Rate-adaptive Multipath Routing: Distributed, Centralized, and Hybrid Architectures

Gábor Németh[a,*], Gábor Rétvári[b]

[a] *Inter–University Centre for Telecommunications and Informatics*
Kassai u. 26., Debrecen, Hungary
[b] *Dept. of Telecommunication and Media Informatics*
Budapest University of Technology and Economics
Magyar tudósok körútja 2, H-1117, Budapest, Hungary

Abstract

With the increasing volume and volatility of Internet traffic, the need for adaptive routing algorithms has become compelling lately. An adaptive routing algorithm controls the rate at which traffic is placed on forwarding paths in concert with the actual user demands, making it possible to avoid congestion even when no information on expected traffic is available. In this paper, we present a new model for rate-adaptive multipath routing, which allows one to analyze distributed, centralized, and hybrid routing architectures within a single framework, and to develop quantitative as well as qualitative arguments regarding their optimality, stability, and realizability. By a novel generalization of oblivious routing, we present a centralized algorithm with provable optimality, and we arrive at the conclusion that congestion can be completely eliminated even if routing decisions are completely pre-computed. We find, though, that the complexity of the centralized scheme can become exponential. Therefore, we develop a hybrid distributed-centralized algorithm that combines the simplicity of distributed algorithms with the efficiency of centralized ones, and we provide numerical studies demonstrating that the hybrid scheme performs well in a broad selection of realistic scenarios.

Keywords: rate-adaptive routing, multipath routing, traffic engineering, convex geometry, centralized routing, distributed routing, hybrid architecture

*Corresponding author
Email addresses: nemethgab@tmit.bme.hu (Gábor Németh), retvari@tmit.bme.hu (Gábor Rétvári)

1. Introduction

Traffic engineering is the art and science of monitoring, analyzing, and optimizing the way traffic is conveyed through a service provider network, in order to deliver the required user experience to customers, to avoid congestion that might cause service disruptions, and to materialize the largest profit margin attainable with the installed network infrastructure [1]. The most important means by which these diverse goals can be realized is a routing algorithm, responsible for mapping traffic demands to the physical network infrastructure.

The characteristics of the traffic that enters and leaves the network is a crucial factor affecting the design of routing algorithms. In cases when the traffic matrix is reasonably static for a longer period of time, historical measurements (and traffic matrices constructed based on them [2]), data mining techniques, and behavioral analyses can be used to make accurate predictions about future demands [3] and provision forwarding paths statically with respect to the predicted traffic characteristics [4, 5].

Internet traffic, however, tends to exhibit substantial variation over a wide range of timescales due to various reasons beyond the control of the operator [6]. The instabilities and oscillations in the inter-domain routing ecosystem [7, 8, 9], the emergence of overlay networks and peer-to-peer applications, traffic bursts caused by flash crowds, the emergence of communications protocols without rate control, and the rapidly changing Internet application landscape, are all factors making accurate traffic matrix estimation and, correspondingly, provisioning static routes increasingly hard [10, 11, 12]. The problem is that when traffic demands change abruptly on a small timescale, the traffic engineering algorithm does not have time to re-adjust static forwarding paths appropriately, leading to congestion, increased packet loss, delay, and jitter, all in all, a deterioration of user experience. Accordingly, traffic engineering algorithms have gradually evolved from what was initially a predominantly static setting [4, 5], through optimizing for multiple traffic matrices [13, 14, 15, 16, 17, 12] towards fully adaptive schemes [18, 19, 20, 21, 22, 23, 24, 25].

A distributed routing algorithm applies local information exclusively for making routing decisions, thus, at first sight, it might be favorable over a centralized one. After all, a routing algorithm relying on the global state must permanently monitor the network and necessitates an additional communications mechanism to feed decisions back to the routers, raising substantial scalability and robustness issues. A distributed scheme, on the other hand, does not need any of these, and hence can be truly scalable and free of single points of failure. Unfortunately, a distributed algorithm, having access to only a limited scope of the network state, might be less efficient [26] and might also induce route oscillations arising from the conflicting interventions of distant actors [27, 19]. Thus, most distributed

schemes resort to periodic probing of forwarding paths and collecting feedback signals to avoid instability, which can cause substantial control overhead [11].

At the moment, very little is known about the price we pay for centralized and distributed schemes in terms of efficiency, congestion, complexity, and stability, and whether there exists a practical middle-ground between the two. A principal reason behind this is that current adaptive routing algorithms are just too complex to be modeled in a single mathematical framework that would facilitate a fair comparison. The model has to be deep enough to expose the whole spectrum of the distributed–centralized trade-off, while still being simple enough to be subjected to a thorough mathematical analysis. In this paper, we propose what we believe is the simplest such model.

In our model, which we call *generalized oblivious routing*, we are given a capacitated network with a set of source-destination pairs, but we do not have access to any *a priori* information on traffic demands. The task is to pre-compute a traffic-matrix-to-path-flow mapping (a so called routing function) that minimizes congestion over *any* combination of demands. The model is oblivious in that the routing function is not allowed to change with the demands.

Our key insight is that different types of routing functions give rise to different routing architectures, be that distributed, centralized, or hybrid, and studying the geometric properties of these routing functions allows us to reason about the respective architectures. Distributed routing, in our model, precisely corresponds to conventional *oblivious routing* algorithms [28, 29, 30, 31, 32, 33, 34]. Accordingly, on the distributed end of the architecture spectrum are routing algorithms that are extremely simple but might cause grave congestion [12, 32]. We show, contrariwise, that on the centralized side we find provably zero congestion over *any* routable traffic matrix and optimizability with respect to any reasonable traffic engineering objective. However, we also find that centralized schemes can become overly complex. The truth turns out to be in the middle: we show a hybrid distributed-centralized scheme that unifies the advantages of the two extremes and we prove that this hybrid scheme is asymptotically optimal. We also demonstrate the viability of this scheme in comprehensive numerical evaluations.

1.1. Related work

Traffic management algorithms strive to eliminate network-wide congestion. To this end, a *flow-control algorithm* can vary the rate at which source nodes emit traffic in concert with the congestion feedback received from the network. Examples are various versions of the venerable Transmission Control Protocol (TCP) and other network utility maximization schemes [35]. Recently, there has been a trend towards generalizing these flow-control algorithms to the case when users communicate over more than one path and they actively control not only the source rate but the fraction of traffic routed along the individual

paths, or the paths themselves, as well [20, 21, 36, 37, 22, 38]. This brings us to the second form of traffic management algorithms: *multipath rate-adaptive routing* [18, 22, 24, 25, 38, 39, 11]. Here, the amount of traffic to be routed is given, and the task is to distribute the load on the forwarding paths in a way to minimize congestion. While flow control is chiefly an end-to-end scheme, multipath rate-adaptive routing is much better suited to intra-domain traffic engineering where the ingress nodes do not have control over the rate of user generated traffic.

Right from the beginning, rate-adaptive routing algorithms were conceived to be distributed [18], which means that only information available to a router locally can be used to make routing decisions. It quickly turned out, however, that conflicting decisions made by routers unaware of each other's state can easily lead to wide-scale route oscillations [27, 19].

A minimalistic approach to eliminate instability is to apply no rate-adaptation at all: in *oblivious routing* traffic splitting ratios are set statically to minimize congestion over *any* combination of demands [28, 29, 30, 31, 32, 33, 34]. Curiously, oblivious routing can be surprisingly efficient: in his seminal work, Räcke showed that in undirected graphs we pay only a polylogarithmic factor in congestion compared to the best attainable routing [29]. He later improved the worst-case bound to purely logarithmic [30], which is asymptotically tight [40]. Directed graphs do not admit a logarithmic upper bound [31], but in most cases relevant to practice the congestion penalty remains under 2 [32]. Even though surprisingly small, this bound still allows substantial link over-subscription [12, 32].

It seems, therefore, that some forms of rate-adaptation is inevitable, but special care must be taken to avoid instabilities. Most recent proposals, therefore, introduce some forms of a signaling mechanism to collect state information from the network. TeXCP applies periodic path probing to collect link utilization information [24], REPLEX uses a complete distance-vector protocol infrastructure to distribute the network state [25], while DATE and TRUMP rely on timely feedback from the network [22, 38]. This leads to control overhead, hampers implementation and deployment, and often causes sub-optimality when links are not allowed to be saturated to full capacity to avoid instability [22, 38]. For a promising approach to mitigate these issues, see [11].

Recently, there has been a trend towards a *hybridization* of routing architectures, where decision making is partially migrated to a central controller [41, 42, 43]. A good example is [39], where a central node computes and sets link weights based on which routers can calculate the best traffic splitting ratios independently. At the extreme, one can move the entire routing algorithm to the central node, yielding a fully centralized architecture in which routers merely execute the forwarding commands sent by the controller [44]. This architecture is particularly well suited for software-defined networks [45, 46]. Unfortunately, a

centralized architecture presents its own set of challenges: the central controller presents a single point of failure and scales badly, and it also needs a stable information exchange mechanism implemented by all network nodes. This, in turn, might induce dead-time control instabilities due to the delay between when data is measured and respective control action is taken.

It seems, consequently, that each of the distributed, centralized, and hybrid multipath rate-control architectural models bring about their very own benefits and pose their own challenges. Measuring these against each other so far has only been possible on a purely ad-hoc basis, supported only by piecemeal analysis and anecdotal evidence, but an all-encompassing mathematical framework has been missing. In this regard, our work can be seen as a sequel to [35]: whereas Chiang *et al.* in [35] provide the first comprehensive mathematical framework to understand *control function layering* in network architectures, ours is the first mathematical framework for understanding the *organization of control* in network architectures, be that distributed, centralized, or hybrid.

1.2. Our contributions

In this paper, we introduce a new model for rate-adaptive multipath routing. In our model, different routing architectures correspond to different constraints on the routing function, the mapping that embodies the very rate-adaptation mechanism, and through a comprehensive analysis of these constraints we are able to measure different routing architectures against one another on a common ground.

The particular contributions are as follows:

- We introduce generalized oblivious routing, our model in which a routing function is pre-computed to minimize congestion with respect to any traffic matrix, and we cast conventional oblivious routing in this model.
- We prove that for any network a centralized adaptive routing algorithm exists that can route any combination of user demands the network admits with zero congestion.
- We present the first theoretical upper bound on the complexity of the centralized algorithm.
- We present a hybrid distributed-centralized algorithm that seems to unify the advantages of the two worlds. We prove that our hybrid scheme is asymptotically optimal.
- Finally, we evaluate the algorithms in extensive numerical studies and we find that distributed schemes indeed cause congestion and centralized schemes can become overly complex, while our hybrid scheme performs well in many realistic scenarios with reasonable complexity.

1.3. Structure of the paper

The paper is organized as follows. We introduce basic notation in Section 2. We state the model of generalized oblivious routing formally in Section 3 and we introduce a geometric framework to describe it in Section 4. In Section 5, we cast conventional oblivious routing algorithms within this framework. In Section 6, we present the centralized algorithm that naturally arises in this model, we show how to compute it, and we analyze its complexity. Then, in Section 7 we discuss a hybrid scheme, in Section 8 we evaluate the algorithms numerically, and we conclude our work in Section 9.

Mathematical proofs of the theorems are generally relegated to the Appendix, with the exception of the few ones necessary to get an understanding of the paper.

2. Notation

Let $G(V, E)$ be a strongly connected directed graph, where nodes represent the routers and directed arcs represent the communication links[1] between routers (see Table 1 for a summary of notation). Let $n = |V|$ and $m = |E|$, and let $c = [c_{ij} > 0 : (i, j) \in E]$ be the column m-vector of (finite) link capacities. Users are represented by the set of unique source-destination pairs $(s_k, d_k) : k \in \mathcal{K} = \{1, \ldots, K\}$, between which a set of static paths $\mathcal{P}_k$ is provisioned[2]. The path set $\mathcal{P}_k$ for user $k \in \mathcal{K}$ is represented by the arc-path-incidence matrix P_k: this matrix has m rows and a column for each of the p_k paths for user k, and the i, j entry is 1 if the jth path of k traverses the ith arc and zero otherwise.

Let θ_k denote the momentary demand of the k-th user presented at the source node s_k. The column K-vector $\theta = [\theta_k : k \in \mathcal{K}] \in \mathbb{R}_+^K$ is called a traffic matrix and $\mathbb{R}_+^K$ is called the *throughput space*. Let $\Theta \subset \mathbb{R}^K$ denote the set traffic matrices can take their values from. This set may be formed by an admission control mechanism, it may be shaped by the capacity of the ingress and egress links of the network (as of the so called *hose-model* [47]), or it might be determined by some other means. We assume that Θ is compact (closed and bounded) and contains the origin.

The task of a multipath rate-adaptive routing algorithm is to accommodate any combination of demands θ on the paths $\mathcal{P}$ in a way as to avoid, or at least to minimize, link over-subscription. The corresponding allocation of demands to the paths is described 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 $u_k = [u_P : P \in \mathcal{P}_k]$ is the path-flow vector of the kth user, p is the number of all paths, and the Euclidean space $\mathbb{R}^p$ is called the *flow space*.

[1] In the rest of this paper we use the terms "link" and "arc" interchangeably.

[2] Path-flow formulation is chosen only for convenience. The results apply equally well to the arc-flow formulation.

Table 1: Notations

$G(V, E)$	a directed graph, with the set of nodes V ($\|V\| = n$) and the set of directed arcs E ($\|E\| = m$)
c	the column m-vector of arc capacities
(s_k, d_k)	source-destination pairs for $k \in \mathcal{K} = \{1, \ldots, K\}$
$\mathcal{P}_k$	the set of $s_k \to d_k$ paths assigned to some $k \in \mathcal{K}$
p_k	the number of paths for user k, $p_k = \|\mathcal{P}_k\|$
p	number of all paths, $p = \sum_{k \in \mathcal{K}} p_k$
P_k	an $m \times p_k$ arc-path incidence matrix for user k
P_k^{ij}	the row of P_k corresponding to arc $(i, j) \in E$
u_P	path-flow routed over 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 routing, a column p-vector $u = [u_k : k \in \mathcal{K}]$
θ	a traffic matrix, a column K-vector $\theta = [\theta_k : k \in \mathcal{K}]$
M	flow polytope, the set of path flows on $\mathcal{P}$ 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 $\mathcal{S} : \mathbb{R}^K \mapsto \mathbb{R}^p$; if $\mathcal{S}$ is affine, we write $\mathcal{S} = F \cdot + g$, where F is a linear transformation and g is a vector
$\mathcal{S}_k$	the routing function for user k with $\mathcal{S}_k : \mathbb{R}^K \mapsto \mathbb{R}^{p_k}$
$\underline{1}$	a vector of all 1s of proper size

In fact, one can think of a rate-adaptive routing algorithm as a mapping that indicates how to associate a path-flow vector u with a given traffic matrix θ. We call this mapping a *routing function*, denoted by $\mathcal{S}$ (for a precise definition, see the next section). Then, rate adaptation can be expressed as $u = \mathcal{S}(\theta, \ldots)$, representing that the path-flows depend on the traffic matrix principally plus, potentially, further parameters like the delay, free capacity, congestion price, etc., collected from the forwarding paths. It will often prove convenient to write $\mathcal{S}$ in a decomposed form, where for each $k \in \mathcal{K}$ a separate function $\mathcal{S}_k$ is used to generate the path-flows u_k.

The congestion produced by a routing function $\mathcal{S}$ is measured by the *maximum link utilization* $\kappa_{\mathcal{S}}(\theta)$, defined as the maximum of the utilizations taken over all

arcs of G when some θ is routed by $\mathcal{S}$:

$$\kappa_{\mathcal{S}}(\theta) = \max_{(i,j)\in E} \frac{1}{c_{ij}} \sum_{k\in\mathcal{K}} P_k^{ij} \mathcal{S}_k(\theta_k, \ldots) \ .$$

3. Generalized oblivious routing

In this paper, our aim is to find a model for rate-adaptive routing that is just sufficiently and necessarily complex to describe real routing algorithms, while also lacks all unnecessary degrees of freedom that would inhibit an insightful analysis. Our model, called *generalized oblivious routing*, is defined by the following principles:

- PRINCIPLE-1: The routing function is oblivious to the demands.
- PRINCIPLE-2: The amount of flow placed to a path by the routing function depends on the traffic matrix exclusively.
- PRINCIPLE-3: The routing function is such that it minimizes the maximum link utilization over any traffic matrix that can appear in the network.

The first principle in essence dictates that the routing function is pre-computed and fixed for the lifetime of the network[3]. The second principle expresses that the only input the routing function takes is the actual traffic matrix. In other words, for $\mathcal{S}$ to be oblivious it must be $\Theta \mapsto \mathbb{R}^p$. Finally, the third principle states that a generalized oblivious routing function is such that it minimizes the worst-case congestion.

In order to formalize the above principles, we need two additional definitions, namely, the *throughput-invariance* rule and the related concept of the *throughput mapping* $\mathcal{T}$. The throughput-invariance rule expresses the natural requirement for a routing function to be an actual flow mapping, that is, for any traffic matrix the flows placed to the individual paths of a user must add up to the actual demand of that user, as of the input traffic matrix. This requirement is formally described with the notion of the throughput mapping, a linear transformation that sums up the path-flows of each user.

Definition 1. *The* throughput mapping $\mathcal{T}$ *is a* $\mathbb{R}^p \mapsto \Theta$ *function* $\mathcal{T}(u) = Qu$*, where* Q *is a* $K \times p$ *matrix, the elements in* k*th 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.*

[3]Naturally, the routing function might indeed change whenever the topology changes, for instance, due to a network upgrade or a device/link failure. These cases are, however, beyond the scope of this paper.

Then, a function $\mathcal{S}$ fulfills throughput-invariance if it is $\Theta \mapsto \mathbb{R}^p$ and the inverse $\mathcal{S}^{-1}$ exists and $\mathcal{S}^{-1} \equiv \mathcal{T}$. Denote the set of such functions by $\bar{\mathcal{S}}$.

Using this formalism, we define a function $\mathcal{S}^*$ as an *oblivious routing function*, satisfying the three principles of generalized oblivious routing, if it solves the following optimization problem:

$$\mathcal{S}^* = \operatorname*{argmin}_{\mathcal{S} \in \bar{\mathcal{S}}} \max_{\theta \in \Theta} \kappa_{\mathcal{S}}(\theta) \ . \tag{1}$$

The performance of $\mathcal{S}$ is measured by the maximum link utilization it produces over any traffic matrix in Θ:

$$\gamma_{\mathcal{S}}(\Theta) = \max_{\theta \in \Theta} \kappa_{\mathcal{S}}(\theta) \ .$$

Easily, the smaller $\gamma_{\mathcal{S}}(\Theta)$ the better the routing. When $\gamma_{\mathcal{S}}(\Theta) \leq 1$ then the routing function orders feasible routing to every traffic matrix in Θ, while $\gamma_{\mathcal{S}}(\Theta) > 1$ marks congestion. We define $\gamma(\Theta) = \gamma_{\mathcal{S}^*}(\Theta)$ as the *absolute performance index*.

4. A geometric framework

In this paper, we use a geometric framework to study routing algorithms. This framework is built on the observation that one can describe routing algorithms with certain geometric objects and can reason about them in purely geometric terms. In the context of this paper, these geometric objects will almost exclusively be provided by polyhedra. First, we introduce some basic definitions from convex analysis [48, 49] and then we present the main geometric concepts of the framework.

A *polyhedron* P is an intersection of finitely many half-spaces $P = \{x : Ax \leq b\} \subseteq \mathbb{R}^d$, where A is some $q \times d$ matrix and b is a column q-vector. A bounded polyhedron is called a *polytope*. The *scalar multiple* of a polytope $P = \{x : Ax \leq b\}$ is defined as $\lambda P = \{x : Ax \leq \lambda b\}$. The *boundary* ∂P of P consists of the set of points $x \in P$ for which one or more inequalities in $Ax \leq b$ hold with equality. Given some points $\{x_1, x_2, \ldots, x_s\}$ in $\mathbb{R}^d$, the convex combination $\text{Conv}\{x_1, \ldots, x_s\}$ is defined as

$$\text{Conv}\{x_1, \ldots, x_s\} = \left\{ x : \exists \lambda_1, \ldots, \lambda_d, \lambda_i \geq 0, \text{ where } x = \sum_{i=1}^{s} \lambda_i x_i, \sum_{i=1}^{s} \lambda_i = 1 \right\} \ .$$

Given a polytope P, some $x \in P$ is an *extreme point* of P if it cannot be generated as the convex combination of two distinct points in P. Any polytope $P = \{x : Ax \leq b\}$ is equivalently described by the convex-combination of its extreme points $x_1, \ldots, x_s$: $P = \text{Conv}\{x_1, \ldots, x_s\}$.

A *simplex* is a d-dimensional polytope arising as the convex-combination of exactly $d+1$ extreme points. A *polyhedral partition* of P is a set of disjunct (apart from the boundaries) polytopes $Q_i : i \in \{1, \ldots, q\}$ so that $P = \bigcup_i Q_i$. A *triangulation* is a polyhedral partition $Q_i : i \in \{1, \ldots, q\}$ so that each Q_i is a simplex. A *boundary-triangulation* is a triangulation that does not introduce interior points, i.e., each Q_i is a convex combination of some subset of the extreme points of P.

4.1. The geometry of rate-adaptive routing

Next, we introduce the most important objects that underly our geometric model of rate-adaptive routing.

Definition 2. *The* flow polytope M *is the subset of the flow space that contains all admissible routings, subject to link capacities and non-negativity constraints:*

$$M = \{u : \sum_{k \in K} P_k u_k \leq c,\ u \geq 0\} \subset \mathbb{R}^p. \tag{2}$$

Definition 3. *The* throughput polytope T *is the set of traffic matrices θ for which there is a routing u that accommodates θ in the network with no link over-utilization:*

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

A sample network with 2 users and 3 paths and the corresponding polytopes are depicted in Fig. 1. The half-space representation of T is as follows:

$$T = \{\theta \geq 0 : \theta_1 + \theta_2 \leq 2 \text{ and } \theta_2 \leq 1\ \} \ .$$

The polytope T has the following properties [50, 51, 42]:

- $T = \mathcal{T}(M)$;
- as an affine mapping of a polytope, T is itself a polytope;
- T is *convex*: $\forall \theta_1, \theta_2 \in T : \text{Conv}\{\theta_1, \theta_2\} \subseteq T$;
- if G is strongly connected, $p_k > 0$ for each $k \in \mathcal{K}$, and $c > 0$, then T is *full-dimensional*;
- T is *down-monotone*: for each $\theta \in T$ it holds that $\forall 0 \leq \tau \leq \theta : \tau \in T$;
- in general, no polynomial size description for T exists, as there are networks for which both the half-space and the vertex representation of T grows as $\Omega(2^K)$.

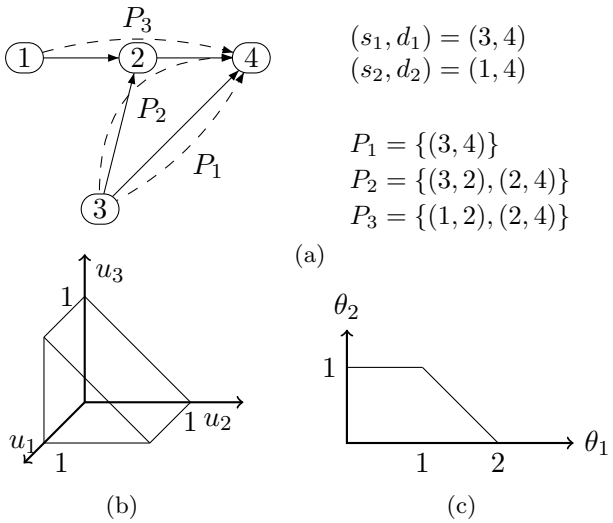


Figure 1: A sample directed network with source-destination pairs and the set of paths for each user (a), and the corresponding flow polytope (b) and throughput polytope (c).

In our geometric framework, the throughput polytope essentially stands for representing the network (observe that T depends only on the network topology, link capacities, the source-destination pairs, and the path set). In this regard, T is more general than the hose model [47], which contains only the constraints arising on the ingress and egress links of the network. In contrast, T contains *all* the constraints arising on any of the bottlenecks within the network.

For our framework to be complete, we still need a geometric object to describe routing functions. This object will be called the feasible region.

Definition 4. *For some routing function $\mathcal{S}$, the* feasible region $\mathcal{R}(S)$ *is the set of traffic matrices that can be routed in the network by S without causing link over-subscription:*

$$\mathcal{R}(\mathcal{S}) = \{\theta \in \Theta : \mathcal{S}(\theta) \in M\} \ .$$

A routing function $\mathcal{S}$ is *optimal* if $\mathcal{R}(\mathcal{S}) = T$. In this case, $\mathcal{S}$ can route any traffic matrix the network admits.

As it turns out, the throughput polytope and the feasible region are enough to develop a deep geometric theory for rate-adaptive routing. The optimization objective of generalized oblivious routing (1) can be written in geometric terms as

$$\mathcal{S}^* = \operatorname*{argmin}_{\mathcal{S}} \{\lambda : \mathcal{S}(\Theta) \subseteq \lambda M\} \ . \tag{3}$$

Correspondingly, $\gamma_{\mathcal{S}}(\Theta) = \min\{\lambda : \mathcal{S}(\Theta) \subseteq \lambda M\}$. The optimization problem (3) asks for the routing function $\mathcal{S}$ whose image $\mathcal{S}(\Theta)$ best fits into the set of feasible

routings M, in the sense that we need to scalar multiple M with the smallest λ so that $\mathcal{S}(\Theta)$ becomes the subset of λM. Here, we invoke the property that scaling the link capacities c with some scalar λ equals, in geometric terms, scalar multiplying M with the same λ to λM (c.f., (2)). In fact, if we scaled the link capacities by exactly $\gamma_{\mathcal{S}}(\Theta)$, then all traffic matrices in Θ would be routed by $\mathcal{S}$ without congestion.

4.2. Piecewise affine routing functions

Piecewise affine routing functions constitute the foundation on which we build our analysis, as they are simple enough to facilitate solving (3) yet broad enough to express most routing methods relevant to practice, like single-path routing, equal-cost multipath, traffic splitting ratios, etc.

A *piecewise affine* (or simply, affine) routing function $\mathcal{S} = \{(\mathcal{S}^i(\theta), \mathcal{D}^i) : i \in \mathcal{I}\}$ is defined as a collection of simple affine functions $\mathcal{S}^i(\theta)$ over a polyhedral partition $\{\mathcal{D}^i\}$ of Θ [52]:

$$\mathcal{S}^i(\theta) = F^i\theta + g^i \text{ whenever } \theta \in \mathcal{D}^i \qquad i \in \mathcal{I} \ ,$$

where F^i are $p \times K$ matrices and g^i are column p-vectors. Equivalently, when decomposed into separate routing functions $\mathcal{S}_k(\theta)$ for the source-destination pairs $k \in \mathcal{K}$ we get:

$$\mathcal{S}^i_k(\theta) = F^i_k\theta + g^i_k \text{ whenever } \theta \in \mathcal{D}^i \qquad i \in \mathcal{I} \ ,$$

where F^i_k are $p_k \times K$ matrices and g^i_k are column p_k-vectors. For affine routing functions the throughput-invariance rule takes the simple form:

$$\forall i \in \mathcal{I} : \underline{1}^T F^i_{kl} = \delta_{kl} = \begin{cases} 1 & \text{if } k = l \\ 0 & \text{otherwise} \end{cases} , \qquad \underline{1}^T g^i_k = 0 \ .$$

A piecewise affine routing function is called *singular* if it contains only a single region. i.e., the entire set Θ is routed by the same affine function:

$$\mathcal{S}(\theta) = F\theta + g \text{ whenever } \theta \in \Theta \ .$$

This is equal to saying that $|\mathcal{I}| = 1$. Otherwise, $\mathcal{S}$ is *compound*.

We say that $\mathcal{S}(\theta)$ is *block-diagonal*, if for the decomposed routing functions $\mathcal{S}^i_k(\theta)$ it holds that each column of the matrix F^i_k except for the kth is zero:

$$\forall i \in \mathcal{I}, \forall k \in \mathcal{K}, \forall l \in \mathcal{K} \setminus \{k\} : F^i_{kl} \equiv 0 \ ,$$

where F^i_{kl} denotes the lth column of F^i_k.

A piecewise affine routing function is called piecewise linear, or simply *linear*, if $\forall i \in \mathcal{I} : g^i \equiv 0$. Otherwise, we call $\mathcal{S}$ (general) *affine*.

For a singular affine routing function $\mathcal{S}(\theta) = F\theta + g$, the feasible region $\mathcal{R}(\mathcal{S})$ is a polytope:

$$\mathcal{R}(\mathcal{S}) = \{\theta : \sum_{k \in \mathcal{K}} P_k(F_k\theta + g_k) \leq c \\ F_k\theta + g_k \geq 0 \quad \forall k \in \mathcal{K}\} \subset \mathbb{R}^K \ . \tag{4}$$

For compound routing functions, $\mathcal{R}(\mathcal{S})$ can be obtained by taking the union of the feasible regions of each $\mathcal{S}^i(\theta) : i \in \mathcal{I}$. This, however, is not necessarily convex, let alone polyhedral.

5. Distributed architectures: oblivious routing

In the rest of this paper, we analyze the routing architectures that arise when taking the generalized oblivious routing model over different piecewise affine routing functions. We begin by showing that simplest of affine routing functions, singular and block-diagonal functions, naturally give rise to conventional oblivious routing algorithms [28, 29, 30, 31, 32, 33, 34]. Then, in the next section we study gradually more complex routing functions and we shall see how these yield centralized and hybrid generalized oblivious routing schemes.

5.1. Singular block-diagonal routing functions

Oblivious routing asks for a setting of traffic splitting ratios at source nodes so that the resultant path-flows induce the smallest possible congestion over any selection of traffic matrices [34]. The fact that a routing algorithm can be realized in a distributed setting, within our model, precisely corresponds to the requirement that the routing function $\mathcal{S}_k$ at some source node $s_k : k \in \mathcal{K}$ depends only on information that is available locally at s_k, and this is exactly the actual demand θ_k of user k. See the simplified architectural model for our running example in Fig. 2.

This leads us to the observation that if a piecewise affine routing function $\mathcal{S}$ is *singular and block-diagonal*, then it can be realized in a distributed setting. Singularity is important as source nodes s_k do not have access to the entire θ vector and so they cannot decide on which region $\mathcal{D}^i$ to choose, and block-diagonality formalizes the requirement for distributed architectures that $\forall k \in \mathcal{K}$: $\frac{\partial \mathcal{S}_k}{\partial \theta_l} = 0$ if $k \neq l$.

The oblivious routing function $\mathcal{S}^*$ for our sample network in Fig. 1 can be computed as follows. First, observe that the second source-destination pair has only a single path, thus all its traffic is sent through this path. On the other hand, the first user has two paths. Let β denote the fraction of traffic sent through node 2. We consider two critical classes of traffic matrices, which produce the largest link load. One is, obviously, the traffic matrix $[1, 1]$, for which the maximum load

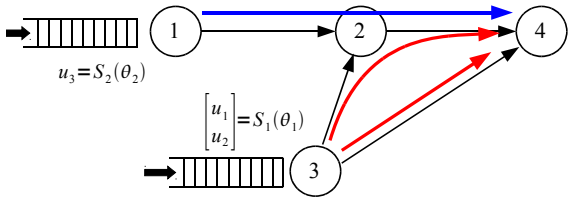


Figure 2: Distributed oblivious routing architecture.

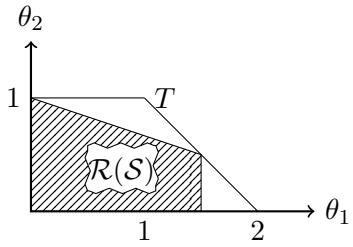


Figure 3: The feasible region for the routing function (5).

$1 + \beta$ occurs on link $(2, 4)$. Second, the demand $[0, 2]$ causes $2(1 - \beta)$ maximum load on link $(3, 4)$. To find the oblivious routing function, we seek β so that the maximum load is minimal. This occurs when $1 + \beta = 2(1 - \beta)$, which yields $\beta = \frac{2}{3}$.

Based on this observation, the oblivious routing function $\mathcal{S}^*$ for our sample network in Fig. 1 is as follows:

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \mathcal{S}^*(\theta) = F\theta + g = \begin{pmatrix} \frac{2}{3} & 0 \\ \frac{1}{3} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} . \tag{5}$$

Observe how the blocks in the diagonal of the matrix F correspond to traffic splitting ratios for the users. For instance, s_1 needs to place two thirds of the demand of user 1 to path P_1 and the rest to P_2, while for the second user all traffic is routed along the single path P_3. Using (4), for the feasible region we get $\mathcal{R}(\mathcal{S}^*) = \{\theta \geq 0 : \theta_1 + 3\theta_2 \leq 3 \text{ and } \theta_1 \leq \frac{3}{2}\}$ (see Fig. 3). We observe that, in this example, $\mathcal{S}^* = F\theta$ with transposition g equal to zero and $F \geq 0$. This is not a coincidence, as the next results suggest.

Lemma 1. *Suppose $0 \in \Theta$. Now, if a piecewise affine routing function $\mathcal{S}$ is singular on Θ then it is also linear.*

The next result allows us to seek singular block-diagonal routing functions $\mathcal{S}(\theta) = F\theta + g$ in the simpler form $\mathcal{S} : \mathcal{S}_k = f_k\theta_k + g_k$, where f_k and g_k are column p_k vectors and $f_k \geq 0$ for all $k \in \mathcal{K}$.

Lemma 2. *Suppose that $\mathcal{S} : \mathcal{S}_k = f_k\theta_k + g_k$ is singular and block-diagonal. Then, there exists $\mathcal{S}' : \mathcal{S}'_k = f'_k\theta_k + g'_k$ with $f'_k \geq 0$ for each $k \in \mathcal{K}$, so that $\mathcal{R}(\mathcal{S}) \subseteq \mathcal{R}(\mathcal{S}')$.*

5.2. Computing the oblivious routing function

To obtain the oblivious routing function, one needs to solve the optimization problem (3) over singular, block-diagonal routing functions. Below, we shall aim for a bit more: we solve (3) for *general* block-diagonal affine functions (not necessarily linear) over an *arbitrary polyhedral set* $\Theta = \{\theta : H\theta \leq h\}$, where H is a $q \times K$ matrix and h is a column q-vector. This generalization will be important later, when we discuss hybrid oblivious routing algorithms.

Our approach builds on the linear programming method of [32] (but see also [42]). First, we write (3) in a verbose form:

$$\min \lambda : \quad f_k \geq 0, \underline{1}^T f_k = 1, \underline{1}^T g_k = 0 \quad \forall k \in \mathcal{K} \tag{6}$$

$$\forall (i,j) \in E : \quad \frac{\sum_{k\in K} P_k^{ij}(f_k\theta_k + g_k)}{c_{ij}} \leq \lambda \qquad \forall \theta \in \Theta \tag{7}$$

Constraint (6) enforces throughput-invariance and (7) requires utilization at each link to be at most λ. Observe that such a constraint is present for every $\theta \in \Theta$, which yields an infinitely large linear program. To overcome this problem, first we organize (7) into a slave problem for each $(i,j) \in E$:

$$\lambda \geq \max \frac{\sum_{k\in K} P_k^{ij}(f_k\theta_k + g_k)}{c_{ij}} \tag{8}$$

$$\textstyle\sum_{k\in\mathcal{K}} P_k u_k \leq c, \; H\theta \leq h \tag{9}$$

$$\underline{1}^T u_k = \theta_k, u_k \geq 0 \qquad \forall k \in \mathcal{K} \tag{10}$$

Dualizing (8)–(10) and collecting all dual slave problems yields a single giant linear program:

$$\min \lambda : \quad f_k \geq 0, \underline{1}^T f_k = 1, \underline{1}^T g_k = 0 \quad \forall k \in \mathcal{K} \tag{11}$$

$$\forall (i,j) \in E : \quad \{w^{ij}c + \lambda^{ij}h + \frac{\sum_{k\in K} P_k^{ij} g_k}{c_{ij}} \quad \leq \lambda \tag{12}$$

$$w^{ij}P_k \geq \underline{1}^T \beta_k^{ij} \qquad \forall k \in \mathcal{K} \tag{13}$$

$$\lambda^{ij}H_k - \beta_k^{ij} \geq \frac{P_k^{ij} f_k}{c_{ij}} \qquad \forall k \in \mathcal{K} \tag{14}$$

$$w^{ij}, \lambda^{ij} \geq 0 \qquad \} \tag{15}$$

where H_k is the kth column of H, w^{ij} and λ^{ij} are the duals to the constraints (9) and β_k^{ij} to constraints (10), specific to the dual subproblem corresponding to each $(i,j) \in E$. This is now a linear program of polynomial size, which can be solved in polynomial time with standard linear program solvers. The oblivious routing function can be obtained from the optimal solution in the form $\mathcal{S}_k^*(\theta) = f_k\theta_k + g_k$,

and the optimal objective function corresponds to $\gamma(\Theta)$. Note that the additional constraints $\forall k \in \mathcal{K} : g_k = 0$ can be added to (11)–(15) to compute linear routing functions instead of general affine ones.

It is important to call the attention to an important difference between conventional oblivious routing and generalized oblivious routing. Generalized oblivious routing aims at minimizing the performance index $\gamma_{\mathcal{S}}(\Theta) = \max_{\theta\in\Theta} \kappa_{\mathcal{S}}(\theta)$, which captures the congestion in absolute terms. Conventional oblivious routing, on the other hand, is defined in terms of a relative measure:

$$\alpha_{\mathcal{S}}(\Theta) = \max_{\theta\in\Theta} \frac{\kappa_{\mathcal{S}}(\theta)}{\kappa_{\mathcal{S}_{\mathrm{OPT}}}(\theta)} \tag{16}$$

that compares the congestion produced by $\mathcal{S}$ to that of a hypothetical optimal routing function $\mathcal{S}_{\mathrm{OPT}}$, not necessarily affine, that attains the smallest possible congestion for all $\theta \in \Theta$ (see the next section for such a routing function). The relative performance index $\alpha(\Theta) = \min_{\mathcal{S}} \alpha_{\mathcal{S}}(\Theta)$ is usually called the *competitive ratio* or oblivious ratio.

The next result guarantees that these two performance metrics are equivalent[4] in the simple case when $\Theta = T$.

Lemma 3. *For any singular linear affine routing function $\mathcal{S}$: $\gamma_{\mathcal{S}}(T) = \alpha_{\mathcal{S}}(T)$.*

In our running example of Fig. 1, we get $\gamma(T) = \alpha(T) = \frac{4}{3}$, putting the worst-case congestion of oblivious routing to $\frac{4}{3}$.

6. Centralized architectures: optimal oblivious routing

Distributed oblivious routing is simple but inefficient. The reason is that singular and block-diagonal routing functions are just too restrictive. In this section, we show that compound affine routing functions, on the other hand, allow for provably optimal routing in an inherently centralized setting.

6.1. Compound affine routing functions

We start by giving an example on the sample network of Fig. 1. A possible routing would 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 P_3. This corresponds to the linear routing function $\mathcal{S}^1$:

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & 0 \\ \frac{1}{2} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} \tag{17}$$

[4]Similar argumentation appears in [32].

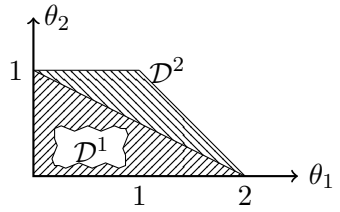


Figure 4: Control regions for the routing function (17)–(20).

The feasible region is as follows:

$$\mathcal{R}(\mathcal{S}^1) = \{\theta : \theta_1 + 2\theta_2 \leq 2,\ \theta \geq 0\}\ . \tag{18}$$

Next, we try to find a routing function $\mathcal{S}^2$ on $T \setminus \mathcal{R}(\mathcal{S}^1)$:

$$\begin{aligned} T \setminus \mathcal{R}(\mathcal{S}^1) = \{\theta \geq 0 : \theta_1 + 2\theta_2 \geq 2 \\ \theta_1 + \theta_2 \leq 2,\ \theta_2 \leq 1\}\ . \end{aligned} \tag{19}$$

An adequate choice for $\mathcal{S}^2$ is

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & -1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} + \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}\ , \tag{20}$$

as the feasible region $\mathcal{R}(\mathcal{S}_2) = \{\theta : 1 \leq \theta_1 + \theta_2 \leq 2, 0 \leq \theta_2 \leq 1\}$ contains $T \setminus \mathcal{R}(\mathcal{S}^1)$. What remained to be done is to determine regions $\mathcal{D}^1$ and $\mathcal{D}^2$. Easily, for the routing function to be feasible on the entire set T, we need to ensure that $\mathcal{D}^1 \subseteq \mathcal{R}(\mathcal{S}^1)$ and $\mathcal{D}^2 \subseteq \mathcal{R}(\mathcal{S}^2)$. Care must also be taken to eliminate overlapping regions, in order for the compound routing function to be well-defined. In our example, we may choose either $\mathcal{S}^1$ or $\mathcal{S}^2$ to cover $\mathcal{R}(\mathcal{S}^1) \cap \mathcal{R}(\mathcal{S}^2)$, but not both. It is convenient to set $\mathcal{D}^1 = \mathcal{R}(\mathcal{S}^1)$ and let $\mathcal{D}^2$ be $T \setminus \mathcal{D}^1$ (see Fig. 4). This way the compound routing function will be continuous over T. The resultant piecewise affine routing function $\mathcal{S}(\theta) = \{(\mathcal{S}^i(\theta), \mathcal{D}^i) : i \in \{1, 2\}\}$ defines a rate-adaptive multipath routing algorithm, which, as the reader easily checks, routes any admissible traffic matrix without congestion.

6.2. Computing the oblivious routing function

The next result guarantees that an optimal affine routing function exists for *any* network.

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

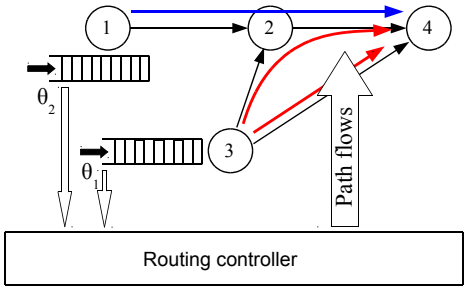


Figure 5: Centralized oblivious routing architecture.

PROOF. Let $f(u,\theta)$ be a linear or convex quadratic objective function and consider the *multiparametric* program

$$z(\theta) = \min f(u,\theta): \quad \sum_{k\in K} P_k u_k \leq c \tag{21}$$

$$\underline{1}^T u_k = \theta_k \qquad \forall k \in \mathcal{K} \tag{22}$$

$$u \geq 0 \tag{23}$$

This is basically a conventional multicommodity flow problem with the specialty that the right-hand-side depends on the input parameter θ. Then, existence of a continuous, piecewise affine function $\mathcal{S}$ that optimizes (21)–(23) is guaranteed by [52]. □

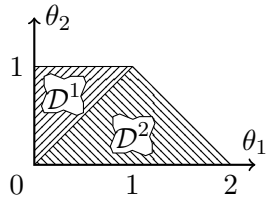
For a more in-depth exposition of this result, consult [44].

The significance of this theorem is that, theoretically, no information on expected traffic is necessary to design a rate-adaptive multipath routing algorithm that guarantees feasibility over $\theta \in T$. One solves a multiparametric linear or quadratic program, which, although computationally quite involving, is viable thanks to recent advances in geometric multiparametric programming [52, 53]. This phase can be done offline and yields an optimal compound affine routing function.

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 $\theta \in \mathcal{D}^i$, evaluates $u = F^i\theta + g^i$ and downloads the resultant traffic splitting ratios to the routers (see Fig. 5 for a schematic model of this architecture). 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 convex quadratic objective functions are permitted. Plausible objectives would be to minimize delay or the maximum link utilization. Furthermore, continuity guarantees smooth transients both within and between regions.

$$\mathcal{S}^1 : \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}$$
$$\mathcal{S}^2 : \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}$$

(a)



(b)

Figure 6: An optimal compound affine routing function (a) and the corresponding triangulation of T (b).

Unfortunately, the centralized routing architecture can become very complex, as there is no polynomial upper bound on the number of regions and individual simple routing functions that emerge when solving the multiparametric linear program [52, 53]. When $\mathcal{I}$ exceeds about 10^5, centralized 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.

Below we provide an alternative way to calculate the routing function, which establishes a firm upper bound on the complexity while still maintaining optimality.

Theorem 2. *For any capacitated network and any boundary-triangulation $Q_i : i \in \{1, \ldots, q\}$ of T, there exists a continuous compound affine routing function $\mathcal{S} = \{(\mathcal{D}^i, \mathcal{S}^i) : i \in \mathcal{I}\}$ so that $|\mathcal{I}| = q$ and $\mathcal{R}(\mathcal{S}) = T$.*

Corollary 1. *The number of the minimal triangulation represents an upper-bound on the complexity of centralized algorithms.*

Note that finding a triangulation for which q is minimal is a very difficult problem [54], and even if we manage to find one q can still be exponential. For the running example, we obtain the routing function as depicted in Fig. 6.

7. Hybrid distributed-centralized architectures

Next, we introduce a hybrid distributed-centralized scheme, in order to combine the useful properties of centralized architectures with the simplicity of distributed ones.

7.1. Compound block-diagonal routing functions

There are two reasons due to which implementing a compound affine routing function $\mathcal{S} = \{(\mathcal{D}^i, \mathcal{S}^i) : i \in \mathcal{I}\}$ needs central control. First, picking the region

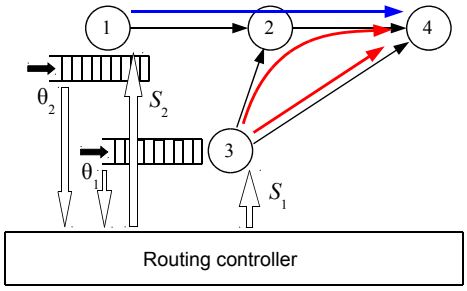
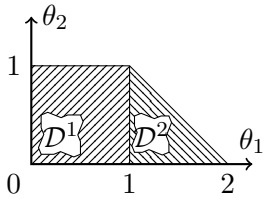


Figure 7: Hybrid oblivious routing architecture.

$$\mathcal{S}^1 : \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}$$

$$\mathcal{S}^2 : \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} + \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}$$

(a)



(b)

Figure 8: A compound block-diagonal routing function (a) and the corresponding control regions (b).

$\mathcal{D}^i : i \in \mathcal{I}$ that contains the actual traffic matrix θ necessitates the knowledge of each coordinate of θ (c.f., (19)). The control does not need to be particularly fast in this regard, as usually only a relatively large change in the traffic matrix can trigger a control region transition. Second, configuring the correct traffic splitting ratios as prescribed by $\mathcal{S}^i(\theta)$ also needs full knowledge of θ (c.f., (20)). This process must be very fast to avoid dead-time control instabilities. Nevertheless, if $\mathcal{S}^i(\theta)$ takes the form of a *compound and block-diagonal* routing function, then the second issue is eliminated as now the individual routing functions $\mathcal{S}^i(\theta)$ can be realized in a distributed fashion and the task of the central node reduces to the coarse-grained control of the routing function rather than the fine-grained control of the traffic splitting ratios. This yields a *hybrid distributed-centralized* routing architecture (see Fig. 7).

For a hybrid routing function on our running example, see Fig. 8. Observe that $\mathcal{S}^1$ and $\mathcal{S}^2$ are block-diagonal and, once downloaded to the routers, do not need central orchestration. Note that this routing function happens to be optimal and continuous, but this cannot be guaranteed in general.

7.2. Computing the oblivious routing function

Unfortunately, neither the multiparametric programming techniques nor the triangulation method can be used to compute compound block-diagonal routing

Algorithm 1 Hybrid oblivious routing algorithm

```
HYBRID_OBLIVIOUS_ROUTING(Θ)
function HYBRID_OBLIVIOUS_ROUTING(X)
    (λ, S) ← min_S λ : S(X) ⊆ λM
    if λ ≤ Γ then
        store (X, S) and return
    end if
    (X_1, X_2) ←GENERATE_CUT(X)
    HYBRID_OBLIVIOUS_ROUTING(X_1)
    HYBRID_OBLIVIOUS_ROUTING(X_2)
end function
```

functions. Thus, based on [41, 44, 43] we developed a new cutting plane algorithm. First, we sketch the algorithmic framework and then we discuss how to generate the cutting planes.

The idea is to compute gradually more efficient and more complex routing functions by iteratively dividing the input set Θ into smaller and smaller regions. In the first iteration, we compute a singular block-diagonal routing function $\mathcal{S}_0$ for the entire set Θ. If $\gamma_{\mathcal{S}_0}(\Theta)$ is below a configured limit, say, Γ, we stop. Otherwise, Θ is too large to be covered with a single routing function, so we generate a cut that divides it into two and recurse into the subregions. In the general step, we divide the actual region X into subregions X_1 and X_2 so that $X = X_1 \cup X_2$ and we calculate separate block-diagonal routing functions $\mathcal{S}_1$ and $\mathcal{S}_2$ for X_1 and X_2. This can be done by solving (11)–(15) over X_1 and X_2. The algorithm terminates if the congestion $\max \gamma_{\mathcal{S}_i}(X_i)$ falls below Γ, or when the number of iterations exceeds a certain threshold.

The pseudo-code for the hybrid oblivious routing algorithm is given in Algorithm 1. The result is a compound block-diagonal routing function $\mathcal{S} = \{(\mathcal{D}^i, \mathcal{S}^i) : i \in \mathcal{I}\}$, in which all the individual routing functions take the form $u_k = f_k^i \theta_k + g_k^i$. This 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 θ and checks whether θ resides in the current region $\mathcal{D}^i$. 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 (see again Fig. 7).

7.3. Generating cutting planes

What remained to be done is to define the function GENERATE_CUT in Algorithm 1. 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 $\theta_k \leq t$ for some $k \in \mathcal{K}$ (c.f., Fig. 8). This way, the regions

become K-dimensional hyper-rectangles, which simplifies the central controller significantly.

In operation, the controller first checks whether the cut generated in the first iteration of Algorithm 1, 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 [55].

Below, we show a method to search for a given set X the orthogonal cut $\theta_k \leq t$, which for any selection of $k \in \mathcal{K}$ and any value of t reduces the performance index the most in the resultant subregions. Choose some $k \in \mathcal{K}$, let $\tau_k = \min_{\theta \in X} \theta_k$ and $T_k = \max_{\theta \in X} \theta_k$, define the set $X_1(t) = X \cap \{\theta : \theta_k \leq t\}$ on $t \in [\tau_k, T_k]$, and let $\gamma_1(t) = \gamma(X_1(t))$. For the "other side" of the cut, let $X_2(t) = X \cap \{\theta : \theta_k \geq t\} : t \in [\tau_k, T_k]$ and let $\gamma_2(t) = \gamma(X_2(t))$.

Theorem 3. *The function $\gamma_1(t)$ is monotonically* increasing *and continuous, and $\gamma_2(t)$ is monotonically* decreasing *and continuous on $[\tau_k, T_k]$.*

Our task is now to find a value t for which maximum congestion on the two subregions is minimal. We observe that *(i)* as $\gamma_1(t)$ and $\gamma_2(t)$ are continuous on $[\tau_k, T_k]$, so is their maximum; *(ii)* $X_1(T_k) = X_2(\tau_k) = X$, so $\gamma_1(T_k) = \gamma_2(\tau_k) = \gamma(X)$; *(iii)* $\max(\gamma_1(t), \gamma_2(t))$ has a unique minimum as one of them is increasing and the other is decreasing; and *(iv)* this occurs when $\gamma_1(t) = \gamma_2(t)$. This t, however, is then easy to find by binary search. Note that neither $\gamma_1(t)$ nor $\gamma_2(t)$ changes *strictly*, thus the t at which the minimum occurs is not necessarily unique, but the resultant $\gamma_1(t) = \gamma_2(t)$ is.

Pseudo-code for the binary search algorithm is presented in Algorithm 2. In every iteration, we compute $\gamma_1(t)$ and $\gamma_2(t)$ for the present value of $t \in [\tau_k, T_k]$ (this amounts to solving the linear program (11)–(15) twice). If $|\gamma_1(t) - \gamma_2(t)| < \epsilon$, we stop. Otherwise, we continue the search until we find a t for which $\gamma_1(t)$ is (approximately) equal to $\gamma_2(t)$. We repeat this for all $k \in \mathcal{K}$, and we choose the dimension for which congestion reduces the most. Finally, if multiple dimensions produce the same worst-case congestion, we divide the current region along its largest diameter. The role of this tie-breaking rule will be revealed immediately.

Let $\mathcal{S}^*$ be the hybrid oblivious routing function calculated by the above algorithm. We show that $\mathcal{S}^*$ is asymptotically optimal, in the sense that if the control regions of $\mathcal{S}^*$ are sufficiently small then maximum link utilization converges to 1. Thanks to the tie-breaking rule in Algorithm 2, this is guaranteed in our case.

Theorem 4. *Let $\mathcal{S}^* = \{(\mathcal{D}^i, \mathcal{S}^i) : i \in \mathcal{I}\}$ be the compound block-diagonal routing function on T obtained by executing Algorithm 1, so that $\forall i \in \mathcal{I} : \mathcal{D}^i \subseteq H_{a_i}^{b_i}$ with*

Algorithm 2 Generate an orthogonal cut on region X

function GENERATE_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($t \in [\tau_l, T_l] : \gamma_1(t) = \gamma_2(t)$)
  end for
  $\rho \leftarrow \min_{l \in \mathcal{K}} \gamma_1(t_l)$
  $k \leftarrow \underset{l \in \mathcal{K}: \gamma_1(t_l) = \rho}{\operatorname{argmin}} (T_l - \tau_l)$
  $X_1 \leftarrow X \cap \{\theta : \theta_k \leq t_k\}$, $X_2 \leftarrow X \cap \{\theta : \theta_k \geq t_k\}$
  return (X_1, X_2)
end function

$b_k^i - a_k^i \leq \epsilon$ *for some* $\epsilon > 0$ *for all* $k \in \mathcal{K}$. *Then,* $\gamma_{\mathcal{S}^*}(T) \leq 1 + \epsilon \max_{(i,j) \in E} \frac{\xi_{ij}}{c_{ij}}$, *where* ξ_{ij} *denotes the number of paths sharing link* $(i, j) \in E$.

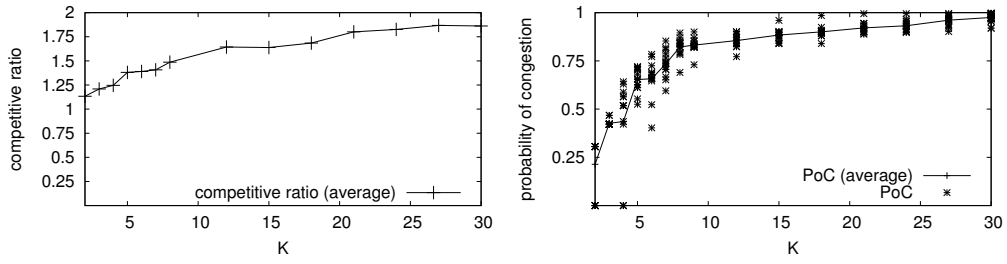
8. Numerical evaluations

So far, we have seen that generalized oblivious routing encompasses distributed, centralized, and hybrid routing architectures, which naturally arise as the applications of the model to different piecewise affine routing functions. In the numerical evaluations presented next, we were curious of the performance of rate-adaptive routing algorithms within these architectures.

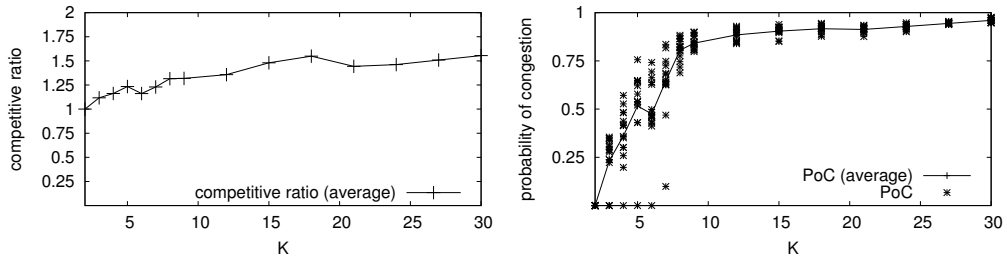
In what follows, for the input traffic matrix set Θ we used the setting $\Theta = T$. This choice allows us to contrast our results to those obtained for the conventional competitive ratio α in the literature (since $\alpha(T) = \gamma(T)$ in this case, as guaranteed by Lemma 3) and to compare to the optimum $\gamma(T) = 1$, which we know is attainable by a centralized algorithm. Extending the numerical analysis to the hose model is for further study.

Below, we show the results for three representative ISP topologies (other topologies showed similar behavior). The first topology is the NSFNET Phase II network [56], while the remaining two, namely AS 3257 (Tiscali, Europe) and AS 1239 (Sprint), come from the ISP data maps of the Rocketfuel dataset [57]. We used the same method as in [32] 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. Details of the topologies are given in Table 2.

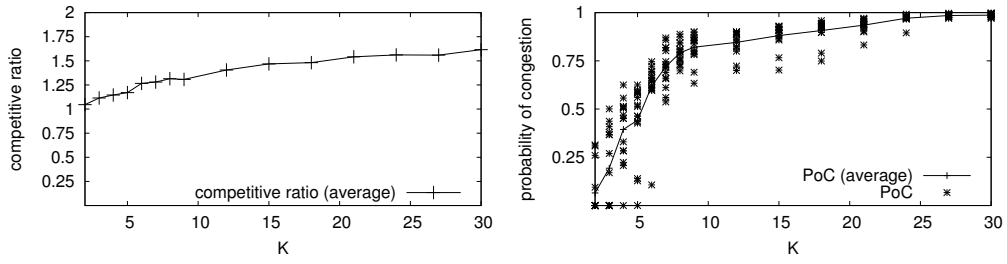
We found that it is the number of source-destination pairs K that determines the complexity and performance of the algorithms to the largest extent, as K directly influences the size and dimension of the linear programs. Thus, all our evaluations were run for increasing K, generating multiple scenarios for each



(a)



(b)



(c)

Figure 9: The absolute performance index γ and the probability of congestion η as the function of the number of users K for the distributed oblivious architecture on (a) the NSF, (b) the AS 3257, and (c) the AS 1239 topologies.

Table 2: Network topologies used in the evaluations

Name	Number of nodes	Number of links
NSFNET Phase II topology	12	30
AS3257 (Tiscali, Europe)	27	128
AS1239 (Sprint)	30	138

topology and each selection of K by selecting the source-destination pairs randomly, and finally averaging the results. The source-destination pairs themselves were chosen according to the bimodal distribution and 2 maximally node-disjoint paths were provisioned per user.

8.1. Distributed oblivious routing

First, we examined the distributed oblivious routing architecture arising over singular and block-diagonal routing functions. This corresponds to conventional oblivious routing, for which plenty of performance studies are available in the literature [32]. Our geometric approach, however, allows to define a new performance measure, called the *probability of congestion* (POC). The POC $\eta(\mathcal{S})$ is defined as the probability that a traffic matrix chosen according to a uniform distribution on T causes link over-utilization at some parts of the network when routed by a routing function $\mathcal{S}$:

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

where $\mathrm{Vol}(X)$ denotes the volume of the set X. In other words, $\eta(\mathcal{S})$ measures the quantity of traffic matrices routable by $\mathcal{S}$ to that of *all* routable traffic matrices. Easily, $\eta(\mathcal{S}) \in [0, 1]$ and for an optimal algorithm $\eta(\mathcal{S}) = 0$. Volumes were calculated using a home-grown Monte–Carlo integration code.

The average results for $\gamma(T)$ and $\eta(\mathcal{S})$ from 50 different evaluations on each K are given in Fig. 9. What is interesting in the diagrams is not that the worst-case congestion γ 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 γ, say, 1.5, there is an overwhelming chance (70-90%) that a randomly picked traffic matrix will overload some link in the network. This suggests that the competitive ratio α (which, recall, coincides with γ in this case) used extensively in the literature, is not really a good measure to characterize the performance of oblivious routing.

8.2. Centralized oblivious routing

In contrast to the distributed case, centralized oblivious routing is theoretically *guaranteed* to eliminate congestion for any traffic matrix in T. The price

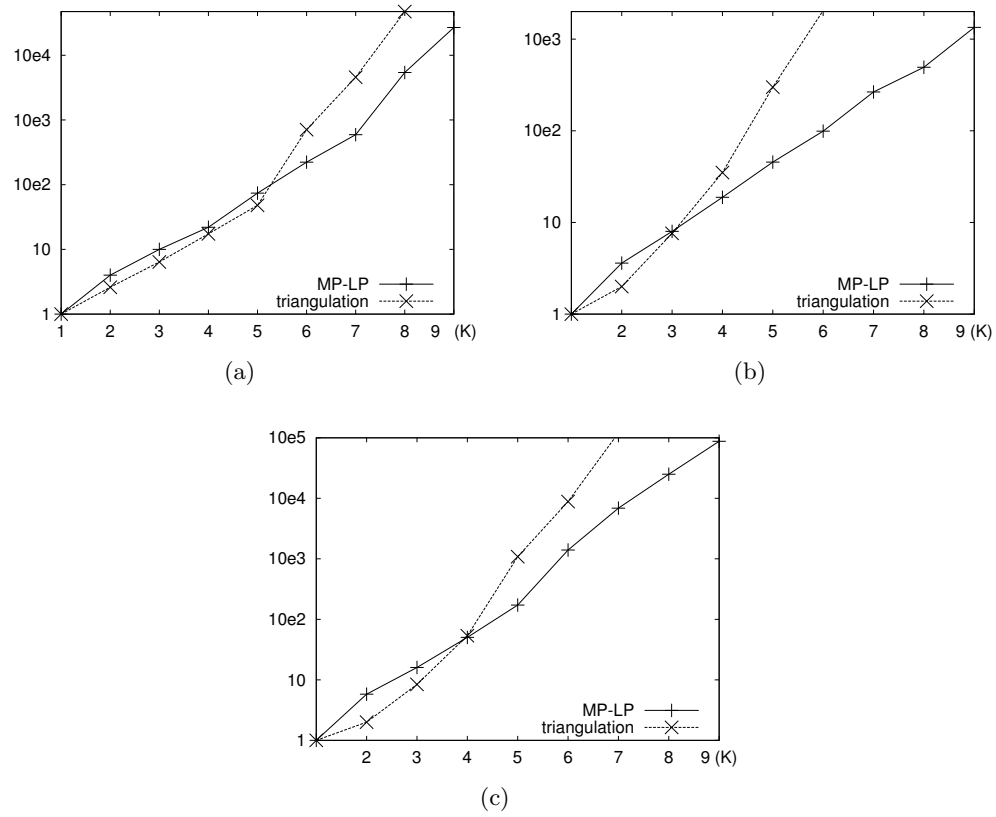


Figure 10: Number of control regions with the multiparametric programming ("MP-LP") and the triangulation technique, as the function of K for (a) the NSF, (b) the AS 3257, and (c) the AS 1239 topology, respectively.

we pay for optimality, however, is increased complexity, manifesting itself as an immense number of control regions needed to implement the routing function as evidenced by the results below.

We used both the multiparametric linear programming approach (Theorem 1) and the boundary triangulation method (Theorem 2) to obtain optimal routing controllers[5]. In particular, we used Multi-Parametric Toolbox (`MPT`, [58]) for Matlab to solve (21)–(23) and `polymake` [59] to compute boundary triangulations. The objective function for (21)–(23) was set for minimizing the total cost of the routing.

The average number of control regions, the principal measure of the complex-

[5]The code of the routing controllers, the network topologies and the path sets used in the paper are available at `https://github.com/ng201/mrx`.

Table 3: A summary and evaluation of generalized oblivious routing.

Routing architecture	Routing function	Pros	Cons
Distributed	singular block-diagonal	scalable, continuous, stable	inefficient
Centralized	compound general affine	optimal, optimizable, continuous	complex, unstable (dead-time)
Hybrid	compound block-diagonal	simple, asymptotically optimal, stable	not optimizable, only piecewise continuous

ity of compound routing functions, on 10 evaluations is depicted in Fig. 10 when K was varied between 1 and 9. The main observations are as follows. First, for networks serving only a couple of users centralized routing is clearly a viable option. However, complexity seems to increase exponentially with the number of users, and it becomes prohibitive when the total number of paths in the system surpasses about 20. In addition, we found that the multiparametric programming approach produces slightly better results. As an added bonus, it also allows for optimizing for arbitrary linear or quadratic objectives, therefore this approach seems more appealing for obtaining centralized routing controllers.

8.3. Hybrid oblivious routing

Finally, we asked to what extent the hybrid architecture can compensate for the inefficiency of the distributed architecture and the complexity of the centralized one.

The results were obtained as follows. We implemented the cutting plane algorithm described in Algorithm 1 in a mixture of Matlab, Perl and the GNU Linear Programming Kit [60]. We gradually increased the iteration limit from 0 to 8. Recall that in each iteration Algorithm 1 subdivides each region into two, which means that the complexity of the controller has increased from 1 to $2^8 = 256$ control regions. This way, the algorithm essentially reproduces conventional oblivious routing at the iteration depth of zero, and from this point it generates consecutively more complex and more efficient routings. The results for the worst-case congestion γ for $K = 7$, $K = 14$, and $K = 21$ averaged over 50 evaluations are depicted in Fig. 11. The level of significance is beyond 95%. Results are given both on linear routing functions (fixing g_k at zero in the affine routing functions $\mathcal{S}_k = f_k\theta_k + g_k$) and general affine functions as well. The former corresponds to setting traffic splitting ratios at routers, while the latter needs a somewhat more sophisticated (although still local) load distribution mechanism.

The results are convincing. In all cases our algorithm achieved significant performance improvement: for $K = 7$ it essentially eliminated congestion for AS 3257 and AS 1239 and halved it for NSF, and the reduction in worst-case congestion is well beyond the confidence level for larger settings of K as well.

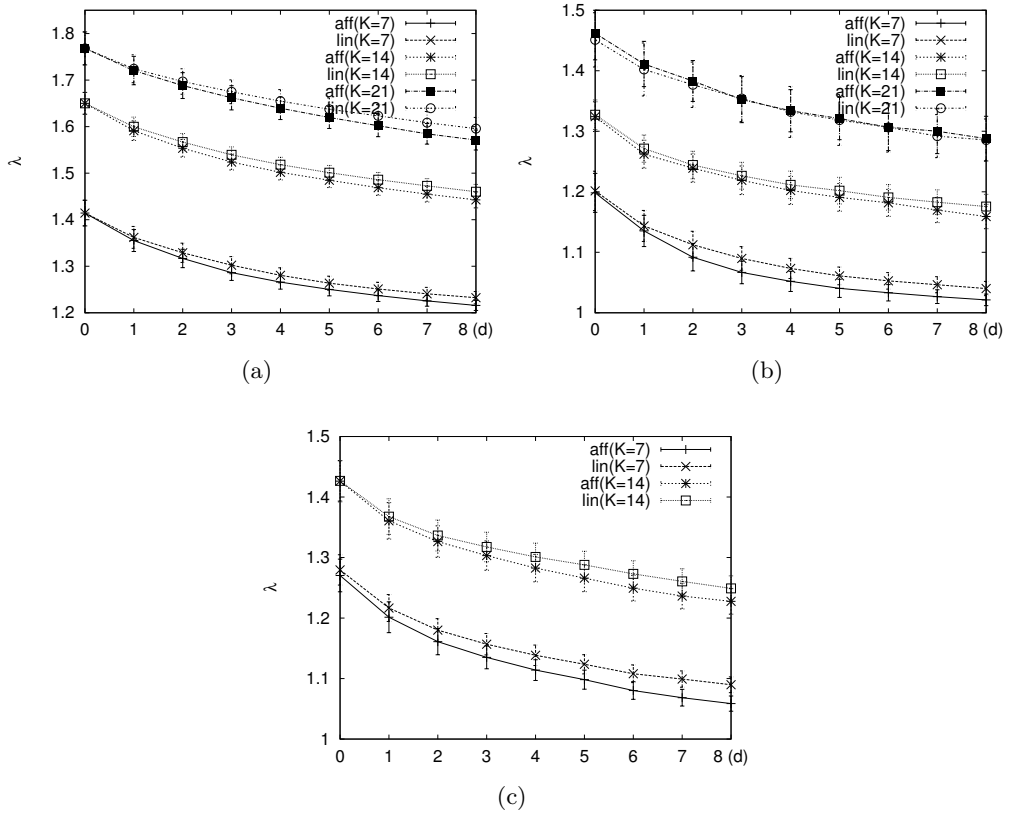


Figure 11: The absolute performance index γ for (a) the NSF, (b) the AS 3257, and (c) the AS 1239 topology, as the function of the iteration depth d with linear ("lin") and affine ("aff") routing functions for $K = 7$, $K = 14$, and $K = 21$.

Recall that this efficiency was achieved with only 256 control regions. We also see that general affine routing functions are somewhat more efficient than linear ones, but the improvement does not seem worth the increased implementation complexity.

9. Conclusions

In this paper, we have introduced generalized oblivious routing as a general model for rate-adaptive multipath routing. The model is defined by three principles, namely, that the mapping from traffic matrices to routes (i.e., the routing function) is completely pre-computed; the routing algorithm does not use any network state or feedback signal apart from the amount of traffic seen at network ingress; and the objective is to minimize the worst-case congestion. It turned out that just these principles, when placed into an expressive geometric framework, can give rise to a rich theory of rate adaptive routing. A summary on the generalized oblivious architectures, the routing functions they are generated with, and the advantages and disadvantages thereof, is given in Table 3.

It seems that the hybrid distributed-centralized scheme realizes an appealing trade-off between the two extremes. It needs only minimal control to pick the routing that best fits the actual user demands, and to do this it uses information that is often present in central network management software anyways. Once the correct routing function is downloaded, routers do not need further central management as long as the traffic matrix does not change too much to warrant a transition to another control region. With the theoretical guarantee on asymptotic optimality, the hybrid scheme looks an appealing option for deploying traffic engineering in service provider networks.

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] S. Gunnar, M. Johansson, and T. Telkamp. Traffic matrix estimation on a large IP backbone: a comparison on real data. In *ACM SIGCOMM conference on Internet measurement*, IMC'04, pages 149–160, 2004.

[3] Kuai Xu, Zhi-Li Zhang, and Supratik Bhattacharyya. Profiling internet backbone traffic: behavior models and applications. *SIGCOMM Comput. Commun. Rev.*, 35:169–180, August 2005.

[4] D. G. Cantor and M. Gerla. Optimal routing in a packet-switched computer network. *IEEE Transactions on Computer*, 23(10):1062–1069, 1974.

[5] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional IP routing protocols. *IEEE Communications Magazine*, 40(10):118–124, Oct 2002.

[6] Matthew Roughan, Albert Greenberg, Charles Kalmanek, Michael Rumsewicz, Jennifer Yates, and Yin Zhang. Experience in measuring backbone traffic variability: models, metrics, measurements and meaning. In *ACM SIGCOMM Workshop on Internet measurment*, IMW '02, pages 91–92, 2002.

[7] Renata Teixeira, Aman Shaikh, Tim Griffin, and Jennifer Rexford. Dynamics of hot-potato routing in IP networks. In *Conference on Measurement and modeling of computer systems*, SIGMETRICS '04/Performance '04, pages 307–319, 2004.

[8] Renata Teixeira, Sharad Agarwal, and Jennifer Rexford. BGP routing changes: merging views from two ISPs. *SIGCOMM Comput. Commun. Rev.*, 35:79–82, October 2005.

[9] Renata Teixeira, Nick G. Duffield, Jennifer Rexford, and Matthew Roughan. Traffic matrix reloaded: Impact of routing changes. In *PAM'05*, pages 251–264, 2005.

[10] Murali Kodialam, T. V. Lakshman, and Sudipta Sengupta. Efficient and robust routing of highly variable traffic. In *In Proceedings of Third Workshop on Hot Topics in Networks (HotNets-III*, 2004.

[11] A. Kvalbein, C. Dovrolis, and C. Muthu. Multipath load-adaptive routing: putting the emphasis on robustness and simplicity. In *IEEE International Conference on Network Protocols*, ICNP 2009, pages 203–212, oct. 2009.

[12] Hao Wang, Haiyong Xie, Lili Qiu, Yang Richard Yang, Yin Zhang, and Albert Greenberg. COPE: traffic engineering in dynamic networks. *SIGCOMM Comput. Commun. Rev.*, 36(4):99–110, 2006.

[13] M. Roughan, M. Thorup, and Y. Zhang. Traffic engineering with estimated traffic matrices. In *ACM SIGCOMM conference on Internet measurement*, IMC '03, pages 248–258, 2003.

[14] C. Zhang, Y. Liu, W. Gong, J. Moll, and R. D. Towsley. On optimal routing with multiple traffic matrices. In *INFOCOM 2005*, volume 1, pages 607–618, 2005.

[15] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal of Selected Areas in Communications*, 20(4):756–767, May 2002.

[16] S. Suri, M. Waldvogel, and P. R. Warkhede. Profile-based routing: a new framework for MPLS traffic engineering. In F. Boavida, editor, *Quality of Future Internet Services*, volume 2156 of *LNCS*. Springer, 2001.

[17] D. Medhi. Multi-hour, multi-traffic class network design for virtual path-based dynamically reconfigurable wide-area ATM networks. *IEEE/ACM Transactions on Networking*, 3(6):809–818, 1995.

[18] R. Gallager. A minimum delay routing algorithm using distributed computation. *Communications, IEEE Transactions on*, 25(1):73–85, jan 1977.

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

[20] Frank Kelly and Thomas Voice. Stability of end-to-end algorithms for joint routing and rate control. *SIGCOMM Comput. Commun. Rev.*, 35(2):5–12, 2005.

[21] P. Key, L. Massoulie, and P.D. Towsley. Path selection and multipath congestion control. In *INFOCOM 2007*, pages 143 –151, May 2007.

[22] J. He, M. Bresler, M. Chiang, and J. Rexford. Towards robust multi-layer traffic engineering: Optimization of congestion control and routing. *Selected Areas in Communications, IEEE Journal on*, 25(5):868–880, June 2007.

[23] Constantino M. Lagoa, Hao Che, and Bernardo A. Movsichoff. Adaptive control algorithms for decentralized optimal traffic engineering in the internet. *IEEE/ACM Trans. Netw.*, 12(3):415–428, 2004.

[24] Srikanth Kandula, Dina Katabi, Bruce Davie, and Anna Charny. Walking the Tightrope: Responsive Yet Stable Traffic Engineering. In *ACM SIGCOMM'05*, August 2005.

[25] Simon Fischer, Nils Kammenhuber, and Anja Feldmann. REPLEX: dynamic traffic engineering based on wardrop routing policies. In *CoNEXT'06*, pages 1–12, 2006.

[26] Jukka Suomela. Survey of local algorithms. *ACM Comput. Surv.*, 45(2):24:1–24:40, March 2013.

[27] A. Khanna and J. Zinky. The revised ARPANET routing metric. *SIGCOMM Comput. Commun. Rev.*, 19(4):45–56, 1989.

[28] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *ACM*

symposium on Theory of computing, STOC '81, pages 263–277, 1981.

[29] Harald Räcke. Minimizing congestion in general networks. In *IEEE Symposium on Foundations of Computer Science*, FOCS '02, pages 43–52, 2002.

[30] Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *ACM symposium on Theory of computing*, STOC '08, pages 255–264, 2008.

[31] Yossi Azar, Edith Cohen, Amos Fiat, Haim Kaplan, and Harald Räcke. Optimal oblivious routing in polynomial time. In *ACM symposium on Theory of computing*, STOC '03, pages 383–388, 2003.

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

[33] Matthias Englert and Harald Räcke. Oblivious Routing for the L_p-norm. *IEEE Foundations of Computer Science*, pages 32–40, 2009.

[34] Harald Räcke. Survey on oblivious routing strategies. In *Proceedings of the 5th Conference on Computability in Europe: Mathematical Theory and Computational Practice*, CiE '09, pages 419–429, 2009.

[35] Mung Chiang, S.H. Low, A.R. Calderbank, and J.C. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, 2007.

[36] Jung-Hoon Yun, Anseok Lee, and Song Chong. Multi-path aggregate flow control for real-time traffic engineering. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pages 1 –5, 2008.

[37] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley. Multi-Path TCP: A joint congestion control and routing scheme to exploit path diversity in the Internet. *Networking, IEEE/ACM Transactions on*, 14(6):1260 –1271, dec 2006.

[38] Jiayue He, Martin Suchara, Ma'ayan Bresler, Jennifer Rexford, and Mung Chiang. Rethinking internet traffic management: from multiple decompositions to a practical protocol. In *ACM CoNEXT'07*, pages 1–12, 2007.

[39] D. Xu, M. Chiang, and J. Rexford. Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering. *Networking, IEEE/ACM Transactions on*, 19(6):1717–1730, 2011.

[40] Yair Bartal and Stefano Leonardi. On-line routing in all-optical networks. In *Proceedings of the 24th International Colloquium on Automata, Languages and Programming*, ICALP '97, pages 516–526, 1997.

[41] Walid Ben-Ameur and Mateusz otkiewicz. Robust routing and optimal partitioning of a traffic demand polytope. *International Transactions in Operational Research*, 18(3):307–333, 2011.

[42] G. Rétvári and G. Németh. Demand-oblivious routing: distributed vs. centralized approaches. In *INFOCOM 2010*, March 2010.

[43] G. Németh and G. Rétvári. Hybrid demand oblivious routing: Hyper-cubic partitions and theoretical upper bounds. In *BROADNETS*, 2010.

[44] G. Rétvári and G. Németh. On optimal multipath rate-adaptive routing. In *15th IEEE Symposium on Computers and Communications (ISCC 2010)*, Riccione, Italy, 2010.

[45] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, April 2008.

[46] L. Yang, R. Dantu, T. Anderson, and R. Gopal. Forwarding and control element separation (ForCES) framework. RFC 3746, April 2004.

[47] J.Andrew Fingerhut, Subhash Suri, and Jonathan S. Turner. Designing least-cost nonblocking broadband networks. *Journal of Algorithms*, 24(2):287–309, 1997.

[48] G.M. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer, 1998.
[49] B. Grünbaum. *Convex Polytopes*. John Wiley & Sons, 1967.
[50] L. Boróczki. Átvihető folyamokat leíró poliéder keresése hálózatokban (in Hungarian), 2006. Student's Tech. Rep.
[51] G. Rétvári, J. J. Bíró, and T. Cinkler. Fairness in capacitated networks: A polyhedral approach. In *INFOCOM 2007*, volume 1, pages 1604–1612, May 2007.
[52] F. Borrelli, A. Bemporad, and M. Morari. Geometric algorithm for multiparametric linear programming. *Journal of Optimization Theory and Applications*, 118:515–540, September 2003.
[53] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38:3–20, January 2002.
[54] Alexander Below. *Complexity of triangulation*. Doctoral thesis, Diss., Technische Wissenschaften ETH Zurich, Nr. 14672, 2002, 2002.
[55] T. Johansen, A. Grancharova, and R. Grancharova. Approximate explicit constrained linear model predictive control via orthogonal search tree. *IEEE Trans. Automatic Control*, 48:810–815, 2003.
[56] 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.
[57] Ratul Mahajan, Neil Spring, David Wetherall, and Tom Anderson. Inferring link weights using end-to-end measurements. In *ACM SIGCOMM Workshop on Internet measurment*, IMW '02, pages 231–236, 2002.
[58] M. Kvasnica, P. Grieder, and M. Baotić. Multi-Parametric Toolbox (MPT), 2004. available online: `http://control.ee.ethz.ch/~mpt/`.
[59] Ewgenij Gawrilow and Michael Joswig. polymake. `http://www.math.tu-berlin.de/polymake/`.
[60] The GLPK project: `http://www.gnu.org/software/glpk/glpk.html`.

Appendix

Proof (Proof of Lemma 1). Throughput-invariance on the singular affine routing function $\mathcal{S}(\theta) = F\theta + g$ at the point $\theta = 0$ gives $\mathcal{T}(\mathcal{S}(0)) = \mathcal{T}(g) \equiv 0$, which can hold only if $g \equiv 0$. □

Proof (Proof of Lemma 2). Let $\mathcal{P}_k = \mathcal{P}_k^- \cup \mathcal{P}_k^+$, where $\forall p \in \mathcal{P}_k^- : f_{kp} < 0$ and $\forall p \in \mathcal{P}_k^+ : f_{kp} \geq 0$. Moreover, $\forall k \in \mathcal{K}$ let $\tau_k = \min\{\theta_k : \theta \in \mathcal{R}(\mathcal{S})\}$ and $T_k = \max\{\theta_k : \theta \in \mathcal{R}(\mathcal{S})\}$. Obviously, if $\tau_k = T_k$, then the routing function $f'_{kp} = (S_k(T_k))_p/T_k$, $g'_{kp} = 0$ satisfies the properties prescribed in the lemma. Otherwise, we construct $\mathcal{S}'$ as follows. Set

$$\forall p \in \mathcal{P}_k^- : f'_{kp} = 0 \text{ and } g'_{kp} = f_{kp}T_k + g_{kp} \ .$$

Note that $\sum_{p\in\mathcal{P}_k^-} g'_{kp} \leq \tau_k$, thus, $T_k - \sum_{p\in\mathcal{P}_k^-} g'_{kp} > 0$. Moreover, let $\nu = \sum_{q\in\mathcal{P}_k^-} g'_{kq}$, and $\forall p \in \mathcal{P}_k^+$ let $\rho_p = f_{kp}T_k + g_{kp}$ and $\mu_p = \frac{\tau_k-\nu}{T_k-\nu}(f_{kp}T_k + g_{kp})$. Finally, set

$$\forall p \in \mathcal{P}_k^+ : f'_{kp} = \frac{\rho_p - \mu_p}{(T_k - \tau_k)}$$

$$g'_{kp} = f_{kp}\tau_k + g_{kp} - \frac{\rho_p - \mu_p}{T_k - \tau_k}\tau_k \ .$$

We show that S' is a routing function with the required properties. First, *(i)* $\mathcal{S}'_k(\tau_k) \geq 0$ and $\mathcal{S}'_k(T_k) \geq 0$; *(ii)* throughput-invariance holds for $\mathcal{S}'_k(\tau_k)$ and $\mathcal{S}'_k(T_k)$; *(iii)* by convexity, non-negativity and throughput invariance holds on the entire set $\theta_k \in [\tau_k, T_k]$; and finally *(iv)* $\forall k \in \mathcal{K}, \forall p \in \mathcal{P}_k, \forall \theta_k \in [\tau_k, T_k]$: $(\mathcal{S}'_k(\theta_k))_p \leq (\mathcal{S}'_k(T_k))_p = (\mathcal{S}_k(T_k))_p$, so $\mathcal{R}(\mathcal{S}) \subseteq \mathcal{R}(\mathcal{S}')$. □

Proof (Proof of Lemma 3). Since $\mathcal{S}$ is singular and linear, $\mathcal{S}(\lambda\theta) = \lambda\mathcal{S}(\theta)$ for any scalar $\lambda \in \mathbb{R}$. Therefore, $\kappa_{\mathcal{S}}(\theta)$ is spherical, and hence it reaches its maximum on the boundary ∂T of T. Then, we get

$$\begin{aligned}\gamma_{\mathcal{S}}(T) &= \max_{\theta \in T} \kappa_{\mathcal{S}}(\theta) = \max_{\theta \in \partial T} \kappa_{\mathcal{S}}(\theta) \\ &= \max_{\theta \in \partial T} \frac{\kappa_{\mathcal{S}}(\theta)}{\kappa_{\mathcal{S}_{\mathrm{OPT}}}(\theta)} = \alpha_{\mathcal{S}}(T) \ ,\end{aligned}$$

using the trivial observation that $\forall \theta \in \partial T : \kappa_{\mathcal{S}_{\mathrm{OPT}}}(\theta) = 1$. □

Proof (Proof of Theorem 2). Let $\theta_1, \ldots, \theta_s$ be the extreme points of T and let $u_1, \ldots, u_s$ be path-flows realizing these extreme points. For any simplex Q_j in a boundary triangulation of T, $Q_j = \mathrm{Conv}\{\theta_{i_0}, \theta_{i_1}, \ldots, \theta_{i_K}\}$ for some $K+1$ affinely independent extreme points of T. Let $u_{i_0}, \ldots, u_{i_K}$ be the path-flows corresponding to the extreme points $\theta_{i_0}, \ldots, \theta_{i_K}$. Without loss of generality, choose θ_{i_0} as a basis point and define the $K \times K$ matrix $B_j = [\theta_{i_1} - \theta_{i_0}, \ldots, \theta_{i_K} - \theta_{i_0}]$. Note that B_j is invertible. Let u_{i_0} be the path-flow realizing θ_{i_0} and U_j be a $p \times K$ matrix defined as $U_j = [u_{i_1} - u_{i_0}, \ldots, u_{i_K} - u_{i_0}]$.

Consider an arbitrary point $\theta \in Q_j$. Then, there exist $\lambda_0, \lambda_1, \ldots, \lambda_K$, $\lambda_k \geq 0$, $\sum_{k=0}^{K} \lambda_k = 1$, so that

$$\begin{aligned}\theta &= \sum_{k=0}^{K} \lambda_k \theta_{i_k} = \sum_{k=0}^{K} \lambda_k \theta_{i_k} - \theta_{i_0} + \theta_{i_0} \\ &= \sum_{k=0}^{K} \lambda_k \theta_{i_k} - \sum_{k=0}^{K} \lambda_k \theta_{i_0} + \theta_{i_0} = \sum_{k=0}^{K} \lambda_k (\theta_{i_k} - \theta_{i_0}) + \theta_{i_0} \\ &= \sum_{k=1}^{K} \lambda_k (\theta_{i_k} - \theta_{i_0}) + \theta_{i_0} = B_j \lambda + \theta_{i_0} \ , \end{aligned} \tag{24}$$

where λ is a column K-vector formed by the coordinates λ_k. Consider the path-flow

$$u = \sum_{k=0}^{K} \lambda_k u_{i_k} = \sum_{k=1}^{K} \lambda_k (u_{i_k} - u_{i_0}) + u_{i_0} = U_j \lambda + u_{i_0} \ . \tag{25}$$

Observe that u is a routing for θ, as $\mathcal{T}(u) = \mathcal{T}(U\lambda + u_{i_0}) = \mathcal{T}(U)\lambda + \mathcal{T}(u_{i_0}) = B\lambda + \theta_{i_0} = \theta$. In addition, $u \in M$ as u is a convex combination of vectors in M. Noting that $\lambda = B_j^{-1}(\theta - \theta_{i_0})$ by (24), we have that

$$u = U_j B_j^{-1}(\theta - \theta_{i_0}) + u_{i_0} = U_j B_j^{-1}\theta + (u_{i_0} - U_j B_j^{-1}\theta_{i_0}) \tag{26}$$

is a feasible singular affine routing function for θ. Since the above holds for any $\theta \in Q$ we conclude that (26) is a routing function on the entire simplex Q_j. Using the above construction on each $Q_i : i \in \{1, \ldots, q\}$ gives a compound affine routing function $\mathcal{S} = \{(\mathcal{S}^j, Q_j) : j \in \{1, \ldots, q\}\}$. Finally, continuity is trivial by (24) and (25). □

Proof (Proof of Theorem 3). Instead of proving this theorem directly, we rather prove a more general result which establishes the continuity of γ over any sequence of sets whose elements are sufficiently "close". Here, closeness is defined in terms of the Hausdorff distance d_H: given polytopes P and Q, $d_H(P, Q) = \max_{x \in P} \min_{y \in Q} ||x - y||$. The following lemma serves as the basis if the proof.

Lemma 4. *Let $X \subset \mathbb{R}_+^K$ and let $\mathcal{S} : \mathcal{S}_k(\theta) = f_k\theta_k + g_k$ be a singular block-diagonal routing function with $\forall k \in \mathcal{K} : f_k \geq 0$ that solves (3) on X. Then, for any $X \subseteq X' \subset \mathbb{R}_+^K$ with $d_H(X, X') \leq \epsilon$, there is a modified singular block-diagonal routing function $\mathcal{S}'$ on X' so that*

$$0 \leq \gamma_{\mathcal{S}'}(X') - \gamma_{\mathcal{S}}(X) \leq \frac{K\Pi\epsilon}{\Lambda} ,$$

where Λ is the minimum link capacity and $\Pi = \max_{k \in \mathcal{K}} \{p_k\}$.

Proof. Starting from the block-diagonal routing function $\mathcal{S} : \mathcal{S}_k(\theta) = f_k\theta_k + g_k, k \in \mathcal{K}$ we construct a modified routing function $\mathcal{S}$ so that $\forall \theta \in X'$: $\mathcal{S}'(\theta) \geq 0$ and $\gamma_{\mathcal{S}}(X) \leq \gamma_{\mathcal{S}'}(X') \leq \gamma_{\mathcal{S}}(X) + \frac{K\Pi\epsilon}{\Lambda}$. Note that $\gamma_{\mathcal{S}'}(X') \geq \gamma_{\mathcal{S}}(X)$ is trivial by $X \subseteq X'$. For each $k \in \mathcal{K}$, define τ_k and T_k as above, and let $\tau_k' = \min_{\theta \in X'} \theta_k$ and $T_k' = \max_{\theta \in X'} \theta_k$. Note that $\tau_k' \leq \tau_k$ and $T_k' \geq T_k$, and $|\tau_k - \tau_k'| \leq \epsilon$ and $|T_k - T_k'| \leq \epsilon$ as $d_H(X, X') \leq \epsilon$. Let $\rho = \mathcal{S}_k(\tau_k)\frac{\tau_k'}{\tau_k} = f_k\tau_k' + \frac{\tau_k'}{\tau_k}g_k$ and $\mu = \mathcal{S}_k(T_k)\frac{T_k'}{T_k} = f_kT_k' + \frac{T_k'}{T_k}g_k$, and consider the function $\mathcal{S}_k'(\theta_k) = \rho + (\mu - \rho)\frac{\theta_k - \tau_k'}{T_k' - \tau_k'}$. Note that $\{\mathcal{S}_k'(\theta_k) : \theta_k \in [\tau_k', T_k']\} = \mathrm{Conv}(\rho, \mu)$. We observe that *(i)* $\mathcal{S}_k'(\tau_k') = \rho \geq 0$ and $\mathcal{S}_k'(T_k') = \mu \geq 0$; *(ii)* $\forall \theta_k \in [\tau_k, T_k]$: $\mathcal{S}_k'(\theta) \geq 0$ by the convexity of $\mathcal{S}_k'$; and *(iii)* throughput-invariance holds for ρ and μ, and so for $\mathcal{S}_k'$ too again by convexity. Hence, $\mathcal{S}' : \mathcal{S}_k', k \in \mathcal{K}$ is an affine block-diagonal routing function. If $\mathcal{S}$ is linear, so is $\mathcal{S}'$. Next, we show that $\mathcal{S}$ and $\mathcal{S}'$ are close to each other in the terms of the Hausdorff metric:

$$\forall \theta_k' \in \left[\tau_k', T_k'\right], \exists \theta_k \in [\tau_k, T_k] : |\mathcal{S}_k'(\theta_k') - \mathcal{S}_k(\theta_k)| \leq \underline{1}\epsilon \tag{27}$$

First, we prove that $\exists \theta_k \in [\tau_k, T_k] : |S'_k(\tau'_k) - S_k(\theta_k)| \leq \underline{1}\epsilon$. We write $|S'_k(\tau'_k) - S_k(\theta_k)| \leq |S'_k(\tau'_k) - S_k(\tau_k)| = |\mathcal{S}_k(\tau_k)\frac{\tau'_k}{\tau_k} - S_k(\tau_k)| = \mathcal{S}_k(\tau_k)\frac{\epsilon}{\tau_k} \leq \underline{1}\epsilon$. Here, we use the fact that $f_k \geq 0$ and $\underline{1}^T f_k = 1$, thus $f_k \leq \underline{1}$. Similar argumentation shows $\exists \theta_k \in [\tau_k, T_k] : |S'_k(T'_k) - S_k(\theta_k)| \leq \underline{1}\epsilon$, which proves (27) by convexity. Putting it all together:

$$
\begin{aligned}
&|\gamma_{S'}(X') - \gamma_S(X)| \\
&= \left| \max_{(i,j)\in E} \max_{\theta \in X'} \frac{\sum_{k=1}^K P_k^{ij} \mathcal{S}'_k(\theta)}{c_{ij}} - \max_{(i,j)\in E} \max_{\theta \in X} \frac{\sum_{k=1}^K P_k^{ij} \mathcal{S}_k(\theta)}{c_{ij}} \right| \\
&\leq \max_{(i,j)\in E} \left\{ \left| \max_{\theta \in X'} \frac{\sum_{k=1}^K P_k^{ij} \mathcal{S}'_k(\theta)}{c_{ij}} - \max_{\theta \in X} \frac{\sum_{k=1}^K P_k^{ij} \mathcal{S}_k(\theta)}{c_{ij}} \right| \right\} \\
&\leq \max_{(i,j)\in E} \frac{1}{c_{ij}} \left\{ \left| \sum_{k=1}^K P_k^{ij} \max_{\theta \in X'} \mathcal{S}'_k(\theta) - \sum_{k=1}^K P_k^{ij} \max_{\theta \in X} \mathcal{S}_k(\theta) \right| \right\} \\
&\leq \max_{(i,j)\in E} \frac{1}{c_{ij}} \left\{ \sum_{k=1}^K P_k^{ij} |\max_{\theta \in X'} \mathcal{S}'_k(\theta) - \max_{\theta \in X} \mathcal{S}_k(\theta)| \right\} \\
&\overset{(1)}{\leq} \max_{(i,j)\in E} \frac{1}{c_{ij}} \left\{ \sum_{k=1}^K P_k^{ij} \underline{1}\epsilon \right\} \leq \frac{K\Pi\epsilon}{\Lambda} \ ,
\end{aligned}
$$

where *(1)* comes by (27). □

A simple corollary of the above proves the increasing property and left-continuity of $\gamma_1(t)$.

Corollary 2. *Let $X \subset \mathbb{R}_+^K$ and let $\mathcal{S} : \mathcal{S}_k(\theta) = f_k\theta_k + g_k$ be a singular block-diagonal routing function with $\forall k \in \mathcal{K} : f_k \geq 0$ that solves* (3) *on X. Then, for any $\delta > 0$ there is $\epsilon > 0$ so that for any convex set $X' \supset X$ with $d_H(X, X') \leq \epsilon$ and the singular block-diagonal routing function $\mathcal{S}'$ that solves* (3) *on X': $|\gamma_{\mathcal{S}}(X) - \gamma_{\mathcal{S}'}(X')| \leq \delta$.*

To show right-continuity, we need a little more work.

Corollary 3. *Let $X \subset \mathbb{R}_+^K$ and let $\mathcal{S} : \mathcal{S}_k(\theta) = f_k\theta_k + g_k$ be a singular block-diagonal routing function with $\forall k \in \mathcal{K} : f_k \geq 0$ that solves* (3) *on X. Then, for any $\delta > 0$ there is $\epsilon > 0$ so that for any convex set $X' \subset X$ with $d_H(X, X') \leq \epsilon$ and the singular block-diagonal routing function $\mathcal{S}'$ that solves* (3) *on X': $|\gamma_{\mathcal{S}}(X) - \gamma_{\mathcal{S}'}(X')| \leq \delta$.*

Proof. Suppose we know $\mathcal{S}'$. Then, using Lemma 4 there is a routing function $\mathcal{S}''$ defined over X' so that $\gamma_{\mathcal{S}'}(X') \leq \gamma_{\mathcal{S}''}(X)$. Then, due to the optimality of $\mathcal{S}$ on X: $|\gamma_{S'}(X') - \gamma_S(X)| \leq |\gamma_{S'}(X') - \gamma_{S''}(X)| \leq \frac{K\Pi\epsilon}{\Lambda} = \delta$. □

Proving the other claim of Theorem 3 that $\gamma_2(t) = \gamma(X_2(t))$ is decreasing and continuous goes along similar lines. □

PROOF (PROOF OF THEOREM 4). Let $\mathcal{S}^* : \{(\mathcal{D}^i, \mathcal{S}^i) : i \in \mathcal{I}\}$ be a compound block-diagonal routing function, obtained by executing Algorithm 1 over the cutting planes generated by Algorithm 2 on the input set Θ. For column K-vectors a and b: $0 \leq a \leq b$, define the K-dimensional hyper-rectangle $H_a^b = \underset{k=1}{\overset{K}{\times}} [a_k, b_k]$. Let $H^b = H_a^b : a = 0$. For any $\theta \in \mathbb{R}_+^K$ let u be the path-flow vector that minimizes the maximum link utilization for θ and define the *trivial routing function* $\mathcal{S}^\theta : \mathcal{S}_k^\theta(\tau) = \frac{u_k}{\theta_k}\tau_k, k \in \mathcal{K}$ over the singleton set $\{\theta\}$.

Lemma 5. *Let $\mathcal{S}^\theta$ be a trivial routing function for some $\theta \in \Theta$.*

(i) $\mathcal{S}^\theta$ is a singular, linear, block-diagonal routing function;

(ii) $\gamma_{\mathcal{S}^*}(H^\theta) \leq \gamma_{\mathcal{S}^\theta}(H^\theta)$; and

(iii) if $\theta \in T$, then $\gamma_{\mathcal{S}^\theta}(H^\theta) \leq 1$.

PROOF. Item *(i)* is trivial from the definition, *(ii)* comes by observing that $\mathcal{S}^\theta$ is a feasible solution of (3) for each $\mathcal{D}^i : i \in \mathcal{I}$ while $\mathcal{S}^*$ is optimal, and *(iii)* is by the down-monotonity of $\mathcal{R}(\mathcal{S}_\theta)$. □

Lemma 6. *Let $0 \leq a \leq b : a, b \in \mathbb{R}_+^K$ with $\max_{k\in\mathcal{K}}(b_k - a_k) \leq \epsilon$ and $H_a^b \cap T \neq \emptyset$, let $\tau = \min\{\tau : \tau \in H_a^b\}$ and let $\mathcal{S}^\tau$ be the trivial routing function for τ. Then,*

$$\gamma_{\mathcal{S}_\tau}(H_a^b \cap T) \leq 1 + \max_{(i,j)\in E} \left\{\frac{\xi_{ij}}{c_{ij}}\right\} \epsilon \ . \tag{28}$$

PROOF. If $H_a^b \subseteq T$, then $\gamma_{\mathcal{S}^\tau}(H_a^b) \leq 1$ by Lemma 5 and so (28) obviously holds. Otherwise, we write

$$\begin{aligned}
\gamma_{\mathcal{S}^\tau}(H_a^b) &\leq \max_{(i,j)\in E} \max_{\theta\in H_a^b \cap T} \frac{\sum_{k=1}^K P_k^{ij}\mathcal{S}_k^\tau(\theta)}{c_{ij}} \\
&\leq \max_{(i,j)\in E} \max_{\theta\in H_a^b} \frac{\sum_{k=1}^K P_k^{ij}\mathcal{S}_k^\tau(\theta)}{c_{ij}} \leq \max_{(i,j)\in E} \frac{\sum_{k=1}^K P_k^{ij}\mathcal{S}_k^\tau(\tau + \underline{1}\epsilon_k)}{c_e} \\
&\leq \max_{(i,j)\in E} \left\{ \frac{\sum_{k=1}^K P_k^{ij}\mathcal{S}_k^\tau(\tau)}{c_{ij}} + \frac{\sum_{k=1}^K P_k^{ij}\mathcal{S}_k^\tau(\underline{1}\epsilon)}{c_{ij}} \right\} \\
&\overset{\text{(i)}}{\leq} 1 + \max_{(i,j)\in E} \frac{\sum_{k=1}^K P_k^{ij}\mathcal{S}_k^\tau(\underline{1}\epsilon)}{c_{ij}} \overset{\text{(ii)}}{\leq} 1 + \max_{(i,j)\in E} \left\{\frac{\xi_{ij}}{c_{ij}}\right\} \epsilon \ ,
\end{aligned}$$

where *(i)* is because $\tau \in T$ by the down-monotonity of T and *(ii)* comes by the definition of ξ_{ij}. □

Putting the above results together: $\gamma_{\mathcal{S}^*}(T) = \max_{i \in I} \gamma_{\mathcal{S}^i}(\mathcal{D}^i) \leq \max_{i \in I} \gamma_{\mathcal{S}^i}(H_{a_i}^{b_i} \cap T) \leq 1 + \max_{(i,j) \in E} \left\{ \frac{\xi_{ij}}{c_{ij}} \right\} \epsilon$ using Lemma 5 and Lemma 6. $\square$