-negative real valued function of the routing f and traffic matrix d. Recall that F is the set of feasible routings and D is the set of feasible traffic matrices. This problem can be formulated as follows,

$$\begin{aligned}
&\min t \\
&s.t. \\
&f_{ij}(l) \text{ is a routing } \forall (i,j) \in H \\
&cost(f,d) \le t \quad \forall d \in D
\end{aligned} \tag{6}$$

The first constraint set is the set of unit flow conservation constraints described in (3), which guarantee that f is a feasible routing. The second set of constraints ensures that t is larger than $cost(f,d)$ for all traffic matrices in the region D. Hence, for a fixed routing f the minimum possible value for t is maximum of $cost(f,d)$ for all $d \in D$. The optimization problem finds the routing f that minimizes t, therefore the solution would be a minmax routing.

The second set of constraints contains one constraint per traffic matrix in D which results in infinite number of constraints. Therefore, this would be a Semi-Infinite Programming (SIP) problem. Depending on the structure of the cost function and traffic region different algorithms are proposed to solve this problem [13]. The most promising cases appear to be when the cost function and the traffic region are convex.

In this paper, we focus on a special case with a linear cost function and a traffic region specified by a set of linear inequalities. More specifically, we consider maximum link utilization as the cost function. This problem can be formulated as a linear programming (LP) problem.

$$\begin{aligned}
&\min t \\
&s.t. \\
&f_{ij}(l) \text{ is a routing } \forall (i,j) \in H \\
&\sum_{i,j} \frac{f_{ij}(l) d_{ij}}{c(l)} \le t \quad \forall \quad l \in L \quad \& \quad d \in D
\end{aligned} \tag{7}$$

The first set of constraint is exactly the same as in (6). From (5), the left hand side of the second constraint is utilization of link l for traffic matrix d. Therefore, the second set of constraints guarantees that every link utilization is less than t for every traffic matrix. Clearly, solution to this LP is the routing that minimizes maximum link utilization. Since there are infinite number of matrices d in the polyhedron D, the optimization problem is a *linear semi-infinite programming (LSIP) problem*.

Exchange methods are one class of numerical solutions for Semi-Infinite programming problems [13]. Instead of solving (7) with infinite number of constraints we solve a simplified version of it with finite number of constraints. In each step a new constraint corresponding to a point $d \in D$ is added to the constraint set and the LP obtained is solved to find a new routing. Then, we check if the current solution satisfies *all* constraints in (7) (even those not included in the simplified version). In order to do that, for each link l, we compute the maximum utilization for all traffic matrices in D, by solving the following LP:

$$\begin{array}{lll}
\max \sum_{i,j} \frac{f_{ij}(l) d_{ij}}{c(l)} & & \\
s.t. & & \text{Dual variables} \\
\sum_k d_{ik} \le \eta_i & i = 1, \cdots, N & r_i(l) \\
\sum_k d_{kj} \le \lambda_j & j = 1, \cdots, N & y_j(l) \\
d_{ij} \le \omega_{ij} & i,j = 1, \cdots, N & q_{ij}(l)
\end{array} \tag{8}$$

In this problem, routing (f) is fixed and we find the traffic matrix in D that causes maximum utilization. For future reference, we have also introduced dual variables for the LP in (8). If the maximum utilization computed in (8) for all links l is less than or equal to the solution of simplified (7), we are done. Otherwise, we have to add the violating links with their corresponding traffic matrices as new constraints to the simplified version of (7) and solve it again.

Note that (8) is a separation oracle for the second set of constraints in (7), hence (7) has polynomial solvability by using the Ellipsoid algorithm [10]. In our formulation, we have considered both hose and pipe model constraints. In fact, we can add any form of linear constraints to this problem.

We can use the exchange method to solve (6) if the cost function is convex. However, in that case the separation oracle would become a concave programming problem, since we have to maximize a convex function. Unfortunately concave programming problems are in general NP-hard.

A. Polynomial Size Single LP Formulation

The iterative algorithm presented in the previous section solves two separate LP problems in each step. It is desirable to combine these two problems in one single LP and solve the problem in one step. However, since one of the problems is minimization and the other is maximization it is not possible to combine them into a single LP directly. To that end, we use the dual of second optimization problem and combine it with the first one. This approach, which basically replaces infinite number of constraints in the original LP problem with dual of the separation oracle LP presented in (8) has been used before in [17] and more recently in the context of oblivious routing in [3] and link-flow-based robust routing with the hose model in [16].

For each link l, there are three sets of non-negative dual variables shown in (8): $r_i(l)$, $y_j(l)$, $q_{ij}(l)$. The dual LP is:

$$\begin{aligned}
&\min \sum_i \eta_i r_i(l) + \sum_j \lambda_j y_j(l) + \sum_{ij} \omega_{ij} q_{ij}(l) \\
&s.t. \\
&r_i(l) + y_j(l) + q_{ij}(l) \ge \frac{f_{ij}(l)}{c(l)} \\
&r_i(l), y_j(l), q_{ij}(l) \ge 0
\end{aligned} \tag{9}$$

Due to the strong duality of linear programming, the optimal objective values of (8) and (9) are equal. For a given routing f we can solve either (8) or (9) to find the maximum utilization of link l. Therefore, a routing f minimizes maximum link utilization, t over all links l and for all traffic matrices $d \in D$ if and only if it is a solution to the following LP:

$$\begin{aligned}
&\min t \\
&s.t. \\
&f_{ij}(l) \text{ is a routing} \quad \forall \ l,i,j \\
&\sum_i \eta_i r_i(l) + \sum_j \lambda_j y_j(l) + \sum_{ij} \omega_{ij} q_{ij}(l) < t \quad \forall \ l \\
&f_{ij}(l) - c(l) \left(r_i(l) + y_j(l) + q_{ij}(l) \right) \le 0 \quad \forall \ l,i,j \\
&r_i(l), y_j(l), q_{ij}(l) \ge 0 \quad \forall \ l,i,j
\end{aligned} \tag{10}$$

The first set of constraints makes sure that the solution is a feasible routing. The second set together with optimization criterion ensures that the solution minimizes maximum objective function of the dual LP (9) for all links, hence due to duality it minimizes maximum link utilization. The third set of constraints is simply constraints of the dual LP (9) that should be satisfied.

If we consider a full-mesh connected network, where every node has traffic destined to every other node, there would be $N(N-1)$ source-destination pairs in the network. In this case, in the single LP (10), there are $N^2(N-1)+MN(N-1)$ constraints in the first set of constraints, M constraints in the second set, $MN(N-1)$ constraints in the third set, and $MN(N-1)+2NM$ constraints in the fourth constraint set.

V. Path-Flow-Based Formulation

We can formulate any network flow problem using a path-based formulation based on directed path flows [2]. Even though the number of directed paths in a network grows exponentially with the network size, there are typically a few paths that carry traffic in the optimal solution. Therefore, we can start with an initial *active* path set, and use a column generation procedure to add new paths to the active path set (only if they can potentially reduce the cost). Figure 1 is a high level block diagram of the algorithm. As illustrated in the block diagram, we also remove those paths (with zero rate) from the active path set to manage number of active paths.

Besides reducing number of constraints and variables there are other advantages in a path-based formulation. The path-based formulation gives us control over the characteristics of the paths selected. For instance, we can directly control total number of paths, number of paths per source-destination pair, and number of hops per path. As another example, let us say that only a subset of nodes in the network can do sophisticated monitoring and we want every packet to visit at least one of these nodes. It is again very straightforward to impose these conditions in a path-based formulation.

The path-based formulation provides appropriate alternate routing solutions for source/link failures. Suppose we have a basic solution for the original network topology. We would like to find an alternate routing to cope with link failures. It is desirable to limit the number of new paths that are introduced in the alternate solution. If we use a link-flow-based formulation and solve the problem for the topology with failed links, there is no guarantee that the original path set is considered for routing. However, if we use a path-based formulation we can use the original path set as the initial set for the new topology, hence making sure it is considered. In this way, number of alternate paths introduced for link failures can be monitored and controlled.

The path-based formulation results in an iterative algorithm. Let P^k be the active path set in kth step, and P^k_{ij} the subset of P^k representing the paths between source-destination pair (i,j), and Π^k_l the subset of P^k passing through link l. By definition we have,

$$\sum_{p\in P^k_{ij}} x_p = 1 \quad \forall\ (i,j)\in H \tag{11}$$

$$f^k_{ij}(l) = \sum_{p\in(P^k_{ij}\cap\Pi^k_l)} x_p \quad \forall\ (i,j)\in H,\ l\in E \tag{12}$$

where $f_{ij}(l)$ is the fraction of flow from i to j that goes through link l at step k and x_p is fraction of traffic of the corresponding source-destination pair sent on path p.

We rewrite the single LP (10) in term of path rates rather than the link flow rates:

$$\begin{aligned}
&\min t\\
&s.t.\\
&\sum_{p\in P^k_{ij}} x_p = 1 \quad \forall\ i,j\\
&\sum_i \eta_i r_i(l) + \sum_j \lambda_j y_j(l) + \sum_{ij} \omega_{ij} q_{ij}(l) < t \quad \forall\ l\\
&\sum_{p\in(P^k_{ij}\cap\Pi^k_l)} x_p - c(l)\left(r_i(l)+y_j(l)+q_{ij}(l)\right) \le 0 \quad \forall\ l,i,j\\
&r_i(l), y_j(l), q_{ij}(l), x_p \ge 0 \quad \forall\ l,i,j,p
\end{aligned} \tag{13}$$

The $MN(N-1)$ flow conservation constraints in (10) are replaced with $N(N-1)$ constraints in the first constraint set in (13). Furthermore, instead of $MN(N-1)$ link flow variables, we have $|P|$ path rate variables. As we will see in simulation results, the number of active paths is much less than number of flow variables in practice.

The LP (13) finds the optimal load distribution and routing solution for the active path set P^k. In the following, we provide an algorithm to update the active path set in each step.

A. Active Path Set Update

We use a column generation method [2] to update the active path set. Suppose that we have a rate variable x_p for every directed path in the network. We know that in the optimal solution most of these variables will be zero. The main idea behind column generation is to consider only a subset of these paths in every step and insert only those paths that can reduce the cost in each step.

Using the LP terminology, at each step, we find paths (variables) with minimum reduced cost (assuming a minimization formulation). Then, if reduced cost of these paths is less than the active paths' reduced cost, we add them to the active path set. The key is to find a simple way for finding a path with minimum reduced cost. To that end, we define a set of link length parameters $w_{ij}(l)$ for link l and source-destination pair (i,j). Then, we show that reduced cost of a path between (i,j) is equal to the path length using $w_{ij}(l)$ for the link length. To that end, consider a standard LP:

$$\begin{aligned}
&\min c^T x\\
&s.t.\\
&Ax \le B\\
&x \ge 0
\end{aligned} \tag{14}$$

Suppose that we use a column generation scheme to solve this problem. At step k a subset of columns are active while others are set to zero. Let λ^k be the vector of optimal dual variables at step k. The reduced cost of every variable x_p is $c_p - A_p^T\lambda^k$, where A_p is pth column of A.

Going back to the LP (13), the path rate x_p does not appear in the cost function and $c_p = 0$. x_p is only present in the first and third set of constraints. Let α_{ij} and $\lambda_{ij}(l)$ correspond to the first and third sets of constraints dual variables. The reduced cost for path p between (i,j) is:

$$r_p = -\alpha_{ij} - \sum_{l \text{ in path } p} \lambda_{ij}(l) \tag{15}$$

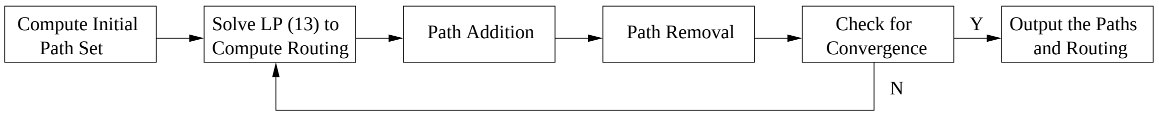

Fig. 1. Path-Flow-Based Robust Routing Algorithm

To find a path with minimum reduced cost between (i, j), since α_{ij} is independent of path p, we have to find path p^* that minimizes the second term,

$$p^* = \arg\min_p \left(-\sum_{l \text{ in path } p} \lambda_{ij}(l)\right) \tag{16}$$

It can be easily shown that the dual variables are non-positive, and hence p^* is a shortest path between (i, j) where length of link l is $w_{ij}(l) = -\lambda_{ij}(l)$. We only need to insert the shortest path if its length is less than other paths that are already active. Therefore, in every step k we compute a shortest path between every source-destination pair. Length of links are different for different source-destination pairs. If length of the computed shortest path is strictly less than current active paths, then we insert the new path to the set.

B. Path Removal

In order to control number of active paths in the problem we can also remove some paths that are not used in the solution at time step k. For each path we define path-idle-age counter. The path-idle-age is initially zero and is set to zero, whenever the corresponding path rate is non-zero. However, at time step k if the cost function is decreased, then we increment path-idle-age of those paths with zero rate. If path-idle-age of a path reaches a positive integer $C > 0$, we remove that path from active path set. Theoretically C can be any positive integer, but based on experiments, for faster convergence we set it to 5 in our simulations.

C. Convergence Criterion

The path-based formulation is an iterative algorithm, so we should have a convergence criterion. If we do not use Path Removal, we can simply iterate until no new path is added. However, practically this may turn out to be infeasible since path update procedure may keep adding paths with no or negligible incentive. Note that by adding paths the LP becomes more complex and it takes more time to find the solution.

The situation is even worse when we use path removal. Path removal may get trapped in a periodic pattern where a group of paths are added and then removed from the active path set. To avoid these problems we have a no-improvement counter that indicates number of consecutive time steps with negligible improvement in the cost function. If the no-improvement counter reaches a pre-specified threshold we decide that the algorithm is converged.

The path-flow-based iterative algorithm steps are summarized in Figure 1: We start with an initial active path set. At every step we find the optimal solution for the current active path set by solving LP (13). Next, we use the dual variables to find the shortest path for every source-destination pair and add them to the active path set, if they are shorter than active paths. Then, we remove those paths that have not been used for C consecutive time steps. We repeat these steps until convergence.

VI. Network Failures

The path based formulation provides a simple approach to cope with network failures. We first review some of the basic and desirable requirements for a failure recovery solution and then show that these desired characteristics are inherent in a path-based solution. Consider the special case of single link failures. Once a link fails, obviously those paths that go through the failed link are not functional anymore. The first natural attempt would be to redistribute traffic among active paths that are still functional. After traffic redistribution if the performance is still not acceptable, we have to add new paths to improve the performance. However, it is desirable to limit number of added new paths after failure.

Traffic redistribution among functional active paths and addition of new paths are straightforward tasks in the path-based framework. We remove the failed links from the network topology and use the functional active paths as the initial active path set. If for a source-destination pair there is no path in the initial set, we add the minimum hop path to the set for them. Obviously, if there is no path available between a source-destination pair in the network, then we can do nothing for that source-destination pair. We use the resulting path set as the initial set for the iterative path-based routing algorithm. In the first step, the algorithm redistributes traffic among paths in the initial active path set. If the performance is acceptable we do not need to add any additional path. Otherwise, we update the active path set, until we achieve the desired performance or the algorithm converges.

In contrast, using link-flow-based formulation we can find the robust routing for the new topology, but there is no direct way to redistribute the load between the functional active paths first or incrementally add new paths to improve the performance.

Furthermore, similar to the approach proposed in [9], it is possible to identify a set of critical links and additional paths that we need to introduce if they fail. These additional paths together with the initial path set are considered as the the initial path set for all link failures. In this way, we can further limit the number of additional paths that need to be introduced.

VII. Simulation on ISP Topologies

A. Topologies and Traffic Matrix Regions

For our experiments we use six topologies listed in Table II. Five topologies (except the Genuity topology) are originally from the Rocketfuel project [23]. However, we use the refined

TABLE II

FINAL NUMBER OF NODES AND LINKS IN TOPOLOGIES USED FOR SIMULATION.

Topologies (AS#)	Number of Nodes	Number of Links
Genuity (1)	23	72
Sprint (1239)	27	126
Ebone (1755)	18	66
Tiscali (3257)	28	132
Exodus (3967)	21	72
Abovenet (6461)	17	74

topologies used in [14] and provided to us by the authors of that paper. In the refined topologies to obtain approximate PoP to PoP topologies, topologies are collapsed so that nodes correspond to cities. We removed the degree one nodes (nodes connected to only one other node) from the topologies.

The Rocketfuel topologies do not have link capacities; once again, we used link capacities as assigned by [14]. The capacity assignment is based on the degree of the cities (nodes) in the network. In [14], the authors assume that nodes with high degrees are level-1 PoPs and the rest are smaller PoPs. For each topology, they detect a knee in the degree distribution of the nodes and nodes are classified into level-1 PoPs and small PoPs based on that. Links connecting level-1 PoPs have 10 Gbps capacity and the rest have 2.5 Gbps capacity. The next step is to establish the traffic matrix regions. We start with the hose model constraints, and assume that node capacity (the upper bound for the total incoming and outgoing traffic) is proportional to the total capacity of the links connected to that node. This assumption is in accordance with the study in [1]. Since links are symmetric, incoming and outgoing capacities of a node are the same. Thus node i incoming and outgoing traffic bounds are,

$$\eta_i = \lambda_i \sim \sum_{l \in O(i)} c_l \quad \forall\ i \in V \tag{17}$$

For the pipe model constraints, we assume that maximum traffic rate between two nodes is proportional to the minimum of the node capacities. The pair capacity cannot be larger than the node capacity. This assumption is also in line with the traffic models suggested in [20] and the gravity model [24]. We set the pipe model upper bound constraints ω_{ij} as follows:

$$\omega_{ij} = \frac{\min(\lambda_i, \lambda_j)}{N/\alpha} \quad \forall\ (i,j) \in H \tag{18}$$

Parameter α ranges from 1 to N in our simulations and control non-uniformity of the traffic, where N is the number of nodes in the network. For $\alpha = N$, the pipe model constraints become irrelevant and every pair's traffic can be as high as minimum of its node capacities. For $\alpha = 1$, traffic becomes very uniform and the hose model constraints become irrelevant, since summation of pipe model constraints for a node would be less than the hose model constraint. Therefore, by changing α we can study relative impact of pipe and hose model constraints. If D_α is the traffic matrix region corresponding to α, then for $\alpha_1 < \alpha_2$ we have $D_{\alpha_1} \subseteq D_{\alpha_2}$. Consequently, the routing cost (maximum link utilization) is a non-decreasing function of α.

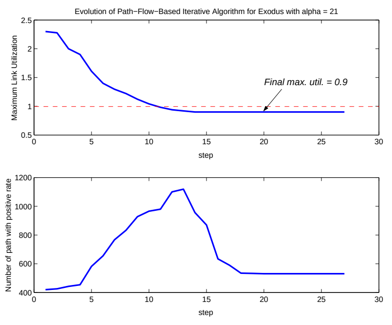

Fig. 2. Evolution of path-flow-based iterative algorithm. Maximum link utilization (top), and number of paths with non-zero rate (bottom) for Exodus topology with alpha = 21.

TABLE III

NUMBER OF PATHS DISTRIBUTION FOR SOURCE-DESTINATION PAIRS IN EXODUS NETWORK WITH ALPHA = 21.

Number of Paths	1	2	3	4
Source-Destination Pair Count	349	42	18	11

B. *Experiments and Results*

1) Primary Results: In this section, we demonstrate basic characteristics of the path-based iterative algorithm for the Exodus network with $\alpha = 21$ for the traffic matrix pipe model constraints. Since there are 21 nodes in this network, $\alpha = 21$ means that the pipe model constraints are not bounding, hence this would result in the traffic matrix region bounded only with the hose model constraint. We performed each experiment on every ISP topology; however, the Exodus network has the largest maximum link utilization in our experiments and has one of the slowest convergence rates with respect to number of iterations. For detailed results, which shows the dynamic behavior of the algorithm, we focus on the Exodus network with maximum α, and present summarized results for other experiments.

Figure 2 shows the evolution of characteristics of the path-based iterative solution as it converges to the optimal solution. The top plot is the maximum link utilization of the routing used in each step and the bottom plot is number of active paths with non-zero rate at each step. Initially, we start with minimum hop paths between source-destination pairs. As the algorithm evolves, new paths are added and some paths are removed from the optimal solution. The final solution cost (maximum link utilization) is 0.9.

For this particular run, the total number of active paths in the final solution is 531. Since there are 420 source-destination pairs, most source-destination pairs use only a single path. Table III shows the final path count distribution. In fact, more than 83% of the source-destination pairs use a single path, and the maximum number of paths for a source-destination pair is four. Therefore, in this configuration, number of additional paths required for robust routing is not significant. In later experiments, we confirm this observation more generally.

2) Effect of Pipe-model Constraints: In this set of experiments, we change the parameter α and study its impact on

TABLE IV
MAXIMUM NUMBER OF PATHS.

Topologies (AS#)	Max. Path Count	Average Max. Path Count
Genuity (1)	1007	1.99
Sprint (1239)	1474	2.10
Ebone (1755)	872	2.85
Tiscali (3257)	2490	2.29
Exodus (3967)	560	1.33
Abovenet (6461)	907	3.33

the final solution. Recall that increasing α increases the pipe model upper bounds, which results in a larger feasible traffic region. In other words, α controls how much traffic can be sent between every source-destination pairs.

Figure 3 shows the cost (maximum link utilization) for different values of α on different networks. As we expect, the cost is a non-decreasing function of α. We also observe a knee effect in the cost plots. There is a threshold value for α such that the optimal cost drops significantly if α is less than the threshold and remain constant if α is above the threshold. The threshold value for most networks is in the neighborhood of $\alpha = 3$. Recent studies have shown that internet traffic can be very non-uniform and as much as 95% of traffic can be carried by half or one third of source destination flows [20]. Roughly speaking, this means that α should be in the neighborhood of 3 in the real networks.

The inference to be drawn here is that even with non-uniform traffic matrices, rough pipe model upper bound estimates derived from traffic profiles are still useful and can have a significant impact on feasible routings and their ultimate performance. Table IV shows the maximum path count for different networks. Even though the path count depends on the network and value of α, its maximum value for all cases is reasonably low. In the last column, we have calculated maximum average number of paths per source-destination pair. In all cases the average is below 3.5; hence, robust routing does not require large numbers of alternate paths. For all networks, the cost (max. link utilization) remains constant if α crosses a threshold value. However, the solution sensitivity to the network parameters are not the same. To illustrate this we focus on the link weights. Recall that $w_{ij}(l)$ is negative of the corresponding shadow price (dual variable) for the third set of constraints in (13). We can rewrite the constraint as follows,

$$\frac{f_{ij}(l)}{c(l)} - (r_i(l) + y_j(l) + q_{ij}(l)) = 0 \tag{19}$$

Shadow prices reveal how much the optimal solution is sensitive to the variations of the constant values of the linear constraints [2]. Equivalently, for the constraint (19), shadow prices show how sensitive is the cost function to the changes in $f_{ij}(l)/c(l)$. Therefore, larger values of $w_{ij}(l)$ indicate higher sensitivity. Sensitivity is an important factor for practical systems, since for instance it is not possible to set the flow rates $f_{ij}(l)$ exactly to the optimal values and we have to check how much the solution is sensitive to the deviations from the optimal values. The maximum link weight values for the Exodus Network for $\alpha = 5, 11, 21$ are respectively $0.06, 0.11, 0.2$. The cost value for all three cases are the same. However, it is clear from the maximum link weights that shadow prices, and hence sensitivities are not the same. In fact, $\alpha = 21$ routing is more than 3 times more sensitive than $\alpha = 5$. We have observed similar sensitivity characteristics in all networks solutions. This behavior suggests that even if α is large enough such that the pipe model constraints seem irrelevant, they are still helpful in providing a less sensitive routing to the flow rate fluctuations.

3) Link Failure: In this section, we consider single link failures. (In our model, we assume links are full-duplex, and when a link fails, communication in both directions is disrupted). Our goal is to find an alternative routing after a failure, ideally minimizing the number of extra paths added. We compute the alternate routing for each failure as follows:

1) Remove the failed link from the graph.
2) Remove all paths involving the failed link.
3) For each source-destination pair with no remaining path, find a minimum hop path and add it to the path set.
4) Use the obtained path set as the initial path set for the path-based robust routing algorithm.
5) Repeat the iterative path-based algorithm until the cost function is not more than 10% above the value when all links were there, or when no new paths are added.

Figure 4 summarizes the result of running this algorithm on the Exodus network with $\alpha = 3$. Again, we selected the Exodus network since it has the highest utilization among the simulated networks, but with a lower value of α, so that the pipe model constraints are bounding too. The top plot shows the maximum link utilization of the alternate routing after each failure. The middle plot shows number of added paths for each link failure and the bottom one shows the total number of paths used for each link failure.

The cost for the Exodus topology without any failure when $\alpha = 3$ is 0.81. For 83% of single link failures, the cost of the alternate routing is less than 10% above the no failure cost. Failure of 5 critical links results in maximum utilization larger than 1 (shown by the vertical bars crossing the red 100% utilization in the figure). For 72% of link failures, the number of added paths is less than 10% of the initial number of paths. As expected, the cases with larger number of added paths coincide with the cases with high cost value. The total number of paths used for 89% of link failures is less than 500 and the maximum number of paths used is 570. There were 480 paths in the solution with no failures. Hence, even in this difficult case, the number of paths in the solution for each link failure is reasonable.

In summary, these preliminary results suggest that path-based formulation is a promising framework that can provide reasonable solution in terms of complexity and performance to compute alternate routings after link failures.

VIII. CONCLUSION

Static routing schemes have obvious and tangible practical benefits in terms of stability (no load-based fluctuations) and ease of deployment (no need for online monitoring/control). However, static schemes are useful only if they are robust to *all* variations in traffic load. We introduced a fixed-size LP

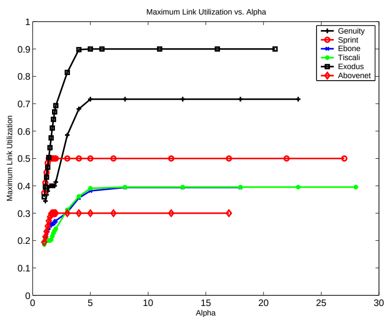

Fig. 3. Effect of pipe model constraints on robust routing performance.

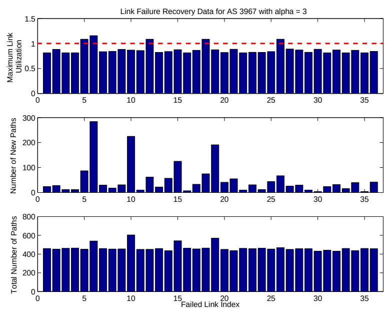

Fig. 4. Link failure recovery data for Exodus topology with alpha = 3.

formulation for multi-path source based fixed robust routing. Our scheme computes a routing that minimizes maximum cost as long as traffic is bounded within a region specified by linear constraints derived from hose and pipe model constraints. We believe our path-flow-based formulation is more useful and practical than what had been considered in earlier literature.

Path-based solution gives the network designer/operator unique and direct control over key path characteristics. For instance, our scheme can be used to limit the hop count, force paths to visit (or not to visit certain node) or use only disjoint paths. We have also described a simple mechanism to find alternate paths to cope with link failures.

Our simulation results show that robust routing generates reasonable path sets (often using less than 3 paths between source-destination pairs in average) that are immune to traffic variations. Our parameter sweeps show that as the pipe-model bounds increase, the maximum utilization increases until a threshold value is attained (as mandated by the hose constraints). However, even in the hose-model regime, the pipe-model constraints mandate how sensitive the solution is to different routing parameters.

Our experiments also lead us to the following conjecture: In a recent paper [16], authors compare robust direct path routing with two phase routing. Their simulation results suggests that the two phase routing performance is very close to the direct path routing. This result is highly counter intuitive since two phase routing sends each packet twice through the network. Some of the results that we present in this paper provides further insight. In [16] only hose model constraints are considered, therefore the original traffic matrices can be very non-uniform. The two-phase routing creates more uniform traffic patterns in the network. Therefore, we conjecture that if we consider both pipe and hose model constraints together, the performance difference between direct and two phase routing will increase. As the pipe model constraints become tighter the performance gap will become larger.

REFERENCES

[1] Abilene “http://www.abilene.internet2.edu”
[2] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, “Network FLows”, *Prentice-Hall*, New Jersey, 1993.
[3] D. Applegate, E. Cohen, “Making intra-domain routing robust to changing and uncertain traffic demands: understanding fundamental tradeoffs”, *ACM SIGCOMM’03,* pp. 313-324, Aug. 2003.
[4] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, H. Racke, “Optimal oblivious routing in polynomial time”, *Proceedings of the 35th ACM symposium on Theory of computing,* pp. 383-388, Aug. 2003.
[5] W. Ben-Ameur, H. Kerivin, “Routing of Uncertain Traffic Demands”, *Optimization and Engineering,* vol.6, no.3, pp. 283-313, Sep. 2005.
[6] N. G. Duffield, P. Goyal, A. G. Greenberg, P. P. Mishra, K. K. Ramakrishnan, J. E. van der Merwe, “A Flexible Model for Resource Managemant in Virtual Private Network”, *ACM SIGCOMM’99,* August 1999.
[7] A. Elwalid, C. Jin, S. Low, and I. Widjaja, “MATE: MPLS Adaptive Traffic Engineering,” *IEEE INFOCOM 2001*, Anchorage, Alaska, 2001.
[8] T. Erlebach, M. Ruegg “Optimal Bandwidth Reservation in Hose-Model VPNs with Multip-Path Routing”, *IEEE INFOCOM’04,* March 2004.
[9] B. Fortz, M. Thorup, “Robust Optimization of OSPF/IS-IS Weights”, *INOC’03,* 2003.
[10] B. Grotschel, L. Lovasz, A. Schriver, “Geometric Algorithms and Combinatorial Optimization”, *Springer-Verlag,* New York, 1988.
[11] T. Guven, C. Kommaredy, R. J. La, M. A. Shayman, B. Bhattacharjee, “Measurement based optimal multi-path routing,” *IEEE Infocom 2004* , Hong Kong, 2004.
[12] ILOG CPLEX, http://www.ilog.com/
[13] R. Hettich, K.O. Kortanek, “Semi-Infinite Programming: Theory, Methods, and Applications”, *SIAM Review,* vol. 35, no. 3, pp. 380-429, Sep. 1993.
[14] S. Kandula, D. Katabi, B. Davie, A. Charny, “Walking the Tightrope: Responsive Yet Stable Traffic Engineering”, *ACM SIGCOMM’05,* vol. 35, no.4, pp. 253-264, August 2005.
[15] M. Kodialam, T.V. Lakshman, S. Sengupta, “Efficient and Robust Routing of Highly Variable Traffic”, *HOT-NETS-III,* Nov. 2004.
[16] M. Kodialam, T.V. Lakshman, S. Sengupta, “Maximum Throughput Routing of Traffic in the Hose Model”, *IEEE INFOCOM’06,* April 2006.
[17] O. L. Mangasarian, “Linear and Nonlinear Separation of Patterns by Linear Programming ”, *Operations Research,* vol. 13, no.3, pp. 444-452, May-Jun. 1965.
[18] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, C. Diot, “Traffic Matrix Estimation: Existing Techniques and New Directions”, *SIGCOMM Comput. Commun. Rev.,* vol. 32, no. 4, pp. 161-174, 2002.
[19] D. Mitra, K. G. Ramakrishna, “A Case Study of Multiservice Multipriority Traffic Engineering Design”, *GLOBECOMM’99,* pp. 1077-1083, 1999.
[20] A. Nucci, A. Sridharan, N. Taft, “The Problem of Synthetically Generating IP Traffic Matrices: Initial Recommendations”, *ACM Computer Communication Review,* vol. 35, no. 3, pp. 19-32 ,July 2005.
[21] Position statements on key routing issues for the next 10 years, Workshop of Internet routing Evolution and design(WIRED), oct. 2003.
[22] G. Prasanna, A. Vishwanath, et al., “Traffic Constraints Instead of Traffic Matrices: Capabilities of a New Approach to Traffic Characterization”, *Proc. of the 18th Int. Teletraffic Congress*, 2003.
[23] N. Spring, R. Mahajan, D. Wetherall, T. Anderson, “Measuring ISP Topologies with Rocketfuel”, *IEEE/ACM Transcation on Networking,* vol. 12, no.1, pp. 2-16, Feb. 2004.
[24] M. Roughan, A. Greenberg, C. Kalmanek, M. Rumsewicz, J. Yates, Y. Zhang, “Experience in Measring Backbone Traffic Variability: Models, Metrics, Measurements and Meaning”, *Proc. of the 18th Int. Teletraffic Congress,* 2003.
[25] C. Zhang, Z. Ge, J. Kurose, Y. Liu, D. Towsley, “Optimal Routing with Multiple Traffic Matrices Tradeoff between Average and Worst Case Performance”, *IEEE ICNP’05,* pp. 215-224, Nov. 2005.
[26] R. Zhang-Shen, N. McKeown, “Designing a Predictable Internet Backbone Network”, *HOT-NETS-III,* Nov. 2004.