Regularization-Based Coflow Scheduling in Optical Circuit Switches

Haisheng Tan[®], *Senior Member, IEEE*, Chi Zhang[®], Chao Xu[®], Yupeng Li[®], *Member, IEEE, ACM*, Zhenhua Han[®], *Member, IEEE*, and Xiang-Yang Li[®], *Fellow, IEEE, ACM*

Abstract—To improve the application-level data efficiency, the scheduling of coflows, defined as a collection of parallel flows sharing the same objective, is prevailing in recent data centers. Meanwhile, optical circuit switches (OCS) are gradually applied to provide high data rate with low power consumption. However, so far few research outputs have covered the flow, let alone the coflow, scheduling in the context of OCS. In this work, we investigate coflow scheduling in OCS-based data centers. We first derive a novel operation called *regularization* processed respectively on the flow traffic demands and the flow start times, which can be efficiently implemented and reduce the circuit reconfiguration frequency dramatically. We then propose a 2-approximation algorithm, called Reco-Sin, for single coflow scheduling to minimize the coflow completion time (CCT). For multiple coflows, we derive Reco-Mul to minimize the total weighted CCT, which can transform any non-preemptive multi-coflow scheduling in packet switches to a scheduling scheme in OCS. Reco-Mul can achieve a constant approximation under the assumption that no tiny flows will be transmitted in OCS. To get rid of this assumption, we present another multiple coflow scheduling scheme, named Reco-Mul+, which has an approximation ratio of O(K). Here, K is the total number of coflows. Extensive simulations based on Facebook data traces show that our approaches outperform state-of-the-art schemes significantly, i.e., one single coflow can be finished up to $1.97 \times$ faster with Reco-Sin, and multiple coflows can be completed up to more than $2 \times$ faster with Reco-Mul and Reco-Mul+.

Index Terms—Coflow Scheduling, Optical Circuit Switch, Approximation Algorithm.

I. INTRODUCTION

IN CURRENT distributed computing frameworks (*e.g.*, MapReduce [2], Dryad [3] and Spark [4]), communication

Manuscript received April 25, 2020; revised October 8, 2020 and January 6, 2021; accepted February 4, 2021; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor H. Shen. Date of publication February 19, 2021; date of current version June 16, 2021. This work was supported in part by the National Key R&D Program of China under Grant 2018YFB0803400, in part by NSFC under Grant 61772489 and Grant 61751211, in part by the Key Research Program of Frontier Sciences (CAS) under Grant QYZDY-SSW-JSC002, and in part by the project of "FANet: PCL Future Greater-Bay Area Network Facilities for Large-Scale Experiments and Applications under Grant LZC0019. A preliminary version of this work titled "Reco: Efficient Regularization-Based Coflow Scheduling in Optical Circuit Switches" was published in Proc. of the 39th IEEE International Conference on Distributed Computing Systems (ICDCS 2019), Dallas, Texas, USA, July 2019. (*Corresponding author: Xiang-Yang Li.*)

Haisheng Tan, Chi Zhang, Chao Xu, and Xiang-Yang Li are with the LINKE Lab and CAS Key Laboratory of Wireless-Optical Communications, University of Science and Technology of China (USTC), Hefei 230052, China (e-mail: hstan@ustc.edu.cn; gzhnciha@mail.ustc.edu.cn; chaoxu94@mail.ustc.edu.cn; xiangyangli@ustc.edu.cn).

Yupeng Li is with the Department of Journalism, Hong Kong Baptist University, Hong Kong (e-mail: ivanypli@gmail.com).

Zhenhua Han is with Mircosoft Research Asia, Shanghai 200232, China (e-mail: hzhua201@gmail.com).

Digital Object Identifier 10.1109/TNET.2021.3058164

data flows in the network may share a common performance objective as they are likely to correspond to the same job from one application. To capture this kind of application-level requirements, a new abstraction of structured data flows, called *coflow*, is proposed to allow applications to convey their semantics to the network [5]. A coflow is a collection of related parallel individual flows that occur typically between two stages of a multi-stage computing task, such as shuffling in the MapReduce. To improve the application-level performance, instead of analyzing a single flow, we turn to optimize the *coflow completion time* (CCT), defined as the duration from its arrival to the completion of all the individual flows.

Based on the development of micro-electro-mechanical system (MEMS) techniques, optical circuit switches (OCS), have become commercially viable in large-scale computing systems [6]. Compared with electrical packet switch, an OCS has much higher data transfer rate but lower power consumption. Meanwhile, OCS has its own communication constraints. Each ingress or egress port in an OCS is restricted to establish at most one circuit for data transmission at a time, called *the port constraint*. A reconfiguration to establish new circuits in OCS will take a fixed period, called *the reconfiguration delay* and denoted as δ , typically in tens of microseconds [7].

In order to minimize the CCT, coflow scheduling, including single coflow scheduling (a.k.a. intra-coflow scheduling on individual flows within a coflow) and multiple coflow scheduling (a.k.a. inter-coflow scheduling), has been extensively studied in traditional electrical packet switches [8]–[11]. In OCS, individual flow scheduling has also been frequently studied (*e.g.* [7], [12]–[17]). However, there are few works on coflow scheduling in OCS. Sunflow [15] mainly focused on the single coflow and provided a simple heuristic policy for multiple coflows. Compared with packet switches, the problem of OCS-based coflow scheduling is challenging mainly due to:

- The port constraint can never be violated when establishing circuits for multiple flow transmission;
- Besides the transmission order of flows/coflows, we have to carefully establish and schedule the circuits to avoid frequent time-consuming circuit reconfigurations;

In this article, we investigate both the single and multiple coflow scheduling problem in OCS-based data centers, aiming at minimizing the CCT and the weighted average CCT, respectively. Our contributions can be summarized as follows:

• We propose a simple and effective operation, called *regularization*, processed respectively on the flow

1558-2566 © 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

Solutions	Coflow-aware	OCS-enable	Performance Guarantee
			for Mulitple Coflows
Varys [8], Aalo [9], CODA [10]	 ✓ 	×	×
Qiu et al. [18], Chen et al. [19],	 ✓ 	X	1
Wang et al. [20], Chowdhury et al. [21], Im et al. [22]			
Heilos [12], c-through [13], Porter et al. [7], Liu et al. [14]	X	1	×
Sunflow [15], Wang et al. [23], Co-scheduler [24]	1	1	×
Reco [this work]	1	1	1

TABLE I Comparison Among the Related Work and Our Results

traffic demands (Sec. IV-B) and the flow start times (Sec. V-A) so that the circuit reconfiguration frequency is reduced significantly. We further derive a scheme of regularization-based colfow scheduling algorithms in OCS.

- Firstly, we derive a single coflow scheduling algorithm, named Reco-Sin, to minimize the CCT with an approximation ratio of 2 (Sec. IV). To the best of our knowledge, this is the first constant approximation algorithm for the all-stop OCS.¹
- For multiple coflow scheduling, we further propose a novel algorithm, named Reco-Mul, to minimize the total weighted CCT (Sec. V). We theoretically prove that Reco-Mul can effectively transform *any* non-preemptive multi-coflow scheduling scheme in packet switches with an approximation ratio of Δ into a feasible OCS-based coflow scheduling with an approximation ratio of $\Delta \cdot \left(1 + \frac{1}{|\sqrt{c}|}\right)^2$. Here, we assume that a nonzero traffic demand in OCS is no less than $c \times \delta$, where *c* is a positive constant and δ is the circuit reconfiguration delay.²
- To get rid of the assumption on the sizes of the nonzero traffic demands, we propose another multiple coflow scheduling algorithm named Reco-Mul+, which can achieve O(K)-approximation ratio. K is the number of coflows. Reco-Mul+ can also reduce the calculation overhead before flow transmission in Reco-Mul.
- Based on real data traces from Facebook, we conduct extensive simulations to compare our algorithms with state-of-the-art schemes (Sec. VI). Our simulations demonstrate that the number of reconfigurations in Reco-Sin can be reduced to only 12.67% of that in the baselines, and hence Reco-Sin can transmit a single coflow up to 1.97× faster. As for multiple coflows, Reco-Mul and Reco-Mul+ outperform the baselines dramatically, which respectively transmit multiple coflows up to 2.33× and 2.76× faster on average.

Paper Organization: We present the related work including a comparison with our results in Sec. II. We formally define the system model and formulate the problem in Sec. III. Our algorithms for the single coflow and multiple coflow scheduling are proposed in Sec. IV and Sec. V, respectively. Extensive simulations are in Sec. VI. Discussion is presented in Sec. VII. We conclude the whole paper in Sec. VIII.

II. RELATED WORK

In this section, we present related works on circuit scheduling and coflow scheduling. We list some of the most representative related works and make a comparison in Table I.

A. Circuit Scheduling

Helios [12] and c-Through [13] proposed hybrid circuit/packet switch architectures. Both focused on the circuit management to minimize the flow completion time. They adopted the well-known Edmonds maximum weighted matching algorithm to schedule the circuits [25]. Later, Porter et al. [7] proposed an improved circuit scheduling algorithm, called Traffic Matrix Scheduling (TMS). Borrowing the idea of BvN [26], [27], their algorithm decomposed the traffic demand matrix into permutation matrix associated with a series of circuit assignments for a predetermined period of time. However, BvN decomposition did not work well in minimizing the number of configurations, which could incur unavoidable and significant reconfiguration delays as we claimed in Sec. IV-A. Paper [28] achieved an approximation algorithm for circuit scheduling under specific settings, where the demands and reconfiguration time were integers, and the minimum demand transmission time was assumed as an integral multiple of the reconfiguration time. Liu et al. [14] developed a flow scheduling algorithm called Solstice to minimize the maximum flow completion time particularly for the hybrid circuit/packet networks. Solstice effectively improved the circuit utilization to reduce the number of configurations, and thus outperformed BvN. Our algorithm Reco-Sin, compared with Solstice, reduces the CCT with even less frequent reconfigurations. and guarantees a performance with a constant approximation ratio.

B. Coflow Scheduling

Most previous works on network-level optimization were agnostic on the application-level performance metrics (*e.g.*, [29]). which introduces negative impact on the application performance (examples can be found in [5], [8], [9]). The *coflow* abstraction was first proposed in [5] to bridge the above gap. Some existing works on coflows focused on minimizing the average CCT (see [5], [8]–[11], [18], [19], [30]–[34]). Varys [8] proposed effective heuristics to schedule coflows aiming at minimizing the average CCT. Without prior knowledge of coflows, Aalo [9] adopted Discretized Coflow-Aware

¹In the all-stop model, a circuit reconfiguration will halt all circuits established in the OCS until the reconfiguration is completed.

²This assumption is reasonable as in practice only elephant flows are transferred through OCS while mice flows can be handled more efficiently by packet switches [14].

Qiu et al. [18] proposed the first deterministic algorithm with a constant approximation ratio for multiple coflow scheduling, which defined the flow completion time as the duration from a common original time point to the completion of the flow regardless of its real arrival time. Paper [35] further improved the approximation ratio in the setting that the coflows arrived to the network at the same time. Essentially, the solutions in [18] and [35] determined the scheduling order of the coflows based on the solutions to a carefully formulated linear program. Wang et al. [20] proposed a 2-approximation algorithm for minimizing weighted coflow completion time without solving LP. Li et al. [11] studied the coflow scheduling and routing problem, and proposed the first online algorithm with performance guarantees for the problem. Liang and Modiano [36] assumed stochastic coflow arrivals, and studied the optimal scaling of coflow-level delay in an $N \times N$ input-queued switch as $N \to \infty$. Chen *et al.* [19] studied the utility optimization for coflow scheduling, who proposed a utility-based scheduler to provide differential treatment to coflows with different degrees of sensitivity. Tian et al. [30] studied dependent coflows of multi-stage jobs to minimize total weighted job completion time. Utopia [31] focused on the near-optimal coflow scheduling with provable isolation guarantee. Sincronia [34] presented a near-optimal network design for coflows that can be implemented on top of any transport layer to support priority scheduling. Chowdhury et al. [21] studied multiple coflow scheduling on general graph and proposed a randomized algorithm with 2-approximation for the single path and free path cases. Im et al. [22] formulated the matroid coflow scheduling problem and got a tight 2-approximation algorithm for minimizing the weighted completion time.

C. Coflow Scheduling in OCS

As stated above, the multi-coflow scheduling problem for packet switches has been extensively studied in the literature. However, most results (*e.g.*, [8], [9], [11], [18]) are based on *bandwidth sharing*, where one ingress (egress) port can simultaneously be connected to multiple egress (ingress) ports to share the port bandwidth. Therefore, directly applying these algorithms in OCS will break the port constraint. Some other flow scheduling algorithms in packet switches, not bandwidth sharing based, can be applied to OCS directly, *e.g.*, Sincronia [34] and Shafiee *et al.* [35]. The drawback is that they will lead to very frequent circuit reconfigurations. Therefore, coflow scheduling in OCS should be carefully studied.

However, so far few research results have been reported for coflow scheduling in OCS. Huang *et al.* [15] might be the first to consider coflow scheduling in optical circuit switches, which presented a constant approximation for single coflow

	TABLE II
ME	IMPORTANT NOTATIC

Notations	Description
N	Number of OCS ports
δ	Reconfiguration delay
-C(u)	<i>u</i> -th circuit establishment
dur(u)	Duration of <i>u</i> -th circuit establishment
$\overline{(C(u), dur(u))}$	<i>u</i> -th circuit assignment
C	Circuit scheduling composed
	of some circuit assignments
ĸ	Set of coflows
w_k	Weight of coflow k
a_k	Arrival time of coflow k
D_k	Demand matrix of coflow k
$d_{i,i}^k$	Demand from <i>i</i> -th ingress port
0,5	to j -th egress port of coflow k
f_k	Finish time of coflow k
T_k	Completion time of coflow k
S	Coflow scheduling
$\gamma(D)$	Minimum number of decomposition
	matrices of demand matrix D
t_{trans}^k	Data transmission time of coflow k
t_{conf}^k	Reconfiguration time of coflow k
$\frac{\rho_k}{\rho_k}$	Maximum value of the sum of
	each row and column of D_k
τ_k	Maximum number of non-zero entries of
	each row and column of D_k
c	Optical transmission threshold
S_p	Non-preemptive coflow scheduling
	in packet switches
$\hat{\mathcal{S}}_o$	Regularized coflow scheduling
	without considering reconfiguration
S_o	Feasible coflow scheduling in OCS
$\eta(\hat{\mathcal{S}}_o, \hat{t})$	Number of reconfiguration required in \hat{S}_o
• • • • •	during the time period $[0, \hat{t}]$
	The parameter in regularized matrix D'
	corresponding to x in the original matrix D

scheduling and a heuristic to schedule multiple coflows in the not-all-stop model. Zhang *et al.* [37] further improved the result in the not-all-stop model, and gave a constant approximation algorithm for multiple coflow scheduling. Paper [38] investigated coflow scheduling in OCS under online settings, where a heuristic algorithm was proposed without any theoretical performance guarantee. Wang *et al.* [23] studied the integrated coflow and circuit scheduling problem with single coflow. Co-scheduler [24] studied the task placement problem in hybrid electrical/optical data center networks to minimize CCT. To the best of our knowledge, our algorithm, Reco is the first approximation algorithm for multiple coflow scheduling in all-stop OCS.

III. PROBLEM FORMULATION

In this section, we present the system model and formally define our coflow scheduling problem in OCS. The main notations used are listed in Table II.

A. System Model

Network Model: Similar to existing works ([8], [9], [15], [18]), the fabric of a data center network is abstracted as one non-blocking circuit switch with N ingress ports and N egress ports (Fig. 1). Data flows are buffered at senders to be transferred from the ingress to the egress ports.



Fig. 1. A 3×3 Network Model: Coflow 1 and 2 are to be transmitted, whose demand matrices are D_1 and D_2 , respectively.

Optical Circuit Switch: To transmit data from ingress port *i* to egress port *j*, OCS needs to establish a circuit between the two ports. Without loss of generality, the bandwidth of a circuit from one ingress to one egress port is normalized to 1, which is fully occupied by only one flow at a time. Recall that circuit establishment should satisfy the port constraint, that is to say, each ingress (egress) port can only establish a connection to one egress (ingress) port at a time. Denote the circuit reconfiguration delay as δ . Multiple circuits can be reconfigured simultaneously during one reconfiguration if they do not share a common port. Here, we adopt the allstop circuit switch model as [7], [13], [14], [16], where a circuit reconfiguration would halt all transmissions in the OCS.³ We define a *circuit establishment*, denoted by C(u), as all the circuits concurrently established in an OCS. Due to the port constraint, C(u) is in fact a matching in an $N \times N$ bipartite graph where the two disjoint vertex groups are the egress and ingress ports, respectively. The duration of the circuit establishment C(u) is denoted as dur(u). We call the pair (C(u), dur(u)) as a *circuit assignment*. Furthermore, a circuit scheduling C is composed of a sequence of circuit assignments $\{(C(1), dur(1)), \cdots, (C(m), dur(m))\}$, where m is the number of assignments. A circuit scheduling is valid if the port constraint is satisfied.

Coflow: A set \mathcal{K} of coflows are to be transmitted through the OCS. Denote $K = |\mathcal{K}|$ as the number of coflows. Each coflow $k \in \mathcal{K}$ has a non-negative weight w_k , which indicates its sensitivity to the latency. Denote a_k as the arrival time of coflow k. Similar to [8], [11], [18], we assume that the flows within one coflow arrive at the same time. An $N \times N$ matrix D_k is to denote the data transmission demand of coflow k, which is called the *demand matrix*. Each entry $d_{i,j}^k$ in this matrix represents the amount of data to be transferred from ingress port i to egress port j. For example, in Fig. 1, $d_{2,3}^2 = 3$ means the data amount in Coflow 2 to transmit from ingress 2 to egress 3 is 3. Since the bandwidth of circuits is normalized to 1, without ambiguity, we also denote $d_{i,j}^k$ as the time needed to transmit all demands from ingress port i to egress port j. As stated in Sec. I, we assume that there is no tiny traffic demand in OCS, *i.e.*, $d_{i,j}^k \ge c \cdot \delta$ for any *i*, *j* and *k*, where *c* is a constant called the optical transmission threshold. We set $f_{i,j}^k$ as the time when the transmission demand from port i to j in coflow k is completed. Then, the time that the whole coflow k is finished is defined as $f_k = \max_{i,j} f_{i,j}^k$. Hence, the coflow

 3 We will show that our results for multiple coflow scheduling can be extended to the *not-all-stop model* (See in Sec. VII).

completion time (CCT) of coflow k, denoted as T_k , is the duration from its arrival to the completion of transmission, calculated as $T_k = f_k - a_k$. Since we consider the data flows have been buffered in the senders, the arrival time of all coflows can be assumed to be the same similar to [8], [11] [18], *i.e.*, $a_k = 0$, $\forall k$.

B. Problem Statement

We next formally define coflow scheduling problems for a single coflow and multiple coflows, respectively.

Problem 1 (Single Coflow Scheduling): Given a single coflow with demand matrix D arriving at a network which is modeled as an $N \times N$ non-blocking OCS, the problem is to find a feasible coflow scheduling so that its CCT is minimized.

Note that as there is only one coflow in the network (with demand matrix D), a circuit scheduling C can be directly transformed into a coflow scheduling S, *i.e.*, at each time, scheduling the traffic demand in D to transmit along all the currently established circuits. Therefore, Problem 1 is equivalent to finding a valid circuit scheduling to finish all the traffic demands in D with the minimum time, which is composed of two parts, the aggregate flow transmission time and and the total delay caused by reconfigurations.

Problem 2 (Multiple Coflow Scheduling): In a network modeled as an $N \times N$ non-blocking OCS, a set \mathcal{K} of coflows arrive with demand matrices $D_k, \forall k \in \mathcal{K}$. The problem is to find a feasible coflow scheduling such that the total weighted coflow completion time, $\sum_{k \in \mathcal{K}} w_k T_k$, is minimized.

The coflow scheduling problem has already been proved NP-hard [15], [35]. In the following, we will propose our novel efficient algorithms to achieve approximate results.

IV. SINGLE COFLOW SCHEDULING

The main challenge for single coflow scheduling in OCS is how to effectively decrease the reconfiguration frequency. We leverage the classical Birkhoff-von Neumann (BvN) decomposition, and design a novel operation, called *regularization*.

A. Birkhoff-Von Neumann Decomposition

According to Birkhoff's Theorem [25], any doubly stochastic matrix⁴ can be decomposed into a set of permutation matrices with specific coefficients by Birkhoff-von Neumann (BvN) decomposition. An example can be found in Fig. 2. Under BvN decomposition, an $N \times N$ matrix can be decomposed to $m \leq N^2 - 2N + 2$ permutation matrices [39]. It is NP-hard to compute the BvN decomposition with the minimum permutation matrices [40].

Given a single coflow with demand matrix D, we can make it as a doubly stochastic matrix D' through increasing the values of some entries, which is called *stuffing*. A BvN decomposition over D' is exactly a circuit scheduling that meets the coflow traffic demand, *i.e.*, each permutation matrix is a circuit establishment whose duration is the coefficient. In fact,

⁴A doubly stochastic matrix is a square matrix of nonnegative real numbers, where the sum of each rows and columns equal to a constant.

if the reconfiguration delay is zero (*i.e.*, $\delta = 0$), Problem 1 can be solved optimally by stuffing and BvN decomposition [18]. However, with non-negligible reconfiguration delays, it might lead to a terrible CCT to get a circuit scheduling based on the BvN decomposition in previous works (*e.g.*, [7], [14], [18]). This is because when preemption occurs, many tiny residual demands are produced leading to extensive reconfigurations. Specifically, we have the following theorem.

Theorem 1: Given a coflow with demand matrix $D \in \mathbb{R}^{N \times N}$, the method that generates a valid circuit scheduling to satisfy the demand through stuffing and BvN decomposition has an approximation ratio of $\Omega(N)$ to minimize the CCT in OCS.

Proof: According to [41], there exists a doubly stochastic matrix $D = [d_{i,j}] \in \mathbb{R}^{N \times N}$ with the minimum number of decomposition matrices as $\gamma(D) = \lceil \frac{N}{2} \rceil \lceil \frac{N+1}{2} \rceil$. We can construct a new matrix $D' = [d'_{i,j}] = [d_{i,j} \cdot \epsilon]$, where ϵ is a small enough positive constant. The minimum number of decomposition matrices of D' is $\gamma(D)$.

D' needs $\gamma(D)$ decomposition matrices based on BvN, that is to say the total reconfiguration delay is $\gamma(D) \cdot \delta$. Thus, the coflow completion time is $\gamma(D) \cdot \delta + t_{trans} = \Theta(N^2) \cdot \delta + t_{trans}$. Here, t_{trans} is the data transmission time depending on the traffic demand that could be close to zero. However, there exists a scheduling could transmit all demand within N circuits and the CCT of this scheduling would be at most $N \cdot \delta + t'_{trans}$, where t'_{trans} could be close to zero. Therefore, the approximation ratio of the algorithm by directly using primary BvN decomposition is $\Omega(N)$.

B. Regularization on Traffic Demand

The deficiency of primitive BvN-based coflow scheduling is due to the frequent preemption, while non-preemptive scheduling may cause long circuit idle time. To address this issue, we propose a pre-processing operation, named *regularization*, which can greatly reduce the reconfiguration frequency with little cost on the circuit idle time. Given a demand matrix D, the process of regularization is to regularize each entry $d_{i,j} \in D$ to $\left\lceil \frac{d_{i,j}}{\delta} \right\rceil \cdot \delta$, and get a new regularized matrix D'.⁵ Fig. 2 is to illustrate the benefit of regularization. A demand

rig. 2 is to industrate the benefit of regularization. A definited matrix D_{ex} is decomposed into 5 permutation matrices each with a coefficient. Each permutation matrix corresponds to a circuit configuration, and each coefficient represents the transmission time on the corresponding configuration. We set $\delta = 100$. Then the time to complete D_{ex} is (107+104+101+ $2+1)+5\times100 = 815$. If we do *regularization*, the resulting matrix D'_{ex} can be decomposed into 3 permutation matrices each with a coefficient of 200. In fact, the actual transmission time of each circuit may be less than 200, because when one circuit finishes transmitting its demand, the OCS will automatically reconfigure the circuit for further transmission. In view of this, the actual completion time of D'_{ex} is (106 + $109 + 103) + 3 \times 100 = 618$, which is much less than 815, the completion time before regularization.

		D_{ex}														
	104	109	102			0	1	0		1	0	0		0	0	1
	103	105	107	=	107 X	0	0	1	+ 104 X	0	1	0	+ 104 X	1	0	0
	108	101	106			1	0	0		0	0	1		0	1	0
						0	1	0		0	0	1				
regular	izati	on			+ 2 X	1	0	0	+ 1 X	0	1	0				
		D'				0	0	1		1	0	0				
	200	200	200			1	0	0		0	1	0		0	0	1
	200	200	200	=	200 X	0	1	0	+ 200 X	0	0	1	+ 200 X	1	0	0
	200	200	200			0	0	1		1	0	0		0	1	0

Fig. 2. An example of decomposition and regularization.

Algorithm 1 Reco-Sin: Single Collow S	Scheduling	
---------------------------------------	------------	--

1 **Input** coflow demand matrix D, reconfiguration delay δ 2 **Output** a circuit scheduling C

 $3 C \leftarrow \emptyset;$

4 $D' \leftarrow$ Run regularization and stuffing on D with δ ;

5 while D' has non-zero entries **do**

Find a matching on D', let P, α be the permutation matrix and the coefficient derived from this matching;
Append (P, α) to C;
D' ← D' − αP;
Return C;

C. Algorithm

Based on the above techniques, given the coflow demand matrix D, our regularization-based single coflow scheduling, called Reco-Sin, consists of following steps: 1) Regularize D and get a doubly stochastic matrix D' by stuffing; 2) Run BvN decomposition on D'; and 3) Regard each permutation matrix as a circuit establishment and its coefficient as the duration to derive a sequence of circuit assignments. The details of Reco-Sin is described in Algorithm 1.

Initially, Algorithm 1 calls the regularization operation (Line 4). It then iteratively calculates a decomposition (Line 6) and computes the permutation matrix with the maximum coefficient (Line 8). We efficiently compute BvN decomposition by max-min matching which is similar to the method in [14]. Each permutation matrix and its coefficient corresponds to a circuit assignment. We repeat the process until the demand matrix has no non-zero entry and return the circuit scheduling C. As discussed in Fig. 2, the actual completion time of the demand matrix after regularization can be much less than the value of coefficients, since the circuit will stop communication as long as the demands on it are finished.

D. Theoretical Analysis

Here we will prove Reco-Sin is 2-approximate, which is the first constant approximation algorithm for this problem.

Denote t'_{trans} and t'_{conf} as the transmission and the configuration time of the scheduling obtained from Reco-Sin, respectively, while t^*_{trans} and t^*_{conf} are the transmission and the configuration time of the optimal solution respectively.

Lemma 1: In Reco-Sin, the reconfiguration time is no greater than the transmission time, i.e., $t'_{conf} \leq t'_{trans}$.

Proof: Let m be the number of assignments obtained from Reco-Sin. Recall that $t'_{conf} = m \cdot \delta$ and

⁵As we only increase the entry values during regularization, a valid circuit scheduling satisfying D' will definitely satisfy D.

 $t'_{trans} = \sum_{i=1}^{m} dur'(i)$. Each $d_{i,j}$ after regularization is an integral multiple of δ 's. Thus, by BvN decomposition, each circuit assignment lasts at least a time of δ , which gives $dur'(i) \ge \delta$, $\forall i = \{1, 2, \dots, m\}$. This implies $t'_{conf} \le t'_{trans}$.

Set T' as the CCT given by Reco-Sin, while the optimal is T^* . Based on the above lemma, we have Theorem 2.

Theorem 2: Reco-Sin is 2-approximate, i.e., $T' \leq 2 \cdot T^*$. Proof: Suppose D and D' are the original and the regularized demand matrix, respectively. Let ρ and ρ' be the maximum value of the sum of each row and column of D and D' respectively. Let τ be the maximum number of non-zero entries of each row or column of D. Obviously, ρ is the lower bound on the transmission time of Problem 1, that is $\rho \leq t^*_{trans}$. As we require at least τ times circuit establishment for D, we have $\tau \delta \leq t^*_{conf}$. Since, by regularization, each entry of D is increased by at most δ , we get $\rho' \leq \rho + \tau \delta$. The regularized algorithm strictly takes the coefficient of BvN decomposition of D' as the duration of each circuit assignment. And we have $\rho' = t'_{trans}$. By Lemma 1, we can derive that

$$T' = t'_{trans} + t'_{conf} \le 2 \cdot t'_{trans} = 2 \cdot \rho'$$

$$\le 2 \cdot (\rho + \tau \delta) \le 2 \cdot T^*.$$

This completes the proof.

Note that, in the proof of Theorem 2, we do not make use of the assumption that $d_{i,j}^k \ge c \cdot \delta$ set in Sec. III, which means Theorem 2 holds for all coflow demand matrices.

V. MULTIPLE COFLOW SCHEDULING

To schedule multiple coflows in an OCS-based data center, we first propose our algorithm, called Reco-Mul, and the theoritical analysis on the approximation ratio.

A. Our Algorithm: Reco-Mul

To merit the benefits of the existing results for packet switches, we design Reco-Mul that can transform any non-preemptive multi-coflow scheduling in packet switches to a scheduling in an OCS model with few circuit reconfigurations. Here, non-preemptive scheduling in packet switches means that there can be at most one flow transmitting at one time on each port, and once a flow starts transmission, it will be completed without preemption. We denote ALG_p as any algorithm that generates a non-preemptive multi-coflow scheduling in packet switches. According to the scheduling S_p returned by ALG_p , our algorithm Reco-Mul uses a regularization-based policy to reduce the reconfiguration frequency effectively.

As S_p is a non-preemptive scheduling, it can be transformed into a feasible scheduling in OCS by trivially introducing a reconfiguration delay when the circuit in the OCS changes. Directly using S_p to schedule coflows in OCS would incur too many reconfigurations. Intuitively, if we can smartly align the start time of the flow transmissions in S_p , we can somehow start multiple conflict-free flows (that do not share the same ports) at the same time such that only one circuit configuration is needed before transmitting these flows. Fig. 3 illustrates an example of the regularization on three conflict-free flows when



Fig. 3. An example of regularization on the flow start times.

Algorithm 2 Reco-Mul: Multiple Coflows Scheduling

Input coflow demand matrices D, coflow weights W, reconfiguration delay δ, optical transmission threshold c
 Output a multiple coflows scheduling S_o
 S_o ← Ø, Ŝ_o ← Ø;
 S_o ← ΔLC (D, W):

- 10 Define $\eta(\hat{S}_o, \hat{t})$ as the number of reconfiguration required in \hat{S}_o during the time period $[0, \hat{t})$;
- 11 for $(\hat{t}_1, \hat{t}_2, i, j, k) \in \hat{\mathcal{S}}_o$ do 12 $\begin{bmatrix} \mathcal{S}_o \leftarrow \mathcal{S}_o \cup (\hat{t}_1 + \delta\eta(\hat{\mathcal{S}}_o, \hat{t}_1), \hat{t}_2 + \delta\eta(\hat{\mathcal{S}}_o, \hat{t}_2), i, j, k); \\ 13 \text{ Return } \mathcal{S}_o; \end{bmatrix}$

 $\sqrt{c} \cdot \delta = 1$. Before regularization, three reconfigurations are needed at t = 0.5, 0.7 and 0.9, while after the start time is regularized to 1, only one reconfiguration is needed. This reduces the cost of reconfiguration greatly.

Here, we define a pseudo-time axis, denoted as \hat{t} , on which the reconfiguration delay δ shrinks to 0. The start point $\hat{t} = 0$ is corresponding to the real time point $t = 0.^6$ We describe our algorithm, Reco-Mul, in Algorithm 2.

Reco-Mul transforms a multi-coflow scheduling of ALG_p into a feasible scheduling (denoted as S_o) in OCS. During the transformation, Reco-Mul first maps the scheduling S_p (obtained in Line 4) to the pseudo-time axis with regularization, which results in a regularized scheduling \hat{S}_o (Line 5 to Line 9). As \hat{S}_o is defined on the pseudo-time axis, which ignores the reconfiguration delay, we add the reconfiguration time into \hat{S}_o to get the final feasible multi-coflow scheduling in OCS, *i.e.*, S_o (Line 10 to Line 12). During the regularization, Reco-Mul *stretches* the start time of the flow transmissions in S_p to a multiple of $\sqrt{c}\cdot\delta$ (Line 6 and Line 7). Next we show the feasibility of the coflow scheduling returned by Reco-Mul.

Lemma 2: The scheduling returned by Reco-Mul, i.e. S_o , is a feasible coflow scheduling in OCS.

Proof: For any two tuples $e_1 = (t_1, t_2, i_1, j_1, k_1)$ and $e_2 = (t_3, t_4, i_2, j_2, k_2)$ in S_p with port conflicts, these two

⁶For instance, on the real time axis, from t_1 to t_2 the OCS is transmitting flow, halts for δ period of a reconfiguration and then resume transmission at $t_3 = t_2 + \delta$. Then, in our pseudo-time axis, $\hat{t}_1 = t_1$, $\hat{t}_2 = t_2$, while $\hat{t}_3 = \hat{t}_2 = t_2$ but not t_3 .

With the regularization operations (Line 5 to Line 9 in Algorithm 2), we have

$$\hat{t}_1 = \left\lfloor \frac{t_1}{\sqrt{c\delta}} \cdot \frac{\lfloor \sqrt{c} \rfloor + 1}{\lfloor \sqrt{c} \rfloor} \right\rfloor \cdot \sqrt{c\delta}$$

and

$$\hat{t}_3 = \left\lfloor \frac{t_3}{\sqrt{c\delta}} \cdot \frac{\lfloor \sqrt{c} \rfloor + 1}{\lfloor \sqrt{c} \rfloor} \right\rfloor \cdot \sqrt{c\delta}.$$

Thus, the duration between \hat{t}_3 and \hat{t}_1 is given by

$$\hat{t}_3 - \hat{t}_1 \ge \left\lfloor \frac{t_2 \cdot \frac{|\sqrt{c}| + 1}{|\sqrt{c}|}}{\sqrt{c\delta}} \right\rfloor \cdot \sqrt{c\delta} - \left\lfloor \frac{t_1 \cdot \frac{|\sqrt{c}| + 1}{|\sqrt{c}|}}{\sqrt{c\delta}} \right\rfloor \cdot \sqrt{c\delta}$$
$$\ge (t_2 - t_1) + \frac{\sqrt{c}}{c}(t_2 - t_1) - \sqrt{c\delta}.$$

Recall that we assume $d_{i,j}^k \ge c \cdot \delta$. Thus we have

$$t_2 - t_1 \ge c \cdot \delta \quad \Rightarrow \quad \hat{t}_3 - \hat{t}_1 \ge t_2 - t_1,$$

Therefore, the new tuples $\hat{e}_1 = (\hat{t}_1, \hat{t}_2, i_1, j_1, k_1)$ and $\hat{e}_2 =$ $(\hat{t}_3, \hat{t}_4, i_2, j_2, k_2)$ do not overlap with each other in the time range, and \hat{S}_o still follows the port constraint. On the other hand, the transmission time in \hat{S}_o for each coflow remains the same as that in S_p , and thus S_o also satisfies all demand matrices. Because reconfiguration time does not violates the port constraint and the demand requirement, the final scheduling S_o is still feasible, which completes the proof.

B. Theoretical Analysis of Reco-Mul

Denote OPT as the total weighted CCT of the optimal non-preemptive scheduling in OCS, *i.e.*, $OPT = \sum_{k=1}^{K} w_k T_k^*$ where T_k^* is the CCT of coflow k in OCS. We further denote $OPT^p = \sum_{k=1}^{K} w_k T_k^{*,p}$ as the total weighted CCT of the optimal scheduling in packet switches, where $T_k^{*,p}$ is the CCT of the coflow k in packet switches. As the optimal non-preemptive scheduling in packet switches, OPT, has no reconfiguration time, we have $OPT^p \leq OPT$.

Denote $T_k^o = t_{trans}^k + t_{conf}^k$ as the CCT of coflow k in the scheduling S_o , where t_{trans}^k and t_{conf}^k are the transmission time (including the time of waiting other coflows' transmissions) and the reconfiguration time of coflow k. Denote T_k^p as the CCT of coflow k in the non-preemptive scheduling in packet switches, *i.e.*, S_p . With any non-preemptive multi-coflow scheduling in packet switches, in the following theorem we show Reco-Mul can transform it to a feasible scheduling in OCS by at most losing a constant factor.

Theorem 3: Reco-Mul returns a feasible multi-coflow scheduling in OCS with an approximation ratio of Δ . $\left(1+\frac{1}{\lfloor\sqrt{c}\rfloor}\right)^2$, where Δ is the approximation ratio of ALG_p . *Proof:* The feasibility of the S_o is proved in Lemma 2. Now, we prove its approximation ratio.

The start time of each flow in S_{o} has been regularized to a multiple of $\sqrt{c\delta}$. Since reconfiguration only happens at the

Algorithm 3 Reco-Mul+: Multiple Coflows Scheduling
1 Input coflow demand matrices \mathcal{D} , coflow weights \mathcal{W} ,
reconfiguration delay δ
2 Output a multiple coflows scheduling S_o
$\mathfrak{Z} \mathcal{D}' \leftarrow \{Regularization(D) \forall D \in \mathcal{D}\};\$
$4\;\mathcal{S}_{o} \leftarrow \varnothing, \hat{\mathcal{S}}_{o} \leftarrow \varnothing;$
5 $\hat{\mathcal{S}}_o \leftarrow ALG_p(\mathcal{D}', \mathcal{W});$
6 for $(\hat{t}_1,\hat{t}_2,i,j,k)\in\hat{\mathcal{S}}_o$ do
7 $\bigcup \mathcal{S}_o \leftarrow \mathcal{S}_o \cup (\hat{t}_1 + \delta \eta(\hat{\mathcal{S}}_o, \hat{t}_1), \hat{t}_2 + \delta \eta(\hat{\mathcal{S}}_o, \hat{t}_2), i, j, k);$
8 Return S_{α} ;

start of transmission, we have $t_{conf}^k \leq \frac{1}{\sqrt{c}} t_{trans}^k$, $\forall k \in \mathcal{K}$. By adding t_{trans}^k at the both sides, we get

$$T_k^o = t_{conf}^k + t_{trans}^k \le \left(1 + \frac{1}{\sqrt{c}}\right) t_{trans}^k.$$
 (1)

Recall the assumption of $d_{i,j}^k \ge c\delta$. After the regularization operations (Line 5 to Line 9 in Algorithm 2), the following inequality holds:

$$\mu_{trans}^{k} \le \left(1 + \frac{1}{\lfloor \sqrt{c} \rfloor}\right) T_{k}^{p}.$$
(2)

This is because \hat{t}_1 is always not greater than $(1 + \frac{1}{\lfloor \sqrt{c} \rfloor}) \cdot t_1$ (refer to Line 6 and Line 7 in Algorithm 2). Thus, by combining the Eqn. (1) and (2), we have $T_k^o \leq \left(1 + \frac{1}{\lfloor \sqrt{c} \rfloor}\right)^2 T_k^p$. With weighted summing up all coflows, we have

$$\sum_{k \in \mathcal{K}} w_k T_k^o \le \left(1 + \frac{1}{\lfloor \sqrt{c} \rfloor}\right)^2 \sum_{k \in \mathcal{K}} w_k T_k^p$$
$$\le \Delta \cdot \left(1 + \frac{1}{\lfloor \sqrt{c} \rfloor}\right)^2 \sum_{k \in \mathcal{K}} w_k T_k^*$$

 \square

This completes the proof. To the best of our knowledge, the multi-coflow scheduling algorithm proposed by Shafiee and Ghaderi in [35] has the best approximation ratio in packet switches (4-approximate, *i.e.*, $\Delta = 4$). So, we can give the approximation ratio of Reco-Mul in the following corollary.

Corollary 1: Reco-Mul is $4 \cdot \left(1 + \frac{1}{\lfloor \sqrt{c} \rfloor}\right)^2$ -approximate with the algorithm given by Shafiee and Ghaderi [35].

C. Algorithm: Reco-Mul+

Reco-Mul can achieve a constant approximation ratio, based on the assumption that a nonzero traffic demand in OCS is no less than $c \times \delta$. Here, c is a constant. To get rid of such assumption, we imitate the method used in Reco-Sin and propose another algorithm Reco-Mul+ for multiple coflow scheduling. Our main idea is to regularize all the coflow demands before they are scheduled, which could also reduce the overhead in Reco-Mul to calculate the detailed scheduling strategy before flow transmission. Reco-Mul+ is described in Algorithm 3.

The process of Reco-Mul+ uses non-preemptive multi-coflow scheduling algorithm in packet switches as a subroutine. First, Reco-Mul+ calls the regularization

operation so that all demands of matrices $D \in D$ are increased to integer multiple of δ (Line 3). Then we run the algorithm designed for the packet switch on D' and get the regularized scheduling \hat{S}_o (Line 5). Finally, we add the reconfiguration time into \hat{S}_o to get the final feasible multi-coflow scheduling in OCS (Line 7) just like the transformation process in Algorithm 2.

D. Theoretical Analysis of Reco-Mul+

In the analysis part we use the algorithm proposed in [35] as the called packet switch subroutine. We define $D_{1..k} = \sum_{l=1}^{k} D_l$ as the aggregate coflow of first k coflows. Define $\rho_{1..k} = \max\{\max_i\{\sum_j d_{i,j}^{1..k}\}, \max_j\{\sum_i d_{i,j}^{1..k}\}\}$, which is the effective size of the aggregate coflow defined in [35]. Set $\rho_{max} = \max_{k \in \mathcal{K}} \rho_k$, $\rho_{min} = \min_{k \in \mathcal{K}} \rho_k$, $\tau_{max} = \max_{k \in \mathcal{K}} \tau_k$, and $\tau_{min} = \min_{k \in \mathcal{K}} \tau_k$. Then we have:

Theorem 4: Reco-Mul+ can achieve $O(K \cdot \max\{\frac{\rho_{max}}{\rho_{min}}, \frac{\tau_{max}}{\tau_{min}}\})$ -approximate for minimizing the weighted coflow completion time, i.e., $\sum_{k \in \mathcal{K}} w_k T_k \leq O(K \cdot \max\{\frac{\rho_{max}}{\rho_{min}}, \frac{\tau_{max}}{\tau_{min}}\}) \sum_{k \in \mathcal{K}} w_k T_k^*$. Proof: First, by solving LP in [35] we can get the

Proof: First, by solving LP in [35] we can get the order between coflows. The coflows are ordered based on \tilde{f}'_k , the optimal completion time of LP, so that $\tilde{f}'_1 \leq \tilde{f}'_2 \leq \cdots \leq \tilde{f}'_K$. According to Theorem 1 in [35], we have $t'_{k,trans} \leq 4\rho'_{1..k}$. Similar with the analysis of Theorem 2, we have

$$T_{k} \leq T'_{k} \leq 2t'_{k,trans} \leq 8\rho'_{1..k}$$

$$\leq 8(\sum_{l=1}^{k} \rho_{l} + \sum_{l=1}^{k} \tau_{l}\delta)$$

$$\leq 8(K\frac{\rho_{max}}{\rho_{min}}t^{*}_{k,trans} + K\frac{\tau_{max}}{\tau_{min}}t^{*}_{k,conf})$$

$$\leq O(K \cdot \max\{\frac{\rho_{max}}{\rho_{min}}, \frac{\tau_{max}}{\tau_{min}}\})T^{*}_{k}.$$

Thus,

$$\sum_{k \in \mathcal{K}} w_k T_k \le O(K \cdot \max\{\frac{\rho_{max}}{\rho_{min}}, \frac{\tau_{max}}{\tau_{min}}\}) \sum_{k \in \mathcal{K}} w_k T_k^*.$$

Corollary 2: Reco-Mul+ is O(K)-approximation for minimizing the unweighted coflow completion time, i.e., $\sum_{k \in \mathcal{K}} T_k \leq O(K) \sum_{k \in \mathcal{K}} T_k^*$.

Proof:

$$\sum_{k \in \mathcal{K}} T_k \leq \sum_{k=1}^K 8\left(\sum_{l=1}^k \rho_l + \sum_{l=1}^k \tau_l \delta\right)$$
$$\leq 8\sum_{k=1}^K (K - k + 1)(\rho_k + \tau_k \delta)$$
$$\leq 8K \sum_{k=1}^K (\rho_k + \tau_k \delta)$$
$$\leq 8K \sum_{k=1}^K T_k^* = O(K) \sum_{k \in \mathcal{K}} T_k^*.$$

TABLE III CATEGORY OF COFLOWS WITH DIFFERENT TRANSMISSION MODE

Modes	S2S	S2M	M2S	M2M
Numbers%	23.38	9.89	40.11	26.62
Sizes%	0.005	0.024	0.028	99.943

VI. PERFORMANCE EVALUATION

In this section, we conduct extensive simulations to evaluate Reco-Sin, Reco-Mul and Reco-Mul+ based on real workloads. Our results and analysis indicates that our algorithms outperform the state-of-the-art baselines in almost all the cases dramatically.

A. Methodology

Workload: The workload that consists of 526 coflows is based on a Hive/MapReduce trace collected on a 3000-machine 150-rack cluster with 10:1 oversubscription ratio. The trace contains the coflow information which are its arrival time and the size of shuffle data at the reducers. Since there is no information about the size of the data flow from the mappers to the reducers, we uniformly allocate the shuffle data of reducers to the mappers, similar to [11]. Similar to Sunflow [15], we extract each record as a coflow demand matrix, whose rows and columns represent the mappers and reducers, respectively. In addition, we add $\pm 5\%$ perturbation of flow size to simulate the real production environment. Coflows are categorized into three types based on the density of their demand matrices as follows. We call the *density* of coflow demand matrix as DS, which represents the proportion of non-zero elements in the matrix and has a value between 0 and 1. We then define 3 types of coflows as follows: 1) Sparse: $DS \leq 0.05$, 2) Normal: $0.05 \leq DS \leq 0.5$, and 3) Dense: $DS \ge 0.5$. The proportions of Sparse, Normal and Dense coflow are 86.31%, 5.13% and 8.56%, respectively.

Coflow Transmission Mode: We categorize the coflows according to the following transmission modes: 1) a *Single-to-Single (S2S)* coflow contains only one flow from one single ingress port to one egress port; 2) a *Single-to-Multiple (S2M)* coflow contains flows from a single ingress port to multiple egress ports; 3) a *Multiple-to-Single (M2S)* coflow consists of flows from multiple ingress ports to one single egress port, and 4) a *Multiple-to-Multiple (M2M)* coflow consists of flows from multiple ingress ports to multiple egress ports. We hence categorize the coflows into four types summarized in Table III.

S2S, S2M and M2S Transmission Modes: Each flow in these types shares one (ingress or egress) port with all other flows. The optimal approach is to schedule them one by one. Hence, both Solstice and Reco-Sin can achieve the optimal CCT.

M2M Transmission Modes: Table III shows that the majority of the coflows in the workload are the M2M transmission type. Their performance is critical to the overall performance. We will see later in this section the difference between Solstice and Reco-Sin in their performances to minimize the CCT.

Simulator: We develop a trace-driven flow-level simulator to perform various algorithms based on the embedded LP solver GUROBI [42]. Our experiments are conducted on a server with a Intel Xeon E5-2620v4 CPU with 32 Gigabyte memory and 2TB hard drive.

 \square

Metrics: Our metrics is the normalized average (95-percentile) weighted or unweighted CCT of a scheme compared with our algorithms Reco-Sin, Reco-Mul or Reco-Mul+. For example, the Normalized CCT of Algorithm A is defined as

Normalized CCT =
$$\frac{\text{the CCT of Algorithm A}}{\text{the CCT of Reco-Mul}}$$
.

The lower the normalized CCT, the better the performance of Algorithm A is. Moreover, the CCT not only consists of the reconfiguration and the flow transmission delay, but also the algorithmic computational delay. Our algorithms mainly contain two steps: preprocessing (*e.g.*, regularization) and scheduling. Based on our simulation, the preprocessing delay is about 1/200 of the scheduling delay. While the scheduling is negligible compared with the flow transmission delay. Therefore, we ignore the computational delay similar to [16].

B. Baselines

We compare the performances of our proposed algorithms Reco-Sin, Reco-Mul and Reco-Mul+ with the following baselines for single and multiple coflow scheduling in minimizing the (weighted) CCT.

1) Lower Bound: the theoretical lower bounm of the column values, and τ is the maximum number of nonzero entriesd on the CCT of a single coflow in the OCS, $T_{lb} = \rho + \tau \delta$, where ρ is the maximum value of the sum of the matrix row values and τ is the maximum number of nonzero entries in rows or columns in matrix.

2) Solstice [14]: the state-of-the-art circuit scheduling algorithm which operates in two steps: first, it transforms the demand matrix D_C into a k-biostochastic matrix (*stuffing*) and second, it iteratively computes a scheduling of configurations (*slicing* operation) to complete the transmission.

3) SEBF [8] + Solstice [14]: Smallest-Effective-Bottleneck-First (SEBF) [8] is to prioritize the coflow with the smallest effective bottleneck flow. However, SEBF can not be applied to the OCS directly as the bandwidth allocation operation can not be used in the context of OCS. Thus, we modify it a bit and combine it with Solstice for multiple coflow scheduling. Specifically, the algorithm of [8] is SEBF(Smallest-Effective-Bottleneck-First)+MADD(Minimum-Allocation-for-Desired-Duration),

where SEBF is used to schedule inter-coflow and MADD is to schedule intra-coflow. However, MADD is based on bandwidth sharing which can not be directly used in OCS. Therefore, we choose Solstice as the intra-coflow scheduling strategy and take SEBF+Solstice as the baseline in OCS.

4) LP-II-GB [18]: LP-II-GB is designed for multiple coflow scheduling problem. It is mainly based on a time interval indexed linear program which can be solved in polynomial time. The algorithm solves a relaxed linear program capturing the primitive problem, and the optimal solution is used to derive an estimation value of the CCT for each coflow. Based on these estimation value computed, it determines the scheduling order of the coflows. For single coflow scheduling, they adopt the BvN method.



Fig. 4. Performance in single coflow scheduling.

5) LP-OV-LS [35]: Besides LP-II-GB, we take another LP-based multiple coflow scheduling algorithm LP-OV-LS for the evaluation of the impact of regularization. Different from LP-II-GB, LP-OV-LS first obtains a transmission order among coflows by solving LP. LP-OV-LS is also the packet switch algorithm currently implemented in Reco-Mul and Reco-Mul+ during the following simulations.

6) Sincronia [34]: Same as LP-OV-LS, Sincronia also generates the final schedule by obtaining the order between coflows. The difference is that Sincronia only uses a simple greedy method to get the order between coflows, which is much more efficient than solving LP.

C. Evaluation of Reco-Sin

We set the parameters of the optical switches according to the practical scenarios [12], [16]. The link bandwidth is set as 100Gbps. The value of the reconfiguration delay, δ , ranges from 1 μ s to 10ms, and its default value is set as 100 μ s.

1) Reconfiguration Frequency: The density of demand matrix can greatly impact the reconfiguration frequency of coflow. Thus, we fix δ as the default value, and compare Reco-Sin's reconfiguration frequency with Solstice's. (In Fig. 4(a), Fig. 4(b), from top to bottom, coflow's density is sparse, normal and dense.) In general, we can observe from Fig. 4(a) that Reco-Sin has a lower reconfiguration frequency than Solstice does. When the demand matrix of coflow is sparse, normal and dense, Solstice spends $2.76 \times$, $7.35 \times$ and $7.89 \times$ more reconfigurations than Reco-Sin, respectively. Note that the performance gap becomes larger with the density increasing. The reason is that the number of permutation matrix after BvN decomposition grows along with the increasing of coflow's density. We can conclude that Reco-Sin outperforms Solstice in optimizing reconfiguration times for all types of coflows.

2) CCT: In this section, we evaluate the performance of Reco-Sin and Solstice in minimizing the CCT. We observe from Fig. 4(b) that Solstice needs more time to finish the same coflow than Reco-Sin does. Specifically, Solstice needs $1.23 \times$, $1.71 \times$ and $1.97 \times$ more time than Reco-Sin to decompose demand matrix when scheduling *sparse*, *normal* and *dense* coflows respectively. This is because of the advantage of less reconfiguration frequency Reco-Sin has and that the proportion of reconfiguration time in CCT declines with the increase of transmission time (*i.e.*, the increase of demand matrix density). In short, Reco-Sin outperforms Solstice in minimizing CCT when scheduling one single coflow.



Fig. 5. Performance in single coflow scheduling.

3) Impact of δ : The variation of δ is an important characteristic decided by the hardware of OCS, and it impact the CCT directly. Similar to previous simulations, from the perspective of reconfiguration times and CCT, we analyze the strength and weakness of Reco-Sin and Solstice.

The curves in Figure. 5(a) present that Solstice spends more time on reconfiguration to complete the same coflow than Reco-Sin. In fact, the variation of δ would not have effect on reconfiguration frequency of Solstice. The Figure. 5(a) depicts when coflow density changes, the number of reconfigurations of Solstice basically remain invariable when δ changes. Meanwhile, the amount of reconfigurations of Reco-Sin declines with the increase of δ . The main reason is that our regularization operation is directly correlated with δ . When δ increases, the elements in the coflow demand matrix will be more aligned after regularization, which in turn reduces the reconfiguration time. When decomposing sparse coflow, Solstice spends $2.17 \sim 2.85 \times$ reconfigurations as many as Reco-Sin does. When the matrix is not sparse, the reconfiguration demands of Solstice can be $3.42 \sim 12.22 \times$ as many as that of Reco-Sin.

Figure. 5(b) illustrates the performance on CCT. Here, we adopt the theoretical lower bound of CCT as the normalized benchmark and indicate performance of Reco-Sin and Solstice. When the coflow demand matrix grows from sparse to dense, the advantage of Reco-Sin is declining. The reason is that the proportion of reconfiguration time in CCT decreases when the coflow matrix is more sparse. Meanwhile, regardless of the demand matrix density, the gap between Reco-Sin and Solstice expands when δ increases. It is due to that the reconfiguration time becomes dominant in CCT with the increasing of δ , and the advantage of Reco-Sin becomes more prominent as it has less frequent reconfigurations. As for the different coflows shown in Fig. 5(b), Solstice can consume $6.55 \times$, $24.64 \times$ and $24.55 \times$ time as much as the lower bound, while the CCT of Reco-Sin consumes only $2.00\times$, $1.99\times$ and $1.94\times$, respectively.

D. Evaluation of Reco-Mul and Reco-Mul+

In this section, we evaluate Reco-Mul and Reco-Mul+ in reducing the CCT and reconfiguration frequency.



Fig. 6. Performance in minimizing unweighted CCT for multiple coflows.

1) Unweighted CCT: First, we set the coflow weights equally and compare the performance of Reco-Mul, Reco-Mul+, LP-II-GB and SEBF+Solstice in minimizing unweighted CCT. The results are shown in Fig. 6. SEBF+Solstice has the worst performance; LP-II-GB comes the second and Reco-Mul+ performs the best. When coflows are sparse, LP-II-GB, SEBF+Solstice and Sincronia are $8.39 \times (5.96 \times)$, $12.47 \times (10.43 \times)$ and $2.75 \times (2.43 \times)$ worse than Reco-Mul, respectively. For the cases of dense coflows, the improvements of Reco-Mul over LP-II-GB, SEBF+Solstice and Sincronia are $1.93 \times (1.66 \times), 3.59 \times$ $(3.75\times)$ and $1.34\times$ $(1.19\times)$ respectively. When scheduling all (i.e., mixed) kinds of coflows, LP-II-GB, SEBF+Solstice and Sincronia need $3.50 \times (4.20 \times), 7.39 \times (6.11 \times)$ and $1.46 \times$ $(1.47\times)$ more time than Reco-Mul, respectively. On this basis, Reco-Mul+ has improved 15.48% (19.33%), 36.30% (28.53%), 50.32% (39.01%) and 60.34% (41.17%) compared to Reco-Mul for the case of sparse, normal, dense and mixed respectively. The reason of the above performance is similar to the one when minimizing the weighted CCT. Due to the limited space, we omit the details. In short, Reco-Mul and Reco-Mul+ outperform LP-II-GB, SEBF+Solstice and Sincronia in minimizing weighted and unweighted CCT.

2) Weighted CCT: SEBF+Solstice can be used to optimize the unweighted CCT, while Reco-Mul and LP-II-GB are for weighted (and, of course, unweighted) CCT. Thus, in this part, we compare the performance of Reco-Mul with LP-II-GB's in minimizing weighted CCT. The coflow weights are set uniformly from [0,1]. Fig. 7 shows that, Reco-Mul and Reco-Mul+ outperform LP-II-GB regardless of the coflow density level. Specifically, for the cases of sparse, normal and dense coflows, Reco-Mul has 67.72%(49.46%), 59.81% (55.14%) and 46.79% (39.34%) performance improvement than LP-II-GB in minimizing the average (95-percentile) weighted CCT, while the corresponding improvement of Reco-Mul+ are 68.40% (51.39%), 69.75%(65.42%) and 56.17% (48.16%) respectively. When multiple coflows contain all the coflows (i.e., coflows with all kinds of density level), Reco-Mul is $2.33 \times (1.86 \times)$ better than LP-II-GB, and Reco-Mul+ is $2.76 \times (2.20 \times)$ better than LP-II-GB. The reason of Reco-Mul's advantage is basically the alignment of the flow start time. While coflow's density increases, the effect of the start time alignment on the performance reduces, which accounts for the performance differences among the cases with different coflow sparsities. In addition, we found although the performance of Sincronia



Fig. 7. Performance in minimizing weighted CCT for multiple coflows.



Fig. 8. Comparison of reconfiguration frequency for multiple coflows.

is better than LP-II-GB, the normalizaed CCT of Sincronia is still $1.56 \times (1.85 \times)$ longer than Reco-Mul (Reco-Mul+).

3) Reconfiguration Frequency: In Sec.VI-C, we present that our proposed Reco-Sin needs less circuit reconfigurations than Solstice does. In this part, we compare the performance of LP-II-GB, SEBF+Solstice, Sincronia, Reco-Mul and Reco-Mul+, in minimizing the reconfiguration frequency. Fig. 8 presents our experiment results. Specifically, if we consider the case where there are coflows of all different density levels, the reconfiguration time of LP-II-GB (SEBF+Solstice, Sincronia) is $3.23 \times (3.63 \times , 1.66 \times)$ larger than Reco-Mul's. When scheduling the sparse, normal and dense coflows, the performance gaps between Reco-Mul and LP-II-GB (SEBF+Solstice, Sincronia) are $3.49 \times (5.26 \times, 2.06 \times), 3.42 \times$ $(4.61\times, 1.84\times)$ and $1.64\times$ $(3.63\times, 1.40\times)$ respectively. Furthermore, Reco-Mul+ only needs $60.19\% \sim 75.2\%$ reconfigurations compared to Reco-Mul. The reason could be the stretching operation makes the schedule of Reco-Mul more sparse than Reco-Mul+, which further causes the difference of CCT between Reco-Mul and Reco-Mul+. When the coflow density increases, the performance gap between Reco-Mul and baseline algorithm decreases. Because higher coflow density reduces the proportion of fragmentary flow demand, thereby diminishes the advantage of Reco-Mul.

4) Impact of δ : Similar to the simulation of Reco-Sin, δ is also a crucial variable which can affect the performance of Reco-Mul and Reco-Mul+. In Reco-Mul, a critical step is stretching the scheduling order of flows to eliminate their conflict on transmission time. Moreover, δ is directly relevant of this stretch process. According to Fig. 9(a), when δ increases from 1µs to 100µs, Reco-Mul gradually expands its advantages. For instance, when $\delta = 1µs$, LP-II-GB (Sincronia) needs $1.16 \times (0.96 \times)$ to finish multiple coflow scheduling than Reco-Mul. When $\delta = 10µs$ and $\delta = 100µs$, Reco-Mul need 55.68% (124.87%) and 33.09% (75.76%) time of LP-II-GB (Sincronia). Because the reconfiguration



Fig. 9. The impact of δ and c in multiple coflow scheduling.

frequency of Reco-Mul will decline while δ enlarges. In the meantime, due to reconfiguration time is a product of reconfiguration frequency and δ , generally, the reconfiguration time of Reco-Mul declines while δ increases to $100\mu s$. On the other hand, reconfiguration frequency of LP-II-GB and Sincronia will not be affected by the variation of δ , so CCT of LP-II-GB and Sincronia become larger with δ 's enlargement. However, we observe that the growth of gap between Reco-Mul and LP-II-GB and Sincronia slowdown when δ increases to 1ms and 10ms. The reason is the excessive growth of δ . When δ increases to ms, the reconfiguration time becomes dominant in CCT. If δ approaches infinity, the performance gap between Reco-Mul (Reco-Mul+) and LP-II-GB (Sincronia) will approach to the ratio of their reconfiguration frequency.

In conclusion, Reco-Mul and Reco-Mul+ are able to maintain the advantages when varying the δ 's. As a matter of fact, our conclusion reflects that Reco-Mul and Reco-Mul+ are adaptable to different kinds of industrial OCS with different reconfiguration delays.

5) Impact of the Constant c: Recall that we assume that there is no tiny flow in OCS, and any traffic demand is at least $c \cdot \delta$ (Sec. III). In practice, the value of c corresponds to the real industrial environment. As shown in Fig. 9(b), the advantage of Reco-Mul gradually is strengthened when c increases. For c=2 to 4, the time required by LP-II-GB for scheduling increases from $1.74 \times$ to $1.96 \times$. When c ranges from 4 to 6, Reco-Mul strengthens its advantages in minimizing CCT from $1.96 \times$ to $3.43 \times$. Thus, the simulation result validates our theoretical analysis. The reconfiguration frequency and time of Reco-Mul will decrease along with the increase of c by the stretching operation. Consequently, the CCT of Reco-Mul will continuously decrease. We also found Sincronia is not sensitive to changes of c, e.g., the normalized CCT of Sincronia increases from 1.32 to 1.67 for c = 2to 6. Besides, we further set c = 20 to evaluate the case where only relatively very large flows are considered. The simulation results illustrate that our methods outperform the baseline consistently.

E. Evaluation of the Impact of Regularization

In this subsection, we conduct a series of experiments to illustrate how much regularization (on the demand matrices of the coflows) as a preprocessing can improve the baselines. We compare the performance of a baseline (*e.g.*, LP-II-GB) and its variation with regularization (*e.g.*, denoted as Reco (LP-II-GB)) while handling different types of coflows.



Fig. 10. Unweighted CCT with and without regularization.



Fig. 11. Weighted CCT with and without regularization.

1) Unweighted CCT: First, we study the impact of regularization on the unweighted CCT as shown in Fig. 10. Overall, regularization improves the performance of LP-II-GB, LP-OV-LS, SEBF+Solstice, and Sincronia by 10.20%, 81.48%, 9.12%, and 81.31%, respectively. The improvement of LP-OV-LS and Sincronia are the most significant, which implies regularization is more beneficial to the fine-grained scheduling algorithms. It should be noted that when the demand matrices are sparse, the CCT of SEBF+Solstice even increased after regularization. This phenomenon is because when the demand matrices are sparse, regularization process cannot reduce the number of switch reconfigurations.

2) Weighted CCT: We also evaluate the impact of regularization on the weighted CCT (Fig. 11). The overall performance improvements on weighted CCT brought by regularization to LP-II-GB, LP-OV-LS and Sincronia are respectively 7.05%, 77.26% and 75.61%. This experimental result is consistent with the result in the experiment of unweighted CCT, which demonstrates the regularization preprocessing can be used to handle coflows with weights effectively.

3) Reconfiguration Frequency: In order to explore the impact of regularization on CCT, we further investigate the variance of reconfiguration frequency under the impact of regularization (shown in Fig. 12). When all kinds of coflows are mixed, the reduced reconfigurations of LP-II-GB, LP-OV-LS, SEBF+Solstice and Sincronia are 22.94%, 81.48%, 12.07%, and 78.76%, respectively. When the demand matrices are sparse, the decreasing of reconfigurations in SEBF+Solstice is only 0.54%, which cannot cover the negative impact of regularization and leads to the increase of CCT.

4) Impact of δ : We also study the impact of δ on the benefit of regularization (Fig. 13). For the mixed types of coflows, with the growth of δ , the gap between



Fig. 12. Reconfiguration frequency with and wihtout regularization.



Fig. 13. Impact of reconfiguration with the growth of δ .

Reco-Mul+(*i.e.*, Reco(LP-OV-LS)) and other baselines is gradually increases, but the magnitude of the increasing gradually decreases. Similar to the discussion in Sec. VI-D, the gap between Reco and baseline algorithms will approach to the ratio of their reconfiguration frequency when δ approaches infinity. Besides, it shows that Reco(LP-OV-LS) and Reco(Sincronia) have similar performance when δ changes, which indicates the ordering generated by LP-OV-LS and Sincronia are almost the same.

VII. DISCUSSION

Mice Flows

OCS is designed for the high-throughput transmission due to its large bandwidth, and the reconfiguration process is a natural handicap to transmit mice flows. Let τ be the maximum number of non-zero entries of each row or column of a demand matrix, then $\tau\delta$ is a lower bound of the total reconfiguration delay, which even the optimal scheduling in OCS can not avoid. It implies that OCS is not ideal to transmit mice flows. Thus, in practical systems, only elephant flows are transferred through the OCS and mice flows can be handled more efficiently by packet switches [14]. Therefore, in the analysis of Reco-Mul, we assume no tiny flows in OCS. Note that here the demarcation of the mice and elephant flows is the traffic demand of $c\delta$ in OCS, *e.g.*, the flow with size larger than 2.5MB will be treated as an elephant flow when the port bandwidth=1Tbps, c = 1 and $\delta = 20\mu s$.

Not-All-Stop OCS

As stated in Sec. III-A, we consider *all-stop* OCS commonly used in previous works [7], [13], [14], [16]. Another

IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 29, NO. 3, JUNE 2021

TABLE IV APPROXIMATION RATIOS FOR COFLOW SCHEDULING IN OCS

Model	Sin	gle coflow	Multiple coflow				
Algorithm	Ν	А	N	А			
Sunflow [15]	2	-	-	-			
Reco-Sin,Reco-Mul	-	2	$4 \cdot f(c)$	$4 \cdot f(c)$			
N: Not-all-stop model, A: All-stop model, $f(c) = \left(1 + \frac{1}{ \sqrt{c} }\right)^2$							

circuit establishment model is *not-all-stop* as adopted by Sunflow [15], where during reconfiguration, communication stops only on the affected ports including the ports to be set up and those to be torn down, while the circuits unchanged can keep transmission. Based on the current manufacturing technique, a pure non-all-stop OCS is still hard to implement. Intuitively, a feasible scheduling in the *all-stop* model is still feasible in the *not-all-stop* model. Moreover, our algorithm Reco-Mul can have the same approximation ratio in the *notall-stop* model. Note that Eqn. 3 in the approximation ratio proof of Reco-Mul (Theorem 3) is still correct by replacing the optional weighted CCT in the *all-stop* model with that in the *not-all-stop*. Therefore, as shown in Table IV, combining the results of Sunflow, the coflow scheduling in both OCS models can achieve constant approximations.

Online Implementation

As stated in Sec. V, any non-preemptive algorithm designed for packet switches can be transformed to a scheduling scheme in OCS based on our preprocessing. Although the algorithm used in Reco is offline (*e.g.*, LP-OV-LS and Sincronia), online coflow scheduling (*e.g.*, [11]) can still be used as the basis of Reco. The only additional step required is regularization, which can be done efficiently. It is left as our future work to investigate the performance of online scheduling schemes in OCS.

VIII. CONCLUSION

In this article, we study the coflow scheduling problem in the context of optical circuit switches enabled data centers. We first propose a novel operation called *regularization* which is simple but effective to avoid frequent circuit reconfigurations. Then, for single coflow scheduling, we present Reco-Sin, an efficient 2-approximation algorithm. As for multiple coflow scheduling, we first present Reco-Mul, based on the regularization operation, which achieves a constant approximation ratio to the optimum for elephant flows. To drop the assumption on the flow size in Reco-Mul and overcome its calculation overhead, we further propose an algorithm, named Reco-Mul+, for multiple coflow scheduling, which has an approximation ratio of O(K). Extensive simulations based on real data traces show that our proposed algorithms significantly outperform the state-of-the-art schemes in optimizing the weighted CCT. In practical systems, the information of a coflow might be known only when it arrives to the network. To this end, one interesting future direction is to derive online coflow scheduling schemes for OCS-based networks.

REFERENCES

- C. Zhang, H. Tan, C. Xu, X.-Y. Li, S. Tang, and Y. Li, "Reco: Efficient regularization-based coflow scheduling in optical circuit switches," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 111–121.
- [2] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.
- [3] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: Distributed data-parallel programs from sequential building blocks," in *Proc. ACM SIGOPS/Eurosys*, 2007, pp. 59–72.
- [4] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proc. HotCloud*, 2010, p. 10.
- [5] M. Chowdhury and I. Stoica, "Coflow: A networking abstraction for cluster applications," in *Proc. 11th ACM Workshop Hot Topics Netw. HotNets-XI*, 2012, pp. 31–36.
- [6] C. Minkenberg *et al.*, "Performance benefits of optical circuit switches for large-scale dragonfly networks," in *Proc. Opt. Fiber Commun. Conf.*, Mar. 2016, pp. 1–3.
- [7] G. Porter *et al.*, "Integrating microsecond circuit switching into the data center," in *Proc. ACM SIGCOMM Conf. SIGCOMM*, Aug. 2013, pp. 447–458.
- [8] M. Chowdhury, Y. Zhong, and I. Stoica, "Efficient coflow scheduling with varys," in *Proc. ACM Conf. SIGCOMM*, Aug. 2014, pp. 443–454.
- [9] M. Chowdhury and I. Stoica, "Efficient coflow scheduling without prior knowledge," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015, pp. 393–406.
- [10] H. Zhang, L. Chen, B. Yi, K. Chen, M. Chowdhury, and Y. Geng, "CODA: Toward automatically identifying and scheduling coflows in the dark," in *Proc. ACM SIGCOMM Conf.*, Aug. 2016, pp. 160–173.
- [11] Y. Li et al., "Efficient online coflow routing and scheduling," in Proc. 17th ACM Int. Symp. Mobile Ad Hoc Netw. Comput., Jul. 2016, pp. 161–170.
- [12] N. Farrington *et al.*, "Helios: A hybrid electrical/optical switch architecture for modular data centers," in *Proc. ACM SIGCOMM*, 2010, pp. 339–350.
- [13] G. Wang et al., "C-through: Part-time optics in data centers," in Proc. ACM SIGCOMM Conf. SIGCOMM - SIGCOMM, 2010, pp. 327–338.
- [14] H. Liu et al., "Scheduling techniques for hybrid circuit/packet networks," in Proc. 11th ACM Conf. Emerg. Netw. Exp. Technol., Dec. 2015, pp. 1–13.
- [15] X. S. Huang, X. S. Sun, and T. S. E. Ng, "Sunflow: Efficient optical circuit scheduling for coflows," in *Proc. 12th Int. Conf. Emerg. Netw. Exp. Technol.*, Dec. 2016, pp. 297–311.
- [16] C.-H. Wang, T. Javidi, and G. Porter, "End-to-end scheduling for all-optical data centers," in *Proc. IEEE Conf. Comput. Commun.* (*INFOCOM*), Apr. 2015, pp. 406–414.
- [17] H. Liu *et al.*, "Circuit switching under the radar with reactor," in *Proc. USENIX NSDI*, 2014, pp. 1–15.
- [18] Z. Qiu, C. Stein, and Y. Zhong, "Minimizing the total weighted completion time of coflows in datacenter networks," in *Proc. 27th ACM Symp. Parallelism Algorithms Archit.*, Jun. 2015, pp. 294–303.
- [19] L. Chen, W. Cui, B. Li, and B. Li, "Optimizing coflow completion times with utility max-min fairness," in *Proc. IEEE INFOCOM - 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.
- [20] Z. Wang *et al.*, "Efficient scheduling of weighted coflows in data centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 9, pp. 2003–2017, Sep. 2019.
- [21] M. Chowdhury, S. Khuller, M. Purohit, S. Yang, and J. You, "Near optimal coflow scheduling in networks," in *Proc. 31st ACM Symp. Parallelism Algorithms Archit.*, Jun. 2019, pp. 123–134.
- [22] S. Im, B. Moseley, K. Pruhs, and M. Purohit, "Matroid coflow scheduling," in *Proc. ICALP*, 2019, pp. 1–13.
- [23] H. Wang, X. Yu, H. Xu, J. Fan, C. Qiao, and L. Huang, "Integrating coflow and circuit scheduling for optical networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 6, pp. 1346–1358, Jun. 2019.
- [24] Z. Li and H. Shen, "Co-scheduler: Accelerating data-parallel jobs in datacenter networks with optical circuit switching," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 186–195.
- [25] J. Edmonds, "Paths, trees, and flowers," *Can. J. Math.*, vol. 17, pp. 449–467, 1965.
- [26] G. Birkhoff, "Tres observaciones sobre el algebra lineal," Univ. Nac. Tucumán Rev. Ser. A, vol. 5, pp. 147–151, Aug. 1946.
- [27] J. Von Neumann, "A certain zero-sum two-person game equivalent to the optimal assignment problem," *Contrib. Theory Games*, vol. 2, pp. 5–12, Sep. 1953.

- [28] B. Towles and W. J. Dally, "Guaranteed scheduling for switches with configuration overhead," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 835–847, Oct. 2003.
- [29] M. Alizadeh et al., "PFabric: Minimal near-optimal datacenter transport," in Proc. ACM SIGCOMM Conf. SIGCOMM, Aug. 2013, pp. 435–446.
- [30] B. Tian, C. Tian, H. Dai, and B. Wang, "Scheduling coflows of multistage jobs to minimize the total weighted job completion time," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2018, pp. 864–872.
- [31] L. Wang, W. Wang, and B. Li, "Utopia: Near-optimal coflow scheduling with isolation guarantee," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2018, pp. 891–899.
- [32] W. Li, X. Yuan, K. Li, H. Qi, and X. Zhou, "Leveraging endpoint flexibility when scheduling coflows across geo-distributed datacenters," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2018, pp. 873–881.
- [33] S. Im, M. Shadloo, and Z. Zheng, "Online partial throughput maximization for multidimensional coflow," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2018, pp. 2042–2050.
- [34] S. Agarwal, S. Rajakrishnan, A. Narayan, R. Agarwal, D. Shmoys, and A. Vahdat, "Sincronia: Near-optimal network design for coflows," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Aug. 2018, pp. 16–29.
- [35] M. Shafiee and J. Ghaderi, "An improved bound for minimizing the total weighted completion time of coflows in datacenters," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1674–1687, Aug. 2018.
- [36] Q. Liang and E. Modiano, "Coflow scheduling in input-queued switches: Optimal delay scaling and algorithms," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2017, pp. 1–9.
- [37] T. Zhang, F. Ren, J. Bao, R. Shu, and W. Cheng, "Minimizing coflow completion time in optical circuit switched networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 2, pp. 457–469, Feb. 2021.
- [38] C. Xu, H. Tan, J. Hou, C. Zhang, and X.-Y. Li, "OMCO: Online multiple coflow scheduling in optical circuit switch," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.
- [39] M. Marcus and R. Ree, "Diagonals of doubly stochastic matrices," *Quart. J. Math.*, vol. 10, no. 1, pp. 296–302, 1959.
- [40] F. Dufossé and B. Uçar, "Notes on birkhoff-von neumann decomposition of doubly stochastic matrices," *Linear Algebra Appl.*, vol. 497, pp. 108–115, May 2016.
- [41] R. A. Brualdi, "Notes on the birkhoff algorithm for doubly stochastic matrices," *Can. Math. Bull.*, vol. 25, no. 2, pp. 191–199, Jun. 1982.
- [42] GUROBI. Gurobi. Accessed: Jan. 2021. [Online]. Available: http://www.gurobi.com



Haisheng Tan (Senior Member, IEEE) received the B.E. and B.S. degrees (Hons.) from the University of Science and Technology of China (USTC) and the Ph.D. degree in computer science from The University of Hong Kong (HKU). He is currently an Associate Professor with USTC. He has published more than 60 articles in prestigious journals and conferences, mainly in the areas of wireless networking, and edge and cloud computing. His research interest includes networking algorithm design and implementation. He recently received the Best Paper Award in WASA'19, CWSN'20, and PDCAT'20.



Chi Zhang received the B.Eng. degree in computer science and technology from the University of Science and Technology of China (USTC), with the honor of The Talent Program in computer and information science and technology, in 2017, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Technology. His current research interests include data center networking, cloud computing, and algorithms.



Chao Xu received the master's degree in computer science and technology from the University of Science and Technology of China in 2019. He is currently with Data Center Network Product, Alibaba Cloud.



Yupeng Li (Member, IEEE) received the Ph.D. degree in computer science from The University of Hong Kong. He was a Post-Doctoral Researcher with the University of Toronto. He is currently an Assistant Professor with Hong Kong Baptist University. His research interest includes network sciences, and in particular the optimization in practical computer and communication networks for performance guarantees. He is also interested in designing effective mechanisms for ride-sharing, a win-win-win transportation paradigm, which aims for better urban

environment, human life quality, and city operation systems. He is a member of the ACM.



Zhenhua Han (Member, IEEE) received the B.Eng. degree in electronic and information engineering from the University of Electronic Science and Technology of China in 2014 and the Ph.D. degree from The University of Hong Kong (HKU). He is currently a Researcher with Microsoft Research Asia. Many of his works have been published in top venues such as IEEE INFOCOM, USENIX OSDI, and the IEEE/ACM TRANSACTIONS ON NETWORKING. His research interests include cloud computing, cluster scheduling, machine learning

systems, online algorithms, and stochastic optimization.



Xiang-Yang Li (Fellow, IEEE) received the bachelor's degree from the Department of Computer Science and the bachelor's degree from the Department of Business Management, Tsinghua University, in 1995, and the M.S. and Ph.D. degrees from the Department of Computer Science, University of Illinois at Urbana–Champaign, in 2000 and 2001, respectively. He was a Full Professor with the Illinois Institute of Technology, Chicago, USA. He is currently a Full Professor and the Executive Dean with the School of Computer Science and Technology,

USTC, Hefei, China. He published a monograph *Wireless Ad Hoc and Sensor Networks: Theory and Applications.* His research interests include artificial intelligent Internet of Things, mobile computing, data sharing and trading, and privacy. He is a fellow of the ACM and an ACM Distinguished Scientist.