Demand-oblivious routing: distributed vs. centralized approaches

Gábor Rétvári, Gábor Németh
Dept. of Telecommunications and Media Informatics
Budapest University of Technology and Economics
Email: {retvari,nemethgab}@tmit.bme.hu

***Abstract*—Until recent years, it was more or less undisputed common-sense that an accurate view on traffic demands is indispensable for optimizing the flow of traffic through a network. Lately, this premise has been questioned sharply: it was shown that setting just a single routing, the so called demand-oblivious routing, is sufficient to accommodate any admissible traffic matrix in the network with moderate link overload, so no prior information on demands is absolutely necessary for efficient traffic engineering. Demand-oblivious routing lends itself to distributed implementations, so it scales well. In this paper, we generalize demand-oblivious routing in a new way: we show that, in contrast to the distributed case, centralized demand-oblivious routing can eliminate link overload completely. What is more, our centralized scheme allows for optimizing the routes with respect to arbitrary linear or quadratic objective function. We realize, however, that a centralized scheme can become prohibitively complex, therefore, we propose a hybrid distributed-centralized algorithm, which, according to our simulations, strikes a good balance between efficiency, scalability and complexity.**

***Index Terms*—traffic engineering, adaptive routing, demand-oblivious routing, convex geometry**

I. Introduction

It is critical for a business-oriented service provider to constantly monitor, analyze and optimize the manner traffic is conveyed through its network in order to deliver the required service to customers, to avoid congestion that might cause disruptions, and to realize the largest profit margin attainable with the given network infrastructure. The art and science of satisfying these diverse requirements simultaneously is called Internet Traffic Engineering [1]. Given that user demands today are highly variable, one of the most important ingredients of traffic engineering is an adaptive routing technique that maps these permanently and unpredictably changing user demands to the physical network infrastructure effectively.

Historically, forwarding paths were optimized with respect to some measured and/or expected traffic matrix (if optimized at all) and over-provisioning of network capacity ensured that no congestion showed up when reality did not match expectations [2], [3]. This static routing strategy, however, has become more and more troublesome to operators recently, as networks are beginning to face a more dynamically changing environment. In consequence, various proposals have surfaced to reduce the significance of traffic matrices in intra-domain traffic engineering [4]–[9], or to straight out eliminate it [10]–[17]. A promising approach is to design a single routing that is suitable for handling not just one, but multiple traffic matrices [5], [6] or, at the extreme, *all* legitimate traffic matrices at the same time [18]–[28]. Using such a *demand-oblivious routing* algorithm eliminates the need to constantly re-calculate and re-deploy routing with the change of user demands.

Demand-oblivious routing lends itself to distributed implementations, because the amount of traffic sent to a forwarding path by a router only depends on information available locally at that router. This ensures simplicity and scalability. Moreover, simulation studies show that it can be unexpectedly efficient in the face of changing demands. Unfortunately, however, no theoretical upper bound exists on the capacity oversubscription, and hence, the congestion it might cause [20].

In this paper, we question the need for distributedness, at least in an intra-domain, unicast setting. A commercial network of our days is usually operated under the authority of a sophisticated central network management system (like HP OpenView or IBM Tivoli) that make available all the necessary monitoring and routing information, plus tested communications substrate, to deploy a centralized adaptive routing algorithm. We show that centralized demand-oblivious routing provides stability, hard QoS guarantees, optimizability with respect to arbitrary linear or quadratic objective function and, in contrast to the distributed case, absolutely no link oversubscription at all no matter how the users vary their demands. It turns out, however, that due to theoretical reasons a centralized scheme scales poorly. Therefore, we also propose a hybrid distributed-centralized routing algorithm, implementable with minimal central control, that combines the benefits of the two approaches for the price of sacrificing optimizability.

The rest of the paper is organized as follows. The next section is devoted to build a completely new geometric framework for demand-oblivious routing. The intention is to simplify the treatment and make the mathematical underpinnings more approachable. The notation, therefore, differs somewhat from that of the literature. Our framework allows us to generalize the notion of oblivious routing in novel ways: we describe the centralized case in Section III and a hybrid scheme in Section IV. Simulation results are discussed in Section V, related work is assessed in Section VI and finally, Section VII concludes the paper.

The first author was supported by the Janos Bolyai Fellowship of the Hungarian Academy of Sciences. The research was supported by NKTH-OTKA grant CNK77802.

II. A GEOMETRIC FRAMEWORK FOR ADAPTIVE ROUTING

We need some basic terms and definitions from convex geometry [29], [30]. A *polyhedron* P is an intersection of finitely many half-spaces: $P = \{x : Ax \leq b\} \subseteq \mathbb{R}^n$ where A is an $m \times n$ matrix and b is a column m-vector. A bounded polyhedron is called a *polytope*. A mapping $\mathcal{T} : \mathbb{R}^n \mapsto \mathbb{R}^m$ is called an *affine transformation* if it takes the form: $\mathcal{T} : y = Cx + d$ where C is an $m \times n$ matrix and d is a column m-vector. The affine transformation of a set X through $\mathcal{T}$ is defined as: $\mathcal{T}(X) = \{Cx + d : x \in X\}$. Affine transformations $\mathcal{T}$ have the following properties:

- $x \in X$ implies $\mathcal{T}(x) \in \mathcal{T}(X)$
- consequently, $X \subseteq Y$ implies $\mathcal{T}(X) \subseteq \mathcal{T}(Y)$
- an affine transformation of a polytope is again a polytope.

The special affine transformation $y = \alpha x, x \in X, \mathbb{R} \ni \alpha > 0$ is called the *scalar multiple of* X, denoted by αX.

In our framework, each user is provisioned a set of paths and the task of routing is to assign path flows for the users in accordance with the actual user demands and the capacity of network links. More precisely, suppose we are given the network topology as a directed graph $G(V, E)$ and a vector of positive, finite link capacities $c = [c_{ij} > 0 : (i, j) \in E]$ (see Table I for a summary on notations). Each user is associated with a unique source-destination pair $(s_k, d_k) : k \in \mathcal{K}$ and supplied with a set of static paths $\mathcal{P}_k$. Additionally, each user k independently presents its momentary demand θ_k at the source node s_k, and it is the task of the adaptive routing algorithm to vary the flows along the individual paths so that user traffic is conveyed from the source nodes to the destination nodes without disruptions. In other words, the routing algorithm adjusts the path flows $u_P : P \in \mathcal{P}_k, k \in \mathcal{K}$ with respect to the actual traffic matrix $[\theta_k : k \in \mathcal{K}]$ in a way as to avoid, or at least to minimize, link oversubscription. A routing is, consequently, represented by a vector of path flows: $u = [u_k : k \in \mathcal{K}] \in \mathbb{R}^{p_1} \times \mathbb{R}^{p_2} \times \ldots \times \mathbb{R}^{p_K} = \mathbb{R}^p$, where p_k is the number of paths for k and p is the number of all paths.

In geometric terms, a routing u is a point in the p-dimensional Euclidean space $\mathbb{R}^p$ and the requirement that it does not violate link capacities is equal to saying that u is included in the *flow polytope* M embedded into this space:

$$M = \{u : \sum_{k \in K} P_k u_k \leq c,\ u \geq 0\} \subset \mathbb{R}^p.$$

M contains all admissible routings, subject to link capacities and non-negativity constraints. In the sequel, we shall often use the following consequence of the above definitions: if M is the flow polytope for a network characterized by the link capacities c, then the flow polytope of the network whose link capacities were scaled by some α positive scalar to αc is the scalar multiple of M, αM.

Consider the affine transformation $\mathcal{T} : \mathbb{R}^p \mapsto \mathbb{R}^K$ that from a routing u generates the corresponding traffic matrix θ by summing up the path flows for each particular path of a user:

$$\theta = \mathcal{T}(u) = [\theta_k = \sum_{P \in \mathcal{P}_k} u_P : k \in \mathcal{K}] = Qu \ .$$

Table I: Notations

$G(V, E)$	a directed graph, with the set of nodes V ($\lvert V\rvert = n$) and the set of directed edges E ($\lvert E\rvert = m$)
c	the column m-vector of edge capacities
(s_k, d_k)	the set of source-destination pairs (or users) for $k \in \mathcal{K} = \{1, \ldots, K\}$
$\mathcal{P}_k$	the set of $s_k \rightarrow d_k$ paths assigned to some $k \in \mathcal{K}$
p_k	the number of paths for user k, $p_k = \lvert\mathcal{P}_k\rvert$
p	number of all paths, $p = \sum_{k \in \mathcal{K}} p_k$
P_k	an $m \times p_k$ matrix. The column corresponding to path $P \in \mathcal{P}_k$ holds the path-arc incidence vector of P
u_P	scalar, describing the traffic routed at path P
u_k	a column-vector, whose components are $u_P : P \in \mathcal{P}_k$ for some $k \in \mathcal{K}$ (whether we mean u_k or u_p will always be clear from the context)
u	a column vector representing a particular choice of u_Ps (a "routing")
θ_k	the demand/throughput of some user $k \in \mathcal{K}$
θ	a column K-vector representing a particular combination of throughputs (a "traffic matrix")
M	flow polytope, the set of path flows corresponding to $\mathcal{P}$, subject to non-negativity and capacity constraints
T	throughput polytope, the set of throughputs realizable over $\mathcal{P}$, subject to capacity constraints
$\mathcal{S}$	a routing function, $\mathcal{S} : \mathbb{R}^K \mapsto \mathbb{R}^p$
$\mathcal{S}_k$	the routing function corresponding to $k \in \mathcal{K}$, $\mathcal{S}_k : \mathbb{R}^K \mapsto \mathbb{R}^{p_k}$

Here, Q is a $K \times p$ matrix. The elements in kth row of Q are all 1 at positions $\sum_{l<k} p_l + 1, \ldots, \sum_{l \leq k} p_l$ and all zero otherwise. Mapping the flow polytope M through $\mathcal{T}$ gives the *throughput polytope* T [31]:

$$\mathbb{R}^p \supset M \xrightarrow{\mathcal{T}:\ \theta = Qu} T \subset \mathbb{R}^K \ .$$

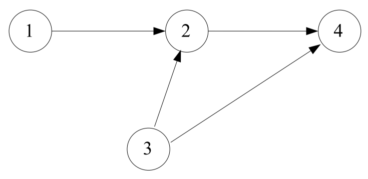
Accordingly, the throughput polytope contains all the traffic matrices realizable in the network by some properly chosen static routing without violating link capacities:

$$T = \{\theta : \exists u \in M \text{ so that } Qu = \theta\} \ .$$

Consequently, we call $\theta \in T$ *admissible*. As is the case with the flow polytope, multiplying the capacity vector by some positive real equals, in geometric terms, multiplying the polytope T with the same scalar. Unfortunately, computing T for arbitrary networks is a difficult problem, already intractable for moderate sized networks [31]. Thus, we shall use it only for theoretical modeling purposes.

The two geometrical objects, the flow polytope and the throughput polytope, will be central to our framework for demand-oblivious routing. A sample network and the corresponding polytopes are depicted in Fig. 1.

The other central objects in our framework are *routing functions*. Remember that the fundamental problem in adaptive routing is to adjust routing to the momentary traffic matrix. Therefore, one can represent a broad class of adaptive routing algorithms as a routing function $\mathcal{S}$ that maps from the throughput space to the flow space: $\mathcal{S} : \mathbb{R}^K \mapsto \mathbb{R}^p$. Given some traffic



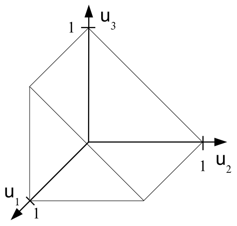
(a) topology

$(s_1, d_1) = (3, 4)$
$(s_2, d_2) = (1, 4)$

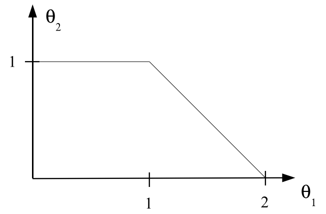
(b) users

$P_1 = \{(3, 4)\}$
$P_2 = \{(3, 2), (2, 4)\}$
$P_3 = \{(1, 2), (2, 4)\}$

(c) paths



(d) flow polytope (M)



(e) throughput polytope (T)

Figure 1: A sample network, source-destination pairs, a set of paths for each user and the corresponding flow and throughput polytopes. Edge capacities all equal 1 unit.

matrix θ, the corresponding routing is obtained by applying $\mathcal{S}$ to θ: $u = \mathcal{S}(\theta)$.

Note, however, that one can not set arbitrary routing functions. In particular, given a traffic matrix θ, the corresponding routing $\mathcal{S}(\theta)$ must realize precisely this θ. In other words, every θ is a fix point of the round-trip from the throughput space to the flow space and back (*throughput invariance*):

$$\forall \theta \in \mathbb{R}^K : \mathcal{T}(\mathcal{S}(\theta)) \equiv \theta \ .$$

In the simplest case, common to many networks of our days, each user is assigned a single path (e.g., the shortest path) and all traffic is routed to that path regardless of whether this causes congestion or not. In such cases, the component $\mathcal{S}_k$ of $\mathcal{S}$ corresponding to some user k is given as $u_k = \mathcal{S}_k(\theta) = \theta_k$.

In order to balance network load, traffic may be split between multiple paths. In this case, $\mathcal{S}_k$ takes the form $u_k = f_k \theta_k$ where f_k is a column vector of size p_k. To respect throughput invariance, elements of f_k must sum up to 1: $\forall k \in \mathcal{K} : 1^T f_k = 1$ (here, 1^T is a row vector of all 1s).

These routing functions are all implementable in a distributed fashion, as the flow sent to a path at a source node only depends on local information. We call a routing function *distributed*, if $\frac{\partial \mathcal{S}_k}{\partial \theta_l} = 0$ if $k \neq l$, wherever the derivative is defined.

In order to improve efficiency, one may allow the routing function to depend on global information as well. An important case of such *centralized* routing functions is when $\mathcal{S}$ takes the form of an affine transformation: $\mathcal{S}(\theta) = F\theta + g$ (component-wise we have $\mathcal{S}_k(\theta) = F_k\theta + g_k$), where F (F_k) is a $p \times K$ ($p_k \times K$) matrix and g (g_k) is a column vector of size p (p_k). Such $\mathcal{S}$ are called *simple affine routing functions*. Throughput invariance implies:

$$1^T F_{kl} = \delta_{kl} = \begin{cases} 1 & \text{if } k = l \\ 0 & \text{otherwise} \end{cases}, \qquad 1^T g_k = 0 \ ,$$

where F_{kl} denotes the lth column of F_k. Note that distributed routings discussed above can also be treated as affine routing functions by restricting F to block-diagonal matrices. Additionally, fixing g at zero yields a *linear* routing function (corresponding to traffic splitting ratios), while letting g to differ from zero yields an *affine distributed* routing function.

An important concept related to routing functions is the *domain*.

Definition 1: The domain $\mathcal{D}(\mathcal{S})$ for some $\mathcal{S}$ describes the set of traffic matrices that can be accommodated in the network by $\mathcal{S}$ without causing link oversubscription:

$$\mathcal{D}(\mathcal{S}) = \{\theta : \mathcal{S}(\theta) \in M\} \subset \mathbb{R}^K \ .$$

For an affine routing function $\mathcal{S}$, the domain is a polytope:

$$\mathcal{D}(\mathcal{S}) = \{\theta : \sum_{k \in \mathcal{K}} P_k(F_k\theta + g_k) \leq c, \ F\theta + g \geq 0\} \ .$$

Considering the sample network depicted in Fig. 1, a possible distributed adaptive routing might be to split the traffic of user 1 evenly between the two paths P_1 and P_2, and route all traffic of user 2 along its only available path. This setting corresponds to the distributed linear routing function $\mathcal{S}_1$:

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & 0 \\ \frac{1}{2} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

However, this routing would mistreat certain traffic matrices. The traffic matrix $\theta = [1, 1]^T$, although clearly admissible, overloads link $(2, 4)$ when fed into $\mathcal{S}_1$. In fact, any θ for which $\theta_1 + 2\theta_2 > 2$ causes overload at that link, which sets the domain of $\mathcal{S}_1$ as:

$$\mathcal{D}(\mathcal{S}_1) = \{\theta : \theta_1 + 2\theta_2 \leq 2, \ \theta \geq 0\} \ .$$

The question arises how to route the traffic matrices that fall outside of the domain of $\mathcal{S}_1$. It turns out that one can not do this using only distributed linear routing functions (note that it certainly *can* be done with other types of distributed adaptive routing schemes [12], [13]). A way to avoid overloading link $(2, 4)$ would be to concert traffic splitting ratios between the two users. Consider the centralized affine routing function $\mathcal{S}_2$:

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & -1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} + \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}$$

The domain of $\mathcal{S}_2$ turns out to be

$$\mathcal{D}(\mathcal{S}_2) = \{\theta : 1 \leq \theta_1 + \theta_2 \leq 2, \quad 0 \leq \theta_2 \leq 1\} \ ,$$

which covers the entire region $T \setminus \mathcal{D}(\mathcal{S}_1)$ as required.

It is tempting to combine $\mathcal{S}_1$ and $\mathcal{S}_2$ into a single *compound routing function*, which then would be suitable to accommodate any admissible traffic matrix $\theta \in T$ with causing no link overload at all. To do this, we associate different routing functions to different regions of the throughput space, so $\mathcal{S}$ takes the form $\mathcal{S} = \{(R^i, \mathcal{S}^i) : i \in \mathcal{I}\}$ where R^is give a disjunct partition of the throughput space, and we set $u = \mathcal{S}^i(\theta)$ whenever $\theta \in R^i$. Easily, the domain of a compound routing function is the union of the domains of its members. Care must be taken, however, to eliminate overlapping regions in order to maintain unambiguity of the compound routing function. In our example, we may choose either $\mathcal{S}_1$ or $\mathcal{S}_2$ to cover $\mathcal{D}(\mathcal{S}_1) \cap \mathcal{D}(\mathcal{S}_2)$, but not both. It is convenient to set R_1 to the entire domain of $\mathcal{S}_1$ and let R_2 be $T \setminus \mathcal{D}(\mathcal{S}_1)$. This way the compound routing function will be continuous over the entire domain, which prevents routing from exhibiting drastic fluctuations.

While one could certainly go on and generalize routing functions further (e.g., consider parameters other than θ or allow nonlinear functions, see e.g. [12], [13]) in order to capture a broader range of adaptive routing algorithms, we stop here. In the next section, we shall show that a compound affine routing function can already guarantee feasibility, continuity, adaptivity to arbitrary user demands, and optimizability.

III. Distributed and centralized demand-oblivious routing

In the literature, a demand-oblivious routing is usually associated with the following optimization problem:

$$\min_{\mathcal{S}} \alpha : \mathcal{S}(T) \subseteq \alpha M \ . \tag{1}$$

The minimal scalar α solving (1) is called the *oblivious ratio* and the corresponding routing function is called *demand-oblivious routing*. The interpretation of the optimization problem (1) is as follows. The set $\mathcal{S}(T)$ represents all the routings one can get when applying $\mathcal{S}$ to the set of admissible traffic matrices T, and we want this $\mathcal{S}(T)$ to fit into the set of feasible routings M as much as possible. The objective is, consequently, to minimize the factor needed to up-scale the flow polytope so that it eventually contains the entire set $\mathcal{S}(T)$. Recalling that αM corresponds to the flow polytope of a network whose capacity is adjusted to αc brings us to the interpretation: α signifies the maximal link over-utilization caused when routing *any* admissible traffic matrix over $\mathcal{S}$. This also implies that $\alpha \geq 1$.

Lemma 1: The optimization problem (1) can be posed equivalently using the notion of domains:

$$\min_{\mathcal{S}} \alpha : \frac{1}{\alpha} T \subseteq \mathcal{D}(\mathcal{S}) \ . \tag{2}$$

In the alternative definition, the oblivious ratio α is interpreted as follows: given an admissible traffic matrix $\theta \in T$ and a routing that solves (2), at least $\frac{1}{\alpha}\theta$ is guaranteed to be routable without violating link capacities. Consequently, the optimization problem (2) calls for finding a routing whose domain contains the largest portion of the set of admissible traffic matrices.

Proof: Direction (1)⇒(2): $\mathcal{S}(T) \subseteq \alpha M \Rightarrow \frac{1}{\alpha}\mathcal{S}(T) \overset{0 \in \mathrm{T}}{=} \mathcal{S}(\frac{1}{\alpha}T) \subseteq M \Rightarrow \frac{1}{\alpha}T \subseteq \mathcal{D}(\mathcal{S})$ by Definition 1.

Direction (2)⇒(1): $\frac{1}{\alpha}T \subseteq \mathcal{D}(\mathcal{S}) \Rightarrow T \subseteq \alpha\mathcal{D}(\mathcal{S}) \Rightarrow \mathcal{S}(T) \subseteq \mathcal{S}(\alpha\mathcal{D}(\mathcal{S})) \overset{0 \in \mathcal{D}(\mathcal{S})}{=} \alpha\mathcal{S}(\mathcal{D}(\mathcal{S})) \subseteq \alpha M$ by Definition 1. ■

If the demand-oblivious routing is searched for in the form of a simple, distributed affine routing function, then the optimization problem (1) can be stated as a linear program of polynomial size [18]. Note that [18] considers the arc-flow formulation, but the conversion to path-flow (and vice versa) is trivial [22]. The resultant traffic splitting ratios can be configured statically at the source nodes, so this scheme scales well. What is more, it proves surprisingly efficient: simulations on real-world and artificial ISP topologies show that the oblivious ratio almost always remains under 2, that is, it rarely causes more than 100% link oversubscription [18]. Nonetheless, it has also been pointed out that in certain directed networks the oblivious ratio grows without limits with the increase of the network size and user population [20].

Distributed demand-oblivious routing is scalable but it is not efficient enough for practical purposes. It is tempting to investigate what we find at the other end of the distributed-centralized spectrum. We have seen in the previous section that if one allows for compound routing functions that depend on global information, then, at least in our sample case, link over-utilization could be avoided. And indeed, the following result states that a centralized affine routing function always exists that, in contrast to distributed schemes, guarantees feasible routing for any admissible traffic matrix.

Theorem 1: For any network, there is a compound affine routing function $\mathcal{S} = \{(R^i, \mathcal{S}^i) : i \in \mathcal{I}\}$ with $\mathcal{I}$ finite so that $\mathcal{D}(\mathcal{S}) = T$ and $\mathcal{S}$ is continuous over T.

Proof: Let $f(u, \theta)$ be some linear or quadratic objective function and consider the multiparametric program

$$\begin{aligned} z(\theta) = \quad & \min f(u, \theta) \\ & \textstyle\sum_{k \in K} P_k u_k \leq c \\ & 1^T u_k = \theta_k \qquad \forall k \in \mathcal{K} \\ & u \geq 0 \end{aligned}$$

This is basically a conventional multicommodity flow problem, though, the right-hand-side now depends on the input parameter θ. It then follows from [32] that the solution can be written as a continuous, piecewise affine function of θ:

$$u = F^i\theta + g^i \text{ whenever } \theta \in R^i \ ,$$

where the regions R^i are disjunct (apart from the boundaries) polyhedral sets that give a polyhedral partition of T. ■

For a more in-depth exposition of this result, consult [33].

The significance of this theorem is that, theoretically, no prior information on traffic matrices is necessary to design a demand-oblivious routing function that guarantees feasibility. One solves a multiparametric linear or quadratic program, which, although computationally quite involving, is viable thanks to recent advances in geometric multiparametric programming algorithms [32], [34]. This phase can be done offline. The result is a set of polyhedral regions and affine functions. In operation, a central controller periodically scans the network, reads the momentary traffic demands θ, solves a series of polyhedron inclusion problems to find $i \in \mathcal{I}$ so that $\theta \in R^i$ and sets $u = F^i\theta + g^i$. An additional benefit is that centralized demand-oblivious routing allows for optimizing the routing function through specifying the objective $f(u, \theta)$. Both linear and quadratic objective functions are permitted. Plausible objectives would be to minimize delay or to minimize the maximum link utilization.

The advantages of centralized demand-oblivious routing are guaranteed feasibility and optimizability. The downside is that it can become arbitrarily complex: unfortunately there is no polynomial upper bound on the number of regions and individual simple routing functions needed to cover T. And, as simulation results confirm (see Section V), the size of the index set $\mathcal{I}$ quickly grows out of control. When $\mathcal{I}$ exceeds about 10^5, centralized demand-oblivious routing becomes impractical as the controller spends most of its time solving polyhedron inclusion problems trying to figure out which individual routing function to apply. Storage requirements too can become an issue. Finally, centralized operations itself could be seen as disadvantageous by some: such a scheme necessitates collecting all the network topology and link capacity information, plus the instantaneous user demands, at a central place. The fact that many of today's provider or enterprise networks are operated using some central network management software, which makes the required information readily available with no additional effort, mitigates this issue somewhat.

IV. A HYBRID APPROACH

Distributed demand-oblivious routing is scalable but can become arbitrarily inefficient. In contrast, centralized schemes are efficient but scale poorly. Obviously, a middle-ground should be found. In this section, we propose a hybrid distributed-centralized scheme with the intention to combine the advantages of the two approaches. Our hybrid scheme will require minimal central control, and it will rather rely on distributed operations to improve scalability while also attempting to be as efficient as possible. The price for improved scalability is, however, giving up optimizability.

The basic idea of geometric multiparametric programming algorithms is to first find some suitable affine function that solves the problem for some setting of the parameters, and then compute the (polyhedral) parameter region over which it is both optimal and feasible, cut away the present region and recurse into the rest of the parameter space. Unfortunately, the regions computed this way can be arbitrarily small, which often results in an immense number of individual routing functions in the output. Instead of this bottom-up approach, we propose a top-down method where the parameter region is supplied explicitly as input to the algorithm. We compute an affine routing function over the region, and if the oblivious ratio falls beyond a configured limit, we stop. Otherwise, the current throughput region is too large to be covered with a single routing function, so we generate a cut that divides the current throughput region into two with the intention to decrease the oblivious ratio over the resultant regions. The cut is chosen to be orthogonal to one of the axes, which makes it possible to construct a binary decision tree facilitating fast online search [35]. Next, the algorithm proceeds by recursing into the subregions. We decided to fall back upon to distributed routing functions inside the regions, so our scheme will require minimal central control, only to lookup the right region based on the actual demands.

In the rest of the section, we develop the tools needed to build our hybrid algorithm, describe it in detail and then discuss its advantages and disadvantages.

A. Demand-oblivious routing over an arbitrary region

So far, the oblivious ratio has been defined with respect to the throughput set T. Our algorithm will need to find it with respect to arbitrary regions, not just T, so first we extend the concept adequately and then we show how to solve the resultant optimization problem.

Definition 2: Given an arbitrary set of traffic matrices $X \subset \mathbb{R}^K$, the oblivious ratio $\alpha(X)$ with respect to X is defined as the optimal solution of the optimization problem:

$$\alpha(X) = \min_{\mathcal{S}} \alpha : \mathcal{S}(X) \subseteq \alpha M \ . \tag{3}$$

Note that $\alpha(X)$ is equivalent to the conventional notion of oblivious ratio when $X = T$. In other cases, $\alpha(X)$ depends on X: it may be smaller than 1 when $X \subset T$, it may be unbounded if X is unbounded, or it might not be defined at all if $X \nsubseteq \mathbb{R}^K_+$. Further note that Lemma 1 does not generalize to arbitrary regions.

Below, we extend the linear programming-based method of [18] to solve (3). The difference will be that we allow for arbitrary polyhedral regions X, not just T. Herein, we briefly reproduce the treatment from that paper with some trivial modifications to handle this extension. For the rest of this paper, we shall assume that X is polyhedral: $X = \{\theta : H\theta \leq h\}$, where H is a $q \times K$ matrix and h is a column q-vector. Additionally, we limit our attention to distributed affine routing functions that take the form $\mathcal{S}_k(\theta) = f_k\theta_k + g_k$ for all $k \in \mathcal{K}$.

For start, expanding (3) yields:

$$\min \alpha : \quad 1^T f_k = 1, 1^T g_k = 0 \qquad \forall k \in \mathcal{K} \tag{4}$$

$$\forall (i,j) \in E : \frac{\sum_{k \in K} P_k^{ij}(f_k\theta_k + g_k)}{c_{ij}} \leq \alpha \quad \forall \theta \in T \cap X \tag{5}$$

where P_k^{ij} denotes the row of the path-arc incidence matrix P_k corresponding to link (i,j). The above is not a linear program, because constraint (5) is present for every $\theta \in T \cap X$, yielding infinitely many constraints. Therefore, we organize (5) into a slave problem for each $(i,j) \in E$:

$$\alpha \geq \max \frac{\sum_{k \in \mathcal{K}} P_k^{ij}(f_k\theta_k + g_k)}{c_{ij}} \tag{6}$$

$$\sum_{k \in \mathcal{K}} P_k u_k \leq c, \ H\theta \leq h \tag{7}$$

$$1^T u_k = \theta_k, u_k \geq 0 \quad \forall k \in \mathcal{K} \tag{8}$$

Dualizing (6)–(8) and collecting all dual slave problems yields a single giant linear program:

$$\begin{aligned}
\min \alpha: \quad & 1^T f_k = 1, 1^T g_k = 0 && \forall k \in \mathcal{K} \quad (9)\\
\forall (i,j) \in E: \{ & w^{ij} c + \lambda^{ij} h + \frac{\sum_{k \in K} P_k^{ij} g_k}{c_{ij}} \le \alpha && (10)\\
& w^{ij} P_k \ge 1^T \beta_k^{ij} && \forall k \in \mathcal{K} \quad (11)\\
& \lambda^{ij} H_k - \beta_k^{ij} \ge \frac{P_k^{ij} f_k}{c_{ij}} && \forall k \in \mathcal{K} \quad (12)\\
& w^{ij}, \lambda^{ij} \ge 0 \qquad \} && (13)
\end{aligned}$$

where H_k is the kth column of H, w^{ij} and λ^{ij} are the duals to the constraints (7) and β_k^{ij} to constraints (8), specific to the dual subproblem corresponding to each $(i,j) \in E$. Note that the additional constraints $\forall k \in \mathcal{K} : f_k \ge 0, g_k = 0$ can be added to compute linear distributed routing functions.

B. Finding the best cut

Suppose that we are given the throughput region X (where X has already been trimmed down by taking the intersection with T) and we find the oblivious ratio $\alpha(X)$ too large in this region, so we decide to cut the region into two and recurse into the resultant subregions. Albeit there are infinitely many cutting planes we can choose from, it is worthwhile to pick one that is orthogonal to one of the axes, that is, can be written in the form $\theta_k \le t$ for some $k \in \mathcal{K}$. This way, the regions become K-dimensional hyper-rectangles, which simplifies the mathematical treatment and facilitates for building a fast orthogonal decision tree for the online phase of the algorithm. On the other hand, we rule out many non-orthogonal cutting planes that may, or may not, produce higher-quality cuts.

Below, we show a method to search for the orthogonal cut $\theta_k \le t$ that, for any selection of $k \in \mathcal{K}$ and any value of t, reduces the oblivious ratio the most. Choose some $k \in \mathcal{K}$ and let $\tau_k = \min_{\theta \in X} \theta_k$ and $T_k = \max_{\theta \in X} \theta_k$. Additionally, consider the set $X(t) = X \cap \{\theta : \theta_k \le t\}$ and define a function $\mu : \mathbb{R} \mapsto \mathbb{R}$ that to the parameter t orders the oblivious ratio corresponding to $X(t)$, $\mu(t) = \alpha(X(t)) = \min_{\mathcal{S}} \alpha : \mathcal{S}(X(t)) \subseteq \alpha M$, where $\mathcal{S}$ is distributed linear or affine. The other side of the cut, $\theta_k \ge t$, defines the region $Y(t) = X \cap \{\theta : \theta_k \ge t\}$ and the corresponding function $\nu(t) = \alpha(Y(t))$. Note that $X(T_k) = Y(\tau_k) = X$, so $\mu(T_k) = \nu(\tau_k) = \alpha(X)$.

Theorem 2: The function $\mu(t)$ is well-defined, monotonically *increasing* and continuous on the domain $[\tau_k, T_k]$. The function $\nu(t)$ is well-defined, monotonically *decreasing* and continuous on the domain $[\tau_k, T_k]$.

The next Lemma implies definedness and monotonicity.

Lemma 2: Given some $Z \in \mathbb{R}^K$ for which $\alpha(Z)$ exists, for any $Y \subseteq Z$: $\alpha(Y)$ exists and $\alpha(Y) \le \alpha(Z)$.

Proof: Easily, if some $\mathcal{S}_Z$ routes some region Z, then it is feasible in (9)–(13) for any $Y \subseteq Z$. It is also true that the optimal solution can only be smaller, so $\alpha(Y) \le \alpha(Z)$. ■

Next, we show right-continuity of $\mu(t)$ and left-continuity of $\nu(t)$. Note that our proofs are a bit more generic, in that we prove continuity over any sequence of convex sets whose elements are sufficiently "close" to each other: $\forall X, \forall \epsilon > 0, \exists X' : d_H(X, X') < \epsilon$, where d_H denotes the Hausdorff metric.

Lemma 3: For any $\delta > 0$ there is an $\epsilon > 0$ so that on any *extended* (convex) set $X' : X \subset X', d_H(X, X') \le \epsilon$, it holds that $\alpha(X') - \alpha(X) \le \delta$.

Proof: Let $\gamma > 0$ be such that $\sum_{k \in \mathcal{K}} \sqrt{p_k} P_k \underline{1} \le \gamma c$ (where $\underline{1}$ is a vector of all 1s) and consider some $\mathcal{S} : \alpha(X) = \alpha(X, \mathcal{S}) \overset{\text{def}}{=} \min_\alpha \alpha : S(X) \subseteq \alpha M$. For each $k \in \mathcal{K}$, let $\tau_k = \min_{\theta \in X} \theta_k$, $T_k = \max_{\theta \in X} \theta_k$, $v_k = \mathcal{S}_k(\tau_k)$ and $V_k = \mathcal{S}_k(T_k)$. Let $\epsilon = \min\left(\frac{\delta}{\gamma}, \min_k \tau_k\right)$. Search $v_k' : 1^T v_k' = \tau_k - \epsilon, v_k' \ge 0, |v_{kp} - v_{kp}'| \le \epsilon$ and $V_k' : 1^T V_k' = T_k + \epsilon, V_k' \ge 0, |V_{kp} - V_{kp}'| \le \epsilon$. This can always be done. Construct the routing function $\mathcal{S}'(\theta) = f_k' \theta_k + g_k', k \in \mathcal{K}$, $f_k' = \frac{V_k' - v_k'}{T_k - \tau_k + 2\epsilon}$, $g_k' = v_k' - \frac{(\tau_k - \epsilon)(V_k' - v_k')}{T_k - \tau_k + 2\epsilon}$. One easily sees that $\mathcal{S}'$ is a routing function on any $X' \supset X, d_H(X, X') \le \epsilon$, i.e., $\forall \theta \in X' : \mathcal{S}'(\theta) \ge 0$ and throughput invariance holds. Additionally, $\forall k \in \mathcal{K} : d_H(\mathcal{S}_k(X), \mathcal{S}_k'(X')) \le \sqrt{p_k}\epsilon$, hence $\forall \theta \in X' : \exists \hat{\theta} \in X$ so that $\mathcal{S}_k'(\theta) \le \mathcal{S}_k(\hat{\theta}) + \sqrt{p_k}\epsilon\underline{1}$. Next, we show that $\mathcal{S}(X) \subseteq \alpha M \Rightarrow \mathcal{S}'(X') \subseteq (\alpha + \delta) M$. Observe $\forall \theta \in X' : \sum_{k \in \mathcal{K}} P_k \mathcal{S}_k'(\theta) \le \sum_{k \in \mathcal{K}} P_k(\mathcal{S}_k(\hat{\theta}) + \sqrt{p_K}\epsilon\underline{1}) \le \alpha c + \gamma \epsilon c \le (\alpha + \delta) c$. Consequently, $\alpha(X') \le \alpha(X) + \delta$, which completes the proof. ■

Finally, continuity comes from the Lemma below.

Lemma 4: For any $\delta > 0$ there is an $\epsilon > 0$ so that on any *reduced* (convex) set $X' : X' \subset X, d_H(X, X') \le \epsilon$, it holds that $\alpha(X) - \alpha(X') \le \delta$.

Proof: Choose $\epsilon > 0$ as in Lemma 3 and consider any $X' \subset X, d_H(X, X') \le \epsilon$. Note that $\alpha(X')$ is defined since $X' \subset X$. Let $\mathcal{S} : \alpha(X') = \alpha(X', \mathcal{S})$ and use Lemma 3 to obtain a routing function $\mathcal{S}'$ on the extension of X' to X. Now, we have $\alpha(X') \le \alpha(X) \le \alpha(X, \mathcal{S}')$, and $\alpha(X, \mathcal{S}') - \alpha(X') \le \delta$, hence $\alpha(X) - \alpha(X') \le \delta$, which completes the proof. ■

Our task is now to find a value t for which the oblivious ratios of the two subregions, in particular the larger one, is minimal. As both $\mu(t)$ and $\nu(t)$ are continuous on $[\tau_k, T_k]$, so is their maximum. We conclude that $\max(\mu(t), \nu(t))$ has a unique minimum on $[\tau_k, T_k]$ and this occurs when $\mu(t) = \nu(t)$. Note that neither $\mu(t)$ nor $\nu(t)$ changes *strictly*, thus the t at which the minimum occurs is not necessarily unique, and it is not guaranteed that we can indeed decrease the oblivious ratio using an orthogonal cut along this dimension. Some other dimension might produce a better cut in such cases.

Our algorithm to generate a cut to subdivide a region along some dimension k employs binary search to find $t \in [\tau_k, T_k] : \mu(t) = \nu(t)$. At every iteration of the search, we compute $\mu(t)$ and $\nu(t)$ for the present value of t (this amounts to solving the linear program (9)–(13) twice). If $|\mu(t) - \nu(t)| < \epsilon$, we stop. Otherwise, we continue the binary search until we find a t for which $\mu(t)$ is (approximately) equal to $\nu(t)$. Doing this for all $k \in \mathcal{K}$ yields an orthogonal cutting plane that is expected to give the largest reduction in the oblivious ratio (see the pseudo-code in Algorithm 1).

C. The hybrid algorithm

Our hybrid distributed-centralized demand-oblivious routing algorithm is a synthesis of the above ideas. The procedure is simple: we start the algorithm from the entire set of admissible

Algorithm 1 Generate an orthogonal cut on region X

```
function BEST_CUT($X$)
    for $l \in \mathcal{K}$
        $\tau_l \leftarrow \min_{\theta \in X} \theta_l$; $T_l \leftarrow \max_{\theta \in X} \theta_l$
        $t_l \leftarrow$ BinarySearch($t \in [\tau_l, T_l] : \mu(t) = \nu(t)$)
        $A_l \leftarrow \mu(t_l)$
    end for
    $k \leftarrow \mathrm{argmin}_{l \in \mathcal{K}} A_l$
    return $(k, t_k)$
end function
```

traffic matrices T, and in each iteration we try to improve the oblivious ratio. First, we compute an oblivious routing for the present region. If we are contented with the result, that is, the oblivious ratio falls beyond a configured limit, say, L, we stop. Otherwise, we use Algorithm 1 to divide the current region into two along an orthogonal cut and recurse into the resultant subregions. The algorithm, starting from the "conventional" oblivious ratio, is expected to improve it in every iteration, while it also generates the compound routing functions with the corresponding regions. A pseudo-code for the hybrid algorithm is given in Algorithm 2.

Algorithm 2 Hybrid demand-oblivious routing algorithm

```
HYBRID_OBLIVIOUS_ROUTING($T$)
function HYBRID_OBLIVIOUS_ROUTING($X$)
    $(\alpha, \mathcal{S}) \leftarrow \min_{\mathcal{S}} \alpha : \mathcal{S}(X) \subseteq \alpha M$
    if $\alpha \leq L$ then
        store $(X, \mathcal{S})$ and return
    end if
    $(k, t_k) \leftarrow$BEST_CUT($X$)
    HYBRID_OBLIVIOUS_ROUTING($X \cap T \cap \{\theta : \theta_k \leq t_k\}$)
    HYBRID_OBLIVIOUS_ROUTING($X \cap T \cap \{\theta : \theta_k \geq t_k\}$)
end function
```

In the present form, Algorithm 2 iterates until the oblivious ratio falls beyond the limit L globally. Note that in some cases the algorithm may never stop, as it is not guaranteed that the oblivious ratio reduces in every iteration. To avoid this, one may terminate the recursion after the number of regions exceeds a predefined value. This makes the algorithm's complexity polynomial, as all the steps are of polynomial complexity too.

The result is a compound routing function $\mathcal{S} = \{(R^i, \mathcal{S}^i) : i \in \mathcal{I}\}$, in which all the individual routing functions take the form $u_k = f_k^i \theta_k + g_k^i$. Thus, our routing function lends itself readily to distributed implementation, and only minimal central control is required to pick the right region $i \in \mathcal{I}$ and the appropriate settings of f_k^i and g_k^i. For this, the central controller periodically determines the actual traffic matrix and checks whether θ still resides in the current region R^i. If yes, no action is taken. Otherwise, the controller searches for a new region and downloads the new settings of f_k^i and g_k^i to the routers. In our hybrid scheme the search simplifies significantly, thanks to the fact that the regions are hyper-rectangles. First, the controller checks whether the cut generated in the first iteration, say, $\theta_{k_1} \leq t_1$, holds for θ. Half of the regions is beneath and the other half is beyond this cut, which immediately rules out half of $\mathcal{I}$. In the next step, the controller checks the cut arising from the second iteration, $\theta_{k_2} \leq t_2$, and so on, in each step halving the remaining $\mathcal{I}$. Organizing the regions into such a decision tree improves the online complexity to $O(\log |\mathcal{I}|)$ (from $O(|\mathcal{I}|)$). For more information on orthogonal decision trees, consult [35]. Note, nonetheless, that in contrast to the centralized scheme the hybrid algorithm does not guarantee continuity over the boundaries of the regions (albeit it does so *inside*), neither it allows for optimizing the routing.

V. Simulation studies

Our first round of simulations focused on the performance of conventional, distributed demand-oblivious routing [18]. We evaluated two performance attributes: the oblivious ratio α as computed by (1), plus a new, geometric characteristic $\eta(\mathcal{S})$ we call *link overload probability*, which corresponds to the chance that a randomly chosen admissible traffic matrix, when routed by $\mathcal{S}$, causes link over-utilization at some parts of the network:

$$\eta(\mathcal{S}) = \frac{\mathrm{Vol}(T \setminus \mathcal{D}(\mathcal{S}))}{\mathrm{Vol}(T)} = 1 - \frac{\mathrm{Vol}(\mathcal{D}(\mathcal{S}))}{\mathrm{Vol}(T)} \ ,$$

where $\mathrm{Vol}(X)$ denotes the volume of the set X.

We used the NSFNET Phase II topology [36], K source-destination pairs were chosen according to the random bimodal distribution (similarly to [18]), and p maximally node-disjoint paths were computed for the users. We examined increasingly complex scenarios by growing K from 1 to 9 (unfortunately, volume computation becomes intractable for larger values of K). The results are shown in Fig. 2 for $p = 2$ and $p = 3$. What is interesting is not that the oblivious ratio seems to increase as the number of users grow (this trend mostly vanishes for higher Ks), but rather that even for very small values of α, say, 1.5, there is an overwhelming chance (70-90%) that a randomly picked traffic matrix will overload some links in the network. This suggests that the oblivious ratio, used extensively in the literature, is not really a good measure to characterize the performance of oblivious routing.

The centralized scheme, however, avoids any forms of link over-subscription, at the price of increasing complexity. Our results (see Fig. 3), obtained on the same scenarios as above, indicate that the complexity of centralized demand-oblivious routing explodes with the increase of user population.

Finally, we evaluated the performance of the hybrid demand-oblivious routing algorithm in both real-world and artificial networks. First, we used the ISP data maps from the Rocketfuel dataset [37]. We applied the same method as in [18] to obtain approximate POP-level topologies: we collapsed the topologies so that nodes correspond to cities, we eliminated leaf-nodes and we set link capacities inversely proportional to the link weights. Source-destination pairs were chosen according to the bimodal distribution and paths were provisioned maximally node-disjoint ($p = 2$). We conducted experiments when the number of users was small ($K = 7$), medium ($K = 21$) and large ($K = 35$) compared to the size

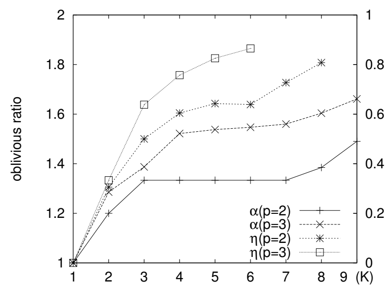


Figure 2: Oblivious ratio and link overload probability for the conventional (distributed) scheme for $p = 2$, resp. $p = 3$, paths per user.

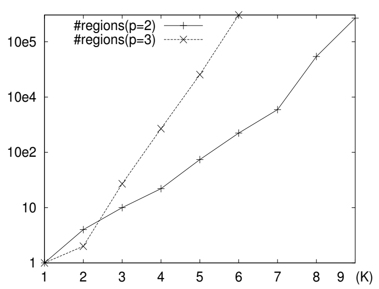


Figure 3: Number of regions in the centralized scheme for $p = 2$, resp. $p = 3$, paths per user. Note the log-scale on the y axis.

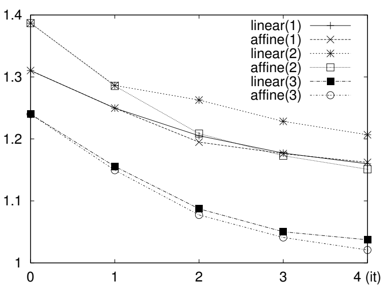


Figure 4: Oblivious ratio of the hybrid algorithm at every iteration in three experiments on AS 3257, $K = 7$.

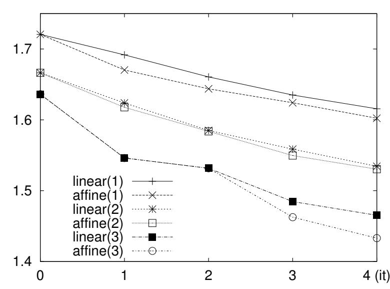


Figure 5: Oblivious ratio of the hybrid algorithm at every iteration in three experiments on AS 3257, $K = 21$.

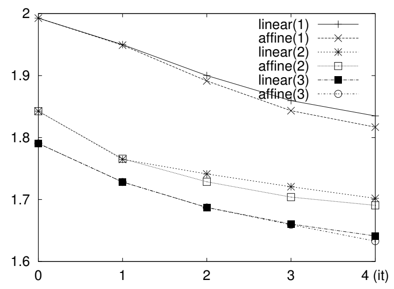


Figure 6: Oblivious ratio of the hybrid algorithm at every iteration in three experiments on AS 3257, $K = 35$.

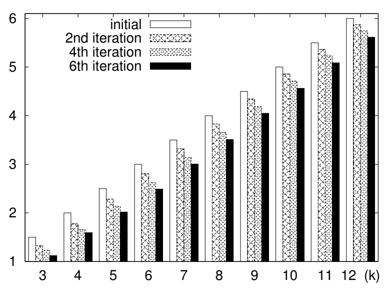


Figure 7: Oblivious ratio for artificial graphs for increasing values of k after the initial, 2nd, 4th and 6th iteration.

of the network, with both linear and affine distributed routing functions. During running the hybrid algorithm, we registered the oblivious ratio of the routing function produced at every iteration. Here, we only include some typical results for AS 3257 (Tiscali, Europe) in Fig. 4, 5 and 6.

We observe that the hybrid algorithm, taking off from the "conventional" oblivious routing obtained at the initial iteration, produces steadily more efficient compound routing functions in each step. In some cases, our algorithm achieved significant improvement in the oblivious ratio: for $K = 7$, case (2), it reduced the ratio from 1.4 to less than 1.2, which amounts to basically halving the overload. In other cases, the improvement is not that significant. In addition, distributed affine routing seems slightly more efficient than linear routing, but the improvement does not seem to be worth the added implementation complexity.

To justify these claims, we conducted experiments on artificial networks as well. In [20, Theorem 7.1], Azar *et al.* give a directed graph of $\binom{k}{2} + k + 1$ nodes and $\binom{k}{2}$ source-destination pairs on which oblivious routing works spectacularly poorly. In particular, the oblivious ratio is at least $\binom{k}{2}$. Running the hybrid scheme on these graphs (see Fig. 7) reveals that even though there is no theoretical guarantee on improvement, our algorithm indeed improves the oblivious ratio even in contrived examples, and only a few regions are enough to produce a better quality demand-oblivious routing.

VI. Related work

Demand-oblivious routing has extensive literature [18]–[28]. Räcke gives a method with polylogarithmic oblivious ratio in undirected graphs [19]; Azar *et al.* prove that no such worst-case bound exists for directed graphs [20]; and [24] presents a randomized algorithm of polylogarithmic oblivious ratio with high probability provided that the demand-distribution is assumed known. A different approach is in [26]. All of these papers treat the distributed case. In contrast, we allow centralized implementations as well. This improves the worst case oblivious ratio to 1. Regarding optimizability and oblivious routing, see [27], [28].

The geometric framework for investigating fundamental properties of networks was introduced by the authors in [31]. The hybrid distributed-centralized algorithm is based on [35]. Similarly to them, we build an orthogonal decision tree for fast online search. However, [35] is limited to hyper-cubic regions whose size is halved in each iteration, while our algorithm calculates optimal orthogonal cuts.

Intra-domain traffic engineering algorithms proposed recently are DATE [14], TEXCP [16], REPLEX [17] and COPE [9]. Perhaps the closest to ours is COPE, which combines the advantages of prediction-based traffic engineering and demand-oblivious routing. Our scheme, in contrast, does not need predicted state.

VII. Conclusions

Demand-oblivious routing is, loosely speaking, a way to set traffic splitting ratios at routers statically so that congestion is minimized, no matter which combination of demands the users pose to the network. In this paper, we generalized this scheme in various ways. First, we showed that if one allows splitting ratios to be set dynamically, depending on local as well as global information, then congestion can be eliminated completely. An added benefit of this centralized demand-oblivious routing scheme is that it allows for optimizing the routing function. However, a centralized scheme necessarily raises grave implementation issues, ranging from prohibitive online complexity and offline storage requirements to instability caused by unpredictable communication delays between the routers and the central controller. Nonetheless, in small enterprise networks, transit ISPs or wherever applicable, a centralized scheme promises with significant boost in network profit by better exploiting the costly network infrastructure and delivering better service.

It turns out that the geometric algorithms for computing the optimal routing function are subject to the explosion of complexity often called as *the curse of dimensionality*. To avoid this trap, we proposed a hybrid distributed-centralized scheme, which reduces the involvement of the central controller to picking the right distributed routing function. Extensive simulations showed that subdividing the throughput space to only a couple of regions already reduces congestion significantly. Unfortunately, appealing properties of the centralized scheme are lost, like feasibility, continuity and optimizability, and future research is necessary to recover at least some of these.

References

[1] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, "Overview and principles of Internet traffic engineering." RFC 3272, May 2002.

[2] D. G. Cantor and M. Gerla, "Optimal routing in a packet-switched computer network," *IEEE Trans. Comp.*, vol. 23, no. 10, pp. 1062–1069, 1974.

[3] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," *IEEE Communications Magazine*, vol. 40, pp. 118–124, Oct 2002.

[4] M. Roughan, M. Thorup, and Y. Zhang, "Traffic engineering with estimated traffic matrices," in *IMC '03*, pp. 248–258, 2003.

[5] C. Zhang, Y. Liu, W. Gong, J. Moll, and R. D. Towsley, "On optimal routing with multiple traffic matrices," in *INFOCOM 2005*, vol. 1, pp. 607–618, 2005.

[6] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS weights in a changing world," *IEEE J. Sel. Area Comm.*, vol. 20, pp. 756–767, May 2002.

[7] S. Suri, M. Waldvogel, and P. R. Warkhede, "Profile-based routing: a new framework for MPLS traffic engineering," in *Quality of Future Internet Services* (F. Boavida, ed.), vol. 2156 of *LNCS*, Springer, 2001.

[8] D. Medhi, "Multi-hour, multi-traffic class network design for virtual path-based dynamically reconfigurable wide-area ATM networks," *IEEE/ACM Trans. Network.*, vol. 3, no. 6, pp. 809–818, 1995.

[9] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg, "COPE: traffic engineering in dynamic networks," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 99–110, 2006.

[10] D. P. Bertsekas, "Dynamic behavior of shortest path routing algorithms for communication networks," *IEEE Trans. on Automatic Control*, vol. 27, pp. 60–74, 1982.

[11] M. Chiang, S. Low, A. Calderbank, and J. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proceedings of the IEEE*, vol. 95, pp. 255–312, Jan 2007.

[12] F. Kelly and T. Voice, "Stability of end-to-end algorithms for joint routing and rate control," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 2, pp. 5–12, 2005.

[13] P. Key, J. J. T. Aveue, L. Massoulié, and J. J. T. Aveue, "Combining multipath routing and congestion control for robustness," in *Conference on Information Sciences and Systems*, 2006.

[14] J. He, M. Bresler, M. Chiang, and J. Rexford, "Towards robust multi-layer traffic engineering: Optimization of congestion control and routing," *IEEE J. Sel. Area Comm.*, vol. 25, pp. 868–880, June 2007.

[15] C. M. Lagoa, H. Che, and B. A. Movsichoff, "Adaptive control algorithms for decentralized optimal traffic engineering in the internet," *IEEE/ACM Trans. Network.*, vol. 12, no. 3, pp. 415–428, 2004.

[16] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the tightrope: Responsive yet stable traffic engineering," in *ACM SIGCOMM'05*, August 2005.

[17] S. Fischer, N. Kammenhuber, and A. Feldmann, "REPLEX: dynamic traffic engineering based on wardrop routing policies," in *Proceedings of CoNEXT'06*, pp. 1–12, 2006.

[18] D. Applegate and E. Cohen, "Making intra-domain routing robust to changing and uncertain traffic demands: understanding fundamental tradeoffs," in *Proceedings of SIGCOMM '03*, pp. 313–324, 2003.

[19] H. Räcke, "Minimizing congestion in general networks," in *FOCS '02*, pp. 43–52, 2002.

[20] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Räcke, "Optimal oblivious routing in polynomial time," *J. Comput. Syst. Sci.*, vol. 69, no. 3, pp. 383–394, 2004.

[21] J. Wellons and Y. Xue, "Oblivious routing for wireless mesh networks," in *ICC '08*, pp. 2969–2973, May 2008.

[22] Y. Li, B. Bai, J. J. Harms, and R. Holte, "Stable and robust multipath oblivious routing for traffic engineering," in *International Teletraffic Congress*, vol. 4516 of *Lecture Notes in Computer Science*, pp. 129–140, Springer, 2007.

[23] D. Applegate, L. Breslau, and E. Cohen, "Coping with network failures: routing strategies for optimal demand oblivious restoration," *SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 1, pp. 270–281, 2004.

[24] M. Hajiaghayi, J. Kim, T. Leighton, and H. Räcke, "Oblivious routing in directed graphs with random demands," in *STOC '05*, pp. 193–201, 2005.

[25] N. Bansal, A. Blum, S. Chawla, and A. Meyerson, "Online oblivious routing," in *SPAA '03*, pp. 44–49, 2003.

[26] B. Towles and W. Dally, "Worst-case traffic for oblivious routing functions," in *SPAA '02*, pp. 1–8, 2002.

[27] P. Harsha, T. Hayes, H. Narayanan, H. Räcke, and J. Radhakrishnan, "Minimizing average latency in oblivious routing," in *SODA '08*, (Philadelphia, PA, USA), pp. 200–207, 2008.

[28] B. Towles, W. Dally, and S. Boyd, "Throughput-centric routing algorithm design," in *SPAA '03*, pp. 200–209, 2003.

[29] G. Ziegler, *Lectures on Polytopes*, vol. 152 of *Graduate Texts in Mathematics*. Springer, 1998.

[30] B. Grünbaum, *Convex Polytopes*. John Wiley & Sons, 1967.

[31] G. Rétvári, J. J. Bíró, and T. Cinkler, "Fairness in capacitated networks: A polyhedral approach," in *INFOCOM'07*, vol. 1, pp. 1604–1612, May 2007.

[32] F. Borrelli, A. Bemporad, and M. Morari, "Geometric algorithm for multiparametric linear programming," *Journal of Optimization Theory and Applications*, vol. 118, pp. 515–540, September 2003.

[33] G. Rétvári and G. Németh, "On optimal multipath rate-adaptive routing," 2010. submitted to ISCC 2010.

[34] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, pp. 3–20, January 2002.

[35] T. Johansen, A. Grancharova, and R. Grancharova, "Approximate explicit constrained linear model predictive control via orthogonal search tree," *IEEE Trans. Automatic Control*, vol. 48, pp. 810–815, 2003.

[36] B. Chinoy and H. W. Braun, "The national science foundation network." Tech. Rep., CAIDA, available online: http://www.caida.org/outreach/papers/1992/nsfn/nsfnet-t1-technology.pdf, Sep 1992.

[37] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "Inferring link weights using end-to-end measurements," in *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pp. 231–236, 2002.