Oblivious Network Design

Anupam Gupta [*] *Mohammad T. Hajiaghayi* [†] *Harald Räcke*[*]

Abstract

Consider the following network design problem: given a network $G = (V, E)$, source-sink pairs $\{s_i, t_i\}$ arrive and desire to send a unit of flow between themselves. The cost of the routing is this: if edge e carries a total of f_e flow (from all the terminal pairs), the cost is given by $\sum_e \ell(f_e)$, where ℓ is some concave cost function; the goal is to minimize the total cost incurred. However, we want the routing to be *oblivious*: when terminal pair $\{s_i, t_i\}$ makes its routing decisions, it does not know the current flow on the edges of the network, nor the identity of the other pairs in the system. Moreover, it does not even know the identity of the function ℓ, merely knowing that ℓ is a concave function of the total flow on the edge. How should it (obliviously) route its one unit of flow? Can we get competitive algorithms for this problem?

In this paper, we develop a framework to model *oblivious network design* problems (of which the above problem is a special case), and give algorithms with poly-logarithmic competitive ratio for problems in this framework (and hence for this problem). Abstractly, given a problem like the one above, the solution is a multicommodity flow producing a "load" on each edge of $L_e = \ell(f_1(e), f_2(e), \ldots, f_k(e))$, and the total cost is given by an "aggregation function" $\mathsf{agg}(L_{e_1}, \ldots, L_{e_m})$ of the loads of all edges. Our goal is to develop oblivious algorithms that approximately minimize the total cost of the routing, knowing the aggregation function agg, but *merely knowing that ℓ lies in some class C*, and having no other information about the current state of the network. Hence we want algorithms that are simultaneously "function-oblivious" as well as "traffic-oblivious".

The aggregation functions we consider are the $\max$ and $\sum$ objective functions, which correspond to the well-known measures of *congestion* and *total cost* of a network; in this paper, we prove the following:

- If the aggregation function is $\sum$, we give an oblivious algorithm with $O(\log^2 n)$ competitive ratio whenever the load function ℓ is in the class of *monotone sub-additive functions*. (Recall that our algorithm is also "function-oblivious"; it works whenever each edge has a load function ℓ in the class.)
- For the case when the aggregation function is $\max$, we give an oblivious algorithm with $O(\log^2 n \log \log n)$ competitive ratio, when the load function ℓ is a *norm*; we also show that such a competitive ratio is not possible for general sub-additive functions.

These are the first such general results about oblivious algorithms for network design problems, and we hope the ideas and techniques will lead to more and improved results in this area.

1 Introduction

In a general network design problem, we are given a graph $G = (V, E)$ with $|V| = n$ nodes and a set of k source-target pairs (s_i, t_i), for $1 \le i \le k$, called the *commodities*: a feasible solution is to build a network capable of simultaneously delivering some required flow amount d_i from each source s_i to its corresponding sink t_i. The goal in many of these problems is to design a network of minimum cost that meets these requirements.

The notion of cost can be a fairly variable one, with different cost measures arising in different contexts: e.g., one can think of the cost to be the total amount of flow in the system, or perhaps the average latency, or perhaps the congestion, or one of several other possibilities. In this paper, we present an abstract cost model in the following way: we are given a *load function* ℓ: for any edge, if $f_i(e)$ is the amount of commodity i sent along the edge e, then

$$\text{load } L(e) = \ell(f_1(e), f_2(e), \ldots, f_k(e)). \tag{1.1}$$

(As a minor aside, each edge can have a "weight" w_e as well, and then the load can be scaled thus: $L_e(e) = w_e L(e)$; however, for simplicity, we will assume that all $w_e = 1$.) The total cost of the entire flow is given by an *aggregation* function $\mathsf{agg} : \mathbb{R}^{|E|} \to \mathbb{R}$, which takes the load of the various edges as input, and outputs the "cost" of the entire flow f; i.e.,

$$\text{cost}(f) = \mathsf{agg}(L(e_1), L(e_2), \ldots, L(e_m)). \tag{1.2}$$

[*]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, U.S.A., Email: `{anupamg,harry}@andrew.cmu.edu`. The first author's research is partly supported by an NSF CAREER award CCF-0448095, and by an Alfred P. Sloan Fellowship.

[†]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 200 Technology Square, Cambridge, MA 02139, U.S.A., Emails: `hajiagha@theory.lcs.mit.edu`.

By instantiating the load and aggregation functions differently, we can give rise to many different problems that arise in traffic routing, and in telephone and computer networks. For instance, the simplest form of the load function is when $L(e) = \ell(f_1(e), \ldots, f_k(e)) = \sum f_i(e)$; simply the total flow passing through an edge e. Other important variants are the cases where the load is a *convex* function of the total passing flow (in the context of the latency of traffic networks, e.g., see [35]), or a *concave* function of the total passing flow (in the context of buy-at-bulk or rent-or-buy network design, e.g., see [4, 36]), and more generally a *monotone sub-additive* function of the passing flow.[1] Typical *aggregation* functions we consider are the $\max$ and $\sum$ operators, which correspond to the *congestion* and the *total cost* of the network.

In this paper, we initiate the study of *oblivious algorithms* for network design problems. Loosely, we want to develop algorithms where a demand pair $\{s_i, t_i\}$ should decide how to send the d_i amount of flow knowing only some information about the edge-load and aggregation functions on the edges, while having no information about which other demand pairs are in the system. Ideally, we would like the algorithms to not even know the precise details of the load functions, but just some defining characteristics; e.g., if each demand-pair just knew that the load function ℓ is some fixed but unknown concave function, what could it do? (A non-oblivious version of this problem was studied by Goel and Estrin [17] for single-sink network design.) Such algorithms that base their routing decisions only on local (and partial) knowledge can naturally be implemented efficiently in distributed environments.

As a simple example, if we know that each $\ell(\vec{f}(e)) = \sum_e f_i(e)$, and the aggregation function is also the sum, then each pair choosing a shortest s_i-t_i path (oblivious of the other pairs) is the optimal algorithm. Now if we change the load function to be $\ell(\vec{f}(e)) = \mathbf{1}_{(\sum_i f_i(e) > 0)}$ while keeping $\mathsf{agg} = \sum$, then we get an *oblivious Steiner forest problem*, for which algorithms were known only for the Steiner *tree* case by Jia et al. [28]. (We will give an algorithm with an improved bound, which also extends for the *forest* case.) On the other hand, if we keep $L(e)$ to be the total flow on e, but instead set the aggregation function $\mathsf{agg} = \max$, we get the *low-congestion oblivious routing* problem; originally studied by Valiant and Brebner [38], polylogarithmic competitive algorithms for this problem for general graphs were given only recently by Räcke [34]. These examples show that as we alter the load and aggregation functions, the resulting problems may exhibit very differing behaviors and levels of hardness.

The focus of the current paper is to develop a general framework for oblivious routing algorithms for network design problems: as mentioned above, we aim to minimize the total cost, which naturally depends on the load and aggregation functions. We mainly consider undirected networks, for the cases when the aggregation function agg is either $\max$ or $\sum$. Our main results will be for cases when the load function $\ell(\vec{f}(e))$ is either a monotone sub-additive function, or a norm.[2] Towards the end of the paper, we mention some results regarding directed graphs, convex load functions, and the non-uniform case (where the load function ℓ_e differs for different edges).

[1]The load function ℓ is sub-additive if $\ell(\vec{f}) + \ell(\vec{f}') \geq \ell(\vec{f} + \vec{f}')$

[2]The load function is a norm if it is monotone, sub-additive and if $\ell(\alpha\vec{f}) = \alpha\ell(\vec{f})$.

1.1 Our Results.

In this paper, we present oblivious algorithms for two broad classes of network design problems; both of these algorithms have poly-logarithmic competitive ratios.

The "Total-Load" aggregation function. We first consider the case when the aggregation function agg is the "sum" function. For this case, our main result is the following:

THEOREM 1.1. *For the case where* $\mathsf{agg} = \sum$, *there is an oblivious routing algorithm whose competitive ratio is* $O(\log^2 n)$ *when the edge load function* $\ell(f_1, f_2, \ldots, f_k)$ *is monotonically increasing and sub-additive. In fact, for any pair* $\{u, v\}$, *we can ensure that all the* u-v *flow is sent along a single path.*

Note that this result implies, among other things, an $O(\log^2 n)$ competitive algorithm for the *oblivious Steiner network design* problem, where the terminal pairs $\{s_i, t_i\}$ arrive online and must be connected using a pre-determined path. This result can be viewed as an extension of results of Jia et al. [28] for the so-called *Universal Steiner Tree* problem, which we discuss at length in Section 1.2. In fact, the same algorithm also works for all load functions ℓ which are concave, thus extending the results of Goel and Estrin [17] to the multicommodity (and traffic-oblivious) case. (Goel and Estrin considered the single-source concave-cost problem, for which they gave an $O(\log n)$-approximation algorithm; their algorithm was not online.)

At a high level, our main proof technique is to use "tree-covers": a family of $O(\log n)$ trees such that each pair of vertices in the graph has a tree in this family that maintains distances well between them. By itself, this idea does not work: we have to carefully construct such a "useful" tree-cover so that we can claim that a high cost on the trees implies a high cost on the original graph as well; i.e., we need to preserve lower bounds. To this end, we rely on the construction of Fakcharoenphol et al. [15], as well as some ideas used by Chan et al. [10] to create "useful" tree covers from their randomized constructions.

The "Congestion" Aggregation function. We then go on to consider the oblivious network design when $\mathsf{agg} = \max$; i.e., we are looking at the edge with the highest load $L(e)$. For this case, we can prove the following result.

THEOREM 1.2. *For the case where* $\mathsf{agg} = \max$, *there is an oblivious routing algorithm whose competitive ratio is* $O(\log^2 n \log\log n)$ *whenever the edge load function* $\ell(f_1, f_2, \dots, f_k)$ *is a norm.*

Note that Theorem 1.2 generalizes the oblivious routing result of Räcke [34] to a much broader class of load functions ℓ than just $L(e) = \sum_i f_i(e)$. The main technique we use to prove this result is to generalize the hierarchical decomposition tree approach of Räcke [34], which was subsequently improved by Harrelson et al. [25] to obtain an $O(\log^2 n \log\log n)$ competitive ratio for the case of $L(e) = \sum_i f_i(e)$.

In addition, we show that there is a sub-additive function for which the competitive ratio is at least $n^{1/4}$, implying that we cannot extend Theorem 1.2 to handle arbitrary sub-additive functions (as in Theorem 1.1).

Other Results.

Finally, we consider the case of convex load functions in directed/undirected graphs and provide several lower bounds in this case (though obtaining a polylogarithmic upper bound remains an open problem).

We end the paper with several interesting open problems.

1.1.1 A Note on Randomness and Tree Distributions.

A natural question is whether the results for the case of $\mathsf{agg} = \sum$ (i.e., for the total load objective function) can be obtained by using standard techniques such as distributions over trees [6, 15]. The reason why these techniques do not give our results is twofold. Firstly, to use those results, the demand pairs need to have some access to a shared source of randomness so that two different demand pairs can use the same random tree to make their decisions, whereas we demand that our algorithms be oblivious of the current state of the system, and have no such shared randomness. Secondly, note that the oblivious strategy in this case is a single path that the pair uses to connect between themselves, and once these paths have been fixed, there is no more randomness in the system. Hence, even if the adversary were to choose the load function ℓ and the demand pairs *after* looking at the paths given by our algorithm, he would not be able to obtain a competitive ratio worse than our guarantees. This is not so when we use distributions over trees in the standard way. (Formally, as in Goel and Estrin [17], we give a guarantee on the $E_{\text{our coins}} \max_{\text{instances}}[\text{competitive ratio}]$ instead of the usual $\max_{\text{instances}} E_{\text{our coins}}[\text{competitive ratio}]$.)

1.2 Related Work.

Network design problems with convex, concave, monotone sub-additive and more general edge load functions often arise in practice, and have been considered extensively in the past literature. There have been numerous approximation algorithms results dealing with *concave* load functions on the edges, trying to model *economies of scale*, a phenomena that has come to be known as *buy-at-bulk*: a partial list includes [4, 7, 36, 29, 18, 19, 33, 37, 30, 22, 21, 16, 11]; some recent hardness results have appeared in [2, 13]. While most of these papers consider off-line problems with centralized control, and the aim of this work is to be online and distributed, there are some other conceptual relations between our work. Karger and Minkoff [29] consider the problem where each vertex obliviously chooses a fixed path to the root; however, they look at approximating the expected cost incurred, and not the competitive ratio. (These results are related to the recently growing body of *stochastic network design* [27, 23, 14] as well, but all these results restrict their attention to the expected costs as well.) The work of Goel and Estrin [17] gave an algorithm for off-line single-sink network design that was function-oblivious for the class of concave-cost functions; our results extend their results to online problems (where the adversary can choose both the demand set and the function).

Our work closely follows the recent work of Jia, Lin, Noubir, Rajaraman, and Sundaram [28] on *universal approximation*. Their paper gives algorithms for the Universal TSP and Universal Steiner tree problems; loosely, a universal solution is a tour (or Steiner tree) such that the subtour (or subtree) induced by the actual demands is a good solution to the actual demands themselves. Our model is cosmetically different, since we do not require oblivious actions to be "induced" by some global solution (and hence we use a different name for our model); however, our results are very close in spirit to theirs.

As mentioned above, our work is related to the substantial work on approximating metrics by distributions over tree metrics [6, 7, 15]. While our results do not follow directly from these results (see the discussion in Section 1.1), they form an important component of our solution. Indeed, the approach of Fakcharoenphol et al., and its use by Chan et al. [10] provide a central inspiration to the techniques of this paper.

Congestion. The notion of "congestion" as the total cost of the system has also been well-studied. Special cases of networks are parallel links between two nodes, which correspond to scheduling problems, and the notion of congestion on these instances corresponds to the *makespan* of the job assignment. The idea of selecting routing paths *oblivious* to the traffic in the network was initially intensively studied for special network topologies: Valiant and Brebner [38] initiated this study, with oblivious routing algorithms on the hypercube. Räcke [34] showed there is an oblivivous routing algorithm with polylogarithmic competitive ratio (w.r.t. edge-congestion) for any undirected graph. Subsequent work [5, 3, 9, 25] improved on these results, showing that these routings can be found in polynomial time, and that the competitive ratio can be improved to $O(\log^2 n \log\log n)$. These results, as well as the hierarchical tree construction used to prove many of these results, have proved useful in many other contexts; see, e.g. [1, 12, 32]).

1.3 Basic Definitions. We represent the network as an unweighted and (unless otherwise stated) undirected graph $G = (V, E)$. We restrict the attention in this writeup to unweighted graphs in order to present our results as clearly and simple as possible; all the results still hold if edges carry a (polynomially bounded) weight, which could be interpreted either as the *edge length* or as the *edge capacity*, depending on the problem. (In fact, most of our results hold for arbitrary positive edge-weights, not only those which are polynomially-bounded; however, in this case, the construction of the oblivious routing scheme in the scenario $\mathsf{agg} = \max$ may not be polynomial time any more.)

For the network G we are given a set of k source-target pairs (s_i, t_i), $1 \leq i \leq k$. An *oblivious routing scheme* for G specifies, for each pair (s_i, t_i), a unit flow from s_i to t_i. This flow determines how the demand from s_i to t_i is routed in the network. For an oblivious algorithm OBL and an optimum algorithm OPT we use $\mathrm{OBL}_{\mathsf{agg}}(\ell, D)$ and $\mathrm{OPT}_{\mathsf{agg}}(\ell, D)$, respectively, to denote the cost produced by the algorithm for aggregation function agg, load-function ℓ and demand-vector D. For a given class C of load functions ℓ, the quantity

$$\max_{\ell\in C}\max_{D}\left\{\frac{\mathrm{OBL}_{\mathsf{agg}}(\ell, D)}{\mathrm{OPT}_{\mathsf{agg}}(\ell, D)}\right\}$$

is the *competitive ratio* of the oblivious routing algorithm with respect to load-functions from C. Note that the term “oblivious” signifies that the routing path chosen by the algorithm is independent both of the the load function ℓ, and also of the set of demands D. Recall that agg will be either the sum (denoted by $\sum$), or the maximum (denoted by $\max$); we will drop the subscript agg if it is clear from the context.

Finally, for a demand vector D and a set I of commodities, we use the notation $D|_I$ to denote the demand vector in which all entries representing commodites not in I have been set to 0.

2 Network Design to Minimize Total Load

In this section we analyze oblivious routing algorithms for the case that the aggregation function agg is the sum of the link loads, and the load function ℓ is a sub-additive function. (For some cases, we can obtain better results when ℓ is a norm.) The main result of this section is the following theorem.

THEOREM 2.1. *Let the aggregation function* $\mathsf{agg} : \mathbb{R}^{|E|} \to \mathbb{R}$ *be the sum. Then the following results hold.*

- *there is an integral oblivious routing scheme that for any sub-additive load function ℓ, and for any demand-matrix D, incurs a cost that is only a factor $O(\log^2 n)$ times the optimum.*
- *There is a fractional oblivious routing scheme that achieves a competitive ratio of $O(\log n)$ for any demand matrix D provided the load-function ℓ is a norm.*

A natural approach to obtain these results would be to try and use the well-known techniques for approximating metrics by distributions over tree metrics [6, 15]; while these techniques do not directly give us the results (as noted in Section 1.1.1), our oblivious routing algorithms are obtained by natural but subtle extensions of those tesults. In particular, we extensively use the scheme of Fakcharoenphol, Rao and Talwar [15] (which we call the FRT-scheme in the rest of our discussions).

2.1 Definitions and Preliminaries. Let us introduce some notation about hierarchical decompositions, and give a brief overview over the FRT-scheme, and its properties that are relavant to our routing algorithms.

2.1.1 Decompositions and Padding.

DEFINITION 2.1. (**Δ-partition**) *Given $\Delta > 0$, a Δ-partition P of a graph $G = (V, E)$ is a partition of the vertex set V into* clusters $V_1, V_2, \ldots$ *such that each induced sub-graph $G[V_1]$ has diameter at most Δ.*

DEFINITION 2.2. (**Hierarchical Decomposition**) *A* hierarchical decomposition *$\mathcal{P}$ of the graph $G = (V, E)$ is a sequence of partitions $P_0, \ldots, P_h$ (with $h = \lceil \log_2(\mathrm{diam}(G)) \rceil$) such that*

1. *The partition P_h has one cluster V, which is the entire node set of G.*
2. *The partition P_i is a 2^i-partition, i.e., a partition in which each cluster has diameter at most 2^i.*
3. *P_i is a* refinement *of P_{i+1}, i.e., each cluster in P_i is completely contained in some cluster of P_{i+1}.*

The following definition will be crucial for our analysis:

DEFINITION 2.3. (**α-padded nodes**) *Given a hierarchical decomposition $P = (P_i)_{i=0}^h$, a node $v \in V$ is α-*padded *in P if for all $i \in [0, h]$, the ball $B(v, \alpha \cdot 2^i)$ is contained in some cluster of P_i.*

Note that the standard definitions of padding (e.g., in [20]) usually refer to padding with respect to a single partition:

indeed, a node is defined to be padded if it is far away from the boundary of its cluster in the partition. In contrast, here we require that a node v is far from the boundary of its cluster at *every* level of the decomposition (i.e., for $\Theta(\log n)$ partitions), which is a much stronger requirement.

DEFINITION 2.4. *An* α-padded probabilistic hierarchical decomposition *is a probability distribution over hierarchical decompositions such that for any node* $v \in V$

$$\mathbf{Pr}[v \textit{ is } \alpha\textit{-padded in } \mathcal{P}] \geq \frac{3}{4} \ ,$$

where $\mathcal{P}$ *denotes the randomly chosen hierarchical decomposition.*

Note that the precise value $3/4$ of the fraction is in no way significant: any constant value greater than $1/2$ would serve for our purposes.

2.1.2 The FRT Cutting Scheme. The FRT scheme [15] is a randomized algorithm that outputs a (random) hierarchical decomposition of an input graph G, i.e., it gives a probability distribution over hierarchical decompositions. The algorithm works as follows: suppose we are given a graph $G = (V, E)$ with unit minimum distance, and diameter $\mathrm{diam}(G)$. We pick a random permutation π on the vertices of G, and also choose a random value β uniformly from the interval $[\frac{1}{2}, 1)$. For each i, we define the "radius" $\gamma_i = \beta \times 2^{i-1}$, and hence γ_i is chosen uniformly from the interval $[2^{i-2}, 2^{i-1}]$.

Given the random choices of the permutation π and scaling parameter β, the construction of the hierarchical decomposition is inductively defined. We define the partition P_h to contain all the vertices V. To obtain the partition P_i from P_{i+1}, consider the vertex u, and let C_u be the cluster containing u in P_{i+1}. Now, the node u is said to be *assigned to* the first vertex v (according to permutation π) that lies in the set $C_u \cap B(u, \gamma_i)$. Finally, for each vertex in V, the set of vertices that are assigned to it in the above set form a cluster of P_i.

Let us note that this indeed defines a hierarchical decomposition, as in Definition 2.2: by construction, each cluster of P_i lies completely within some cluster in P_{i+1}. Furthermore, if nodes x and y lie in some cluster in P_i, they are assigned to the same node v, and hence are at distance at most $\gamma_i \leq 2^{i-1}$ from v; by the triangle inequality, they are at distance at most $2\gamma_i = 2^i$ from each other.

LEMMA 2.1. *Given any graph G, the FRT-scheme defines an* α*-padded probabilistic hierarchical decomposition with* $\alpha = \Omega(\frac{1}{\log n})$.

Proof. Since the FRT scheme outputs a random hierarchical decomposition, it suffices to show that each node is $\Omega(1/\log n)$-padded with probability $3/4$. Hence, let us analyze the probability that the random level-i partition of the FRT scheme cuts a ball $B(v, \alpha \cdot 2^i)$ with radius $\alpha \cdot 2^i$ around some node $v \in V$. The analysis is very similar to that in [15], and is given here for completeness. We introduce the following notation: we call a node u a *protecting node* for ball $B(v, \alpha 2^i)$ and (random) radius γ_i if $\gamma_i \geq d(u, v) + \alpha 2^i$. Similarly, we call u a *cutting node* for ball $B(v, \alpha 2^i)$ and radius γ_i if

$$d(u, v) - \alpha 2^i < \gamma_i < d(u, v) + \alpha 2^i \ .$$

We say a node *first cuts* the ball if it is a cutting node and all other cutting nodes and protecting nodes appear after u in the permutation.

Note that the definition of cutting nodes is somewhat pessimistic, in the sense that a cutting node for radius γ_i may not actually end up cutting the ball $B(v, \alpha 2^i)$ in the FRT scheme; indeed, the balls $B(u, \gamma_i)$ and $B(v, \alpha 2^i)$ may turn out to be disjoint, or $B(v, \alpha 2^i)$ may be contained in $B(u, \gamma_i)$. However, if the ball $B(v, \alpha 2^i)$ is indeed cut, then there must exist a node u that first cuts according to the above definition. In this case, two events must happen: (a) the radius $\gamma_i \in [d(u, v) - \alpha 2^i, d(u, v) + \alpha 2^i]$, and (b) all nodes that are closer to v than u (which would all either cut or protect the ball) appear after u in the permutation π. The first event happens with probability $(2\alpha 2^i)/2^{i-2} = 8\alpha$ since γ_i is chosen uniformly from the interval $[2^{i-2}, 2^{i-1}]$ which has length 2^{i-2}. The second event occurs with probability $\frac{1}{s}$ if u is the s^{th}-closest node to v.

Furthermore, a node u whose distance from v is greater than $2^{i-1} + \alpha 2^i = (1 + 2\alpha)2^{i-1}$, or whose distance from v is less than $2^{i-2} - \alpha 2^i = (1 - 4\alpha)2^{i-2}$ cannot cut the ball at all. This gives the following bound on the probability that the ball $B(v, \alpha 2^i)$ around v is cut on level i.

$$\begin{aligned}\mathbf{Pr}\big[B(v,\alpha 2^i) \text{ is cut on level } i\big] & \\ &\leq \mathbf{Pr}[\exists \text{ a node } u \text{ that first cuts the ball}] \\ &\leq \sum\nolimits_u \mathbf{Pr}[u \text{ cuts the ball first}] \\ &\leq 8\alpha \sum_{s=|B(v,(1-4\alpha)2^{i-2})|}^{|B(v,(1+2\alpha)2^{i-1})|} \frac{1}{s}\end{aligned}$$

Now, we can estimate the probability that any of the $(\alpha 2^i)$-balls around v is cut in the respective level.

$$\begin{aligned}&\mathbf{Pr}\big[\exists\ i \text{ such that } B(v, \alpha 2^i) \text{ is cut at level } i\big] \\ &\qquad \leq 8\alpha \sum_i \sum_{s=|B(v,(1-4\alpha)2^{i-2})|}^{|B(v,(1+2\alpha)2^{i-1})|} \frac{1}{s}\end{aligned} \tag{2.3}$$

We say a node u *contributes to the above sum for level* i if $(1 - 4\alpha)2^{i-2} \leq d(u, v) < (1 + 2\alpha)2^{i-1}$; in this case, its contribution is $\frac{1}{s}$, if u is the s-closest node to x. We show that a node only contributes to a constant number of levels; this will imply that the probability in (2.3) is at most

$O(\alpha \log n)$, which will prove the result. Indeed, suppose u contributes to the sum for levels i and i' with $i > i'$. Then $(1-4\alpha)2^{i-2} \leq d(u,v) \leq (1+2\alpha)2^{i'-1}$, and if $\alpha < 1/4$, then $i < i' + 1 + \log \frac{1+2\alpha}{1-4\alpha}$. Now, choosing $\alpha \leq \frac{1}{8}$ gives $i < i' + 3$, and hence the node u can only contribute to at most 3 different levels; this implies that

$$\textbf{Pr}\left[\exists\, i \text{ such that } B(v, \alpha 2^i) \text{ is cut at level } i\right] \leq 24\alpha \cdot \sum_{s=1}^{n} \frac{1}{s} \leq O(\alpha \log n) \ .$$

We can choose $\alpha = \Omega(\frac{1}{\log n})$ such that the above probability is less than 1/4. This finishes the proof of the lemma. □

Before we end, note that each hierarchical decomposition P can be associated with a *decomposition tree* T_P, whose vertices correspond to the clusters in the various decompositions of P, and whose edges connect clusters $C \in P_i$ and $C' \in P_{i-1}$ iff $C' \subset C$. (As an aside, if we assign a length of 2^i to such edges, we get back the same HSTs that were given in [15].) Given this correspondence, we will move between hierarchical decompositions and their decomposition trees in the rest of the discussion.

2.2 The Well-Padded Tree Cover. We construct a *tree cover* of the graph in the following way: we take $M = O(\log n)$ samples from the distribution of Lemma 2.1, and let $\mathcal{T}$ denote the decomposition trees in this sample. With high probability, the set $\mathcal{T}$ is a good *tree-cover* of G: i.e., for any pair of vertices $x, y \in V$, there is a tree $T_i \in \mathcal{T}$ with $d_{T_i}(x,y) \leq O(\log n)\, d(x,y)$. By itself, this does not suffice for our purposes; we need the following stronger property:

> Given a parameter $z \in \{1, \log n\}$, for every pair of vertices $s, t \in V$, there must exist at least z decomposition trees in $\mathcal{T}$ such that both s and t are α-padded in the corresponding hierarchical decomposition.

We will call this family $\mathcal{T}$ to be a *well-padded tree cover*. The proof of the following theorem is standard, and is deferred to the full version of the paper.

THEOREM 2.2. *A set of $O(\log n)$ independent samples drawn from the distribution of Lemma 2.1 satisfies the property above with high probability.*

Moreover, the above randomized tree cover construction can be made deterministic by using standard derandomization techniques. A detailed description is deferred to a full version of the paper.

2.3 The Routing Algorithm. Given a well-padded tree cover $\mathcal{T}$ constructed in the previous section, note that each tree $T \in \mathcal{T}$ gives us a routing scheme for G in the following way: for every tree node $v_t \in T$, we choose a graph node $v \in C_{v_t}$, where C_{v_t} denotes the cluster corresponding to v_t. We call the node $v \in V$ the *node that simulates* v_t. For every edge (u_t, v_t) in T we connect the graph nodes simulating u_t and v_t via some shortest path in G. Now, given any pair of nodes $x, y \in V$ and a tree $T \in \mathcal{T}$, the routing-path $P_T(x,y)$ between them is obtained by first determining all tree edges on the path from $\{x\}$ to $\{y\}$ in T (recall that the clusters $\{x\}$ and $\{y\}$ are leaf nodes in T) and then concatenating the paths that correspond to these edges. We call this path $P_T(x,y)$ the *x-y path for tree T*.

Finally, we can define our oblivious routing algorithm. Given any source-target pair $(s,t) \in V \times V$, we mark all trees $T \in \mathcal{T}$ such that s and t are both α-padded in T; by the above property, there are at least z such trees in $\mathcal{T}$. We now select z of these trees arbitrarily, and route a flow of $1/z$ along the s-t path for each tree T among these z chosen selected trees, thus giving us a unit flow between s and t. Note for the case $z = 1$, this flow is integral—i.e., we have a single path between s and t. In the next section, we show that the oblivious strategy (with the parameter z set to be 1, or to $\log n$ respectively) gives the performances claimed in the two parts of Theorem 2.1.

2.4 Analysis of the Oblivious Scheme. We first give an upper bound on the cost of the oblivious routing algorithm on a fixed tree $T \in \mathcal{T}$. We can view this cost as being incurred by the paths that simulate the edges of the tree. Let for a fixed tree $T \in \mathcal{T}$, v_t denote a level-l node of this tree, and let p_t denote the parent of v_t. Moreover, let v and p denote the graph nodes in G that simulate v_t and p_t, respectively. We first analyze the cost $\text{cost}(v_t)$ for sending flow from v to p in the graph. Let $f^i_{\text{obl}}(v_t)$ denote the commodity i flow sent by the oblivious routing algorithm along this tree edge (v_t, p_t) (and hence along the path simulating it). Let I_t denote the set of commodities i for which $f^i_{\text{obl}}(v_t) > 0$. (In this case, $f^i_{\text{obl}}(v_t)$ is equal to zero if $i \notin I_t$, and equal to $\frac{1}{z} D_i$ if $i \in I_t$, since a $\frac{1}{z}$-fraction of the total flow is sent along any path). We also have that

$$\text{cost}_{\text{obl}}(v_t) = \ell(f^1_{\text{obl}}(v_t), \ldots, f^k_{\text{obl}}(v_t)) \cdot 2^{l+1} \ ,$$

since the v-p path emulating the edge (v_t, p_t) has length at most 2^{l+1}.

The total cost $\text{OBL}_T(\ell, D)$ of the oblivious algorithm for tree T can be calculated by summing the above term first over all level l nodes, and then over all levels in the tree. This gives

$$\begin{aligned} \text{OBL}_T(\ell, D) &\leq \sum_l \sum_{\substack{\text{nodes } v_t \\ \text{on level } l}} \ell(f^1_{\text{obl}}(v_t), \ldots, f^k_{\text{obl}}(v_t)) \cdot 2^{l+1} \\ &= \sum_l \sum_{\substack{\text{nodes } v_t \\ \text{on level } l}} \ell(1/z \cdot D|_{I_t}) \cdot 2^{l+1} \ . \end{aligned} \tag{2.4}$$

We now give a similar lower bound on the cost of an optimum solution.

CONDITION 2.1.

$$\mathrm{OPT}(\ell, D) \geq \frac{\alpha}{4} \cdot \sum_{l} \sum_{\substack{\text{nodes } v_t \\ \text{on level } l}} \ell(D|_{I_t}) \cdot 2^l$$

Proof. Fix a tree T. We partition the edges of G into $O(\log n)$ *nearly disjoint* classes, i.e. an edge will belong to at most two classes and there will be one class for each level of the tree. Then we show that an optimum algorithm induces cost at least

$$\frac{\alpha}{2} \cdot \sum_{\substack{\text{nodes } v_t \\ \text{on level } l}} \ell(D|_{I_t}) \cdot 2^l \tag{2.5}$$

on edges in class l. This then will yield the desired bound by summing over all levels.

The assignment of edges to classes is as follows. We say an edge is in class ℓ (meaning that it counts to the *l-th* level) if its distance to the nearest well-padded node falls into the interval $[\lfloor \alpha 2^{l-1} \rfloor, \lfloor \alpha 2^l \rfloor]$.[3] Note that by this definition each graph edge belongs to at most two different classes

Now, fix a level l. The hierarchical decomposition corresponding to T, partitions the nodes of G into different clusters on this level and these clusters correspond to the level l nodes v_t. By the definition of level l edges it follows that for such an edge both endpoints must be contained in some cluster C_{v_t}, which means that the clusters partition the level l edges into disjoint sets.

Fix one of the level l nodes v_t. Recall that I_t is the set of demands for which the oblivious routing algorithm sends traffic between node v_t and its parent p_t in the tree. For a commodity i this only happens if the cluster C_{v_t} that corresponds to tree node v_t separates source and target of this commodity, and moreover, the terminal node inside C_{v_t} must be at distance at least $\alpha 2^\ell$ from the boundary of C_{v_t} (this is because we only send flow along the trees in which the commodity is α-padded). We now show that the cost of the optimum solution on level l edges inside C_{v_t} is large.

Let E_d denote the set of edges such that their distance from the closest terminal inside C_{v_t} is some fixed value $d \in [\lfloor \alpha 2^{l-1} \rfloor, \lfloor \alpha 2^l \rfloor]$ (Note that all these edges lie within C_{v_t} and are in class l). These edges form a cut that separates all commodities in I_t. Let for $i \in I_t$ and $e \in E_d$, $f^i_{\mathrm{opt}}(e)$ denote the flow along edge e for commodity i in the optimum solution. The optimum cost for edges in E_d is

$$\begin{aligned} &\sum_{e \in E_d} \ell(f^1_{\mathrm{opt}}(e), \ldots, f^k_{\mathrm{opt}}(e)) \\ &\qquad \geq \ell(\textstyle\sum_{e \in E_d} f^1_{\mathrm{opt}}(e), \ldots, \sum_{e \in E_d} f^k_{\mathrm{opt}}(e)) \ , \end{aligned} \tag{2.6}$$

[3]The distance of a graph edge (x, y) from a node v is the distance from v to the farther of x and y.

because of sub-additivity. Since the edges in E_d form a cut separating the terminals for any commodity $i \in I_t$, the optimum solution must send a corresponding flow of D_i across the cut E_d; hence, for each $d \in [\lfloor \alpha 2^{l-1} \rfloor, \lfloor \alpha 2^l \rfloor]$, $\sum_{e \in E_d} f^i_{\mathrm{opt}}(e) \geq D_i$. Using Inequality 2.6, we get

$$\sum_{e \in E_d} \ell(f^1_{\mathrm{opt}}(e), \ldots, f^k_{\mathrm{opt}}(e)) \geq \ell(D|_{I_t}) \ .$$

There are $\lfloor \alpha 2^l \rfloor - \lfloor \alpha 2^{l-1} \rfloor + 1 \geq \alpha 2^{l-1}$ different choices for d which gives a cost of at least $\alpha 2^{l-1} \cdot \ell(D|_{I_t})$ on level l edges inside C_{v_t}. Summing over all nodes v_t gives Equation 2.5 and summing over all edge-classes l gives the claim, because an edge is contained in at most two classes. □

To finish the proof of Theorem 2.1 we have to compare the upper bound on the oblivious cost (Equation 2.4) to the lower bound on the optimum cost (Claim 2.1). First suppose that $z = 1$, i.e., we have an integral routing scheme. Then we get

$$\begin{aligned} \mathrm{OBL}_T(\ell, D) &\leq \sum_{l} \sum_{\substack{\text{nodes } v_t \\ \text{on level } l}} \ell(1/z \cdot D|_{I_t}) \cdot 2^{l+1} \\ &\leq \sum_{l} \sum_{\substack{\text{nodes } v_t \\ \text{on level } l}} \ell(D|_{I_t}) \cdot 2^{l+1} \\ &\leq \frac{8}{\alpha} \mathrm{OPT}(\ell, D) \ . \end{aligned}$$

Now the total cost of the oblivious algorithm is obtained by summing the above expression over all trees $T \in \mathcal{T}$. This gives

$$\begin{aligned} \mathrm{OBL}(\ell, D) &\leq |\mathcal{T}| \cdot \frac{8}{\alpha} \cdot \mathrm{OPT}(\ell, D) \\ &\leq O(\log^2 n) \cdot \mathrm{OPT}(\ell, D) \ , \end{aligned}$$

which proves the first part of the theorem. For the special case when the load function is a norm, we can get an improvement of a factor of z in Inequality 2.4 thus:

$$\begin{aligned} \mathrm{OBL}_T(\ell, D) &\leq \sum_{l} \sum_{\substack{\text{nodes } v_t \\ \text{on level } l}} \ell(1/z \cdot D|_{I_t}) \cdot 2^{l+1} \\ &= \sum_{l} \sum_{\substack{\text{nodes } v_t \\ \text{on level } l}} 1/z \cdot \ell(D|_{I_t}) \cdot 2^{l+1} \\ &\leq \frac{8}{z\alpha} \mathrm{OPT}(\ell, D) \ . \end{aligned}$$

For $z = \Theta(\log n)$ and $|\mathcal{T}| = \Theta(\log n)$ this gives $\mathrm{OBL}(\ell, D) \leq O(\log n) \cdot \mathrm{OPT}(\ell, D)$, which gives the second statement of the theorem.

Note that the $O(\log^2 n)$ guarantee for general sub-additive functions comes from two different logarithmic losses: there are $O(\log n)$ trees, and we are only guaranteed a padding of $\Omega(1/\log n)$. We do not know a lower bound of worse than $\Omega(\log n)$ on the problem, and hence it would be interesting to remove some of these factors.

3 Network design to minimize congestion

In this section we analyze oblivious routing algorithms for the case that the aggregation function agg is the maximum of the link loads. The oblivious routing algorithms developed in [34] and [25] for congestion minimization consider the special case when the load function ℓ is the sum, i.e., the load of an edge is the total flow going through the edge. This also gives a polylogarithmic competitive ratio if the load-function is a concave function of the total flow along an edge:

OBSERVATION 3.1. *There is an $O(\log^2 n \log\log n)$-competitive oblivious routing algorithm when the aggregation function* agg $=\max$*, and the load function $\ell(f_1(e),\dots,f_k(e)) = g(\sum_i f_i(e))$ where g is some concave function.*

Note that such a load function ℓ which is a concave function of the total flow through the edge has been widely used in the network-design literature, since it models the effects of "economies of scale", where the marginal cost of using bandwidth decreases as the utilization of the bandwidth increases. In order to obtain an algorithm that works for more general load functions, we have to revisit details from the proof of the oblivious routing schemes of [34] and [25]. The main result of this section is the following.

THEOREM 3.1. *Let the aggregation function* agg : $\mathbb{R}^{|E|} \to \mathbb{R}$ *be the maximum load on any edge. Then there is an oblivious routing algorithm which has a competitive ratio of $O(\log^2 n \log\log n)$ whenever the load function ℓ is a norm.*

Proof. The oblivious routing scheme of [34] and [25] are based on a hierarchical decomposition tree (see Section 2 for the definition of a hierarchical decomposition of a graph and the corresponding tree). For every cluster C in the hierarchy there is a distribution function $\rho_C : C \to [0,1]$, $\sum_{v\in C} \rho_C(v) = 1$ over nodes of C. An edge between a level i cluster C_i and a level $i+1$ cluster is simulated by redistributing a flow that is distributed according to function ρ_{C_i} into a distribution according to $\rho_{C_{i+1}}$ (or vice versa). (Note that doing this redistribution along a path between leaf nodes $\{x\}$ and $\{y\}$ generates a flow between x and y in the graph G)

In [25] it was shown that it is possible to find a hierarchical decomposition and a distribution function ρ_{C_i} for every cluster such that all edges (C_i, C_{i+1}) in the decomposition tree can simultaneously send a flow of $|E_{C_i}|$ in the above manner so that a graph edge only has to carry a total flow of $O(\log^2 n \log\log n)$ (E_{C_i} denotes the set of edges that leave cluster C_i). We will call this the flow-solution of the hierarchical decomposition in the following. We conclude the theorem from the above property of the flow-solution. First we give a lower bound on the cost of the optimum solution.

CONDITION 3.1. *Let for a cluster C, I_C denote the set of commodities that are separated by C. Then*

$$\mathrm{OPT}(\ell, D) \ge \frac{\ell(D|_{I_C})}{|E_C|} \ ,$$

where $\mathrm{OPT}(\ell, D)$ denotes the cost of the optimum solution.

Proof. Let $\vec{f}_{\mathrm{opt}}(e)$ denote the flow-vector along edge e in the optimum solution. By definition of the aggregation function we have

$$\begin{aligned}\mathrm{OPT}(\ell, D) &= \max_e\{\ell(\vec{f}_{\mathrm{opt}}(e))\}\\ &\ge \frac{1}{|E_C|}\sum_{e\in E_C} \ell(\vec{f}_{\mathrm{opt}}(e)) \ge \frac{1}{|E_C|}\ell\left(\textstyle\sum_{e\in E_C} \vec{f}_{\mathrm{opt}}(e)\right)\\ &\ge \frac{1}{|E_C|}\ell\left((\textstyle\sum_{e\in E_C} \vec{f}_{\mathrm{opt}}(e))|_{I_C}\right) \ge \frac{1}{|E_C|}\ell(D|_{I_C}) \ ,\end{aligned}$$

where the last step follows since a commodity in I_C is cut by cluster C. $\square$

Now, we give an upper bound on the cost generated by the oblivious routing scheme of [25]. Let for an edge e and a cluster C, $\alpha_C(e)$ denote the flow along edge e in the flow-solution for distributing flow from ρ_C to $\rho_{C'}$, where C' denotes the parent of C in the decomposition tree. The cost of the oblivious algorithm on edge e is

$$\mathrm{cost}_{\mathrm{obl}}(e) \le \ell\left(\frac{\alpha_C(e)}{|E_C|}\cdot D|_{I_C}\right) \ ,$$

since $\alpha_C(e)/|E_C|$ is the fraction of the flow that leaves cluster C and goes through e. By the properties of a norm we get

$$\begin{aligned}\mathrm{OBL}(\ell, D) &\le \max_e\left\{\ell\left(\textstyle\sum_C \frac{\alpha_C(e)}{|E_C|}\cdot D|_{I_C}\right)\right\}\\ &\le \max_e\left\{\textstyle\sum_C \frac{\alpha_C(e)}{|E_C|}\cdot \ell(D|_{I_C})\right\}\\ &\le \max_e\left\{\textstyle\sum_C \alpha_C(e)\cdot \mathrm{OPT}(\ell, D)\right\}\\ &\le O(\log^2 n \log\log n)\cdot \mathrm{OPT}(\ell, D) \ ,\end{aligned}$$

where the last step follows since $\sum_C \alpha_C(e)$ is the total flow along e in the flow-solution which is less than $O(\log^2 n \log\log n)$. $\square$

3.1 Lower Bounds for General Load Functions. Our results suggest the following natural question: can we extend the results of Theorem 3.1 to the case of sub-additive functions, perhaps with an additional polylogarithmic loss in the competitive ratio (as we did in Section 2)? The following lemma shows that such a result is not possible: indeed, there is a sub-additive function for which any oblivious routing algorithm obtains a bad competitive ratio *even on the complete graph*.

LEMMA 3.1. *For the case* agg $= \max$*, let the load-function* ℓ *be defined as* $\ell(f_1(e), \ldots, f_k(e)) := \sum_{i=1}^{k} \sqrt{f_i(e)}$*. Then any oblivious routing algorithm has competitive ratio* $\Omega(\sqrt[4]{n})$*, even on the complete graph.*

Intuitively, the reason for this bad competitive ratio is that the $\max$ aggregation function favors solutions that split flow between many different routing paths (especially in the case when only a single commodity is active), whereas the load function ℓ defined above favors aggregating flow (especially if many commodities are active). The complete proof is deferred to the full version.

4 Further Results

In this section we give an overview of further results in our oblivious network design model and describe further appliations. The proofs of these results will appear in the full version of the paper.

4.1 Network design for convex load functions. Till now, we have been considering situtations where the load functions ℓ on edges have been concave functions, norms, or most generally, subadditive functions of the flows using the edge. We now direct our attention to the oblivious network design problem when the load function ℓ is *convex* in the total flow going through e. We show that the competitive ratio in such situations may be large even when the aggregation function agg is the sum of the edge loads; however, these results are for the case when the load-functions are *not uniform*, i.e., different edges may have different load-functions.

LEMMA 4.1. *If the aggregation-function* agg *is the sum and the load-functions are allowed to be non-uniform, there is a network with load-functions* ℓ_e *being polynomials of degree at most* 2 *for which any oblivious algorithm has a competitive ratio of* $\Omega(n)$*.*

For the more restrictive case of directed graphs, one can achieve a lower bound of $\Omega(\sqrt{n})$ even with *uniform* load-functions ℓ (i.e., the same load function for each edge e).

LEMMA 4.2. *There are directed graphs with the same quadratic load-function on each edge, for which the competitive ratio of any oblivious routing scheme is* $\Omega(\sqrt{n})$*.*

Finally, there is also a lower bound for a quadratic load function in undirected networks.

LEMMA 4.3. *For a quadratic load-function there exist undirected graphs in which the competitive ratio of any oblivious routing algorithm is* $\Omega(\log^{3/2} n)$*.*

The proof of this lemma is contained in the final version of this paper.

4.2 Improved bounds for small doubling dimension. If the metric on the underlying graph has a small or even constant doubling dimension (see [20] for a definition) we can improve our results for the case that the aggregation function agg is the sum of the link-loads.

THEOREM 4.1. *Let the aggregation-function* agg *be the sum. Given a graph in which the distance-metric between node-pairs has doubling constant* λ*, there is an oblivious routing algorithm with competitive ratio* $O(\lambda^{O(1)} \cdot \log n)$ *for the case of monotone, sub-additive load-functions. This algorithm is integral.*

4.3 Improved results for universal TSP. Jia et al. [28] introduced the universal TSP-problem that asks for a permutation π_V over the vertex-set V of a graph such that for any sub-set $X \subset V$ the permutation π_X induced on X by π_V is close to an optimum traveling salesman tour for X. Jia et al. [28] gave algorithms with perfomance guarantee $O(\log^4 n)$ for this problem (i.e., the tour given on X by π_V is at most an $O(\log^4 n)$ factor larger than an optimal tour for X.)

Based on our well-padded tree cover we can prove the following theorem.

THEOREM 4.2. *There is an* $O(\log^2 n)$*-competitive algorithm for the universal TSP-problem.*

The above theorem improves on the result of Hajiaghayi et al. [24] who show an $O(\log^2 n)$ factor for planar networks. The result has been obtained after the original sub-mission to the SODA-conference.

5 Open problems

An obvious open problem is, of course, whether the competitive ratios of our algorithms can be improved. For both scenarios the best lower bounds on the competitive ratio are $\Omega(\log n)$. For the total load scenario this follows from a lower bound on online Steiner tree by Imase and Waxman [26], and for the congestion scenario the bound was independently proved in [31] and [8].

Another interesting direction for further research is the problem of designing oblivious algorithms for other aggregation-functions and other load-functions than the ones we discussed in this paper. Of particular interest is e.g. the model that is obtained by choosing the aggregation-function to be the sum and the load-function to be the square of the total flow going along an edge. This cost-model corresponds to minimizing average latency in a network where the links have linear latency-functions. In [35] this cost-model was considered in a game-theoretic setting and it is a challenging task to also find oblivious algorithms for it.

Acknowledgments We would like to thank Daniel Golovin, Robert Kleinberg, and Tom Leighton for fruitful discussions.

References

[1] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. S. Naor, *A general approach to online network optimization problems*, in Proc. of the 15th SODA, 2004, pp. 942–951.

[2] M. Andrews, *Hardness of buy-at-bulk network design*, in Proc. of the 45th FOCS, 2004, pp. 115–124.

[3] D. Applegate and E. Cohen, *Making intra-domain routing robust to changing and uncertain traffic demands: understanding fundamental tradeoffs*, in Proc. of the SIGCOMM, 2003, pp. 313–324.

[4] B. Awerbuch and Y. Azar, *Buy-at-bulk network design*, in Proc. of the 38th FOCS, 1997, pp. 542–547.

[5] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Räcke, *Optimal oblivious routing in polynomial time*, in Proc. of the 35th STOC, 2003, pp. 383–388.

[6] Y. Bartal, *Probabilistic approximations of metric spaces and its algorithmic applications*, in Proc. of the 37th FOCS, 1996, pp. 184–193.

[7] ———, *On approximating arbitrary metrics by tree metrics*, in Proc. of the 30th STOC, 1998, pp. 161–168.

[8] Y. Bartal and S. Leonardi, *On-line routing in all-optical networks*, Theor. Comput. Sci., 221 (1999), pp. 19–39. Also in *Proc. 24th ICALP*, 1997, pp. 516–526.

[9] M. Bienkowski, M. Korzeniowski, and H. Räcke, *A practical algorithm for constructing oblivious routing schemes*, in Proc. of the 16th SODA, 2003, pp. 24–33.

[10] H. T.-H. Chan, A. Gupta, B. M. Maggs, and S. Zhou, *On hierarchical routing in Doubling metrics*, in Proc. of the 16th SODA, 2005, pp. 762–771.

[11] M. Charikar and A. Karagiozova, *On non-uniform multicommodity buy-at-bulk network design*, in Proc. of the 37th STOC, 2005, pp. 176–182.

[12] C. Chekuri, S. Khanna, and B. Shepherd, *The all-or-nothing multicommodity flow problem*, in Proc. of the 36th STOC, 2004, pp. 156–165.

[13] J. Chuzhoy, A. Gupta, J. S. Naor, and A. Sinha, *On the approximability of network design problems*, in Proc. of the 16th SODA, 2005, pp. 943–951.

[14] B. C. Dean, M. X. Goemans, and J. Vondrák, *Approximating the stochastic knapsack problem: The benefit of adaptivity*, in Proc. of the 45th FOCS, 2004, pp. 208–217.

[15] J. Fakcharoenphol, S. B. Rao, and K. Talwar, *A tight bound on approximating arbitrary metrics by tree metrics*, in Proc. of the 35th STOC, 2003, pp. 448–455.

[16] N. Garg, R. Khandekar, G. Konjevod, R. Ravi, F. S. Salman, and A. Sinha, *On the integrality gap of a natural formulation of the single-sink buy-at-bulk network design formulation*, in Proc. of the 8th IPCO, 2001, pp. 170–184.

[17] A. Goel and D. Estrin, *Simultaneous optimization for concave costs: Single sink aggregation or single source buy-at-bulk*, in Proc. of the 14th SODA, 2003, pp. 499–505.

[18] S. Guha, A. Meyerson, and K. Munagala, *Hierarchical placement and network design problems*, in Proc. of the 41st FOCS, 2000, pp. 603–612.

[19] S. Guha, A. Meyerson, and K. Mungala, *A constant factor approximation for the single sink edge installation problem*, in Proc. of the 33rd STOC, 2001, pp. 383–388.

[20] A. Gupta, R. Krauthgamer, and J. R. Lee, *Bounded geometries, fractals, and low–distortion embeddings*, in Proc. of the 44th FOCS, 2003, pp. 534–543.

[21] A. Gupta, A. Kumar, M. Pál, and T. Roughgarden, *Approximations via cost-sharing: A simple approximation algorithm for the multicommodity rent-or-buy problem*, in Proc. of the 44th FOCS, 2003, pp. 606–615.

[22] A. Gupta, A. Kumar, and T. Roughgarden, *Simpler and better approximation algorithms for network design*, in Proc. of the 35th STOC, 2003, pp. 365–372.

[23] A. Gupta, M. Pál, R. Ravi, and A. Sinha, *Boosted sampling: Approximation algorithms for stochastic optimization problems*, in Proc. of the 36th STOC, 2004, pp. 417–426.

[24] M. T. Hajiaghayi, R. D. Kleinberg, and F. T. Leighton, *Improved lower and upper bounds for universal TSP in planar metrics*, in Proc. of the 17th SODA, 2006. to appear.

[25] C. Harrelson, K. Hildrum, and S. B. Rao, *A polynomial-time tree decomposition to minimize congestion*, in Proc. of the 15th SPAA, 2003, pp. 34–43.

[26] M. Imase and B. M. Waxman, *Dynamic steiner tree problem*, SIAM J. Discrete Math., 4 (1991), pp. 369–384.

[27] N. Immorlica, D. Karger, M. Minkoff, and V. Mirrokni, *On the costs and benefits of procrastination: Approximation algorithms for stochastic combinatorial optimization problems*, in Proc. of the 15th SODA, 2004, pp. 684–693.

[28] L. Jia, G. Lin, G. Noubir, R. Rajaraman, and R. Sundaram, *Universal approximations for TSP, Steiner tree, and set cover*, in Proc. of the 37th STOC, 2005, pp. 386–395.

[29] D. R. Karger and M. Minkoff, *Building Steiner trees with incomplete global knowledge*, in Proc. of the 41st FOCS, 2000, pp. 613–623.

[30] A. Kumar, A. Gupta, and T. Roughgarden, *A constant-factor approximation algorithm for the multicommodity rent-or-buy problem*, in Proc. of the 43rd FOCS, 2002, pp. 333–342.

[31] B. M. Maggs, F. Meyer auf der Heide, B. Vöcking, and M. Westermann, *Exploiting locality for networks of limited bandwidth*, in Proc. of the 38th FOCS, 1997, pp. 284–293.

[32] B. M. Maggs, G. L. Miller, O. Parekh, R. Ravi, and S. L. M. Woo, *Finding effective support-tree preconditioners*, in Proc. of the 17th SPAA, 2005, pp. 176–185.

[33] A. Meyerson, K. Munagala, and S. A. Plotkin, *Cost-distance: Two metric network design*, in Proc. of the 41st FOCS, 2000, pp. 624–630.

[34] H. Räcke, *Minimizing congestion in general networks*, in Proc. of the 43rd FOCS, 2002, pp. 43–52.

[35] T. Roughgarden and É. Tardos, *How bad is selfish routing?*, J. ACM, 49 (2002), pp. 236–259.

[36] F. S. Salman, J. Cheriyan, R. Ravi, and S. Subramanian, *Approximating the single-sink link-installation problem in network design*, SIAM J. Optimization, 11 (2000), pp. 595–610.

[37] K. Talwar, *Single-sink buy-at-bulk LP has constant inte-*

grality gap, in Proc. of the 9th IPCO, 2002, pp. 475–486.

[38] L. G. VALIANT AND G. J. BREBNER, *Universal schemes for parallel communication*, in Proc. of the 13th STOC, 1981, pp. 263–277.