Non-clairvoyant Scheduling of Coflows

Akhil Bhimaraju  
IIT Madras

Debanuj Nayak  
IIT Gandhinagar

Rahul Vaze  
TIFR, Mumbai

Abstract—The coflow scheduling problem is considered: given an input/output switch with each port having a fixed capacity, find a scheduling algorithm that minimizes the weighted sum of the coflow completion times respecting the port capacities, where each flow of a coflow has a demand per input/output port, and coflow completion time is the finishing time of the last flow of the coflow. The objective of this paper is to present theoretical guarantees on approximating the sum of coflow completion time in the non-clairvoyant setting, where on a coflow arrival, only the number of flows, and their input-output port is revealed, while the critical demand volumes for each flow on the respective input-output port is unknown. The main result of this paper is to show that the proposed BlindFlow algorithm is $8p$-approximate, where $p$ is the largest number of input-output port pairs that a coflow uses. This result holds even in the online case, where coflows arrive over time and the scheduler has to use only causal information. Simulations reveal that the experimental performance of BlindFlow is far better than the theoretical guarantee.

I. INTRODUCTION

Coflow scheduling is a recent popular networking abstraction introduced to capture application-level computation and communication patterns in data centers. For example, in distributed/parallel processing systems such as MapReduce [1] or Hadoop [2], Dryad [3], jobs/flows alternate between computation and communication stages, where a new stage cannot start until all the required sub-jobs/flows have been processed in the preceding stage. Therefore, the metric of performance is the delay seen by the last finishing job/flow in a stage unlike the conventional latency notion of per-job/flow delay.

To better abstract this idea, a coflow [4] is defined that consists of a group of flows, where the group is identified by the computation requirements. The completion time of a coflow is defined to be the completion time of the flow that finishes last in the group. The main ingredients of the basic scheduling problem are as follows. There is a switch with $m_i$ input and $m_o$ output ports, and each port can process jobs at a certain maximum capacity. Coflows arrive over time, where each coflow has a certain weight (measures relative importance) and each flow of a coflow corresponds to a certain demand volume that has to be processed over a particular input-output port pair. Among the currently outstanding coflows, the scheduler’s job is to assign processing rates for all the flows (on respective input-output ports), subject to ports’ capacity constraints, with the objective of minimizing the weighted sum of the coflow completion time.

The value of the time-stamp at which the coflow “completes” is defined as the completion time. If we subtract the coflow’s release time from the completion time, we get the amount of time the coflow spent in the system, and this is called the flow time. The problem of minimizing the weighted sum of flow times has been shown to be NP-hard to even approximate within constant factors even when demand volumes of all the flows are known [5]. Thus, similar to prior work on coflow scheduling, in this paper, we only consider the completion time problem, where coflows can be released over time (the online problem). Solving the completion time problem even in the online case has been considered quite extensively in literature [12], [13], [27] (and references therein).

The coflow scheduling problem (CSP) for minimizing completion time is NP-hard, since a special case of this problem, the concurrent open shop (COS) problem is NP-hard (see [6] for definition of COS and reduction of COS to CSP). Because of the NP-hardness, the best hope of solving the CSP is to find tight approximations. For the COS problem, the best known approximation ratio is 2 [7] that is also known to be tight [8].

Work in approximating the CSP began with intuitive heuristics such as Varys [6], Baraat [9], and Orchestra [10] that showed reasonable empirical performance, which was then followed by theoretical work that showed that a 5-approximation is possible [11], [12]. However, no tight lower bounds better than 2 are known yet. A randomized 2 approximation was proposed in [13]. A 12 approximation was also derived in [12] for the online case.

Prior theoretical work on CSP only considered a clairvoyant setting [11]–[13], where as soon as a coflow arrives, the demand volumes for each of its flows per input-output port are also revealed, which can be used to find the schedule. In general, this may not always be possible as argued in [14] for various cases, such as pipelining used between different stages of computation [3], [15], [16] or task failures/speculation [1], [3], [17]. Recent research has shown that prior knowledge of flow sizes is indeed not a plausible assumption in many cases, but it might be possible to estimate the volumes [18].

In this paper, we consider the more general non-clairvoyant setting for solving the CSP, where on a coflow arrival, only its weight (its relative importance), the number of flows, and their corresponding input-output ports are revealed, while the demand volumes for each flow on the respective input-output port is unknown. Any flow departs from the system as soon as its demand volume is satisfied. The departure is then notified to the algorithm.

Non-clairvoyant model for CSP has been considered in
[14], where a heuristic algorithm Aalo based on Discretized Coflow Aware Least Attained Service (D-CLAS) was used. This method was further refined in [19] using statistical models for flow sizes. The objective of this paper is to present theoretical guarantees on approximating the non-clairvoyant CSP. There is significant work on non-clairvoyant scheduling in the theoretical computer science literature, for example for makespan minimization [20], [21], average stretch [22], flowtime [23], [24], flowtime plus energy minimization with speed scaling [25], [26], or with precedence constraints [27], but to the best of our knowledge not on the CSP.

For the non-clairvoyant CSP, we propose an algorithm BlindFlow, which is also online (does not need information about future coflows). Let $p$ be the largest number of distinct input-output port pairs any coflow uses. The main result of this paper is to show that BlindFlow is $8p$ approximate, where the guarantee is with respect to clairvoyant offline optimal algorithm. This result holds even in the online case, i.e., when coflows arrive (arbitrarily) over time and the algorithm only has causal information. As a corollary of our result, we get that a modified BlindFlow algorithm is $4p$ approximate for the COS problem (that is a special case of CSP) in the non-clairvoyant setting, which to the best of our knowledge was not known.

Our proof technique involves expressing the clairvoyant fractional CSP as a linear program (LP) and considering its dual. Then via the primal-dual method, we couple the rates allocated by BlindFlow to the dual variables of the clairvoyant fractional LP and then invoke weak duality, which is rather a novel idea in the CSP literature, inspired by [27].

The approximation guarantees derived in this paper depend on the problem instance via the parameter $p$, and are not constant unlike the clairvoyant case [11], [12]. The reasons thereof are briefly commented on in Remark 4. The proof ideas are, however, novel, and the bounds are useful when the maximum number of ports each coflow uses, $p$, is small, or the number of total port pairs is small. Moreover, as the simulations show (both synthetic and real-world trace data based), the performance of BlindFlow is far superior than the $8p$ approximation guarantee. The simulation performance of the BlindFlow algorithm is similar to the heuristic algorithm Aalo, even though Aalo outperforms BlindFlow because of multiple specific additions in Aalo which are appealing but are difficult to analyze.

II. System Model

Consider a switch with $m_i$ input and $m_o$ output ports as shown in Fig. 1. Without loss of generality we will assume that $m_i = m_o = m$. Coflow $k$ is the pair $(C_k, R_k)$, where $C_k = [d_{ijk}]$ is an $m \times m$ matrix with non-negative entries and $R_k$ is a non-negative real number, that represents its release time, the time after which it can be processed. For the $(i,j)^{th}$ flow in coflow $k$, we need to transfer $d_{ijk}$ amount of data from the $i^{th}$ input port to the $j^{th}$ output port of an $m \times m$ switch.

All the ports are capacity constrained and the $i^{th}$ input port can process $c_{i}^{op}$ units of data per unit time while the $j^{th}$ output port can process $c_{j}^{op}$ units of data per unit time. There are $n$ coflows in the system $\{(C_k, R_k)\}_{k=1}^n$, and we want to schedule them in such a way to minimize the sum of weighted completion times, as defined using the optimization program OPT below.

$$\begin{align*}
\text{minimize} & \quad \sum_{k} w_k T_k \\
\text{subject to} & \quad \sum_{i,j,k}^{c} x_{ijkt} T_k \geq w_k T_k \\
& \quad \sum_{i,j,k}^{c} x_{ijkt} \leq c_{ij}^{op} \quad \forall i, j, k, t \\
& \quad \sum_{i,j,k}^{c} x_{ijkt} \leq c_{jk}^{op} \quad \forall i, j, k, t
\end{align*}$$

where, $\{w_k\}$’s are the weights of each coflow, $x_{ijkt}$ is the rate assigned to the $(i,j)^{th}$ flow of coflow $k$ at time $t$, and $T_k$ is the completion time of coflow $k$. Constraints (1) and (2) together ensure that all the demands of a coflow are completed by time $T_k$. Constraints (3) and (4) are capacity constraints on input port $i$ and output port $j$ for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, m$.

Let $\{x_{ijkt}^{opt}, T_k^{opt}\}$ be the optimal solution to OPT and let the value of OPT when we use these values be $J_{opt}$, i.e., $J_{opt} = \sum_{k} w_k T_k^{opt}$.

OPT cannot be solved except for small or trivial cases due to the presence of $T_k$ in the limits of summation in constraint (2). The problem of finding the optimal schedule for minimizing weighted completion time for coflows has been proven to be NP-hard [6]. In prior work, assuming full knowledge of $\{C_k\}$, algorithms with theoretical guarantees on their approximation ratios have been derived in [11], [12].

In this paper, as discussed in Sec. I, we consider the non-clairvoyant case, where only the indices of the non-zero entries of $\{C_k\}$ are revealed and not the exact values of $\{d_{ijk}\}$. In addition, we assume that as the soon as the flow $k$ is released at time $R_k$, its weight is also revealed. This corresponds to only knowing the presence or absence of a flow to be fulfilled on a particular input-output pair but not the precise demand requirement on it. If weights are also unknown, in the following, we can let all weights $w_k = 1$ for all flows $k$ without changing any of our results.
Additionally, we consider the online setting, where we have no prior knowledge about the existence of a coflow before its release time \( R_k \). Thus, at time \( t \), the information available is only about the set of flows that are yet to complete using the variables \( 1^t_{ijk} \) for \( k \in Q_t \), where \( Q_t \) is the set of coflows released by time \( t \), i.e., \( Q_t = \{ k \mid t \geq R_k \} \). \( 1^t_{ijk} \) is 1 if the \((i,j)\)th flow of coflow \( k \) is yet to finish and 0 otherwise.

Let \( n_{kt} = \sum_{i,j} 1^t_{ijk} \) be the number of unfinished flows of coflow \( k \) at time \( t \). Let \( 1^t_k \) be the indicator whether or not the entire coflow has finished, i.e., \( 1^t_k = 1 \) if at least one among \( \{1^t_{ijk} \mid (i,j) \in m \times m \} \) is 1 and \( 1^t_k = 0 \) otherwise. When \( 1^t_k = 0 \), let \( n_{kt} \) be any non-zero real number to make notation simpler (this would mean \( 1^t_k/n_{kt} \) is always defined).

Next, we define a problem instance parameter \( p \), which will be used to express our approximation guarantee for the non-clairvoyant CSP.

**Definition 1.** Let \( p \) be the maximum number of unfinished flows in any coflow at any time, i.e., \( p = \max_k \{ n_{kt} \} \), the maximum value among \( \{ n_{kt} \} \) at \( t = 0 \).

Remark 1.

Note that for a particular input-output port pair, we can have at most one flow per coflow (definition of coflow). This implies that \( p \) is at most the maximum number of distinct input/output port pairs used by any coflow, and hence \( p \leq m^2 \).

### III. BlindFlow Algorithm

We propose the following non-clairvoyant algorithm to approximate the problem OPT. We divide the capacity of a port among all the flows that require that particular port in proportion to the flow weights. This is a natural choice, since the demand volumes for each flow are unknown. More precisely, BlindFlow allocates the rate \( r_{ijk}(t) \) in (5) to the \( k \)th coflow on the \((i,j)\)th input-output port pair at time \( t \), as

\[
r_{ijk}(t) = \frac{w_k 1^t_{ijk}}{\sum_{l \in Q_t} \sum_u \frac{w_l}{c_{il}} 1^t_{ijl} + \sum_{l \in Q_t} \sum_v \frac{w_l}{c_{vl}} 1^t_{ivol}} \quad \text{(5)}
\]

For \( t < R_k \), \( r_{ijk}(t) \) is obviously 0. Letting the “weight” of a flow to mean the weight of the coflow it belongs to, an outstanding flow on input-output port pair \( i,j \) gets a rate proportional to the ratio of its weight and the sum of the weights of all other flows that need either the input port \( i \) or the output \( j \), normalized to the port capacities \( c_{il} \) and \( c_{vl} \). Note that the flows that need both the input port \( i \) and the output port \( j \) are counted twice in the denominator of (5).

**Remark 1.** Equation (5) might produce a schedule where the rates of some flows can be increased without violating the feasibility on any port. A better rate allocation is given by

\[
r_{ijk}(t) = \frac{\max \left( \sum_{l \in Q_t} \sum_u \frac{w_l}{c_{il}} 1^t_{ijl}, \sum_{l \in Q_t} \sum_v \frac{w_l}{c_{vl}} 1^t_{ivol} \right)}{w_k 1^t_{ijk}}, \text{ replacing the } + \text{ operator with max in the denominator. Any performance guarantee on (5) automatically holds for this rate allocation as well.}
\]

BlindFlow is a very simple algorithm, and is clearly non-clairvoyant (does not use demand information \( d_{ijk} \)) and online (does not use information about future coflow arrivals to schedule at the current time).

**An example** Consider a simple example where we have a \( 2 \times 2 \) switch with port capacities 1 on all the ports and 2 coflows in the system. At some time \( t \), let the indicator matrices that indicate the outstanding flows for these coflows be \( I_1 \) and \( I_2 \), where \( I_1 \) is the \( 2 \times 2 \) matrix \( [1^t_{ij}] \) and \( I_2 \) is defined similarly. For this example, assume that these indicator matrices are given by:

\[
I_1 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \text{ and } I_2 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.
\]

This example is the same as the one shown in Fig. 1. Let the weights for the coflows be \( w_1 = 1 \) and \( w_2 = 2 \). From (5), the rate for the \((1,1)\) flow of coflow 1 is

\[
r_{111} = \frac{1}{(1 \cdot 1 + 1 \cdot 1 + 2 \cdot 1) + (1 \cdot 1 + 2 \cdot 1 + 2 \cdot 1)} = \frac{1}{9}.
\]

Similarly we get from (5) the other rates as

\[
r_1 = \begin{bmatrix} 1/9 & 0 \\ 1/6 & 0 \end{bmatrix} \text{ and } r_2 = \begin{bmatrix} 2/9 & 2/9 \\ 0 & 2/7 \end{bmatrix}.
\]

Here, the \( i,j \) entry of \( r_1 \) is the rate allocated to the \((i,j)\) port-pair of coflow 1. \( r_2 \) is defined similarly.

The main result of this paper on the approximation ratio achieved by BlindFlow is as follows.

**Theorem 1.** The rate allocation (5) of the BlindFlow algorithm is feasible and is \( 8p \)-approximate. In particular, if \( J_{OPT} \) is the optimal weighted coflow completion time, then BlindFlow produces a schedule with a weighted coflow completion time that is no larger than \( 8p \times J_{OPT} \).

**Remark 2.** The approximation ratio guarantee is independent of the number of coflows, the volumes of coflows, and the capacities of the input-output ports, and is only a function of the parameter \( p \) (Definition 1). The parameter \( p \), is the number of flows with distinct input/output port requirements maximized over all co-flows. Theoretically, \( p \) can be as large as \( m^2 \), however, for large switches (where \( m^2 \) is very large), actual coflows typically have \( p \) much smaller than \( m^2 \). Thus, the guarantee is still meaningful. Moreover, note that the approximation guarantee is with respect to the clairvoyant offline optimal algorithm. In the clairvoyant case, the approximation ratio guarantee (of 5) is independent of the input [11], [12]. However, introducing additional complexity into the problem, such as non-clairvoyance (this paper) or dependence across coflows [28], seems to inevitably make the guarantee a function of \( m \).

**Proof Sketch:** We prove Theorem 1 using a series of claims in the subsequent sections, where the main steps are as follows. We first decrease the rates allocated by BlindFlow to a baseline rate. Since these rates are lower, any guarantees on the baseline rate algorithm apply to BlindFlow as well. Then we “speed-up” the switch by a factor of \( 4p \) while using the baseline
rates. So any guarantee on the faster switch will apply to the original problem with additional factor of \(4p\). Next, we write a fractional LP formulation, FLP, the value of whose optimal solution is smaller than the optimal value of the objective we are trying to minimize, \(J_{\text{opt}}\). This implies that any dual feasible solution to FLP will have a dual objective that is smaller than \(J_{\text{opt}}\). We then produce a dual feasible solution using the speed-up rates whose dual objective is equal to half the weighted coflow completion time obtained by running the faster switch. This gives us the \(8p\) guarantee after combining the \(4p\) penalty.

**Claim 1.** The rate allocation made by BlindFlow in (5) is always feasible.

**Proof:** Any rate allocation that satisfies \(\sum_{k \in Q_t} \sum_{j \in Q_t} r_{ijk}(t) \leq c_{ij}^p\) for all input ports \(i\) and \(\sum_{k \in Q_t} \sum_{j \in Q_t} r_{ijk}(t) \leq c_{ij}^p\) for all output ports \(j\) is a feasible schedule. For any input port \(i\), we have,

\[
\sum_{k \in Q_t} \sum_{j \in Q_t} r_{ijk}(t) = \sum_{k \in Q_t} \sum_{j \in Q_t} \sum_{\ell \in Q_t} \sum_{u \in Q_t} \frac{w_{jk}}{t} \cdot 1_{ijk} + \sum_{\ell \in Q_t} \sum_{v \in Q_t} \frac{w_{jk}}{t} \cdot 1_{ijk} \\
\leq \sum_{k \in Q_t} \sum_{j \in Q_t} \sum_{\ell \in Q_t} \sum_{v \in Q_t} \frac{w_{jk}}{t} = c_{ij}^p.
\]

Similar argument follows for each output port as well.

Let coflow \(k\) finish at time \(T_k\) when we use the schedule determined by (5). Let \(J_{\text{ALG}}\) be the weighted completion time produced by BlindFlow, i.e., \(J_{\text{ALG}} = \sum_{k} w_k T_k\).

**A baseline allocation** To analyze the rates allocated by BlindFlow in (5), we define the following “baseline” algorithm for scheduling the coflows:

\[
r_{ijk}^{\text{BASE}}(t) = \begin{cases} 
\frac{w_k 1_{ijk}}{t} & \text{for } t \geq 4p R_k \\
0 & \text{for } t < 4p R_k,
\end{cases}
\]

(6)

where \(R_k\) is the release time of the coflow \(k\).

Note that we do not require the rate allocation using the baseline algorithm (6) to be feasible or causal since we would not actually run this algorithm on a switch. We use the rates in (6) just as a means to upper bound the weighted completion time using (5), as we show in the subsequent discussion.

Compared to (5), in (6), we compute the rate using weights of all the unfinished flows, and not just the ones that have been released. In particular, the summation in the denominator here includes the flows that may be released in the future unlike (5). Moreover, we do not schedule any flow in coflow \(k\) until time \(4p R_k\), unlike BlindFlow, where we start scheduling as soon as it is released at \(t = R_k\). The rate allocation to a particular flow using BlindFlow might decrease over time if new coflows are released. This does not happen with the baseline rate allocation as we consider all the unfinished coflows in the denominator of the expression. However, the allocation in (6) gives us strictly smaller rates than what is allocated by BlindFlow because the denominator in BlindFlow can never be greater than the sum of all the current and future flows sharing the same input or output port. But as we see, the rates in (6) are sufficient to prove our performance guarantee. Let the weighted completion time obtained by using the rates allocated by (6) be \(J_{\text{BASE}}\).

**Claim 2.** \(J_{\text{ALG}} \leq J_{\text{BASE}}\).

**Proof:** At any time \(t\), given the same set of unfinished flows, the rates we get by using (5) are greater than or equal to the rates we get using (6) for every flow. By using induction from \(t = 0\), where the set of unfinished flows would be the same for both the algorithms, we can conclude the claim.

**IV. THE AUGMENTED SWITCH**

Using ideas from [27], we first prove an approximation guarantee after “speeding up” the switch by a certain factor. Later, we can relax this assumption at a cost to our guarantee equal to the speed-up factor. For a switch, the speed-up is in terms of adding additional capacity to its ports. We now describe this setup formally.

Consider a switch where input ports have a capacity of \(4p\) instead of \(c_{ij}^p\) (likewise for output ports). This means that now we can process up to \(4p c_{ij}^p\) units of demand over each port per time unit. Hypothetically, consider scheduling the coflows over this new faster switch using the following formula:

\[
r_{ijk}^{\text{AUG}}(t) = 4p \times \frac{w_k 1_{ijk}}{t} + \sum_{l \in \mathbb{Q}} \sum_{u \in \mathbb{Q}} \frac{w_{jk}}{t} \cdot 1_{ijkl}.
\]

(7)

for \(t \geq R_k\). Note that just like in (6), we add the weights from all the coflows in the denominator of (7) and not just the released coflows like (5). However, we start processing coflow \(k\) at rate (7) at time \(R_k\) unlike (6), where we wait till time \(4p R_k\).

Let the weighted completion time we obtain by running (7) be \(J_{\text{AUG}}\). If we stretch the time axis by \(4p\) and reduce \(r_{ijk}^{\text{AUG}}(t)\) by a factor of \(4p\), we get \(r_{ijk}^{\text{BASE}}\). But since we are stretching the time axis, the completion times we get by using (6) are \(4p\) times as big compared to the completion times produced by (7). This gives us the following claim.

**Claim 3.** \(J_{\text{BASE}} = 4p J_{\text{AUG}}\), where \(J_{\text{AUG}}\) is the weighted sum of completion times when we run the augmented rates (7).

**Remark 3.** For the duration in which \(1_{ijk} = 1\), i.e., till the time the demand on the \((i,j)\) port pair of coflow \(k\) is not fully satisfied, the corresponding rate from (7), \(r_{ijk}^{\text{AUG}}(t)\), is a non-decreasing function of \(t\). This is because the terms in the denominator of (7) can only decrease as time progresses.

From here on, \(1_{ijk} = 1\) would be the indicator of whether or not the demand \(d_{ijk}\) has been satisfied when we use the rate allocation from (7). In the following sections, we shall prove that using the rates from (7), we get the sum of weighted completion times to be no worse than twice what we get by using the optimal schedule for OPT. Since this is obtained by compressing the time axis by \(4p\), and the rate allocation in (7) does no worse than twice the optimal for OPT, this gives us the \(8p\) guarantee.
V. THE FRACTIONAL LP

Since the problem of minimizing weighted completion time (OPT) is NP-hard, we use a simpler problem that can be written as a linear program. This is the problem of minimizing the “fractional” completion time. For a single flow, the fractional completion time is calculated by dividing the job into small chunks and taking the average completion time of the different chunks. Intuitively, fractional completion time should be less than the actual completion time, since for the actual completion time we only consider the time when the last chunk finishes. We extend this concept to coflows and formally prove that fractional completion time is indeed less than the actual completion time. This gives us a lower bound on \( J_{OPT} \), and as we see subsequently, \( J_{AUG} \) is not far from this lower bound.

Consider the following linear program that represents the fractional CSP.

\[
\begin{align*}
\text{minimize } & \sum_k w_k \sum_{t \geq R_k} f_{kt} \\
\text{subject to } & \sum_{s=R_k}^t f_{ks} \leq \sum_{s=R_k}^t \frac{x_{ijks}}{d_{ijk}}, \forall i,j,k \text{ with } t \geq R_k, \quad (8) \\
& \sum_{t \geq R_k} f_{kt} \geq 1, \forall k, \quad (9) \\
& \sum_{k \in Q_t} x_{ijkt} \leq c_{ij}^{OP}, \forall j,t, \quad (10) \\
& \sum_{k \in Q_t} \sum_{i,j} x_{ijkt} \leq c_{ij}^{op}. \forall i,t. \quad (11)
\end{align*}
\]

Here, \( \sum_{t \geq R_k} t f_{kt} \) is the fractional completion time of coflow \( k \). The variables \( \{x_{ijkt}\} \) and \( \{f_{kt}\} \) are defined for \( t \geq R_k \). Additionally, \( x_{ijkt} \) is defined for \( d_{ijk} \neq 0 \), but for simplicity of presentation, we drop mentioning this everywhere. \( x_{ijkt} \) represents the rate allocated for the coflow \( k \) on port pair \((i,j)\) at time \( t \). The fraction of demand \( d_{ijk} \) that has completed by time \( t \) is given by \( \sum_{s=R_k}^t \left( \frac{x_{ijks}}{d_{ijk}} \right) \). We define \( f_{kt} \) so that \( \sum_{t=R_k}^\infty f_{kt} \) is equal to the minimum among these \( \sum_{s=R_k}^t \left( \frac{x_{ijks}}{d_{ijk}} \right) \) fractions over all the flows of coflow \( k \). Intuitively, \( f_{kt} \) is the “fraction” of coflow \( k \) that has finished during time slot \( t \), so that \( \sum_{t=R_k}^\infty f_{kt} \) is the fraction of coflow completed by time \( t \).

Constraint (8) defines \( \{f_{kt}\} \), and constraint (9) ensures that the demands of all the flows in a coflow have been completely satisfied eventually. Constraints (10) and (11) are similar to constraints (4) and (3) and ensure that capacity constraints are satisfied for all the ports.

The following is a formal proof of the above intuitive ideas that FLP is indeed a lower bound for OPT.

Claim 4. The optimal value of FLP is a lower bound on \( J_{OPT} \), the optimal value of OPT.

Proof: Consider the optimal schedule \( \{x_{ijkt}^{OPT}\} \) of OPT. Note that this need not be the optimal solution for FLP.

Since \( \{x_{ijkt}^{OPT}\} \) is a feasible schedule for OPT, the capacity constraints (11) and (10) are satisfied as (3) and (4) are satisfied. Define

\[
f^*_kt = \min_{i,j} \left\{ \sum_{s=R_k}^t \frac{x_{ijks}^{OPT}}{d_{ijk}} - \min_{i,j} \left\{ \sum_{s=R_k}^{t-1} \frac{x_{ijks}^{OPT}}{d_{ijk}} \right\} \right\}, \forall k, t \geq R_k.
\]

\[
\sum_{s=R_k}^t x_{ijks}^{OPT} \leq d_{ijk}, \forall i,j,k \text{ with } t \geq R_k. \quad (12)
\]

Since \( \sum_{s=R_k}^t \frac{x_{ijks}^{OPT}}{d_{ijk}} \) is a non-decreasing function of \( t \) \((\geq R_k)\) for any feasible schedule \( \{x_{ijkt}\} \), for any \((i,j)\), the min of all these functions over \((i,j)\) is also a non-decreasing function. This ensures that \( f^*_kt \geq 0 \) for all \( k,t \geq R_k \). From the definition of \( f^*_kt \), via a telescopic sum argument, we have,

\[
\sum_{s=R_k}^t f^*_kt = \min_{i,j} \left\{ \sum_{s=R_k}^t \frac{x_{ijks}^{OPT}}{d_{ijk}} \right\} \leq 1, \forall k. \quad (13)
\]

Equations (12) and (13) ensure the feasibility of the solution \( \{x_{ijkt}^{OPT}, f^*_kt\} \) for FLP. Now we show that FLP objective for OPT, from constraint (1) in OPT, we get

\[
\sum_{s=R_k}^t x_{ijks}^{OPT} \geq 0, \forall i,j,k. \quad (14)
\]

Therefore we get

\[
\sum_{k \geq R_k} t f^*_kt \leq \sum_{t=R_k}^T \sum_{k \geq R_k} T^k f^*_kt = T^k. \quad (15)
\]

As we have a feasible solution \( \{x_{ijkt}^{OPT}, f^*_kt\} \), where FLP has a value less than or equal to \( J_{OPT} \), the optimal solution to FLP will also have a value less than or equal to \( J_{OPT} \).

Next, we consider the dual program of FLP, and show that the optimal dual objective is greater than or equal to half the sum of weighted completion times, \( J_{AUG} \), obtained by using the rates allocated by (7).

VI. THE DUAL PROGRAM

Let the dual variables corresponding to constraints (8), (9), (10), and (11) be \( g_{ijkt}, \alpha_k, \phi_{jt}, \) and \( \theta_{it} \) respectively. The dual program for FLP is given by the following.

\[
\begin{align*}
\text{maximize } & \sum_k \alpha_k - \sum_{j,t} c_{ij}^{OP} \phi_{jt} - \sum_{i,t} c_{ij}^{op} \theta_{it} \\
\text{subject to } & \alpha_k \leq t w_k + \sum_{j,t} g_{ijkt} \quad (16) \\
& \sum_{s \geq t} g_{ijks} d_{ijk} \leq \phi_{jt} + \theta_{it} \quad (17)
\end{align*}
\]
We will define a new variable $\alpha_{kt}$ and use this to set $\alpha_k$. Recall that we are using the rates allocated by (7), and $1_{ijk}$ is 1 if the $(i,j)$ flow of coflow $k$ has not yet finished by time $t$ with rates (7) and 0 otherwise, and $n_{kt} = \sum_{i,j} 1_{ijk}$ is the total number of unfinished flows in coflow $k$ at time $t$ when using rates (7). Consequently, define the dual variables as follows.

$$\alpha_{kt} = w_k 1_{k}, \quad \alpha_k = \sum_{k} \alpha_{kt}, \quad \tag{16}$$

$$\theta_{it} = \frac{1}{4_c^OP} \sum_{j,k} \frac{w_k}{n_{kt}} t_{ijk}, \quad \tag{17}$$

$$\phi_{jt} = \frac{1}{4_c^OP} \sum_{i,k} \frac{w_k}{n_{kt}} t_{ijk}, \quad \tag{18}$$

$$\gamma_{ijkt} = \frac{1}{n_{kt}} t_{ijk}. \quad \tag{19}$$

Note that the choice of the dual variables (16), (17), (18), and (19), and hence the dual objective, depends on the rates (7) via $\{1_{ijk}\}$ and $\{n_{kt}\}$. Let the value of DLP, when we process coflows with rates (7) on the augmented switch and define the dual variables as above, be $J_{DUAL}$.

**Claim 5.** $J_{DUAL} = \frac{1}{4} J_{AUG}.$

**Proof:** First consider the first term in the dual objective:

$$\sum_k \alpha_k = \sum_{k,t} \alpha_{kt} = \sum_{t} \sum_k w_k 1_k = J_{AUG}. \quad \tag{20}$$

The second term in the dual objective is:

$$\sum_{j,t} c_{j,t}^OP \phi_{jt} = \frac{1}{4} \sum_{i,j,k,t} \frac{w_k}{n_{kt}} 1_{ijk},$$

$$= \frac{1}{4} \sum_{t} \sum_k \frac{w_k}{n_{kt}} \left( \frac{1}{\sum_{i,j} 1_{ijk}} \right),$$

$$= \frac{1}{4} \sum_{t} \sum_k \frac{w_k}{n_{kt}} 1_k = \frac{1}{4} J_{AUG}. \quad \tag{21}$$

Similarly, we can show that

$$\sum_{i,t} \theta_{it} = \frac{1}{4} J_{AUG}. \quad \tag{22}$$

Combining (20), (21), and (22) proves the claim.

Next, we show that the dual variables (16)-(19) are feasible for DLP.

**Claim 6.** The defined dual variables (16), (17), (18), and (19) are feasible when running the augmented switch rates (7), i.e., $J_{DUAL}$ is produced by a feasible solution to DLP.

**Proof:** For any $t \geq R_k$, $\alpha_k = \sum_{s,t} \alpha_{ks} + \sum_{s,t} \alpha_{ks}$. Since $\alpha_{ks} = w_k 1_k$, $\sum_{s,t} \alpha_{ks} \leq tw_k$. From the definition of $\gamma_{ijkt}$, we get

$$\sum_{i,j} \gamma_{ijkt} = \sum_{i,j} \frac{w_k}{n_{kt}} 1_{ijk} = \sum_{i,j} \frac{w_k}{n_{kt}} 1_{ij} = \frac{w_k}{n_{kt}} 1_k.$$

Since $\alpha_{ks} = \sum_{i,j} \gamma_{ijkt}$, $\sum_{s,t} \alpha_{ks} \leq \sum_{i,j} \gamma_{ijkt}$ holds with equality. This shows the feasibility of constraint (14).

Now we show the feasibility of constraint (15). If $1_{ijk} = 0$, then the constraint is clearly satisfied. Consider the case where $1_{ijk} = 1$. Since $n_{ks} \geq 1$ whenever $1_{ijk} = 1$,

$$\sum_{s,t} \gamma_{ijks} d_{ijk} = \sum_{s,t} \frac{w_k}{n_{ks}} 1_{ij} \leq \frac{w_k}{n_{ks}} \sum_{s,t} 1_{ij} = \frac{w_k}{n_{ks}} 1_k.$$
Let coflow $k$ not need the other term in the denominator of (5) to ensure 4 in claim 3 instead of stretching of $2^4$ instead of the baseline algorithm. Since the rates for the baseline algorithm are twice those in (6), and it now starts at baseline instead of $2^4$.

For the concurrent open shop problem, the Corollary 1. The number of coflows $n$, number of ports on each side $m$, the maximum number of non-zero entries in the demand matrices $p$, maximum demand for any flow $D$, and the last release time for any coflow $T$ are given as parameters.

1) The number of coflows $n$, number of ports on each side $m$, the maximum number of non-zero entries in the demand matrices $p$, maximum demand for any flow $D$, and the last release time for any coflow $T$ are given as parameters.

2) For each coflow $k$
   a) a number between 1 and $p$ is chosen uniformly at random, which is the number of non zero entries in that coflow, defined as $p_k$.
   b) $p_k$ many $(i,j)$ input-output pairs corresponding to the $p_k$ flows of coflow $k$ are chosen uniformly at random from the $m^2$ possible input-output port pairs.
   c) Each of $p_k$ pairs is given a demand from 1 to $D$ chosen uniformly at random.
   d) For each coflow, a release time is chosen uniformly at random from a time interval from $[0,T]$.

Using this synthetic data, we run the following experiments, that illustrate the effect of the parameter $p$ and the number of coflows $n$ on the performance of BlindFlow. In Fig. 2(a) and 2(b), we compare the performance of BlindFlow, a clairvoyant lower bound on the coflow completion time from [11], and the non-clairvoyant algorithm Aalo [14], as a function of $p$ and $n$ respectively. For Fig. 2(a), we use $n = 20$, number of ports on each side $m = 15$, maximum demand on any flow $D = 15$ and last release time $T = 50$, while for Fig. 2(b), we use $p = 140$ and keep all the other parameters the same. The performance of the BlindFlow algorithm is close to but inferior to that of Aalo. However, BlindFlow is much easier to implement than Aalo. The performance of the non-clairvoyant BlindFlow is worse than the clairvoyant lower bound as expected. Importantly, the ratio between the two does not seem to scale with $p$, and is relatively small in contrast to the theoretical guarantee of $8p$ we have obtained.

Next, in Fig. 3, we compare the performance of BlindFlow on the real world data that is based on a Hive/MapReduce trace collected by Chowdhury et al. [4] from a Facebook cluster available at [29]. This trace has been used previously as well [6], [11], [14]. The original trace is from a 3000-machine 150-rack MapReduce cluster at Facebook. The original trace has 526 coflows, however, for simulation feasibility on limited
machines, we use the first 100 coflows from this trace and execute the three algorithms on this. For our simulation, we assume that the rate of flow of any link at maximum capacity is 1 Mbps. Once again we see that the performance of BlindFlow is far better than the \(8p\) guarantee that we have derived compared to the clairvoyant lower bound. Moreover, for this simulation as well, Aalo outperforms BlindFlow, however, as stated before, BlindFlow is easier to implement, and is amenable for obtaining theoretical guarantee compared to the clairvoyant optimal algorithm, unlike Aalo for which no theoretical guarantee is available.

VIII. CONCLUSIONS

In this paper, for the first time, we have derived theoretical guarantees on the approximation ratio of weighted coflow completion time problem in the non-clairvoyant setting. The non-clairvoyant setting is both more robust, since the exact demand is unknown, and theoretically challenging, since we are comparing against the optimal algorithm that is clairvoyant. The guarantee we obtain compared to the clairvoyant optimal algorithm is a function of \(p\), the maximum number of flows that any coflow can have, however, as shown via simulations, the actual performance is superior to the derived guarantee. It is not clear immediately whether the guarantee is a function of \(p\), because we are comparing against the clairvoyant optimal algorithm or the analysis itself is loose. We believe the results of this paper will lead to further progress in the area of non-clairvoyant coflow scheduling.

REFERENCES