Journal of Network and Computer Applications 135 (2019) 47–61

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Joint optimization of tasks placement and routing to minimize Coflow Completion Time

Yingya Guo [a,c], Zhiliang Wang [b,c,*], Han Zhang [d], Xia Yin [a,c], Xingang Shi [b,c], Jianping Wu [a,c]

[a] Department of Computer Science and Technology, Tsinghua University, China
[b] Institute for Network Sciences and Cyberspace, Tsinghua University, China
[c] Beijing National Research Center for Information Science and Technology, China
[d] School of Cyber Space and Technology, Beihang University, China

ARTICLE INFO

Keywords:
Data-parallel job
Coflow Completion Time
Task placement
Routing inter-DC WAN
Approximation algorithm

ABSTRACT

Coflow is defined as the parrellel flows between two successive computation stages of data-parallel jobs. Reducing Coflow Completion Time (CCT) is important to improve the performance of data-parallel applications in geo-distributed datacenter networks. CCT is influenced by two factors: tasks placement and coflow routing. Previous works that optimize only one factor are insufficient in reducing CCT.

In this paper, we explore the joint optimization of tasks placement and coflow routing to reduce CCT. We formulate the joint optimization of a single coflow as a Mixed Integer Non-Linear Programming problem, and propose an approximate algorithm PRO with an approximation ratio of $(1 + \epsilon)$. Moreover, we propose algorithms to optimize multiple coflows scheduling offline and online. Through extensive experiments, we demonstrate that our proposed algorithms have superior performance in reducing CCT or average CCT compared with other algorithms.

1. Introduction

As with the rapid development of Internet, many geo-distributed applications, e.g., online searching services, e-commerce services, generate huge amount of data across different regions in geo-distributed datacenters. Such huge amount of data is analyzed by data-parallel jobs to provide a prompt response to user requests. A traditional method to process the huge amount of geo-distributed data is to move all the data to a single datacenter and analyze them with data-parallel jobs inside a datacenter. However, there are some limitations existing in this centralized manner. First, Inter-DataCenter Wide Area Network (Inter-DC WAN) bandwidth is an expensive and limited resource. Moving huge amount of data across inter-DC WAN consumes a lot of bandwidth (Laoutaris et al., 2011). Second, moving data across regions may expose user information and privacy, considering privacy and security concerns (Hu et al., 2016), it is undesirable to move the data across regions. Third, with data growing explosively, a single datacenter may not be capable of accommodating such huge amount of data in the future (Ji and Li, 2016). Therefore, it is more efficient and secure to run the data-parallel jobs across geographically distributed datacenters. The data after locally processed by data-parallel jobs is smaller in size and can be transferred more efficiently among inter-DC WAN. In this paper, we focus on optimizing completion time of data-parallel jobs in inter-DC WAN to reduce response time of user requests. Optimizing completion time of data-parallel jobs means reducing Coflow Completion Time (CCT). Here, coflow (Chowdhury et al., 2011) is defined as the parallel flows between two successive computation stages of data-parallel jobs and Coflow Completion Time (CCT) is the completion time of the last flow in the parallel flows. Therefore, our goal is to reduce CCT. Fig. 1 shows a data-parallel job across inter-DC WAN. Coflow is generated in geo-distributed datacenters and needs to be transferred across inter-DC WAN to reduce tasks.

For coflow transfer, the bottleneck is scarce inter-DC WAN bandwidth (Hu et al., 2016). Therefore, for CCT optimization, we should make the most of bandwidth resources. Two factors, task placement and coflow routing, influence a coflow optimization. The placement of reduce tasks determines the endpoints of parallel flows in a coflow. A better reduce tasks placement can reduce the volume of data trans-

* Corresponding author. East Main Building 9-401, Tsinghua University, Beijing, China.
E-mail address: wzl@cernet.edu.cn (Z. Wang).

https://doi.org/10.1016/j.jnca.2019.02.031
Received 8 August 2018; Received in revised form 22 February 2019; Accepted 27 February 2019
Available online 4 March 2019
1084-8045/© 2019 Elsevier Ltd. All rights reserved.

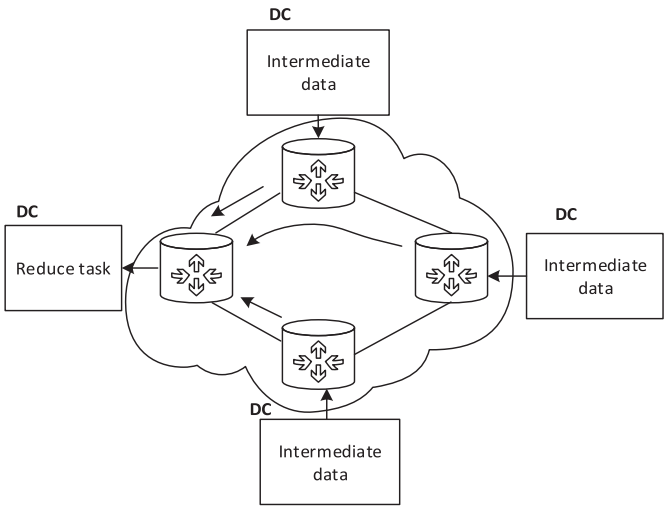


Fig. 1. A data-parallel job across inter-DC WAN. The parallel flows between map tasks and reduce task are called a coflow.

ferred in inter-DC WAN and this will influence the bandwidth resources assignment. Coflow routing influences the bandwidth obtained by the individual flows. The previous studies are insufficient in reducing CCT since they either optimize reduce tasks placement (Hu et al., 2016) or coflow routing (Zhang et al., 2016a). In our optimization problem, task placement and routing are jointly optimized for a coflow. However, these two factors intertwine with each other. More concretely, the placement of tasks determines the destination of parallel flows in a coflow, and thus influencing the routing of flows in a coflow. Therefore, the joint optimization of those two factors poses a great challenge.

In this paper, we propose a framework (see Fig. 2) that jointly optimizes tasks placement and coflow routing. The information for optimizing tasks placement can be obtained from applications in upper layer and the information for optimizing coflow routing can be obtained from network resources in lower layer. A joint optimization of applications in upper layer and network resources in lower layer can better utilize network resources to improve the performance of applications. We first formulate the joint optimization problem as a Mixed Integer Non-Linear Programming (MINLP) problem. Then an approximate algorithm PRO (Placement and Routing Optimization) with an approximation ratio of $(1 + \epsilon)$ is proposed to solve the problem. Next, we introduce offline and online scheduling algorithms to optimize average CCT of multiple coflows. Extensive experiments demonstrate that CCT can be greatly reduced by PRO compared with the algorithms that only optimize reduce tasks placement or coflow routing. Moreover, our proposed offline and online scheduling algorithms have superior performance in minimizing average CCT.

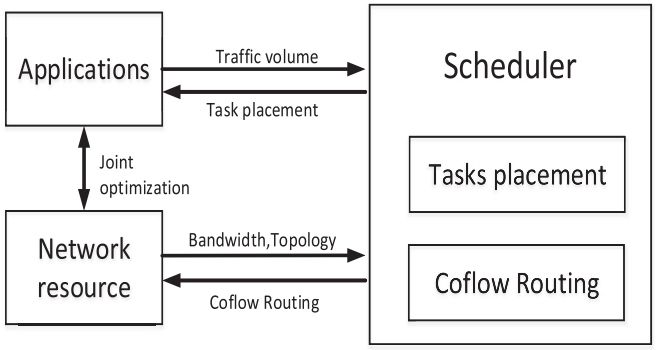


Fig. 2. Proposed framework. The inter-DC WAN bandwidth, topology and traffic volume are known ahead. Our scheduler jointly optimizes tasks placement and coflow routing to minimize CCT.

Our main contributions are summarized as follows:

- We study the joint optimization of tasks placement and coflow routing to minimize CCT.
- We develop a mathematical model and formulate the joint optimization problem as a MINLP problem, which is NP-hard. We propose an approximate algorithm PRO with theoretical guarantees to optimize CCT. We also develop offline and online scheduling algorithms to optimize average CCT of multiple coflows.
- We conduct extensive experiments to demonstrate that PRO reduces CCT compared with the state-of-art placement-optimization-only methods and routing-optimization-only methods, respectively. In addition, our proposed offline and online scheduling algorithms outperform other algorithms in minimizing average CCT.

The rest of paper is organized as follows. Section 2 introduces the related work. Section 3 introduces the motivation of our work. Section 4 formulates the model of a single coflow optimization. In Section 5, we propose an approximate algorithm PRO to solve the problem and analyze the upper and lower bound of PRO. In Section 6, we discuss scheduling algorithms for offline and online optimization of multiple coflows. In Section 7, we make a evaluation of our proposed algorithms on real network topologies. We make a conclusion in Section 8.

2. Related work

In this section, we briefly present related works on coflow scheduling, coflow routing and tasks placement, respectively.

Coflow scheduling: Mosharaf et al. first introduce the concept of coflow and propose an architecture Orchestra to optimize the scheduling of coflow in datacenter network in (Chowdhury et al., 2011). Afterwards, they propose Varys (Chowdhury et al., 2014) and Aalo (Chowdhury and Stoica, 2015) to minimize CCT and guarantee coflow deadlines with efficient coflow scheduling algorithms. In (Qiu et al., 2015), Qiu et al. minimize total weighted completion time of coflows in datacenter networks. They are the first to propose polynomial deterministic algorithm for coflows scheduling problem with an approximate ratio of $\frac{67}{3}$. In (Chen et al., 2016), to provide differential treatment to coflows with different degrees of sensitivity, Chen et al. introduce a new utility optimal scheduler among competing coflows and satisfy the max-min fairness among these coflows. In (Dogar et al., 2014), R. Dogar et al. design and implement a decentralized coflow scheduling system which schedules coflows in a FIFO order but dynamically changes the order of scheduling when heavy tasks are encountered. In (Zhang et al., 2016b), Zhang et al. first propose CODA to automatically identify coflows with machine learning techniques and propose an error-tolerant scheduler that can tolerate occasional identification errors. In (Agarwal et al., 2018), Agarwal et al. present a near optimal network design that supports priority scheduling for inter-coflows and achieves near-optimal CCT. They ignore the optimization for intra-coflow. In (Li et al., 2018a), Li et al. focus on optimizing endpoint placement to minimize the average CCT of coflows across geo-distributed datacenters. They only optimize task placement without considering routing optimization. In (Wang et al., 2018), Wang et al. investigate that CCT could be further decreased by jointly leveraging multiple coflow-level attributes. They focus on inter-coflow optimization with preemptive scheduling in datacenter networks, while our work mainly focuses on intra-coflow optimization and adopts non-preemptive scheduling to optimize inter-coflow in inter-DC WAN. In (Zhao et al., 2018), Zhao et al. propose an online reducer placement framework to minimize the average CCT in cloud clusters. It ignores routing optimization of coflow.

Coflow routing: In (Zhao et al., 2015), Zhao et al. optimize average CCT in a intra-DC network by integrating routing and scheduling.

In coflow scheduling, they choose to schedule the coflow with Minimum Remaining Time First (MRTF) scheduling, which is preemptive, to minimize average CCT. In coflow routing, they choose to formulate the path selection problem into an integer multi-commodity flow problem and propose a heuristic algorithm to solve it. In (Li et al., 2016), Li et al. propose an online algorithm OMCoflow to optimize the routing and scheduling of coflows, which avoids the problem of frequently rerouting in (Zhao et al., 2015). The theoretical analysis shows that OMCoflow can obtain a good competitive ratio in minimizing the average CCT. In (Zhang et al., 2016a), Zhang et al. optimize the transfer routing with k-shortest paths and adaptive scheduling to guarantee the transfer deadline. In (Li et al., 2018b), Li et al. propose a bandwidth management and network path selection mechanism for multiplexed datacenter networks. They optimize at flow granularity, while we optimize at coflow granularity.

Tasks placement (Pu et al., 2015): is the first paper to achieve low query response times in inter-DC WAN by optimizing the data and reduce tasks placement. In this work, Pu et al. assume the bottlenecks are the bandwidth of uplinks and downlinks between the geo-distributed datacenter sites and inter-DC WAN core. They employ an efficient linear formulation to solve the problem of tasks placement. In (Hu et al., 2016), Hu et al. are the first to optimize the reduce tasks placement to minimize CCT in inter-DC WAN. They propose Flutter, an online network-aware and stage-aware scheduling algorithm, to determine the placement of reduce tasks in inter-DC WAN. The problem is formulated as an integer linear programming and transformed to an LP problem. The CCT can be reduced by up to 25% in Flutter. Li et al. (2017) propose to minimize the traffic generated by geo-distributed big data analytic through jointly optimizing input data movement and tasks placement. They employ chance-constrained optimization techniques to guarantee the job completion time with high probability. In (Zeng et al., 2017), Zeng et al. optimize CCT through joint optimization of flow scheduling and processor placement in datacenter networks, but they do not optimize coflow routing. In (Gu et al., 2014), Gu et al. only optimize task placement to reduce energy cost and transmission cost of data chunks in geo-Distributed datacenter networks. In (Prisacari et al., 2014), Prisacari et al. optimizes flow routing and task placement in Dragonfly Networks. Both of these two works optimize at flow granularity, while our work optimizes at coflow granularity.

The routing and reduce tasks placement both influence the completion time of a coflow. However, there are no previous works that jointly optimize these two factors in inter-DC WAN. Therefore, in this paper, we consider the joint optimization of these two factors to reduce CCT. This paper is an extension of our previous work (Guo et al., 2017). Here, we formulate the problem, propose the approximation algorithm, present theoretical analysis of the approximation algorithm, and evaluate our proposed method with extensive experiments.

3. A motivating example

In this section, we introduce the motivation of our work with an example.

To explain the motivation of jointly optimizing two factors on reducing CCT, we give an example in Fig. 3. As shown in Fig. 3, we obtain CCT in methods only optimizing task placement, only optimizing coflow routing and jointly optimizing two factors. Here, for simplicity, we assume that there is one coflow and one reduce task. For method only optimizing task placement, datacenter 2 is selected to place the reduce task according to the maximizing data locality principle (Pu et al., 2015; Zaharia et al., 2010; Isard et al., 2009) (see Fig. 3(b)). The routes of the flows are determined according to the shortest path routing protocol. Once the routing paths are obtained, CCT is determined by calculating the maximum of the completion time of four flows in a coflow. Therefore, the CCT of method only optimizing task placement is $max\{\frac{3}{1}, \frac{(6+3)}{1}, \frac{1}{2}, \frac{(1+1)}{2}\} = 9s$. For method only optimizing coflow routing, the routes of the flows is determined by formulating routing problem as a MINLP (Zhao et al., 2015) (see Fig. 3(c)). Here, we set datacenter 2 to place the reduce task. The CCT of method only optimizing coflow routing is $max\{\frac{3}{1}, \frac{6}{2}, \frac{6}{2}+\frac{1}{2}, \frac{6}{2}+\frac{(1+1)}{2}\} = 4s$. For method jointly optimizing two factors, datacenter 5 is computed to place reduce task. Moreover, the routing paths of flows from datacenter 2 to 5, from datacenter 3 to 5, from datacenter 1 to 5, from datacenter 4 to 5, are computed to be $2 \rightarrow 3 \rightarrow 4 \rightarrow 5$, $3 \rightarrow 2 \rightarrow 1 \rightarrow 5$, $1 \rightarrow 5$ and $4 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow 5$, respectively (see Fig. 3(d)). The CCT of method jointly optimizing two factors is $max\{\frac{3}{2}, \frac{7}{2}, \frac{3}{2}+\frac{1}{1}, \frac{3}{2}+\frac{(1+1)}{1}\} = 3.5s$. Therefore, it is necessary to consider coflow routing and reduce task placement into coflow optimization.

4. Single coflow optimization

Multiple coflows optimization is a complex and difficult problem. Therefore, in this section, we first study the problem of minimizing

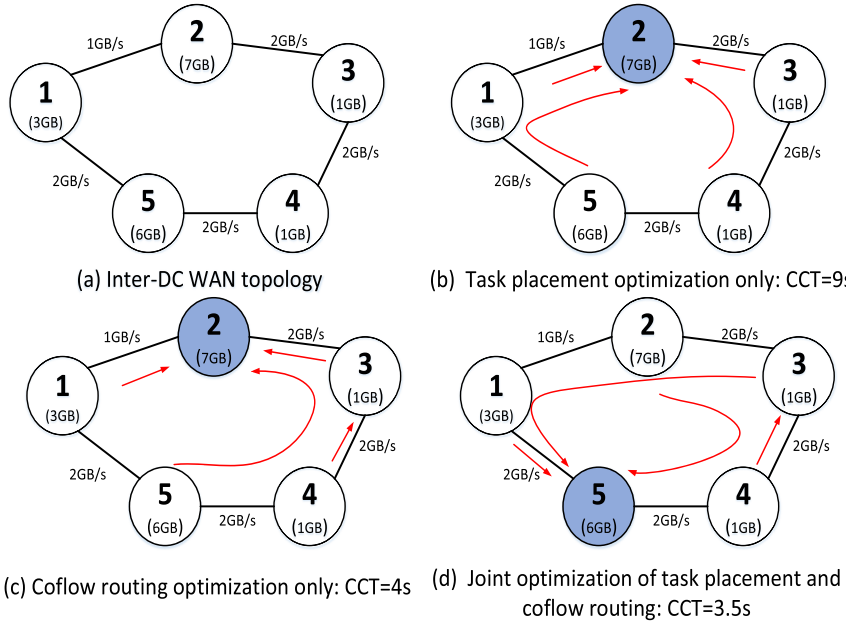


Fig. 3. An example of CCT computed in different optimization methods: (a) inter-DC WAN topology containing five datacenters; (b) CCT computed in method only optimizing task placement; (c) CCT computed in method only optimizing coflow routing; (d) CCT computed in method jointly optimizing task placement and coflow routing. Here, the numbers in parentheses and on the links represent volume of traffic and link bandwidth, respectively. Red lines show the routing paths computed by different optimization strategies. The blue node is the datacenter which is selected to place reduce task. For simplicity, OSPF link weights are set to be 1. (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

Table 1
Definition of notations.

D	Datacenter set
E	Inter-DC link set
R_l	Available capacity of edge $l \in E$
π	Reduce task set
C	Coflow completion time
y_{ij}	A binary variable. $y_{ij} = 1$ denotes that reduce task i is placed at datacenter j
m_{kj}	The volume of intermediate data transferred between datacenters k and j
b_{kj}	The computed bandwidth for a flow between datacenters k and j
x_{kj}^t	A binary variable. $x_{kj}^t = 1$ denotes the flow between datacenters k and j is routed through t-th path
$I_{p_{kj}^t}^l$	$I_{p_{kj}^t}^l = 1$ if $l \in p_{kj}^t$; otherwise, $I_{p_{kj}^t}^l = 0$
p_{kj}	The path set between datacenter k and j
p_{kj}^t	The t-th path between datacenter k and j
S_i	The set of input datacenters that transfer flows for reduce task i
Q_j	The number of available computing slot of datacenter j
ρ	The reduce tasks placement set

completion time of a single coflow. We formulate the mathematical model of joint optimization problem as a MINLP. Then we transform the MINLP into an Linear Programming (LP) problem. Finally, we round the fractional solution to make it feasible to the original problem.

4.1. Network model

Suppose that resources of intra-DC network are sufficient and the computation time of map phase within each datacenter is negligible (Hu et al., 2016). The intermediate data size and the number of reduce tasks are known ahead. This is reasonable because in Spark, the intermediate data can be obtained through the MapOutputTracker (Hu et al., 2016) and the number of reduce tasks can also be known through the TaskSet in Spark DAG scheduler (Zaharia et al., 2012). All flows are routed through a single path to avoid packets reordering.

The Inter-DC WAN consists of a datacenter set $D = \{1, 2, \ldots, d\}$ and inter-DC links set E. R_l represents the available capacity of link l, which equals total capacity minus background traffic. m_{kj} is the volume of intermediate data transferred between datacenters k and j. y_{ij} indicates whether reduce task i is placed at datacenter j or not. x_{kj}^t denotes whether the flow between datacenters k and j is routed through t-th path or not. The definition of the notations is summarized in Table 1. We jointly optimize the placement of reduce tasks y_{ij} and the routing of a coflow x_{kj}^t among Inter-DC WAN so that CCT can be minimized. Given the information aforementioned, our optimization problem can be formulated as follows:

$$\textbf{minimize} \quad C \tag{1}$$

$$s.t. \sum_{j=1}^{d} y_{ij} = 1, \quad \forall i \in \pi \tag{1a}$$

$$\sum_{i=1}^{\pi} y_{ij} \le Q_j, \quad \forall j \in D \tag{1b}$$

$$C = y_{ij} m_{kj} / b_{kj}, \quad \forall i \in \pi, k \in S_i, j \in D \tag{1c}$$

$$\sum_{i \in \pi} \sum_{k \in S_i} \sum_{j \in D} \sum_{t=1}^{|p_{kj}|} y_{ij} b_{kj} x_{kj}^t I_{p_{kj}^t}^l \le R_l, \quad \forall l \in E \tag{1d}$$

$$\sum_{t=1}^{|p_{kj}|} y_{ij} x_{kj}^t = 1, \quad \forall k \in S_i, i \in \pi, j \in D \tag{1e}$$

$$x_{kj}^t, y_{ij} \in \{0, 1\}, b_{kj} \ge 0 \tag{1f}$$

As shown in equation (1), our goal is to minimize CCT. Equations (1a)–(1e) are constraints which are composed of two parts: placement constraints ((1a)-(1b)) and routing constraints ((1c)-(1e)). Equation (1a) requires a reduce task must be placed at one and only one datacenter. In equation (1b) shows that total number of reduce tasks assigned to datacenter j cannot exceed its available computing slot Q_j. b_{kj} is a variable, which indicates the computed bandwidth of the flow between datacenters k and j. Equation (1c) computes CCT, which is the completion time of the last flow in a coflow. Equation (1d) constrains that the bandwidth of link l consumed by a coflow should not exceed the available capacity of l. Equation (1e) implies that each flow chooses one and only one path between the source and destination. (1f) shows that x_{kj}^t, y_{ij} are binary variables and b_{kj} is non-negative.

The integer variables x_{kj}^t, y_{ij} in (1) make the problem a mixed integer programming and a product of two variables in (1d) makes the problem a nonlinear programming. Therefore, the problem formulated in (1) is a MINLP problem, which is generally proved to be NP-hard (Cormen et al., 2009).

4.2. Transformation into an LP problem

According to problem formulation (1), we solve the problem in two steps. We first decide the placement of reduce tasks. The details of task placement algorithm will be depicted in details in section 5. We assume the placement of reduce tasks composes a set ρ. Then in the second phase, we relax the original MINLP problem into an LP problem and solve the LP problem efficiently with an LP solver. When the placement of reduce tasks is determined, our problem can be formulated as (2):

$$\textbf{minimize} \ C \tag{2}$$

$$C = m_{kj} / b_{kj}, \quad \forall i \in \pi, k \in S_i, j \in \rho \tag{2a}$$

$$\sum_{i \in \pi} \sum_{k \in S_i} \sum_{j \in \rho} \sum_{t=1}^{|p_{kj}|} b_{kj} x_{kj}^t I_{p_{kj}^t}^l \le R_l, \quad \forall l \in E \tag{2b}$$

$$\sum_{t=1}^{|p_{kj}|} x_{kj}^t = 1, \quad \forall k \in S_i, i \in \pi, j \in \rho \tag{2c}$$

$$x_{kj}^t \in \{0, 1\}, b_{kj} \ge 0 \tag{2d}$$

Equation (2a) is not a linear form. To transform (2a) into a linear form, we introduce a new variable α to be $\frac{1}{C}$. By substituting $\frac{1}{C}$ with α, equation (2a) can be transformed into a linear form: $m_{kj} * \alpha = b_{kj}$. We observe that there are binary variables x_{kj}^t, so we relax the constraint

Algorithm 1 PRO.

Input: E, R_l, D, S_i, Q_j
Output: y_{ij}, C, x_{kj}^t
1 **foreach** $j \in D$ **do**
2 p_{kj}^t = solve problem (4);
3 (C_j, x_{kj}^t) = Rounding (p_{kj}^t) ;
4 Sort datacenters in D as D' in increasing order of C_j ;
5 **foreach** $i \in \pi$ **do**
6 **foreach** $j \in D'$ **do**
7 **if** $Q_j > 0$ **then**
8 $y_{ij} = 1$;
9 $Q_j = Q_j - 1$;
10 break ;
11 p_{kj}^t = solve problem (4);
12 (C, x_{kj}^t) = Rounding (p_{kj}^t) ;
13 return (y_{ij}, C, x_{kj}^t) ;

(2d) to $x_{kj}^t \in [0,1]$. Then our problem can be transformed into the following formulation:

$$\textbf{minimize } C \tag{3}$$

$$m_{kj} * \alpha = b_{kj}, \quad k \in S_i, i \in \pi, j \in \rho \tag{3a}$$

$$(2b) - (2c) \tag{3b}$$

$$x_{kj}^t \in [0,1], b_{kj} \geq 0 \tag{3c}$$

In problem (3), we decide to transform equation (2b) into an linear expression. Since $m_{kj}{}^* \alpha = b_{kj}$, equation (2b) can be transformed into $\sum_{i\in\pi}\sum_{k\in S_i}\sum_{j\in\rho}\sum_t m_{kj} * \alpha * x_{kj}^t I_{p_{kj}^t}^l \leq R_l$. Because there is a product of two variables (α and x_{kj}^t) in the inequation, we set $P_{kj}^t = \alpha * x_{kj}^t$ and problem (3) can be transformed to problem (4).

$$\textbf{maximize } \alpha \tag{4}$$

$$\sum_{i\in\pi}\sum_{k\in S_i}\sum_{j\in\rho}\sum_{t=1}^{|p_{kj}|} m_{kj} P_{kj}^t I_{p_{kj}^t}^l \leq R_l, \quad l \in E \tag{4a}$$

$$\sum_t P_{kj}^t = \alpha, \quad k \in S_i, j \in \rho \tag{4b}$$

$$P_{kj}^t \geq 0 \tag{4c}$$

P_{kj}^t, α are variables and problem (4) is an LP. We can use an LP solver (e.g., CPLEX, GUROBI) to solve problem (4) efficiently and obtain the optimal solution P_{kj}^t, α. Because binary variable $x_{kj}^t \in \{0,1\}$ is relaxed to $x_{kj}^t \in [0,1]$, the solutions x_{kj}^t we obtain may be fractional. The fractional solutions are infeasible to problem (2). Therefore, to make the solution feasible to the original problem, we propose a deterministic rounding algorithm. The details of the rounding algorithm are shown in section 5.

5. Algorithm description

In this section, we first introduce the deterministic approximation algorithm PRO that jointly optimizes coflow routing and tasks placement. Next, we present the theoretical analysis of PRO and give the upper and lower bound of PRO. Finally, we analyze the complexity of PRO.

5.1. Algorithm details

We propose an approximate algorithm PRO (Placement and Routing Optimization) to solve the optimization problem (1). The algorithm works as follows: we sort the placement of the reduce tasks in increasing order of CCT. Then we choose the placement with minimal CCT and available computing slot. When the reduce tasks placement is determined, we solve problem (4) to obtain the routing that minimizes CCT. The details of PRO are shown in Algorithm 1.

Algorithm 2: Rounding.

Input: p_{kj}^t
Output: (C, x_{kj}^t)
1: round the solution $x_{kj}^{t'} = 1$, with $t', p_{kj}^{t'} = \mathbf{max}_t P_{kj}^t$
2: Solve the LP to obtain α with $x_{kj}^{t'} = 1$ in problem (4)
3: $C = 1/\alpha$
4: return (C, x_{kj}^t)

5.1.1. PRO

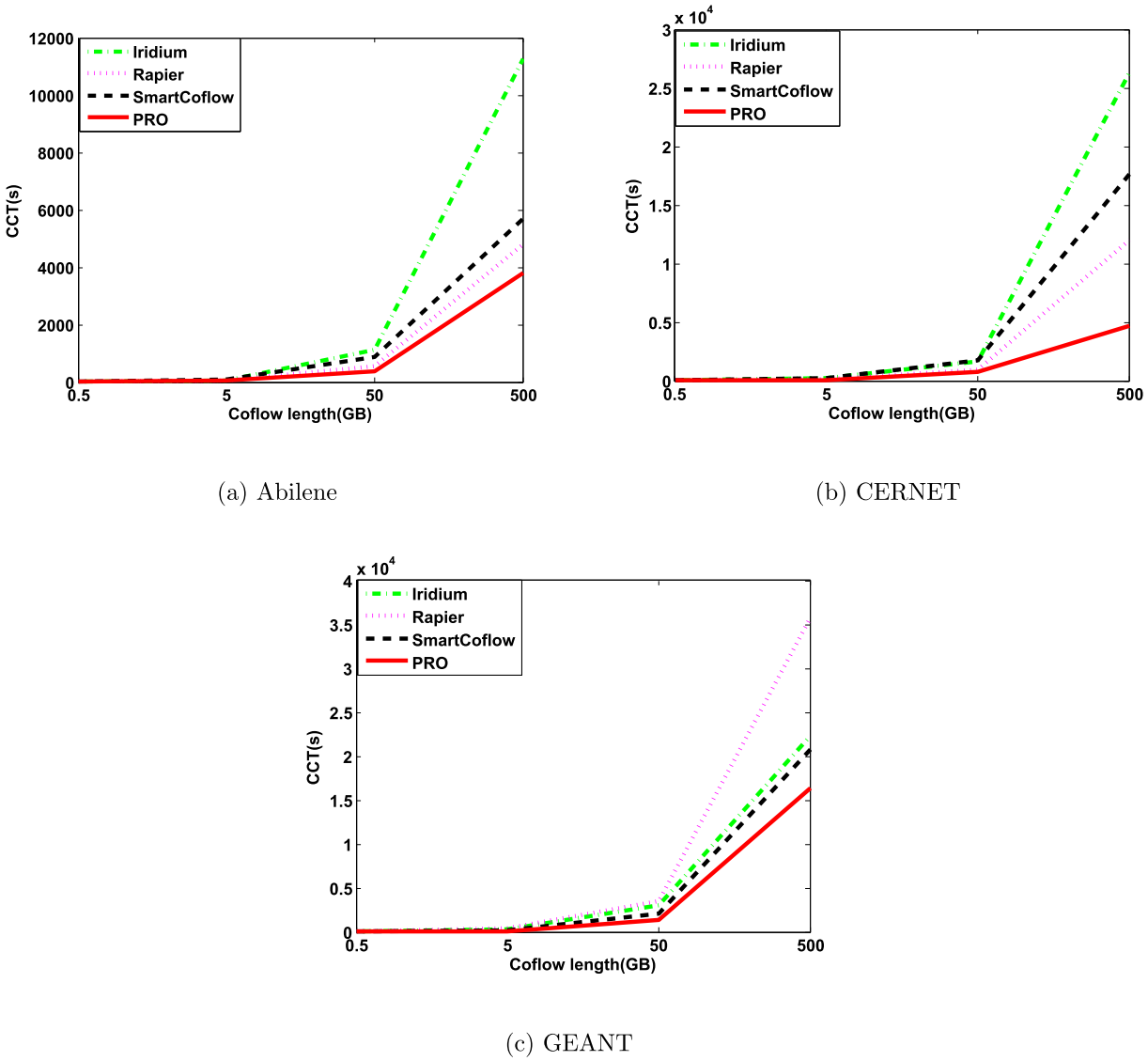
In Algorithm 1, we first compute C_j with datacenter j chosen as the reduce tasks placement (line 1–3). Then we sort the datacenters in increasing order of C_j (line 4). For each reduce task, we greedily determine its placement in a datacenter with minimal CCT and available computing slot (line 5–10). After the reduce tasks placement is determined, we transform the original problem into an LP problem (4) and solve it with an LP solver (line 11). Finally, we round the solution to obtain the feasible routing to problem (1) (line 12).

5.1.2. Rounding

In Algorithm 2, we introduce Rounding algorithm to obtain the flow routes and CCT. The Rounding algorithm is designed as follows: given the solutions x_{kj}^t, we route the flows on the path t', which satisfies the constraints $P_{kj}^{t'} = \mathbf{max}_t P_{kj}^t$ (line 1). Substituting P_{kj}^t with $\alpha * x_{kj}^t$ in equation (4a), problem (4) turns to an LP with variables α. We can solve this problem with LP solvers easily and obtain α (line 2). Finally, we can calculate CCT by equation $C = 1/\alpha$ (line 3). Therefore, through solving the LP problem, we can obtain the coflow routing with minimal CCT.

Table 2
Three network topologies.

Topologies	Nodes	Links
Abilene	12	30
CERNET	14	32
GEANT	23	74



(a) Abilene

(b) CERNET

(c) GEANT

Fig. 4. CCT under various coflow lengths. We can obtain that under different coflow lengths, our proposed PRO obtains a lower CCT compared to Rapier and Iridium.

Table 3
Improvement ratio of PRO in Abilene.

Coflow length (GB)	0.5	5	50	500	Average
Iridium	53.3%	48.1%	72.9%	22.4%	49.2%
Rapier	17.8%	21.2%	40.1%	24.5%	25.9%
SmartCoflow	52.2%	44.1%	69.4%	48.2%	53.5%

5.2. Approximation ratio analysis

In this part, we give an analysis of the lower bound and upper bound of PRO in Theorem 1 and Theorem 2. Let C_O denote the optimal CCT of problem (2) and C_R denote the CCT of problem (4) with rounded x_{kj}^t.

Theorem 1. ***(Lower bound)*** *We have* $C_O \leq C_R$.

Proof. If we can prove that rounded x_{kj}^t of problem (4) satisfies the constraints (2a)-(2d) and is a feasible solution to problem (2), according to the fact that an optimal solution to a problem always performs no worse than the feasible solution, we can easily obtain $C_O \leq C_R$. Therefore, in the following, we prove rounded x_{kj}^t is a feasible solution to

Table 4
Improvement ratio of PRO in Cernet.

Coflow length (GB)	0.5	5	50	500	Average
Iridium	64.1%	66.3%	64.1%	60.6%	63.8%
Rapier	40.8%	44.3%	41.3%	43.1%	42.4%
SmartCoflow	62.7%	64.5%	64.0%	60.8%	63.0%

Table 5
Improvement ratio of PRO in Geant.

Coflow length (GB)	0.5	5	50	500	Average
Iridium	34.9%	34.5%	33.5%	37.3%	35.1%
Rapier	56.6%	55.1%	54.9%	60.1%	56.7%
SmartCoflow	31.5%	32.9%	33.1%	37.0%	33.6%

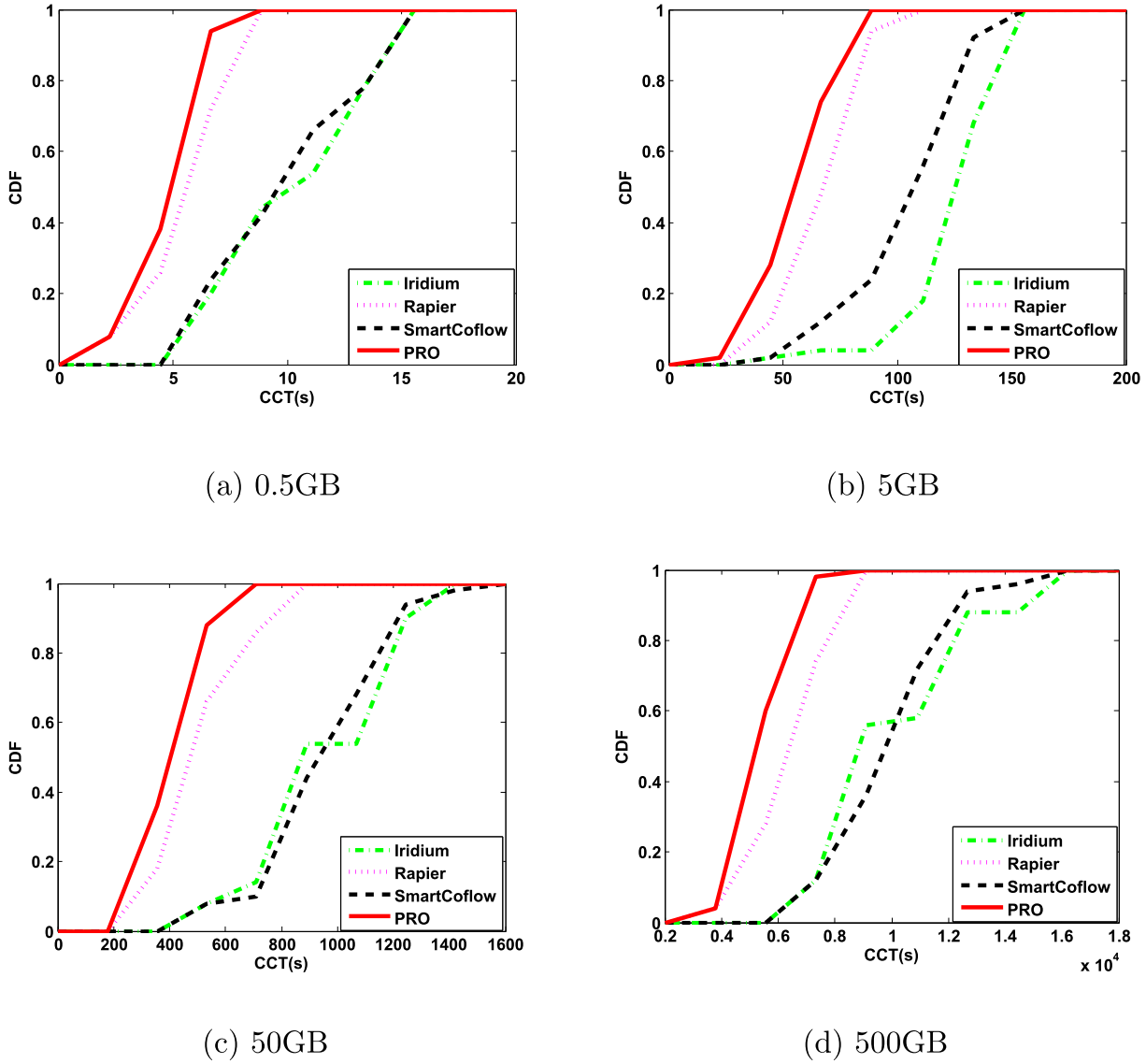


Fig. 5. CDF curves of CCT under various coflow lengths of Abilene.

problem (2).

First, because x_{kj}^t is a feasible solution to problem (4), we can obtain that it satisfies (4a) and (4b). According to (4a) and $P_{kj}^t = \alpha * x_{kj}^t$, we can obtain the following relation:

$$\sum_{i\in\pi}\sum_{k\in S_i}\sum_{j\in\rho}\sum_{t=1}^{|p_{kj}|} b_{kj} * x_{kj}^t I_{p_{kj}^t}^l = \sum_{i\in\pi}\sum_{k\in S_i}\sum_{j\in\rho}\sum_{t=1}^{|p_{kj}|} m_{kj}\alpha * x_{kj}^t I_{p_{kj}^t}^l$$

$$= \sum_{i\in\pi}\sum_{k\in S_i}\sum_{j\in\rho}\sum_{t=1}^{|p_{kj}|} m_{kj} P_{kj}^t I_{p_{kj}^t}^l \le R_l$$

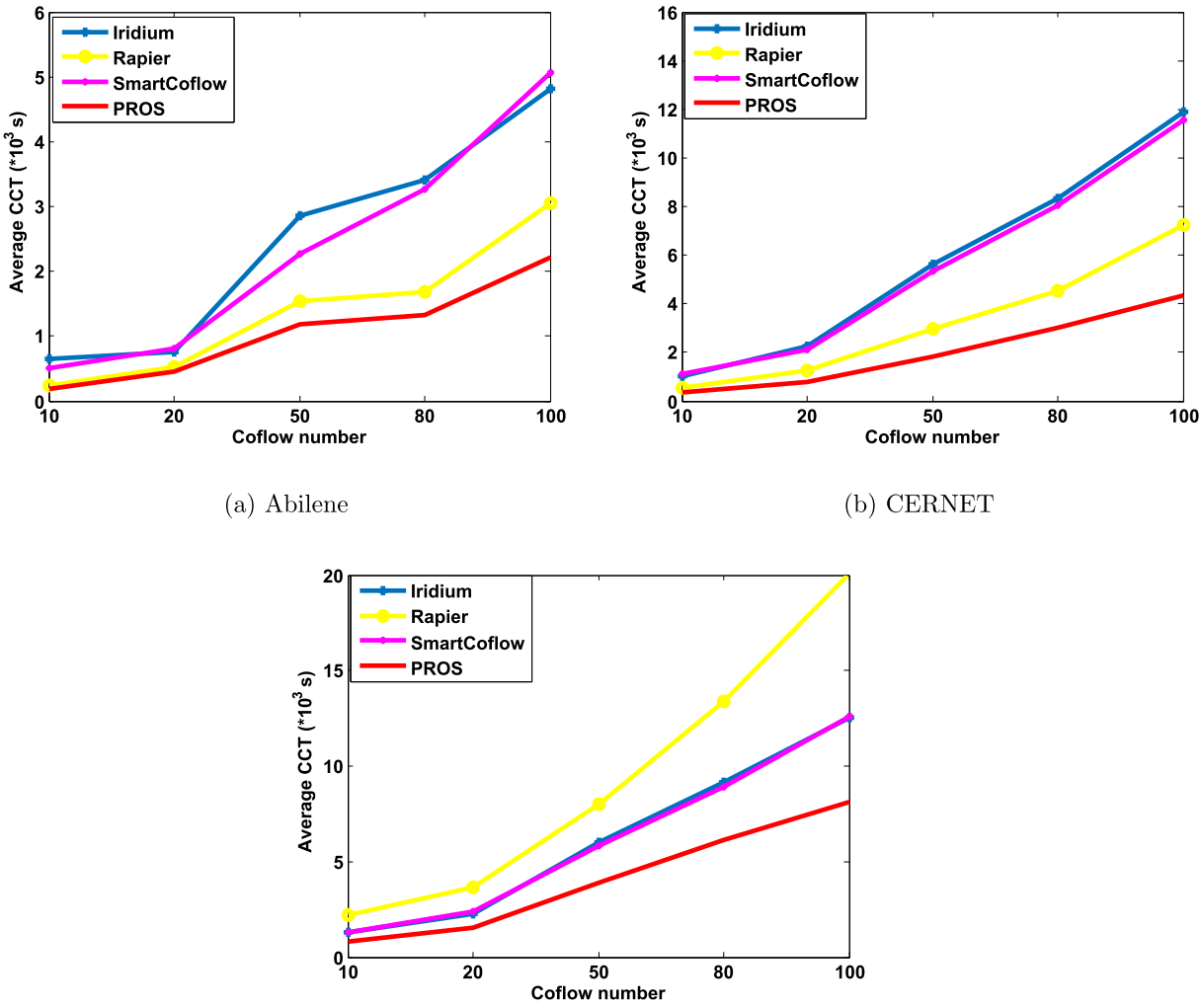
which satisfies constraint (2b). What's more, according to (4b), we have the following equation:

$$\sum_t P_{kj}^t = \sum_t \alpha * x_{kj}^t = \alpha, \quad k \in S_i, j \in \rho$$

We can obtain x_{kj}^t satisfies $\sum_t x_{kj}^t = 1$, which is the (2c). Because x_{kj}^t is a rounded solution, therefore, we can easily prove: $x_{kj}^t \in \{0,1\}$, which satisfies equation (2d)

According to our above analysis, we can prove the rounded x_{kj}^t of problem (4) satisfies the constraints (2a)-(2d) and is a feasible solution to problem (2). Therefore, we have $C_O \le C_R$.

Theorem 2. ***(Upper bound)*** $C_R \le (1+\epsilon)^* C_O, \forall \epsilon > 0$.



(a) Abilene

(b) CERNET

(c) GEANT

Fig. 6. Average CCT under various number of coflows.

Proof. We assume the optimal objective value of problem (4) is α_{upper} and $\alpha_O = 1/C_O$. Because problem (4) is a relaxation of problem (1), we can easily conclude that:

$$\alpha_{upper} \geq \alpha_O \tag{5}$$

In Algorithm 2, we route the flow on the path k with maximum $p_{kj}^t = \alpha_{upper} * x_{kj}^t$, therefore, $\forall \epsilon > 0$, we can obtain:

$$p_{kj}^t > \frac{\alpha_{upper}}{1+\epsilon} * x_{kj}^t \tag{6}$$

Substituting p_{kj}^t with equation (6) in (4a), we can have:

$$\sum_{k \in S} \sum_j \sum_t m_{kj} * \frac{\alpha_{upper}}{1+\epsilon} * x_{kj}^t < \sum_{k \in S_i} \sum_j \sum_t m_{kj} * p_{kj}^t \leq R_l \tag{7}$$

Therefore, $\frac{\alpha_{upper}}{1+\epsilon}$ is a feasible solution to problem (4) with given rounded x_{kj}^t α_R is the optimal solution to problem (4) with given rounded x_{kj}^t. We can obtain:

$$\alpha_R \geq \frac{\alpha_{upper}}{1+\epsilon} \geq \frac{\alpha_O}{1+\epsilon} \tag{8}$$

Finally, we can obtain, for $\forall \epsilon > 0$:

$$C_R \leq (1+\epsilon) * C_O \tag{9}$$

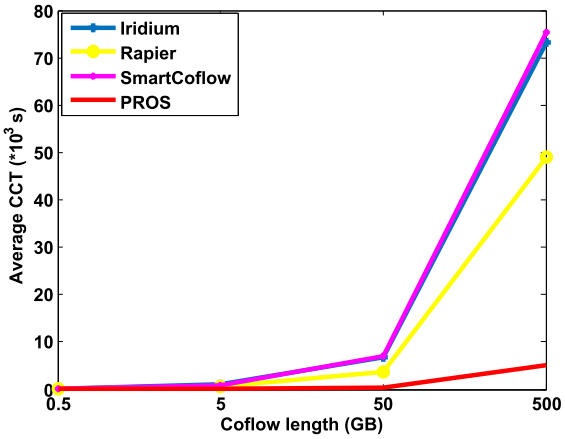
By previous analysis, the approximate ratio of our proposed algorithm is $(1 + \epsilon)$.

5.3. Algorithm complexity

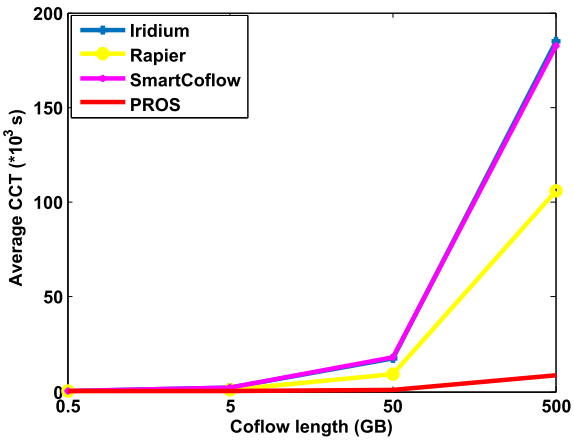
Let $|D| = n, |E| = m, |\pi| = k$, in our algorithm, the time complexity of determining the reduce tasks placement is $O(kn)$ and the complexity of finding all the paths between two datacenters is $O()n(m + nlogn)$. The variables of LP problem are $O(kn^2(m + nlogn))$ and constraints are $O(kn)$. Therefore, the complexity of solving the LP problem is $O(LP(O(kn^2(m + nlogn)), O(kn)))$, where $LP(m, n)$ represents the running time of solving an LP with m variables and n constraints.

6. Multiple coflows scheduling

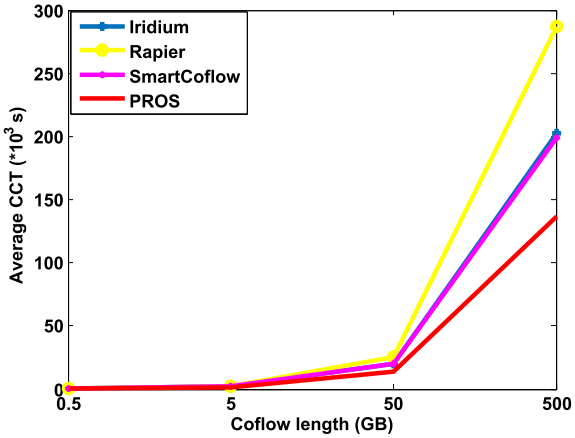
In this section, we consider optimizing scheduling of multiple coflows $F = \{f_i, i \in [1, r]\}, |F| = r$. We need to optimize the scheduling between inter-coflows to minimize the average CCT of multiple coflows. We first present an offline algorithm for multiple coflows scheduling. Then we transform the offline scheduling algorithm into an online scheduling algorithm.



(a) Abilene



(b) CERNET



(c) GEANT

Fig. 7. Average CCT under various lengths of coflows.

6.1. Offline scheduling

Because of non-negligible signaling overhead and performance loss of preemptive scheduling (Yu et al., 2016), we adopt to non-preemptive coflows scheduling in our approach. We propose the scheduling algorithm PROS (Placement and Routing Optimization of coflowS, Algorithm 3) to minimize the average CCT.

In our algorithm, we schedule the coflow with the shortest completion time first. We first calculate the CCT with Algorithm 1 (line 1–2) by optimizing the reduce tasks placement and routing. Then we sort the coflows in increasing order of CCT and obtain the sorted coflows vector F^* (line 3). Finally, we schedule the coflows according to the order of coflows in the sorted vector F^* to minimize the average CCT (line 4–6).

6.2. Online scheduling

Because coflows arrive online, we consider to transform the offline scheduling to online scheduling. In non-preemptive scheduling, the bandwidth and routes for coflows stay unchanged when coflows are being transferred. The coflow once starts being transferred cannot be preempted until finishes. We adopt the well-known minimum remaining time first (MRTF) policy. To avoid starvation, we schedule the coflow which waits for more than a threshold Γ when a current coflow finishes. The online scheduling algorithm is shown in Algorithm 4.

In Algorithm 4, we wake up our scheduling algorithm when a new coflow arrives or a current coflow finishes (line 1). If a new coflow arrives, we compute the minimal CCT through algorithm PRO and insert the coflow into the unscheduled coflow set I (line 2–4). If the current coflow finishes, we check if there is a coflow with waiting time expiring a threshold Γ (line 8). If exists, we schedule this coflow immediately (line 9–11), otherwise, we schedule the coflow with the minimum CCT in unscheduled coflow set I (line 12–13).

7. Evaluation

In this section, we conduct simulation and testbed experiments to evaluate the performance of our proposed algorithms. Our coflow data is randomly generated. The extensive experiments demonstrate that CCT can be better reduced with the joint optimization of tasks placement and routing, compared with placement-optimization-only or routing-optimization-only algorithms. In addition, average CCT can also be reduced with our proposed offline and online scheduling algorithms.

Algorithm 3 PROS.

Input: E, F, D
Output: y_{ij}, x_{kj}^t
foreach $i \in r$ **do**
  Obtain CCT C_{f_i} of f_i by Algorithm PRO ;
Sort the coflows in increasing order of C_{f_i} and obtain F^*;
foreach $f_i \in F*$ **do**
  Obtain x_{kj}^t and y_{ij} of f_i by Algorithm PRO ;
  schedule coflow f_i ;

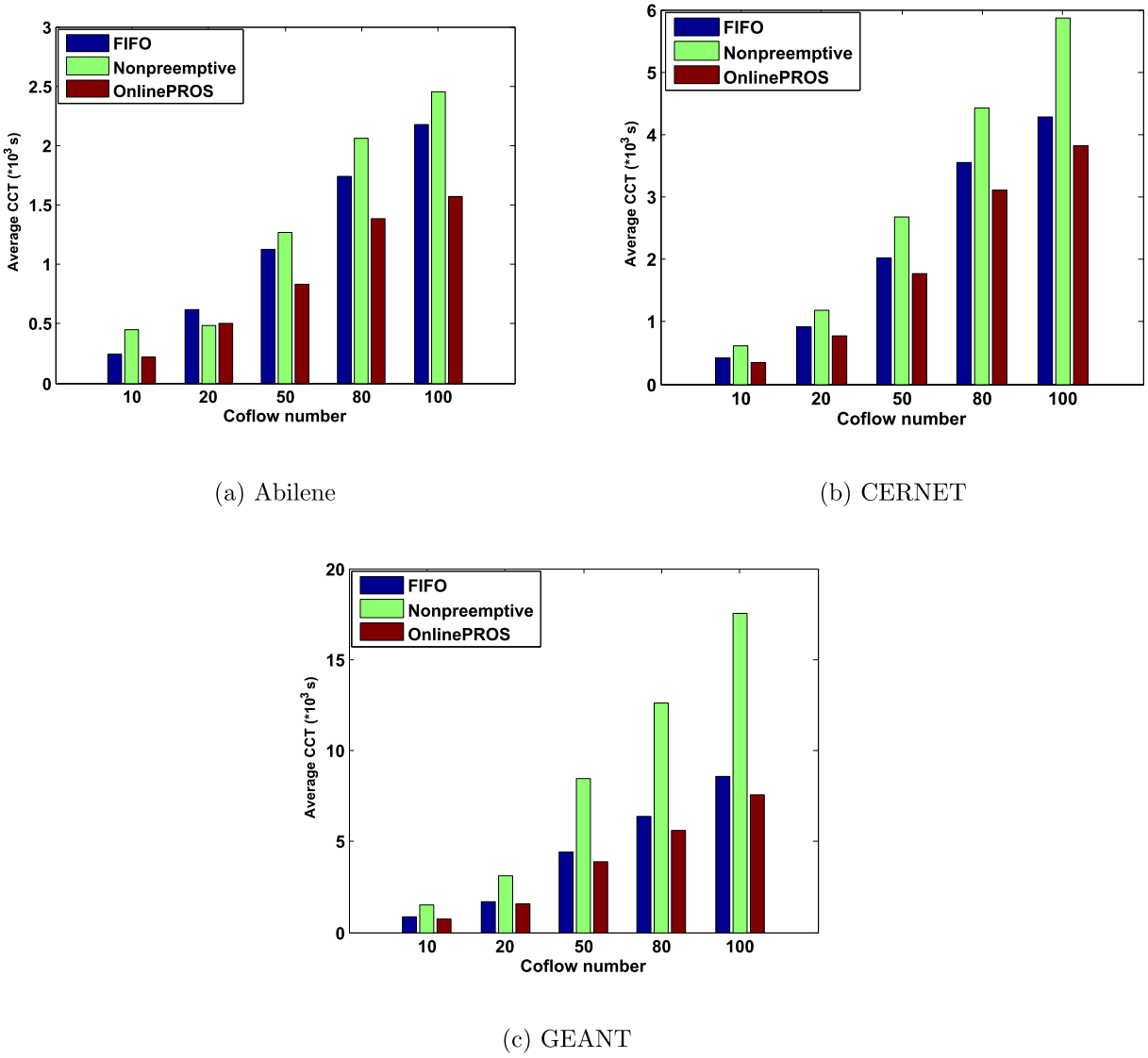


Fig. 8. Online scheduling of multiple coflows. Coflows is evenly distributed and coflow interval is set to 10s.

7.1. Evaluation setting

In the simulation, we develop a flow-level simulator written in C++. We exploit real network topologies: America research and education network (Abilene), China education and research network (Cernet), and Europe research and education network (Geant) in our experiments. The detailed information of the topologies is shown in Table 2. We employ an LP solver CPLEX (CPLEX, 2009) to solve LP problems in our experiments. The intermediate data sizes are randomly generated. The bandwidths of all the inter-DC WAN links are set to 100 Mbps (Hu et al., 2016). The experiments run on a personal computer with CPU i5-3320 M 2.6 GHz and 4 GB of RAM. We compare our algorithm with the following three algorithms:

Routing only (Rapier): The reduce tasks are located at random datacenters and the routing is computed by the algorithm Rapier in (Zhao et al., 2015). In this method, we only optimize coflow routing.

Tasks placement only (Iridium) The reduce tasks are selected by task placement algorithm in Iridium (Pu et al., 2015), while the routing of the coflow is based on k-shortest paths. In this method, we only optimize tasks placement.

Smartcoflow: The coflow optimization algorithm proposed in (Li et al., 2018a) with only task placement optimization.

Table 6
Improvement ratio under different coflow number compared with FIFO.

Topology	10	20	50	80	100	Average
Abilene	15.9%	3.8%	13.9%	9.7%	16%	11.86%
CERNET	3.0%	7.8%	9.8%	14.1%	11.4%	9.22%
GEANT	8.9%	13.6%	7.6%	13.2%	12.9%	11.24%

7.2. Simulation

7.2.1. Single coflow

We evaluate CCT under various coflow lengths. The coflow length is the size of the largest flow in a coflow. We set the coflow length to 0.5 GB, 5 GB, 50 GB, 500 GB, respectively and draw the curves that CCT varies with coflow lengths in Fig. 4. We conduct 50 experiments to evaluate CCT under different algorithms. As shown in Fig. 4, we can observe that with the increasing of coflow length, CCT is increasing. The curves of PRO stay below Iridium, Rapier and SmartCoflow. It demonstrates that our proposed algorithm PRO can obtain a much lower CCT compared to Iridium, Rapier and SmartCoflow. To be more concrete about the improvement ratio, we compute the improvement ratio of our proposed PRO compared to other three algorithms in Tables 3–5. The improvement ratio of our algorithm PRO is computed as follows:

$$Impro(PRO) = \frac{CCT(*) - CCT(PRO)}{CCT(*)} \tag{10}$$

Here, we discuss the importance of routing and task placement on overall performance. As shown in Tables 3–5, for Abilene network, compared with Iridium, Rapier and SmartCoflow, the improvement ratio of CCT in PRO is 49.2%, 25.9% and 53.5% on average. For Cernet network, the Improvement ratio of CCT in PRO is 63.8%, 42.4% and 63.0% on average. The experiments on Abilene and Cernet network demonstrate that CCT can be better reduced in algorithm that only optimizes routing (Rapier) than algorithms that only optimize task placement (Iridium and SmartCoflow). For Geant network, compared with Iridium, Rapier and SmartCoflow, the Improvement ratio of CCT in PRO is 35.1%, 56.7% and 33.6% on average. The experiments on Geant network demonstrate that CCT can be better reduced in algorithms that only optimize Task placement (Iridium and SmartCoflow) than algorithm that only optimize routing (Rapier). The experimental results on different topologies demonstrate that the importance of routing and task placement on overall performance varies in different topologies. Therefore, it is necessary to jointly optimize routing and task placement to reduce CCT.

Then, we draw the CDF curves of CCT under different coflow lengths for Abilene. As shown in Fig. 5, the curves of PRO stay above Iridium, Rapier and SmartCoflow. We can observe that CCT can be reduced greatly compared to Iridium, Rapier and SmartCoflow on average in Abilene.

7.2.2. Multiple coflows

We evaluate average CCT under various number and lengths of multiple coflows in offline scheduling PROS. We employ different algorithms: PROS, Rapier, Iridium, and SmartCoflow to minimize average CCT.

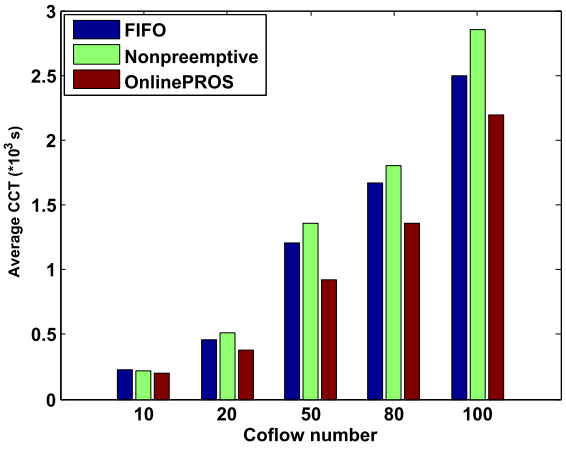
We draw curves that average CCT varies with number of coflows in Fig. 6. The coflow length is set to 5 GB. We can observe that with the increasing of coflow number, average CCT increases. The curves of PROS stay below Iridium, Rapier and SmartCoflow. It demonstrates that PROS can obtain a lower average CCT compared to Iridium, Rapier and SmartCoflow. Therefore, our algorithm PROS outperforms Iridium, Rapier, SmartCoflow in multiple coflows scheduling.

Table 7
Improvement ratio under different coflow number compared with Non-preemptive (Yu et al., 2016).

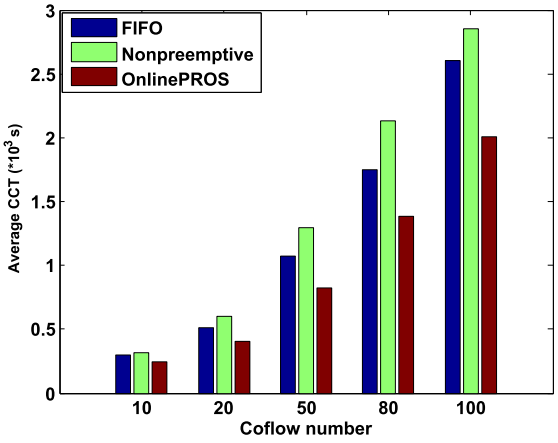
Topology	10	20	50	80	100	Average
Abilene	52.7%	20.1%	34.4%	33.0%	35.8%	35.2%
CERNET	45.5%	35.3%	34.7%	29.9%	35.2%	36.1%
GEANT	53.4%	50.5%	54.4%	55.4%	56.9%	54.1%

Algorithm 4 Online non-pre emptive scheduling.

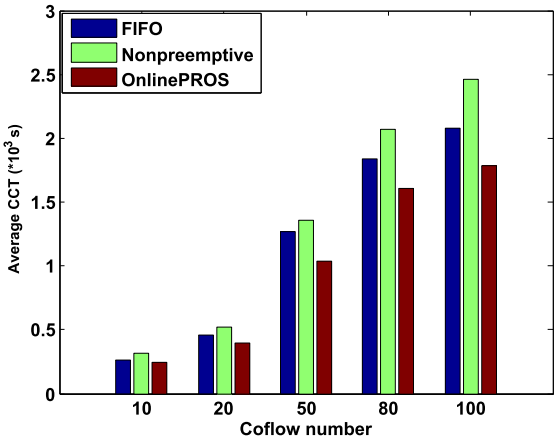
```
while a new coflow arrives or a coflow finishes do
    if a new coflow f_n arrives then
        Compute the CCT of f_n through PRO ;
        Insert f_n into unscheduled coflow set I ;
    if the current coflow finishes then
        wait_time_expire = false ;
        foreach f_u ∈ I do
            if waiting time of f_u > Γ then
                schedule f_u ;
                wait_time_expire = true ;
                break;
        if wait_time_expire == false then
            schedule the coflow with the minimum CCT in set I ;
```



(a) Interval=1s



(b) Interval=5s



(c) Interval=20s

Fig. 9. Online scheduling of multiple coflows in Abilene with different coflow arrival intervals.

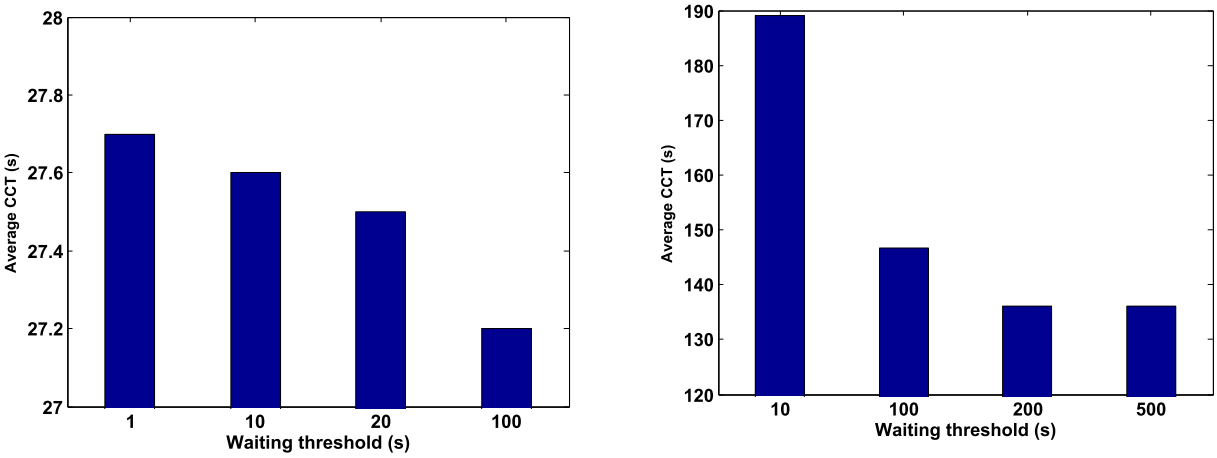
Then in Fig. 7, we show curves for the evaluation of average CCT on various coflow lengths. The number of coflows is set to 50. We can observe that with the increasing of coflow length, average CCT increases. PROS can obtain a lower average CCT compared to Iridium, Rapier and SmartCoflow.

From the previous simulation and analysis, we conclude that by joint optimization of routing and tasks placement, our proposed algorithm outperforms the routing-only and tasks-placement-only algorithms in offline scheduling algorithms.

Moreover, we compare our proposed online scheduling of multiple coflows (OnlinePROS) with FIFO and non-preemptive scheduling in (Yu et al., 2016). In FIFO algorithm, coflows are scheduled according to their arriving time. The coflows arriving first are scheduled first. While in our online scheduling algorithm, coflows with shorter CCT are scheduled first. Coflows arrive every 10 s and are distributed evenly. The coflow length is set to 5 GB. We plot the curves that average CCT varies with different number of coflows in Fig. 8. As shown in Fig. 8, we can observe that average CCT is increasing with the increasing of coflow numbers. Then in Table 6 and Table 7, we compute the improvement ratio of our online scheduling algorithm under different coflow number. We obtain that our algorithm reduces average CCT by 11.86%, 9.22%, 11.24% on average under different topologies compared to FIFO. Our algorithm reduces average CCT by 35.2%, 36.1%, 54.1% on average under different topologies compared to algorithm in (Yu et al., 2016). In summary, our non-preemptive online scheduling of multiple coflows performs better than FIFO and algorithm in (Yu et al., 2016) in minimizing average CCT.

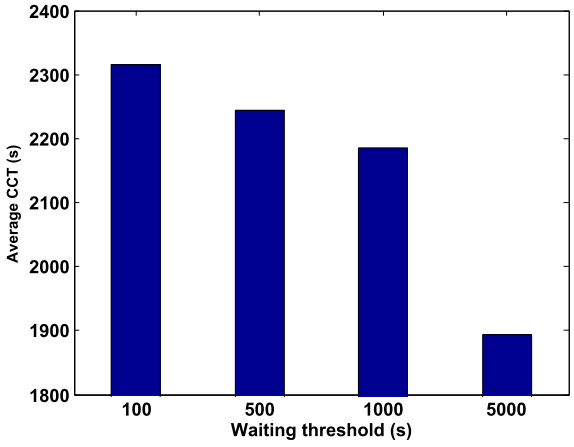
In Fig. 9, we draw the bar graphs that the average CCT varies with different coflow arrival intervals. The coflow arrival intervals are set to 1s, 5s and 20s, respectively. Our online scheduling algorithm performs better than FIFO and Nonpreemptive with different coflow arrival intervals. When the interval becomes larger, the gap between our algorithm with FIFO becomes smaller. For if the interval is very large, there will leave no coflows in the unscheduled list in our algorithm and each coflow will be scheduled upon its arrival, which approximates the performance of FIFO.

To analyze the impact of waiting time threshold on CCT, we plot the curves that average CCT varies with different waiting thresholds in Fig. 10. As shown in Fig. 10, we observe that average CCT decreases with the increasing of waiting threshold under different coflow lengths.



(a) coflow length=0.5GB

(b) coflow length=5GB



(c) coflow length=50GB

Fig. 10. Average CCT under different waiting time thresholds with different coflow lengths in Abilene.

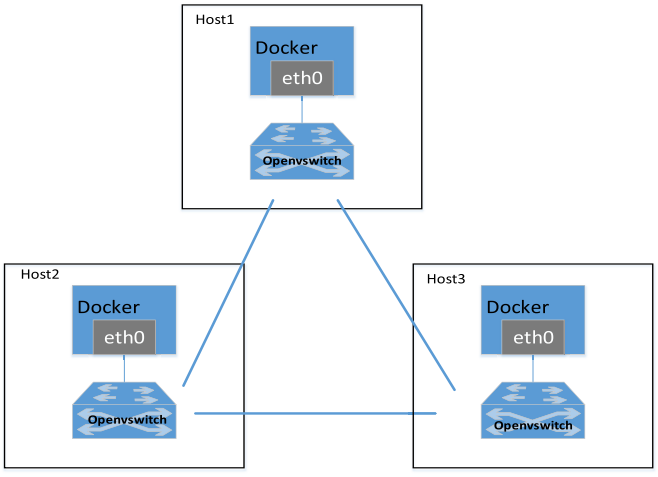


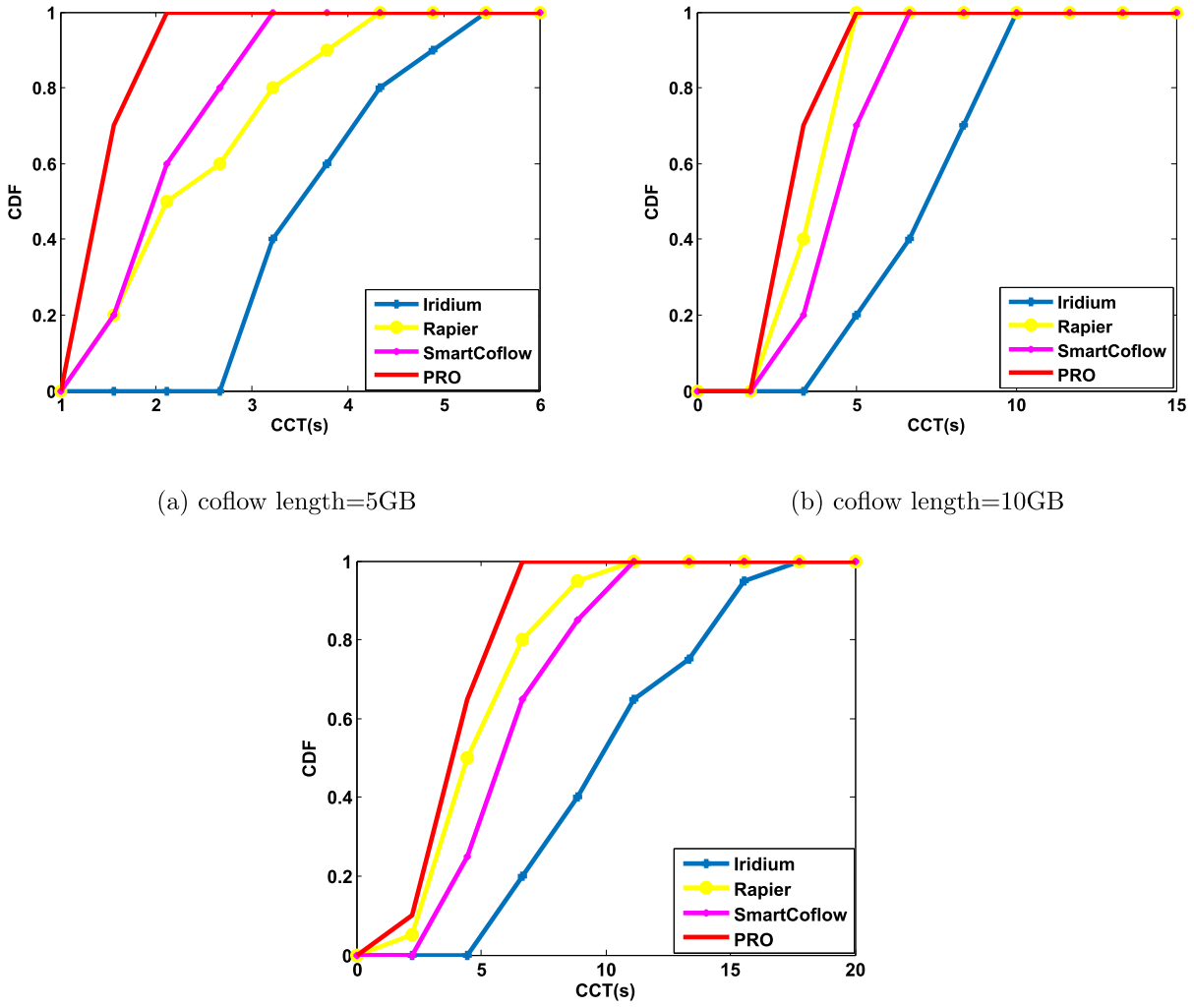
Fig. 11. A small-scale testbed topology.

This is because larger value of waiting time threshold means that more coflows with small CCT can be scheduled with high priority.

7.3. Testbed experiments

Finally, we evaluate the performance of our algorithm on a small-scale testbed. The testbed is composed of three openvswitches, which are connected by VXLAN tunnels (see Fig. 11). Each docker container is attached to an open switch and flows are generated from docker containers. We record the CCT under different coflow lengths and draw the CDF curves in Fig. 12. As shown in Fig. 12, the curve of PRO is above other algorithms. The results demonstrate that our algorithm outperforms other algorithms in minimizing CCT.

PRO can be easily implemented as a scheduling module in big data analytic applications, such as Hadoop or Spark. We can replace the default scheduling algorithm with our proposed algorithm PRO to optimize coflows. PRO will pick the optimal datacenters for reduce task placement, and determine the routing for coflows by configuring programmable switches in data plane.



(a) coflow length=5GB

(b) coflow length=10GB

(c) coflow length=20GB

Fig. 12. CDF curves of CCT under different coflow lengths.

8. Conclusion

In this paper, we are the first to reduce CCT by jointly optimizing the tasks placement and coflow routing. We formulate the joint optimization problem as a MINLP problem and propose an approximate algorithm PRO with the theoretical guarantee to solve the problem. Compared with the algorithms that only optimize the tasks placement or only optimize coflow routing, PRO can better reduce CCT. Moreover, we propose offline and online non-preemptive scheduling algorithms to optimize multiple coflows. The experiments demonstrate that both offline and online scheduling algorithms can reduce average CCT compared with FIFO. In the future research, we consider priorities of different coflows and use weighted CCT as optimization objective.

Acknowledgment

This work is partially supported by the National Key Research and Development Program of China [grant number No.2017YFB0803004].

References

Agarwal, S., Rajakrishnan, S., Narayan, A., Agarwal, R., Shmoys, D., Vahdat, A., 2018. Sincronia: near-optimal network design for coflows. In: Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication. ACM, pp. 16–29.

Chen, L., Cui, W., Li, B., Li, B., 2016. Optimizing coflow completion times with utility max-min fairness. In: Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on. IEEE, pp. 1–9.

Chowdhury, M., Stoica, I., 2015. Efficient coflow scheduling without prior knowledge. Comput. Commun. Rev. 45 (5), 393–406.

Chowdhury, M., Zaharia, M., Ma, J., Jordan, M.I., Stoica, I., 2011. Managing data transfers in computer clusters with orchestra. Comput. Commun. Rev. 41 (4), 98–109.

Chowdhury, M., Zhong, Y., Stoica, I., 2014. Efficient coflow scheduling with varys. In: ACM Conference on SIGCOMM, pp. 443–454.

Cormen, T.T., Leiserson, C.E., Rivest, R.L., 2009. Introduction to algorithms. Resonance 1 (9), 14–24.

CPLEX, I.I., 2009. V12. 1: user's manual for cplex. Int. Bus. Machines Corp. 46 (53), 157.

Dogar, F.R., Karagiannis, T., Ballani, H., Rowstron, A., 2014. Decentralized task-aware scheduling for data center networks. Comput. Commun. Rev. 44 (4), 431–442.

Gu, L., Zeng, D., Li, P., Guo, S., 2014. Cost minimization for big data processing in geo-distributed data centers. IEEE Trans. Emerg. Topics. Comput. 2 (3), 314–323.

Guo, Y., Wang, Z., Yin, X., Shi, X., Wu, J., 2017. Joint optimization of task placement and routing in minimizing inter-dc coflow completion time. In: Computer Communication and Networks (ICCCN), 2017 26th International Conference on. IEEE, pp. 1–2.
Hu, Z., Li, B., Luo, J., 2016. Flutter: scheduling tasks closer to data across geo-distributed datacenters. In: Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on. IEEE, pp. 1–9.
Isard, M., Prabhakaran, V., Currey, J., Wieder, U., Talwar, K., Goldberg, A., 2009. Quincy: fair scheduling for distributed computing clusters. In: Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles. ACM, pp. 261–276.
Ji, S., Li, B., 2016. Wide area analytics for geographically distributed datacenters. Tsinghua Sci. Technol. 21 (2), 125–135.
Laoutaris, N., Sirivianos, M., Yang, X., Rodriguez, P., 2011. Inter-datacenter bulk transfers with netstitcher. Comput. Commun. Rev. 41 (4), 74–85.
Li, Y., Jiang, H.C., Tan, H., Zhang, C., Chen, G., Zhou, J., Lau, F.C.M., 2016. Efficient online coflow routing and scheduling. In: ACM International Symposium on Mobile Ad Hoc NETWORKING and Computing, pp. 161–170.
Li, P., Guo, S., Miyazaki, T., Liao, X., Jin, H., Zomaya, A., Wang, K., 2017. Traffic-aware geo-distributed big data analytics with predictable job completion time. IEEE Trans. Parallel Distrib. Syst.
Li, W., Yuan, X., Li, K., Qi, H., Zhou, X., 2018. Leveraging endpoint flexibility when scheduling coflows across geo-distributed datacenters. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications. IEEE, pp. 873–881.
Li, W., Guo, D., Liu, A.X., Li, K., Qi, H., Guo, S., Munir, A., Tao, X., 2018. Coman: managing bandwidth across computing frameworks in multiplexed datacenters. IEEE Trans. Parallel Distrib. Syst. (5), 1013–1029.
Prisacari, B., Rodriguez, G., Heidelberger, P., Chen, D., Minkenberg, C., Hoefler, T., 2014. Efficient task placement and routing of nearest neighbor exchanges in dragonfly networks. In: Proceedings of the 23rd International Symposium on High-Performance Parallel and Distributed Computing. ACM, pp. 129–140.
Pu, Q., Ananthanarayanan, G., Bodik, P., Kandula, S., Akella, A., Bahl, P., Stoica, I., 2015. Low latency geo-distributed data analytics. Comput. Commun. Rev. 45 (5), 421–434.
Qiu, Z., Stein, C., Zhong, Y., 2015. Minimizing the total weighted completion time of coflows in datacenter networks. In: Proceedings of the 27th ACM Symposium on Parallelism in Algorithms and Architectures. ACM, pp. 294–303.
Wang, S., Zhang, J., Huang, T., Pan, T., Liu, J., Liu, Y., 2018. Multi-attributes-based coflow scheduling without prior knowledge. IEEE/ACM Trans. Netw. 26 (4), 1962–1975.
Yu, R., Xue, G., Zhang, X., Tang, J., 2016. Non-preemptive coflow scheduling and routing. In: Global Communications Conference (GLOBECOM). IEEE, pp. 1–6. IEEE, 2016.
Zaharia, M., Borthakur, D., Sen Sarma, J., Elmeleegy, K., Shenker, S., Stoica, I., 2010. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In: Proceedings of the 5th European Conference on Computer Systems. ACM, pp. 265–278.
Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M.J., Shenker, S., Stoica, I., 2012. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. USENIX Association. 2–2.
Zeng, D., Zhang, J., Gu, L., Li, P., Yao, H., 2017. Minimize coflow completion time via joint optimization of flow scheduling and processor placement. In: GLOBECOM 2017-2017 IEEE Global Communications Conference. IEEE, pp. 1–5.
Zhang, H., Chen, K., Bai, W., Han, D., Tian, C., Wang, H., Guan, H., Zhang, M., 2016. Guaranteeing deadlines for inter-datacenter transfers. In: Proceedings of the Tenth European Conference on Computer Systems, pp. 1–14.
Zhang, H., Chen, L., Yi, B., Chen, K., Chowdhury, M., Geng, Y., 2016. Coda: toward automatically identifying and scheduling coflows in the dark. In: Proceedings of the 2016 Conference on ACM SIGCOMM 2016 Conference. ACM, pp. 160–173.
Zhao, Y., Chen, K., Bai, W., Yu, M., 2015. Rapier: integrating routing and scheduling for coflow-aware data center networks. In: IEEE Conference on Computer Communications, pp. 424–432.
Zhao, Y., Tian, C., Fan, J., Guan, T., Qiao, C., 2018. Rpc: joint online reducer placement and coflow bandwidth scheduling for clusters. In: 2018 IEEE 26th International Conference on Network Protocols (ICNP). IEEE, pp. 187–197.

Yingya Guo received the B.S degree in computer science from Xiamen University, China, in 2013. She is currently a PhD candidate at the Department of Computer Science and Technology, Tsinghua University. Her research interests include traffic engineering, routing optimization and Software Defined Networking.

Zhiliang Wang received the B.E., M.E. and Ph.D. degrees in computer science from Tsinghua University, China in 2001, 2003 and 2006 respectively. Currently he is an Associate Professor in the Institute for Network Sciences and Cyberspace at Tsinghua University. His research interests include formal methods and protocol testing, next generation Internet, network measurement.

Han Zhang received the B.S. degree in Computer Science and Technology from JiLin University and Ph.D. in Tsinghua University. He is now working in the School of Cyber Science and Technology, Beihang University. His research concerns computer networks, network security and AI. He is a member of IEEE.

Xia Yin received the B.E., M.E. and Ph.D. degrees in computer science from Tsinghua University in 1995, 1997 and 2000 respectively. She is a Full professor in Department of Computer Science and Technology at Tsinghua University. Her research interests include future Internet architecture, formal method, protocol testing and large-scale Internet routing.

Xingang Shi received the B.S. degree from Tsinghua University and the PhD degree from The Chinese University of Hong Kong. He is now working in the Institute for Network Sciences and Cyberspace at Tsinghua University. His research interests include network measurement and routing protocols.

Jianping Wu received his B.S., M.S., and Ph.D. from Tsinghua University. He is a Full professor and director of Network Research Center, Ph.D. Supervisor of Department of Computer Science, Tsinghua University. From 1994, he has been in charge of China Education and Research Network (CERNET). He is a member of Information Advisory Committee, Office of National Information Infrastructure, Secretariat of State Council of China, and is also a vice president of Internet Society of China (ISC). His research interests include next generation Internet, IPv6 deployment and technologies, Internet protocol design and engineering.