Designing Optimal Compact Oblivious Routing for Datacenter Networks in Polynomial Time

Kanatip Chitavisutthivong[†], Chakchai So-In[‡], Sucha Supittayapornpong[†]
[†]Vidyasirimedhi Institute of Science and Technology, Thailand
[‡]Khon Kaen University, Thailand

***Abstract*—Recent datacenter network topologies are shifting towards heterogeneous and structured topologies for high throughput, low cost, and simple manageability. However, they rely on sub-optimal routing approaches that fail to achieve their designed capacity. This paper proposes a process for designing optimal oblivious routing that is programmed compactly on programmable switches. The process consists of three contributions in tandem. We first transform a robust optimization problem for designing oblivious routing into a linear program, which is solvable in polynomial time but cannot scale for datacenter topologies. We then prove that the repeated structures in a datacenter topology lead to a structured optimal solution. We use this insight to formulate a scalable linear program, so an optimal oblivious routing solution is obtained in polynomial time for large-scale topologies. For real-world deployment, the optimal solution is converted into forwarding rules for programmable switches with stringent memory. With this constraint, we utilize the repeated structures in the optimal solution to group the forwarding rules, resulting in compact forwarding rules with a much smaller memory requirement. Extensive evaluations show our process i) obtains optimal solutions faster and more scalable than a state-of-the-art technique and ii) reduces the memory requirement by no less than 90% for most considered topologies.**

I. Introduction

The design of network topologies for data centers is currently shifting towards heterogeneous, structured topologies for high capacity, low cost, and low manageability [1], [2]. This attempts to replace the folded-Clos family, including FatTree [3], Google's Jupiter [4], Facebook's Fabric [5], and Microsoft's VL2 [6]. Alternative topologies have been proposed, such as Xpander [7], FatClique [2], DRing [8], SlimFly [9], and other server-centric networks, where servers have routing capability, such as BCube [10] and DCell [11]. However, some of them [2], [7], [8], [11] rely on sub-optimal routing approaches, while the others [9], [10] use dynamic routing that requires specialized hardware. This leads to an active research problem of how to design oblivious routing that achieves designed capacity without specialized hardware for these and future topologies [12].

The oblivious routing approach is widely used in production networks because of its robustness and simplicity. The routes from every source to every destination are designed to be robust to traffic variation, so neither re-configuration

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.

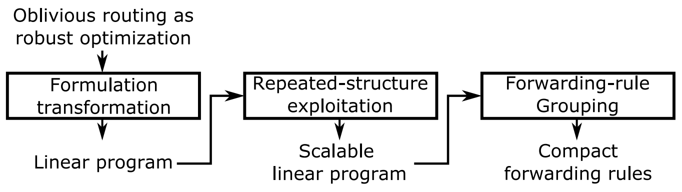
Oblivious routing as robust optimization
Formulation transformation
Linear program
Repeated-structure exploitation
Scalable linear program
Forwarding-rule Grouping
Compact forwarding rules

Fig. 1. The design process of optimal compact oblivious routing.

nor dynamic routing is required when traffic changes. For example, the folded-Clos family often employs the Equal-Cost Multi Paths (ECMP) routing approach [13] that splits traffic equally across all equal-cost paths, whose cost is independent of traffic. Another oblivious routing approach is Valiant Load Balancing (VLB) [14], [15], in which a traffic from a source is split equally to a set of intermediate switches, each then routes the received traffic to a destination. While, the ECMP and VLB approaches are optimal respectively for the folded-Clos and clique networks, they are sub-optimal for the alternative topologies due to topological differences.

Generally, designing an optimal oblivious routing solution for an arbitrary network topology is equivalent to solving a robust optimization problem [12], [15]–[17]. The work in [16] views this problem as a game and derived a linear program for intra-domain networks, albeit small-scale networks in comparison to datacenter networks. The recent work in [12] exploits the repeated network structures to reduce the complexity of a robust optimization problem, so an optimal oblivious routing solution is obtained for larger network sizes. In short, the first work can obtain the optimal routing solution in polynomial time (in the size of input instance, which is extremely large for large-scale networks), while the second work can scale to larger networks but could take non-polynomial time due to the complexity of robust optimization. This beg an open question: *Could we design optimal oblivious routing in polynomial time that also scales for large datacenter networks to achieve the best of both worlds?*

The memory constraint is another important issue for the real-world deployment of the oblivious routing. After an optimal routing solution is obtained, it is converted to forwarding rules that determine how traffic is split at each switch in a network. These rules are stored on switches with limited memory capacity, which becomes an issue for large-scale datacenter networks with thousands of destinations. The previous work in [18] trade-off split accuracy with memory requirement, so the rules can fit available memory. This beg another question: *Could we reduce the memory requirement given an optimal oblivious routing solution?*

In this paper, we propose a process for designing optimal compact oblivious routing for datacenter networks to address the two challenges above, as shown in Figure 1. We first formulate an oblivious routing problem as robust optimization for general datacenter networks, including those alternative topologies to the folded-Clos family. We then transform the robust optimization into a linear program using decomposition and duality techniques. This reduces the complexity of optimal oblivious routing design to just solving a linear program that is polynomial-time solvable and is tractable for small-scale datacenter networks. Since datacenter networks usually have repeated structures, we show the existence of an automorphism-invariant optimal solution, whose variables have repeated values. From this insight, we formulate a scalable linear program that can scale to larger network sizes. As a result, we can design optimal oblivious routing for large-scale datacenter networks with polynomial-time complexity, which addresses the first challenge. For the second challenge, we exploit the structure of the optimal oblivious routing solution, obtained from the scalable linear program, to group the forwarding rules. This grouping compacts the forwarding rules and significantly reduces memory requirement, which addresses the second challenge.

Extensive evaluations of the scalable linear program on various datacenter network topologies and sizes show that the linear program is faster and more scalable than the state-of-the-art technique in [12] and another linear formulation in [16]. Moreover, the forwarding rules grouping significantly reduces forwarding rules' memory requirement by no less than 90% for the majority of considered topologies. We also provide possible applications of the optimal compact oblivious routing.

The contributions of this work are summarized as follows:

- We transform a robust optimization problem for designing oblivious routing into a linear program, so an optimal oblivious routing solution is obtained in polynomial time for small-scale topologies.
- We prove that a datacenter network topology with repeated structures has an optimal solution whose variables also have repetition. We use this insight to formulate a scalable linear program that is solvable in polynomial time and is tractable for larger topologies.
- We utilize the structure of the optimal solution of the scalable linear program, to compact forwarding rules and reduce memory requirement for real-world deployment.

Besides, we release our code to the research community.

This paper is organized as follows. Section II describes the model of datacenter networks and the oblivious routing problem. Section III transforms the problem into a linear program. The repeated network structures are exploited to improve scalability in Section IV. Section V describes a grouping technique to compact forwarding rules for real-world deployment. Extensive evaluations are provided in Section VI before the conclusion in Section VII.

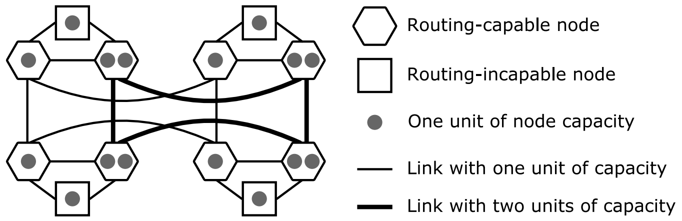
Routing-capable node
Routing-incapable node
One unit of node capacity
Link with one unit of capacity
Link with two units of capacity

Fig. 2. A simple network consists of routing-capable nodes, routing-incapable nodes, and links. These nodes and links could have different capacities.

II. System Model

We formally model datacenter networks, traffic demands, and routing components before formulating an oblivious routing optimization problem.

A. Network model

Our network model generalizes the previous model in [12] to capture both the traditional server-switch architecture [1], [2], [7]–[9], [19], [20] and the server-centric architecture [10], [11]. The differences between these architectures are that every server in the former has only one direct connection to a switch and has no routing capability, while every server in the latter has the routing capability and may have multiple direct connections to other servers or switches. To deal with these differences, we model a datacenter network by a graph with two types of nodes: routing-capable nodes and routing-incapable nodes, as shown in Figure 2. Let $\mathcal{S}$ and $\mathcal{P}$ be the sets of routing-capable nodes and routing-incapable nodes respectively. The set of all nodes is denoted by $\mathcal{N} = \mathcal{S} \cup \mathcal{P}$.

Nodes are interconnected by links. Node i and node j are connected by link (i,j) with capacity C_{ij}. We denote the set of all links by $\mathcal{L}$. Every link is bi-directional, so $(i,j) \in \mathcal{L}$ if and only if $(j,i) \in \mathcal{L}$ and $C_{ij} = C_{ji}$ for every $(i,j) \in \mathcal{L}$.

B. Traffic model

Node n has capacity H_n to independently send and receive traffic for every $n \in \mathcal{N}$. We denote the set of nodes that have positive capacities by $\mathcal{H} = \{n \in \mathcal{N} : H_n > 0\}$. A commodity (u,v) is a pair of source node u and destination node v where both nodes have positive capacity. We denote the set of all commodities by $\mathcal{C} = \big\{(u,v) \in \mathcal{H}^2 : u \neq v\big\}$.

Node u sends traffic to node v and creates a traffic demand for commodity (u,v). This traffic demand is denoted by t^{uv}. The traffic demands from every commodity form a traffic matrix $[t^{uv}] \in \mathbb{R}_+^{|\mathcal{C}|}$, where $\mathbb{R}_+$ is a set of non-negative reals.

Since traffic matrices inside a datacenter network could be unpredictable, we consider the set of all possible traffic matrices [12], [15], [21]:

$$\mathcal{T} = \left\{ \begin{array}{cl} \sum_{v \in \mathcal{H}_{-u}} t^{uv} \leq H_u & ,\forall u \in \mathcal{H} \\ \sum_{u \in \mathcal{H}_{-v}} t^{uv} \leq H_v & ,\forall v \in \mathcal{H} \\ t^{uv} \in \mathbb{R}_+ & ,\forall (u,v) \in \mathcal{C} \end{array} \right\}, \tag{1}$$

where $\mathcal{A}_{-x}$ contains all members of $\mathcal{A}$ but excludes x, i.e., $\mathcal{A}_{-x} = \mathcal{A} \backslash \{x\}$. The first constraint in (1) limits the amount of traffic generated from each node by the node's capacity.

Similarly, the second constraint bounds the amount of traffic each node can receive by the node's capacity.

C. Routing model

We adopt the multi-commodity flow model with the commodity set $\mathcal{C}$. For every commodity $(u,v) \in \mathcal{C}$, the share of commodity's demand over link (i,j) is denoted by f_{ij}^{uv} for every link $(i,j) \in \mathcal{L}$. Notice that, given a traffic demand t^{uv}, the amount of the demand over link (i,j) is $t^{uv} f_{ij}^{uv}$. These shares over all links in the network, $\left\{f_{ij}^{uv} : (i,j) \in \mathcal{L}\right\}$, form the shares of commodity (u,v). We denote the shares of all commodities by $[f_{ij}^{uv}]$, where $[f_{ij}^{uv}] \in \mathbb{R}_+^{|\mathcal{C}| \times |\mathcal{L}|}$.

Recall that there are two node types. A routing-capable node can route every traffic independent of the traffic's commodity. A routing-incapable node can only route traffic corresponding to the commodities involving the node, i.e., the node is either a source or a destination of the traffic. Define $\mathbb{I}[x]$ as an indicator function that returns 1 if the statement x is true; otherwise 0. We define the set of all possible shares as follows:

$$\mathcal{F} = \left\{ \begin{array}{ll} \sum\limits_{j \in \mathcal{O}(i)} f_{ij}^{uv} - \sum\limits_{j \in \mathcal{I}(i)} f_{ji}^{uv} = \mathbb{I}[i=u] - \mathbb{I}[i=v] & \\ \qquad\qquad\qquad ,\forall (u,v) \in \mathcal{C}, \forall i \in \mathcal{N} & \\ f_{ij}^{uv} = 0 \qquad\quad ,\forall (u,v) \in \mathcal{C}, \forall (i,j) \in \mathcal{N} \times \mathcal{P}_{-v} & \\ f_{iu}^{uv} = f_{vi}^{uv} = 0, \forall (u,v) \in \mathcal{C}, \forall i \in \mathcal{N} & \\ f_{ij}^{uv} \in \mathbb{R}_+ \qquad ,\forall (u,v) \in \mathcal{C}, \forall (i,j) \in \mathcal{L} & \end{array} \right\}, \tag{2}$$

where $\mathcal{O}(i)$ and $\mathcal{I}(i)$ are respectively the set of nodes corresponding to the outgoing links from node i and the set of nodes corresponding to the incoming links to node i. The first constraint is the share conservation where the total share at every source and every destination is one. The second constraint enforces that every routing-incapable node does not involve in the routes of other commodities unrelated to the node. The third constraint ensures that both source and destination are the endpoints of their corresponding traffic.

D. Oblivious routing formulation

Given shares $[f_{ij}^{uv}]$ and a traffic matrix $[t^{uv}]$, we define the congestion ratio of link (x,y) by the proportion of total traffic demand on the link to the link's capacity [16], [17]:

$$\sum_{(u,v) \in \mathcal{C}} \frac{t^{uv} f_{xy}^{uv}}{C_{xy}}.$$

An oblivious routing problem aims to design the shares $[f_{ij}^{uv}]$ that minimizes this congestion ratio across every link in the network and under all possible traffic demands.[1] We formulate this oblivious routing problem as follows:

$$\min_{[f_{ij}^{uv}] \in \mathcal{F}} \max_{\substack{(x,y) \in \mathcal{L}, \\ [t^{uv}] \in \mathcal{T}}} \sum_{(u,v) \in \mathcal{C}} \frac{t^{uv} f_{xy}^{uv}}{C_{xy}}. \tag{3}$$

This formulation has two challenges: the robust objective with infinite possibilities of traffic demands and the scalability of the formulation for large networks. The first challenge stems from the fact that the congestion ratio is a function of every traffic demand in the traffic set $\mathcal{T}$, so the entire set must be considered. The second challenge is due to the sizes of datacenter networks, resulting in large numbers of decision variables and constraints in the formulation. We tackle the two challenges successively in Section III and Section IV.

[1]This is equivalent to the performance ratio in [16], [17] when the minimum possible maximum link utilization is ignored.

III. Linear Formulation

This section tackles the challenge of the robust objective by transforming the robust formation in (3) to a simpler linear program, so an optimal oblivious routing solution can be designed for appropriate network sizes. This transformation is inspired by [16], which considers a different network model without the traffic set and uses different proof techniques.

We first observe that each link has its worst-cast traffic matrix, yielding its maximal congestion ratio. Therefore, we introduce traffic matrix $[t_{xy}^{uv}] \in \mathcal{T}$ for every link (x,y) to decouple the inner maximization in (3) as follows:

$$\max_{\substack{(x,y) \in \mathcal{L}, \\ [t^{uv}] \in \mathcal{T}}} \sum_{(u,v) \in \mathcal{C}} \frac{t^{uv} f_{xy}^{uv}}{C_{xy}} = \max_{(x,y) \in \mathcal{L}} \max_{[t_{xy}^{uv}] \in \mathcal{T}} \sum_{(u,v) \in \mathcal{C}} \frac{t_{xy}^{uv} f_{xy}^{uv}}{C_{xy}} \tag{4}$$

The above decomposition allows us to consider the sub-problem, $\max_{[t_{xy}^{uv}] \in \mathcal{T}} \sum_{(u,v) \in \mathcal{C}} \frac{t_{xy}^{uv} f_{xy}^{uv}}{C_{xy}}$, which finds the worst-case congestion ratio of link (x,y), in isolation. Given a considered link (x,y) and a share solution $[f_{ij}^{uv}]$, this sub-problem is rewritten as the following formulation:

$$\begin{array}{lll} \text{Maximize} & \sum\limits_{(u,v) \in \mathcal{C}} \frac{t_{xy}^{uv} f_{xy}^{uv}}{C_{xy}} & \\ \text{Subject to} & \sum\limits_{v \in \mathcal{H}_{-u}} t_{xy}^{uv} \leq H_u & ,\forall u \in \mathcal{H} \\ & \sum\limits_{u \in \mathcal{H}_{-v}} t_{xy}^{uv} \leq H_v & ,\forall v \in \mathcal{H} \\ & t_{xy}^{uv} \in \mathbb{R}_+ & ,\forall (u,v) \in \mathcal{C}. \end{array} \tag{5}$$

Instead of solving this sub-problem directly, we consider its dual problem with dual variables $[\beta_{xy}^u]_{u \in \mathcal{H}}$ and $[\gamma_{xy}^u]_{u \in \mathcal{H}}$:

$$\begin{array}{lll} \text{Minimize} & \sum\limits_{u \in \mathcal{H}} H_u \left(\beta_{xy}^u + \gamma_{xy}^u\right) & \\ \text{Subject to} & \frac{f_{xy}^{uv}}{C_{xy}} - \beta_{xy}^u - \gamma_{xy}^v \leq 0 & ,\forall (u,v) \in \mathcal{C} \\ & \beta_{xy}^u \in \mathbb{R}_+, \ \gamma_{xy}^u \in \mathbb{R}_+ & ,\forall u \in \mathcal{H}. \end{array} \tag{6}$$

Lemma 1. *For a given link (x,y) and a share solution $[f_{ij}^{uv}]$, the formulation in* (6) *is the dual of the problem in* (5).

Proof: Fix a link (x,y) and a share solution $[f_{ij}^{uv}]$. We derive a dual problem by the duality technique [22]. Define non-negative dual variables $\left[\beta_{xy}^u\right]_{u \in \mathcal{H}}$, $\left[\gamma_{xy}^u\right]_{u \in \mathcal{H}}$ respectively for the first two inequality constraints of the sub-problem in (5). We define the Lagrangian associated with this problem as

$$L\left(\left[t^{uv}_{xy}\right],\left[\beta^{u}_{xy}\right],\left[\gamma^{v}_{xy}\right]\right)=\sum_{(u,v)\in\mathcal{C}}\frac{t^{uv}_{xy}f^{uv}_{xy}}{C_{xy}}$$
$$-\sum_{u\in\mathcal{H}}\beta^{u}_{xy}\left(\sum_{v\in\mathcal{H}_{-u}}t^{uv}_{xy}-H_u\right)-\sum_{v\in\mathcal{H}}\gamma^{v}_{xy}\left(\sum_{u\in\mathcal{H}_{-v}}t^{uv}_{xy}-H_v\right)$$
$$=\sum_{(u,v)\in\mathcal{C}}t^{uv}_{xy}\left(\frac{f^{uv}_{xy}}{C_{xy}}-\beta^{u}_{xy}-\gamma^{v}_{xy}\right)+\sum_{u\in\mathcal{H}}H_u\left(\beta^{u}_{xy}+\gamma^{u}_{xy}\right).$$

The dual function is defined as

$$D\left(\left[\beta^{u}_{xy}\right],\left[\gamma^{v}_{xy}\right]\right)=\max_{\left[t^{uv}_{xy}\right]\in\mathbb{R}^{|\mathcal{C}|}_{+}}L\left(\left[t^{uv}_{xy}\right],\left[\beta^{u}_{xy}\right],\left[\gamma^{v}_{xy}\right]\right).$$

Note that this dual function is undefined, $D\left(\left[\beta^{u}_{xy}\right],\left[\gamma^{v}_{xy}\right]\right)\to\infty$ when $\frac{f^{uv}_{xy}}{C_{xy}}-\beta^{u}_{xy}-\gamma^{v}_{xy}>0$ for some commodity $(u,v)\in\mathcal{C}$. We only consider the well-defined dual function $D\left(\left[\beta^{u}_{xy}\right],\left[\gamma^{v}_{xy}\right]\right)=\sum_{u\in\mathcal{H}}H_u\left(\beta^{u}_{xy}+\gamma^{u}_{xy}\right)$ when $\frac{f^{uv}_{xy}}{C_{xy}}-\beta^{u}_{xy}-\gamma^{v}_{xy}\le 0$ for every commodity $(u,v)\in\mathcal{C}$. This is because a dual problem will minimize this dual function, i.e., $\min_{\left[\beta^{u}_{xy}\right]\in\mathbb{R}^{|\mathcal{H}|}_{+},\left[\gamma^{v}_{xy}\right]\in\mathbb{R}^{|\mathcal{H}|}_{+}}D\left(\left[\beta^{u}_{xy}\right],\left[\gamma^{v}_{xy}\right]\right)$, resulting in the dual problem in (6), which proves the lemma. ■

We use the dual problem in (6) to transform the robust optimization problem in (3) into the following linear program:

$$\begin{aligned}\text{Minimize}\quad & \eta\\ \text{Subject to}\quad & \sum_{u\in\mathcal{H}}H_u\left(\beta^{u}_{ij}+\gamma^{u}_{ij}\right)\le\eta\ \ ,\forall(i,j)\in\mathcal{L}\\ & \frac{f^{uv}_{ij}}{C_{ij}}-\beta^{u}_{ij}-\gamma^{v}_{ij}\le 0\ \ ,\forall(u,v)\in\mathcal{C},\forall(i,j)\in\mathcal{L}\\ & \beta^{u}_{ij}\in\mathbb{R}_{+},\ \gamma^{u}_{ij}\in\mathbb{R}_{+}\ \ ,\forall u\in\mathcal{H},\forall(i,j)\in\mathcal{L}\\ & [f^{uv}_{ij}]\in\mathcal{F}.\end{aligned}\tag{7}$$

Theorem 1. *The robust oblivious routing formulation in* (3) *and the linear program in* (7) *are equivalent.*

Proof: The proof replaces the sub-problem in (5) with the dual problem in (6) on the right-hand side of equation (4). Let $g_{xy}([f^{uv}_{ij}])$ represent the dual problem in (6) of link (x,y) given routing variables $[f^{uv}_{ij}]$. The replacement yields:

$$\max_{\substack{(i,j)\in\mathcal{L},\\ [t^{uv}]\in\mathcal{T}}}\sum_{(u,v)\in\mathcal{C}}\frac{t^{uv}f^{uv}_{ij}}{C_{ij}}=\max_{(x,y)\in\mathcal{L}}g_{xy}([f^{uv}_{ij}]).$$

Since the left-hand side of the above equation is the inner maximization of the oblivious routing formulation in (3), it follows that the formulation is equivalent to

$$\min_{[f^{uv}_{ij}]\in\mathcal{F}}\ \max_{(x,y)\in\mathcal{L}}g_{xy}([f^{uv}_{ij}]).$$

The above min-max problem can be transformed into a linear program with an auxiliary variable η as follows:

$$\begin{aligned}\text{Minimize}\quad & \eta\\ \text{Subject to}\quad & g_{xy}([f^{uv}_{ij}])\le\eta\quad,\forall(x,y)\in\mathcal{L}\\ & [f^{uv}_{ij}]\in\mathcal{F}.\end{aligned}$$

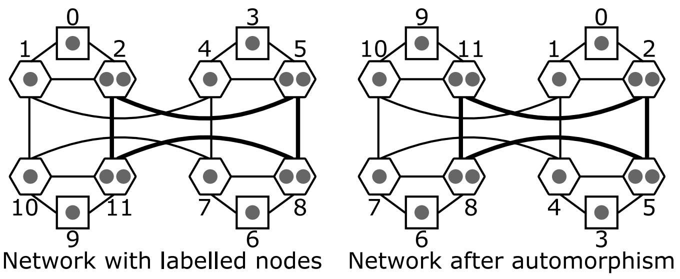

Fig. 3. Automorphism example. Nodes in the right network are nodes in the left network relabeled by the automorphism $\phi(n)=n-3\mod 12$. Both networks have the same adjacency, link capacity distribution, node types, and node capacity distribution.

Finally, replacing the dual problem $g_{xy}([f^{uv}_{ij}])$ with its objective function and constraints in (6) gives the linear program in (7) and proves the theorem. ■

The implication of Theorem 1 is that an optimal oblivious routing solution of the robust formulation in (3) can be obtained by solving the linear program in (7). Since linear programming is solvable in polynomial time [23], [24], we can design optimal oblivious routing in polynomial time for appropriate network sizes, in terms of variables and constraints. However, the scales of datacenter networks could render the linear program in (7) intractable in practice. The next section tackles this scalability issue.

IV. Scalable Linear Formulation

This section tackles the scalability of the linear formulation in (7) for large-scale datacenter networks. We exploit the repeated structures in a network topology to scale the formulation. This idea is inspired by the work in [12], which is non-polynomial time and has no routing-incapable nodes.

A. Automorphism in datacenter networks

The repeated structures in a datacenter network is captured by graph automorphism. Graph automorphism is a mapping of nodes in a graph to the same set of nodes in the same graph such that the new graph with the mapped nodes is similar to the original graph [25]. In particular, the described automorphism preserves node adjacency. We extend this automorphism to our network model by the definition of our automorphism.

Definition 1. *A mapping function $\phi:\mathcal{N}\to\mathcal{N}$ is an automorphism if the following conditions hold:*

1) *Adjacency: $(\phi(i),\phi(j))\in\mathcal{L}$, $\forall(i,j)\in\mathcal{L}$.*
2) *Link capacity: $C_{\phi(i)\phi(j)}=C_{ij}$, $\forall(i,j)\in\mathcal{L}$.*
3) *Type: $\phi(n)\in\mathcal{S}$, $\forall n\in\mathcal{S}$ and $\phi(n)\in\mathcal{P}$, $\forall n\in\mathcal{P}$.*
4) *Node capacity: $H_{\phi(n)}=H_n$, $\forall n\in\mathcal{N}$.*

The automorphism in Definition 1 is illustrated in Figure 3. The first condition ensures that the adjacency between nodes are preserved under the automorphism. The second condition keeps the capacity of every link the same after the automorphism mapping. The type and capacity of every node also stay the same under the automorphism from the last two conditions.

Let Φ be the set of all automorphisms satisfying Definition 1. Since the numbers of nodes and links in a network are

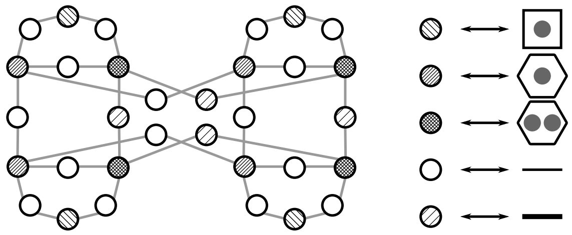

Fig. 4. A graph with colored vertices is constructed by Algorithm 1 from the simple network in Figure 2.

finite, there are finite permutations, and the cardinality of the set is finite, $|\Phi| < \infty$. In addition, we define $\hat{\Phi}$ as the set of generators that entirely generate the automorphism set Φ. This generator set can be obtained from off-the-shelf software, such as nauty [26].

Algorithm 1: Graph construction for generator set $\hat{\Phi}$

```
Initialize a graph with a vertex set $\mathcal{V}$ and an edge set $\mathcal{E}$
Initialize dictionaries $\Gamma, W, W'$ with values as colors
for $n \in \mathcal{N}$ do
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{n\}$
    if $(\mathbb{I}[n \in \mathcal{S}], H_n) \notin W$ then
        $W[(\mathbb{I}[n \in \mathcal{S}], H_n)] \leftarrow$ new color
    $\Gamma[n] \leftarrow W[(\mathbb{I}[n \in \mathcal{S}], H_n)]$
for $(i,j) \in \mathcal{L}$ do
    Let $n$ be a new vertex
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{n\}$
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(i,n),(n,j)\}$
    if $C_{ij} \notin W'$ then
        $W'[C_{ij}] \leftarrow$ new color
    $\Gamma[n] \leftarrow W'[C_{ij}]$
return a graph with sets $\mathcal{V}, \mathcal{E}$ and dictionary $\Gamma$
```

To obtain the generator set $\hat{\Phi}$, we encode the four properties in Definition 1 on a network topology as a graph with colored vertices by Algorithm 1. Figure 4 shows the graph constructed from the network in Figure 2 by the algorithm. The algorithm constructs a graph with edges and colored vertices from a network topology as follows. Vertices associated with nodes having the same type and node capacity are assigned the same color. Any two vertices associated with nodes having different types or different capacities are given different colors. This construction ensures the preservation of node type and node capacity in the generator set. For links in the network, each link yields a new vertex with two edges, each connects to the vertex associated with each side of the link. Vertices associated with links having the same capacity are assigned the same color. Any two vertices associated with links having different capacities are given different colors. This construction ensures the properties on adjacency and link capacity are enforced.

We use the automorphism set Φ and its generator set $\hat{\Phi}$ to identify a special optimal oblivious routing solution of the linear program in (7) in the next section.

B. Automorphism-invariant optimal solution

In this section, we show that the repeated structures in a datacenter network lead to an optimal oblivious routing solution that also has repeated structures.

We first argue that the linear program in (7) has an optimal solution. We know that the robust optimization in (3) aims to minimize the congestion ratio over the traffic set, which is assumed to be nonempty, and the sum of all link capacities in the network is finite. Therefore, the minimal congestion ratio exists. Since the linear program in (7) is equivalent to the robust optimization in (3) by Theorem 1, it must have an optimal solution. Let $\left(\eta^*, [\beta_{ij}^{u*}], [\gamma_{ij}^{u*}], [f_{ij}^{uv*}]\right)$ be an optimal solution of the linear program in (7). We then show that there is another optimal solution that can be constructed from an automorphism.

Lemma 2. *A solution $\left(\eta, [\beta_{ij}^{u}], [\gamma_{ij}^{u}], [f_{ij}^{uv}]\right)$ is an optimal solution of the linear program in* (7) *given any $\phi \in \Phi$ and*

$$\eta = \eta^*, \quad \beta_{ij}^{u} = \beta_{\phi(i)\phi(j)}^{\phi(u)*}, \quad \gamma_{ij}^{u} = \gamma_{\phi(i)\phi(j)}^{\phi(u)*}, \quad f_{ij}^{uv} = f_{\phi(i)\phi(j)}^{\phi(u)\phi(v)*}.$$

Proof: Fix an automorphism $\phi \in \Phi$. Since $\eta = \eta^*$, the objective cost under the new solution equals the optimal objective cost. We then need to show that the solution, constructed from the automorphism, is feasible and satisfies all constraints in (7) and (2). We begin with the domains of variables β_{ij}^{u} and γ_{ij}^{u} in (7) and variables f_{ij}^{uv} in (2). These domains are nonnegative reals, so the variables of the constructed solution are in the same domains. Next, we consider the constraints in (7).

Considering the first constraint in (7) with link (i,j), we have that

$$\begin{aligned}\sum_{u \in \mathcal{H}} H_u \left(\beta_{ij}^{u} + \gamma_{ij}^{u}\right) &= \sum_{u \in \mathcal{H}} H_{\phi(u)} \left(\beta_{\phi(i)\phi(j)}^{\phi(u)*} + \gamma_{\phi(i)\phi(j)}^{\phi(u)*}\right) \\ &= \sum_{u \in \mathcal{H}} H_u \left(\beta_{\phi(i)\phi(j)}^{u*} + \gamma_{\phi(i)\phi(j)}^{u*}\right) \leq \eta^* = \eta.\end{aligned}$$

The first equality substitutes the solution and uses the node capacity property in Definition 1. Reindexing $u \in \mathcal{H}$ leads to the second equality. The last inequality holds because the constraint associated with link $(\phi(i), \phi(j))$ holds under the optimal solution. Thus, the first constraint in (7) is satisfied.

Considering the second constraint in (7) with commodity (u,v) and link (i,j), we have that

$$\frac{f_{ij}^{uv}}{C_{ij}} - \beta_{ij}^{u} - \gamma_{ij}^{v} = \frac{f_{\phi(i)\phi(j)}^{\phi(u)\phi(v)*}}{C_{\phi(i)\phi(j)}} - \beta_{\phi(i)\phi(j)}^{\phi(u)*} - \gamma_{\phi(i)\phi(j)}^{\phi(v)*} \leq 0.$$

The first equality substitutes the solution and uses the link capacity property in Definition 1. The last inequality holds because the constraint associated with commodity $(\phi(u), \phi(v))$ and link $(\phi(i), \phi(j))$ holds under the optimal solution. Thus, the second constraint in (7) is satisfied.

We then consider the constraints in (2). Considering the second constraint with commodity $(u,v) \in \mathcal{C}$ and link $(i,j) \in \mathcal{N} \times \mathcal{P}_{-v}$, we have that $f_{ij}^{uv} = f_{\phi(i)\phi(j)}^{\phi(u)\phi(v)*}$. Since node j is routing-incapable and $j \neq v$, node $\phi(j)$ is also routing-incapable and $\phi(j) \neq \phi(v)$ from Definition 1. It follows that $f^{\phi(u)\phi(v)*}_{\phi(i)\phi(j)}=0$ because the optimal share of commodity $(\phi(u),\phi(v))$ never routes traffic over link $(\phi(i),\phi(j))$ as $\phi(j)\in\mathcal{P}_{-\phi(v)}$. Therefore, we have $f^{uv}_{ij}=0$, and the second constraint in (2) is satisfied.

Considering the third constraint in (2) with commodity $(u,v)\in\mathcal{C}$ and node $i\in\mathcal{N}$, we have that $f^{uv}_{iu}=f^{\phi(u)\phi(v)*}_{\phi(i)\phi(u)}=0$ because the optimal share of commodity $(\phi(u),\phi(v))$ never routes traffic back to the traffic's source node. A similar argument can be applied to the constraint $f^{uv}_{vi}=0$ as the optimal share of commodity $(\phi(u),\phi(v))$ never routes traffic away from the traffic's destination node. Therefore, The third constraint in (2) is satisfied. For the first constraint in (2), the proof that the constraint is satisfied is similar to the proof in [12] and is omitted for brevity. Thus, it holds that $[f^{uv}_{ij}]\in\mathcal{F}$.

Altogether, the solution yields the same optimal objective cost and satisfies all constraints of the linear program in (7). It must be an optimal solution. ■

The implication of Lemma 2 is that we can construct multiple optimal solutions from an optimal solution and the automorphism set. Next, we use these optimal solutions to identify an optimal solution with repetitive variables.

Theorem 2. *There exists an automorphism-invariant solution* $\left(\hat{\eta},[\hat{\beta}^u_{ij}],[\hat{\gamma}^u_{ij}],[\hat{f}^{uv}_{ij}]\right)$ *that is optimal for the linear program in* (7) *and satisfies:*

$$\hat{f}^{uv}_{ij}=\hat{f}^{\phi(u)\phi(v)}_{\phi(i)\phi(j)},\ \hat{\beta}^u_{ij}=\hat{\beta}^{\phi(u)}_{\phi(i)\phi(j)},\ \hat{\gamma}^u_{ij}=\hat{\gamma}^{\phi(u)}_{\phi(i)\phi(j)},\ \forall\phi\in\Phi.$$

Proof: We first construct a solution before showing that it is optimal and satisfies the automorphism-invariant property. Let $\left(\hat{\eta},[\hat{\beta}^u_{ij}],[\hat{\gamma}^u_{ij}],[\hat{f}^{uv}_{ij}]\right)$ be a solution such that

$$\hat{\eta}=\eta^*,\qquad \hat{f}^{uv}_{ij}=\frac{1}{|\Phi|}\sum_{\phi\in\Phi}f^{\phi(u)\phi(v)*}_{\phi(i)\phi(j)},$$
$$\hat{\beta}^u_{ij}=\frac{1}{|\Phi|}\sum_{\phi\in\Phi}\beta^{\phi(u)*}_{\phi(i)\phi(j)},\qquad \hat{\gamma}^u_{ij}=\frac{1}{|\Phi|}\sum_{\phi\in\Phi}\gamma^{\phi(u)*}_{\phi(i)\phi(j)},$$

where $\left(\eta^*,[\beta^{u*}_{ij}],[\gamma^{u*}_{ij}],[f^{uv*}_{ij}]\right)$ is an existing optimal solution.

Since $\hat{\eta}=\eta^*$, the objective cost under the solution equals to the optimal cost. We then need to show that all constraints in (7) are satisfied.

The linearity in the construction of the solution implies that i) the domains of variables β^u_{ij} and γ^u_{ij} in (7), ii) the domains of variables f^{uv}_{ij} in (2), and iii) the second and third constraints in (2) are satisfied. The proof that the first constraint in (2) is satisfied is similar to the proof in [12] and is omitted for brevity. We next consider the other constraints in (7).

Considering the first constraint in (7) with link (i,j), we have that

$$\sum_{u\in\mathcal{H}}H_u\left[\frac{1}{|\Phi|}\sum_{\phi\in\Phi}\beta^{\phi(u)*}_{\phi(i)\phi(j)}+\frac{1}{|\Phi|}\sum_{\phi\in\Phi}\gamma^{\phi(u)*}_{\phi(i)\phi(j)}\right]$$
$$=\frac{1}{|\Phi|}\sum_{\phi\in\Phi}\sum_{u\in\mathcal{H}}H_{\phi(u)}\left(\beta^{\phi(u)*}_{\phi(i)\phi(j)}+\gamma^{\phi(u)*}_{\phi(i)\phi(j)}\right)\le\eta^*.$$

The first equality uses the node capacity property in Definition 1. The last inequality uses the fact that the inner summation is at most η^* since the optimal solution satisfies the constraint with link $(\phi(i),\phi(j))$. Therefore, the first constraint in (7) is satisfied.

Considering the second constraint in (7) with commodity (u,v) and link (i,j), we have that

$$\frac{1}{C_{ij}|\Phi|}\sum_{\phi\in\Phi}f^{\phi(u)\phi(v)*}_{\phi(i)\phi(j)}-\frac{1}{|\Phi|}\sum_{\phi\in\Phi}\beta^{\phi(u)*}_{\phi(i)\phi(j)}-\frac{1}{|\Phi|}\sum_{\phi\in\Phi}\gamma^{\phi(v)*}_{\phi(i)\phi(j)}$$
$$=\frac{1}{|\Phi|}\sum_{\phi\in\Phi}\left[\frac{f^{\phi(u)\phi(v)*}_{\phi(i)\phi(j)}}{C_{\phi(i)\phi(j)}}-\beta^{\phi(u)*}_{\phi(i)\phi(j)}-\gamma^{\phi(v)*}_{\phi(i)\phi(j)}\right]\le 0.$$

The first equality uses the link capacity property in Definition 1. The last inequality uses the fact that the expression in the summation is non-positive since the optimal solution satisfies the constraint with commodity $(\phi(u),\phi(v))$ and link $(\phi(i),\phi(j))$. Therefore, the second constraint in (7) is satisfied. Altogether, the solution is feasible and optimal.

Finally, we show the solution satisfies the automorphism-invariant property. We first consider $\hat{\beta}^u_{ij}$ and any $\phi\in\Phi$:

$$\hat{\beta}^{\phi(u)}_{\phi(i)\phi(j)}=\frac{1}{|\Phi|}\sum_{\phi'\in\Phi}\beta^{\phi'(\phi(u))*}_{\phi'(\phi(i))\phi'(\phi(j))}$$
$$=\frac{1}{|\Phi|}\sum_{\phi''\in\Phi}\beta^{\phi''(u)*}_{\phi''(i)\phi''(j)}=\hat{\beta}^u_{ij}.$$

The second equality used the fact that the set of all automorphism mapping is a group and the summation is over the entire set. Therefore, every variable $\hat{\beta}^u_{ij}$ is automorphism-invariant. Similar arguments prove the automorphism-invariant property of variables $\hat{f}^{uv}_{ij}$ and $\hat{\gamma}^u_{ij}$ and are omitted. Thus, the solution is automorphism-invariant. This proves the theorem. ■

The implication of Theorem 2 is that the linear program in (7) always has an optimal solution whose variables form groups. Every variable in each group takes the same value, i.e., $\hat{\beta}^u_{ij}=\beta^{\phi(u)}_{\phi(i)\phi(j)}$ for all $\phi\in\Phi$. We use this insight to formulate a new linear program targeting the automorphism-invariant optimal solution to improve scalability.

C. Representative variables

We first identify how variables $[\beta^u_{ij}],[\gamma^u_{ij}],[f^{uv}_{ij}]$ of the linear program in (7) form groups, so the variables in each group can be represented by a representative variable. The challenge of this identification is its combinatorial nature.

We observe from Theorem 2 that the variables $[\beta^u_{ij}]$ and $[\gamma^u_{ij}]$ share the same group relations as they share the same index set, $\mathcal{H}\times\mathcal{L}$. We therefore develop an efficient algorithm to group the variables based on the generator set $\hat{\Phi}$ as summarized in Algorithm 2. The algorithm searches over the index set. It takes an index from the set and utilizes the generator set to find all indices sharing the same group. Once a group is formed, the algorithm takes an unvisited index and continues the search process until all groups are formed. The time-complexity of Algorithm 1 is $O(|\mathcal{H}|\,|\mathcal{L}|\,|\hat{\Phi}|)$, since every index is visited once and $|\hat{\Phi}|$ indices are searched over per visited index. Using the generator set is more efficient than the entire automorphism set, which is exponentially large.

The algorithm outputs the representative index set $\hat{\mathcal{G}}$ and the dictionary ω containing automorphisms that can map each variable to its representative. In particular, let $\hat{\beta}_{ij}^{u}$ and $\hat{\gamma}_{ij}^{u}$ be representative variables for every $(u,i,j) \in \hat{\mathcal{G}}$. They represent variables $[\beta_{ij}^{u}]$ and $[\gamma_{ij}^{u}]$ of the linear program in (7) as follows:

$$\beta_{ij}^{u} \xRightarrow{\text{represented by}} \varphi\left[\beta_{ij}^{u}\right] = \hat{\beta}_{\phi(i)\phi(j)}^{\phi(u)} \quad \text{where } \phi = \omega\left[u,i,j\right]$$

$$\gamma_{ij}^{u} \xRightarrow{\text{represented by}} \varphi\left[\gamma_{ij}^{u}\right] = \hat{\gamma}_{\phi(i)\phi(j)}^{\phi(u)} \quad \text{where } \phi = \omega\left[u,i,j\right]$$

for every $(u,i,j) \in \mathcal{H} \times \mathcal{L}$. We use $\varphi[x]$ to denote the representative of x.

Algorithm 2: Identification of $[\hat{\beta}_{ij}^{u}]$ and $[\hat{\gamma}_{ij}^{u}]$

```
Initialize empty sets $\mathcal{Q}, \mathcal{Z}, \hat{\mathcal{G}}$
Initialize dictionaries $D, \omega$
for $(u,i,j) \in \mathcal{H} \times \mathcal{L}$ do
    if $(u,i,j) \in \mathcal{Z}$ then
        continue
    $\hat{\mathcal{G}} \leftarrow \hat{\mathcal{G}} \cup \{(u,i,j)\}$
    $D[u,i,j] \leftarrow \phi_{\text{identity}}$
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(u,i,j)\}$
    while $\mathcal{Q}$ is not empty do
        Pop $(a,b,c)$ from $\mathcal{Q}$
        $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{(a,b,c)\}$
        for $\phi \in \hat{\Phi}$ do
            if $(\phi(a), \phi(b), \phi(c)) \notin \mathcal{Z}$ then
                $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\phi(a), \phi(b), \phi(c))\}$
                $D[\phi(a), \phi(b), \phi(c)] \leftarrow \phi(D[a,b,c])$
for $(u,i,j) \in \mathcal{H} \times \mathcal{L}$ do
    $\omega[u,i,j] \leftarrow (D[u,i,j])^{-1}$
return representative index set $\hat{\mathcal{G}}$ and dictionary $\omega$
```

Algorithm 3: Construction of index set $\hat{\mathcal{M}}$

```
Initialize empty sets $\mathcal{Z}, \hat{\mathcal{M}}$
for $(i,j) \in \mathcal{L}$ do
    Let $e = (a_{uij}, b_{uij})_{(u,i,j) \in \hat{\mathcal{G}}}$ where $a_{uij}$ and $b_{uij}$
     are respectively the counts of representative $\hat{\beta}_{ij}^{u}$
     and $\hat{\gamma}_{ij}^{u}$ in $\sum_{n \in \mathcal{H}} H_n \left(\hat{\beta}_{ij}^{n} + \hat{\gamma}_{ij}^{n}\right)$
    if $e \in \mathcal{Z}$ then
        continue
    $\hat{\mathcal{M}} \leftarrow \hat{\mathcal{M}} \cup \{(i,j)\}$
    $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{e\}$
return representative constraint indices $\hat{\mathcal{M}}$
```

For the share variables $[f_{ij}^{uv}]$, their groups can be obtained from Algorithms 2 and 3 in [12], whose multi-commodity formulation includes a related conservation constraint similar to ours in (2). Algorithm 2 in [12] outputs the set of representative commodities, $\hat{\mathcal{C}}$, and a dictionary π containing automorphisms that can map each commodity to its representative commodity. For each representative commodity $(u,v) \in \hat{\mathcal{C}}$, Algorithm 3 in [12] outputs the set of representative links of share variables, $\hat{\mathcal{L}}^{uv}$, and a dictionary σ^{uv} containing automorphisms that can map each share variable to its representative. In particular, let $\hat{f}_{ij}^{uv}$ be a representative share variable for every $(u,v) \in \hat{\mathcal{C}}$ and every $(i,j) \in \hat{\mathcal{L}}^{uv}$. They represent shares $[f_{ij}^{uv}]$ of the linear program in (7) as follows:

$$f_{ij}^{uv} \xRightarrow{\text{represented by}} \varphi\left[f_{ij}^{uv}\right] = \hat{f}_{\phi'(\phi(i))\phi'(\phi(j))}^{\phi(u)\phi(v)},$$

where $\phi = \pi\left[u,v\right]$, and $\phi' = \sigma^{\phi(u)\phi(v)}\left[i,j\right]$ for every $(u,v,i,j) \in \mathcal{C} \times \mathcal{L}$.

These representatives $[\hat{\beta}_{ij}^{u}], [\hat{\gamma}_{ij}^{u}], [\hat{f}_{ij}^{uv}]$ significantly reduce the variables of the linear program in (7), resulting in a scalable linear program.

D. Automorphism-invariant formulation

Finally, we formulate the scalable linear program for large networks as follows:

$$\begin{aligned}
\text{Minimize} \quad & \eta \\
\text{Subject to} \quad & \sum_{u \in \mathcal{H}} H_u \left(\varphi\left[\beta_{ij}^{u}\right] + \varphi\left[\gamma_{ij}^{u}\right]\right) \le \eta \ , \forall (i,j) \in \hat{\mathcal{M}} \\
& \frac{\hat{f}_{ij}^{uv}}{C_{ij}} - \varphi\left[\beta_{ij}^{u}\right] - \varphi\left[\gamma_{ij}^{v}\right] \le 0 \\
& \qquad\qquad , \forall (u,v) \in \hat{\mathcal{C}}, \forall (i,j) \in \hat{\mathcal{L}}^{uv} \\
& \hat{\beta}_{ij}^{u} \in \mathbb{R}_{+}, \ \hat{\gamma}_{ij}^{u} \in \mathbb{R}_{+}, \forall (u,i,j) \in \hat{\mathcal{G}} \\
& [\hat{f}_{ij}^{uv}] \in \hat{\mathcal{F}},
\end{aligned} \tag{8}$$

where the set $\hat{\mathcal{F}}$ is defined in (9).

$$\hat{\mathcal{F}} = \left\{ \begin{array}{l}
\sum_{j \in \mathcal{O}(i)} \hat{f}_{ij}^{uv} - \sum_{j \in \mathcal{I}(i)} \hat{f}_{ji}^{uv} = \mathbb{I}\left[i=u\right] - \mathbb{I}\left[i=v\right] \\
\qquad\qquad\qquad , \forall (u,v) \in \hat{\mathcal{C}}, \forall i \in \mathcal{N} \\
\hat{f}_{ij}^{uv} = 0 \qquad\quad , \forall (u,v) \in \hat{\mathcal{C}}, \forall (i,j) \in \hat{\mathcal{L}}^{uv} \cap (\mathcal{N} \times \mathcal{P}_{-v}) \\
\hat{f}_{iu}^{uv} = \hat{f}_{vi}^{uv} = 0, \forall (u,v) \in \hat{\mathcal{C}}, \forall (i,u), (v,i) \in \hat{\mathcal{L}}^{uv} \\
\hat{f}_{ij}^{uv} \in \mathbb{R}_{+} \qquad , \forall (u,v) \in \hat{\mathcal{C}}, \forall (i,j) \in \hat{\mathcal{L}}^{uv}
\end{array} \right\} \tag{9}$$

The set $\hat{\mathcal{M}}$ in (8) is constructed by Algorithm 3 to remove unnecessary constraints, causing intractability for large networks.

The scalable linear program in (8) designs an optimal oblivious routing solution in polynomial time for large networks. It is faster and more scalable than the state-of-the-art technique [12] as shown in Section VI. The next section describes how an optimal solution is further optimized for practical deployment.

V. Compact Forwarding Rules

An optimal oblivious routing solution, obtained from the scalable formulation in (8), could be distributively deployed on switches in a datacenter network. Every switch decides how

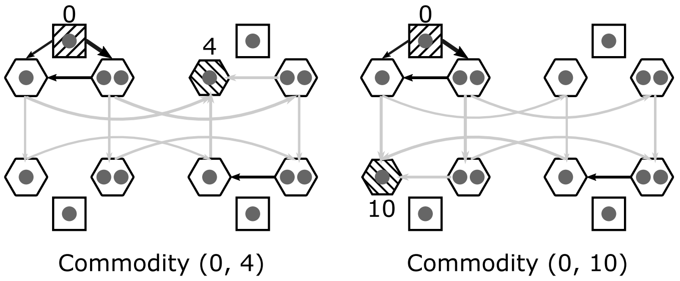

Fig. 5. Optimal oblivious routing solution of commodities $(0, 4)$ and $(0, 10)$, which have the same representative, is presented. The black arrows represent similar split weights that can be grouped at each corresponding node.

traffic is split over next-hop switches for each commodity.[2] In particular, switch i splits the traffic of commodity (u, v) to next-hop switch j over link (i, j) with weight

$$w_{ij}^{uv} = \frac{f_{ij}^{uv}}{\sum_{j' \in \mathcal{O}(i)} f_{ij'}^{uv}} \quad , \forall (u, v) \in \mathcal{C}, \forall i \in \mathcal{N}, \forall j \in \mathcal{O}(i). \tag{10}$$

These weights could be configured into programmable switches in the form of forwarding rules [18], [27], [28]. However, a switch has limited memory for storing the rules from multiple commodities in a large datacenter network. This practical limitation could hinder the deployment of the optimal oblivious routing solution.

We observe that some rules, derived from the commodities having the same representative, are similar as shown in Figure 5. We use this insight to group the forwarding rules in order to reduce memory requirement. Algorithm 4 summarized our grouping method. For each node in a network, the commodities having the same representative are grouped by the similarity of split weights, i.e., $(i, j, \varphi\left[f_{ij}^{xy}\right])_{(i,j)\in\mathcal{O}(n)}$ (ignoring the common denominator in (10)). The commodities having similar split weights form a group, so one collection of forwarding rules is sufficient for these commodities. The algorithm outputs the collection of grouped rules, $\{W_n\}_{n\in\mathcal{N}}$. For node n, each collection of grouped rules is stored in the dictionary W_n whose key e represents a collection of split weights and the value $W_n[e]$ is the set of commodities using the weights. Note that Algorithm 4 is highly parallelizable as the grouping process can be parallelized for each node.

In short, Algorithm 4 utilizes the repeated structures in the optimal oblivious routing solution, which is automorphism invariant, to reduce memory requirement for the deployment of forwarding rules in real-world switches. In addition, if the reduced requirement exceeds the available memory of a switch, an approximation technique [18] could be applied to our grouped rules to trade-off between optimality and available memory. In other words, our grouping method circumvents the unnecessary trade-off when it is avoidable.

VI. Evaluation

Our scalable linear program in (8) is evaluated over various datacenter network topologies and sizes. Its scalability is evaluated in terms of computation times and problem sizes. The efficiency of our grouping method in Algorithm 4 is evaluated for the same topologies. Ultimately, we show the applicability of our work for an existing server-centric topology.

[2]TCP reordering effect can be avoided by splitting traffic based on flows instead of packets.

Algorithm 4: Forward-rule grouping

```
Initialize empty set $\mathcal{Q}$
Initialize dictionary $\{W_n\}_{n\in\mathcal{N}}$
for $n \in \mathcal{N}$ do
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{n\}$
    for $(u, v) \in \hat{\mathcal{C}}$ do
        for $(x, y)$ represented by $(u, v)$ do
            Let $e = (i, j, \varphi\left[f_{ij}^{xy}\right])_{(i,j)\in\mathcal{O}(n)}$
            if $e \notin W_n$ then
                $W_n[e] \leftarrow \emptyset$
            $W_n[e] \leftarrow W_n[e] \cup \{(x, y)\}$
return Grouped forwarding rules $\{W_n\}_{n\in\mathcal{N}}$
```

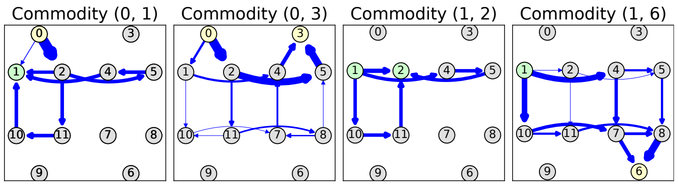

Fig. 6. The optimal oblivious routing solution of the network in Figure 2. The four commodities correspond to all combinations of node types.

Every evaluation is executed on a commodity computer with an Intel Core i9-12900K processor and 128GB memory. All linear programs are solved by Gurobi [29]. We use nauty [26] to compute generator sets.

A. Topology setting and brief background

We motivate the topologies used in our evaluation as follows. FatClique is designed for low-cost manageability, regarding topology deployment and expansion [2]. SlimFly focuses on throughput performance by the use of low diameter graphs with high-radix switches [9]. BCube aims for shipping-container-based datacenters [10]. FatClique and SlimFly follow the server-switch architecture, while BCube belongs to the server-centric architecture.

These topologies are constructed as follows. For FatClique, we set the numbers of switches in a sub-block, sub-blocks in a block, and blocks to an identical value. We vary this value from 2 to 12. For SlimFly, we use its provided topologies with the number of network radices ranging from 5 to 43. For BCube, each topology is built from 4-port switches, and we vary the number of levels of switches from 2 to 5. The maximum number of nodes in the topologies generated by these settings is 2304.

B. Correctness of optimal solutions

In every evaluation, we confirm the correctness of the optimal solutions obtained from (8) by comparing them with the solutions from the state-of-the-art technique in [12]. When

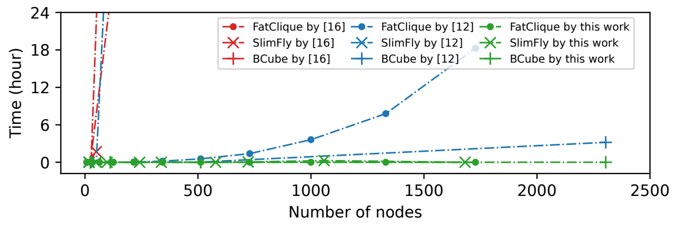

Fig. 7. The optimization time at different sizes of FatClique, SlimFly and BCube. The maximum times of the scalable formulation are 0.008 sec. for FatClique, 2.97 min. for SlimFly, and 0.026 sec. for BCube. Note that SlimFly topologies beyond 50 nodes cannot be solved by [16] due to insufficient computing memory.

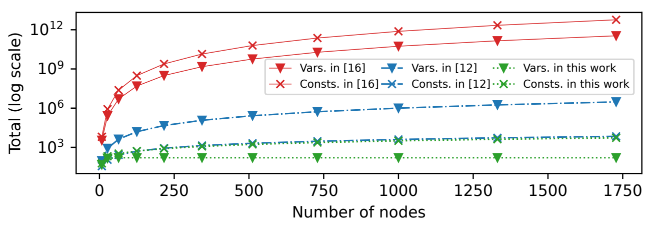

Fig. 8. The numbers of variables and constraints at various sizes of FatClique. For the largest FatClique, the numbers of variables and constraints in the scalable formulation are 160 and 5326 respectively.

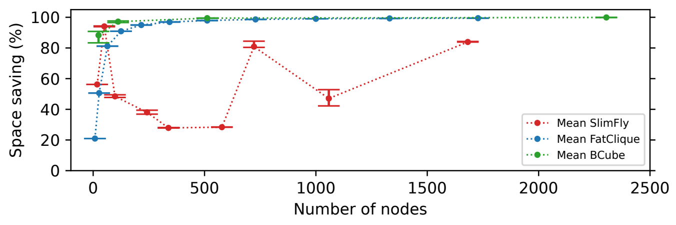

Fig. 9. The space saving (percentage) at different sizes of FatClique, SlimFly, and BCube. Each mean value is computed from all nodes in a topology. The maximum and minimum values are represented by horizontal bars.

the technique fails to compute solutions, we adopt another technique in [30] to validate our solutions.

Moreover, as shown in Figure 6, the scalable linear program in (8) handles routing-incapable nodes correctly for the simple network in Figure 2. Every routing-incapable node that is unrelated to a considered commodity does not involve in the commodity's routing.

C. Scalability in terms of computation time

We vary the sizes of FatClique, SlimFly, and BCube. We record the optimization times of our scalable formulation in (8), the state-of-the-art technique in [12], and the other linear program in [16]. The limit of the optimization time is set to 24 hours. The results are plotted in Figure 7. Our work takes much lesser time to find optimal routing solutions than the other techniques for all topologies and all sizes. The work in [12] scales well for FatClique and BCube but fails to obtain optimal routing solutions within the time limit for SlimFly beyond 98 nodes. The linear program in [16] cannot scale beyond 112 nodes for all considered topologies due to insufficient computing memory.

D. Scalability in terms of variables and constraints

Figure 8 shows the numbers of variables and constraints at different sizes of FatClique from 8 to 1728 nodes, which follows the same evaluation in [12]. Our scalable formulation in (8) is much leaner than the other techniques regarding the numbers of variables and constraints. While the linear program in [16] can obtain an optimal routing solution in polynomial time, its formulation size grows exponentially as the topology size increases, resulting in the insufficient memory issue in Section VI-C. Since the state-of-the-art technique in [12] iteratively solves two linear programs, we only count the total numbers of variables and constraints from the two programs at the first iteration without additional constraints from later iterations. Yet, its formulation size is still larger than ours.

E. Reduction of forwarding rules

The efficiency of the forwarding rule grouping in Algorithm 4 is measured by the percentage of space saving, which is the proportion of rule reduction to non-grouped rules. Figure 9 shows the space saving at various sizes of FatClique, SlimFly, and BCube. Our grouping method reduces more than 90% of the non-grouped forwarding rules under FatClique with no less than 216 nodes and under BCube with no less than 112 nodes. The space saving for SlimFly highly depends on topology configuration.

F. Possible application for BCube

BCube employs dynamic source routing to utilize multipath capacity. Each source selects the best-quality path from its candidate paths, based on the current maximum available bandwidth. This approach constantly measures available bandwidth, introduces complexity, and may need specialized hardware and network stack. Instead, one could employ the oblivious routing approach on BCube for simpler production.

Table I shows the results of our optimal routing solution and equal split, which is equivalent to a uniform selection of paths in BCube's dynamic routing. The optimal solution gains 1.8×-6.7× congestion improvement over the equal split.

VII. Conclusion

This paper proposes the polynomial-time process for designing optimal compact oblivious routing for datacenter networks. In the process, an optimal oblivious routing solution is designed by solving a scalable linear program, derived from the transformation of a robust optimization problem and the exploitation of repeated network structures. After obtaining an optimal routing solution, the process compacts the forwarding rules, converted from the optimal solution, in order to reduce memory requirement for real-world deployment.

Potential future work includes topology asymmetry, network failure, and prior traffic distribution.

The authors have provided public access to their code at https://github.com/NDS-VISTEC/DCN-Oblivious-Routing.

TABLE I
THE MAXIMUM CONGESTION RATIO IN BCUBE WITH OBLIVIOUS ROUTING

BCube	Maximum congestion ratio		
#Nodes	This work	Equal split	Improvement
24	2.50	4.50	1.80x
112	4.00	11.49	2.87x
512	5.50	21.50	3.91x
2304	6.97	47.19	6.77x

REFERENCES

[1] 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

[2] 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

[3] 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

[4] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, 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

[5] 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/

[6] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: 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

[7] A. Valadarsky, G. Shahaf, M. Dinitz, and M. Schapira, "Xpander: Towards optimal-performance datacenters," in *Proceedings of the 12th International on 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

[8] 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

[9] 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.

[10] 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/

[11] 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/dcell-a-scalable-and-fault-tolerant-network-structure-for-data-centers/

[12] 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.

[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, and S. Sengupta, "Traffic-oblivious routing in the hose model," *IEEE/ACM Transactions on Networking*, vol. 19, no. 3, pp. 774–787, 2011.

[16] 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

[17] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Racke, "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

[18] N. Kang, M. Ghobadi, J. Reumann, A. Shraer, and J. Rexford, "Efficient traffic splitting on commodity switches," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2716281.2836091

[19] 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

[20] 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.

[21] 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

[22] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[23] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, ser. STOC '84. New York, NY, USA: Association for Computing Machinery, 1984, p. 302–311. [Online]. Available: https://doi.org/10.1145/800057.808695

[24] M. B. Cohen, Y. T. Lee, and Z. Song, "Solving linear programs in the current matrix multiplication time," in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 938–942. [Online]. Available: https://doi.org/10.1145/3313276.3316303

[25] C. Godsil and G. F. Royle, *Algebraic Graph Theory*, ser. Graduate Texts in Mathematics. Springer, 2001, no. Book 207.

[26] 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

[27] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, p. 69–74, mar 2008. [Online]. Available: https://doi.org/10.1145/1355734.1355746

[28] 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

[29] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2021. [Online]. Available: https://www.gurobi.com

[30] B. Towles and W. Dally, "Worst-case traffic for oblivious routing functions," *IEEE Computer Architecture Letters*, vol. 1, no. 1, pp. 4–4, 2002.