Optimal Oblivious Routing in Polynomial Time

Yossi Azar[*] Edith Cohen[†] Amos Fiat[*] Haim Kaplan[*] Harald Räcke[‡]

ABSTRACT

A recent seminal result of Räcke is that for any network there is an oblivious routing algorithm with a polylog competitive ratio with respect to congestion. Unfortunately, Räcke's construction is not polynomial time. We give a polynomial time construction that guarantee's Räcke's bounds, and more generally gives the true optimal ratio for any network.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Routing protocols*
F.2.0 [**Analysis of Algorithms and Problem Complexity**]: General

General Terms

Algorithms

Keywords

Oblivious routing, Ellipsoid algorithm

1. INTRODUCTION

Communication routing is obviously one of the key technological issues today. The literature includes hundreds of papers and literally dozens of problem variants on this issue. Problem variants are characterized by parameters such as packet routing *vs.* virtual circuit routing, fixed *vs.* infinite buffers, specific *vs.* general networks, probabilistic input distribution models *vs.* worst case input distributions, deterministic *vs.* randomized routing algorithms, online *vs.* offline routing algorithms, distributed *vs.* centralized algorithms, etc. Examples of some surveys on this vast body of literature are [7, 8].

In this paper we focus on the problem of online virtual circuit routing on general networks with the goal of minimizing the congestion.

Two fundamental approaches towards routing in networks are to route adaptively depending on the current loads in the network, or to route obliviously, without any knowledge of the current state of the network. Obviously, adaptive protocols may achieve reduced congestion but are harder to implement.

Raghavan and Thompson [10] designed an offline routing algorithm for routing virtual circuits on general networks and any set of demands that well approximates the lowest possible congestion. The set of routes is chosen according to the specific demands and thus their algorithm may be considered adaptive.

In the online setting, [2] design a routing algorithm that is $\log n$ competitive with respect to congestion. This algorithm routes calls based on the current congestion on the various links in the network, this can be achieved via centralized control and serializing the routing requests. [3] gave a distributed algorithm that repeatedly scans the network so as to choose the routes. This algorithm requires shared variables on the edges of the network and hence is hard to implement. Note that both the online algorithms above depend on the demands and are therefore adaptive.

The first paper to perform a worst case theoretical analysis on oblivious routing is the paper of Valiant and Brebner [12] who considered routing on specific network topologies such as the hypercube. They give an efficient randomized oblivious routing algorithm. Borodin and Hopcroft [4] and subsequently [6] have shown that deterministic oblivious routing algorithms cannot well approximate the minimal load on any non-trivial network. While the results were given in terms of packet routing, it follows from their proof that it also holds for virtual circuit routing.

In a recent paper, Räcke [9] gives the very surprising construction of a polylog competitive oblivious routing algorithm for general undirected networks. It seems truly astonishing that one can come close to minimal congestion without any information on the current load in the network. As Räcke's algorithm is oblivious, it can be trivially implemented in a distributed fashion. Räcke's algorithm is randomized, and, consequent to the work of [4], randomization

[*]School of computer science, Tel Aviv University, Tel Aviv 69978, Israel. E-mail: {azar,fiat,haimk}@cs.tau.ac.il.
Research supported in part by the Israel Science Foundation and by the IST Program of the EU.
[†]AT&T research labs, 180 park ave. Florham park, NJ, USA. Email:edith@research.att.com.
[‡]Heinz Nixdorf Institute and Department of Mathematics and Computer Science, Paderborn University, 33102 Paderborn, Germany. Email:harry@uni-paderborn.de.

STOC '03 June 9–11, 2003. San Diego, California, USA.

is required.

Although Räcke's algorithm is randomized, it can also be viewed as the randomized rounding of a deterministic multi commodity flow. We call this multi commodity flow a *routing*. Räcke's oblivious routing algorithm finds a unit flow between every pair of nodes i, j. Such a flow can be translated into a set of at most $|E|$ paths, each of which carries some fraction of the unit flow. Räcke chooses the route between i and j by choosing one of these paths with a probability equal to the flow through the path.

Räcke's main result is that for any network, there exists a routing such that for any set of demands, the maximum edge congestion using this routing, is at most polylog(n) times the optimal congestion for this specific set of demands. Unfortunately, his algorithm for producing this routing requires solving NP-hard problems, and is therefore non-polynomial. Moreover, Räcke's construction provides a uniform bound on all graphs. Hence, for any specific graph, a better routing may potentially exist, *i.e.*, one that guarantees a smaller competitive ratio w.r.t. congestion.

In this paper we give a polynomial time algorithm that produces the optimal routing for any network. In particular, this means that Räcke's construction is performed in polynomial time. In addition, we compute the optimal routing for any given network. Our construction computes the optimal routing for both directed and undirected networks. However, we also show that Räcke's polylog(n) upper bound is false for general directed graphs by providing a $\sqrt{n}$ lower bound.

Our techniques are based on linear programming with an infinite number of constraints. Thus, we use the Ellipsoid algorithm with a separation oracle [5]. Our separating oracle is itself implemented in polynomial time using a different set of linear programs.

2. PRELIMINARIES

Consider a graph $G(V, E)$, directed or undirected, with capacities $c(e) > 0$ for $e \in E$.

We use a well known reduction from undirected graphs to directed graphs that replaces each undirected edge $e = (u, v)$, with capacity $c(e)$, with the directed gadget u, x, y, v which consists of five directed edges: four directed edges $e_1 = (u, x), e_2 = (v, x), e_3 = (y, u), e_4 = (y, v)$, all of which have infinite capacity, and the directed edge $e_5 = (x, y)$ with capacity $c(e)$. This transformation preserves the property that a multi commodity flow is feasible on the undirected graph if and only if it is feasible on the directed graph.

Thus, as of this point on we only consider directed graphs.[1]

A ***multi-commodity flow*** $\mathbf{g}$ in G is defined as a solution to the system

$$\begin{array}{lrcl} \forall e \in E\ \forall i \forall j \neq i: & & & \\ & g_{ij}(e) & \geq & 0 \\ \forall k \forall i \neq k \forall j \neq k, i: & & & \\ & \sum_{e \in \mathrm{OUT}(k)} g_{ij}(e) - \sum_{e \in \mathrm{IN}(k)} g_{ij}(e) & = & 0 \end{array} \qquad (1)$$

where each set of values for $g_{ij}(e)$ for $e \in E$ defines a single-commodity flow from i to j. The demand (of commodity ij) delivered from i to j is

$$d_{ij} = \sum_{e \in \mathrm{OUT}(i)} g_{ij}(e) - \sum_{e \in \mathrm{IN}(i)} g_{ij}(e) \ .$$

The total flow induced by $\mathbf{g}$ on the edge $e \in E$ is

$$\textsc{flow}(e, \mathbf{g}) = \sum_{ij} g_{ij}(e) \ .$$

We now define the *congestion* incurred on an edge $e \in E$ by the flow $\mathbf{g}$ as the ratio of the flow on the edge to the capacity of the edge

$$\textsc{edge-cong}(e, \mathbf{g}) = \frac{\textsc{flow}(e, \mathbf{g})}{c(e)} \ .$$

Since we are interested in routing many different sets of demands in the same way, we use the notion of a ***routing from*** i ***to*** j as a flow of value 1 from i to j. We denote such a flow by f_{ij}, and denote its value on an edge $e \in E$ by $f_{ij}(e)$. We can use a routing to deliver demand of d_{ij} from i to j simply by scaling f_{ij} by a factor of d_{ij}. We shall refer to a set $\mathbf{f}$ of $n(n-1)$ routings from i to $j \neq i$ for every pair (i, j) as a ***routing***. Routings are specified by the set of linear constraints

$$\begin{array}{lrcl} \forall e \in E\ \forall i \forall j \neq i: & & & \\ & f_{ij}(e) & \geq & 0 \\ \forall i \forall j \neq i: & & & \\ & \sum_{e \in \mathrm{OUT}(i)} f_{ij}(e) - \sum_{e \in \mathrm{IN}(i)} f_{ij}(e) & = & 1 \\ \forall k \forall i \neq k \forall j \neq k, i: & & & \\ & \sum_{e \in \mathrm{OUT}(k)} f_{ij}(e) - \sum_{e \in \mathrm{IN}(k)} f_{ij}(e) & = & 0 \end{array} \qquad (2)$$

A *demand matrix* is an $n \times n$ nonnegative matrix where the diagonal entries are 0. Instead of talking directly about multi commodity flows, we will often find it convenient to talk about a pair of a demand matrix and a routing. The *flow* on an edge $e \in E$ when routing the demand matrix $\mathbf{D}$ using the routing $\mathbf{f}$ is

$$\textsc{flow}(e, \mathbf{f}, \mathbf{D}) = \sum_{ij} D_{ij} * f_{ij}(e).$$

Similarly, the *congestion* incurred on an edge $e \in E$ when routing the demand matrix $\mathbf{D}$ using the routing $\mathbf{f}$ is

$$\textsc{edge-cong}(e, \mathbf{f}, \mathbf{D}) = \frac{\textsc{flow}(e, \mathbf{f}, \mathbf{D})}{c(e)} \ .$$

The *congestion* of routing $\mathbf{D}$ using $\mathbf{f}$ is the maximum edge congestion, that is,

$$\textsc{congestion}(\mathbf{f}, \mathbf{D}) = \max_{e \in E} \textsc{edge-cong}(e, \mathbf{f}, \mathbf{D}) \ .$$

For a demand matrix $\mathbf{D}$ we denote by $\textsc{opt}(\mathbf{D})$, the minimum congestion possible for any routing. $\textsc{opt}(\mathbf{D})$ is the solution of the LP

minimize Z such that

$\mathbf{f}$ is a routing

$$\forall e \in E, \ \textsc{edge-cong}(e, \mathbf{f}, \mathbf{D}) \leq Z.$$

The variables of this LP are the $mn(n-1)$ variables $f_{ij}(e)$ which specify the routing and the minimization variable Z. Its solution constitutes the optimal routing for the particular set of demands $\mathbf{D}$. The *performance ratio* of a routing $\mathbf{f}$ on demands $\mathbf{D}$ is the ratio of $\textsc{congestion}(\mathbf{f}, \mathbf{D})$ and $\textsc{opt}(\mathbf{D})$.

[1]The results of Räcke that the congestion is polylog does not hold for digraphs, only for undirected graphs. What we do is find the best oblivious routing for any directed graph. In particular, for digraphs that are equivalent to undirected graphs the result of Räcke will hold.

We also define the *oblivious performance ratio* of a routing $\mathbf{f}$, which is the maximum performance ratio it can obtain, over all demand matrices, that is,

$$\text{OBLIV-PERF-RATIO}(\mathbf{f}) = \sup_{\mathbf{D}} \frac{\text{CONGESTION}(\mathbf{f}, \mathbf{D})}{\text{OPT}(\mathbf{D})} \ . \qquad (3)$$

We are interested in obtaining an *optimal oblivious routing* for a network G. An optimal oblivious routing minimizes the oblivious performance ratio (Equation 3), that is

$$\arg\min_{\mathbf{f}} \text{OBLIV-PERF-RATIO}(\mathbf{f}) \ ,$$

the performance ratio of an optimal oblivious routing is denote by

$$\text{OBLIV-OPT}(G) = \min_{\mathbf{f}} \text{OBLIV-PERF-RATIO}(\mathbf{f}) \ .$$

For any graph G, an upper bound of n^2 on the value of OBLIV-OPT(G) is immediate.[2] Räcke proved that for undirected graphs this value is at most polylog(n).

3. LP FORMULATION

We are now ready to state our main theorem.

THEOREM 3.1. *There is a polynomial time algorithm that for any input network G (directed or undirected) outputs a routing $\mathbf{f}$ such that* OBLIV-PERF-RATIO$(\mathbf{f})$ = OBLIV-OPT(G).

The running time of our algorithm will be polynomial in the number of nodes n, and in REP(C), the size of the bit representation of the edge capacities. If the input network is undirected we apply the transformation discussed earlier. We first observe that the problem of computing an optimal oblivious routing can be stated as an LP with $mn(n-1)+1$ variables, but infinite (continuous) number of constraints. The variables in this LP are the routing variables $\mathbf{f}$ and the minimization parameter z. The constraints of this LP are the routing constraints (Equations 2) which specify that the variables $\mathbf{f}$ constitute a routing, and for every demand matrix $\mathbf{D}$, and an edge $e \in E$, we have the constraint

$$\text{EDGE-CONG}(e, \mathbf{f}, \mathbf{D}) \leq z\text{OPT}(\mathbf{D}) \qquad (4)$$

Note that the demand matrices $\mathbf{D}$ and the respective optimal congestion values OPT$(\mathbf{D})$ are constants in this LP. We refer to this LP as BLP.

Our solution essentially solves BLP using the Ellipsoid method with a separation oracle. The separation oracle algorithm either confirms that a candidate routing $\mathbf{f}$ has a small enough OBLIV-PERF-RATIO$(\mathbf{f})$, or returns a "violated constraint" from (4) (namely, a demand matrix $\mathbf{D}$ and an edge $e \in E$, which maximizes EDGE-CONG$(e, \mathbf{f}, \mathbf{D})/OPT(\mathbf{D})$).

In order to establish, however, that the optimal value itself and the running time of the algorithm are polynomial, we need some additional arguments. In Section 5 we will show that the constraints in (4) can be pruned as follows.

[2]The optimal flow with respect to congestion is at least as bad as the worst congestion obtained by routing a single pair of source/destination. Hence, if we use the optimal flow (wrt congestion) for each pair then the total congestion is at most n^2 times the congestion of the worst case pair which results in an n^2 approximation. To find the optimal flow with respect to congestion, we solve the max flow problem for the source/destination pair, and use this routing.

LEMMA 3.2. *There exists an (exponential size) set $V(H_1)$ of demand matrices such that*

- *Each $\mathbf{D} \in V(H_1)$ has* OPT$(\mathbf{D}) = 1$ *and coefficients of size polynomial in n and* REP(C), *and*
- *It is sufficient to include in the representation of BLP only the constraints*

$$\forall e \in E,\ \forall \mathbf{D} \in V(H_1)\ \text{EDGE-CONG}(e, \mathbf{f}, \mathbf{D}) \leq z \qquad (5)$$

(all other constraints in (4) are redundant).

An important corrolary of Lemma 3.2 is that the optimal solution of BLP has a polynomial time representation, since every constraint has polynomial size and the number of variables is polynomial.

The Lemma 3.2 will also allow us to establish the polynomial time bound. First, the polynomial size of the optimal value allow us to argue that the Ellipsoid algorithm can terminate within a polynomial number of iterations. Second, the separation oracle that we will provide returns as "violated constraint" a constraint from this restricted set. In particular, the size of the constraint is polynomial in n and REP(C) and does not depend on the input routing $\mathbf{f}$. This implies that the size of the numbers (representation of the Ellipsoid produced in each iteration) remains polynomial.

In the next Section we show how BLP can be solved in polynomial time assuming Lemma 3.2 and the existence of an appropriate "separation oracle" (the orcale algorithm will be presented in Section 6.)

4. APPLYING THE ELLIPSOID TO BLP

Standard transformation allows to solve any LP by solving a polynomial number of systems of linear inequalities. In our case also, instead of working directly with BLP we will work with systems of linear inequalities LI(T) specified by a scalar value T. The variables of LI(T) is the routing $\mathbf{f}$ and the constraints are the routing constraints (Equations 2) and the constraints

$$\forall \mathbf{D} \in V(H_1)\ \forall e \in E\ \ \text{EDGE-CONG}(e, \mathbf{f}, \mathbf{D}) \leq T \ . \qquad (6)$$

It is easy to see (using the fact that REP(OBLIV-OPT(G)) is polynomial) that a binary search using a polynomial number of LI(T) instances with different values of T would allow us to obtain OBLIV-OPT(G). A binary search on $T \in [1, n^2]$ can find the smallest value of T for which LI(T) is feasible. Recall that T ranges over $[1, n^2]$ for arbitrary directed graphs and $[1, \text{polylog} n]$ for digraphs derived from undirected graphs.[3] Using this reduction, Theorem 3.1 thus follows from the following Theorem.

THEOREM 4.1. *Given a network G, capacities $c(e)$, $e \in E$, and a scalar $T > 0$, in time polynomial in n,* REP(C), *and* REP(T)[4]

[3]We comment that the Ellipsoid algorithm can be applied directly with BLP, (that is, without performing a binary search using LI(T)). In each step the oracle returns the pair $(e, \mathbf{D})$ that maximizes congestion for the current candidate routing. We use the systems LI(T) for convenience of presentation, since they are more compatible with the standard presentation of the Ellipsoid.

[4]Since REP(OBLIV-OPT(G)) is polynomial, it suffices to use T values such that REP(T) is polynomial.

- *Decide that the system LI(T) is infeasible (that is, $T <$ $\text{OBLIV-OPT}(G)$). Or,*
- *Find a routing* $\mathbf{f}$ *which solves the system LI(T).*

Our proof of Theorem 4.1 is based on applying the Ellipsoid algorithm to LI(T) using the following separation oracle (the oracle algorithm is provided in Section 6).

Separation Oracle.

- **Input:** A network G, capacities $c(e)$, and a routing $\mathbf{f}$.
- **Output:**
  - $\text{OBLIV-PERF-RATIO}(\mathbf{f})$
  - A demand matrix $\mathbf{D} \in V(H_1)$ and an edge e, such that

$$\text{OBLIV-PERF-RATIO}(\mathbf{f}) = \frac{\text{EDGE-CONG}(e, \mathbf{f}, \mathbf{D})}{\text{OPT}(\mathbf{D})}$$

If $\text{OBLIV-PERF-RATIO}(\mathbf{f}) \leq T$, then $\mathbf{f}$ is a feasible point of LI(T) and the algorithm terminates. Otherwise,

$$\text{EDGE-CONG}(e, \mathbf{f}, \mathbf{D}) \leq T$$

is a constraint of LI(T) violated by $\mathbf{f}$.

Our algorithm terminates if the separation oracle certifies that the current candidate $\mathbf{f}$ solves LI(T). Moreover, if the algorithm does not terminate after certain number of $\text{poly}(n, \text{REP}(C), \text{REP}(T))$ iterations (where $\text{REP}(C)$ is the binary representation size of $c(e)$ $e \in E$), we declare that the system LI(T) is infeasible. Correctness and polynomiality of this algorithm follow from the following two key observations. First, the polynomial bound on the number of iterations follows using the standard bounds on the size of the initial ellipsoid and the smallest "volume" of the feasible set.

The second key observation is that each iteration of the Ellipsoid algorithm can be performed in polynomial time. To see that it suffices to show that the representation of each new ellipsoid is polynomially bounded. This follows from the fact that the representation size of the "violated" constraint returned by the separation oracle is polynomial in $(n, \text{REP}(C))$; since the demand matrix $\mathbf{D}$ is in $V(H_1)$. Note that without this bound on the size of the "violated constraint" (demand matrix) returned, if the size depends on the input routing, we can be in a feedback situation where the representation of the ellipsoid grows by a polynomial factor in each iteration.

To conclude the proof of Theorem 4.1, it remains to show that the separation oracle can be implemented in polynomial time in $(n, \text{REP}(C), \text{REP}(\mathbf{f}))$ and that we can indeed restrict the constraints to demand matrices in $V(H_1)$.

5. RESTRICTING BLP

Observe that if we scale a demand matrix $\mathbf{D}$, the ratio

$$\frac{\text{EDGE-CONG}(e, \mathbf{f}, \mathbf{D})}{\text{OPT}(\mathbf{D})}$$

remains fixed. It thus suffices to use as the constraints of BLP only demand matrices that can be routed with minimum congestion equal to 1, as all other demand matrices are scaled versions of such a matrix. We denote the demand matrices $\mathbf{D}$ with $\text{OPT}(\mathbf{D}) \leq 1$ by

$$H_1 = \{\mathbf{D} | \text{OPT}(\mathbf{D}) \leq 1\} \ .$$

We thus can now consider the equivalent LP, where the constraints (4) are trimmed to include only: for all $\mathbf{D} \in H_1$ and $e \in E$

$$\text{EDGE-CONG}(e, \mathbf{f}, \mathbf{D}) \leq z \tag{7}$$

Recall that every such constraint (7) is

$$\sum_{ij} D_{ij} * f_{ij}(e) \leq zc(e) \ .$$

We now show that H_1 constitutes a polyhedron on $n(n-1)$ dimensional space, defined by a polynomial (in n) number of inequalities, with coefficients in the set $\{\pm 1, c(e)\, e \in E\}$. We then argue that we can further trim the constraints (7) to demand matrices that constitute the vertices of H_1.

The polyhedron H_1

The polyhedron H_1 is the projection of the following on the variables $\mathbf{D}$.

1. The conservation constraints that guarantee that for all i, j, $\mathbf{g}$ constitutes a flow shipping D_{ij} units from i to j:

$$\begin{array}{lrcl}
\forall i \forall j \neq i: & & & \\
 & D_{ij} & \geq & 0 \\
\forall e \in E\ \forall i \forall j \neq i: & & & \\
 & g_{ij}(e) & \geq & 0 \\
\forall i \forall j \neq i: & & & \\
 & \sum_{e \in \text{OUT}(i)} g_{ij}(e) - \sum_{e \in \text{IN}(i)} g_{ij}(e) & = & D_{ij} \\
\forall k \forall i \neq k \forall j \neq k, i: & & & \\
 & \sum_{e \in \text{OUT}(k)} g_{ij}(e) - \sum_{e \in \text{IN}(k)} g_{ij}(e) & = & 0
\end{array} \tag{8}$$

2. Constraints that say that the flow $\mathbf{g}$ has congestion at most 1.

$$\forall e \in E \quad \text{FLOW}(e, \mathbf{g}) \leq c(e) \tag{9}$$

It is not hard to verify that the feasible solutions of this system are all demands matrices $\mathbf{D}$ such that the demands can be routed by some flow with congestion of at most 1, that is, $\text{OPT}(\mathbf{D}) \leq 1$.

The set $V(H_1)$ contains the vertices of the polyhedron H_1. Since the polyhedron has $\text{poly}(n)$ constraints and variables, with coefficients of size $\text{REP}(C)$, each vertex has representation of size $\text{poly}(n, \text{REP}(C))$.

In Section 6 we will see that for any routing $\mathbf{f}$ and an edge e, $\arg\max_{\mathbf{D} \in H_1} \text{EDGE-CONG}(e, \mathbf{f}, \mathbf{D})$ always includes a vertex of H_1. Thus, we can trim all non-vertex constraints in (7).

6. SEPARATION ORACLE SOLVER

The algorithm works by solving a set of LP's. For every edge $e \in E$ in turn the oracle algorithm solves the LP OLI($e, G, \mathbf{f}$) defined as follows. The LP OLI($e, G, \mathbf{f}$) has variables D_{ij}, $g_{ij}(a)$ for every edge $a \in E$. The objective of OLI($e, G, \mathbf{f}$) is

$$\text{Maximize } \text{EDGE-CONG}(e, \mathbf{f}, \mathbf{D})$$

subject to the constraints $\mathbf{D} \in H_1$.

It is not hard to see that

$$\text{OBLIV-PERF-RATIO}(\mathbf{f}) = \max_{e \in E} \text{OLI}(e, G, \mathbf{f}) \ .$$

Our oracle algorithm first solves the $|E|$ instances of LP's OLI$(e, G, \mathbf{f})$ for $e \in E$, and then selects an edge e (and the respective $\mathbf{D}$) for which the returned value OLI$(e, G, \mathbf{f})$ is maximized.

It is well known that (at least one) of the maxima of a linear objective function over a polyhedron are obtained on a vertex of the polyhedron and that polynomial time LP algorithms can obtain such a vertex maximum. We assume that our poly time LP solver, when applied to each OLI$(e, G, \mathbf{f})$, returns a demand matrix that is a vertex of H_1.

7. DIRECTED GRAPHS

In this section we show that there are directed graphs in which the value of OBLIV-OPT(G) is large (i.e. much more than polylog(n)). Specifically, we show the following theorem.

THEOREM 7.1. *There is a directed graph G of n vertices such that* OBLIV-OPT(G) *is at least* $\Omega(\sqrt{n})$.

PROOF. Consider a graph G with $n = \binom{k}{2} + k + 1$ vertices denoted by $a_{i,j}$ for all $1 \le i < j \le k$ and b_i for $1 \le i \le k$ and a vertex t. The edges of the directed graph are all of unit capacity and are as follows: $(a_{i,j}, b_i)$ and $(a_{i,j}, b_j)$ for all $1 \le i < j \le k$ and (b_i, t) for $1 \le i \le k$. Clearly, there are exactly two paths between $a_{i,j}$ to t. One is $a_{i,j}, b_i, t$ and the other is $a_{i,j}, b_j, t$. Assume that the oblivious routing routes $p_{i,j}$ for $i < j$ on the route $a_{i,j}, b_i, t$ and $q_{i,j} = 1 - p_{i,j}$ on the route $a_{i,j}, b_j, t$. Let $l_i = \sum_{j=1}^{i-1} q_{j,i} + \sum_{i+1}^{k} p_{i,j}$. Clearly

$$\begin{aligned} \sum_{i=1}^{k} l_i &= \sum_{i=1}^{k} \sum_{j=1}^{i-1} q_{j,i} + \sum_{i=1}^{k} \sum_{i+1}^{k} p_{i,j} = \sum_{i<j} (q_{i,j} + p_{i,j}) \\ &= \binom{k}{2} \cdot 1 = k(k-1)/2 \ . \end{aligned}$$

Hence there exists l_x such that $l_x \ge (k-1)/2$. Consider the following demands. A demand of one from $a_{j,x}$ to t for all $1 \le j \le x - 1$ and a demand of one from $a_{x,j}$ to t for all $x + 1 \le j \le k$. Clearly by using the oblivious routing the load on the edge (b_i, t) is $l_x \ge (k-1)/2$. However, the optimal load is at most one since we can route the demand from $a_{j,x}$ by the route $a_{j,x}, b_j, t$ and the demand from $a_{x,j}$ by the route $a_{x,j}, b_j, t$. Since $k = \Omega(\sqrt{n})$ this completes the proof. $\square$

8. UNDIRECTED GRAPHS

A notion relevant to undirected networks is symmetry of a routing. For this discussion, replace each undirected edge e by two directed edges e' and e''. A routing $\mathbf{f}$ is *symmetric* if for all $e \in E$ and nodes i and j we have $f_{ij}(e') = f_{ji}(e'')$.

LEMMA 8.1. *When G is undirected, there is always a symmetric optimal oblivious routing.*

PROOF. Consider a lower triangular matrix L with $L_{ij} = 0$ for $j \ge i$ and $L_{ij} \ge 0$ otherwise. We refer to all demand matrices such that $D_{ij} + D_{ji} = L_{ij}$ as being in the same equivalence class. We first claim that all demand matrices $\mathbf{D}$ in the same equivalence class have the same OPT$(\mathbf{D})$. To see the claim, consider a demand $\mathbf{D}$ and a routing $\mathbf{f}$. Consider now the symmetric routing $\mathbf{f'}$ constructed as follows

$$f'_{ij}(e') = f'_{ji}(e'') = \frac{D_{ij} f_{ij}(e') + D_{ji} f_{ji}(e'')}{D_{ij} + D_{ji}} \ .$$

It is not hard to verify that the congestion of routing $\mathbf{D}$ according to $\mathbf{f}$ is the same as when routing it according to $\mathbf{f'}$. Moreover, all demand matrices in the equivalence class of $\mathbf{D}$ incur the same congestion when routed according to $\mathbf{f'}$. The claim follows.

Consider an optimal oblivious routing $\hat{\mathbf{f}}$ which is not symmetric, and consider the symmetric routing where $\overline{f}_{ij}(e') = (\hat{f}_{ij}(e') + \hat{f}_{ji}(e''))/2$. First note that the symmetric routing $\overline{\mathbf{f}}$ incurs the same congestion (and thus the same performance ratio, using the claim above) on all demand matrices that are in the same equivalence class. We next show that $\overline{\mathbf{f}}$ is at least as good as $\hat{\mathbf{f}}$, that is incurs a better worst performance ratio. To do that it is sufficient to show that for any demand matrix $\mathbf{D}$, there is a matrix $\mathbf{D'}$ in $\mathbf{D}$'s equivalence class such that

$$\text{CONGESTION}(\overline{\mathbf{f}}, \mathbf{D}) \le \text{CONGESTION}(\hat{\mathbf{f}}, \mathbf{D'}) \ .$$

(Recall that CONGESTION$(\overline{\mathbf{f}}, \mathbf{D})$ is the same for all matrices in the equivalence class.) To see that, consider an edge e. We construct the matrix $\mathbf{D}^e$ in the equivalence class that maximizes the congestion on e: that is, if $\hat{f}_{ij}(e') > \hat{f}_{ji}(e'')$ we set $\mathbf{D}^e_{ij} = L_{ij}$ and $\mathbf{D}^e_{ji} = 0$, and set the reverse otherwise. The congestion incurred by routing $\mathbf{D}$ according to $\overline{\mathbf{f}}$ on the arcs e' and e'' is at most that of routing $\mathbf{D}^e$ according to $\hat{\mathbf{f}}$. We now consider all edges $e \in E$ in turn and choose $\mathbf{D'}$ to be $\mathbf{D}^e$ with maximum congestion. $\square$

The lemma shows that when the input network is undirected, it is sufficient to search for a symmetric optimal routing (which are specified by $mn(n-1)$ rather than $2mn(n-1)$ variables).

9. EXTENSIONS

The metric we considered was minimizing edge congestion, but a closer look at our method reveals that the methodology is able to perform many different optimizations, some of them may be relevant in practice. One example is minimizing node congestion (this corresponds to router load on IP networks) which is the ratio of the total traffic traversing a node to its capacity. It is also possible to consider edge and node congestion simultaneously; to consider linear combinations of edges or nodes; to add additive factor to the congestion formula; and to limit the class of demand matrices in some ways (for example limit the sum of demands or partial sums of certain demands, require 0 demand between certain pairs), limiting dilation, and so forth. The limiting factor in the selection of the optimization function is preserving the ability to express the problem and the separation oracle using linear constraints.

A natural question is whether there exists a more efficient polynomial time algorithm for computing the optimal oblivious ratio of networks. Another interesting question is what is the optimal oblivious ratio of current AS (Autonomous System) topologies. These questions are addressed in subsequent work [1]. First, it is shown that the problem of computing the optimal oblivious routing can be solved using a single polynomial-size LP; this formulation allows for a considerably faster running times, both asymptotically and in practice. An implementation applied on ISP network topologies available from the Rocketfuel [11] project reveals that the optimal oblivious ratio for these topologies ranges between 1.4–2, a perhaps surprisingly low ratio.

10. REFERENCES

[1] David Applegate and Edith Cohen. Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs. In *Proceedings of the Proceedings of the ACM Symposium on Communications Architectures & Protocols (SIGCOMM)*, pages 313–324, 2003.

[2] James Aspnes, Yossi Azar, Amos Fiat, Serge A. Plotkin, and Orli Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *Journal of the ACM*, 44(3):486–504, 1997. Also in *Proc. 25th STOC*, 1993, pp. 623–631.

[3] Baruch Awerbuch and Yossi Azar. Local optimization of global objectives: Competitive distributed deadlock resolution and resource allocation. In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 240–249, 1994.

[4] Allan Borodin and John E. Hopcroft. Routing, merging and sorting on parallel models of computation. *Journal of Computer and System Sciences*, 30(1):130–145, 1985.

[5] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, New York, NY, USA, 1988.

[6] Christos Kaklamanis, Danny Krizanc, and Thanasis Tsantilas. Tight bounds for oblivious routing in the hypercube. In *Proceedings of the 2nd ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 31–36, 1990.

[7] Frank Thomson Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays • Trees • Hypercubes*. Morgan Kaufmann, San Mateo, CA, 1992.

[8] Stefano Leonardi. On-line network routing. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms: The State of the Art*, volume 1442 of *LNCS*, pages 242–267. Springer, 1998.

[9] Harald Räcke. Minimizing congestion in general networks. In *Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 43–52, 2002.

[10] Prabhakar Raghavan and Clark D. Thompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

[11] Neil T. Spring, Ratul Mahajan, and David Wetherall. Measuring ISP topologies with rocketfuel. In *Proceedings of the ACM Symposium on Communications Architectures & Protocols (SIGCOMM)*, pages 133–145, 2002.

[12] Leslie G. Valiant and Gordon J. Brebner. Universal schemes for parallel communication. In *Proceedings of the 13th ACM Symposium on Theory of Computing (STOC)*, pages 263–277, 1981.