Leaf: Improving QoS for Reconfigurable Datacenters with Multiple Optical Circuit Switches

Jingzhou Wang[1,2] Gongming Zhao[1,2] Hongli Xu[1,2] Haibo Wang[3]
[1]School of Computer Science and Technology, University of Science and Technology of China
[2]Suzhou Institute for Advanced Research, University of Science and Technology of China
[3]Department of Computer Science, University of Kentucky

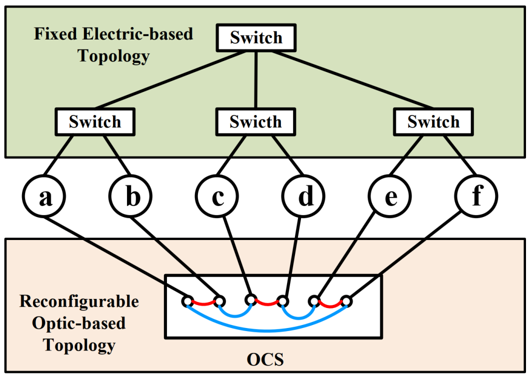
***Abstract*—Facing the huge volume of traffic and intensive traffic dynamics, the traditional datacenter architectures are behind the curve due to the fixed topology and the demand-oblivious nature. Driven by the traffic pattern, the reconfigurable technologies, like optical circuit switches (OCSes), are a promising alternative to further improve throughput when facing traffic dynamics, thanks to the high bandwidth and low reconfiguration time. However, existing works on OCSes are relatively elementary. Some of previous works only deploy single OCS which cannot adapt to large-scale datacenters, while others either cannot guarantee near-optimality or overlook practical issues like limited fiber capacity. These solutions may cause low throughput and long reconfiguration time, resulting in poor QoS. In this paper, we present *Leaf* to maximize throughput thus to further improve QoS, by deploying multiple OCSes to carry traffic in a cooperation manner. Furthermore, we also show its efficient reconfiguration ability and near-optimality. The formulated problem is a new k-weight limited matching problem and is proven to be $\mathcal{NP}$-hard, which can be solved by a new proposed approximation algorithm with bounded approximation ratio. To evaluate our proposed solution, simulation experiments are conducted with both real-world and synthetic datasets. Compared with state-of-the-arts works, *Leaf* can improve throughput by $42.16\%-68.92\%$, and reduce runnning time by $68.87\%-78.72\%$.**

***Index Terms*—Reconfigure, QoS, Datacenter, Optical Circuit Switch**

I. Introduction

Current data-intensive web applications like high-resolution video streaming, online games and distributed machine learning [1] have led to a significant increase of communication traffic in datacenters over the decades [2]. Motivated by the rapid increasing communication demand and the traffic pattern features, great efforts have been made to design more efficient datacenter structure, like Fat-tree [3], Hypercube [4] and Butterfly [5]. However, these solutions all have in common that the architecture designs are fixed, resulting that datacenters are *demand-oblivious* [6], making datacenters perform poorly and damage QoS when facing intensive dynamics of traffic [7]. Moreover, these solutions are also electric-based, requiring complex wiring and fail to scale to accommodate growth [8]. Therefore, many researchers tend to explore a more agile and efficient manner to help improving QoS by dynamically adjusting the datacenter topology and connections between racks when facing traffic dynamics [9].

In recent years, emerging reconfigurable technologies have enabled an alternative solution to current datacenter architecture, which introduces the possibility to ***reconfigure*** the datacenter structure and topology at runtime [10]. Among several alternatives, optical circuit switches (OCSes) have been a popular choice, thanks to the low reconfiguration time (a few μs) [11], high bandwidth transmission (hundreds of Gbps) [11], and being compatible with traditional electric-based datacenter networks [12].


Fixed Electric-based Topology
Switch
Switch
Swicth
Switch
a
b
c
d
e
f
Reconfigurable Optic-based Topology
OCS

Time	Burst Traffic/Configuration
T1	{a-b, c-d, e-f}
T2	{a-f, b-c, d-e}

Fig. 1. A datacenter topology with racks a-f, four traditional switches and one optical circuit switch (OCS).

In brief, an OCS uses Micro-Electro-Mechanical Systems (MEMS) to directly transfer the carrier laser beams from input ports to output ports without decoding any packets [13]. The configuration between input-output ports, *i.e.*, the connections, can be easily reconfigured by MEMS according to traffic dynamics. Specifically, as shown in Fig. 1, the datacenter consists of six racks denoted from a to f, each of which is equipped with a Top-of-Rack (ToR) switch. The upper part of the figure is the fixed electric-based network topology with four traditional switches, which are static and oblivious to the traffic dynamics. While the lower part is the reconfigurable optic-based topology with one OCS. There are six ports of the OCS and each port connects one ToR switch. The topology in this figure adopts the *separation* paradigm proposed in [9] and [14]. The authors propose that mice and regular flows are transmitted via the fixed topology, while reconfigurable optic-based network carries elephant or burst flows. Suppose that the traffic bursts between $a-b$, $c-d$ and $e-f$ at time T_1. Then we have the configuration to connect corresponding racks, consisting of three connections colored by red. While at time T_2, traffic dynamics take place, and the burst traffic would exist between $a-f$, $b-c$ and $d-e$. Thus, we reconfigure the OCS to connect corresponding racks, which also consists of three connections yet colored by blue. Since the reconfiguration of OCSes only takes a few μs, it is very efficient to switch the configuration to improve throughput.

In the context of OCSes, there have been several kinds of researches, like flow scheduling [15] [16] and OCS deployment [17] [18]. However, these works didn't fully take advantages of OCSes [9], since they ignore the low reconfiguration time and only focus on the high-bandwidth of OCSes [15]. Thus, another major problem is raised: To fully leverage the low reconfiguration time feature and improve QoS for reconfigurable datacenters, ***how can we swiftly and near-optimally compute configurations for OCSes to maximize throughput, so that datacenters can better deal with traffic dynamics?*** To solve the raised problem, authors in [8] first propose a solution with single OCS. They formulate the throughput maximization problem as a maximum *graph matching* problem and solve it with Edmond's blossom algorithm [19]. This solution is also studied in C-through [11] and Solstice [20]. However, these works only focus on single OCS, which fail to be extended to multiple OCSes scenario. Existing works on multiple OCSes are relatively fewer, and they either cannot guarantee near-optimality [13], or overlook practical constraints like limited fiber capacity [21] [22]. Thus, these solutions cannot fully maximize throughput when facing traffic dynamics, and cause huge waste of traffic, which would damage QoS.

In this paper, we present ***Leaf***, focusing on how to improve QoS by maximizing throughput with multiple OCSes in datacenters considering practical and efficient features. Specifically, ***Leaf*** shows how to leverage multiple OCSes to form connections between racks with limited and non-uniform fiber capacity. In our solution, the configurations of different OCSes are not independent. They can cooperate with each other to carry more traffic between a pair of racks which holds heavy communication demand, thus to avoid much waste of traffic. This method can solve the shortcomings of [21] and [22]. Therefore, the throughput result of our solution is further improved. In fact, the formulated problem of ***Leaf*** is theoretically and mathematically complex. Even if we ignore some constraints of the problem, it remains $\mathcal{NP}$-hard. Nevertheless, we will consider more constraints in our model, making our scheme more difficult but also more practical, compared with existing works. The main contributions of this paper can be summarized as follows:

- We comprehensively summarize current reconfigurable solutions for datacenters and analyze their pros and cons with an example to strengthen our motivation.
- We give the problem formulation of ***Leaf***, *i.e.*, the k-weight limited matching problem, and we analyze its $\mathcal{NP}$-hardness and inapproximability.
- We propose a new approximation algorithm for ***Leaf*** based on greedy-iteration. Furthermore, we prove the proposed algorithm yields $\frac{1}{2}$-approximation and time complexity is $\mathcal{O}(km(\log m+n))$, where k is the number of OCSes and m/n is the number of edges/vertices in formulated graph.
- We conduct extensive experiments to compare ***Leaf*** with state-of-the-art solutions. With both real-world and synthetic datasets, we evaluate the throughput and running time of ***Leaf*** to show its superior performance.

II. Background and Motivation

This section discusses existing works on reconfiguring OCSes for throughput maximization, and show their shortcomings with a motivating example.

A. Mathematical Modeling

Before we move on, there is a significant fact about mathematical modeling of OCSes' configurations. Back to Fig. 1, we can see that the connections belonging to the same configuration are *rack-disjoint*, *i.e.*, every rack can only establish single connection in the OCS. For instance, in the configuration colored by red, rack a can only be connected to b, until the OCS is reconfigured. This is because circuit switching allow connections to be formed between only two racks each time [10]. Thus, from the mathematical perspective, ***each configuration of an OCS can be regarded as a matching of network topology***. Therefore, most of related works would leverage graph matching theory and algorithms, which is also a key idea of this paper.

B. Current Solutions and Limitations

We summarize the advantages and disadvantages of existing works on throughput maximization of OCS in Table I. In this section, k, n, m represent the number of OCSes, the number of racks and the number of communication demands, respectively. Here *communication demand* represents that whether two racks have to communicate with each other. Back in Fig. 1, we have $m=3$ at time T_1, including communication demands $a-b$, $c-d$ and $e-f$. According to our research, there are mainly three categories of solutions as follows.

1) Single OCS solution: Achieving a reconfigurable datacenter with single OCS is relatively simple. The authors of [8] and [11] first leverage single OCS to improve throughput in datacenters. The formulated problem is the maximum graph matching problem, which is solved by Edmond's blossom algorithm [23], the time complexity of which is $\mathcal{O}(mn^2)$. However, it is very natural to adopt multiple OCSes in current datacenter for more throughput. These solutions clearly are not suitable for the situation.

2) Multiple OCSes solution: Helios: One of the representative work of multiple OCSes solution is Helios [13]. The authors propose to leverage multiple OCSes to further improve throughput. The solution is straightforward, greedily selecting current configuration with maximum traffic volume for OCSes, and the time complexity is $\mathcal{O}(kmn^2)$. However, this is proven to be not optimal [22], and authors didn't give any near-optimality guarantee. Furthermore, this work only considers

TABLE I
COMPARISON OF ADVANTAGES AND DISADVANTAGES OF EXISTING WORKS

Solutions	Practical Features			Efficient Features	
	Multiple OCSes	Limited Fiber Capacity	Non-uniform Fiber Capacity	Near Optimality	Time Complexity
Single OCS (*e.g.*, [20], [8], [11])	✗	✓	✗	✓	$\mathcal{O}(mn^2)$
Helios [13]	✓	✓	✗	✗	$\mathcal{O}(kmn^2)$
k-DM [21], [22]	✓	✗	✗	✓	$\mathcal{O}(kn(n\log n + m))$
Leaf	✓	✓	✓	✓	$\mathcal{O}(km(\log m + n))$

uniform capacity of fibers, which is less practical. According to [24], the capacity of a fiber can range from 10 Gbps to 10 Tbps. Thus, it is necessary to take non-uniform fiber capacity into consideration.

3) Multiple OCSes solution: k-Disjoint Matching (k-DM): To explore how to guarantee near-optimality in throughput maximization with multiple OCSes, the works of [22] and [21] leverage different mathematical theories to help design algorithms. Specifically, they formulate the problem as a k-Disjoint Matching (k-DM) problem, *i.e.*, every edge in the topology only belongs to one matching. To solve the problem, they propose a series of algorithms. A representative algorithm called Blossom-iteration uses the Blossom algorithm repeatedly for each OCS. The authors prove that the approximation ratio is less than $\frac{7}{9}$, and the time complexity is $\mathcal{O}(kn(n\log n + m))$. However, these works focus on mathematical theories and overlook the practical conditions, like limited and non-uniform fiber capacity. Since authors assume the capacity of fibers to be unlimited, excessive traffic may be assigned to fibers, causing waste of traffic.

C. A Motivating Example

We use Figs. 2-3 to show an example to motivate our work on reconfigurable datacenters with multiple OCSes.

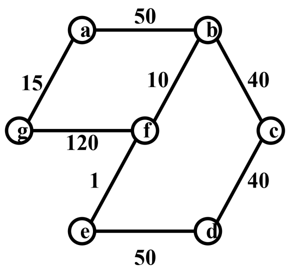


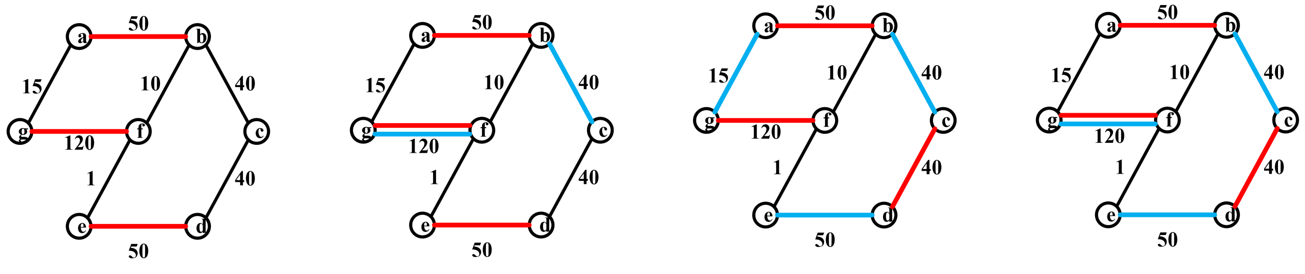
Fig. 2. The topology of the motivating example. The vertices ($a-g$) represent different Top-of-Rack switches, while the black lines are communication demand. The weight of each edge represents specific traffic volume.

Fig. 2 shows the network topology and corresponding traffic information. In general, there are 7 racks within the datacenter, and the Top-of-Rack (ToR) switches are denoted by vertices labeled from a to g. The edges represent the *communication demand* between two racks. If two racks need to communicate with each other, the two corresponding vertices are connected with an edge. In this example, we have 8 communication demands/edges. The weight of each edge represents specific *traffic volume* of each communication demand. According to the spatial and temporal features of datacenter traffic [25], the traffic volume varies greatly among rack pairs. For instance, in this example, the racks f and g have 120 units of traffic volume, while racks a and c do not need to communicate with each other. The information above constitutes *a simple, undirected and edge-weighted graph.*

The configuration of an OCS can be regarded as *a matching* of the formulated graph, as we discussed in Section II-A. Since we consider practical constraints, we require that the fiber capacity should be limited, assuming that all optical fibers can hold 50 units of traffic. Moreover, we calculate the *actual throughput* according to fiber's capacity and the traffic volume assigned to it, abandoning the overloaded traffic. That's to say, when we find a configuration for an OCS, not all traffic can be transmitted, due to the limited capacity of fibers. For instance, we find a configuration for the motivating example connecting $a-g$, $b-f$, $c-d$. The theoretical total traffic volume is $15+10+40=65$ units. However, if given three optical fibers with 20 units of capacity, the actual throughput is only $15+10+20=45$ units, abandoning 20 units of traffic between c and d. We consider two OCSes in this example. The four solutions are shown in Fig. 3 and the results comparison is shown in Table II.

First, consider traditional solutions, by deploying single OCS. The set of maximum weighted matching can be acquired by Edmond's blossom algorithm [23]. In our motivating example, the matching includes $\{(a,b),(d,e),(f,g)\}$, which means that the configuration is connecting $a-b$, $d-e$ and $f-g$, as shown in Fig. 3(a). The theoretical total traffic volume is 220 units. However, since the capacity of fibers is 50 units, we have to abandon 70 units of traffic volume between f and g. Thus, we only have 150 units of actual throughput. Existing works like [20] and [11] adopted this solution.

Helios would greedily choose the matching with the maximum weight from current topology. Then, the graph would update residual edge weight. The results are shown in Fig. 3(b). First, the matching colored by red including $\{(a,b),(d,e),(f,g)\}$ will be chosen, which is the same as the one in Fig. 3(a). Considering 50 units of capacity, the residual traffic volume of rack pairs $a-b$, $d-e$, $f-g$ will be reduced from $50, 50, 120$ units to $0, 0, 70$ units, respectively. Thus, the second matching colored by blue only includes $\{(b,c),(f,g)\}$,



(a) Single OCS solution (b) Helios (c) k-DM (d) ***Leaf***

Fig. 3. Four kinds of solutions comparison. The colored lines are different connections of OCSes' configurations. Connections with the same color belongs to one configuration. Mathematically, configurations can be regarded as matchings of the topology.

which means the second OCS can only connect $b - c$ and $f - g$. The actual throughput is 240 units.

With the advancement of the work [22], authors formulate the throughput maximization problem as a k-Disjoint Matching problem, and propose a series of algorithms (like Blossom-iteration) to solve it. Note that, the authors consider that capacity of fibers is unlimited. Thus, under this circumstance, different configurations of OCSes are completely *connection-disjoint*, *i.e.*, a pair of racks are connected by at most one OCS. In our motivating example, we can acquire the solution as shown in Fig. 3(c). The first matching colored by red includes $\{(a, b), (c, d), (f, g)\}$. The second matching colored by blue includes $\{(a, g), (b, c), (d, e)\}$. The actual units of throughput are 245 units.

By adopting our solution ***Leaf***, the derived result can be seen in Fig. 3(d). The first and the second matchings are colored by red and blue respectively. Since the fiber capacity is taken into consideration, the configurations of different OCSes are no longer connection-disjoint. They can cooperate to carry more traffic between two racks. In this example, the communication demand between f and g is chosen in two configurations, *i.e.*, the two OCSes will simultaneously establish connections between f and g.to carry traffic The two matchings are $\{(a, b), (c, d), (f, g)\}$ and $\{(b, c), (d, e), (f, g)\}$, and the actual throughput is 280 units. Compared with single OCS solution, Helios and k-DM, ***Leaf*** can further improve actual throughput by 86.67%, 16.67% and 14.29%, respectively.

Moreover, since we also consider the situation where the capacity of fibers is non-uniform, we can further improve the actual throughput when fibers with different capacity are provided. For instance, in our motivating example, assume that one fiber has 60 units of capacity while others still have 50 units of capacity. Then the fiber with 60 units of capacity will be used to carry traffic between f and g. Under this circumstance, the actual throughput will be improved to 290 units. However, since other solutions do not take non-uniform capacity into consideration, the fiber with higher capacity may be randomly deployed, probably causing waste of capacity.

This example fully demonstrates the potential of ***Leaf*** to further improve throughput in reconfigurable datacenters with multiple OCSes. However, we will show the formulated problem of ***Leaf*** is theoretically and mathematically difficult in the next section.

III. Preliminaries

This section presents the formulation of ***Leaf***, including detailed modeling of reconfigurable datacenters and the formulated problem of throughput maximization for multiple OCSes. Finally we analyze its mathematical complexity compared with existing problems and further prove its $\mathcal{NP}$-hardness and inapproximability.

A. Reconfigurable Datacenter Model

We formulate the reconfigurable datacenter as a graph model. In general, a datacenter deploys a set of racks, and each rack has a Top-of-Rack (ToR) switch. We denote the set of ToR switches as $V = \{v_1, v_2, ..., v_{|V|}\}$. Then, every two ToR switches may have communication demand. We use $E = \{e_1, e_2, ..., e_{|E|}\}$ to represent the demand set and D to represent the specific traffic volume, where $D : E \to N_0$. Then, we have a simple, undirected, edge-weighted graph $G = (V, E, D)$. Here the ToR switches set V represents the vertex set of the graph. The communication demand set E can be regarded as the edge set, and the traffic volume function D means the weight of each edge.

For instance, in Fig. 2, we have the switch set $V = \{a, b, c, d, e, f, g\}$, with $|V| = 7$. The communication demand set is $E = \{(a, b), (b, c), ..., (b, f)\}$, with $|E| = 8$. The traffic volume is defined as $D : (a, b) \to 50, (b, c) \to 40, (c, d) \to 40, (d, e) \to 50, (e, f) \to 1, (f, g) \to 120, (a, g) \to 15, (b, f) \to 10$. The datacenter network topology is denoted by $G = (V, E, D)$.

B. Problem Formulation

OCSes use Micro-Electro-Mechanical Systems (MEMS) to redirect carrier laser beams from input and output ports to form the connection between two ToRs. Thus, from mathematical perspective, an OCS $i \in I$ and its configuration M^i (how the OCS connects ToR switches) can be regarded as a matching of the topology G. Mathematically, a matching is a subset of the edge set, where every two edges are disjoint, *i.e.*, they do not share the same vertex. We use variable $x^i_{e,l}$ to represent how much traffic volume of communication demand e in configuration M^i is carried by a fiber $l \in L^i$, where L^i represents the set of fibers corresponding to the OCS

TABLE II
COMPARISON RESULTS OF MOTIVATING EXAMPLE.

Solutions	Configurations (Matchings)	Actual Throughput	Total Throughput
Single OCS	$\{(a,b),(d,e),(f,g)\}$	150	150
Helios	$\{(a,b),(d,e),(f,g)\}$	150	240
	$\{(b,c),(f,g)\}$	90	
k-DM	$\{(a,b),(c,d),(f,g)\}$	140	245
	$\{(a,g),(b,c),(d,e)\}$	105	
Leaf	$\{(a,b),(c,d),(f,g)\}$	140	280
	$\{(b,c),(d,e),(f,g)\}$	140	

i. The limited capacity of each fiber l is denoted as $c(l)$. We denote the total actual throughput of a configuration as $D(M^i) = \sum_{e,l} x^i_{e,l}$, representing how much traffic can this OCS hold. Our objective is to find $k = |I|$ configurations, such that $\sum_{i \in I} D(M^i)$ is maximized.

For instance, in Fig. 3(d), our solution acquires configurations for two OCSes as follows: $M^1 = \{(a,b),(c,d),(f,g)\}$ and $M^2 = \{(b,c),(d,e),(f,g)\}$. The traffic over each fiber can be represented as $x^1_{(a,b)} = 50, x^1_{(c,d)} = 40, x^1_{(f,g)} = 50$, and $x^2_{(b,c)} = 40, x^2_{(d,e)} = 50, x^2_{(f,g)} = 50$. We omit l in subscript for simplicity, since all fibers' capacity are uniform ($c(l) = 50$). The actual throughput is $D(M^1) = 140$ and $D(M^2) = 140$. We formulate the problem as follows.

$$\max \sum_{i \in I} D(M^i)$$

$$S.t. \begin{cases} x^i_{e,l} \le c(l), & \forall e \in E, l \in L^i, i \in I \\ \sum_{i \in I} \sum_{l \in L^i} x^i_{e,l} \le D(e), & \forall e \in E \\ e \cap e' = \emptyset, & \forall e, e' \in M^i, i \in I \\ D(M^i) = \sum_{e,l} x^i_{e,l}, & \forall M^i, i \in I \\ x^i_{e,l} \ge 0, & \forall e \in E, l \in L^i, i \in I \end{cases} \quad (1)$$

The first set of inequality represents the fiber capacity constraint.The second set of inequality means the traffic volume constraint. The third set of equation means M^i only include edges which do not share the same vertex. We calculate the actual throughput of a matching in the fourth set of equation. Our objective is to maximize the total actual throughput.

This is a new problem compared with k-Disjoint Matching problem defined in [22], since we allow an edge can be included in two or more matchings. Moreover, due to the limited capacity of fibers, our matchings' weight (*i.e.*, OCSes' actual throughput) is limited. Thus, the problem is called the *k-weight limited matching problem*.

C. Problem Complexity Analysis

This section shows the complexity of the k-weight limited matching problem of ***Leaf***, and further proves its $\mathcal{NP}$-hardness and inapproximability.

1) Multiple Optical Circuit Switches: Currently, many researches focus on the problem that how to maximize throughput with single OCS [20], [11], [14], which is can be formulated as a simple weight maximum matching problem and we can find the optimal solution in polynomial time. The situation where multiple OCSes are deployed is first considered in [13], which is solved by greedily choosing the maximum weight matching. However, the authors in [22] point out that multiple matchings problem is $\mathcal{NP}$-hard, and the greedy solution is not optimal. Thus, it is tricky to design an efficient algorithm with bounded approximation ratio to solve the problem.

2) Limited and Non-uniform Fiber Capacity: Although authors in [22] and [21] have discussed how to deal with the situation where multiple OCSes are adopted, they overlook some practical constraints like limited fiber capacity. In fact, the formulated problem in their works is a special case of ours. Supposing that the minimum fiber capacity is larger than the maximum weight of all edges, our problem can be generalized to theirs.

3) $\mathcal{NP}$-Hardness and Inapproximability: Sections above show that our problem is more complex compared with existing ones. This section formally proves its $\mathcal{NP}$-hardness and inapproximability.

Theorem 1: The k-weight limited matching problem defined in Eq. (1) is $\mathcal{NP}$-hard.

Proof: We prove the $\mathcal{NP}$-hardness by showing that k-weight limited matching problem can be reduced to k-disjoint matching problem [22]. Specifically, we assume that the minimum capacity of all fibers is larger than the maximum weight of all edges. That is to say, an arbitrary fiber can carry communication demand between any pair of ToR switches. Thus, there no longer exist multiple matchings share the same edge. Our problem becomes the k-disjoint matching problem, which is $\mathcal{NP}$-hard.

Moreover, we can also assume there exists an efficient algorithm which can solve k-weight limited matching problem, denoted by Alg. A. Specifically, Alg. A can acquire the optimal solution under the situation where the capacity of fibers can either be larger or smaller than the weight of edges. Therefore, we can definitely use Alg. A to acquire the optimal solution where the minimum capacity of all fibers is larger than the maximum weight of all edges, which is exactly the scenario of k-disjoint matching problem. Thus, the k-weight limited matching problem is $\mathcal{NP}$-hard too. ■

Theorem 2: It is $\mathcal{NP}$-hard to approximate the k-weight limited problem within a factor of $(1-\epsilon)$ for any $\epsilon \in [0, \frac{1}{m}]$, where m is the number of edges and ϵ is an arbitrarily small value.

Before we prove the inapproximability of the k-weight limited problem, a famous lemma needs to be introduced.

Lemma 3: It is $\mathcal{NP}$-complete to determine whether the chromatic index of a cubic graph is 3 or 4. Here a cubic graph refers to a graph where all vertices have degree 3 [26].

Proof: We consider a cubic graph with all edges' weights are uniform (setting to 1). If the chromatic index of the cubic graph is 3, the optimal solution with $k = 3$ is m. Meanwhile, all algorithms with approximation ratio larger than $(1 - \frac{1}{m})$ can acquire a solution more than $m(1 - \frac{1}{m}) = m - 1$. That is to say, the solutions will include all edges. However, if the chromatic index is 4, these algorithms with $k = 3$ can at most contain $m - 1$ edges. Thus, we can use these algorithms to determine whether the chromatic index is 3 or 4 for any cubic graph, which is contradictory to the Lemma 3. ∎

In all, the above analysis shows that the k-weight limited matching problem is much more difficult compared with existing problems. Thus, designing an approximation algorithm with low time complexity and bounded approximation ratio is far from trivial and urgently needed.

IV. Algorithm Design

This section we propose an algorithm based on the classic greedy-iteration algorithm [21]. Furthermore, the time complexity and approximation ratio are analyzed.

A. Improved Greedy-Iteration Algorithm

This section presents the approximation algorithm based on greedy-iteration [27] for ***Leaf*** as shown in Alg. 1. The inputs of proposed algorithm require a simple, undirected, edge-weighted graph G abstracted from real datacenter topology, and the number of OCSes k. We need to find k-weight limited matchings (*i.e.*, configurations) and how to allocate traffic volume for each fiber of the corresponding OCS as outputs.

For each OCS i, we first initialize the matching set M^i as an empty set. Then, we sort all edges $e \in E$ according to their current weights and all fibers $l \in L^i$ according to their capacity, both in the descending order (Lines 1-4). For each fiber l, we should find a suitable edge for it (Lines 5-22). First, the edge should not be adjacent to any edges in M^i (Lines 9-13). We use $flag_2 \in \{0, 1\}$ to represent whether this edge is adjacent to any edges in M^i ($flag_2 = 1$) or not ($flag_2 = 0$). If not, we then assign the weight of edge to the fiber, and update the residual weight (Lines 14-18). If so, we will check the next edge (Lines 19-20). However, if we cannot find one more suitable edge, it means current matching is complete for the OCS i. We should break current loop and search a matching for next OCS (Lines 23-25). We use $flag_1 \in \{0, 1\}$ to represent whether there is a suitable edge for M^i ($flag_1 = 1$) or not ($flag_1 = 0$). Finally, the algorithm returns k-weight limited matchings M^i and traffic volume on fibers of each OCS $x^i_{e,l}$.

Algorithm 1 Improved Greedy-Iteration algorithm

```
Require: A simple, undirected, edge-weighted graph $G$, and OCS set $i \in I$
1:  for $i \in I$ do
2:      Initialize $M^i = \emptyset$
3:      Sort all edges $e \in E$ according to $D(e)$ in the descending order
4:      Sort all fibers $l \in L^i$ according to $c(l)$ in the descending order
5:      for $l \in L^i$ do
6:          $flag_1 \leftarrow 0$
7:          for $e \in E$ do
8:              $flag_2 \leftarrow 0$
9:              for $e' \in M^i$ do
10:                 if $e \cap e' \neq \emptyset$ then
11:                     $flag_2 \leftarrow 1$
12:                 end if
13:             end for
14:             if $flag_2 = 0$ then
15:                 $flag_1 \leftarrow 1$
16:                 $M^i \leftarrow M^i \cup e$
17:                 $x^i_{e,l} \leftarrow \max[c(l), D(e)]$
18:                 $D(e) \leftarrow \max[0, D(e) - c(l)]$
19:             else
20:                 Continue
21:             end if
22:         end for
23:         if $flag_1 = 0$ then
24:             Break
25:         end if
26:     end for
27: end for
28: Return $\{M^i\}$ and $\{x^i_{e,l}\}$
```

B. Performance Analysis

In this section, we analyze the time complexity and approximation ratio of the proposed algorithm.

Theorem 4: The time complexity of algorithm is $\mathcal{O}(km(\log m + n))$, where k is the number of OCSes and m is the number of communication demands (edges) and n is the number of ToR switches (vertices).

Proof: The algorithm involves k iterations, each of which consists of following three parts: sorting edges and fibers, selecting an edge for a fiber, determining whether an edge is adjacent to current matching set M^i. The running time of each part is $\mathcal{O}(sort(m))$, $\mathcal{O}(m|L^i|)$, $\mathcal{O}(|M^i|)$, respectively. Here $|L^i|$ represents the number of fibers, and $|M^i|$ means the number of edges in a matching, which are both $\mathcal{O}(n)$. Thus, the total time complexity is $\mathcal{O}(k(sort(m) + mn + n))$. Here $\mathcal{O}(sort(m))$ represents the time complexity of a sorting algorithm. The time complexity can be regarded as $\mathcal{O}(k(m \log m + mn + n)) = \mathcal{O}(km(\log m + n))$, if a comparison-based sorting algorithm is adopted [28]. ∎

Theorem 5: The approximation ratio of Alg. 1 is $\frac{1}{2}$, when the minimum capacity of all fibers is larger than the maximum weight of all edges.

Due to limited space, the proof is omitted here.

Theorem 6: The $\frac{1}{2}$-approximation ratio is a tight bound.

Proof: Consider a 2-dimensional hypercube graph as shown in Fig. 4. We define the nodes denoted from a to h

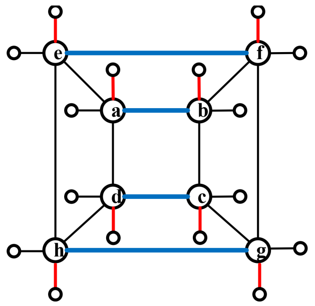


Fig. 4. The 2-dimensional hypercube. The solution of proposed algorithm is colored by blue, and the optimal solution is colored by red.

as hypercube nodes. Every hypercube node has two adjacent non-hypercube nodes. Let the edge connecting two hypercube nodes be the hypercube edge, and the edge connecting one hypercube node and one non-hypercube node be the non-hypercube edge. Assume that the weight of all non-hypercube edges is 1, and the weight of all hypercube edges is $1+\epsilon$, where ϵ is an arbitrarily small positive value. Our algorithm can acquire the solution colored by blue, including 4 hypercube edges with total weight $4+\epsilon$, while the optimal solution is colored by red, including 8 non-hypercube edges with total weight 8. The approximation ratio is $\frac{4+4\epsilon}{8} = \frac{1}{2} + \frac{\epsilon}{2}$. When $\epsilon \to 0$, the approximation ratio $\frac{1}{2} + \frac{\epsilon}{2} \to \frac{1}{2}$. ■

V. Evaluation

This section presents the simulation experiments to compare ***Leaf*** with existing works. All experiments are performed on a machine with 11th Gen Intel(R) CORE(TM) i7-11800H clocked at 2.3GHz and 32.0 GB of RAM.

A. Simulation Settings

1) Benchmarks: We compare ***Leaf*** with the following state-of-the-arts solutions.

- **Single OCS solution (SOCS).** This solution is adopted by the work [11], [8] and [20]. In specific, given a traffic matrix for pairs of racks, Edmonds' Blossom algorithm is directly applied to acquire the maximum weighted matching. In this paper, we use the function *max_weight_matching* of Python package *NetworkX* of version 3.1 [29] which is based on the Blossom algorithm.
- **Multiple OCSes solution (MOCS)**. We consider two MOCS solutions in this paper. The first one is originally proposed in [22], *i.e.*, the k-DM problem and its Blossom-iteration algorithm, denoted by *MOCS-ori*. Moreover, since the Helios proposed in [13] consider the capacity of fibers, we combine MOCS-ori with Helios, taking the capacity of fibers into consideration. We use *MOCS-im* to denote the new method.

2) Traffic Trace Datasets: Our simulations are based on both real-world and synthetic traffic traces, which have been widely adopted in previous works for evaluation.

- **Microsoft.** This traffic trace and topology is proposed by Microsoft Research in the work [14], and reproduced in [30]. This topology includes 600 racks, and the flows are generated with a Poisson arrival rate. The size of a flow is based on distributions in previous work [31]. The source and destination racks are based on production clusters.
- **Graph 500**. The Graph 500 Recursive-MATrices (RMAT) is used to generate random graphs with specific edge weight by varying the RMAT parameters [32]. In this paper, we adopt the parameters $(0.55, 0.15, 0.15, 0.15)$, which is evaluated by the works [33], [22], and [21], denoted by *rmat_b*. Specifically, rmat_b includes 1024 vertices and 6787 edges. The weights of edges follow an exponential distribution with values ranging from 1 to 500000.

3) OCS Settings: We consider Glimmerglass 64-port OCSes [13]. The capacity of fibers ranges in $\{20, 40, 60, 80, 100\}$ Gbps. According to [22], the number of OCSes is chosen from $k \in \{2, 4, 8, 16, 32, 64\}$.

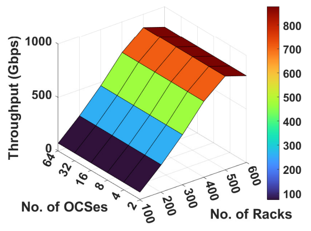
4) Performance Metrics: We adopt the following two important metrics related to QoS, which are also adopted in the previous works [22] and [21].

- **Actual Throughput.** Since we take the limited and non-uniform fiber capacity into consideration, the theoretical traffic volume of one configuration can only be *partially* calculated as actual throughput, discussed in Section II-C.
- **Running Time**. The reconfigurable time is the key to deal with traffic dynamics in datacenters. Thus we record the start and end time stamp of executing the four algorithms to measure the running time of each one.

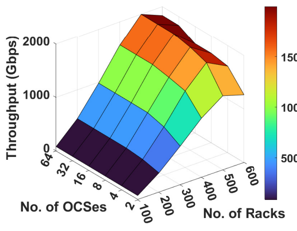
B. Performance Comparison

We conduct two groups of experiments to evaluate the performance of ***Leaf***. The first group of experiments compares the actual throughput by varying the number of racks and OCSes in datacenters. Results are shown in Figs. 5-6.

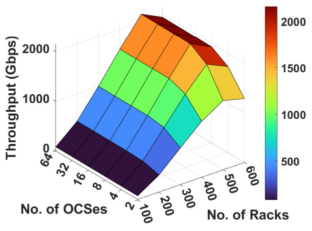
Overall, ***Leaf*** can always acquire the highest throughput results compared with SOCS, MOCS-ori and MOCS-im. For instance, in Fig. 5, with the Microsoft dataset, given 32 OCSes and 600 racks, the throughput results of SOCS, MOCS-ori, MOCS-im and ***Leaf*** are 809.61 Gbps, 1831.93 Gbps, 1939.42 Gbps and 2603.09 Gbps. ***Leaf*** improves the throughput by 68.92%, 42.16% and 34.24%, compared with SOCS, MOCS-ori and MOCS-im, respectively. In Fig. 6, with the Graph 500 dataset, given 8 OCSes and 300 racks, the throughput results of SOCS, MOCS-ori, MOCS-im and ***Leaf*** are 705.81 Gbps, 1348.81 Gbps, 1657.01 Gbps and 4565.49 Gbps. ***Leaf*** improves the throughput by 547.51%, 238.65% and 175.49%, compared with SOCS, MOCS-ori and MOCS-im, respectively. The reason why ***Leaf*** always acquires the highest throughput results is that we take many practical facts into consideration. Compared with SOCS, ***Leaf*** can deploy multiple OCSes for higher throughput. Compared with MOCS-ori, ***Leaf*** can



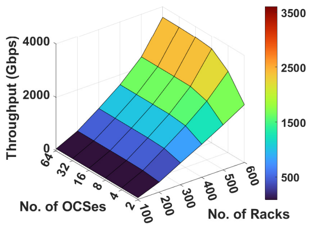
(a) SOCS.



(b) MOCS-ori.

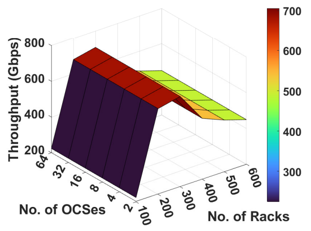


(c) MOCS-im.

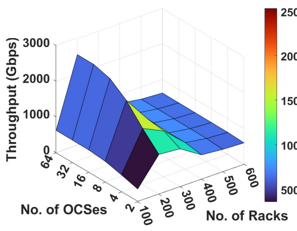


(d) ***Leaf***.

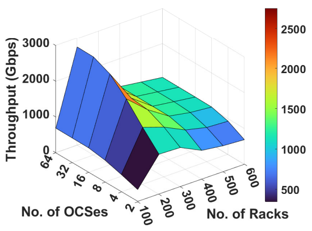
Fig. 5. Actual Throughput vs. Number of Racks and Number of OCSes with the Microsoft dataset.



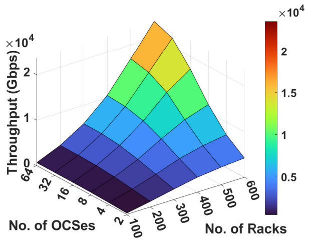
(a) SOCS.



(b) MOCS-ori.



(c) MOCS-im.



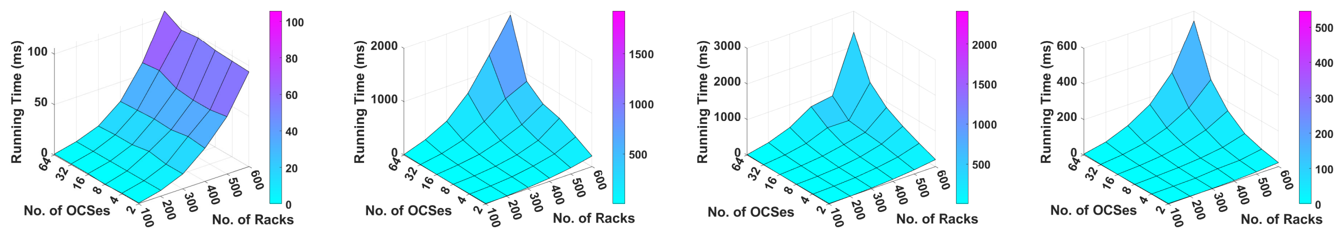
(d) ***Leaf***.

Fig. 6. Actual Throughput vs. Number of Racks and Number of OCSes with the Graph 500 dataset.

carry more traffic between two racks. MOCS-ori overlooks the limited capacity of fibers causing severe wasted traffic. Although MOCS-im is capacity-aware, this solution can not place fibers with non-uniform capacity properly. Thus, the throughput results of MOCS-im are only slightly higher.

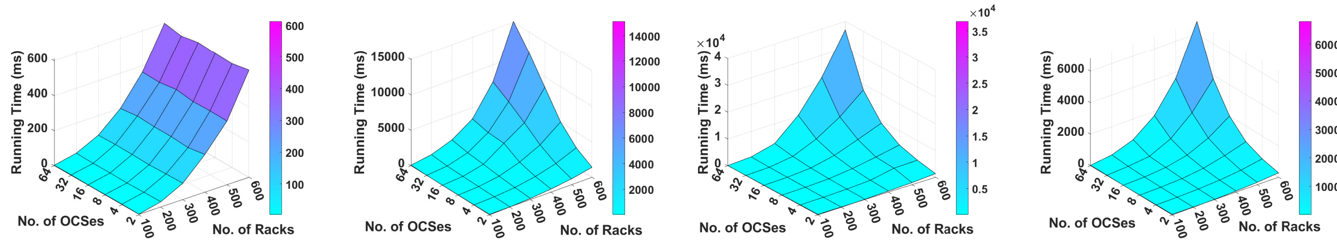
Another interesting and worth-noting observation is that the growth trends of SOCS, MOCS-ori and MOCS-im differ in terms of the number of racks and the number of OCSes. Specifically, with a specific number of racks, given more OCSes, the throughput will increase but tend to be steady after a threshold. For instance, in Fig. 5, with 500 racks, SOCS, MOCS-ori and MOCS-im can achieve 876.05 Gbps, 1995.88 Gbps and 2153.27 Gbps of throughput respectively with 16 OCSes, given the Microsoft dataset. And when the number of OCSes reaches 32 and 64, the throughput results of three solutions remain the same. Specially, since SOCS only deploys one OCS, the varying number of OCSes does not affect the throughput results of SOCS. As comparison, for a specific number of OCSes, when given more racks, the throughput results of SOCS, MOCS-ori and MOCS-im will decrease after reaching a threshold. For instance, in Fig. 5, when there are 600 racks and 16 OCSes, the throughput results of SOCS, MOCS-ori and MOCS-im are 809.61 Gbps, 1673.67 Gbps and 1939.417 Gbps, respectively. But with 500 racks, the corresponding throughput results are 876.5 Gbps, 1995.88 Gbps and 2153.27 Gbps, respectively. Thus, the throughput results of 600 racks are lower than those of 500 racks. And we can also learn that in Fig. 6, the throughput results of SOCS, MOCS-ori and MOCS-im are decreasing when the number of racks exceeds 200. The reason is that when the size of datacenter is larger, the matchings acquired by the algorithms also become much more complex. SOCS, MOCS-ori and MOCS-im does not take non-uniform capacity of fibers into consideration and tend to place fibers to carry the traffic demand with less traffic to avoid waste. Thus, the communication demand with traffic are neglected, causing the throughput results to decrease. However, ***Leaf*** takes the limited and non-uniform fibers into consideration, which is why the throughput of ***Leaf*** grows when the numbers of racks increases. For instance, with the Microsoft dataset, given 4 OCSes, the throughput results of ***Leaf*** are 81.76 Gbps, 455.33 Gbps, 999.93 Gbps, 1564.97 Gbps, 2207.43 Gbps and 3039.14 Gbps, when there are 100, 200, 300, 400, 500, 600 racks, respectively. Furthermore, in Fig. 5(d), with a specific number of racks, the throughput results of ***Leaf*** also tend to be steady when given more OCSes, with the Microsoft dataset. Specifically, when there are 400 racks, the throughput results tend to be steady when the number of OCSes reaches 8, *i.e.*, the threshold is 8. And when given 500 and 600 racks, the throughput results tend to be steady when the numbers of OCSes reach 16 and 32, respectively, which means the thresholds become 16 and 32, respectively. This is because as the datacenter network size grows, the size of traffic demand also grows. Therefore, the threshold of OCSes' number also enlarges. However, in Fig. 6(d), it seems that the throughput results of ***Leaf*** increase along with the number of OCSes without a threshold. It is because the Graph 500 dataset is more traffic demand-intensive, *i.e.*, there exist much more edges in its formulated graph. Thus, we cannot observe the steady trend with only 64 OCSes.

The second group of experiments compares the running time by varying the number of racks and the number of OCSes in a datacenter to check the efficiency of ***Leaf***. The results are shown in Figs. 7-8. Overall, the running time results of MOCS-



(a) SOCS. (b) MOCS-ori. (c) MOCS-im. (d) ***Leaf***.

Fig. 7. Running Time vs. Number of Racks and Number of OCSes with Microsoft datasets.



(a) SOCS. (b) MOCS-ori. (c) MOCS-im. (d) ***Leaf***.

Fig. 8. Running Time vs. Number of Racks and Number of OCSes with Graph 500 datasets.

ori, MOCS-im and ***Leaf*** increase with the growing numbers of racks and OCSes, while the running time results of SOCS only increase along with the number of racks. It is worth noting that SOCS always achieve the lowest running time and the running time results of SOCS does not affected by number of OCSes. For instance, in Fig. 8, with the Graph 500 dataset, given 400 racks, the running time results of SOCS are around 200 ms. And when there are 500 racks, the running time is around 360 ms. The reason why the running time of SOCS is the lowest is that SOCS only needs to acquire one matching configuration for single OCS, while other three solutions need to acquire multiple configurations for multiple OCSes.

Furthermore, we can also learn from these figures that ***Leaf*** always acquires the second lowest running time results. For instance, in Fig. 7, with the Microsoft dataset, when there are 32 OCSes and 600 racks, the running time results of SOCS, MOCS-ori, MOCS-im and ***Leaf*** are 96.68 ms, 863.23 ms, 1262.77 ms and 268.70 ms, respectively. In Fig. 8, with the Graph 500 dataset, when there are 16 OCSes and 400 racks, the running time results of SOCS, MOCS-ori, MOCS-im and ***Leaf*** are 205.94 ms, 2750.83 ms, 2758.33 ms and 485.52 ms, respectively. Compared with MOCS-ori and MOCS-im, ***Leaf*** reduces the running time by 82.36% and 82.41%, respectively. We should note that, the theoretical reason for the running time results of ***Leaf*** is the complexity of proposed algorithm. Since the algorithm of ***Leaf*** is based on the efficient greedy algorithm, and the time complexity is only $\mathcal{O}(km(\log m+n))$, while the algorithms of MOCS-ori and MOCS-im are both based on Blossom algorithm, the time complexity of which is $\mathcal{O}(kn(n\log n+m))$. Here k, n, m is the number of OCSes, the number of nodes and the number of edges, respectively. Thus, it is obvious that our proposed algorithm is less time consuming and can perform well when facing traffic dynamics.

In conclusion, the two groups of experiments check both the throughput and efficiency of ***Leaf***. The experiments show the superior performance of ***Leaf*** with different datacenter network sizes and different datasets. Overall, compared with SOCS, MOCS-ori and MOCS-im, ***Leaf*** can improve throughput by about 547.51%, 238.65% and 175.49%, respectively. In addition, ***Leaf*** can reduce the running time by 82.36% and 82.41%, compared with MOCS-ori and MOCS-im, respectively. Therefore, in terms of QoS, ***Leaf*** can bring a significant improvement compared with existing solutions.

VI. Conclusion

This paper presents ***Leaf***, making reconfigurable datacenters more practical and efficient. We formulate the problem as a k-weight limited matching problem, and we also prove its $\mathcal{NP}$-hardness and inapproximability. Based on the classical greedy-iteration algorithm, we propose a new approximation algorithm with bounded approximation ratio and low time complexity. Furthermore, we conduct extensive experiments based on both real-world and synthetic datasets to show the superior performance on throughput and running time.

Acknowledgment

The corresponding author of this paper is Gongming Zhao. This article was supported in part by the National Science Foundation of China (NSFC) under Grant 62372426 and Grant 62102392, in part by the National Science Foundation of Jiangsu Province under Grant BK20210121, in part by the Hefei Municipal Natural Science Foundation under Grant 2022013, in part by the Youth Innovation Promotion Association of Chinese Academy of Sciences under Grant 2023481.

References

[1] M. Khani, M. Ghobadi, M. Alizadeh, Z. Zhu, M. Glick, K. Bergman, A. Vahdat, B. Klenk, and E. Ebrahimi, "Sip-ml: High-bandwidth optical network interconnects for machine learning training," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, ser. SIGCOMM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 657–675. [Online]. Available: https://doi.org/10.1145/3452296.3472900

[2] J. Wang, G. Zhao, H. Xu, Y. Zhao, X. Yang, and H. Huang, "Trust: Real-time request updating with elastic resource provisioning in clouds," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, 2022, pp. 620–629.

[3] W. M. Mellette, A. C. Snoeren, and G. Porter, "P-fattree: A multi-channel datacenter network topology," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, ser. HotNets '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 78–84. [Online]. Available: https://doi.org/10.1145/3005745.3005746

[4] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: A high performance, server-centric network architecture for modular data centers," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, p. 63–74, aug 2009. [Online]. Available: https://doi.org/10.1145/1594977.1592577

[5] J. Kim, W. J. Dally, and D. Abts, "Flattened butterfly: A cost-efficient topology for high-radix networks," *SIGARCH Comput. Archit. News*, vol. 35, no. 2, p. 126–137, jun 2007. [Online]. Available: https://doi.org/10.1145/1273440.1250679

[6] M. Nance Hall, K.-T. Foerster, S. Schmid, and R. Durairajan, "A survey of reconfigurable optical networks," *Optical Switching and Networking*, vol. 41, p. 100621, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1573427721000187

[7] C. Griner, J. Zerwas, A. Blenk, M. Ghobadi, S. Schmid, and C. Avin, "Cerberus: The power of choices in datacenter topology design - a throughput perspective," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 5, no. 3, dec 2021. [Online]. Available: https://doi.org/10.1145/3491050

[8] G. Wang, D. G. Andersen, M. Kaminsky, M. Kozuch, T. Ng, K. Papagiannaki, M. Glick, and L. Mummert, "Your data center is a router: The case for reconfigurable optical circuit switched paths," 2009.

[9] J. Zerwas, C. Györgyi, A. Blenk, S. Schmid, and C. Avin, "Duo: A high-throughput reconfigurable datacenter network using local routing and control," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 7, no. 1, mar 2023. [Online]. Available: https://doi.org/10.1145/3579449

[10] K.-T. Foerster and S. Schmid, "Survey of reconfigurable data center networks: Enablers, algorithms, complexity," *SIGACT News*, vol. 50, no. 2, p. 62–79, jul 2019. [Online]. Available: https://doi.org/10.1145/3351452.3351464

[11] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. E. Ng, M. Kozuch, and M. Ryan, "C-through: Part-time optics in data centers," in *Proceedings of the ACM SIGCOMM 2010 Conference*, ser. SIGCOMM '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 327–338. [Online]. Available: https://doi.org/10.1145/1851182.1851222

[12] X. Ye, Y. Yin, S. J. B. Yoo, P. Mejia, R. Proietti, and V. Akella, "Dos: A scalable optical switch for datacenters," in *Proceedings of the 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ser. ANCS '10. New York, NY, USA: Association for Computing Machinery, 2010. [Online]. Available: https://doi.org/10.1145/1872007.1872037

[13] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: A hybrid electrical/optical switch architecture for modular data centers," in *Proceedings of the ACM SIGCOMM 2010 Conference*, ser. SIGCOMM '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 339–350. [Online]. Available: https://doi.org/10.1145/1851182.1851223

[14] M. Ghobadi, R. Mahajan, A. Phanishayee, N. Devanur, J. Kulkarni, G. Ranade, P.-A. Blanche, H. Rastegarfar, M. Glick, and D. Kilper, "Projector: Agile reconfigurable data center interconnect," in *Proceedings of the 2016 ACM SIGCOMM Conference*, ser. SIGCOMM '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 216–229. [Online]. Available: https://doi.org/10.1145/2934872.2934911

[15] H. Wang, X. Yu, H. Xu, J. Fan, C. Qiao, and L. Huang, "Integrating coflow and circuit scheduling for optical networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 6, pp. 1346–1358, 2019.

[16] R. Jiang, T. Zhang, and C. Yi, "Effective coflow scheduling in hybrid circuit and packet switching networks," in *2023 IEEE Symposium on Computers and Communications (ISCC)*, 2023, pp. 1156–1161.

[17] R. Urata, H. Liu, K. Yasumura, E. Mao, J. Berger, X. Zhou, C. Lam, R. Bannon, D. Hutchinson, D. Nelson, L. Poutievski, A. Singh, J. Ong, and A. Vahdat, "Apollo: Large-scale deployment of optical circuit switching for datacenter networking," in *2023 Optical Fiber Communications Conference and Exhibition (OFC)*, 2023, pp. 1–3.

[18] S. Zhang, R. Kraemer, X. Xue, N. Tessema, H. Freire Santana, E. Tangdiongga, and N. Calabretta, "Photonic integrated multicast switch-based optical wireless data center network," *Journal of Optical Communications and Networking*, vol. 15, no. 7, pp. C54–C62, 2023.

[19] J. Edmonds, "Matroids and the greedy algorithm," *Mathematical programming*, vol. 1, pp. 127–136, 1971.

[20] H. Liu, M. K. Mukerjee, C. Li, N. Feltman, G. Papen, S. Savage, S. Seshan, G. M. Voelker, D. G. Andersen, M. Kaminsky, G. Porter, and A. C. Snoeren, "Scheduling techniques for hybrid circuit/packet networks," 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/10.1145/2716281.2836126

[21] K. Hanauer, M. Henzinger, L. Ost, and S. Schmid, "Dynamic demand-aware link scheduling for reconfigurable datacenters," in *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, 2023, pp. 1–10.

[22] K. Hanauer, M. Henzinger, S. Schmid, and J. Trummer, "Fast and heavy disjoint weighted matchings for demand-aware datacenter topologies," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, 2022, pp. 1649–1658.

[23] A. Shoemaker and S. Vare, "Edmonds' blossom algorithm," *CME*, 2016.

[24] P. J. Winzer, "Scaling optical fiber networks: Challenges and solutions," *Optics and Photonics News*, vol. 26, no. 3, pp. 28–35, 2015.

[25] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, p. 123–137, aug 2015. [Online]. Available: https://doi.org/10.1145/2829988.2787472

[26] I. Holyer, "The np-completeness of edge-coloring," *SIAM Journal on Computing*, vol. 10, no. 4, pp. 718–720, 1981. [Online]. Available: https://doi.org/10.1137/0210055

[27] T. Campbell and T. Broderick, "Bayesian coreset construction via greedy iterative geodesic ascent," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 698–706. [Online]. Available: https://proceedings.mlr.press/v80/campbell18a.html

[28] K. Iwama and J. Teruyama, "Improved average complexity for comparison-based sorting," *Theoretical Computer Science*, vol. 807, pp. 201–219, 2020, in memory of Maurice Nivat, a founding father of Theoretical Computer Science - Part II. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304397519304487

[29] A. Hagberg and D. Conway, "Networkx: Network analysis with python," *URL: https://networkx. github. io*, 2020.

[30] M. Bienkowski, D. Fuchssteiner, J. Marcinkowski, and S. Schmid, "Online dynamic b-matching: With applications to reconfigurable datacenter networks," *SIGMETRICS Perform. Eval. Rev.*, vol. 48, no. 3, p. 99–108, mar 2021. [Online]. Available: https://doi.org/10.1145/3453953.3453976

[31] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "Pfabric: Minimal near-optimal datacenter transport," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, ser. SIGCOMM '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 435–446. [Online]. Available: https://doi.org/10.1145/2486001.2486031

[32] R. C. Murphy, K. B. Wheeler, B. W. Barrett, and J. A. Ang, "Introducing the graph 500," *Cray Users Group (CUG)*, vol. 19, pp. 45–74, 2010.

[33] A. Khan, A. Pothen, M. Mostofa Ali Patwary, N. R. Satish, N. Sundaram, F. Manne, M. Halappanavar, and P. Dubey, "Efficient approximation algorithms for weighted \$b\$-matching," *SIAM Journal on Scientific Computing*, vol. 38, no. 5, pp. S593–S619, 2016. [Online]. Available: https://doi.org/10.1137/15M1026304