

# Oblivious Routing for the $L_p$ -norm

Matthias Englert\*

Department of Computer Science and DIMAP

University of Warwick

Coventry, UK

Matthias.Englert@warwick.ac.uk

Harald Räcke\*

Department of Computer Science and DIMAP

University of Warwick

Coventry, UK

H.Raecke@warwick.ac.uk

**Abstract**— Gupta et al. [13] introduced a very general multi-commodity flow problem in which the cost of a given flow solution on a graph  $G = (V, E)$  is calculated by first computing the link loads via a load-function  $\ell$ , that describes the *load* of a link as a function of the flow traversing the link, and then aggregating the individual link loads into a single number via an aggregation function  $\text{agg} : \mathbb{R}^{|E|} \rightarrow \mathbb{R}$ .

In this paper we show the existence of an oblivious routing scheme with competitive ratio  $O(\log n)$  and a lower bound of  $\Omega(\log n / \log \log n)$  for this model when the aggregation function  $\text{agg}$  is an  $L_p$ -norm.

Our results can also be viewed as a generalization of the work on approximating metrics by a distribution over dominating tree metrics (see e.g. [4], [5], [8]) and the work on minimum congestion oblivious routing [20], [14], [21]. We provide a convex combination of trees such that routing according to the tree distribution approximately minimizes the  $L_p$ -norm of the link loads. The embedding techniques of Bartal [4], [5] and Fakcharoenphol et al. [8] can be viewed as solving this problem in the  $L_1$ -norm while the result of Räcke [21] solves it for  $L_\infty$ . We give a single proof that shows the existence of a good tree-based oblivious routing for any  $L_p$ -norm.

For the Euclidean norm, we also show that it is possible to compute a tree-based oblivious routing scheme in polynomial time.

**Keywords**-oblivious routing; metric embeddings; norm;

## 1. INTRODUCTION

We analyze a general version of a multi-commodity flow problem as introduced by Gupta et al. [13]. In this problem we are given a graph  $G = (V, E)$  with  $|V| = n$  nodes, and a collection of  $R$  routing requests  $\mathcal{R} = \langle R_i = (s_i, t_i, d_i, k_i) : i \in \{1, \dots, R\} \rangle$ , where a request  $R_i$  consists of a source-target pair  $(s_i, t_i) \in V \times V$  in the graph, a demand  $d_i$  specifying the amount of traffic to be sent, and a type  $k_i \in \{1, \dots, K\}$  that specifies some abstract *type of the routing request* like e.g. a desired quality of service level. The task is to satisfy each request  $R_i$  by sending a flow of type  $k_i$  and value  $d_i$  from source  $s_i$  to target  $t_i$  in the network, thereby establishing a multi-commodity flow.

The *cost* of this multi-commodity flow solution is calculated as follows. Let for each edge  $e \in E$ ,  $f_k(e)$  denote the amount of flow of type  $k$  that is sent along  $e$ . Then the *load*

$L(e)$  of the edge is given via a load function  $\ell : \mathbb{R}^K \rightarrow \mathbb{R}$  by

$$L(e) = \ell([f_1(e), \dots, f_K(e)]) .$$

The *cost* of the solution is calculated by aggregating the individual edge loads into a single number via an aggregation function  $\text{agg} : \mathbb{R}^{|E|} \rightarrow \mathbb{R}$ , i.e.,

$$\text{cost} = \text{agg}([L(e_1), \dots, L(e_{|E|})]) .$$

This very general framework gives rise to optimization problems of different flavors depending on the choice of the load function, the aggregation function, and the set of different flow types. When choosing  $K = 1$ ,  $\ell = \text{id}$ , and  $\text{agg} = \max$  it is a maximum concurrent multicommodity flow problem. When choosing  $K = 1$ ,  $\text{agg} = \sum$ , and  $\ell$  as a concave function it becomes a buy-at-bulk or rent-or-buy network design problem. When choosing  $K = \binom{n}{2}$ ,  $\ell = \max$ ,  $\text{agg} = \sum$ , and when each  $(s, t)$ -pair gets a unique type we obtain a fractional Steiner forest problem where one needs to buy fractional edge-capacities such that each pair appearing in a request is connected in the resulting graph.

Gupta et al. [13] developed oblivious algorithms for these types of problems. An oblivious routing algorithm must choose its routing paths independent of the traffic in the network. This means that the routing path (or the flow) chosen for a request between a source  $s$  and a target  $t$  may only depend on the structure of  $G$ , on the cost-functions  $\text{agg}$  and  $\ell$ , on the pair  $(s, t)$ , and, in case randomization is allowed, on some random input. One can view an oblivious routing algorithm as specifying a unit flow between every source-target pair before any routing requests are known. Given a request between a pair  $s$  and  $t$ , this request is then routed according to the pre-computed  $(s, t)$ -flow, i.e., in order to determine the flow for the request, the pre-computed flow is simply scaled to meet the request's demand. Due to their simplicity oblivious algorithms can be implemented very efficiently in a distributed environment as they base routing decisions only on local knowledge.

The main results of Gupta et al. [13] deal with the design of oblivious routing algorithms when the load-function  $\ell$  is either a monotone sub-additive function or a monotone norm,

\*The authors acknowledge the support of DIMAP (the Centre for Discrete Mathematics and its Applications) during this work. The first author is supported by EPSRC grant EP/F043333/1.

and the aggregation function is either the  $\sum$  or the max-operator. They obtain polylogarithmic competitive ratios for these scenarios.

In this work we extend these results to the case where the aggregation function is an  $L_p$ -norm and  $\ell$  is a monotone norm, by proving the existence of an oblivious routing scheme with competitive ratio  $O(\log n)$  for these cost-measures. This also implies a competitive ratio of  $O(\log^p n)$  for the scenario with  $K = 1$ ,  $\ell(f(e)) = (f(e))^p$ , and  $\text{agg} = \sum$ . The latter result is of particular interest in the context of modeling the latency of traffic networks (see e.g. [23], [22], [9]). In this line of work every edge in the network possesses a latency function describing the latency incurred by all traffic on the edge as a function of the edge flow. The goal is to minimize the average latency of the users which corresponds to minimizing  $\sum_e f(e) \cdot \text{latency}(f(e))$ . If the latency functions are linear (see [9] for a natural scenario of linear latencies in TCP/IP communication networks), this corresponds to our cost model with  $K = 1$ ,  $\ell = f(e)^2$ , and  $\text{agg} = \sum$ . More generally if the latency function is given by  $\text{latency}(f(e)) = (f(e))^{p-1}$  we obtain an  $O(\log^p n)$ -competitive oblivious routing algorithm for minimizing average latency.

Note that we assume that the latency function is the same on every edge. Our results also hold for the slightly more general model where each edge has a weight  $w_e$  and the load of an edge is  $L(e) = w_e \ell(f(e))$ , but we cannot accommodate different latency functions in general.

Our results can also be viewed as a generalization of the work on approximating metrics by a distribution over dominating tree metrics (see e.g. [4], [5], [8]) and the work on minimum congestion oblivious routing [20], [14], [21]. We provide a convex combination of trees such that routing according to the tree distribution approximately minimizes the  $L_p$ -norm of the link loads. The embedding techniques of Bartal [4], [5] and Fakcharoenphol et al. [8] can be viewed as solving this problem in the  $L_1$ -norm while the result of Räcke [21] solves it for  $L_\infty$ . We give a single proof that shows the existence of a good tree-based oblivious routing for any  $L_p$ -norm.

### 1.1. Previous Work

As mentioned in the previous section the model that we analyze in this paper has been introduced in a slightly different form by Gupta et al. [13]. The main result of their paper is an  $O(\log^2 n)$ -competitive oblivious *integral* routing algorithm for the case that  $\text{agg} = \sum$  and  $\ell$  is a monotone sub-additive function. Integral means that routing requests are not allowed to use a fractional flow but have to be served by sending the traffic along a single path between the corresponding end-points. As a second result they give a competitive ratio of  $O(\log^2 n \log \log n)$  for the case that  $\text{agg} = \max$  and  $\ell$  is a monotone norm. They also show that the additional requirement that  $\ell$  is a norm is necessary as there exist sub-additive functions for which no oblivious

routing algorithm is  $o(n^{1/4})$ -competitive if the aggregation function is the max-operator.

One specific feature of their algorithms is that they are not only traffic-oblivious but also (partially) *function-oblivious*, i.e., they do not depend on the actual load function but the same algorithm works for any load function, provided that it is a monotone sub-additive function or a monotone norm, respectively. The routing schemes we develop have the same property.

The concept of function-oblivious routing was introduced in a paper by Goel and Estrin [11] who give an  $O(\log n)$ -approximation to the single-target case for arbitrary concave load functions. This has recently been improved to an  $O(1)$ -approximation by Goel and Post [12].

For the case  $K = 1$ ,  $\ell = \text{id}$ ,  $\text{agg} = \sum$  our problem reduces to minimizing the total traffic in the network which can trivially be obtained by shortest path routing (which is oblivious). However, when the solution has to be tree-based (i.e., the routing is done according to a convex combination of trees) then it essentially boils down to the problem of embedding metrics into a distribution over dominating tree metrics. This concept has been introduced by Bartal [4], [5] and has later been improved by Fakcharoenphol, Rao, and Talwar [8] who give a competitive ratio of  $O(\log n)$ , which is optimal. This means there is a lower bound of  $\Omega(\log n)$  for our problem with  $K = 1$ ,  $\ell = \text{id}$ ,  $\text{agg} = \sum$ , if we are aiming for a tree-based solution.

If we do not require a tree-based solution, then the fractional Steiner tree variant ( $K = \binom{n}{2}$ ,  $\ell = \max$ ,  $\text{agg} = \sum$ ) of our problem gives a lower bound of  $\Omega(\log n)$  due to Imase and Waxman [16],<sup>1</sup> which holds even if we do not require the algorithm to be oblivious but merely require that it be an online algorithm. Similarly, the min-congestion routing version ( $K = 1$ ,  $\ell = \text{id}$ ,  $\text{agg} = \max$ ) also gives a lower bound of  $\Omega(\log n)$  already in the online scenario, which has been independently proven by Bartal and Leonardi [6] and Maggs et al. [18].

Another lower bound is due to Chen, Roughgarden, and Valiant [7]. They show that when  $K = 1$  and  $\text{agg} = \sum$ , then already the offline problem (all demands known in advance) is difficult if function-obliviousness w.r.t. an arbitrary sub-additive load function  $\ell$  is required. They show that there exists a graph, a set of demands, and two sub-additive cost-functions such that it is not possible to route close to optimal w.r.t to both cost-functions simultaneously as required for function-obliviousness. They obtain a lower bound of  $\Omega(\sqrt{\log n})$ . However, this lower bound does not extend to the case where the load function  $\ell$  is a norm.

The problem of designing algorithms with the goal of minimizing the  $L_p$ -norm of the link loads has been mostly studied in the context of scheduling or load balancing on

<sup>1</sup>Their lower bound is phrased in an integral model but it is straightforward to verify that it holds for the fractional model, as well.

parallel machines, which in our scenario corresponds to a network consisting of parallel links.

Awerbuch et al. [2] design an online algorithm for the *unrelated machine model* in which the size of a job depends on the machine it is placed on. They obtain a competitive ratio of  $O(p)$  and their algorithm can also be extended to work in a routing scenario. In [1] Avidor et al. give a constant factor approximation for the load balancing problem on identical machines in the  $L_p$ -norm. Finally, Azar et al. [3] present an approximation algorithm that obtains a ratio of 2 for all norms *simultaneously* in the *restricted assignment model* (in which jobs are of equal size but may have different subsets of machines they can be assigned to).

Harsha et al. [15] consider the problem of designing oblivious routing algorithms for the  $L_2$ -norm or equivalently for the sum-of-squares cost-measure. They obtain a competitive ratio of  $O(\log n)$  and  $O(\log^2 n)$ , respectively, for the restricted case that all routing requests are directed to a single target. However, their approach is very much restricted to the  $L_2$ -norm and the single-target case and cannot be generalized to either the multicommodity case or an arbitrary  $L_p$ -norm.

In [17] Lawler and Narayanan consider the problem of minimizing all  $L_p$ -norms simultaneously with an oblivious routing algorithm. They present an algorithm that obtains a competitive ratio of  $O(\sqrt{n})$  and they show that this is best possible. This negative result shows that it is not possible to design oblivious routing algorithms with a good competitive ratio that are *function-oblivious* w.r.t. the aggregation function  $\text{agg}$ . Therefore, in all our results we will assume that we know the aggregation function, and our algorithms will only be function oblivious w.r.t. the load function  $\ell$ .

### 1.2. Our Results

We extend the results of Gupta et al. [13] to the case where the aggregation function is an  $L_p$ -norm by showing the following theorem.

**Theorem 1.** *For  $p \geq 1$  there exists an oblivious routing scheme that is  $O(\log n)$ -competitive when the aggregation function  $\text{agg}$  is an  $L_p$ -norm and the load function  $\ell$  is a monotone norm.*

Our routing schemes are *tree-based*, i.e., they are induced by a convex combination of trees like the results of Bartal [4], [5], Fakcharoenphol et al. [8] and Räcke [21]. This means that they can be used for much more than just routing. Any communication problem with the goal of minimizing the  $L_p$ -norm of the link-loads can be solved approximately in  $G$  by first solving it on trees and then mapping the solution into  $G$ . Compare [4], [5], [8], [20], [21] for a description of this technique in the  $L_1$ -norm and  $L_\infty$ -norm, respectively.

To obtain our result, we first show that our problem can be reduced to the problem of finding a good tree-based routing matrix that has a small operator norm (see Section 3). Then we define a two player zero-sum game with the property that

the payoff in a pure Nash equilibrium essentially corresponds to the competitive ratio of the best tree-based oblivious routing scheme (see Section 4). A result by Räcke [21] can be used to show that the payoff in a pure equilibrium is bounded by  $O(\log n)$  (see Lemma 4). Finally, we show that, for any  $L_p$ -norm, the game has a pure Nash equilibrium (see Lemma 5).

As mentioned in the introduction, Theorem 1 also gives results for minimizing average latency when the latency function on an edge  $e$  has the form  $\text{latency}(e) = (f(e))^{p-1}$ , where  $f(e)$  describes the total flow along the edge ( $K = 1, \ell = \text{id}, \text{agg} = \|\cdot\|_p^p$  which is equivalent to  $K = 1, \ell = (f(e))^p, \text{agg} = \sum$ ). The resulting competitive ratio is  $O(\log^p n)$ . In Section 7 we show that this cannot be improved significantly by proving the following theorem.

**Theorem 2.** *Let  $p > 1$ ,  $K = 1$ ,  $\ell(f(e)) = (f(e))^p$ , and  $\text{agg} = \sum$ . There is no oblivious algorithm that obtains a competitive ratio of  $o(\log^p n / (\log \log n)^p)$  for this scenario.*

By showing that, for  $p = 2$ , Theorem 1 can be made constructive, we demonstrate that our approach can also lead to polynomial time algorithms computing good routing schemes. In particular, in Section 6, we present an algorithm that computes an oblivious routing scheme with a logarithmic competitive ratio for the Euclidean norm in polynomial time.

## 2. FORMAL DEFINITION OF THE MODEL

We are given an undirected unweighted network  $G = (V, E)$ , for which we consider the following multi-commodity flow problem. Let  $[K] := \{1, \dots, K\}$  denote a set that describes the possible *types* that a flow between two nodes of  $G$  may have (we assume that  $K$  is polynomial in  $n := |V|$ ). For every  $k \in [K]$  and  $(s, t) \in V \times V$  a demand-value  $d_{s,t}^k$  describes the amount of traffic of type  $k \in [K]$  that has to be sent from source node  $s \in V$  to target node  $t \in V$ . A solution of this multi-commodity flow problem consists of a collection of  $\binom{n}{2}K$  flows in the network.

The *cost* of a given solution  $f$  to the flow problem is calculated as follows. Let for an edge  $e \in E$ ,  $f_k(e)$  denote the total traffic of type  $k \in [K]$  sent along  $e$ . We call the vector  $\vec{f}(e) := [f_1(e), \dots, f_K(e)]$  the *flow vector along  $e$  for solution  $f$* . The *load*  $L_f[e]$  induced on edge  $e$  is given by  $L_f[e] := \ell(\vec{f}(e))$ , where the *load function*  $\ell$  is a monotone norm. Let  $\vec{L}_f := [L_f[e_1], \dots, L_f[e_{|E|}]]$  denote the *load vector* induced by solution  $f$ . The *total cost* of the solution is given by an *aggregation function*  $\text{agg} : \mathbb{R}^{|E|} \rightarrow \mathbb{R}$  which takes the load of the various edges as input and outputs the *cost* of the entire flow, i.e.,  $\text{cost}(f) = \text{agg}(\vec{L}_f)$ . In this paper we assume that the aggregation function is an  $L_p$ -norm.

An *oblivious routing algorithm* has to make its routing decisions a priori without knowing the demands in the network. It specifies for every source target pair  $(s, t) \in V \times V$  a unit flow from  $s$  to  $t$  in the network. Suppose that for an edge  $e$ ,  $g_{s,t}(e)$  denotes the fraction of the unit flow, that is sent

between  $s$  and  $t$ , on  $e$ . For all  $k \in [K]$  the demand  $d_{s,t}^k$  will be routed by simply scaling the unit flow  $\langle g_{s,t}(e) : e \in E \rangle$  by  $d_{s,t}^k$ .

We introduce the following notation: For a type  $k \in K$  we introduce an  $\binom{n}{2}$ -dimensional demand vector  $\vec{d}_k$  that contains one entry for every node pair describing the amount of type  $k$  demand between this pair. The total demand is then specified by an  $\binom{n}{2} \times K$  dimensional demand-matrix  $D$  whose  $k$ -th column vector is  $\vec{d}_k$ .

The unit flow between pair  $(s, t)$  of an oblivious algorithm OBL is denoted with  $\text{OBL}_{s,t}$  and encoded as an  $|E|$ -dimensional column vector. We combine the flows into an  $|E| \times \binom{n}{2}$  dimensional routing matrix OBL whose columns correspond to the vectors  $\text{OBL}_{s,t}$  (we will usually identify an oblivious algorithm with its routing matrix, i.e., we call both the algorithm and its routing matrix OBL).

Routing a given demand matrix  $D$  with an oblivious algorithm OBL results in the matrix  $\text{OBL} \cdot D$ . The row  $\vec{f}_{\text{obl}}(e) := (\text{OBL} \cdot D)_e$  corresponding to edge  $e$  contains the  $K$ -dimensional flow vector along  $e$  that is induced by routing the demands in  $D$  with the oblivious algorithm. The cost of this solution is  $\text{cost}(\vec{f}_{\text{obl}}) = \text{agg}([\ell(\vec{f}_{\text{obl}}(e_1)), \dots, \ell(\vec{f}_{\text{obl}}(e_{|E|}))])$ .

An oblivious algorithm OBL is *tree-based* if its routing matrix is induced by a convex combination of trees. This means the following. Define a *decomposition tree*  $T_G$  for  $G$  as a rooted tree whose leaf nodes correspond to the nodes in the graph  $G$  (i.e., there is a bijection between leaf nodes in  $T_G$  and nodes in  $V$ ). An embedding  $(m_V, m_E)$  of  $T_G$  into  $G$  consists of two functions: the function  $m_V$  for mapping nodes of  $T_G$  to nodes of  $G$  and the function  $m_E$  for mapping edges of  $T_G$  to paths in  $G$ .  $m_V$  maps each leaf node of  $T_G$  to its corresponding graph node (as defined by the bijection), and  $m_E$  maps an edge  $(u, v)$  of  $T_G$  to a path between  $m_V(u)$  and  $m_V(v)$ .

*Routing according to  $T_G$*  means that a routing path between  $(x, y) \in V \times V$  is sent between the corresponding leaf nodes in  $T_G$  and then a path in  $G$  is constructed by mapping the  $x - y$  path in  $T_G$  to a path in  $G$  via the embedding functions. A *tree-based* routing matrix is a matrix whose flows can be interpreted as being sent according to a convex combination of decomposition trees.

Let for a demand matrix  $D$ ,  $f_{\text{opt}}^D$  denote a solution to the corresponding flow problem that minimizes the cost, and let  $f_{\text{obl}}^D$  denote the solution obtained by a given oblivious algorithm OBL. The goal is to find an oblivious routing algorithm with as low a *competitive ratio* as possible, where the *competitive ratio* is defined as

$$\max_D \left\{ \frac{\text{cost}(f_{\text{obl}}^D)}{\text{cost}(f_{\text{opt}}^D)} \right\} = \max_D \left\{ \frac{\|\vec{L}_{f_{\text{obl}}^D}\|_p}{\|\vec{L}_{f_{\text{opt}}^D}\|_p} \right\}.$$

### 3. REDUCTION TO MATRIX NORMS

The competitive ratio of an oblivious algorithm is given by the formula  $\max_D \{ \|\vec{L}_{\text{obl}}^D\|_p / \|\vec{L}_{\text{opt}}^D\|_p \}$ , where  $\vec{L}_{\text{obl}}^D$  and  $\vec{L}_{\text{opt}}^D$  denote the load vector of the oblivious and the optimal solution, respectively. The following theorem allows to considerably simplify this expression for tree based oblivious algorithms. Let for an oblivious routing matrix OBL,  $\overline{\text{OBL}}$  denote the  $|E| \times |E|$  matrix obtained from OBL after deleting all columns corresponding to node pairs that are not connected by an edge.

**Lemma 3.** *Assume that the optimum load vector for a given demand matrix  $D$  is  $\vec{L}_{\text{opt}}$ . Then the load vector  $\vec{L}_{\text{obl}}$  induced by a tree-based oblivious algorithm OBL fulfills  $\vec{L}_{\text{obl}} \preceq \overline{\text{OBL}} \cdot \vec{L}_{\text{opt}}$ . This in particular means that the cost of the oblivious algorithm is at most  $\|\overline{\text{OBL}} \cdot \vec{L}_{\text{opt}}\|_p$ .*

*Proof:* Suppose that the optimum flow solution for a given demand matrix  $D$  uses the flow vector  $\vec{f}_{\text{opt}}(e)$  along edge  $e$ . We first show how to change the demand matrix in such a way that a) the optimum load vector does not change, and b) the load of a tree-based oblivious algorithm does not decrease on any edge.

For an edge  $e = (u, v) \in E$  we set  $D'_{(u,v)} := \vec{f}_{\text{opt}}(e)$ , this means the demand along  $e$  is equal to the flow vector along  $e$  in the optimum solution (pairs of nodes that are not connected by an edge don't get any demand). The optimum solution for demand  $D'$  will be the same as for the original demand  $D$ . This holds because  $f_{\text{opt}}$  is clearly feasible for the new demand (One can obtain  $f_{\text{opt}}$  by simply sending each demand along the corresponding edge at which it appears), and because any "improved" solution  $f'_{\text{opt}}$  for  $D'$  would give rise to a better solution for demand  $D$ .

For a tree-based oblivious algorithm OBL the flow-vector  $\vec{f}_{\text{obl}}^{D'}(e) = (\text{OBL} \cdot D')_e$  on an edge  $e$  that results from routing demand  $D'$ , dominates the vector  $\vec{f}_{\text{obl}}^D(e) = (\text{OBL} \cdot D)_e$  that results from routing demand  $D$ . To see this consider a flow path  $p = (s, v_1, v_2, \dots, v_s, t)$  that carries flow of some type  $k$  between nodes  $s$  and  $t$  in the original flow solution, and assume for the time being that the oblivious routing matrix OBL consists of a single tree (instead of a convex combination of trees). When given  $D$  as input the flow that the optimum solution routes along path  $p$  will be routed by the oblivious algorithm on a shortest tree path between the leaf nodes corresponding to  $s$  and  $t$ , respectively. However, if one is given  $D'$  then the demand that originally was between  $s$  and  $t$  and of which OPT routed some fraction along path  $p$ , will be split into several "demand-pieces" one for every edge of  $p$ . Consequently, the oblivious algorithm has to route in its tree, first from the leaf node of the source  $s$  to the leaf node of  $v_1$ , then to the leaf node of  $v_2$ , and so on, until finally routing to the leaf node corresponding to the target  $t$ . In particular, this path will contain every edge on the shortest path between  $s$  and  $t$  in the tree. This means that the

traffic corresponding to flow path  $p$  does not decrease on any tree-edge. Since, this holds for any tree it holds for a convex combination of trees. This shows that  $\vec{f}_{\text{obl}}^D(e) \preceq \vec{f}_{\text{obl}}^{D'}(e)$  for every edge  $e$ .

The above discussion shows that we can assume without loss of generality that we have a demand matrix  $D$  such that the demand  $D_e$  along edge  $e$  is equal to  $\vec{f}_{\text{opt}}(e)$  and that pairs that are not connected do not have demand. Hence,

$$\begin{aligned} \vec{L}_{\text{obl}}(e) &= \ell(\text{OBL}_e \cdot D) = \ell(\sum_{e'} \text{OBL}_{e,e'} \cdot D_{e'}) \\ &\leq \sum_{e'} \text{OBL}_{e,e'} \cdot \ell(D_{e'}) = \sum_{e'} \text{OBL}_{e,e'} \cdot \vec{L}_{\text{opt}}(e') \\ &= \overline{\text{OBL}}_e \cdot \vec{L}_{\text{opt}} \end{aligned}$$

as desired, where the second to last equality uses the fact that the load function  $\ell$  is the same for every edge. Also note that the inequality in the above expression uses the fact that  $\ell$  is a norm, i.e.,  $\ell(\vec{a} + \vec{b}) \leq \ell(\vec{a}) + \ell(\vec{b})$  and  $\ell(\lambda \cdot \vec{a}) = \lambda \cdot \ell(\vec{a})$  (actually we only need  $\ell(\lambda \cdot \vec{a}) \leq \lambda \cdot \ell(\vec{a})$  but this already implies equality). ■

In order to design a tree-based oblivious algorithm with a good competitive ratio (say  $O(\log n)$ ) Lemma 3 tells us that one needs to find a *tree-based* routing matrix OBL such that

$$\max_{\text{load-vector } \vec{L}} \left\{ \frac{\|\overline{\text{OBL}} \cdot \vec{L}\|_p}{\|\vec{L}\|_p} \right\} \leq O(\log n) .$$

This basically asks whether there is a tree-based matrix  $\overline{\text{OBL}}$  with a small induced matrix  $p$ -norm  $\|\overline{\text{OBL}}\|_p = \max_{\vec{L}} \left\{ \frac{\|\overline{\text{OBL}} \cdot \vec{L}\|_p}{\|\vec{L}\|_p} \right\} \leq O(\log n)$ . The following lemma shows a necessary condition for this to be possible, namely it shows that for any load-vector there exists an oblivious routing matrix that performs well w.r.t. this load-vector.

**Lemma 4.** *For any given load vector  $\vec{L}$  there exists a tree-based routing matrix OBL such that  $\|\overline{\text{OBL}} \cdot \vec{L}\|_p \leq O(\log n) \cdot \|\vec{L}\|_p$ .*

*Proof:* Racke [21] gives for any undirected graph  $G$  a tree-based oblivious routing scheme that is  $O(\log n)$ -competitive with respect to congestion. Translating his scenario into our model means that the aggregation function is  $\text{agg} = \max$  the load function is  $\ell = \text{id}$  and  $K = 1$ .

Let OBL be this algorithm designed for a graph  $G'$  that is isomorphic to  $G$  but in which an edge  $e$  is equipped with a capacity  $\text{cap}(e) = L[e]$ . Now, suppose that a demand vector is given that specifies a demand of  $L[e]$  along every edge  $e$  in  $G'$ . Routing this demand directly along the edge gives a load of  $\text{cap}(e)^{-1} L[e] = 1$  on every edge and, hence, a congestion of 1. Routing the demand with the oblivious algorithm gives a load-vector of  $\overline{\text{OBL}} \cdot \vec{L}$ , and since the oblivious algorithm is  $O(\log n)$ -competitive we have  $(\overline{\text{OBL}} \cdot \vec{L})_e \leq O(\log n) \cdot L[e]$  for every edge  $e$ . ■

#### 4. THE GAME

In the following we define a continuous two player zero-sum game  $\mathcal{G}_{\|\cdot\|_p}$  that is closely related to the oblivious routing problem. Let the strategy set  $S_M$  of the first player (called the matrix player) be the set of tree-based oblivious routing matrices. The strategy set  $S_L := \{\vec{L} \in \mathbb{R}_{\geq 0}^{|E|} \mid \|\vec{L}\|_p \leq 1\}$  of the second player (called the vector player) is the set of positive  $|E|$ -dimensional vectors of length at most one (measured in the  $L_p$ -norm).

The payoff function  $f : S_M \times S_L \rightarrow \mathbb{R}$  is given by  $f(M, \vec{L}) \mapsto \|(M + M_\varepsilon) \vec{L}\|_p$  and has to be minimized by the matrix player and maximized by the vector player, where  $M_\varepsilon$  denotes an  $|E| \times |E|$  matrix in which every entry is  $\varepsilon > 0$ . Note that both strategy sets are compact and that the payoff function is continuous. Therefore, this defines a continuous game which, due to Glicksberg's [10] generalization of the Kakutani fixed point theorem, has a Nash equilibrium in mixed strategies. The following lemma shows that the game also possesses a pure Nash equilibrium.

**Lemma 5.** *For  $p \geq 1$  the game  $\mathcal{G}_{\|\cdot\|_p}$  possesses a pure Nash equilibrium.*

*Proof:* Let  $\rho$  and  $\sigma$  be mixed strategies played by the matrix and the vector player, respectively, where both  $\rho$  and  $\sigma$  are regular probability measures. The expected value of the payoff function is  $\iint f(M, \vec{L}) d\rho(M) d\sigma(\vec{L})$ . Due to the triangle inequality, we have, for every vector  $\vec{L}$  and regular probability measure  $\rho$ ,

$$\begin{aligned} \int f(M, \vec{L}) d\rho(M) &= \int \|(M + M_\varepsilon) \vec{L}\|_p d\rho(M) \\ &\geq \left\| \left( \int M d\rho(M) + M_\varepsilon \right) \vec{L} \right\|_p \\ &= f \left( \int M d\rho(M), \vec{L} \right) . \end{aligned}$$

This means playing the pure strategy  $\int M d\rho(M)$  does not worsen the payoff for the matrix player *regardless* of the strategy of the vector player. Hence, the game also possesses an equilibrium in which the matrix player plays a pure strategy.

In the following we assume that the matrix player plays the pure strategy  $M'$  in a Nash equilibrium. Theorem 6 in Section 5 shows that there is a unique vector  $\vec{L}^*$  maximizing the payoff function  $f(M, \vec{L}) = \|(M + M_\varepsilon) \vec{L}\|_p$  for a fixed matrix  $M$ . Let  $\vec{L}'$  denote the unique vector that maximizes  $f(M', \vec{L})$ . Clearly,  $\sigma(\{\vec{L}'\}) = 1$ , as otherwise the vector player could increase the expected payoff by switching her strategy to  $\vec{L}'$ , which contradicts the notion of an equilibrium. We conclude that the game is in equilibrium if the two players play  $M'$  and  $\vec{L}'$ , respectively, as their pure strategies.

The above argument only works for  $p > 1$  as Theorem 6 does not hold for  $p = 1$ . Nevertheless it is easy to see that the game  $\mathcal{G}_{\|\cdot\|_1}$  still possesses a pure Nash equilibrium. ■

**Theorem 1.** For  $p \geq 1$ , there exists an oblivious routing scheme that is  $O(\log n)$ -competitive when the aggregation function is the  $L_p$ -norm, and the load function  $\ell$  is a monotone norm.

*Proof:* From Lemma 5 we know that the game  $\mathcal{G}_{\|\cdot\|_p}$  has a pure Nash equilibrium. Let  $M$  and  $\vec{L}$  denote the pure strategy played in this equilibrium by the matrix player and the vector player, respectively. The fact that  $\vec{L}$  is a best response implies  $f(M, \vec{L}) \geq f(M, \vec{L}')$  for all  $\vec{L}'$ . From Lemma 4 we know that there exists a routing matrix  $\widehat{M}$  with  $\|\widehat{M} \cdot \vec{L}\|_p \leq O(\log n)$ . Since  $M$  is a best response we have that  $f(M, \vec{L}) \leq f(\widehat{M}, \vec{L}) = \|(\widehat{M} + M_\varepsilon)\vec{L}\|_p \leq \|\widehat{M}\vec{L}\|_p + \|M_\varepsilon\vec{L}\|_p \leq O(\log n)$ , where the last inequality follows by setting  $\varepsilon := 1/|E|$  and observing that  $\|M_\varepsilon\vec{L}\|_p \leq \varepsilon \cdot |E| \cdot \|\vec{L}\|_p \leq 1$ . Hence, we obtain that for all  $L'$

$$\|M\vec{L}'\|_p \leq f(M, \vec{L}') \leq f(M, \vec{L}) \leq O(\log n) .$$

This shows that  $\|M\|_p \leq O(\log n)$  and gives rise to an oblivious routing scheme with competitive ratio  $O(\log n)$ . ■

## 5. UNIQUENESS

In this section we show that for a given  $|E| \times |E|$  matrix  $M = (m_{ij})_{i,j \in \{1, \dots, n\}}$  with strictly positive entries, and any  $p > 1$ , the vector  $\vec{x}^*$  maximizing the function  $f : \mathbb{R}_{\geq 0}^{|E|} \setminus \{\vec{0}\} \rightarrow \mathbb{R}$ ,  $f(\vec{x}) = \|M\vec{x}\|_p^p / \|\vec{x}\|_p^p$  is unique up to scalar multiplication. This fact is used in Lemma 5. We first show that every critical point of the function (a point with gradient  $\vec{0}$ ) is a local maximum.

Assume that  $\vec{x}$  is a critical point. Define the functions  $g(\vec{x}) = \|M\vec{x}\|_p^p$  and  $h(\vec{x}) = \|\vec{x}\|_p^p$ . Then  $\frac{\partial}{\partial x_i} g = M_i^t \vec{\nabla} h(M\vec{x})$ , and  $\frac{\partial}{\partial x_i} h(\vec{x}) = px_i^{p-1}$ . The partial derivative  $\frac{\partial}{\partial x_i} f$  is then given by

$$\frac{\partial}{\partial x_i} f(\vec{x}) = p \frac{\sum_s m_{si} (M_s \vec{x})^{p-1} \cdot \|\vec{x}\|_p^p - x_i^{p-1} \cdot \|M\vec{x}\|_p^p}{\|\vec{x}\|_p^{2p}} . \quad (1)$$

Since  $\vec{x}$  is a critical point we know that the gradient  $\vec{\nabla} f(\vec{x}) = \vec{0}$ . We can calculate the second partial derivatives  $\frac{\partial^2}{\partial x_i \partial x_j} f$  under the condition that all the first partial derivatives are 0. The second partial derivatives are given by

$$\frac{\partial^2}{\partial^2 x_i} f(\vec{x}) = p(p-1) \left( \frac{\sum_s m_{si}^2 (M_s \vec{x})^{p-2} \cdot \|\vec{x}\|_p^p}{\|\vec{x}\|_p^{2p}} - \frac{x_i^{p-2} \cdot \|M\vec{x}\|_p^p}{\|\vec{x}\|_p^{2p}} \right)$$

and for  $i \neq j$  by

$$\frac{\partial^2}{\partial x_i \partial x_j} f(\vec{x}) = p(p-1) \frac{\sum_s m_{si} m_{sj} (M_s \vec{x})^{p-2}}{\|\vec{x}\|_p^p} .$$

Without loss of generality we may assume that  $\|\vec{x}\| = 1$  and  $\|M\vec{x}\| = 1$  (by scaling  $\vec{x}$  and  $M$ , respectively) which

simplifies the second partial derivatives to

$$\frac{\partial^2}{\partial^2 x_i} f(\vec{x}) = p(p-1) \sum_s m_{si}^2 (M_s \vec{x})^{p-2} - x_i^{p-2} \quad \text{and}$$

$$\frac{\partial^2}{\partial x_i \partial x_j} f(\vec{x}) = p(p-1) \sum_s m_{si} m_{sj} (M_s \vec{x})^{p-2} .$$

Let  $H_f = \left( \frac{\partial^2}{\partial x_i \partial x_j} \right)_{i,j \in \{1, \dots, n\}}$  denote the Hessian matrix of  $f$ . Then for a direction  $\vec{\varepsilon}$ ,  $\vec{\varepsilon}^t H_f(\vec{x}) \vec{\varepsilon}$  is the second order term in the approximation of  $f(\vec{x} + \vec{\varepsilon})$  via the Taylor series expansion of  $f$ . We show that this term is strictly negative unless  $\vec{\varepsilon} = \lambda \vec{x}$ . This means, that in the neighborhood of  $\vec{x}$ ,  $f(\vec{x})$  is decreasing, and hence  $\vec{x}$  is a local maximum. We get

$$\frac{\vec{\varepsilon}^t H_f(\vec{x}) \vec{\varepsilon}}{p(p-1)} = \sum_{(i,j)} \sum_s m_{si} m_{sj} (M_s \vec{x})^{p-2} \varepsilon_i \varepsilon_j - \sum_i x_i^{p-2} \varepsilon_i^2 . \quad (2)$$

The fact that the gradient at  $\vec{x}$  is  $\vec{0}$  combined with Equation 1 provides the following equation for  $x_i^{p-2}$ :

$$x_i^{p-2} = \frac{1}{x_i} \sum_s m_{si} (M_s \vec{x})^{p-1}$$

$$= \frac{1}{x_i} \sum_s m_{si} (\sum_j m_{sj} x_j) (M_s \vec{x})^{p-2} .$$

Note that since the gradient at  $\vec{x}$  is  $\vec{0}$  and all entries of  $M$  are strictly positive, Equation 1 gives that  $x_i \neq 0$  for every  $i$ . Plugging the above into Equation 2 gives

$$\frac{\vec{\varepsilon}^t H_f(\vec{x}) \vec{\varepsilon}}{p(p-1)} = \sum_{(i,j)} \sum_s m_{si} m_{sj} (M_s \vec{x})^{p-2} \varepsilon_i \varepsilon_j$$

$$- \sum_{(i,j)} \sum_s m_{si} m_{sj} \frac{x_j}{x_i} (M_s \vec{x})^{p-2} \varepsilon_i^2$$

$$= \sum_s (M_s \vec{x})^{p-2} \left( \sum_{\{i,j\}: i \neq j} 2m_{si} m_{sj} \varepsilon_i \varepsilon_j - \sum_{\{i,j\}: i \neq j} m_{si} m_{sj} \left( \frac{x_j}{x_i} \varepsilon_i^2 + \frac{x_i}{x_j} \varepsilon_j^2 \right) \right) ,$$

where the equality follows since the contribution of an ordered pair  $(i, i)$  to the left sum and the right sum are the same. The contribution for an unordered pair  $\{i, j\}$  to the bracketed sum in the last expression is

$$m_{si} m_{sj} \cdot (2\varepsilon_i \varepsilon_j - \frac{x_j}{x_i} \varepsilon_i^2 - \frac{x_i}{x_j} \varepsilon_j^2)$$

$$= -m_{si} m_{sj} x_i x_j \cdot \left( \frac{\varepsilon_i}{x_i} - \frac{\varepsilon_j}{x_j} \right)^2 \leq 0 .$$

Note that this expression is less or equal to zero and that it becomes zero if and only if  $\frac{\varepsilon_i}{x_i} = \frac{\varepsilon_j}{x_j}$ . Since, this holds for all pairs we get that the total sum is less or equal to zero and becomes zero if and only if  $\vec{\varepsilon} = \lambda \vec{x}$  for some  $\lambda$ .

**Theorem 6.** Let  $p > 1$  and let  $M$  denote a matrix with strictly positive entries. Then the function  $f : S^{n-1} \rightarrow \mathbb{R}$ ,  $f(\vec{x}) \mapsto \|M\vec{x}\|_p / \|\vec{x}\|_p$  has a unique maximum.

*Proof:* We restricted the function  $f$  to the positive orthant of the unit sphere (which is equivalent to normalizing our positive vectors  $\vec{x}$  to  $\|\vec{x}\|_2 = 1$ ). Then the above result shows that any critical value of this function is a *strict* local maximum, i.e., for a critical value  $\vec{x}$ ,  $\vec{x}$  is the unique maximum in an  $\varepsilon$ -neighborhood around  $\vec{x}$ . This implies the theorem since if there exist two maxima, there also exists another critical point (usually a saddle point) which will have a partial second derivative that is non-negative and this gives a contradiction. ■

In fact, we can conclude the following slightly more general statement which will be useful in Section 6:

**Lemma 7.** *There is no upper level set  $L_\alpha := \{\vec{x} \mid f(\vec{x}) \geq \alpha\}$  with more than one component.*

## 6. COMPUTING TREE BASED ROUTING MATRICES FOR THE EUCLIDEAN NORM

In this section we show how to make our main theorem constructive for the case  $p = 2$ , that is, we give an algorithm that computes a tree-based oblivious routing scheme that is  $O(\log n)$  competitive if the aggregation function is the Euclidean norm. The algorithm runs in polynomial time, however, we did not attempt to optimize the degree of the polynomial.

**Theorem 8.** *There exists a polynomial time algorithm that computes an oblivious routing scheme with competitive ratio  $O(\log n)$  when the aggregation function is the  $L_2$ -norm and the load function  $\ell$  is a monotone norm.*

In order to prove the theorem we need to show how to compute a tree-based routing matrix OBL with  $\|\widehat{\text{OBL}}\|_2 \leq O(\log n)$ . We will exploit the fact that due to Lemma 4 we can, for any given vector  $\vec{x}$ , efficiently compute a tree-based matrix  $M$  with  $\|M\vec{x}\| \leq O(\log n)$ .

Furthermore, for any given matrix  $M$  we can efficiently approximate a unit vector  $\vec{x}$  that maximizes  $\|M\vec{x}\|_2$  (an approximation with an additive error of at most  $1/(2|E|^2)$  in every component is sufficient for our purpose). This follows since the vector maximizing  $\|M\vec{x}\|_2$  is an eigenvector of  $M^t M$  and efficient algorithms are known to compute eigenvectors of symmetric matrices (see e.g. [19]).

All entries of a routing matrix lie between 0 and 1. In the following we assume for simplicity that the entries of a routing matrix lie in the interval  $[1/|E|, 1 + 1/|E|]$  instead. Analogously to Section 4, this can be achieved by performing all operations on the matrix  $M + M_{1/|E|}$  instead of  $M$ , where  $M_{1/|E|}$  denotes an  $|E| \times |E|$  matrix whose entries are equal to  $1/|E|$ . Note that this change worsens the operator norm of the matrix, and therefore the competitive ratio, only by an additive constant of 1.

Define  $f(M, \vec{x}) := \|M\vec{x}\|_2 / \|\vec{x}\|_2$ . Our goal is to find a routing matrix  $M$  such that, for every load vector  $\vec{x}$ ,  $f(M, \vec{x}) = O(\log n)$ . For this, we start with an arbitrary tree-based routing matrix  $M_{(0)}$ . In an iteration  $i$ , we first compute

the unique unit vector  $\vec{x}_{(i-1)}^*$  maximizing  $f(M_{(i-1)}, \vec{x})$ , where the uniqueness follows from Theorem 6. If the value  $f(M_{(i-1)}, \vec{x}_{(i-1)}^*)$  is less than  $c \cdot \log n$ , for some suitable constant  $c$ , we have found the desired matrix and the algorithm terminates. Otherwise we apply Lemma 4 and compute a routing matrix  $\widehat{M}_{(i-1)}$  with  $f(\widehat{M}_{(i-1)}, \vec{x}_{(i-1)}^*) \leq \frac{c}{2} \cdot \log n - 2$  and choose  $M_{(i)} := \lambda \cdot \widehat{M}_{(i-1)} + (1 - \lambda) \cdot M_{(i-1)}$ , where the right choice of  $\lambda$  will be addressed in the proof of Lemma 9.

The following lemma shows that the function value  $f(M_{(i)}, \vec{x}_{(i)})$  decreases substantially in each iteration.

**Lemma 9.** *For each iteration  $i$  of the above algorithm we have*

$$f(M_{(i)}, \vec{x}_{(i)}^*) \leq f(M_{(i-1)}, \vec{x}_{(i-1)}^*) - \frac{1}{\text{poly}(|E|)}.$$

*Proof:* Let  $N_\alpha(\vec{x}) := \{\vec{x}' \mid \|\vec{x} - \vec{x}'\|_\infty \leq \alpha\}$  denote the  $\alpha$ -neighborhood around  $\vec{x}$  measured in the  $L_\infty$ -norm. Note that, if for a routing matrix  $M$  and a vector  $\vec{x}$ ,  $f(M, \vec{x}) \leq \frac{c}{2} \cdot \log n - 2$ , then we have, for every vector  $\vec{x}' \in N_{1/|E|^2}(\vec{x})$ ,  $f(M, \vec{x}') \leq \frac{c}{2} \cdot \log n$ .

We choose

$$\alpha := \left( \frac{1}{2 \cdot (|E| + 1)} \right)^{12} \quad \text{and} \quad \lambda := \frac{1}{8} \left( \frac{1}{2 \cdot (|E| + 1)} \right)^{31}.$$

If the algorithm does not terminate after iteration  $i - 1$  we have  $f(M_{(i-1)}, \vec{x}_{(i-1)}^*) \geq c \cdot \log n$  and therefore we can conclude, for every unit vector  $\vec{x} \in N_\alpha(\vec{x}_{(i-1)}^*)$ ,

$$\begin{aligned} f(M_{(i)}, \vec{x}) &\leq \lambda f(\widehat{M}_{(i-1)}, \vec{x}) + (1 - \lambda) f(M_{(i-1)}, \vec{x}) \\ &\leq \lambda \cdot \frac{c}{2} \log n + (1 - \lambda) f(M_{(i-1)}, \vec{x}_{(i-1)}^*) \\ &\leq f(M_{(i-1)}, \vec{x}_{(i-1)}^*) + \lambda \left( \frac{c}{2} \log n - c \cdot \log n \right) \\ &\leq f(M_{(i-1)}, \vec{x}_{(i-1)}^*) - \frac{1}{\text{poly}(|E|)}, \end{aligned}$$

where the second step follows because  $\widehat{M}_{(i-1)}$  was computed such that  $f(\widehat{M}, \vec{x}) \leq \frac{c}{2} \log n$  for every  $\vec{x} \in N_{1/|E|^2}(\vec{x}_{(i-1)}^*)$ .

To derive a similar inequality for vectors outside of the  $\alpha$ -neighborhood of  $\vec{x}_{(i-1)}^*$  we make use of the following claim:

**Claim 10.** *Let  $M$  be a routing matrix and  $\vec{x}^*$  be the unit vector maximizing  $f(M, \vec{x})$ , then, for every unit positive vector  $\vec{x}' \notin N_\alpha(\vec{x}^*)$ ,*

$$f(M, \vec{x}') \leq f(M, \vec{x}^*) - \frac{1}{8} \left( \frac{1}{2 \cdot (|E| + 1)} \right)^{30}.$$

*Proof:* Let  $M$  be a routing matrix and  $\vec{x}^*$  be the normalized vector maximizing  $f(M, \vec{x}) := \|M\vec{x}\|_2 / \|\vec{x}\|_2$ . In the following we prove that for every normalized vector  $\vec{x}'$  on the border of the  $\alpha$ -neighborhood of  $\vec{x}^*$ , that is,  $\|\vec{x}' - \vec{x}^*\|_\infty = \alpha$ ,

$$f(M, \vec{x}')^2 \leq f(M, \vec{x}^*)^2 - \frac{1}{8} \left( \frac{1}{2 \cdot (|E| + 1)} \right)^{28}.$$

Due to Lemma 7, this implies for every normalized positive vector  $\vec{x}' \notin N_\alpha(\vec{x}^*)$

$$f(M, \vec{x}')^2 \leq f(M, \vec{x}^*)^2 - \frac{1}{8} \left( \frac{1}{2 \cdot (|E| + 1)} \right)^{28}.$$

Since  $a^2 \leq b^2 - c$  implies  $a \leq b - c/b^2$  for  $c > 0$  and  $b > 2$ , we can then conclude

$$\begin{aligned} f(M, \vec{x}') &\leq f(M, \vec{x}^*) - \frac{1}{8} \left( \frac{1}{2 \cdot (|E| + 1)} \right)^{28} \frac{1}{f(M, \vec{x}^*)^2} \\ &\leq f(M, \vec{x}^*) - \frac{1}{8} \left( \frac{1}{2 \cdot (|E| + 1)} \right)^{30}, \end{aligned}$$

which completes the proof of the claim.

We start by stating the following claim about the value of the Hessian  $H_{f^2}$  of  $f^2$  in the neighborhood of  $\vec{x}^*$  (who's almost straightforward proof is omitted).

**Claim 11.** *For every positive vector  $\vec{y} \in N_\alpha(\vec{x}^*)$  and every vector  $\vec{\varepsilon}$  with the property that  $\vec{\varepsilon}$  contains at least one positive and one strictly negative entry,*

$$\vec{\varepsilon}^t H_{f^2}(M, \vec{y}) \vec{\varepsilon} \leq - \left( \frac{\|\vec{\varepsilon}\|_\infty}{2} \right)^2 \left( \frac{1}{2 \cdot (|E| + 1)} \right)^4.$$

Now let  $\vec{x}' \in N_\alpha(\vec{x}^*)$  be a normalized vector on the border of the  $\alpha$ -neighborhood of  $\vec{x}^*$ . From the Taylor expansion of  $f^2$  at  $\vec{x}^*$  it follows that if, for every vector  $\vec{y} \in N_\alpha(\vec{x}^*)$ ,  $(\vec{x}^* - \vec{x}')^t H_{f^2}(M, \vec{y})(\vec{x}^* - \vec{x}') \leq -\gamma$ ,  $f(M, \vec{x}')^2 \leq f(M, \vec{x}^*)^2 + \vec{\nabla} f(M, \vec{x}^*)^2 - \gamma/2 = f(M, \vec{x}^*)^2 - \gamma/2$ . This implies Claim 10 since, due to Claim 11,

$$\begin{aligned} (\vec{x}^* - \vec{x}')^t H_{f^2}(M, \vec{y})(\vec{x}^* - \vec{x}') &\leq - \left( \frac{\alpha}{2} \right)^2 \left( \frac{1}{2 \cdot (|E| + 1)} \right)^4 \\ &= - \frac{1}{4} \left( \frac{1}{2 \cdot (|E| + 1)} \right)^{28}. \end{aligned}$$

For any unit vector  $\vec{x} \notin N_\alpha(\vec{x}_{i-1}^*)$ , Claim 11 gives

$$\begin{aligned} f(M_{(i)}, \vec{x}) &\leq \lambda f(\widehat{M}_{(i-1)}, \vec{x}) + (1 - \lambda) f(M_{(i-1)}, \vec{x}) \\ &\leq \lambda(|E| + 1) + f(M_{(i-1)}, \vec{x}^*) \\ &\quad - \frac{1}{8} \left( \frac{1}{2 \cdot (|E| + 1)} \right)^{30} \\ &\leq f(M_{(i-1)}, \vec{x}_{i-1}^*) - \frac{1}{\text{poly}(|E|)}, \end{aligned}$$

due to our choice of  $\lambda$ .  $\blacksquare$

Note that the above proof of Lemma 9 still goes through if we compute  $\widehat{M}_{(i-1)}$  only with respect to a vector approximating  $\vec{x}_{(i-1)}^*$  within an additive error of  $1/(2|E|^2)$  in each component.

**Lemma 12.** *The algorithm above terminates after  $\text{poly}(|E|)$  steps.*

*Proof:* The lemma follows directly from Lemma 9 and the fact that  $f(M, \vec{x}) \leq (|E| + 1)$  for any routing matrix  $M$  and vector  $\vec{x}$ , width  $\|\vec{x}\|_p \leq 1$ , and in particular,  $f(M_{(0)}, \vec{x}_{(0)}^*) \leq |E| + 1$ .  $\blacksquare$

## 7. LOWER BOUND

In this section we give a lower bound on the competitive ratio of any oblivious routing algorithm.

**Theorem 2.** *Let  $p > 1$ ,  $K = 1$ ,  $\ell = \text{id}$ , and  $\text{agg} = \|\cdot\|_p^p$ . There is no oblivious algorithm that obtains a competitive ratio of  $o(\log^p n / (\log \log n)^p)$ .*

*Proof-sketch:* We construct a random graph  $G = (V, E)$  on  $n$  vertices by choosing edges independently with probability  $q = \Theta(\log^c n/n)$ , for a parameter  $c$  to be chosen later. For such a graph it is straightforward to show via Chernoff bounds that any cut  $(S, V \setminus S)$  will have a number of edges crossing it that is close to the expected value  $q \cdot |S| \cdot |V \setminus S|$  with high probability.

In the following assume that the constant in the definition of  $q$  is chosen appropriately such that any cut  $(S, V \setminus S)$  in  $G$  has at least  $\log^c n / n \cdot |S| \cdot |V \setminus S|$  crossing edges and that any single vertex has at most  $O(\log^c n)$  adjacent edges.

Now any two vertices have at least  $\log^c(n)$  edge-disjoint path of length at most  $O(\log n)$  between them. Furthermore, most node-pairs (at least 50%) are at distance at least  $\Omega(\log n / (2c \log \log n))$  as a node can have at most  $(O(\log^c n))^d$  nodes within distance at most  $d$ . From this it also follows that in expectation for at least 50% of the edges it holds that after removing the edge the distance between the two endpoints is at least  $\Omega(\log n / (2c \log \log n))$ . Let  $E' \subset E$  denote the subset of edges of  $G$  that have this property and assume that  $|E'| \geq \frac{1}{4}|E|$  (which will hold with high probability).

Assume you are given an oblivious routing algorithm OPT that for an edge  $e = (s, t)$  in  $E'$  routes at least half of the demand between  $s$  and  $t$  along this edge. Then the cost of the oblivious algorithm when this is the only demand in the network is at least  $1/2$ . However, an optimal algorithm could distribute this demand evenly among the  $\log^c(n)$  edge disjoint path that exist between  $s$  and  $t$ . This would result in a cost of  $O(\log n) \cdot \log^c n \cdot (1/\log^c n)^p = O(\log n) \cdot (1/\log^c n)^{p-1}$ , which is smaller than  $1/\log^p n$  for  $c \geq (p+1)/(p-1)$ . This means in this case the oblivious algorithm does not obtain a good competitive ratio.

Now assume that the oblivious algorithm schedules for every edge in  $E'$  at most half of the demand of the edge along the edge. Then this oblivious algorithm induces a poor routing if all edges in  $E'$  have a demand of 1. This holds because the described traffic pattern can be routed optimally by simply routing each demand directly along the corresponding edge, resulting in a cost of at most  $|E'|$ . The oblivious algorithm creates a total traffic of at least  $\frac{1}{2} \cdot |E'| \cdot \Omega(\log n / (2c \log \log n))$ . Even if this traffic were

divided evenly among all edges, the average edge load would be  $\Omega(\log n / (2c \log \log n))$  resulting in a cost of  $\Omega(|E'| \cdot \log^p n / (\log \log n)^p)$ , which shows that also in this case the oblivious algorithm has a bad competitive ratio. ■

#### ACKNOWLEDGMENTS

We would like to thank Peter Allen, Anupam Gupta, Mike Paterson, and Christian Sohler for helpful discussions.

#### REFERENCES

- [1] A. Avidor, Y. Azar, and J. Sgall, "Ancient and new algorithms for load balancing in the  $L_p$  norm," *Algorithmica*, vol. 29, no. 3, pp. 422–441, 2001.
- [2] B. Awerbuch, Y. Azar, E. F. Grove, M.-Y. Kao, P. Krishnan, and J. S. Vitter, "Load balancing in the  $L_p$  norm." in *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science (FOCS)*, 1995, pp. 383–391.
- [3] Y. Azar, L. Epstein, Y. Richter, and G. J. Woeginger, "All-norm approximation algorithms," *Journal of Algorithms*, vol. 52, no. 2, pp. 120–133, 2004.
- [4] Y. Bartal, "Probabilistic approximations of metric spaces and its algorithmic applications," in *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science (FOCS)*, 1996, pp. 184–193.
- [5] —, "On approximating arbitrary metrics by tree metrics," in *Proceedings of the 30th ACM Symposium on Theory of Computing (STOC)*, 1998, pp. 161–168.
- [6] Y. Bartal and S. Leonardi, "On-line routing in all-optical networks," *Theoretical Computer Science*, vol. 221, no. 1-2, pp. 19–39, 1999, also in *Proc. 24th ICALP*, 1997, pp. 516–526.
- [7] H.-L. Chen, T. Roughgarden, and G. Valiant, "Designing networks with good equilibria," in *Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2008, pp. 854–863.
- [8] J. Fakcharoenphol, S. B. Rao, and K. Talwar, "A tight bound on approximating arbitrary metrics by tree metrics," in *Proceedings of the 35th ACM Symposium on Theory of Computing (STOC)*, 2003, pp. 448–455.
- [9] E. J. Friedman, "Genericity and congestion control in selfish routing," in *Proceedings of the 43rd Annual IEEE Conference on Decision and Control (CDC)*, 2004, pp. 4667–4672.
- [10] I. Glicksberg, "A further generalization of the kakutani fixed point theorem, with application to nash equilibrium points," *Proceedings of the American Mathematical Society*, vol. 3, no. 1, pp. 170–174, 1952.
- [11] A. Goel and D. Estrin, "Simultaneous optimization for concave costs: Single sink aggregation or single source buy-at-bulk," in *Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2003, pp. 499–505.
- [12] A. Goel and I. Post, "An oblivious  $O(1)$ -approximation for single source buy-at-bulk," in *Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS)*, 2009, to appear.
- [13] A. Gupta, M. T. Hajiaghayi, and H. Räcke, "Oblivious network design," in *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006, pp. 970–979.
- [14] C. Harrelson, K. Hildrum, and S. B. Rao, "A polynomial-time tree decomposition to minimize congestion," in *Proceedings of the 15th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2003, pp. 34–43.
- [15] P. Harsha, T. P. Hayes, H. Narayanan, H. Räcke, and J. Radhakrishnan, "Minimizing average latency in oblivious routing," in *Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2008, pp. 200–207.
- [16] M. Imase and B. M. Waxman, "Dynamic steiner tree problem," *SIAM Journal on Discrete Mathematics*, vol. 4, no. 3, pp. 369–384, 1991.
- [17] G. F. Lawler and H. Narayanan, "Mixing times and  $L_p$  bounds for oblivious routing," in *Proceedings of the 5th Workshop on Analytic Algorithms and Combinatorics (ANALCO)*, 2009, pp. 66–74.
- [18] B. M. Maggs, F. Meyer auf der Heide, B. Vöcking, and M. Westermann, "Exploiting locality for networks of limited bandwidth," in *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science (FOCS)*, 1997, pp. 284–293.
- [19] P. Rabinowitz, *A First Course in Numerical Analysis*, 2nd ed. Courier Dover Publications, 2001.
- [20] H. Räcke, "Minimizing congestion in general networks," in *Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science (FOCS)*, 2002, pp. 43–52.
- [21] —, "Optimal hierarchical decompositions for congestion minimization in networks," in *Proceedings of the 40th ACM Symposium on Theory of Computing (STOC)*, 2008, pp. 255–264.
- [22] T. Roughgarden, "The price of anarchy is independent of the network topology," *Journal of Computer and System Sciences*, vol. 67, no. 2, pp. 341–364, 2003.
- [23] T. Roughgarden and É. Tardos, "How bad is selfish routing?" *Journal of the ACM*, vol. 49, no. 2, pp. 236–259, 2002.