Integrating Coflow and Circuits Scheduling for Optical Networks

IEEE INFOCOM 2018, Paper #

***Abstract*—Optical Circuit Switches (OCS) are increasingly used in cluster networks due to their data rate, energy and longevity advantages over electrical packet switches. Concurrently, an emerging crucial requirement for modern data-parallel clusters is to achieve high application-level communication efficiency when servicing structured traffic flows (a.k.a. Coflows) from distributed data processing applications. Only completing all flows in a coflow is meaningful to an application. To optimize application performance, both traffic scheduling and switch scheduling must be jointly considered at the level of a coflow rather than individual flows. However, prior solutions have significant limitation: they only consider traffic scheduling in the traditional network, or consider the switch scheduling for only one switch, which are both insufficient. To this end, we present TSS, a coflow-aware network optimization framework that seamlessly integrates traffic and switch scheduling for better application performance. Using a large-scale simulations, we demonstrate that TSS significantly reduces the average coflow completion time (CCT) by up to 66.3% compared to the state-of-the-art scheduling-only solution.**

Index Terms*—*Coflow, Routing and scheduling, OCS.

I. Introduction

Optical circuit switches (OCS) are increasingly used in cluster networks because of the robustness, high link bandwidth and low energy consumption. In an optical circuit switch network, a connection is set up by establishing a lightpath from the source node to the destination node. A lightpath is an optical channel that may span multiple fiber links to provide a circuit-switched interconnection between two nodes which avoids the expensive optical-electrical-optical conversions and makes the high bandwidth available. Apart from these advantages, an OCS has two constraints. First, and input (output) port can set up at most one circuit to an output (input) port at a time. That is, no input (output) port may connect to multiple output (input) ports. Second, each circuit can be reconfigured with a fixed time delay δ, which means it takes time δ to set up a new circuit and during the period of δ, the communication stops on the the input (output) ports on the circuits to be set up or torn down. However, other circuits remain unchanged and keep working.

Cluster computing frameworks such as MapReduce [1], Dryad [2], Spark [3] and so on have become the mainstream platforms for data processing and analysis in todays cloud services. A common feature of these different computing paradigms is that they all implement a data flow computing model, in which a group of data flows need to pass through a sequence of intermediate processing stages before generating the final results. These intermediate flow transfers can account for more than 50% of job completion time [4], and have a significant impact on job performance. Therefore, optimizing such flow transfers is important for applications. The term coflow is defined as the set of all flows transferring data between two stages of a job [5]. To optimize application performance, we need to optimize flow transfers at the level of coflow rather than individual ones. This is because the job completion time depends on the time it takes to complete the entire coflow, instead of the time to complete individual flows composing it. For example, in MapReduce [1] and BSP [6], a stage cannot complete, or sometimes even start, before it receives all the flows in a coflow from the previous stage. From an applications perspective, when a stage is pending for the input data, the CPU often sits idle or is under-utilized. As a result, reducing the coflow completion time (CCT) can further improve CPU utilization, maximizing application performance and job throughput in a given time period.

Recently, some studies [7] [8] [9] have devoted on minimizing average CCT. These works can be mainly divided into two main categories. However, we demonstrate that both two categories be insufficient for scheduling optimization, especially for multi-hop OCS, which will reduce the user's QoS. One category is the Coflow scheduling through routing and scheduling in the traditional network [5]. First, since this work only focuses on the traditional (or non-optical) network, this method does not consider the switch scheduling for coflow scheduling. Second, due to the optical switching feature, the route path between two devices in the optical network is time aware. As a result, the result for the traditional network can not work well for the optical network. The second categories is called Sunflow [10], which studies the switch scheduling on one switch for minimizing CCT. First, this work does not consider the impact of flow routing and traffic scheduling on the CCT. Second, this work only considers the case with one switch, thus it is difficult to be extended to the general multi-hop optical network.

Therefore, *it is an urgent need to minimize CCT by joint optimization of traffic scheduling and switch scheduling in the optical networks*. The main contributions of this paper are:

1) We formulate the problem of how to determine which feasible lightpaths and when to forward flows and schedule the OCS network to minimize the coflow completion time (CCT). The NP-hardness is also analyzed.
2) To solve this problem, we separate this problem into two subproblems and prove their NP-hardness. For the first subproblem, we given an approximation algorithm with approximation ratio $\frac{4}{3} - \frac{1}{3k}$, where k is the number of

feasible lightpaths per ToR switch pair. For the second subproblem, we given an approximation algorithm with approximation ratio h, where h is h is the maximum hop count of all the lightpaths.

3) We evaluate the proposed algorithms with simulations. The simulation results show that our algorithms can reduce the CCT about 40-80% compared with the Baseline.

The rest of this paper is organized as follows. Section II formalizes the COCIS problem, and gives the NP-hardness proof. In Section III, we separate COCIS into two subproblems and propose corresponding approximation algorithms. The approximation ratios are also analyzed in this section. The simulation results are reported in Section IV. In Section V, we introduce the related works. We conclude the paper in Section VI.

II. Problem Formulation

A. Network Model

We start with the conceptional model used to study the scheduling algorithm. In an optical circuit switched (OCS) network, the switch ports are connected to Top-of-Rack (ToR) switches, and each ToR switch is connected to k machines. The link bandwidth is B. The source and destination nodes of a flow are ToR switches. Each ToR switch can serve as a source and destination simultaneously. We assume no buffering in the optical circuit switched netowrk, hence all the buffering occurs in the edge of the network, i.e. with the ToR switches. A flow can be forwarded only when the lightpath is set up between the source node and the destination node. In this article, the optical circuit switched (OCS) network and optical network are the same meaning.

With this network model, we have the following assumptions for the network.

- Each source-destination ToR switch pair has k feasible lightpaths. Since a ToR switch has k output ports which are connected to k machines, we assume that the k feasible lightpaths exist. What's more, if k feasible lightpaths are in the same port of a ToR, these lightpaths will be no big difference with a single lightpath and these lightpaths even need a extra delay during circuits reconfiguration. On this account, we assume that the these k lightpaths will pass through different ports of the ToR switch pair. For a ToR switch pair $(i.j)$, the k-feasible lightpath set is denoted as $P_{i,j}$. The feasible lightpath set for all ToR switch pairs is denoted by $P = \bigcup_{i,j} P_{i,j}$.
- Inspired by [4] [8], we assume that our network runs in a centralized manner, which accords with many recent centralized optical networks [11] [12]. In this paper, we assume that the information about a coflow is given and the central scheduler can control all the switches in the network.

B. Traffic Model

sec:trafficmodel In this part, we focus on the traffic model called *coflow*, which is defined as the set of all flows transferring between two stages of a job. Because the ToR switch is not just a host, there may be more than one flows from ToR switch i to ToR switch j. A flow set of a coflow can be denoted as C, where each element $\Gamma_{i,j} \in C$ indicates flow set $\Gamma_{i,j}$ that include all flows from ToR switch i to ToR switch j. Note that $|C|$ is the number of flows in a coflow C. Denote $f_{i,j}^m$ as the m^{th} flow of $\Gamma_{i,j}$ and $d_{i,j}^m$ as the amount of data of $f_{i,j}^m$. In this paper the flow is unsplittable.

Combined with the Section II-A, we have some remarks. In the network model, there is a feasible ligthpath set $P_{i,j}$ for a source-destination ToR switch pair (i, j). Likewise, there is a flow set $\Gamma_{i,j} = \{f_{i,j}^1, f_{i,j}^2, ..., f_{i,j}^m\}$ for the ToR switch pair (i, j). That is, for each flow set $f_{i,j}$, there corresponds a lightpath set $P_{i,j}$.

C. Routing and Scheduling Objective

For a coflow C, the objective is to minimize the Coflow Completion Time (CCT), which is the duration to finish all the flows in a coflow, so as to speed up application level performance. To achieve this goal, we propose two notions as follows.

- **Feasible lightpath set:** For a flow set $\Gamma_{i,j}$, there may be more than one lightpath from ToR switch i to j. combined with the assumption in II-A, the feasible lightpath set has k feasible paths. Denote $P_{i,j}$ as the feasible path set of flow set $\Gamma_{i,j}$.
- **Active period for a lightpath:** Because of the OCS network scheme, a lightpaths in our network model changes when a new circuit is set up or the former circuit is torn down. So a lightpath lasts for a period, called "active period". It's worth noting that an active period is an interval and the interval length is the **duration** for a lightpath.

With these notions, we illustrate our problem. Our objective is to minimize the coflow completion time (CCT). What we know are network model and coflow information. We need to determine which feasible lightpaths and when to forward flows. Besides, we need to schedule the OCS network to finish the transmission of a coflow. So our problem is called **Coflow and Circuits Scheduling problem (COCIS)**.

The COCIS problem is NP-hard. Furthermore, we will show the two subproblems, coflow scheduling (COS) and circuits scheduling (CIS), are NP-hard too. The COS problem is a special case that the circuits scheduling in the OCS network is fixed. Likewise, the CIS problem is the special case that the coflow routing and scheduling are fixed. After we prove that COS and CIS are NP-hard, COCIS is NP-hard too.

Theorem 1: The COS problem is NP-hard.

Theorem 2: The CIS problem is NP-hard.

The proofs of lemmas 1 and 2 have been relegated to the Appendixes A and B, respectively.

III. Algorithm and analysis

In this section, we focus on the algorithm details for the COCIS problems. COCIS is a joint optimization of coflow and OCS network to minimize the CCT. As we can see in II-C, not only COCIS but also two subproblems of COCIS are NP-hard. In view of the complexity and difficulty of the COCIS problem, we separate our framework into two steps, which will be introduced in III-A.

The rest of the algorithm section is organized as follows. Section III-A shows the overall framework of our algorithm. In section III-B, we introduce the first step our framework, which aims at coflow routing and scheduling. Section III-C introduces the second step of our framework. Finally, section III-E discusses some other things about our algorithms.

A. Framework

In this part, we give the overall description of our framework. We separate our framework into two steps. First, we determine the route from the k feasible lightpath set and the order distribution of each flow in the same lightpath. Second, given the route and duration of each lightpath, we schedule the circuits in the OCS network. Our framework is shown in Alg. 1.

Algorithm 1 Framework

1: **Step 1:Coflow Routing and Order Distribution** /* determine a lightpath for each flow in coflow C and the order for flows in the same lightpath */
2: **Step 2: Circuits Scheduling for Each Lightpath** /* assign each lightpath an active time whose length is the duration in step 1 */

The first step is coflow routing and order distribution. It aims at scheduling the coflow. Each flow has k-feasible lightpaths and coflow routing will determine which lightpath the flow will pass through. That is, our coflow scheduling includes routing and order distribution. If multiple flows pass through the same lightpath, the forwarding order is another problem. It's what the order distribution does. After appointing all the flows in C, we will compute the duration of each lightpath, which is the input of step 2. The second step in the framework get the duration of each lightpath. It will schedule the circuits of the whole network to minimize the CCT and output the scheduling scheme. Each lightpath will have an active period whose interval length is its duration. Because step 2 appoints each lightpath the exact duration. So the order in the same lightpath can be changed. That is, we only need to know the duration of each ligthpath. However, in the practical implementation of step 1, we do have an order for each flow to get a better approximation performance.

B. Greedy Routing and Order Distribution (GROD)& Analysis

We introduce step 1 of our framework. Given the Coflow C and the feasible lightpath set for a source-destination ToR pair, we assign each ligthpath a number of flows with an greedy algorithm. There might be an exponential number of feasible lightpaths between a source ToR switch and a destination ToR switch. Following [13] [14], we assume that the central scheduler has pre-computed k feasible lightpaths between each pair of ToR switches corresponding to the k output port of a ToR switch. Note that $P = \bigcup_{i,j} P_{i,j}$ is the feasible lightpath set of the network. For a coflow, the algorithm will be executed iteratively until all the source-destination ToR pairs are traversed. It's worth noting that the routes of feasible lightpaths are ensured and the circuits scheduling that makes lightpaths active is the responsibility of step 2. There is a flow set for each ToR pair, for example, $\Gamma_{i,j}$. Without loss of generality, we use ToR pair (i, j) and flow set $\Gamma_{i,j}$ as an example. For flow set $\Gamma_{i,j} = \{f^1_{i,j}, f^2_{i,j}, ...f^m_{i,j}\}$ and feasible lightpath set $P_{i,j} = \{p^1_{i,j}, p^2_{i,j}, ..., p^k_{i,j}\}$, the first step sorts all flows in $\Gamma_{i,j}$ non-increasingly according to their amount of data. In the second step, for each flow $f \in \Gamma_{i,j}$, appoint f to the lightpath with the least duration. duration of a lightpath means the total processing time to forward the flows. Finally, the lightpath that is appointed a new flow will update the duration. The GROD algorithm outputs the required duration for each lightpath. The duration matrix is denoted by $T = \{t_{i,j,l}\}$.

Algorithm 2 GROD: Greedy Routing and Duration Assignment

Input: C, $P_{i,j}, \forall i, j$
2: **Output: Duration Matrix** T
for each $\Gamma_{i,j} \in C$ **do**
4: Initiate duration array $T_{i,j} = 0$ for each element.
Sort all flows in $\Gamma_{i,j}$ non-increasingly according to their amount of data
6: **for** each $f \in \Gamma_{i,j}$ **do**
Assign the flow to the lightpath with the least duration in $P_{i,j}$
8: Update the duration of the lightpath

Approximation Bound Analysis for GROD. We analyze the approximation performance of the proposed GROD algorithm. Because GROD is executed iteratively, the approximation bound will be the same for each source-destination ToR switch pair. Without loss of generality, we use a source-destination ToR swtich pair (i, j) as an example. Note that $GROD(I)$ is the maximum duration for k lightpaths in $P_{i,j}$ with GROD and $OPT(I)$ is the optimal result when the input flow set is I. The approximation ratio is R_{GROD}.

Theorem 3: The approximation ratio $R_{GROD} \leq \frac{4}{3} - \frac{1}{3k}$.

Proof: If $k = 1$, because $GROD(I) = OPT(I)$ and $\frac{4}{3} - \frac{1}{k} = 1 = \frac{GROD(I)}{OPT(I)}$.

For $k > 1$, we assume that $I = \{f_1, f_2, ..., f_m\}$ is the flow set with the least flow number that contradicts to the approximation ratio. The amount of data for the flows is $\{d_1, d_2, ...d_m\}$, and the duration of flows are $\{t_1, ..., t_m\}$. Thus the scheduling order is $\{f_1, f_2, ..., f_m\}$ and the maximum duration of each lightpath is $GROD(I)$. We set the last finished flow is f_{m_1} and $m_1 = n$. Because if

$m_1 < n$, there exists another flow set $I' = \{f_1, f_2, ..., f_{m_1}\}$ whose maximum duration is also $GROD(I)$. That is $GROD(I') = GROD(I)$. However, it's optimal result $OPT(I') \leq OPT(I)$. So we have:

$$\frac{GROD(I')}{OPT(I')} \geq \frac{GROD(I)}{OPT(I)} > \frac{4}{3} - \frac{1}{3k} \tag{1}$$

The above equation contradicts to our assumption that I is the flow set with the least flow number that contradicts to the approximation ratio, so $m_1 = m$. Now, we prove that if I exists, $m \leq 2k$.

Because f_m is the last finished flow, f_m starts at $GROD(I) - t_m$. At that time, this lightpath had the least duration. So we have:

$$\begin{aligned} GROD(I) - t_m &\leq \frac{1}{k}\sum_{q=1}^{m-1} t_q \\ GROD(I) &\leq \frac{1}{k}\sum_{q=1}^{m} t_q + \frac{k-1}{k} t_m \end{aligned} \tag{2}$$

Combined with $OPT(I) \geq \sum_{q=1}^{m} t_q$, we have:

$$\begin{aligned} &\frac{4}{3} - \frac{1}{3k} \leq \frac{GROD(I)}{OPT(I)} \leq \frac{OPT(I) + \frac{k-1}{k} t_m}{OPT(I)} \\ &\Rightarrow 4k - 1 < 3k + 3(k-1)\frac{t_m}{OPT(I)} \\ &\Rightarrow \ OPT(I) < 3t_m \end{aligned} \tag{3}$$

Because t_m is the least number among $\{t_1, ..., t_m\}$, $m < 2k$. Furthermore, we proof GROD is the optimal solution when $m < 2k$. We set $m = 2k$, since if $m \neq 2k$, we set $t_q = 0, \forall q = m+1, .., 2k$. We set assume lightpath has two flows: f_x and f_y. If $x, y \leq k$, there exists a lightpath whose two flows are $f_s, f_t, s, t > k$. Because $t_x, t_y \geq t_s, t_t$, we swap t_y and t_t. Then $t_x + t_t \leq t_x + t_y$ and $t_s + t_y \leq t_x + t_y$. The scheduling scheme O' after the swap is no big than the optimal solution, so O' is an optimal solution too. For scheduling scheme O', $f_1, ...f_k$ are appointed to the k different feasible lightpaths, so do $f_{k+1}, ...f_{2k}$. The scheduling scheme is the same as the GROD algorithm. So we have:

$$\frac{GROD(I)}{OPT(I)} = 1 \leq \frac{4}{3} - \frac{1}{3k} (k \geq 2) \tag{4}$$

In conclusion, on one hand, when $m = 1$, GROD is the optimal solution, so the approximation ratio is 1. On the other hand, when $m \geq 2$, if there exists a counterexample I, m must be no more than $2k$. Furthermore, we show GROD is the optimal solution if $m \leq 2k$. We have proven that the counterexample I doesn't exist, so lemma 3 is proved. ■

C. Circuits Scheduling & Analysis

After the first step of the framework, we get the output: Duration matrix $T = \{t_{i,j,l}\}$. $t_{i,j,l}$ represents the duration of l^{th} lightpath in $P_{i,j}$. In this section, we schedule the circuits in the OCS network to minimize the coflow completion time. Because the circuits scheduling will change the status of ports and lightpaths dynamically, we keep a table holding information about each lightpath and port of when they are free or busy, denoted by NST (Network Status Table). Once a port is released, we will search the lightpath set to find whether there exists a lightpath $p_{i,j,l}$ along which the ports are all free or not. If so, we will take up the ports and forward the flows until all the flows appointed to the lightpath are forwarded. What's more, lightpath $p_{i,j,l}$ will be excluded from the lightpath set and the NST will change the status of the lightpath $p_{i,j,l}$ and corresponding ports. Their status will be busy for the duration $t_{i,j,l}$. If not, we will wait until the next port is released. Our algorithm terminates when all the lightpath is excluded from lightpath set. Obviously, our loop can end because if there is lightpath remain in the lightpath set, it can at least wait until all the ports are free. Our circuits scheduling algorithm is given in Alg. 3.

B	bandwidth of the link
$P\{p_{i,j,l}\}$	feasible lightpath set for all the ToR pairs
NST	Network Status Table
$Start(.)$	the start time of a lightpath
$End(.)$	the end time of a lightpath
$T = \{t_{i,j,l}\}$	duration matrix for the lightpath set P
δ	the reconfiguration time
T_L	lower bound for the CCT
T_O	CCT of optimal solution
Δt_i	idle interval between p_{i-1} and p_i
h	the maximum hop count of lightpaths

TABLE I: Some Important Notations for Section III-C

Approximation Bound Analysis for CIS. Next, we analyze the circuits scheduling performance. In order not to conflict with the whole paper, **new notions in this section only suitable for this section**.

Before the analysis of approximation bound, we discuss the lower bound for our problem. For a port j, there are lightpath $p_1, p_2, ..., p_N$ that pass through the port. Note that the transmission time is $t_1, ..., t_1$. So the minimum time to finish the task is:

$$T_L^j = \sum_{i=1}^{N} t_i \tag{5}$$

Note that the lower bound is T_L. we have:

$$T_L = \max\{T_L^1, T_L^2, ...,\} \tag{6}$$

Theorem 4: Algorithm CIS completes the transmission within h times of the optimal solution, where h is the maximum hop count of all the lightpaths.

Proof: Because a link connects to two ports, one input port and one output port, we use input port j as an example. sort the lightpaths that pass through input port j by the time they are scheduled to start transmission, denoted by $p_1, ..., p_N$. t_i denotes the time to transmit (including the reconfigure time delay δ), and Δt_i represents the idle interval between p_{i-1} and p_i.

During Δt_i, port j is idle. According to our algorithm, it means links of lightpath $f_i, f_{i+1}, ..., f_N$ are all busy during Δt_i and at least one port of each lightpath is transmitting data for other lightpaths. Consider the last lightpath p_N, it

Algorithm 3 CIS: Circuits Scheduling Algorithm

t = Begin time
while $P \neq \varnothing$ **do**
  for $p_{i,j,l}$ in P **do**
    $temp = \delta + t_{i,j,l}$
    Search NST for all ports along the lightpath $P_{i,j,l}$
    if all ports along $P_{i,j,l}$ is free at time t **then**
      Arrange lightpath $p_{i,j,l}$ to start at t :
      $Start(p_{i,j,l}) = t$
      $End(p_{i,j,l}) = t + temp$
      Set lightpath $p_{i,j,l}$ is busy during t to $t + temp$ in NST
      for each ports p along the lightpath $p_{i,j,l}$ **do**
        Set p is busy during t to $t + temp$ in NST
      $P = P - p_{i,j,l}$
  t =the next nearest release time in NST

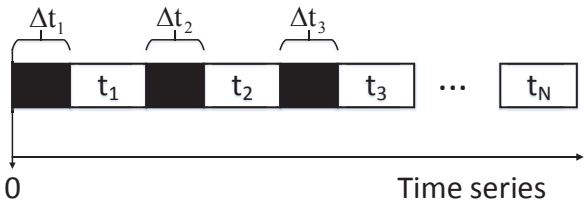

Fig. 1: Timeline of a port

has waited for $\sum_{i=1}^{N} \Delta t_i$ totally. During the waiting time, the worst case is that all input ports, except port j, have finish their task. Combined with the lower bound, we know that a port finishes its task within T_L. Note that lightpath p_N has h_N input ports. We have:

$$\sum_{i=1}^{N} \Delta t_i \leq (h_N - 1)T_L \leq (h-1)T_L \tag{7}$$

The transmission time of port j is $\sum i = 1^N t_i$. According to Eq. 5, we know:

$$\sum_{i=1}^{N} t_i \leq T_L \tag{8}$$

So the completion time for port j is:

$$\sum_{i=1}^{N} t_i + \sum_{i=1}^{N} \Delta t_i \leq (h-1)T_L + T_L = hT_L \tag{9}$$

The completion time of the optimal solution is denoted by T_O. Because T_L is lower bound to transmit the task, we have:

$$T_L < T_O \tag{10}$$

The port j is arbitrary, so algorithm CIS completes the transmission within h times of the optimal solution. Lemma 4 is proved.

■

D. Time Conplexity

In this section, we analyze the time complexity of GROD and CIS.

Theorem 5: GROD's time complexity is $O(|\Gamma| \log m)$, where m is the number of flows for each ToR switch pair.

Proof: GROD is executed for each ToR switch pair. For each pair (i, j), there is a flow set $f_{i,j}$ and $|f_{i,j}| = m$. We sort the flow set with time complexity $O(m \log m)$, then appoint the flow with time complexity $O(m)$. So the total time complexity is $O(|\Gamma|(\log m + 1)) = O(|\Gamma| \log m)$. ■

Theorem 6: CIS's time complexity is $O(h|P|^2)$. Where h is the maximum hop count of lightpaths.

Proof: The first loop is for all lightpaths in P. In the first loop, we first check the ports along the lightpath with time complexity $O(|P|h)$. After that we will go into the secong loop for all the lightpaths in P with time complexity $O(|P|^2)$. In the second loop, we will check the ports along the lightpath with time complexity $O(h|P|^2)$. So the time complexity is $O(h|P|^2)$. ■

E. Discussion

The previous GROD and CIS algorithms are for intra-coflow scheduling. In this section, we will discuss inter-coflow scheduling. When multiple coflows compete for network resource, an important goal is to accommodate the network resource management policies for these coflows.

However, under different usage scenarios, we may need to use different management policies. For example, the policy may require that coflow requested by privileged users are preferred over that of regular users. Another example is that the policy may require the later-staged coflows yield to earlier-staged coflows to avoid the potential creation of stragglers that unnecessary prolong job runtime [15]. So the management is flexible and dependent on different constraints and objectives.

On the other hand, our GROD algorithm has already computed the required duration of each lightpath, and the CIS assign the lightpath the exact required duration. if we add a new task from another coflow, it may change the NST and furthermore increase the CCT of the original coflow. An stable way is to run each intra-coflow in order like [10], which is one of the only a few researches that discussed the inter-coflow scheduling.

IV. Evaluation

In this section, we will present the results from the large-scale simulation.

A. Settings

Our simulation adopt the two dimensional HyperX topology [16], which has 81 switches, each has attached to 16 other switches. Because our network is an optical network, the switches are optical circuit switches. There are multiple ToR switches each of which is connected to k optical circuits switches. The link bandwidth $B = 10Gbps$, which is the common link bandwidth in optical networks [17] [18]. The

average flow size is $100Mbps$, which is almost equal to that of [5] in DCNs. In our simulation, we have two distribution for flows: 1)2-8 distribution, where twenty percent of flows are elephant flows and other eighty percent flows are mice flows. 2)Normal distribution, the average size of flows is $100Mbps$ [19] [20]. Our objective is to minimize the CCT. We will evaluate the impacts to CCT in four dimensions.

- **k, the number of feasible lightpaths for each ToR switch pair:** In our paper, we assume that each ToR switch is connected to k optical circuit switches and each ToR switch pair has k feasible lightpaths. In our simulation, we set $k = 3, 4, 5, 6$ and our default k is 4.In the premise of not causing confusion, we omit the "for each ToR witch pair" and say that k **is the feasible lightpath number or the number of feasible lightpaths.**
- **The number of ToR switches:** the ToR switches are access nodes of optical networks. If the ToR switches are sparse, the work load of the network will be light. If the ToR swtiches are dense, the work load of the lightpath will be heavy. In our simulation, we set the ToR switch number is 9-54. When the ToR switch number is 9 and feasible lightpath number k is 3, only one in three optical circuit switches is connected to a ToR switch. When the ToR switch number is 54 and feasible lightpath number k is 6, every optical switch is connected to 4 ToR switches. So our range of ToR switch number can cover different kinds of networks. Our default ToR switch number is 27.
- **Reconfiguration delay δ:** In optical networks, each circuit can be reconfigured with a fixed time delay δ, In our simulation we set δ=0.01ms,0.1ms,1ms,5ms and 10ms, where 10ms is the typical delay of a 3D-MEMS optical switch [10] and 0.01ms is the delay of the latest research [18]. In our simulation, we set the default time delay is 1ms, which is the accessible and common time delay for optical switches today.
- **Flow number per ToR switch pair:** For each ToR switch pair, there is a flow set. In our simulation, we set the flow number per ToR switch pair is 10,20,30,40. Our default number is 20.

B. Baseline for Comparison

Because our paper focus on the coflow scheduling and circuits scheduling and our framework solves the problem in two steps: GROD and CIS. Here we introduce the baseline for GROD and CIS respectively.

- **Baseline for GROD**: GROD schedules each flow from the k feasible lightpaths. A single way is to choose a fixed lightpath for each ToR switch pair. What's more, the less links the lightpath has, the less ports the lightpath will take up. So we choose the lightpath with the least hop count.
- **Baseline for CIS**: CIS is to schedule the multiple optical circuit switches. As shown in Section V, most previous works focused on the scheduling of a single switch. We adopt the algorithm of Solstice [21] whose performance is significantly better than TMS and Edmond. To let Solstice be suitable for the multi-hop scheduling. We use the improved Solstice as the baseline for CIS.

After introducing the baseline for each step, we use four algorithms for comparison in our simulation. For ease of reading. The tabs and introductions of four algorithms are as follows. Because the tabs will be shown in the figures, so we will name the tabs as short as possible. For example, we wil use "base" to represent the meaning of "baseline".

- **GROD+CIS**: This is our proposed algorithm. We don't give unnecessary details.
- **Base+CIS**: The first step is the baseline of GROD, and the second step is our CIS algorithm. We compare Base+CIS with GROD+CIS and Baseline.
- **GROD+base**: The first step is CIS, and the second step is the baseline of GROD. We compare Base+CIS with GROD+CIS and Baseline
- **Baseline**: Both of the two steps are baseline algorithms, so we call it the Baseline of GROD+CIS. if we name this algorithm like the above ones, it should be Base+Base. However, For ease of writing, we name it Baseline.

C. Performance Comparison

The simulation are performed under four dimensions. Here we illustrate the impacts of four dimensions.

Impact of feasible lightpath number k: To evaluate how the algorithms are influenced by the feasible lightpath number k. We fix the other parameters to be defaulted. The result are shown in Figure 2-5, where the horizontal axis is k, ranging from 3 to 6. In Fig. 2 and 3, the vertical axis is the coflow completion time (CCT) compared to Baseline. As a result, for all the three algorithms, the CCTs compared to Baseline go down as k increases. For example, in Fig. 3, the CCT of GROD+CIS compared to Baseline goes down from 0.55 to 0.28. That's because when k goes up, the load on each feasible lightpath is lower and the corresponding duration is smaller. What's more, when k is small, GROD+Base sometimes performs better than Base+CIS. For example, in Fig. 2, when $k = 3$, GROD+base will reduce 6% CCT compared to Base+CIS. However, when $k = 6$, GROD+base will increase 6% CCT compared to base+CIS. That's because our Baseline for CIS is more sensitive to k. Our proposed GROD+CIS algorithm can reduce 40-70% CCT compared to Baseline. Even for base+CIS and GROD+base, our GROD+CIS algorithm can reduce the CCT by 20%-40%, especially when $k = 6$.

In Fig. 4 and 5, the vertical axis is CCT(s). As the feasible lightpath number k increases, there are more lightpaths in the network. As a result, the CCTs of four algorithms decrease. For example, in Fig. 4, the CCT of baseline goes down from 7.8s to 6.2s when k increases from 3 to 6.

Impact of ToR switch number: To evaluate how the algorithms are influenced by the ToR switch number. We fix the other parameters to be defaulted. The results are shown in Figure 6-9, where the horizontal axis is the number

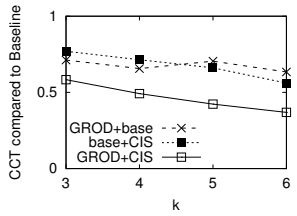

Fig. 2: CCT compared to Baselinevs. feasible path number k with 2-8 distribution

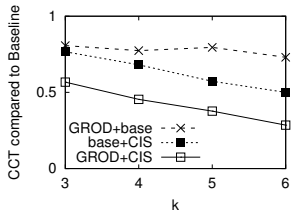

Fig. 3: CCT compared to Baseline vs. feasible path number k with normal distribution

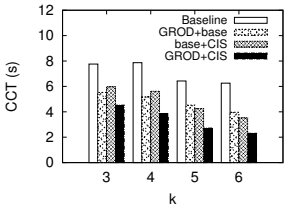

Fig. 4: CCT vs. feasible path number k with 2-8 distribution

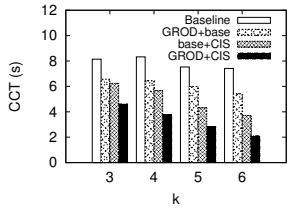

Fig. 5: CCT vs. feasible path number k with normal distribution

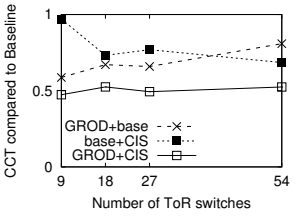

Fig. 6: CCT compared to Baseline vs. Number of ToR switches with 2-8 distribution

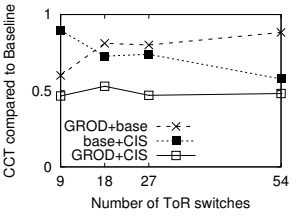

Fig. 7: CCT compared to Baseline vs. Number of ToR switches with normal distribution

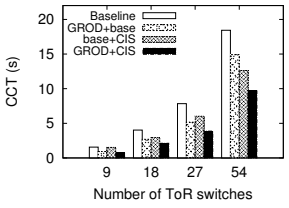

Fig. 8: CCT vs. Number of ToR switches with 2-8 distribution

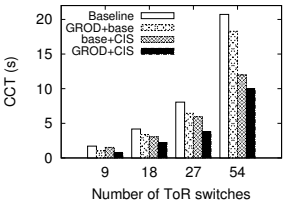

Fig. 9: CCT vs. Number of ToR switches with normal distribution

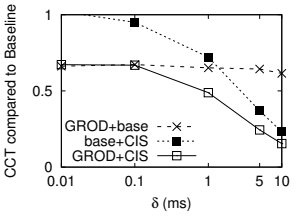

Fig. 10: CCT compared to Baseline vs. configuration delay δ with 2-8 distribution

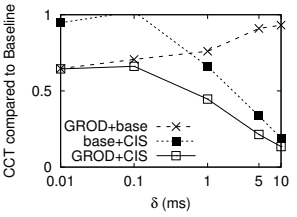

Fig. 11: CCT compared to Baseline vs. configuration delay δ with normal distribution

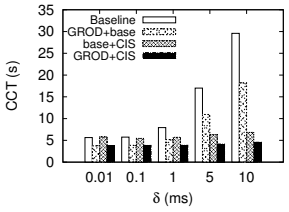

Fig. 12: CCT vs. configuration delay δ with 2-8 distribution

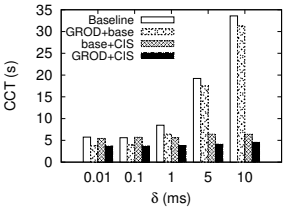

Fig. 13: CCT vs. configuration delay δ with normal distribution

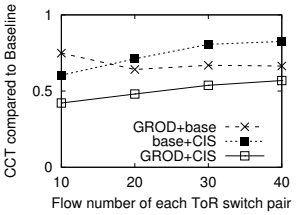

Fig. 14: CCT compared to Baseline vs. flow number per ToR pair with 2-8 distribution

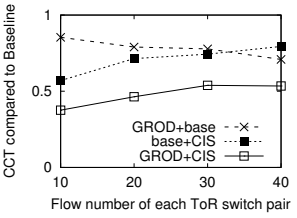

Fig. 15: CCT compared to Baseline vs. flow number per ToR pair with normal distribution

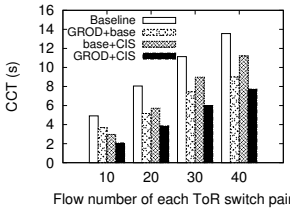

Fig. 16: CCT vs. flow number of each ToR switch pair with 2-8 distribution

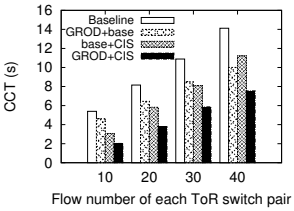

Fig. 17: CCT vs. flow number of each ToR switch pair with normal distribution

of ToR switches in the network, ranging from 9 to 54. In Fig. 6 and 7, the vertical axis is the coflow completion time (CCT) compared to Baseline. As a result, the CCT compared to Baseline change little as the number of ToR switches increases for GROD+CIS. But for GROD+base (base+CIS), it goes up (down) as the number of ToR switches increases. What's more, when ToR switch number is small, GROD+base performs better than base+CIS. For example, in Fig. 6, when there are 27 ToR switches, GROD+base will reduce 10% CCT compared to base+CIS. However, when there are 54 ToR switches, GROD+base will increase 10% CCT compared to base+CIS. Our proposed GROD+CIS algorithm can reduce 50% CCT compared to Baseline. Even for base+CIS and GROD+base, our GROD+CIS algorithm can reduce the CCT by 15%-30%. That's because our proposed algorithm helps to reduce the CCT by combining coflow scheduling and circuits scheduling.

In Fig. 8 and 9, the vertical axis is CCT(s). As the number of ToR switches increases, there are more flows in the network. As a result, the CCTs of four algorithms increase dramatically. For Baseline, the CCT go up from 2s to 18s when ToR switch number increases from 9 to 54.

Impact of configuration delay: To evaluate how the algorithms are influenced by the configuration delay δ. We fix the other parameters to be defaulted. The result are shown in Figure 10-13, where the horizontal axis is δ, ranging from 0.01ms to 10ms. In Fig. 10 and 11, the vertical axis is the coflow completion time (CCT) compared to Baseline. As a result, for GROD+CIS and Bseline+CIS, the CCTs compared to Baseline go down as δ increases. But for GROD+base, it goes up. For example, in Fig. 11, the CCT of GROD+CIS compared to Baseline goes down from 0.65 to 0.17. That's

because when delta goes up, the reconfiguration times plays an more important role and our proposed CIS algorithm will not change the reconfiguration times, while the baseline of CIS does. For example, in Fig. 11, when $\delta = 1$, GROD+base and GROD+CIS will reduce approximately 80% CCT compared to base+CIS and GROD+base.

In Fig. 12 and 13, the vertical axis is CCT(s). As δ increases, GROD+base and Baseline dramatically increase the CCT, while base+CIS and GROD+CIS keep stable and much smaller CCT. It shows that our CIS can reduce the reconfiguration times.

Impact of flow number per ToR switch pair: To evaluate how the algorithms are influenced by the flow number per ToR switch pair. We fix the other parameters to be defaulted. The results are shown in Figure 14-17, where the horizontal axis is the flow number per ToR switch pair, ranging from 10 to 40. In Fig. 14 and 15, the vertical axis is the coflow completion time (CCT) compared to Baseline. As a result, for base+CIS and GROD+CIS, the CCTs compared to Baseline go up as the flow number per ToR switch pair increases. However, for GROD+base, it goes down, especially for normal distribution. For example, in Fig. 15, the CCT of GROD+CIS compared to Baseline goes up from 0.39 to 0.55. But for GROD+base, it goes down from 0.85 to 0.72. That's because when the flow number per ToR switch pair increases, the δ will be less significant compared to the flow traffic. Just like Fig. 13, when δ decreases, the CCT compared to Baseline of GROD+CIS and base+CIS increase a bit, but the that of GROD+base keep stable. So it's not because our CIS algorithm doesn't work but essentially because the δ is less significant compared to the flow traffic when the flow number increases. What's more, when the flow number per ToR switch pair is small, base+CIS sometimes performs better than GROD+base. For example, in Fig. 14, when the flow number per ToR switch pair is 10, base+CIS reduces 20% CCT compared to GROD+base. However, when the flow number per ToR switch pair is 40, base+CIS increases 20% CCT compared to GROD+base.

In Fig. 16 and 17, the vertical axis is CCT(s). As the flow number per ToR switch pair increases, there are more flows in the network. As a result, the CCTs of four algorithms increases. For Baseline in Fig. 17, the CCT goes up from 5.5s to 14s when the flow number per ToR switch pair increases from 10 to 40.

D. Running Time

In this part, we evaluate how the feasible lightpath number k and ToR switch number can influence the running time. The results are shown in Fig. 18 and 19, where the vertical axis is running time. In Fig. 18, as k increases, the running time increases too. But compared to the Fig. 19, the running time increases dramatically when the number of ToR switches increases. That's because the lightpath number is $O(n^2)$ for the number of ToR switches while k can only change the number of lightpaths linearly.

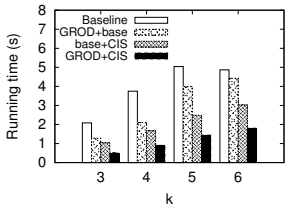

Fig. 18: Running time vs. feasible lightpath number k with 2-8 distribution

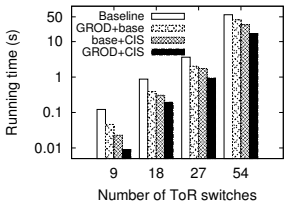

Fig. 19: Running time vs. ToR switch number with 2-8 distribution

V. Related works

A coflow is a flow set transferring data between two stages [7]. For a better application performance, we need to reduce the coflow completion time (CCT). Many works have been done in the classical networks. However, in optical circuit switched networks, how to reduce the CCT remains an open problem. There are two important ways to reduce the CCT. One is coflow scheduling, the other is scheduling the circuits in the OCS network. In the part, we will introduce the corresponding works respectively.

Many works have studied the coflow scheduling in the data center networks. The author of [7] summarized the traffic patterns and explicitly proposed the concept of coflow. After that, researches like [8] and [9] started to apply the coflow concept in their optimizations. All of the above works focused on when to forward the flows while neglecting the routing part. Only the author of [5] added the routing problem for coflows in the DCNs. However, the classical networks are packet switched networks. The OCS networks are circuits switched networks [10]. The coflow scheduling for packet switched networks are not suitable for circuits switched networks because optical switches have port constraints and the lightpath is not always available. To the best of our knowledge, no research has studied the coflow scheduling (including routing part) in the OCS network ever before. Most of the researches about optimizing the CCT in the OCS network focus on the circuits scheduling.

Another important method is circuits scheduling. Because the optical circuit switches have port constraint, a natural topic is scheduling the circuits to minimize the CCT of the coflow. The researches like [22] and [18] scheduled the circuits with the classic BvN matrix decomposition algorithm [23]. To use BvN, they would pre-process the demand matrixto meet the inpt assumption of BvN and decompose the pre-processed matrix into assignments and weights for each assignment. The author of [21] added dummy demand to the demand matrix in it's pre-process step. The above researches are based on the *all-stop* model, which assumes communication stops on all optical circuits during reconfiguration. It's unnecessary because circuits unchanged are not impacted and keep serving traffic [10]. Besides, the above researches all focused on the scheduling in a single switch not the whole optical network. The author of [24] focused on the scheduling for optical networks based on the *all-stop* model. Sunflow [10] studied the circuits scheduling based on the *not-all-stop*

model, but it's scheduling is a single optical circuit switch scheduling not the circuits scheduling of a OCS network. To the best of our knowledge, no research has studied the circuits scheduling of the whole OCS network.

Coflow scheduling and circuits scheduling both influence the CCT. The coordination of coflow scheduling and circuits scheduling is complicated but necessary. Our paper will integrate these two ways to minimize the CCT.

VI. Conclusion

In this paper, we formulate the coflow and circuits scheduling (COCIS)problem and prove it's NP-hardness. What's more, we prove it's two subproblems, coflow scheduling (COS) and circuits scheduling (CIS), are NP-hard too. To solve these two subproblems, we propose the corresponding approximation algorithms with analyzed approximation ratio. Later in our simulation, we show that our proposed GROD+CIS algorithm can reduce the CCT by 40%-80%.

References

[1] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[2] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," in *ACM SIGOPS operating systems review*, vol. 41, no. 3. ACM, 2007, pp. 59–72.

[3] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets." *HotCloud*, vol. 10, no. 10-10, p. 95, 2010.

[4] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. ACM, 2011, pp. 98–109.

[5] Y. Zhao, K. Chen, W. Bai, M. Yu, C. Tian, Y. Geng, Y. Zhang, D. Li, and S. Wang, "Rapier: Integrating routing and scheduling for coflow-aware data center networks," in *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, 2015, pp. 424–432.

[6] T. Cheatham, A. Fahmy, D. Stefanescu, and L. Valiant, "Bulk synchronous parallel computingła paradigm for transportable software," in *Tools and Environments for Parallel and Distributed Systems*. Springer, 1996, pp. 61–76.

[7] M. Chowdhury and I. Stoica, "Coflow: A networking abstraction for cluster applications," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*. ACM, 2012, pp. 31–36.

[8] M. Chowdhury, Y. Zhong, and I. Stoica, "Efficient coflow scheduling with varys," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 443–454.

[9] F. R. Dogar, T. Karagiannis, H. Ballani, and A. Rowstron, "Decentralized task-aware scheduling for data center networks," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 431–442.

[10] X. S. Huang, X. S. Sun, and T. E. Ng, "Sunflow: Efficient optical circuit scheduling for coflows." in *CoNEXT*, 2016, pp. 297–311.

[11] D. Simeonidou, R. Nejabati, and M. Channegowda, "Software defined optical networks technology and infrastructure: Enabling software-defined optical network operations," in *Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC), 2013*. IEEE, 2013, pp. 1–3.

[12] R. Ramamurthy and B. Mukherjee, "Fixed-alternate routing and wavelength conversion in wavelength-routed optical networks," *IEEE/ACM Transactions on networking*, vol. 10, no. 3, pp. 351–367, 2002.

[13] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 15–26.

[14] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks." in *NSDI*, vol. 10, 2010, pp. 19–19.

[15] M. Chowdhury and I. Stoica, "Efficient coflow scheduling without prior knowledge," in *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4. ACM, 2015, pp. 393–406.

[16] J. H. Ahn, N. Binkert, A. Davis, M. McLaren, and R. S. Schreiber, "Hyperx: topology, routing, and packaging of efficient large-scale networks," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*. ACM, 2009, p. 41.

[17] B. C. Chatterjee, N. Sarma, and E. Oki, "Routing and spectrum allocation in elastic optical networks: A tutorial," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1776–1800, 2015.

[18] G. Porter, R. Strong, N. Farrington, A. Forencich, P. Chen-Sun, T. Rosing, Y. Fainman, G. Papen, and A. Vahdat, *Integrating microsecond circuit switching into the data center*. ACM, 2013, vol. 43, no. 4.

[19] I. Antoniou, V. V. Ivanov, V. V. Ivanov, and P. Zrelov, "On the log-normal distribution of network traffic," *Physica D: Nonlinear Phenomena*, vol. 167, no. 1, pp. 72–85, 2002.

[20] S. Dulman, T. Nieberg, J. Wu, and P. Havinga, "Trade-off between traffic overhead and reliability in multipath routing for wireless sensor networks," in *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, vol. 3. IEEE, 2003, pp. 1918–1922.

[21] H. Liu, M. K. Mukerjee, C. Li, N. Feltman, G. Papen, S. Savage, S. Seshan, G. M. Voelker, D. G. Andersen, M. Kaminsky *et al.*, "Scheduling techniques for hybrid circuit/packet networks," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*. ACM, 2015, p. 41.

[22] N. Farrington, G. Porter, Y. Fainman, G. Papen, and A. Vahdat, "Hunting mice with microsecond circuit switches," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*. ACM, 2012, pp. 115–120.

[23] G. Birkhoff, "Tres observaciones sobre el algebra lineal," *Univ. Nac. Tucumán Rev. Ser. A*, vol. 5, pp. 147–151, 1946.

[24] C.-H. Wang, T. Javidi, and G. Porter, "End-to-end scheduling for all-optical data centers," in *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, 2015, pp. 406–414.

[25] G. Mosheiov, "Multi-machine scheduling with linear deterioration," *INFOR: Information Systems and Operational Research*, vol. 36, no. 4, pp. 205–214, 1998.

[26] T. Gonzalez and S. Sahni, "Open shop scheduling to minimize finish time," *Journal of the ACM (JACM)*, vol. 23, no. 4, pp. 665–679, 1976.

Appendix A
Proof of Lemma 1

Proof: For the COS problem, the OCS network scheduling scheme is fixed. We consider a special case that the coflow only has the flow set $f_{i,j} = \{f_{i,j}^1, f_{i,j}^2, ...\}$ from source ToR switch i to destination ToR switch j. There is a lightpath set $P_{i,j}$ which includes multiple lightpaths $p \in P_{i,j}$ connecting ToR switch i and ToR switch j. Each lightpath has active periods. What's more, there is an interval T between two active periods. Let's consider a special case that interval T of each lightpath is 0 and each lightpath starts at the initial moment. Thus, each light path is active all the time, our objective is to minimize the completion time. Then our problem because the *multi machine scheduling* problem, which is known to be NP-hard [25]. The special case of our problem is NP-hard, so our problem is NP-hard too. ■

Appendix B
Proof of Lemma 2

Proof: For the CIS problem, the route and required duration of each lightpath are given. We need to schedule the optical circuit switches of the whole network. Let's consider a special case that the OCS network has only one optical

switch. Then the route of each lightpath will become the input-output pair. When the reconfiguration delay $\delta = \infty$, the preemption penalty $= \infty$. Thus the CIS problem reduces to the non-preemptive open-shop problem to minimize work span [26], where each job (input port i) requires processing time (duration of the lightpath) on a specific machine (output port j). The open-shop problem is non-preemptive, which means a machine (output port j) would always finish processing a job (lightpath from one input port i) before moving to another job. The problem is NP-hard [26]. The special case of the CIS problem is NP-hard, so CIS is NP-hard too. ■