

# Optimal Oblivious Routing with Concave Objectives for Structured Networks

Kanatip Chitavisutthivong, Sucha Supittayapornpong, Pooria Namyar, Mingyang Zhang,  
Minlan Yu, and Ramesh Govindan

**Abstract**—Oblivious routing distributes traffic from sources to destinations following predefined routes with rules independent of traffic demands. While finding optimal oblivious routing with a concave objective is intractable for general topologies, we show that it is tractable for structured topologies often used in datacenter networks. To achieve this, we apply graph automorphism and prove the existence of the optimal automorphism-invariant solution. This result reduces the search space to targeting the optimal automorphism-invariant solution. We design an iterative algorithm to obtain such a solution by alternating between convex optimization and a linear program. The convex optimization finds an automorphism-invariant solution based on representative variables and constraints, making the problem tractable. The linear program generates adversarial demands to ensure the final result satisfies all possible demands. Since the construction of the representative variables and constraints are combinatorial problems, we design polynomial-time algorithms for the construction. We evaluate the iterative algorithm in terms of throughput performance, scalability, and generality over three potential applications. The algorithm i) improves the throughput up to 87.5% for partially deployed FatTree and achieves up to  $2.55\times$  throughput gain for DRing over heuristic algorithms, ii) scales for three considered topologies with a thousand switches, iii) applies to a general structured topology with non-uniform link capacity and server distribution.

**Index Terms**—Oblivious routing, datacenter networks, convex optimization, graph automorphism.

## I. INTRODUCTION

Topology design for datacenter networks has gained attention due to the need to construct high capacity datacenters at low cost and low management complexity [1]–[12]. Although several topologies have been proposed, only the folded-Clos family of topologies [1]–[4] achieves designed capacity with existing routing solutions including Equal-Cost Multi Path (ECMP) [13] and Valiant Load-Balancing (VLB) [14], [15]. For topology designs that deviate from folded-Clos [6]–[11]

This work was supported by Office of the Permanent Secretary, Ministry of Higher Education, Science, Research and Innovation (OPS MHESI), Thailand Science Research and Innovation (TSRI), Vidyasirimedhi Institute of Science and Technology (VISTEC) under Grant No. RGNS 65-216, in part by the U.S. National Science Foundation under grant No. CNS-1955422, and in part by ACE, one of the seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA. The early version of this work appeared in the Proceedings of IEEE INFOCOM 2022 [DOI: 10.1109/INFOCOM48880.2022.9796682].

Corresponding author: Sucha Supittayapornpong (sucha.s@vistec.ac.th).

Kanatip Chitavisutthivong and Sucha Supittayapornpong are with Vidyasirimedhi Institute of Science and Technology, Thailand.

Pooria Namyar, Mingyang Zhang, and Ramesh Govindan are with University of Southern California, United States.

Minlan Yu is with Harvard University, United States.

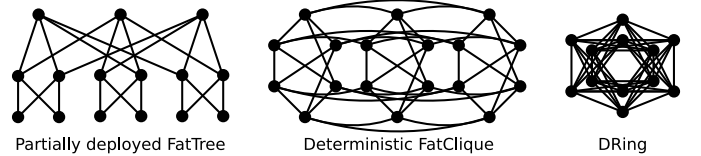


Fig. 1. Examples of highly structured topologies

the design of scalable routing algorithms that can achieve their designed capacity is an open question.

Routing inside datacenter networks can be categorized into traffic-aware routing and oblivious routing. Traffic-aware routing reduces network congestion and improve overall throughput by regularly adjusting routes and fractions of traffic demands over the routes according to queue occupancy or traffic demands [6], [7], [16]. These advantages come at a cost of specialized hardware and routing complexity. Alternatively, oblivious routing is much simpler. Traffic demands are distributed according to predefined routes and shares of the demands over each route. In particular, the routing is oblivious to the current traffic demand, so regular configuration of routes is unnecessary [15]. Because of this simplicity, oblivious routing, including ECMP and VLB, is deployed in several large-scale datacenter networks [1]–[4]. A real-world measurement at Facebook also suggests oblivious techniques work well leaving little room for improvement using advanced traffic-aware routing [17].

Although existing oblivious routing is widely deployed, it achieves designed capacity only for the folded-Clos and clique topologies. Specifically, ECMP can achieve the designed capacity of folded-Clos topologies including FatTree [1], Google’s Jupiter [3] and Facebook’s Fabric [4]. VLB achieves designed capacity for a clique topology and works reasonably well in Microsoft’s VL2 [2]. However, recent datacenter topology designs have moved away from the folded-Clos family [8]–[12] and, for these, existing oblivious routing approaches cannot be used to achieve their designed capacity.

Of these new designs, highly structured topologies, such as FatClique [11] and DRing [12] (Fig. 1), are more manageable (*i.e.*, they are easier to install and expand) than random topologies [8], [9]. For these structured topologies, this paper develops a general oblivious routing algorithm that relies on hardware support for multi-path routing (WCMP) already available in commodity switches.

For any network, oblivious routing can be formulated as a robust multi-commodity flow problem with a concave ob-

jective in which the number of constraints grows factorially with the number of switches in the network. Even for a small network, the problem size, in terms of numbers of variables and constraints, can easily overwhelm an optimization solver's memory and renders the problem intractable.

For structured topologies, however, we show that we can use the topological structure and graph automorphism to reduce the problem size to the point that is tractable for any off-the-shelf solver running on commodity hardware.

To do this, we first prove the existence of an optimal solution that is invariant to automorphism—the solution is a permutation of itself. Based on this result, we reduce the search space of an optimal solution to the solution that is automorphism-invariant. Using graph automorphism, we formulate a robust multi-commodity flow problem with significantly fewer variables and constraints to target this optimal solution. While the formulation has robust constraints, we observe that the optimal solution is in a much lower dimensional space compared to the original robust formulation, due to the automorphism-invariant property of the optimal solution. Therefore, we only need to consider a smaller subset of traffic demands instead of considering all possible demands.

This leads to the design of an iterative algorithm that alternates between convex optimization and a linear program. The convex optimization finds an automorphism-invariant routing solution. The linear program generates adversarial traffic demands to make sure the end result of the iterative algorithm is optimal and satisfies all possible traffic demands. Both optimization problems are based on representative variables and constraints, which significantly reduce the problem sizes. However, the construction of representative variables and constraints is a combinatorial problem, associated with the exponentially large number of automorphisms. We design polynomial-time algorithms to construct these representatives by utilizing the generators of the automorphisms.

We evaluate our iterative algorithm in terms of throughput, scalability, and generality over three potential applications. i) The algorithm improves the throughput up to 87.5% over a heuristic algorithm for partially deployed FatTree topologies. It also yields up to  $2.55\times$  throughput gain for DRing topologies. ii) The algorithm is scalable and provides the optimal oblivious routing solution for FatClique, DRing, and partially deployed FatTree topologies with a thousand switches. iii) We demonstrate the generality of the algorithm by considering a structured topology with non-uniform link capacity and server distribution.

The contributions of this work are threefold, which leads to a more general oblivious routing formulation supporting traditional linear objective functions and fairness-aware functions.

- We prove the existence of an automorphism-invariant optimal solution of an oblivious routing problem with a concave objective in every structured topology. This reduces the search space of optimal solutions.
- We design the iterative algorithm that targets the automorphism-invariant optimal solution using graph automorphism. The algorithm is tractable in comparison to solving the intractable oblivious routing formulation.

- We develop the polynomial-time construction of the algorithm and illustrate three applications of the algorithm.

The paper is organized as follows. Section II presents related works. Section III models a datacenter network and formulates the oblivious routing problem. Section IV proves the existence of the optimal solution that is automorphism invariant. Section V uses this insight to develop the iterative algorithm targeting the automorphism-invariant optimal solution. Section VI provides an efficient construction of representative variables and constraints. Our approach is evaluated in Section VII, and Section VIII concludes our work.

## II. RELATED WORK

Oblivious routing has long been studied in two perspectives: algorithmic construction with performance bounds [18]–[25] and mathematical optimization [20]–[23], [26]–[29].

**Oblivious routing with competitive ratio:** In 1980s, Valiant *et al.* [18], [19] studied the competitive ratio, which is the maximal ratio of the congestion resulting from an oblivious routing algorithm to the optimal congestion for any traffic matrix, and proposed an  $O(\log n)$ -competitive ratio oblivious routing algorithm for hypercube topology where  $n$  is the number of nodes in a network. The key idea of Valiant's routing scheme is that the traffic from a source is equally split to intermediate nodes, and every intermediate node routes the traffic to its destination. Applegate *et al.* [27], [28] proved the  $\Omega(\sqrt{n})$  lower-bound of competitive ratio for a directed graph.

Alternatively, an oblivious routing scheme can be efficiently constructed from decomposition trees from hierarchical decomposition. In [24], Räcke proposed the oblivious routing scheme for undirected graphs with competitive ratio  $O(\log^3 n)$  using a single decomposition tree. Later in [25], Räcke utilized the convex combination of decomposition trees [30] to obtain an  $O(\log n)$ -competitive ratio routing scheme. Nonetheless, the aforementioned methods focus on competitive ratios instead of optimal oblivious routing schemes.

**Optimal oblivious routing:** Azar *et al.* [26] proposed a polynomial-time linear program with a separate oracle for optimal oblivious routing solution. Later, Applegate *et al.* [27], [28] proposed an efficient single polynomial-size linear program by combining the dual formulation of the oracle with the main linear program. Kodialam *et al.* [20]–[23] focused on the hose traffic model and independently proposed a single polynomial-size linear program. They also proposed a generalized two-phase routing scheme where the traffic can be split to intermediate nodes with predetermined split ratios. The scheme is proven to achieve at least half of the optimal throughput in some cases. For a clique topology, Zhang-Shen *et al.* [14], [29] utilized the unique property of employing two-phase routing on a clique network to obtain the optimal link capacity allocation for robust backbone networks. However, all methods mentioned above are either intractable for large-scale topologies or too specific to some topologies. Instead, our work proposes a new method for highly structured large-scale topologies and supports concave objective functions, which have not been considered in all previous works.

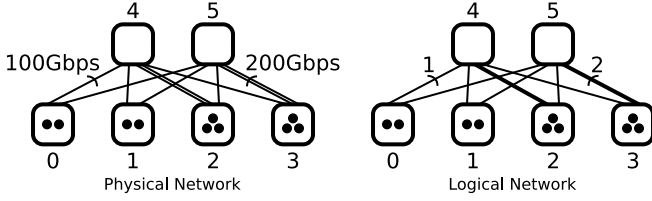


Fig. 2. Aggregated link capacity in the physical network is normalized to the number of physical links in the logical network. Switches are labeled by 0 to 5. Switches 0 to 3 are connected to 2, 2, 3, 3 servers respectively.

### III. SYSTEM MODEL

This section formally models a datacenter network, traffic demands, and the oblivious routing formulation. These models are general and should fit most practical datacenter networks.

#### A. Datacenter network model

A datacenter network is an interconnection of servers and ethernet switches. Each server has a single full-duplex port for bi-directional communication and is connected to a switch<sup>1</sup>. Each network switch has a finite number of full-duplex ports for interconnection with servers and other switches. For example, Broadcom's Tomahawk 4 Ethernet switch chip can be configured as 256 ports at 100Gbps [33]. Two devices are physically connected by connecting ports from both ends. In practice, there could be multiple physical links between two devices to increase communication capacity. Our model considers a logical link between any two devices, and the logical capacity equals the combined capacity of all physical links between the devices. Since the capacity of physical links are identical, we normalize the logical link capacity by the physical link capacity<sup>2</sup>, and we only consider the number of physical links between devices. This model based on logical links is illustrated in Fig. 2.

A datacenter network is a directed graph with the set of switches  $\mathcal{S}$  and the set of logical links  $\mathcal{L}$ . Every logical link connects two switches. The graph is assumed to be connected. A directed link  $(i, j)$  connects switch  $i$  to switch  $j$  with capacity  $C_{ij}$  equaling the number of physical links between the switches. Because of the full-duplex ports, link  $(i, j) \in \mathcal{L}$  if and only if  $(j, i) \in \mathcal{L}$ , and both links have identical capacity, i.e.,  $C_{ij} = C_{ji}$  for every  $(i, j) \in \mathcal{L}$ . To account for servers, let  $H_s$  be the number of servers connected to switch  $s$  for every  $s \in \mathcal{S}$ . Then, we define  $\mathcal{S}_k$  as the set of switches with  $k$  servers attached,  $\mathcal{S}_k = \{s \in \mathcal{S} : H_s = k\}$  for every non-negative integer  $k$ . The set  $\mathcal{S}_0$  contains all switches with no servers attached. We further define a set of switches with server(s) attached as  $\mathcal{H} = \mathcal{S} \setminus \mathcal{S}_0$ .

We adopt the multi-commodity model to directly measure throughput between every pair of source and destination switches with server(s) attached. Denote the set of commodities by  $\mathcal{C} = \{(u, v) \in \mathcal{H}^2 : u \neq v\}$ . This set is also used for traffic demand modeling.

<sup>1</sup>When a server has multiple ports and routing capability, as in server-centric topologies [31], [32], such a server can be viewed as an ethernet switch attached with multiple servers.

<sup>2</sup>When the capacities of physical links are not identical, the normalization uses the greatest common divisor.

#### B. Traffic model

Traffic inside a datacenter network is a combination of demands generated by servers attached to different switches. Since a commodity is defined at the switch level, the demand from a switch is the aggregate demand from its attached servers to servers on other switches. This demand is limited by the capacity of the server-facing links, the logical links between a switch and its attached servers. From the normalization, every switch can source and receive traffic demands at most the number of its attached servers. We denote the traffic demand of commodity  $(u, v)$  by  $t^{uv}$  for every  $(u, v) \in \mathcal{C}$ . A combination of traffic demands from every commodity forms a traffic matrix  $[t^{uv}] \in \mathbb{R}_+^{|\mathcal{H}|^2}$  where  $\mathbb{R}_+$  is a set of non-negative reals. The set of all possible traffic matrices is denoted by  $\mathcal{T}$ , which is also called the hose traffic model [23], [34], where

$$\mathcal{T} = \left\{ [t^{uv}] \in \mathbb{R}_+^{|\mathcal{H}|^2} : \begin{array}{ll} \sum_{v \in \mathcal{H}} t^{uv} \leq H_u & \forall u \in \mathcal{H} \\ \sum_{u \in \mathcal{H}} t^{uv} \leq H_v & \forall v \in \mathcal{H} \\ t^{uu} = 0 & \forall u \in \mathcal{H} \end{array} \right\}.$$

The first and second constraints ensure every switch can source and receive traffic at most the total capacity of all server-facing links. The last constraint ensures that a switch internally forwards traffic between servers connected to it.

This traffic set is used to design optimal oblivious routes between every commodity with predictable throughput performance. For example, if all routes can deliver every traffic matrix in  $\mathcal{T}$ , no congestion from capacity violation will occur.

#### C. Oblivious routing formulation

Oblivious routing distributes traffic demand of every commodity over links in the network. We assume traffic can be split arbitrarily at any intermediate switch similar to [14], [15]. Every splitting proportion is independent of the traffic demand. For example, with the 1 : 2 split proportion, 1 unit of traffic demand is split to 1/3 and 2/3, and 2 units are split to 2/3 and 4/3. Hence, the routing is oblivious to traffic demands. Next, we formulate an oblivious routing optimization problem.

Recall that we adopt the multi-commodity model with the commodity set  $\mathcal{C}$  where a commodity is a pair of source and destination switches. For each commodity  $(u, v)$ , we decouple traffic demand  $t^{uv}$  from routing and splitting by defining  $f_{ij}^{uv}$  as a share of the demand over a link  $(i, j)$  such that the actual traffic over the link is  $t^{uv} f_{ij}^{uv}$ .

Since, in general, a datacenter's traffic demand can be any traffic matrix in  $\mathcal{T}$ , the throughput of commodity  $(u, v)$  is defined by a non-negative factor  $\theta^{uv}$  that scales every traffic demand of the commodity in the traffic set. For example, the worst-case throughput, described in [10], [15], [35], is the worst scaling over all commodities,  $\min_{(u,v) \in \mathcal{C}} \theta^{uv}$ , such that the overall worst-case throughput equals 1/2 means any traffic matrix  $T \in \mathcal{T}/2$  does not violate any link capacity.

For generality, our formulation maximizes a concave function  $g : \mathbb{R}_+^{|\mathcal{C}|} \rightarrow \mathbb{R}$  of the throughput variables  $\{\theta^{uv}\}$ . This general concave function includes the worst-case throughput above, the worst-case throughput with marginal throughput [36]

$$A \min_{(u,v) \in \mathcal{C}} \theta^{uv} + \sum_{(u,v) \in \mathcal{C}} \theta^{uv}, \quad (1)$$

the proportional fair function [37]

$$\sum_{(u,v) \in \mathcal{C}} \log(\theta^{uv}), \quad (2)$$

and other functions such as the average delay [38] and the  $\alpha$ -fairness function [39]. We thus formulate the oblivious routing problem with a concave objective as follows:

$$\begin{aligned} & \text{Maximize} \quad g(\{\theta^{uv}\}) \\ & \text{Subject to} \quad \sum_{j \in \mathcal{O}(i)} f_{ij}^{uv} - \sum_{j \in \mathcal{I}(i)} f_{ji}^{uv} = \\ & \quad \theta^{uv} \{ \mathbb{I}[i = u] - \mathbb{I}[i = v] \}, \quad \forall i \in \mathcal{S}, \forall (u, v) \in \mathcal{C} \\ & \quad \sum_{(u,v) \in \mathcal{C}} t^{uv} f_{ij}^{uv} \leq C_{ij}, \quad \forall (i, j) \in \mathcal{L}, \forall [t^{uv}] \in \mathcal{T} \\ & \quad f_{ij}^{uv}, \theta^{uv} \in \mathbb{R}_+, \quad \forall (u, v) \in \mathcal{C}, \forall (i, j) \in \mathcal{L}, \end{aligned} \quad (3)$$

where  $\mathcal{O}(i)$  and  $\mathcal{I}(i)$  are respectively the sets of switches that switch  $i$  has out-going links to and in-coming links from<sup>3</sup>. The first constraint is a conservation of share at every switch. The second constraint is a robust link capacity that considers all possible traffic matrices.

Although the formulation can be transformed to a linear program with the extreme points of the traffic set (similar to the technique used in [34]), the sheer numbers of commodities  $O(|\mathcal{H}|^2)$ , directed links  $O(|\mathcal{L}|)$ , and the extreme points  $O(|\mathcal{H}|!)$  at the scale of datacenter networks can easily overwhelm the available memory of any off-the-shelf solver and its ability to obtain optimal solutions. In particular, the numbers of variables and constraints are respectively  $O(|\mathcal{H}|^2|\mathcal{L}|)$  and  $O(|\mathcal{H}|^2|\mathcal{S}| + |\mathcal{L}||\mathcal{H}|!)$ . Another formulation in [27] can reduce these numbers to polynomial; even so, the formulation can still overwhelm available memory easily as shown in our evaluations. In the next section, we leverage topological structure in some datacenter networks to achieve tractability.

#### IV. CHARACTERIZATION OF OPTIMAL SOLUTIONS

This section first introduces graph automorphisms that identify “similar” structure in datacenter networks together with the objective function in (3). We then prove the existence of an optimal oblivious routing solution that also has a “similar” structure.

##### A. Graph automorphism

Graph automorphism is a permutation of nodes in the graph such that the adjacency between nodes before and after the permutation is the same. For a datacenter network, we extend the adjacency preservation of graph automorphism to include the preservation of servers at a switch, link capacity, and the objective function as illustrated in Fig. 3 and defined formally below.

**Definition 1** (Automorphism). *Given an objective function  $g$  and a network topology with switch set  $\mathcal{S}$ , link set  $\mathcal{L}$ , link*

<sup>3</sup>We have  $\mathcal{O}(i) = \mathcal{I}(i)$  under the full-duplex assumption.

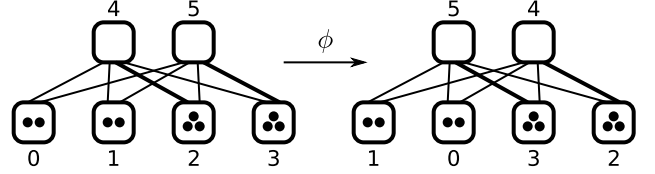


Fig. 3. Suppose the objective function is  $\sum_{(u,v) \in \mathcal{C}} \log \theta^{uv}$ . The right network is an automorphism of the left network with  $\phi(x) = x + 1$  when  $x$  is even and  $\phi(x) = x - 1$  when  $x$  is odd. Both networks have the same adjacency and distributions of link capacities and servers.

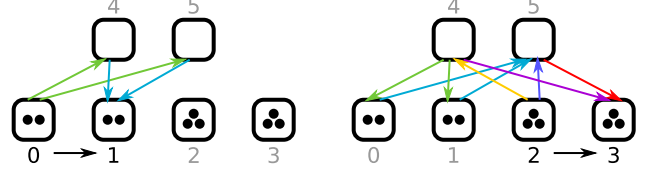


Fig. 4. The share variables of commodities  $(0,1)$  and  $(2,3)$  in an automorphism-invariant optimal solution are shown. Each arrow represents a share variable. For each commodity, the shares having the same value are assigned an identical color. They are automorphism invariant, e.g.,  $f_{04}^{01*} = f_{05}^{01*}$  under the automorphism that permutes switches 4 and 5. For commodity  $(2,3)$ , we have  $f_{40}^{23*} = f_{41}^{23*}$  under the automorphism that permutes switches 0 and 1. For throughput variables, we have  $\theta^{01*} = \theta^{10*}$  and  $\theta^{23*} = \theta^{32*}$  under the automorphism that permutes switch pairs  $(0,1)$  and  $(2,3)$ .

capacities  $\{C_{ij}\}$ , and the numbers of servers at every switch  $\{H_u\}$ , an automorphism  $\phi : \mathcal{S} \rightarrow \mathcal{S}$  preserves:

- 1) *Adjacency*:  $(\phi(i), \phi(j)) \in \mathcal{L}$  for every  $(i, j) \in \mathcal{L}$ .
- 2) *Link capacity*:  $C_{ij} = C_{\phi(i)\phi(j)}$  for every  $(i, j) \in \mathcal{L}$ .
- 3) *Number of Servers*:  $H_u = H_{\phi(u)}$  for every  $u \in \mathcal{S}$ .
- 4) *Objective function*:  $g(\{\theta^{uv}\}) = g(\{\theta^{\phi(u)\phi(v)}\})$ .

Note that the invariant of an objective function in Definition 1 means the objective function is indifferent when commodities are permuted under an automorphism; for example,  $g(\{\theta^{uv}\}) = \sum_{(u,v) \in \mathcal{C}} \log \theta^{uv} = \sum_{(u,v) \in \mathcal{C}} \log \theta^{\phi(u)\phi(v)} = g(\{\theta^{\phi(u)\phi(v)}\})$ .

The set of all automorphisms is denoted by  $\Phi$ . While the size of this set could grow exponentially large, it is finite, i.e.,  $|\Phi| < \infty$ . Next, we use these automorphisms to reduce the search space for an optimal solution of the formulation in (3).

##### B. Existence of an automorphism-invariant optimal solution

We will show the existence of an automorphism-invariant optimal solution of the formulation in (3). In an automorphism-invariant optimal solution, every decision variable in a group under automorphisms in  $\Phi$  takes the same value as shown in Fig. 4.

We first show that applying an automorphism to an optimal solution results in another optimal solution.

**Lemma 1.** *Suppose  $\{f_{ij}^{uv*}, \theta^{uv*}\}$  is an optimal solution of the formulation in (3). Given any automorphism  $\phi \in \Phi$ , the solution  $\{f_{ij}^{uv}, \theta^{uv}\}$  is also an optimal solution such that  $f_{ij}^{uv} = f_{\phi(i)\phi(j)}^{\phi(u)\phi(v)*}$  and  $\theta^{uv} = \theta^{\phi(u)\phi(v)*}$ .*

*Proof.* To show that the solution from an automorphism is also an optimal solution, we show that the solution leads to the optimal objective value and the feasibility of all constraints.

Proving the objective value is optimal is straightforward from the invariant of an objective function in Definition 1 as

$$g(\{\theta^{uv}\}) = g(\{\theta^{\phi(u)\phi(v)*}\}) = g(\{\theta^{uv*}\}).$$

We then consider the feasibility of constraints.

From a share conservation constraint at switch  $i$  and commodity  $(u, v)$  in (3), the difference between the out-going shares and the in-coming shares is

$$\begin{aligned} \Delta(u, v, i) &= \sum_{j \in \mathcal{O}(i)} f_{\phi(i)\phi(j)}^{\phi(u)\phi(v)*} - \sum_{j \in \mathcal{I}(i)} f_{\phi(j)\phi(i)}^{\phi(u)\phi(v)*} \\ &= \sum_{j \in \mathcal{O}(\phi(i))} f_{\phi(i)j}^{\phi(u)\phi(v)*} - \sum_{j \in \mathcal{I}(\phi(i))} f_{j\phi(i)}^{\phi(u)\phi(v)*}. \end{aligned} \quad (4)$$

This is because the adjacency preservation of the automorphism in Definition 1. Since the share conservation holds true for the optimal solution, the difference in (4) becomes:

$$\begin{aligned} \Delta(u, v, i) &= \theta^{\phi(u)\phi(v)*} \{\mathbb{I}[\phi(i) = \phi(u)] - \mathbb{I}[\phi(i) = \phi(v)]\} \\ &= \theta^{uv} \{\mathbb{I}[i = u] - \mathbb{I}[i = v]\}. \end{aligned}$$

This shows that the share conservation constraint at switch  $i$  and commodity  $(u, v)$  under the solution from an automorphism is feasible. This holds true for every switch  $i \in \mathcal{S}$  and every commodity  $(u, v) \in \mathcal{C}$ .

For the link capacity constraint at link  $(i, j)$  and traffic matrix  $[t^{uv}]$  in (3), the left-hand side under the solution from an automorphism equals

$$\begin{aligned} \sum_{(u,v) \in \mathcal{C}} t^{uv} f_{\phi(i)\phi(j)}^{\phi(u)\phi(v)*} &= \sum_{(u,v) \in \mathcal{C}} t^{\phi^{-1}(u)\phi^{-1}(v)} f_{\phi(i)\phi(j)}^{uv*} \\ &\leq C_{\phi(i)\phi(j)} = C_{ij}. \end{aligned}$$

The first equality follows from the re-indexing of terms in the summation over the commodity set. The inequality uses the fact that the capacity constraint of link  $(\phi(i), \phi(j))$  under the optimal solution is feasible when the traffic matrix is  $[t^{\phi^{-1}(u)\phi^{-1}(v)}]$ , which is a member of the traffic set  $\mathcal{T}$ . The last equality follows the capacity preservation in Definition 1. Therefore, the link capacity constraint at link  $(i, j)$  and traffic matrix  $T = [t^{uv}]$  under the solution from an automorphism is feasible. Again, this holds true for every link  $(i, j) \in \mathcal{L}$  and every traffic matrix  $[t^{uv}] \in \mathcal{T}$ .

In short, we have shown that the solution from an automorphism yields the optimal objective value and leads to feasibility of all constraints. Thus, the solution is optimal.  $\square$

Lemma 1 implies that the automorphism of an optimal solution is another optimal solution. Next, we show that there exists an optimal solution that is invariant to automorphism.

**Theorem 1.** *There exists an automorphism-invariant optimal solution  $\{\hat{f}_{ij}^{uv}, \hat{\theta}^{uv}\}$  to the formulation in (3) such that  $\hat{f}_{ij}^{uv} = \hat{f}_{\phi(i)\phi(j)}^{\phi(u)\phi(v)*}$  and  $\hat{\theta}^{uv} = \hat{\theta}^{\phi(u)\phi(v)*}$  for every  $(u, v) \in \mathcal{C}$ , every  $(i, j) \in \mathcal{L}$ , and every  $\phi \in \Phi$ .*

*Proof.* The formulation in (3) always has an optimal solution since the overall capacity from all links is limited. Let  $\{f_{ij}^{uv*}, \theta^{uv*}\}$  be such an optimal solution. From Lemma 1, the solution  $\{f_{\phi(i)\phi(j)}^{\phi(u)\phi(v)*}, \theta^{\phi(u)\phi(v)*}\}$  is an optimal solution for

every automorphism  $\phi \in \Phi$ . We construct the automorphism-invariant solution  $\{\hat{f}_{ij}^{uv}, \hat{\theta}^{uv}\}$  as follows:

$$\begin{aligned} \hat{f}_{ij}^{uv} &= \frac{1}{|\Phi|} \sum_{\phi \in \Phi} f_{\phi(i)\phi(j)}^{\phi(u)\phi(v)*}, \quad \forall (u, v) \in \mathcal{C}, \forall (i, j) \in \mathcal{L} \\ \hat{\theta}^{uv} &= \frac{1}{|\Phi|} \sum_{\phi \in \Phi} \theta^{\phi(u)\phi(v)*}, \quad \forall (u, v) \in \mathcal{C} \end{aligned}$$

We then show that this solution is automorphism-invariant. Specifically, it holds for any  $\phi' \in \Phi$  that

$$\begin{aligned} \hat{f}_{\phi'(i)\phi'(j)}^{\phi'(u)\phi'(v)*} &= \frac{1}{|\Phi|} \sum_{\phi \in \Phi} f_{\phi(\phi'(i))\phi(\phi'(j))}^{\phi(\phi'(u))\phi(\phi'(v))*} \\ &= \frac{1}{|\Phi|} \sum_{\phi \in \Phi} f_{\phi(i)\phi(j)}^{\phi(u)\phi(v)*} = \hat{f}_{ij}^{uv}. \end{aligned}$$

The second equality holds because the automorphism set  $\Phi$  is a group, so i) the composition of two automorphism mapping functions gives an automorphism in the group and ii) such composition over  $\Phi$  yields the same  $\Phi$ . Similarly, the same holds true for  $\hat{\theta}^{uv*}$ .

$$\begin{aligned} \hat{\theta}^{\phi'(u)\phi'(v)*} &= \frac{1}{|\Phi|} \sum_{\phi \in \Phi} \theta^{\phi(\phi'(u))\phi(\phi'(v))*} \\ &= \frac{1}{|\Phi|} \sum_{\phi \in \Phi} \theta^{\phi(u)\phi(v)*} = \hat{\theta}^{uv}. \end{aligned}$$

Therefore, the solution is automorphism-invariant. Next, we show that it is feasible and optimal. The feasibility can be easily shown from the facts that the solution is a convex combination of optimal solutions and all constraints are affine. For the optimality, let  $g^*$  be the optimal objective value of the problem in (3). We have from the concavity of an objective function and Jensen's inequality [40] that

$$\begin{aligned} g(\{\hat{\theta}^{uv}\}) &= g\left(\frac{1}{|\Phi|} \sum_{\phi \in \Phi} \{\theta^{\phi(u)\phi(v)*}\}\right) \\ &\geq \frac{1}{|\Phi|} \sum_{\phi \in \Phi} g(\{\theta^{\phi(u)\phi(v)*}\}) = g^*, \end{aligned}$$

where the last equality uses the fact that all terms in the summation equal  $g^*$ . Since, the solution  $\{\hat{\theta}^{uv}\}$  is feasible, it must be that  $g(\{\hat{\theta}^{uv}\}) = g^*$ , and the solution is optimal.  $\square$

Theorem 1 implies that there is an optimal solution whose variables in the same group under automorphisms take the same value. We use this insight in the next section to reduce the number of decision variables and constraints of the oblivious routing formulation.

## V. FINDING AN AUTOMORPHISM-INVARIANT OPTIMAL SOLUTION

We now formulate an optimization problem that targets an automorphism-invariant solution. The formulation defines a new set of representative variables representing groups of variables under the automorphisms. Furthermore, unnecessary constraints and traffic matrices are removed from the formulation, resulting in a tractable iterative algorithm.

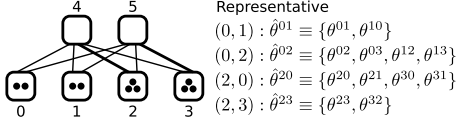


Fig. 5. Throughput variables form four groups. Each group is represented by a representative throughput variable and a corresponding commodity. The set of representative commodities is  $\hat{\mathcal{C}} = \{(0, 1), (0, 2), (2, 0), (2, 3)\}$ . Commodity  $(0, 1)$  has a representative throughput  $\hat{\theta}^{01}$  representing variables  $\theta^{01}, \theta^{10}$ .

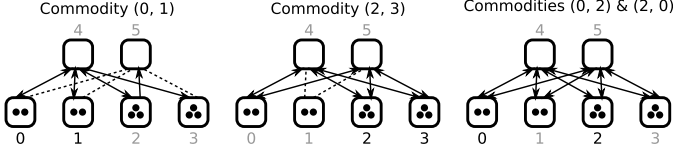


Fig. 6. Representative share variables of each representative commodity. An arrow represents a representative share variable in the direction of the arrow. For example, with commodity  $(0, 1)$ , the link  $(0, 4)$  has two representative variables  $\hat{f}_{04}^{01}, \hat{f}_{40}^{01}$ , and the link  $(2, 5)$  has only one representative variable  $\hat{f}_{25}^{01}$ . Representative share variables for each representative commodity depend on the network structure and the commodity.

#### A. Representative variables

From Theorem 1, we observe that the optimal values of throughput variables form groups under automorphisms. In particular, the values are identical for every commodity in the same group, whose members can be mapped to one another by some automorphisms, i.e., a group of commodities containing commodity  $(u, v)$  is  $\{(\phi(u), \phi(v)) : \forall \phi \in \Phi\}$ . Therefore, we can pick a commodity in each group as a representative commodity for the group. We denote the set of all representative commodities by  $\hat{\mathcal{C}}$  and define a representative throughput variable  $\hat{\theta}^{uv}$  for every representative commodity  $(u, v) \in \hat{\mathcal{C}}$ . Fig. 5 shows an example of representative commodities and throughput variables. Further, we define a function  $\pi : \mathcal{C} \rightarrow \Phi$  such that the input commodity under the output automorphism is the representative commodity of the input, i.e.,  $\phi = \pi(u, v)$  and  $(\phi(u), \phi(v)) \in \hat{\mathcal{C}}$ . Section VI-B describes an efficient algorithm to construct  $\hat{\mathcal{C}}$  and  $\pi$ .

The representative shares can be defined using the same process. For each representative commodity  $(u, v) \in \hat{\mathcal{C}}$ , we observe from Theorem 1 that the optimal values of share variables form groups under automorphisms. In particular, the values are identical for every link in the same group, whose members can be mapped to one another by some automorphisms that does not affect  $(u, v)$ , i.e., the group of links containing link  $(i, j)$  is  $\{(\phi(i), \phi(j)) : (\phi(u), \phi(v)) = (u, v) \exists \phi \in \Phi\}$ . Therefore, we can pick a link in each group as a representative link for the group. We denote the set of all representative links for a representative commodity  $(u, v)$  by  $\hat{\mathcal{L}}^{uv}$  and define a representative share variable  $\hat{f}_{ij}^{uv}$  for every  $(u, v) \in \hat{\mathcal{C}}$  and every  $(i, j) \in \hat{\mathcal{L}}^{uv}$ . Fig. 6 shows representative share variables for a simple network. Further, we define a mapping function  $\sigma^{uv} : \mathcal{L} \rightarrow \Phi$  such that the input link under the output automorphism is the representative link of the input, i.e.,  $\phi = \sigma^{uv}(i, j)$  and  $(\phi(i), \phi(j)) \in \hat{\mathcal{L}}^{uv}$ . Section VI-C describes an efficient algorithm to construct  $\hat{\mathcal{L}}^{uv}$  and  $\sigma^{uv}$ .

Since the representative variables of throughput and share

have been defined, the last step is to map every variable in (3) to these representatives. We define the mapping of throughput and share variables in (3) to their representatives as follows:

$$\begin{aligned} \varphi[\theta^{uv}] &= \hat{\theta}^{\phi(u), \phi(v)} \quad \forall (u, v) \in \mathcal{C} \quad \text{where } \phi = \pi(u, v), \\ \varphi[f_{ij}^{uv}] &= \hat{f}_{\phi'(i), \phi'(j)}^{\phi(u), \phi(v)} \quad \forall (u, v) \in \mathcal{C}, \forall (i, j) \in \mathcal{L} \\ &\quad \text{where } \phi = \pi(u, v) \text{ and } \phi' = \sigma^{\phi(u), \phi(v)}(i, j). \end{aligned}$$

Using the above mapping and the representative variables, we formulate an optimization problem targeting an automorphism-invariant optimal solution in Theorem 1 as follows:

$$\begin{aligned} &\text{Maximize} \quad g(\{\varphi[\theta^{uv}]\}) \\ &\text{Subject to} \quad \sum_{j \in \mathcal{O}(i)} \varphi[f_{ij}^{uv}] - \sum_{j \in \mathcal{I}(i)} \varphi[f_{ji}^{uv}] = \\ &\quad \hat{\theta}^{uv} \{\mathbb{I}[i = u] - \mathbb{I}[i = v]\}, \quad \forall i \in \mathcal{S}, \forall (u, v) \in \hat{\mathcal{C}} \\ &\quad \sum_{(u, v) \in \mathcal{C}} t^{uv} \varphi[f_{ij}^{uv}] \leq C_{ij}, \quad \forall (i, j) \in \mathcal{L}, \forall [t^{uv}] \in \mathcal{T} \\ &\quad \hat{f}_{ij}^{uv}, \hat{\theta}^{uv} \in \mathbb{R}_+, \quad \forall (u, v) \in \hat{\mathcal{C}}, \forall (i, j) \in \hat{\mathcal{L}}^{uv}. \end{aligned} \quad (5)$$

The total numbers of variables and constraints in (5) are significantly reduced by considering the representative variables. Notice that the share conservation constraints are defined over representative commodities  $\hat{\mathcal{C}}$  instead of  $\mathcal{C}$ . The next sections address the challenging robust link capacity constraint.

#### B. Removing unnecessary link constraints

The number of link capacity constraints in (5) grows like  $O(|\mathcal{L}||\mathcal{H}|!)$ . However, many constraints are unnecessary and can be removed. We observe that traffic demands on link  $(i', j')$  can be aggregated for each representative share.

$$\begin{aligned} \sum_{(u, v) \in \mathcal{C}} t^{uv} \varphi[f_{i'j'}^{uv}] &= \sum_{(u, v) \in \hat{\mathcal{C}}} \sum_{(i, j) \in \hat{\mathcal{L}}^{uv}} \hat{f}_{ij}^{uv} \sum_{(a, b) \in \mathcal{C}: \varphi[f_{i'j'}^{ab}] = \hat{f}_{ij}^{uv}} t^{ab} \\ &= \sum_{(u, v) \in \hat{\mathcal{C}}} \sum_{(i, j) \in \hat{\mathcal{L}}^{uv}} \hat{f}_{ij}^{uv} \alpha_{ij}^{uv}, \end{aligned} \quad (6)$$

where  $\alpha_{ij}^{uv} = \sum_{(a, b) \in \mathcal{C}: \varphi[f_{i'j'}^{ab}] = \hat{f}_{ij}^{uv}} t^{ab}$ .

Any two links that have the same set of coefficient alphas under some automorphism are duplicate, and we only need to consider one of them as a representative constraint. Let  $\hat{\mathcal{L}}$  be the set of representative link constraints. Section VI-D provides an efficient algorithm to construct this set. Therefore, the link capacity constraint in (5) can be replaced by

$$\sum_{(u, v) \in \mathcal{C}} t^{uv} \varphi[f_{ij}^{uv}] \leq C_{ij}, \quad \forall (i, j) \in \hat{\mathcal{L}}, \forall [t^{uv}] \in \mathcal{T}. \quad (7)$$

Notice that the reduction from  $\mathcal{L}$  to  $\hat{\mathcal{L}}$  is by considering the representative variables and traffic demands. Ignoring them and only considering automorphisms of links can lead to an under-constrained formulation and a sub-optimal solution.

### C. Traffic matrix selection

Including the entire traffic set  $\mathcal{T}$  into the link capacity constraint is impractical due to the continuity of the set. While it is possible to consider the set of extreme points, the number of such points is  $O(|\mathcal{H}|!)$  for a simple case of double stochastic matrices when the number of servers per switch is identical.

To alleviate this issue, we construct the traffic set iteratively based on two observations. First, the worst-case traffic load on different representative link is caused by a different set of traffic matrices. Therefore, we define a set of traffic matrices  $\mathcal{T}_{ij}$  considered for representative link  $(i, j) \in \hat{\mathcal{L}}$  and formulate the optimization parameterized by these sets as follows.

$$\begin{aligned}
 & \mathbf{R}(\{\mathcal{T}_{ij}\}_{(i,j) \in \hat{\mathcal{L}}}) : \\
 & \text{Maximize} \quad g(\{\varphi[\theta^{uv}]\}) \\
 & \text{Subject to} \quad \sum_{j \in \mathcal{O}(i)} \varphi[f_{ij}^{uv}] - \sum_{j \in \mathcal{I}(i)} \varphi[f_{ji}^{uv}] = \\
 & \quad \hat{\theta}^{uv} \{\mathbb{I}[i = u] - \mathbb{I}[i = v]\}, \quad \forall i \in \mathcal{S}, \forall (u, v) \in \hat{\mathcal{C}} \\
 & \quad \sum_{(u,v) \in \mathcal{C}} t^{uv} \varphi[f_{ij}^{uv}] \leq C_{ij}, \forall (i, j) \in \hat{\mathcal{L}}, \forall [t^{uv}] \in \mathcal{T}_{ij} \\
 & \quad \hat{f}_{ij}^{uv}, \hat{\theta}^{uv} \in \mathbb{R}_+, \quad \forall (u, v) \in \hat{\mathcal{C}}, \forall (i, j) \in \hat{\mathcal{L}}^{uv}. \quad (8)
 \end{aligned}$$

While solving the above formulation with insufficient traffic matrices in the traffic sets  $\{\mathcal{T}_{ij}\}$  leads to a sub-optimal oblivious routing solution, the sub-optimal solution can still guide which traffic matrices should be included in the sets. This leads to the second observation that traffic matrices also have automorphisms, and one matrix can represent many matrices. This observation can be seen from the coefficient alphas in (6). The implication is that including only a small subset of traffic matrices in each traffic set  $\mathcal{T}_{ij}$  is sufficient to achieve an optimal oblivious routing solution. We therefore generate representative traffic matrices based on a solution of (8) by solving a simple linear program parameterized by the solution for every link  $(i', j') \in \hat{\mathcal{L}}$  as follows.

$$\begin{aligned}
 & \mathbf{T}_{i'j'} \left( \left\{ \hat{f}_{ij}^{uv} \right\}_{(i,j) \in \hat{\mathcal{L}}^{uv}} \right) : \\
 & \text{Maximize} \quad \sum_{(u,v) \in \mathcal{C}} t^{uv} \varphi[f_{i'j'}^{uv}] \\
 & \text{Subject to} \quad \sum_{v \in \mathcal{H} \setminus \{u\}} t^{uv} \leq H_u, \quad \forall u \in \mathcal{H} \\
 & \quad \sum_{u \in \mathcal{H} \setminus \{v\}} t^{uv} \leq H_v, \quad \forall v \in \mathcal{H} \\
 & \quad t^{uv} \in \mathbb{R}_+, \quad \forall (u, v) \in \mathcal{C}. \quad (9)
 \end{aligned}$$

For every representative link  $(i', j')$ , the optimization in (9) finds the worst-case traffic matrix with respect to a given set of representative shares. The obtained traffic matrix is added to  $\mathcal{T}_{ij}$ , which is considered in (8) for later iterations. This process is summarized next.

### D. Iterative algorithm

In Algorithm 1, the traffic sets are initialized with an initial traffic matrix. The algorithm iteratively finds a routing solution

---

### Algorithm 1: Optimal oblivious routing

---

```

Initialize  $\mathcal{D}_{ij} \leftarrow \{[t^{uv}]_{\text{init}}\}$  for every  $(i, j) \in \hat{\mathcal{L}}$ 
while  $\bigcup_{(i,j) \in \hat{\mathcal{L}}} \mathcal{D}_{ij} \neq \emptyset$  do
     $\mathcal{T}_{ij} \leftarrow \mathcal{T}_{ij} \cup \mathcal{D}_{ij}$  for every  $(i, j) \in \hat{\mathcal{L}}$ 
     $\left\{ \hat{f}_{ij}^{uv} \right\}_{(i,j) \in \hat{\mathcal{L}}^{uv}}, \left\{ \hat{\theta}^{uv} \right\}_{(u,v) \in \hat{\mathcal{C}}} \leftarrow \mathbf{R}(\{\mathcal{T}_{ij}\}_{(i,j) \in \hat{\mathcal{L}}})$ 
    for  $(i', j') \in \hat{\mathcal{L}}$  do
         $[t^{uv}] \leftarrow \mathbf{T}_{i'j'} \left( \left\{ \hat{f}_{ij}^{uv} \right\}_{(i,j) \in \hat{\mathcal{L}}^{uv}} \right)$ 
        if  $\sum_{(u,v) \in \mathcal{C}} t^{uv} \varphi[f_{i'j'}^{uv}] > C_{i'j'}$  then
             $\mathcal{D}_{i'j'} \leftarrow \{[t^{uv}]\}$ 
        else
             $\mathcal{D}_{i'j'} \leftarrow \emptyset$ 
return  $\left\{ \hat{f}_{ij}^{uv} \right\}_{(i,j) \in \hat{\mathcal{L}}^{uv}}, \left\{ \hat{\theta}^{uv} \right\}_{(u,v) \in \hat{\mathcal{C}}}$ 

```

---

according to these traffic sets. The routing solution is used to generate more traffic matrices, which in turn help improving the next routing solution, by including those that violate link capacity constraints into the traffic sets. This process continues until no traffic matrices violating the link capacity constraints are found.<sup>4</sup> Thus, the final routing solution is optimal because it is an automorphism-invariant solution that optimizes the objective and satisfies all conservation constraints in (8) and that no traffic matrices violate capacity under this solution. Note that Algorithm 1 is non-polynomial time due to the alternation.

The initial traffic matrix  $[t^{uv}]_{\text{init}}$  in Algorithm 1 can significantly reduce the number of iterations required to obtain the optimal solution. We set this matrix to the solution of the weighted bipartite matching problem, where the node sets are  $\mathcal{U} = \mathcal{V} = \mathcal{H}$ , and the weight between any two nodes  $(u, v) \in \mathcal{U} \times \mathcal{V}$  equals the product of their minimum path length in the network topology times their maximum traffic load  $\min(H_u, H_v)$ . This initialization is inspired by the works in [35], [42], [43], in which all switches have the same number of servers.

## VI. EFFICIENT IMPLEMENTATION

Algorithm 1 and its sub-routines in the previous section rely heavily on representative sets  $\hat{\mathcal{C}}, \hat{\mathcal{L}}^{uv}, \hat{\mathcal{L}}$ , and the mapping functions  $\pi$  and  $\sigma^{uv}$ . Constructing them from the automorphism set  $\Phi$ , whose size grows exponentially, is a combinatorial problem. This section describes polynomial-time algorithms for efficient construction, based on automorphism generators.

### A. Generators of automorphisms

Generators are a smaller set of automorphisms that can generate a whole set of automorphisms. Specifically, every automorphism in the set is a combination of the generators. This set can be obtained from off-the-shelf software, such as *nauty* [44], for an undirected graph with vertices, edges, and a set of colors assigned to the vertices.

<sup>4</sup>This process is similar to constraint generation techniques [41].

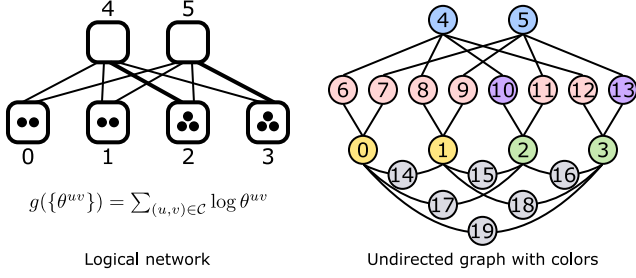


Fig. 7. A logical network is transformed to an undirected graph with colors.

The objective function in (5) and our datacenter network model with link capacities and numbers of servers at each switch can be transformed to an undirected graph with colors as shown in Fig. 7. Every switch is translated to a vertex in the graph. To preserve the number of servers at each switch under automorphism, an identical color is assigned to the vertices whose number of servers is the same, and different colors are assigned to vertices having different numbers of servers. For links and capacities, each link is transformed into an auxiliary vertex with two edges. One end of both edges is connected to the auxiliary vertex, and the other ends connect to the two switches to which the original link is adjacent. To preserve link capacity under automorphism, the auxiliary vertices of links with the same capacity are assigned the same color, and auxiliary vertices of links with different capacities are assigned different colors. To preserve the invariant of the objective function, a vertex and edges are added between the source and destination of each commodity. The vertices associated with commodities whose permutation does not change the objective function are assigned an identical color. Otherwise, the vertices are assigned different colors if their associated commodities change the objective function under some permutation.

The above transformation yields an undirected graph with a set of colors as an input to the off-the-shelf software that outputs the generators. We denote the set of these generators by  $\hat{\Phi}$ , whose generated automorphisms satisfy Definition 1. It is used to efficiently generate representative sets and mapping functions used extensively in the previous section.

### B. Representative commodity

The representative commodity set  $\hat{\mathcal{C}}$  and the mapping functions  $\pi$ , used for the formulation in (5), are constructed by Algorithm 2.

Algorithm 2 searches over the commodity set. It picks a commodity, assigns it as a representative commodity, and finds all represented commodities. The search process utilizes the generator set  $\hat{\Phi}$ , instead of the exponentially large automorphism set  $\Phi$ , to find represented commodities. The algorithm visits each commodity once, so the complexity is  $O(|\mathcal{C}| |\hat{\Phi}|)$ . Once all represented commodities of the picked representative are found, the algorithm picks an unpicked commodity and continues the process. In the process, the dictionary  $D$  keeps track of automorphisms that map representatives to their represented commodities. The last step inverts each automorphism

---

### Algorithm 2: Representative commodity construction

---

```

Initialize empty sets  $\mathcal{Q}, \mathcal{P}, \hat{\mathcal{C}}$  and dictionaries  $D, \pi$ 
for  $(u, v) \in \mathcal{C}$  do
  if  $(u, v) \in \mathcal{P}$  then
     $\perp$  continue
   $\hat{\mathcal{C}} \leftarrow \hat{\mathcal{C}} \cup \{(u, v)\}$ 
   $D(u, v) \leftarrow \phi_{\text{identity}}$ 
   $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(u, v)\}$ 
  while  $\mathcal{Q}$  is not empty do
    Pop  $(a, b)$  from  $\mathcal{Q}$ 
     $\mathcal{P} \leftarrow \mathcal{P} \cup \{(a, b)\}$ 
    for  $\phi \in \hat{\Phi}$  do
      if  $(\phi(a), \phi(b)) \notin \mathcal{P}$  then
         $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\phi(a), \phi(b))\}$ 
         $D(\phi(a), \phi(b)) \leftarrow \phi(D(a, b))$ 
for  $(u, v) \in \mathcal{C}$  do
   $\perp \pi(u, v) \leftarrow (D(u, v))^{-1}$ 
return Representative set  $\hat{\mathcal{C}}$  and function  $\pi$ 

```

---

in  $D$  and constructs the dictionary  $\pi$  storing automorphisms that map represented commodities to their representatives.

### C. Representative share

For each representative commodity  $(u, v)$ , the set of representative shares  $\hat{\mathcal{L}}^{uv}$  and the mapping functions  $\sigma^{uv}$ , used for the formulation in (5), are constructed by Algorithm 3.

---

### Algorithm 3: Representative share construction

---

```

Initialize empty sets  $\mathcal{Q}, \mathcal{P}, \hat{\mathcal{L}}^{uv}$  and dictionary  $D, \sigma^{uv}$ 
 $\hat{\Phi}^{uv} \leftarrow \{\phi \in \hat{\Phi} : (u, v) = (\phi(u), \phi(v))\}$ 
for  $(i, j) \in \mathcal{L}$  do
  if  $(i, j) \in \mathcal{P}$  then
     $\perp$  continue
   $\hat{\mathcal{L}}^{uv} \leftarrow \hat{\mathcal{L}}^{uv} \cup \{(i, j)\}$ 
   $D(i, j) \leftarrow \phi_{\text{identity}}$ 
   $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(i, j)\}$ 
  while  $\mathcal{Q}$  is not empty do
    Pop  $(a, b)$  from  $\mathcal{Q}$ 
     $\mathcal{P} \leftarrow \mathcal{P} \cup \{(a, b)\}$ 
    for  $\phi \in \hat{\Phi}^{uv}$  do
      if  $(\phi(a), \phi(b)) \notin \mathcal{P}$  then
         $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\phi(a), \phi(b))\}$ 
         $D(\phi(a), \phi(b)) \leftarrow \phi(D(a, b))$ 
for  $(i, j) \in \mathcal{L}$  do
   $\perp \sigma^{uv}(i, j) \leftarrow (D(i, j))^{-1}$ 
return Representative set  $\hat{\mathcal{L}}^{uv}$  and function  $\sigma^{uv}$ 

```

---

Given a representative commodity  $(u, v)$ , the algorithm first constructs a commodity-preserved generator set  $\hat{\Phi}^{uv}$  containing all generators that have no effect on the given commodity. The algorithm then searches over the links set

using these commodity-preserved generators. It picks a link as a representative link and finds all corresponding represented links. This process utilizes the generator set  $\hat{\Phi}^{uv}$  to search for represented links. The algorithm visits each link once, so its complexity is  $O(|\mathcal{L}||\hat{\Phi}^{uv}|)$ . Once all represented links of the picked representative are found, the algorithm picks an unpicked link and continues the process. The dictionary  $D$  keeps track of automorphisms that map representatives to their represented links. The last step inverts each automorphism in  $D$  and constructs the dictionary  $\sigma^{uv}$  storing automorphisms that map represented links to their representatives.

#### D. Representative link capacity constraint

The set of representative links  $\hat{\mathcal{L}}$  for the capacity constraints in (7) can be constructed by removing redundant constraints under automorphisms. More precisely, two constraints of links  $(x, y)$  and  $(g, h)$  are redundant when 1) their link capacities are identical,  $C_{xy} = C_{gh}$ , and 2) their traffic loads in (6) are identical under the traffic set  $\mathcal{T}$ . The former is easy to identify, but the latter is challenging as the comparison is over the whole traffic set. Theorem 2 simplifies this comparison.

**Theorem 2.** *Given links  $(x, y)$  and  $(g, h)$ , their capacity constraints in terms of representative shares are*

$$\begin{aligned} \sum_{(u,v) \in \hat{\mathcal{C}}} \sum_{(i,j) \in \hat{\mathcal{L}}^{uv}} \hat{f}_{ij}^{uv} \sum_{(a,b) \in \mathcal{C}: \varphi[f_{xy}^{ab}] = \hat{f}_{ij}^{uv}} t^{ab} &\leq C_{xy} \quad \forall [t^{uv}] \in \mathcal{T}, \\ \sum_{(u,v) \in \hat{\mathcal{C}}} \sum_{(i,j) \in \hat{\mathcal{L}}^{uv}} \hat{f}_{ij}^{uv} \sum_{(a,b) \in \mathcal{C}: \varphi[f_{gh}^{ab}] = \hat{f}_{ij}^{uv}} t^{ab} &\leq C_{gh} \quad \forall [t^{uv}] \in \mathcal{T}. \end{aligned}$$

The two constraints are identical when 1) their link capacities are identical, i.e.,  $C_{xy} = C_{gh}$ , and 2) their numbers of traffic demands associated with a representative share are identical for every representative share such that the following holds for every  $(u, v) \in \hat{\mathcal{C}}$  and every  $(i, j) \in \hat{\mathcal{L}}^{uv}$ :

$$\left| \left\{ (a, b) \in \mathcal{C} : \varphi[f_{xy}^{ab}] = \hat{f}_{ij}^{uv} \right\} \right| = \left| \left\{ (a, b) \in \mathcal{C} : \varphi[f_{gh}^{ab}] = \hat{f}_{ij}^{uv} \right\} \right|. \quad (10)$$

*Proof.* The first condition is straightforward. We prove the second condition as follows. Considering link  $(i', j')$ , we observe that every commodity in  $\{(a, b) \in \mathcal{C} : \varphi[f_{i'j'}^{ab}] = \hat{f}_{ij}^{uv}\}$  is represented by a common commodity  $(u, v)$  as they share the same representative share variable  $\hat{f}_{ij}^{uv}$ . It follows that these commodities generate an identical set of traffic demands, so the set of aggregated traffic demands associated with the representative share variable are identical if the numbers of members in both sets are the same, i.e., when (10) holds, then

$$\left\{ \sum_{(a,b) \in \mathcal{C}: \varphi[f_{xy}^{ab}] = \hat{f}_{ij}^{uv}} t^{ab} : \forall [t^{ab}] \in \mathcal{T} \right\} = \left\{ \sum_{(a,b) \in \mathcal{C}: \varphi[f_{gh}^{ab}] = \hat{f}_{ij}^{uv}} t^{ab} : \forall [t^{ab}] \in \mathcal{T} \right\}.$$

Therefore, comparing the number of traffic demands associated with each representative share is sufficient.  $\square$

The implication of Theorem 2 is that we only need to compare the number of traffic demand terms associated with each representative share to determine whether two constraints are identical under automorphism. This leads to a simpler construction of the representative capacity constraints.

The set of representative capacity constraints is constructed by computing the distribution of the numbers of traffic demand terms associated with representative shares. Then, for every group of links having the same distribution, we take one link from the group and use it as the representative of the group.

## VII. EVALUATION

The performance, scalability, and generality of Algorithm 1 are evaluated over three potential applications. The algorithm is implemented in Python 3. The generators of automorphisms are obtained from `nauty` [44]. All evaluation is performed on a commodity computer with Intel Core i9-12900K processor and 128GB memory. Every optimization problem is solved by MOSEK [45]. Furthermore, all optimal routing solutions are double-checked for capacity violation using the method in [42].

### A. Throughput performance

**Partially deployed FatTree:** A fully deployed FatTree topology [1], constructed from 32-port switches, can accommodate 8192 servers. In practice, this topology is incrementally deployed in blocks when additional servers are needed. However, the routing for the fully deployed topology is not optimal for the partially deployed one due to the imbalance at the core switches, i.e., each core switch in Fig. 1 has two links to one block and a link to another block. Therefore, we apply our algorithm to find the optimal oblivious routing for partially deployed FatTree with different numbers of aggregation blocks. Note that the fully deployed topology has 32 aggregation blocks.

We evaluate the performance of our algorithm by the worst-case throughput,  $\min_{(u,v) \in \mathcal{C}} \theta^{uv}$ . The result is shown in Fig. 8 and is compared to a heuristic algorithm in [46], which tries to balance the imbalance by weighting flows regarding their bottleneck capacity. Our algorithm yields 12.5% – 87.5% throughput improvement over the heuristic algorithm when their throughput values are different. In addition to the worst-case throughput in Fig. 8, Fig. 9 shows the normalized total throughput,  $\frac{1}{|\mathcal{C}|} \sum_{(u,v) \in \mathcal{C}} \theta^{uv}$ , from the max-min with marginal throughput objective in (1) and the proportional fair objective in (2). The latter yields slightly higher normalized total throughput.

**DRing:** The DRing topology [12] is designed for moderate-scale datacenter networks. A DRing topology is constructed from supernodes where each supernode consists of switches. Every pair of switches is directly connected if they lie in adjacent supernodes. We compare the performance of the optimal routing solution from our algorithm with the solution from Shortest-Union(2) routing scheme suggested in [12].

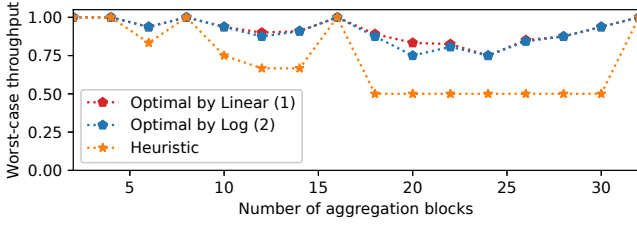


Fig. 8. Worst-case throughput values under partially deployed FatTree. At 30 aggregation blocks, the throughput improvement is 87.5%.

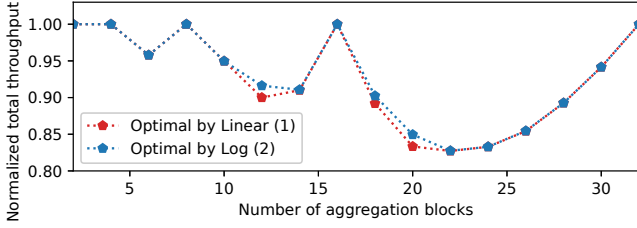


Fig. 9. Normalized total throughput values under partially deployed FatTree.

Table I shows the improvement of the worst-case throughput from three sizes of DRing. The small-size DRing is constructed from 6 supernodes with 2 switches per supernode and 10 servers per switch, as shown in Fig. 10. The large-size DRing is constructed from 10 supernodes with 20 switches per supernode and 80 servers per switch. The last setting follows the original paper [12]. The result shows that our optimal solution yields  $1.36\times$  to  $2.55\times$  throughput gain over the Shortest-Union(2) routing scheme.

Fig. 10 compares the routing solutions obtained from our algorithm and Shortest-Union(2) under the small-size DRing. There are two groups of commodities represented by commodities (0, 1) and (0, 2). There is no differences between the two routing schemes for commodity (0, 1), as shown in the left plot. However, the traffic is distributed differently under the two routing schemes for commodity (0, 2), as shown in the middle and right plots. Our optimal routing solution improves the worst-case throughput by dispersing traffic to every switch.

### B. Scalability under various topologies

**FatClique:** The FatClique topology [11] has been proposed for high manageability datacenter networks without routing. Our algorithm could provide this missing routing. Each FatClique topology is built as follows. The numbers of blocks, sub-blocks per block, and switches per sub-block are set to an identical value. We range this value from 2 to 12.

TABLE I  
THE WORST-CASE THROUGHPUT UNDER DRINGS

| Topology         | Worst-case throughput |                | Gain  |
|------------------|-----------------------|----------------|-------|
|                  | Shortest-Union(2)     | Optimal Linear |       |
| Small            | 0.25                  | 0.34           | 1.36x |
| Large            | 0.11                  | 0.28           | 2.55x |
| DRing paper [12] | 0.07                  | 0.16           | 2.29x |

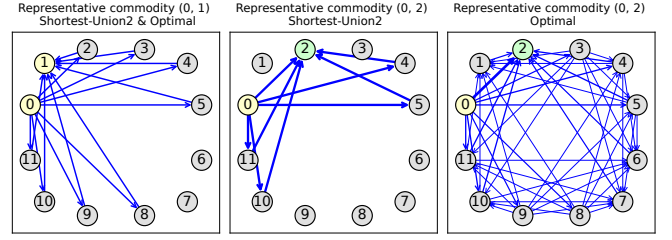


Fig. 10. The example of routing solutions under a small-size DRing. Each node represents a switch. The source and destination are assigned an identical color if they lie within the same supernode.

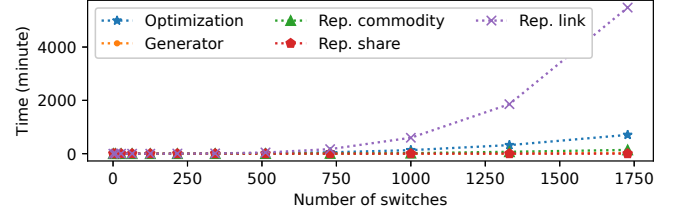


Fig. 11. The overall computation times at different FatClique's sizes are broken down into computing automorphism generators by *nauty* (Generator), finding representative commodities by Algorithm 2 (Rep. commodity), finding representative shares by Algorithm 3 (Rep. share), finding representative links utilizing Theorem 2 (Rep. link), and the execution time of Algorithm 1 (Optimization).

The computation time of Algorithm 1 at different sizes of FatClique is shown in Fig. 11. While finding the representative links is the most time-consuming part, it can be pre-computed before executing Algorithm 1. Fig. 12 shows the sizes of the strawman formulation in (3) and the automorphism-invariant formulation in (8) in terms of numbers of variables and constraints assuming only one traffic matrix is considered,  $|T| = 1$ . It is easy to see that the latter formulation is much smaller than the former one. In fact, the formulation in (3) cannot be computed beyond 216 switches due to insufficient computing memory. Fig. 13 shows the numbers of traffic matrices required for finding the optimal oblivious routing solutions in formulations (3) and (8). Again, the numbers of required traffic matrices in Algorithm 1 is much smaller than the set of all extreme points.

**Other topologies:** Next, we consider the optimization times of other topologies. We use the same settings for FatClique and partially deployed FatTree. For DRing, The number of switches per supernode is twice the number of supernodes, which is incremented by 2 from 4 to 20.

The result in Fig. 14 shows the optimization times of Algorithm 1 with the objectives in (1) and (2) and the optimization times of the linear program in [27].<sup>5</sup> The linear program cannot scale beyond 32 switches due to insufficient memory for all three topologies. Our algorithm scales well, and its optimization times under the two objectives are comparable.

### C. Structured topology with non-uniformity

Our algorithm is applicable for a more general structured topology with non-uniform link capacities and server distribu-

<sup>5</sup>The heuristic in [46] for FatTree does not involve solving optimization.

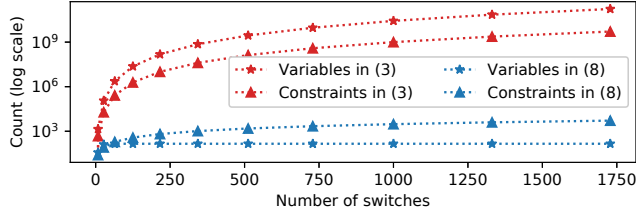


Fig. 12. The numbers of variables and constraints, assuming  $|\mathcal{T}| = 1$ , in formulations (3) and (8) are plotted at different sizes of FatClique. The maximum numbers for formulation (8) are respectively 144 and 5185 when topology has 1728 switches.

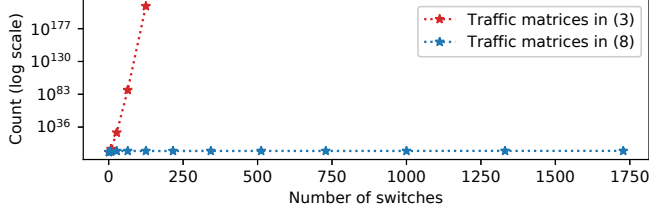


Fig. 13. The numbers of traffic matrices per link capacity constraint in formulations (3) and (8) are plotted at different sizes of FatClique. The maximum number for formulation (8) is 81 when topology has 1331 switches, while the numbers beyond  $10^{308}$  are not plotted for formulation (3) with more than 125 switches.

tion as illustrated in Fig. 15. The top-left plot shows the network with 4 groups of switches,  $\{0, 1\}$ ,  $\{2, 3, 4\}$ ,  $\{5, 6, 7, 8\}$ ,  $\{9, 10, 11\}$ , where the first three groups have different numbers of servers per switch and the last group has none. Logical links with different thicknesses have different capacities. The other plots show the optimal oblivious routing for different representative commodities. It is worth mentioning that the optimal routing solution for each representative commodity can be different. For example, commodities (0, 1) only uses one-hop intermediate switches, while commodity (0, 2) disperses traffic to all switches. This behavior differs from [15] where traffic demands from every commodity are dispersed to the same set of intermediate switches.

## VIII. CONCLUSION

This paper presents an iterative algorithm to find the optimal oblivious routing for structured topologies. The algorithm utilizes the insight that the optimal routing solution is automorphism invariant in order to reduce the solution space and

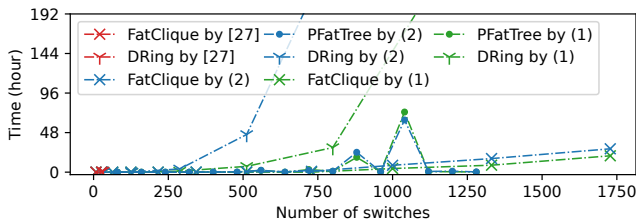


Fig. 14. The optimization time at different size of FatClique, partially deployed FatTree, and DRing. The limit of optimization time is set to 192 hours. The maximum number of switches is 1728 switches. Note that the smallest partially-deployed FatTree cannot be solved by [27] due to insufficient memory.

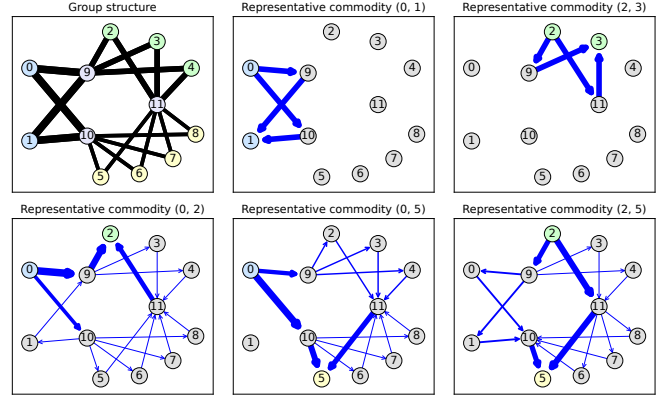


Fig. 15. Optimal oblivious routing for non-uniform and structured topology

complexity. This algorithm is applicable for designing oblivious routing for existing datacenter network topologies and future structured topologies with non-uniform link capacities and server distribution.

## REFERENCES

- [1] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, Aug. 2008. [Online]. Available: <https://doi.org/10.1145/1402946.1402967>
- [2] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "V12: A scalable and flexible data center network," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 51–62, Aug. 2009. [Online]. Available: <https://doi.org/10.1145/1594977.1592576>
- [3] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannan, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Hölzle, S. Stuart, and A. Vahdat, "Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 183–197, Aug. 2015. [Online]. Available: <https://doi.org/10.1145/2829988.2787508>
- [4] A. Andreyev, "Introducing data center fabric, the next-generation facebook data center network." [Online]. Available: <https://engineering.fb.com/2014/11/14/production-engineering/introducing-data-center-fabric-the-next-generation-facebook-data-center-network/>
- [5] J. H. Ahn, N. Binkert, A. Davis, M. McLaren, and R. S. Schreiber, "Hyperx: Topology, routing, and packaging of efficient large-scale networks," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, ser. SC '09. New York, NY, USA: Association for Computing Machinery, 2009. [Online]. Available: <https://doi.org/10.1145/1654059.1654101>
- [6] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *2008 International Symposium on Computer Architecture*, 2008, pp. 77–88.
- [7] M. Besta and T. Hoefler, "Slim fly: A cost effective low-diameter network topology," in *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2014, pp. 348–359.
- [8] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers randomly," in *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. San Jose, CA: USENIX Association, Apr. 2012, pp. 225–238. [Online]. Available: <https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/singla>
- [9] A. Valadarsky, G. Shahaf, M. Dinitz, and M. Schapira, "Xpander: Towards optimal-performance datacenters," in *Proceedings of the 12th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 205–219. [Online]. Available: <https://doi.org/10.1145/2999572.2999580>

- [10] A. Singla, P. B. Godfrey, and A. Kolla, "High throughput data center topology design," in *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. Seattle, WA: USENIX Association, Arp. 2014, pp. 29–41. [Online]. Available: <https://www.usenix.org/conference/nsdi14/technical-sessions/presentation/singla>
- [11] M. Zhang, R. N. Mysore, S. Supittayapornpong, and R. Govindan, "Understanding lifecycle management complexity of datacenter topologies," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*. Boston, MA: USENIX Association, Feb. 2019, pp. 235–254. [Online]. Available: <https://www.usenix.org/conference/nsdi19/presentation/zhang>
- [12] V. Harsh, S. A. Jyothi, and P. B. Godfrey, "Spineless data centers," in *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*, ser. HotNets '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 67–73. [Online]. Available: <https://doi.org/10.1145/3422604.3425945>
- [13] D. Thaler and C. Hopps. Rfc2991: Multipath issues in unicast and multicast next-hop selection. [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc2991>
- [14] R. Zhang-Shen and N. McKeown, "Guaranteeing quality of service to peering traffic," in *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*, 2008, pp. 1472–1480.
- [15] M. Kodialam, T. V. Lakshman, J. B. Orlin, and S. Sengupta, "Preconfiguring ip-over-optical networks to handle router failures and unpredictable traffic," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 5, pp. 934–948, 2007.
- [16] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'10. USA: USENIX Association, 2010, p. 19.
- [17] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 123–137, Aug. 2015. [Online]. Available: <https://doi.org/10.1145/2829988.2787472>
- [18] L. G. Valiant and G. J. Brebner, "Universal schemes for parallel communication," in *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, ser. STOC '81. New York, NY, USA: Association for Computing Machinery, 1981, p. 263–277. [Online]. Available: <https://doi.org/10.1145/800076.802479>
- [19] L. G. Valiant, "A scheme for fast parallel communication," *SIAM Journal on Computing*, vol. 11, no. 2, pp. 350–361, 1982. [Online]. Available: <https://doi.org/10.1137/0211027>
- [20] M. Kodialam, T. Lakshman, and S. Sengupta, "Efficient and robust routing of highly variable traffic," in *In Proceedings of Third Workshop on Hot Topics in Networks (HotNets-III)*, 2004.
- [21] M. Kodialam, T. V. Lakshman, and S. Sengupta, "Maximum throughput routing of traffic in the hose model," in *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, April 2006, pp. 1–11.
- [22] M. Kodialam, T. V. Lakshman, J. B. Orlin, and S. Sengupta, "Oblivious routing of highly variable traffic in service overlays and ip backbones," *IEEE/ACM Transactions on Networking*, vol. 17, no. 2, pp. 459–472, 2009.
- [23] M. Kodialam, T. V. Lakshman, and S. Sengupta, "Traffic-oblivious routing in the hose model," *IEEE/ACM Transactions on Networking*, vol. 19, no. 3, pp. 774–787, 2011.
- [24] H. Räcke, "Minimizing congestion in general networks," in *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, 2002, pp. 43–52.
- [25] H. Räcke, "Optimal hierarchical decompositions for congestion minimization in networks," in *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, ser. STOC '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 255–264. [Online]. Available: <https://doi.org/10.1145/1374376.1374415>
- [26] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Räcke, "Optimal oblivious routing in polynomial time," in *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, ser. STOC '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 383–388. [Online]. Available: <https://doi.org/10.1145/780542.780599>
- [27] D. Applegate and E. Cohen, "Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs," in *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 313–324. [Online]. Available: <https://doi.org/10.1145/863955.863991>
- [28] D. Applegate and E. Cohen, "Making routing robust to changing traffic demands: Algorithms and evaluation," *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1193–1206, 2006.
- [29] R. Zhang-Shen and N. McKeown, "Designing a predictable internet backbone with valiant load-balancing," in *Proceedings of the 13th International Conference on Quality of Service*, ser. IWQoS'05. Berlin, Heidelberg: Springer-Verlag, 2005, p. 178–192.
- [30] J. Fakcharoenphol, S. Rao, and K. Talwar, "A tight bound on approximating arbitrary metrics by tree metrics," in *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, ser. STOC '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 448–455. [Online]. Available: <https://doi.org/10.1145/780542.780608>
- [31] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: A scalable and fault-tolerant network structure for data centers," in *SIGCOMM08*. Association for Computing Machinery, Inc., August 2008. [Online]. Available: <https://www.microsoft.com/en-us/research/publication/dc-cell-a-scalable-and-fault-tolerant-network-structure-for-data-centers/>
- [32] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, S. Lu, and G. Lv, "Bcube: A high performance, server-centric network architecture for modular data centers," in *ACM SIGCOMM*. Association for Computing Machinery, Inc., August 2009. [Online]. Available: <https://www.microsoft.com/en-us/research/publication/bcube-a-high-performance-server-centric-network-architecture-for-modular-data-centers/>
- [33] Broadcom. Tomahawk4 / bcm56990 series. [Online]. Available: <https://www.broadcom.com/products/ethernet-connectivity/switching/strataxgs/bcm56990-series>
- [34] S. Supittayapornpong, B. Raghavan, and R. Govindan, "Towards highly available clos-based wan routers," in *Proceedings of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 424–440. [Online]. Available: <https://doi.org/10.1145/3341302.3342086>
- [35] P. Namyar, S. Supittayapornpong, M. Zhang, M. Yu, and R. Govindan, "A throughput-centric view of the performance of datacenter topologies," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, ser. SIGCOMM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 349–369. [Online]. Available: <https://doi.org/10.1145/3452296.3472913>
- [36] S. Supittayapornpong, P. Namyar, M. Zhang, M. Yu, and R. Govindan, "Optimal oblivious routing for structured networks," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, 2022, pp. 1988–1997.
- [37] F. Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, vol. 8, no. 1, pp. 33–37, 1997. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.4460080106>
- [38] R. Srikant and L. Ying, *Communication Networks: An Optimization, Control and Stochastic Networks Perspective*. USA: Cambridge University Press, 2014.
- [39] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 556–567, 2000.
- [40] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.
- [41] S. Boyd and L. Vandenberghe, "Localization and cutting-plane methods," *From Stanford EE 364b lecture notes*, 2007.
- [42] B. Towles and W. Dally, "Worst-case traffic for oblivious routing functions," *IEEE Computer Architecture Letters*, vol. 1, no. 1, pp. 4–4, 2002.
- [43] S. A. Jyothi, A. Singla, P. B. Godfrey, and A. Kolla, "Measuring and understanding throughput of network topologies," in *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2016, pp. 761–772.
- [44] B. D. McKay and A. Piperno, "Practical graph isomorphism, ii," *Journal of Symbolic Computation*, vol. 60, pp. 94–112, 2014. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0747717113001193>
- [45] M. ApS, *MOSEK Fusion API for Python 9.3.21*, 2022. [Online]. Available: <https://docs.mosek.com/9.3/pythonfusion/index.html>
- [46] J. Zhou, M. Tewari, M. Zhu, A. Kabbani, L. Poutievski, A. Singh, and A. Vahdat, "Wcmp: Weighted cost multipathing for improved fairness in data centers," in *Proceedings of the Ninth European Conference on Computer Systems*, ser. EuroSys '14. New York, NY, USA: Association for Computing Machinery, 2014. [Online]. Available: <https://doi.org/10.1145/2592798.2592803>



**Kanatip Chitavisutthivong** received the B.Eng. degree in computer engineering from Kasetsart University, Thailand. He is currently pursuing the M.Eng. degree in information science and technology from Vidyasirimedhi Institute of Science and Technology, Thailand. His research interests include routing, datacenter networking, and optimization.



**Sucha Supittayapornpong** is a faculty member in the School of Information Science and Technology at Vidyasirimedhi Institute of Science and Technology, Thailand. He received his B.Eng. in computer engineering from Kasetsart University, his M.Eng. in telecommunications from Asian Institute of Technology, and his Ph.D. in Electrical Engineering from the University of Southern California. His research interests include datacenter networking, performance optimization, and operations research.



**Pooria Namyar** is a Ph.D. student in Electrical Engineering at the University of Southern California. He received his Bachelor's degree in Electrical Engineering from the Sharif University of Technology, IRAN, in 2019. His research focuses on optimizing the performance and availability of large-scale cloud systems.



**Mingyang Zhang** is a software engineer at Google. He received his PhD in computer science from University of Southern California. His research interests include datacenter networks and systems.



**Minlan Yu** is a Gordon McKay professor at Harvard School of Engineering and Applied Science. She received her B.A. in computer science and mathematics from Peking University and her M.A. and PhD in computer science from Princeton University. Her research interests include data networking, distributed systems, enterprise and data center networks, and software-defined networking.



**Ramesh Govindan** is the Northrop Grumman Chair in Engineering and Professor of Computer Science and Electrical Engineering at the University of Southern California. His research interests include routing and measurements in large internets, networked sensing systems, and mobile computing systems.