Reducing Reconfiguration Time in Hybrid Optical-Electrical Datacenter Networks

Shuyuan Zhang
Shanghai Jiao Tong University
Shanghai, China
zhang-shuyuan@sjtu.edu.cn

Shu Shan
Shanghai Jiao Tong University
Shanghai, China
shan.shu@sjtu.edu.cn

Shizhen Zhao*
Shanghai Jiao Tong University
Shanghai, China
shizhenzhao@sjtu.edu.cn

ABSTRACT

We study how to reduce the reconfiguration time in hybrid optical-electrical Datacenter Networks (DCNs). With a layer of Optical Circuit Switches (OCSes), hybrid optical-electrical DCNs could reconfigure their logical topologies to better match the on-going traffic patterns, but the reconfiguration time could directly affect the benefits of reconfigurability. The reconfiguration time consists of the topology solver running time and the network convergence time after triggering reconfiguration. However, existing topology solvers either incur high algorithmic complexity or fail to minimize the reconfiguration overhead.

In this paper, we propose a novel algorithm that combines the ideas of bipartition and Minimum Cost Flow (MCF) to reduce the overall reconfiguration time. For the first time, we formulate the topology solving problem as an MCF problem with piecewise cost, which strikes a better balance between solver complexity and solution optimality. Our evaluation shows that our algorithm can significantly reduce the network convergence time while consuming less topology solver running time, making its overall performance superior to existing algorithms. Our code and test cases are available at a public repository [25].

CCS CONCEPTS

• **Networks** → **Control path algorithms**; *Data center networks*; *Logical / virtual topologies.*

KEYWORDS

Datacenter network, Topology reconfiguration time, Optical circuit switches, Minimum cost flow.

ACM Reference Format:
Shuyuan Zhang, Shu Shan, and Shizhen Zhao. 2023. Reducing Reconfiguration Time in Hybrid Optical-Electrical Datacenter Networks. In *7th Asia-Pacific Workshop on Networking (APNET 2023), June 29–30, 2023, Hong Kong, China.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3600061.3600071

*Corresponding author.

APNET 2023, June 29–30, 2023, Hong Kong, China

ACM ISBN 979-8-4007-0782-7/23/06.
https://doi.org/10.1145/3600061.3600071

1 INTRODUCTION

The interest in hybrid optical-electrical Datacenter Networks (DCNs) has been growing as it offers the capability of performing traffic-aware topology designs. The dominant network topology utilized in current DCNs is still the Clos topology [1, 12, 21]. It provides universal bandwidth to arbitrary traffic patterns, but can be highly cost-suboptimal for bandwidth provision. As network bandwidth keeps increasing, building Clos networks is becoming cost-prohibitive [7]. The traffic in DCNs is highly skewed and time-varying [4]. By adapting the network topology to the traffic patterns, the performance-cost ratio can be improved. This necessitates a reconfigurable topology, which can be achieved by introducing novel optical network components [10] such as Optical Circuit Switches (OCSes), free-space optics, wireless radios, etc.

We study how to reduce the reconfiguration time in hybrid optical-electrical DCNs. When the DCN traffic pattern changes, we may need to compute a new topology for this traffic pattern, and then reconfigure the DCN from its old topology to this new topology. The reconfiguration time consists of the topology solver running time and the network convergence time after triggering reconfiguration. The former one depends on the algorithmic complexity and the latter one depends on the number of links to be changed during the reconfiguration process[1]. Therefore, we need a topology solver with low algorithmic complexity, while being able to reduce the number of reconfigured links.

Existing algorithms are limited because they either incur high topology solver running time or yield a highly suboptimal solution that requires many link reconfigurations. The topology optimization problem can be formulated as an Integer Linear Programming (ILP) problem with the objective to minimize the total number of reconfigured links. However, due to the NP-hardness of most ILP problems, even the most advanced commercial ILP solver cannot solve the problem directly at a real-world scale [26]. To reduce the algorithmic complexity, some algorithms take advantage of the homogeneity of the DCN physical topologies. Zhao et al. [26] present a greedy minimal-rewiring algorithm using Minimum Cost Flow (MCF), but experimental results show that the total number of rewires can be far from optimal. On the other hand, Google's patent [17] presents an algorithm that utilizes the idea of bipartition and achieves a lower number of rewires than the MCF-based algorithm, but it is still based on ILP and can be very slow in practice.

In this paper, we propose an algorithm that combines the advantages of MCF and bipartition to reduce the total number of rewires, while ensuring a polynomial running time. The standard form of

[1]We must ensure uninterrupted service during reconfiguration. If the change between the old topology and the new topology is too large, the reconfiguration process need to be preformed in several steps [20], which increases the total network convergence time.

an MCF problem has a linear cost for each link. We note that the cost function of each link can be generalized to a convex piecewise linear function. This observation allows us to develop a polynomial algorithm with the minimum number of rewires for the case where there are two OCSes. We then generalize this algorithm to the n-OCS cases using an recursive bipartition approach. Our evaluation shows that our algorithm exhibits very low algorithmic complexity, while achieving a low rewiring ratio at the same time. Our code and test cases are available at a public repository [25].

The structure of this paper is presented as follows. Section 2 outlines the limitations of existing algorithms and highlights the importance of improving algorithm performance. In Section 3, we formulate our model and outline the problem addressed. The description and analysis of our algorithm are presented in Section 4. The performance of our algorithm is demonstrated through evaluations in Section 5. A discussion of related work is provided in Section 6, and the paper concludes with our final thoughts and proposed future work in Section 7.

2 MOTIVATION

2.1 Limitations of Existing Algorithms

As our problem can be formulated as an ILP problem, the most straightforward approach is to use a general ILP solver to directly solve the problem. We refer to this as the Brute Force Algorithm. When the physical topology is *uniform* (See Definition 3.1), two more practical algorithms have been proposed in the literature. Specifically, Zhao et al. [26] propose a greedy algorithm that utilizes MCF algorithms, while Google's patent [17] presents a bipartition algorithm. We refer to these algorithms as the Greedy MCF Algorithm and the Bipartition Algorithm, respectively.

Brute Force Algorithm. Although this method is optimal, it is too slow for our application. Our evaluations have shown that even the most advanced commercial ILP solver, such as Gurobi [13], cannot solve a small cluster having 4 OCSes within the time limit of 1 minute. In fact, in many cases, the solver fails to find a feasible solution to our problem, because it does not exploit the special structure of our problem.

Greedy MCF Algorithm. As reported in [26], although this method is fast, it has a poor approximation ratio. In the test cases presented in the literature, the optimal reconfiguration ratio is below 10%, whereas the reconfiguration ratio achieved by the algorithm is around 20%. A possibility is that this algorithm focuses too much on local optimal solutions at each step, losing sight of potential future improvements.

Bipartition Algorithm. The bipartition of the algorithm requires solving a subproblem, but the details of the solving process are not provided in the original literature. It is only mentioned that an ILP problem is solved. As a result, it cannot be guaranteed that this method can be solved in polynomial time. In our evaluation, we have found that this algorithm provides a better approximation ratio than the Greedy MCF Algorithm, but it is significantly slower.

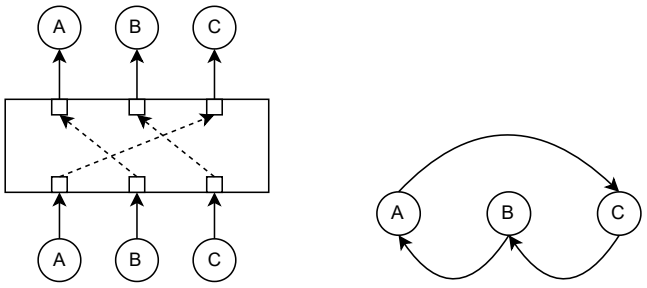
2.2 The Importance of Reducing Reconfiguration Time

In this section, we highlight the importance of reducing reconfiguration time in hybrid optical-electrical DCNs.

Effectiveness of computed optimal topology. Reducing reconfiguration time is crucial to ensure that the computed optimal topology is effective. The primary purpose of reconfigurable topology is to allow the topology to adapt to the changing traffic pattern. The target topology is determined at the start of the reconfiguration. If the reconfiguration time is too long, the target topology may not be able to reflect the current traffic pattern, which can affect the performance of the topology.

Frequency of reconfiguration. Reducing reconfiguration time is crucial to increasing the reconfiguration frequency, which may improve the performance of reconfigurable DCNs. Google has announced its use of MEMS-based OCSes in the new generation of Jupiter [20]. Currently its reconfiguration takes hours to finish, while the reconfiguration time of MEMS-based OCSes is merely tens or hundreds of milliseconds [22]. There is potential to increase the reconfiguration frequency and previous literature has shown that more frequent reconfiguration may improve the performance of networks [23].

3 PROBLEM



(a) Physical topology and the matching of the OCS.

(b) Corresponding logical topology.

Figure 1: An example of the effect of OCSes. There are 3 nodes connecting to a 3-port OCS. The dashed lines represent the reconfigurable matching in the OCS.

An OCS possesses many input and output ports that can be interconnected with (electrical) switches. A complete matching between the input and output ports can be configured within the OCS. The physical connections between the OCS and the switches are referred to as the **physical topology**, while the corresponding equivalent topology in the absence of the OCS is referred to as the **logical topology**. An example of this is shown in Figure 1.

Remark. Our proposed model is not limited to OCSes, but can be extended to encompass other components that serve similar functions, such as patch panels.

Consider a flat topology comprising of m Top-of-Rack (ToR) switches and n OCSes, where the uplinks of the ToR switches are only connected to the OCSes. The physical topology of the network is characterized by two key parameters, denoted as $a \in \mathbb{Z}_{\geq 0}^{m \times n}$ and $b \in \mathbb{Z}_{\geq 0}^{m \times n}$. Here, a_{jk} represents the number of connections *from* the k-th OCS *to* the j-th switch, and b_{ik} signifies the number of connections *from* the i-th switch *to* the k-th OCS. Note that the forwarding of OCSes is directional, so it may happen that $a_{jk} \neq$

b_{ik} for some i, j, k. The logical topology is characterized by $c \in \mathbb{Z}_{\geq 0}^{m\times m}$. c_{ij} is the number of equivalent connections from the i-th switch to the j-th switch. For convenience, we define index sets $I = \{1, 2, \ldots, m\}$, $J = \{1, 2, \ldots, m\}$ and $K = \{1, 2, \ldots, n\}$.

Remark. We adopt a flat topology solely for the purpose of illustrating our approach, owing to its simplicity and ease of comprehension. Nevertheless, our methodology can be effortlessly extended to accommodate diverse network topologies. To illustrate, consider a 2-layer Clos topology wherein the two switch layers are treated as a single layer, and our model is readily applicable.

The matching of all OCSes can be represented by $x \in \mathbb{Z}_{\geq 0}^{m\times m\times n}$. x_{ijk} is the number of equivalent connections from the i-th switch to the j-th switch established by the forwarding of the k-th OCS. Given the physical and logical topology, a feasible matching should satisfy the following constraints.

$$\sum_{i\in I} x_{ijk} = a_{jk}, \quad \forall j \in J, k \in K \tag{1}$$

$$\sum_{j\in J} x_{ijk} = b_{ik}, \quad \forall i \in I, k \in K \tag{2}$$

$$\sum_{k\in K} x_{ijk} = c_{ij}, \quad \forall i \in I, j \in J \tag{3}$$

Constraints (1) and (2) are derived from the limit of the physical topology, and constraint (3) is derived from the limit of the logical topology. We define the set of all feasible matchings as $S(a, b, c)$. In the rest of the paper, we use $u \in S(\alpha, \beta, c')$ and $x \in S(a, b, c)$ to represent the old and new matching of OCSes, respectively. Here u is a known parameter, while x is the decision variables. Since physical topology seldom changes, we assume that $\alpha = a$ and $\beta = b$. This means a and b are determined by u, so we only need two parameters to determine our problem, c and u.

In order to reduce the network convergence time, we need to choose an objective function that properly reflects such time. One possible choice is to use the minimum number of disconnections $\sum_{i\in I}\sum_{j\in J}\sum_{k\in K}(u_{ijk} - x_{ijk})^+$, where $x^+ := \max\{x, 0\}$ [26]. Therefore, our target is to solve the following optimization problem, denoted by $\mathrm{OPT}(c, u)$.

$$\begin{aligned} \min_x \quad & \sum_{i\in I}\sum_{j\in J}\sum_{k\in K}(u_{ijk} - x_{ijk})^+ \\ \text{s.t.} \quad & x \in S(a, b, c) \end{aligned}$$

Remark. There is potential to refine the objective function. It should be noted that connections between switches and OCSes can be regarded to be weighted. For instance, the cost of disconnecting a link may relate to the volume of traffic on it. If $u_{ijk} = 2$, and we assign weights of 1 and 2 to the two existing links, the updated objective function of them can be expressed as

$$f_0(x_{ijk}) = \begin{cases} -2x_{ijk} + 3, & 0 \leq x_{ijk} < 1, \\ -x_{ijk} + 2, & 1 \leq x_{ijk} < 2, \\ 0, & x_{ijk} \geq 2, \end{cases} \tag{4}$$

which is a convex piecewise-linear function. In fact, our proposed algorithm still works provided that the objective function can be expressed as $\sum_{i\in I}\sum_{j\in J}\sum_{k\in K} f_{ijk}(x_{ijk})$, where all $f_{ijk}(\cdot)$ are convex and piecewise-linear. The function $\sum_{i\in I}\sum_{j\in J}\sum_{k\in K}(u_{ijk} - x_{ijk})^+$ introduced earlier serves as a particular example of this class of functions. For the sake of simplicity, we continue to use it as our objective function throughout the rest of the paper.

While this problem may seem easy, it turns out to be quite challenging. The following facts provide supporting evidence.

- It is NP-complete to determine whether $S(a, b, c)$ is empty, even when $a_{jk}, b_{ik}, c_{ij} \in \{0, 1\}, \forall i \in I, j \in J, k \in K$ [14].
- Under the assumption $P \neq NP$, there is an $\epsilon > 0$ s.t. there is no polynomial-time n^ϵ-approximation algorithm for problem (5), which means that it is highly inapproximable. In particular, there is no constant ratio approximation of problem (5) [9]. Even when $a_{jk} = b_{ik} = c_{ij} = 1$, $p_{ijk} \in \{0, 1\}$, $\forall i \in I, j \in J, k \in K$, problem (5) is still NP-hard [11].

$$\begin{aligned} \min_x \quad & \sum_{i\in I}\sum_{j\in J}\sum_{k\in K} p_{ijk}x_{ijk} \\ \text{s.t.} \quad & x \in S(a, b, c) \\ & m = n \end{aligned} \tag{5}$$

The problem is challenging partly because the physical topology is arbitrary. However, since the physical topology is designed by us, we can impose certain constraints on it to make the problem easier to solve. One such constraint is as follows.

Definition 3.1. A physical topology is defined to be *proportional*, if there exist $r \in \mathbb{Z}_{>0}^n$ and $\alpha, \beta \in \mathbb{Z}_{>0}^m$, s.t. $a_{jk} = r_k\alpha_j, \forall j \in J, k \in K$ and $b_{ik} = r_k\beta_i, \forall i \in I, k \in K$. Specifically, we say it is *uniform* if $r_k = 1, \forall k \in K$.

The constraints of uniform physical topology have been presented in previous literature [26]. In this paper we relax this constraint slightly, allowing the physical topology to be proportional.

4 ALGORITHM DESIGN

4.1 An Exact Polynomial-Time Algorithm for a Special Case

Our algorithm is inspired by the fact that when $n = 2$, $\mathrm{OPT}(c, u)$ can be exactly solved in polynomial time.

When $n = 2$, constraint (3) becomes $x_{ij1} + x_{ij2} = c_{ij}$, so we can rewrite the objective function and other constraints using $x_{ij2} = c_{ij} - x_{ij1}$ to obtain the following problem.

$$\begin{aligned} \min_{x_{ij1}} \quad & \sum_{i\in I}\sum_{j\in J}[(u_{ij1} - x_{ij1})^+ + (u_{ij2} - c_{ij} + x_{ij1})^+] \\ \text{s.t.} \quad & \sum_{i\in I} x_{ij1} = a_{j1}, \quad \forall j \in J && \text{(6a)} \\ & \sum_{j\in J} x_{ij1} = b_{i1}, \quad \forall i \in I && \text{(6b)} \\ & x_{ij1} \leq c_{ij}, \quad \forall i \in I, j \in J && \text{(6c)} \\ & \sum_{i\in I}(c_{ij} - x_{ij1}) = a_{j2}, \quad \forall j \in J && \text{(6d)} \\ & \sum_{j\in J}(c_{ij} - x_{ij1}) = b_{i2}, \quad \forall i \in I && \text{(6e)} \end{aligned}$$

Constraints (6d) and (6e) are redundant because if $a_{j1} + a_{j2} \neq \sum_{i\in I} c_{ij}$ for some $j \in J$ or $b_{i1} + b_{i2} \neq \sum_{j\in J} c_{ij}$ for some $i \in I$, then $S(a, b, c)$ is empty. Otherwise, constraints (6d) and (6e) can be

derived from constraints (6a) and (6b). In the following discussion we will ignore constraints (6d) and (6e).

We can solve it using MCF algorithms. Integral MCF problem is defined as follows. In a directed multigraph, each node may have $s \in \mathbb{Z}$ units of supply. We say that a node has d units of demand if it has $-d$ units of supply. There are flows on the arcs of the graph. The sizes of flows are non-negative integers. The flows satisfy the constraint that the amount of flow entering a node, including the supply of the node, is equal to that leaving the node. Each arc have a cost for each unit of flow and a capacity. The total amount of flows that pass an arc cannot be greater than the capacity of the arc. The goal is to find a flow allocation that satisfies the constraints and has the smallest total cost.

The problem is thus equivalent to the following MCF problem. There are m supply nodes $\{s_1, s_2, \ldots, s_m\}$ and m demand nodes $\{d_1, d_2, \ldots, d_m\}$. The supply node s_i has b_{i1} units of supply, and the demand node d_j has a_{j1} units of demand. This setting models the constraints (6a) and (6b). For each pair of (s_i, d_j), consider the function

$$f_{ij}(x) = (u_{ij1} - x)^+ + (u_{ij2} - c_{ij} + x)^+, \quad x \in [0, c_{ij}].$$

This is a convex piecewise-linear function. Assume that it has q noncontinuous points $\{x_1, x_2, \ldots, x_q\}$ and define $x_0 = 0$, $x_{q+1} = c_{ij}$. Assume that on $[x_{p-1}, x_p]$ the slope of $f_{ij}(\cdot)$ is γ_p. Then we add $q+1$ arcs from s_i to d_j. For the p-th arc, the cost is γ_p and the capacity is $x_p - x_{p-1}$. This models the objective function and constraint (6c).

Integral MCF problem is a special ILP that can be solved in polynomial time, and in practice the solving is usually fast [24], so when $n = 2$, our problem can be solved efficiently.

4.2 The General Algorithm

For general cases where $n > 2$, we can merge some OCSes to be a larger OCS so that the physical topology can be seen as if it has only 2 OCSes. The merging is an approximation because it widens the range of reconfiguration. Then we can solve the approximated problem using the algorithm in Section 4.1. To obtain a real feasible solution, we then need to decompose the solution on each imaginary OCS, which requires solving two subproblems recursively. The formal pseudocode of this process is shown in Algorithm 1.

Now we focus on the correctness of our algorithm. If the function call SOLVE($c^{\text{Root}}, u^{\text{Root}}$) successfully returns, it is obvious that the return value is a feasible solution of OPT($c^{\text{Root}}, u^{\text{Root}}$). However, it is not trivial that the function call will return. The tricky part is that the OPT(c, u') in line 8 is not necessarily feasible. Here we prove that OPT(c, u') is always feasible.

THEOREM 4.1. *The function call* SOLVE($c^{\text{Root}}, u^{\text{Root}}$) *will always return successfully.*

PROOF. To avoid confusion, here we temporarily rename the OPT(c, u') in line 8 to OPT($c^{\text{Sub}}, u^{\text{Sub}}$). Obviously, there exist K_1, K_2, s.t. $\{K_1, K_2\}$ is a non-empty bipartition of some subset of K, and

[2] $\{S_1, S_2\}$ is defined to be a non-empty bipartition of S if $S_1 \cup S_2 = S$, $S_1 \cap S_2 = \emptyset$ and $S_1, S_2 \neq \emptyset$.

Algorithm 1: The general algorithm.

```
1  Function SOLVE(c, u):
2      s ← the number of the third dimension of u
3      if s = 1 then
4          return c
5      end
6      Choose any non-empty bipartition[2] {K_1, K_2} of
         {1, 2, ..., s}. WLOG, assume that K_1 = {1, 2, ..., t},
         K_2 = {t + 1, t + 2, ..., s}
7      u'_{ijp} ← Σ_{k∈K_p} u_{ijk}, ∀i ∈ I, j ∈ J, p ∈ {1, 2}
                                          ▷ The merging step.
8      x* ← the optimal solution of OPT(c, u')
                  ▷ Using the algorithm in Section 4.1.
9      u^(1)_{ijk} ← u_{ijk}, ∀i ∈ I, j ∈ J, k ∈ K_1
10     u^(2)_{i,j,(k−t)} ← u_{ijk}, ∀i ∈ I, j ∈ J, k ∈ K_2
11     c^(p)_{ij} ← Σ_{k∈K_p} x*_{ijk}, ∀i ∈ I, j ∈ J, p ∈ {1, 2}
12     x^(p) ← SOLVE(c^(p), u^(p)), ∀p ∈ {1, 2}
                                       ▷ The decomposition step.
13     x*_{ijk} ← x^(1)_{ijk}, ∀i ∈ I, j ∈ J, k ∈ K_1
14     x*_{ijk} ← x^(2)_{i,j,(k−t)}, ∀i ∈ I, j ∈ J, k ∈ K_2
15     return x*
```

$u^{\text{Sub}}_{ijp} = \sum_{k \in K_p} u^{\text{Root}}_{ijk}, \forall p \in \{1, 2\}$. Then

$$x_{ij1} = \frac{\sum_{k \in K_1} r_k}{\sum_{k \in K_1 \cup K_2} r_k} \cdot \sum_{k \in K_1 \cup K_2} u^{\text{Root}}_{ijk}$$

satisfies all the constraints of OPT($c^{\text{Sub}}, u^{\text{Sub}}$), except for the constraint that x must be integers, because

(i) for constraint (6c),

$$\begin{aligned} x_{ij1} &\le \sum_{k \in K_1 \cup K_2} u^{\text{Root}}_{ijk} \\ &= \sum_{k \in K_1} u^{\text{Root}}_{ijk} + \sum_{k \in K_2} u^{\text{Root}}_{ijk} \\ &= u^{\text{Sub}}_{ij1} + u^{\text{Sub}}_{ij2} \\ &= c^{\text{Sub}}_{ij}, \end{aligned}$$

(ii) for constraint (6a),

$$\begin{aligned} \sum_{i \in I} x_{ij1} &= \frac{\sum_{k \in K_1} r_k}{\sum_{k \in K_1 \cup K_2} r_k} \cdot \sum_{k \in K_1 \cup K_2} \sum_{i \in I} u^{\text{Root}}_{ijk} \\ &= \frac{\sum_{k \in K_1} r_k}{\sum_{k \in K_1 \cup K_2} r_k} \cdot \sum_{k \in K_1 \cup K_2} a^{\text{Root}}_{jk} \\ &= \frac{\sum_{k \in K_1} r_k}{\sum_{k \in K_1 \cup K_2} r_k} \cdot \alpha_j \sum_{k \in K_1 \cup K_2} r_k \quad \text{(Definition 3.1)} \\ &= \sum_{k \in K_1} a^{\text{Root}}_{jk} = \sum_{i \in I} \sum_{k \in K_1} u^{\text{Root}}_{ijk} = \sum_{i \in I} u^{\text{Sub}}_{ij1} \\ &= a^{\text{Sub}}_{j1}, \end{aligned}$$

(iii) and similarly, for constraint (6b), $\sum_{j\in J} x_{ij1} = b^{\text{Sub}}_{i1}$.

A famous property of MCF problems is that if there exists a real number feasible solution, then there must exist an integer feasible solution [24], so $\text{OPT}(c^{\text{Sub}}, u^{\text{Sub}})$ is feasible. □

We now analyze the time complexity. For simplicity, we assume that in line 6, we always choose an even bipartition, i.e., $t = \lfloor s/2 \rfloor$. Assume that the time needed for the root call $\textsc{solve}(c^{\text{Root}}, u^{\text{Root}})$ is $T(m, n)$. Then the recursive step in line 12 takes $2T(m, n/2)$. The running time of line 8 depends on the concrete MCF algorithm used. For example, the time complexity of the cost-scaling algorithm is $O(N^2 M \log(NC))$, where N is the number of nodes, M is the number of arcs, and C is the largest arc cost [16]. Our algorithm requires constructing a graph with $2m$ nodes and at most $3m^2$ arcs. Therefore, if we use the cost-scaling algorithm for line 8, its time complexity will be $O(m^4 \log m)$. Other simple array copying and summing operation in our algorithm takes $O(m^2 n)$. Therefore, we have

$$\begin{cases} T(m, n) = T(m, n/2) + O(m^4 \log m) + O(m^2 n), \\ T(m, 1) = 1. \end{cases}$$

Applying the Master Theorem [5], we obtain that the time complexity is $O(m^4 n \log m + m^2 n \log n)$, where the first factor $m^4 n \log m$ is the main factor.

5 EVALUATION

5.1 Evaluation Setup

Simulation Scenario. Our evaluation setup considers a DCN where ToR switches are interconnected via a layer of OCSes that have uniform connections to the ToR switches. We assume continuous monitoring of the traffic between ToR switches, and our objective is to switch the OCSes at a fixed interval of Δt to enable the logical topology to adapt to changes in the traffic pattern.

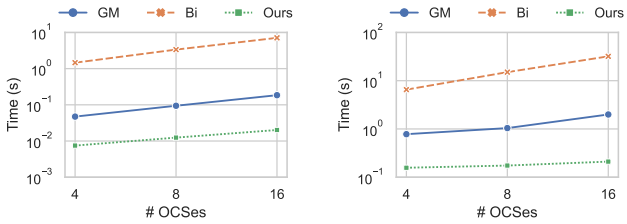
Data source. We use the open traffic traces of Facebook datacenters to generate the desired logical topologies [3]. In detail, given the traffic matrix T aggregated in a time period, where T_{ij} represents the volume of traffic from the i-th switch to the j-th switch, the desired logical topology $x_{ij} \in \{0, 1\}^{m\times m}$ is obtained by solving the following problem.

$$\begin{aligned} \min_{x} \quad & \sum_{i\in I}\sum_{j\in J} T_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{i\in I} x_{ij} = \sum_{k\in K} a_{jk}, \quad \forall j \in J \\ & \sum_{j\in J} x_{ij} = \sum_{k\in K} b_{ik}, \quad \forall i \in I \end{aligned}$$

Each trace provides information for one day. We choose $\Delta t = 10$ mins, so each trace will provide around 144 test cases.

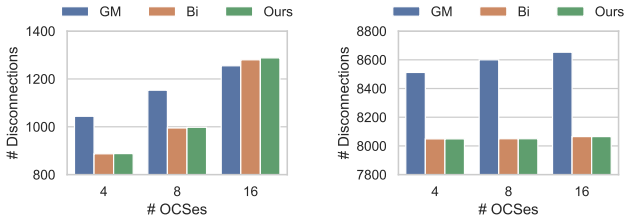
Parameters. As for physical topology, we assume that each ToR switch has 32 uplinks that are equally connected to the OCSes. We tested our algorithm on cases where there are 4, 8, and 16 OCSes. We use the traces of Facebook Cluster A and B, which have 155 and 324 ToR switches, respectively.

Implementation. We implemented our simulation program in Java, using Gurobi as the ILP solver and the JGraphT library [18] for the MCF problem solver.



(a) Cluster A. (b) Cluster B.

Figure 2: The average running time. Note that the vertical axis of the graphs is plotted on a logarithmic scale.



(a) Cluster A. (b) Cluster B.

Figure 3: The average number of disconnections.

5.2 Evaluation Results

The results of the evaluation are summarized in Figure 2 and Figure 3. We did not include the results of the Brute Force Algorithm, because even in the simplest cases (Cluster A with 4 OCSes), this algorithm sometimes cannot find a feasible solution in 1 minute.

Running time. Figure 2 displays the running time of the evaluated algorithms. Our proposed algorithm overwhelmingly outperforms the others in terms of speed, and in some instances, it is up to 10 times faster than the second fastest algorithm, the Greedy MCF Algorithm.

Number of disconnections. Figure 3 shows the number of disconnections of the schemes calculated by the algorithms. The approximation ratio of our algorithm is comparable to that of the Bipartition Algorithm, and in most cases, our algorithm has a better approximation ratio than the Greedy MCF Algorithm.

To conclude, our algorithm achieves significant speed improvements while maintaining a comparable approximation ratio.

6 RELATED WORK

In this section, we mainly focus on the related theoretical results concerning this problem. A comprehensive discussion of the advantages and disadvantages of the existing practical algorithms has been presented in Section 2.1.

The Graver basis and n-fold ILP. By imposing a constraint $x_{ijk} \in \{0, 1\}$, the objective function can be transformed into a purely linear form. Then our problem falls into the category of n-fold ILP problems [8], which can be solved in polynomial time by leveraging the unique structure of the Graver basis of the constraint matrix, provided that m is considered as a constant. Recently,

a state-of-the-art algorithm has been proposed to solve such problems in near-linear time [15]. Unfortunately, these algorithms suffer from the limitation that the Graver basis needs to be determined in advance, which has an exponential size of m [6]. To address this issue, Altmanová et al. [2] proposed an improved algorithm. However, it is still impractical for the scale of our problem.

Convex cost MCF problem. Essentially, the bipartition step of our algorithm involves solving an integral MCF problem with a convex objective function. Such problems can be easily converted to ones with a purely linear objective function by splitting each arc into multiple arcs, as we do in the paper. The advantages of this approach is that we can apply mature classic MCF algorithms, but the disadvantage is that if we consider the weighted case (see the remark for function (4)), the piecewise-linear objective function in the equivalent MCF problem may have too many intervals, so there will be too many arcs in the corresponding graph. A possible improvement is to use algorithms specialized for convex cost MCF problems, such as [19].

7 CONCLUSION AND FUTURE WORK

In this paper, we presented a polynomial-time algorithm for reducing the reconfiguration time in hybrid optical-electrical DCNs. The evaluation driven by real-world traces showed that our algorithm outperforms existing approaches. We believe that our proposed algorithm can enhance the performance of hybrid optical-electrical DCNs and facilitate the adoption of OCSes in DCNs.

Moving forward, we intend to further study this field. Specifically, we aim to establish more theoretical properties of our algorithm, such as worst-case bounds. Moreover, we plan to conduct experiments on actual hardware instead of software simulators to validate the effectiveness of our algorithm in a practical setting.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China under Grant 62272292 and 61902246.

REFERENCES

[1] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. *ACM SIGCOMM computer communication review* 38, 4 (2008), 63–74.
[2] Kateřina Altmanová, Dušan Knop, and Martin Koutecký. 2019. Evaluating and tuning n-fold integer programming. *Journal of Experimental Algorithmics (JEA)* 24 (2019), 1–22.
[3] Chen Avin, Manya Ghobadi, Chen Griner, and Stefan Schmid. 2020. On the complexity of traffic traces and implications. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 4, 1 (2020), 1–29.
[4] Theophilus Benson, Aditya Akella, and David A Maltz. 2010. Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. 267–280.
[5] Jon Louis Bentley, Dorothea Haken, and James B Saxe. 1980. A general method for solving divide-and-conquer recurrences. *ACM SIGACT News* 12, 3 (1980), 36–44.
[6] Yael Berstein and Shmuel Onn. 2009. The Graver complexity of integer programming. *Annals of Combinatorics* 13 (2009), 289–296.
[7] Kai Chen, Ankit Singla, Atul Singh, Kishore Ramachandran, Lei Xu, Yueping Zhang, Xitao Wen, and Yan Chen. 2013. OSA: An optical switching architecture for data center networks with unprecedented flexibility. *IEEE/ACM Transactions on Networking* 22, 2 (2013), 498–511.
[8] Jesús A De Loera, Raymond Hemmecke, Shmuel Onn, and Robert Weismantel. 2008. N-fold integer programming. *Discrete Optimization* 5, 2 (2008), 231–241.
[9] Jesús A De Loera and Shmuel Onn. 2006. All linear and integer programs are slim 3-way transportation programs. *SIAM Journal on Optimization* 17, 3 (2006), 806–821.
[10] Klaus-Tycho Foerster and Stefan Schmid. 2019. Survey of reconfigurable data center networks: Enablers, algorithms, complexity. *ACM SIGACT News* 50, 2 (2019), 62–79.
[11] Alan M Frieze. 1983. Complexity of a 3-dimensional assignment problem. *European Journal of Operational Research* 13, 2 (1983), 161–164.
[12] Albert Greenberg, James R Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A Maltz, Parveen Patel, and Sudipta Sengupta. 2009. VL2: A scalable and flexible data center network. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*. 51–62.
[13] Gurobi Optimization, LLC. 2023. Gurobi Optimizer Reference Manual. https://www.gurobi.com
[14] Robert W Irving and Mark R Jerrum. 1994. Three-dimensional statistical data security problems. *SIAM J. Comput.* 23, 1 (1994), 170–184.
[15] Klaus Jansen, Alexandra Lassota, and Lars Rohwedder. 2020. Near-linear time algorithm for n-fold ILPs via color coding. *SIAM Journal on Discrete Mathematics* 34, 4 (2020), 2282–2299.
[16] Zoltán Király and Péter Kovács. 2012. Efficient implementations of minimum-cost flow algorithms. *arXiv preprint arXiv:1207.6381* (2012).
[17] Weiqiang Li, Rui Wang, and Jianan Zhang. 2022. Configuring data center network wiring. US Patent 11,223,527.
[18] Dimitrios Michail, Joris Kinable, Barak Naveh, and John V Sichi. 2020. JGraphT—A Java library for graph data structures and algorithms. *ACM Transactions on Mathematical Software (TOMS)* 46, 2 (2020), 1–29.
[19] Michel Minoux. 1986. Solving integer minimum cost flows with separable convex cost objective polynomially. *Netflow at Pisa* (1986), 237–239.
[20] Leon Poutievski, Omid Mashayekhi, Joon Ong, Arjun Singh, Mukarram Tariq, Rui Wang, Jianan Zhang, Virginia Beauregard, Patrick Conner, Steve Gribble, et al. 2022. Jupiter evolving: Transforming google's datacenter network via optical circuit switches and software-defined networking. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 66–85.
[21] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, et al. 2015. Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network. *ACM SIGCOMM computer communication review* 45, 4 (2015), 183–197.
[22] CALIENT Technologies. 2023. CALIENT Data Sheet. https://www.calient.net
[23] Min Yee Teh, Shizhen Zhao, Peirui Cao, and Keren Bergman. 2022. Enabling Quasi-Static Reconfigurable Networks With Robust Topology Engineering. *IEEE/ACM Transactions on Networking* (2022).
[24] Laurence A Wolsey. 2020. *Integer programming*. John Wiley & Sons.
[25] Shuyuan Zhang and Shu Shan. 2023. Repository. https://github.com/apnet23-reducing/evaluation
[26] Shizhen Zhao, Rui Wang, Junlan Zhou, Joon Ong, Jeffrey C Mogul, and Amin Vahdat. 2019. Minimal Rewiring: Efficient Live Expansion for Clos Data Center Networks.. In *NSDI*. 221–234.