Prediction-based Coflow Scheduling

Olivier Brun, Balakrishna Prabhu, Oumayma Haddaji

Prediction-based Coflow Scheduling

Olivier Brun, Balakrishna Prabhu and Oumayma Haddaji*

LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France

October 6th, 2023

Abstract

We explore the problem of minimizing the weighted average coflow completion time of coflows when flow sizes are unknown but unreliable predictions on them are available. We propose to use Sincronia, a 4-approximation algorithm when actual flow sizes are known, directly on the predictions. For this variant of Sincronia, we obtain a finite upper bound on the approximation ratio that combines unreliability on predictions and the weights of the coflows. On several numerical experiments, it is shown that this bound is too conservative, and that in practice Sincronia with predictions has a much better performance on average as well as in the worst-case.

1 Introduction

Cloud computing infrastructures have popularized the execution of massively parallel computing algorithms on vast amounts of data. Nowadays, most cloud providers allow their customers to launch parallel computations on their own data using Big Data applications such as MapReduce, Hadoop or Spark [15, 39]. These parallel applications typically alternate between computation stages and communication stages, during which the application's tasks exchange intermediate results using the datacenter's internal network. For some workloads, these data transfers can account for more than 50% of job completion times [12].

Indeed, for massively parallel applications, data transfers result in thousands of communication flows that operate almost concurrently within the network. While datacenter networks are high-speed and low-latency networks, they were designed to optimize the performance of individual flows, without considering the needs of the applications generating these flows. However, the performance of a parallel application depends on when all

*The work of O. Haddaji was done at LAAS-CNRS while she was a student at Ecole Polytechnique de Tunisie.

data transfers have been completed, as the intermediate results from the previous iteration are necessary to start a new iteration.

While parallel task scheduling is a well-explored topic, the scheduling of data flows in datacenter networks has only been studied since 2012, with the foundational work of Chowdhury and Stoica [10]. They introduced the concept of coflow to study the scheduling of data flows from concurrent parallel applications sharing the resources of a datacenter network. A coflow is defined as a collection of semantically-related data flows generated by a parallel application between two iterations. The coflow completion time (CCT) is the time at which all flows of a coflow have completed. In MapReduce, for instance, a coflow is composed of all flows sent from mapper to reducer nodes. These flows are launched as soon as the mapper nodes finish their computation tasks.

Chowdhury and Stoica showed that by scheduling flows from different concurrent coflows, it was possible to significantly improve application performance. Since then, efficient coflow scheduling methods aiming to minimize the average CCT have been studied in the literature. Some of them have been devised for the offline problem, in which all coflows to be scheduled are initially present in the system (or with known release dates), whereas others were devised for the online problem in which coflows arrive over time.

Furthermore, coflow scheduling was mainly considered in the clairvoyant setting, where upon arrival of a coflow, the source and destination ports as well as the precise volume of each and every of its constituent flow are also revealed. Efficient algorithms can be devised for this setting, but they require *a priori* information about coflows which is hard to obtain in practice. Therefore, some authors have also investigated the non-clairvoyant setting, where on a coflow arrival, only the number of flows and their input and output port are revealed, while their volumes remains unknown.

The present work goes beyond the conventional clairvoyant and non-clairvoyant settings by exploring an alternative one in which predictions on the flow sizes are revealed to the coflow scheduler. These predictions could be obtained for instance from historical data on previous executions using machine learning (ML) techniques. The issue is that the actual flow size information remains unavailable and the ML predictions are unreliable. In that context, the main question we investigate is how to exploit these predictions for coflow scheduling, and whether it is even advisable to do so.

1.1 Contributions

We consider the problem of scheduling coflows with the objective of minimizing the average weighted CCT when only predictions on the actual volumes of coflows are given. We propose using Sincronia directly on the predictions and provide worst-case upper bound on its approximation ratio compared to the optimal weighted CCT computed using the actual volumes. This bound

is obtained by combining two different upper bounds: one obtained from using the predictions and the other which is independent of the predictions.

Assuming that the predictions lie in the interval $[\mu_{min}x, \mu_{max}x]$, with x being the actual value, we first show that the CCT of Sincronia is at most $4(\mu_{min}^{-1}\mu_{max})^2$ of the optimal value. This upper bound coincides with that of Sincronia when there is no error in the predictions, that is, when $\mu_{min} = \mu_{max} = 1$. However, as the interval of predictions becomes large, this bound can become arbitrarily big, going to infinity as $\mu_{min} \to 0$ or $\mu_{max} \to \infty$. In this case, the bound becomes too large to be useful.

To overcome this drawback, we obtain a second upper bound that is independent of the quality of the predictions and that depends uniquely on the weights of the coflows: $(\sum_k w_k)/(\min_k w_k)$, where w_k is weight of coflow w_k.

The worst-case performance of Sincronia with predictions is then just the minimum of these two upper bounds. It can be seen that performance of Sincronia with predictions remains bounded even when the interval of predictions becomes large.

Finally, on several numerical examples we observe that in practice, these bounds are too conservative and the average performance of Sincronia with predictions remains close to that of optimal value computed with the actual volumes.

1.2 Organization

The paper is organized as follows. Section 2 is devoted to related work. Section 3 introduces the main notations used throughout the paper and presents the BlindFlow and Sincronia algorithms proposed for non-clairvoyant and clairvoyant coflow scheduling, respectively. In Section 4, we present our main theoretical results for the setting in which only unreliable ML predictions on flow sizes are available for coflow scheduling. Section 5 is devoted to numerical results. Finally, some conclusions are drawn and future research directions are suggested in Section 6.

2 Related Work

Since the introduction of the coflow concept by Chowdhury and Stoica [10, 14], several coflow scheduling algorithms have been proposed [35]. The problem can be viewed as a concurrent open-shop scheduling problem, but with coupled resources as transmitting a flow requires capacity on both the input and output ports. Most studies have focused on minimizing the average CCT in the clairvoyant case, that is, assuming complete information on the coflows. Even under this simplifying assumption, the problem is known to be NP-hard [13] and inapproximable below a factor of 2 [5, 29]. The offline minimisation of the average CCT can be formulated as an integer

linear problem (ILP) [23], but this approach does not scale well and cannot cope with the very large instances encountered in datacenter networks. Research has therefore focused on approximation algorithms and heuristics [1, 2, 7, 9, 24, 31, 32, 34, 36, 41].

One of the first clairvoyant schedulers for minimizing the average CCT was Varys [13], which is still a basic reference today as it introduced many key concepts. Varys schedules coflows iteratively by considering at each stage the bottleneck port (given the remaining coflows to be scheduled), and choosing the coflow with the least volume on that port. The rate allocation to the flows of the chosen coflow is calculated so that they all terminate simultaneously. Another popular scheduler is Sincronia which was proposed by Agarwal et al [1]. Based on the observation that an arbitrary rate allocation is not necessarily achievable by the transport layer, they schedule coflows by assigning priorities to them using a primal-dual algorithm for which an approximation ratio of 4 can be proven.

Beside coflow scheduling, task placement can also be leveraged to favour data locality and minimise network contention, thereby minimizing the average CCT. This joint problem of coflow placement and scheduling was considered in [26] and [44] in the clairvoyant setting.

The above works focus on CCT minimization. The nature of the problem is radically different for real-time parallel applications with strict deadline constraints [22]. In this case, coflow admission control and scheduling must be carried out jointly. The objective is then to maximise the number of accepted coflows, while ensuring that they can meet their deadlines [20, 21, 33, 37, 38].

Although algorithms such as Varys or Sincronia provide fairly efficient solutions to the coflow scheduling problem, they require *a priori* information about the coflows, which is difficult to obtain in practice. Existing non-clairvoyant solutions such as Aalo [11] generalize the Least Attained Service (LAS) scheduling discipline to solve this problem (see also [17,42]). However, they fail to identify the flows transmitted on bottleneck ports which result in a poor rate allocation. The Fai algorithm [19] attempts to solve this problem. The CODA solution, on the other hand, assumes that the scheduler is not even warned of the arrival of coflows, and proposes a machine-learning scheme for identifying the flows belonging to the same coflow before scheduling them [40]. Baraat [16] is another decentralized task-oriented scheduling algorithm for datacenter networks. It performs FIFO-LM scheduling in a decentralized manner, without any explicit coordination between network routers. Another very relevant work is [6] in which Bhimaraju et al. propose the BlindFlow algorithm for non-clairvoyant coflow scheduling. This algorithm, which generalizes the round robin (RR) scheduling discipline, is shown to be $8\,p$-approximate, where p is the maximum number of flows that any coflow can have.

To the extent of our knowledge, the present paper is the first one that

goes beyond the conventional clairvoyant and non-clairvoyant settings. We consider a setting in which the actual flow sizes are unknown, but unreliable predictions on them are available. The improvement of online algorithms via ML predictions is a recent line of research which has attracted a lot of studies in the last few years, in particular for online scheduling, see e.g. [3, 4, 8, 18, 25, 27, 43]. Though coflow scheduling significantly differs from the more traditional scheduling problems addressed in these references, our work is inspired by them, and in particular [18, 27].

3 Background Material

As customary for coflow scheduling, we represent the datacenter network using the *big switch model* [13], which abstracts out the datacenter network fabric as one big switch interconnecting servers. The underlying assumption is that the fabric core can sustain 100% throughput and only the ingress and egress ports are potential congestion points. This assumption is well-satisfied by modern datacenter networks as, due to large bisection capacity and customary usage of load balancing, traffic congestion is typically observed only at the rack access ports leading to the ToR switches. We let $\mathcal{L}$ be the set of ports and denote by b_ℓ he capacity of port $\ell \in \mathcal{L}$

We consider the offline setting in which all coflows are initially present in the system. We let $\mathcal{C} = \{1, 2, \ldots, n\}$ be the set of coflows. Each coflow k is a collection $F_k = \{1, 2, \ldots, n_k\}$ of flows, where flow j of coflow k is characterized by its source port $s^{k,j}$, its destination port $t^{k,j}$ and its volume $v^{k,j}$. Each coflow k may be assigned a weight w_k (default weight is 1). We define the constant $x_\ell^{k,j}$ as 1 if flow $j \in F_k$ of coflow $k \in \mathcal{C}$ uses port ℓ (that is, $s^{k,j} = \ell$ or $t^{k,j} = \ell$), and as 0 otherwise. We also define $F_{k,\ell}$ as the set of flows in F_k that use port ℓ, that is, the set of flows $j \in F_k$ such that $x_\ell^{k,j} = 1$. In the following, we shall denote by C_k the completion time of coflow $k \in \mathcal{C}$. The objective is to schedule coflows so as to minimize the weighted CCT $\sum_{k\in\mathcal{C}} w_k C_k$.

In the clairvoyant setting where flow sizes $v^{k,j}$ are known, the problem can be stated as follows:

$$\min_{r} \quad \sum_{k\in\mathcal{C}} w_k C_k \tag{P1}$$

$$\text{s.t.} \quad \sum_{k\in\mathcal{C}} \sum_{j\in F_{k,\ell}} r^{k,j}(t) \le b_\ell, \quad \forall \ell \in \mathcal{L}, \forall t \in \mathcal{T}, \tag{1}$$

$$\int_0^{C_k} r^{k,j}(t)\, dt \ge v^{k,j}, \quad \forall j \in F_k, \forall k \in \mathcal{C}, \tag{2}$$

where $r^{k,j}(t) \in \mathbb{R}_+$ is the rate allocated to flow $j \in F_k$ at time t and $\mathcal{T}$ is the time horizon. Constraint (1) expresses that, at any instant t, the total

rate that port ℓ assigns to flows cannot exceed its capacity b_ℓ. Constraint (2) ensures that the data of flows of each coflow k should be completely transmitted before its completion time C_k.

3.1 Non-clairvoyant Scheduling of Coflows

In [6], the authors present theoretical guarantees on approximating the sum of coflow completion times in the non-clairvoyant setting, where on a coflow arrival, only the number of flows and their input-output ports are revealed, while the flow volumes are unknown.

They propose an algorithm which is called BlindFlow. It divides the capacity of each port among all the flows that require that particular port in proportion to flow weights, so that the maximum rate that can be allocated to flow $j \in F_k$ on port ℓ at time t is

$$r_\ell^{k,j}(t) = \frac{w_k x_\ell^{k,j} \mathbb{1}^{k,j}(t)}{W_\ell(t)}\, b_\ell, \tag{3}$$

where $W_\ell(t) = \sum_{c \in \mathcal{C}} w_c \sum_{f \in F_c} x_\ell^{c,f} \mathbb{1}^{c,f}(t)$ and $\mathbb{1}^{k,j}(t)$ is 1 if flow j of coflow k is yet to finish at time t, and 0 otherwise. As each flow uses two ports, a natural rate allocation, which is referred to as the round robin (RR) allocation in the following, is to allocate rate $r_{RR}^{k,j}(t) = \min\{r_i^{k,j}(t), r_o^{k,j}(t)\}$ to flow $j \in F_k$ at time t, where i and o are its input and output ports, respectively. It is easy to see that it yields

$$r_{RR}^{k,j}(t) = \frac{w_k \mathbb{1}^{k,j}(t)}{\max\{W_i(t)/b_i, W_o(t)/b_o\}}. \tag{4}$$

The rate $r_{BF}^{k,j}(t)$ allocated to this flow at time t by BlindFlow is slightly different and is given by

$$r_{BF}^{k,j}(t) = \frac{w_k \mathbb{1}^{k,j}(t)}{W_i(t)/b_i + W_o(t)/b_o}. \tag{5}$$

in which the max operator has been replaced by a sum. Note that (5) might produce a schedule where the rates of some flows can be increased without violating the feasibility on any port. Nevertheless, it should be clear that any performance guarantee obtained for BlindFlow also holds for better rate allocations, such as the RR allocation in (4). The main result proven in [6] is stated in Theorem 1.

Theorem 1. *The rate allocation* (5) *of BlindFlow is feasible and* $8 \times p$ *approximate, where* $p = \max_{k \in \mathcal{C}} |F_k|$ *is the maximum number of flows that any coflow can have.*

3.2 Clairvoyant Scheduling of Coflows

We now consider the clairvoyant setting. We first present an ILP formulation of the problem, and then describe the Sincronia algorithm.

3.2.1 ILP formulation

A time-indexed mixed ILP formulation was proposed in [23] for minimizing the average CCT in the clairvoyant seeting. Let T be a time-horizon and assume that it is partitioned into T_s disjoint slots of duration Δ units of time. We let $\mathcal{T} = \{1, \ldots, T_s\}$. The model computes the fraction of the total volume to be transferred by each flow at each time slot together with the completion time of each coflow. Three types of decision variables are required. Variable $f_t^{k,j} \in [0,1]$ represents the fraction of the volume $v^{k,j}$ of flow $j \in F_k$ sent during time slot t, whereas the binary variable y_t^k is defined as 1 if t is the final time-slot used by coflow k, and 0 otherwise. Finally, the variable $\gamma_t^k \in [0,1]$ is defined as the unused percentage of the final time-slot t for coflow k. Note that with these notations the completion time of coflow k can be written as $C_k = \sum_{t \in \mathcal{T}} \Delta\,(t y_t^k - \gamma_t^k)$. The model is as follows.

$$\text{Min.} \sum_{k \in \mathcal{C}} \sum_{t \in \mathcal{T}} \Delta\,(t y_t^k - \gamma_t^k), \tag{6}$$

s.t.

$$\sum_{t \in \mathcal{T}} y_t^k = 1,\ k \in \mathcal{C}, \tag{7}$$

$$\sum_{t \in \mathcal{T}} f_t^{k,j} = 1,\ j \in F_k, k \in \mathcal{C}, \tag{8}$$

$$\gamma_t^k \leq y_t^k,\ k \in \mathcal{C}, t \in \mathcal{T}, \tag{9}$$

$$\sum_{k \in \mathcal{C}} \sum_{j \in F_{k,\ell}} f_t^{k,j} v^{k,j} \leq b_\ell \Delta,\ \ell \in \mathcal{L}, t \in \mathcal{T}, \tag{10}$$

$$\sum_{t' \geq t} f_{t'}^{k,j} \leq \sum_{t' \geq t} y_{t'}^k,\ j \in F_k, k \in \mathcal{C}, t \in \mathcal{T}, \tag{11}$$

$$\sum_{j \in F_{k,\ell}} f_t^{k,j} v^{k,j} \leq (1 - \gamma_t^k) b_\ell \Delta,\ k \in \mathcal{C}, \ell \in \mathcal{L}, t \in \mathcal{T}, \tag{12}$$

$$\gamma_t^k, f_t^{k,j} \in [0,1],\ j \in F_k, k \in \mathcal{C}, t \in \mathcal{T}, \tag{13}$$

$$y_t^k \in \{0,1\},\ k \in \mathcal{C}, t \in \mathcal{T}. \tag{14}$$

Constraints (7) select exactly one final time-slot for each coflow. Constraints (8) guarantee that all flows are served. Constraints (9) link y and γ variables. Constraints (10) represent the port capacity constraints. Constraints (11) ensure that, for every coflow, all flows are sent before the final

time-slot. Finally, Constraints (12) decreases the port capacity during the final time-slot. This allows to compute the unused part of the final time-slot of each coflow.

The above problem formulation can be solved with a linear programming solver. As expected, however, this approach does not scale well and cannot cope with the very large instances encountered in datacenter networks. Nevertheless, formulation (6)-(14) is useful for assessing the quality of other coflow scheduling algorithms, at least on small-scale scenarios.

3.2.2 Sincronia

Sincronia is one of the most popular clairvoyant coflow scheduling algorithm [1]. It guarantees a $4-$approximation, the best known approximation ratio. We shall describe below the algorithm used by Sincronia to schedule a set of coflows. In addition to the notations introduced previously, we also let $p_{\ell,k} = \sum_{j \in F_{k,\ell}} \frac{v^{k,j}}{b_\ell}$ denotes the total transmission time of coflow $k \in \mathcal{C}$ at port $\ell \in \mathcal{L}$.

Sincronia first computes a σ-order, that is, a priority order over coflows. If $\sigma(k) < \sigma(k')$, then the flows of coflow k are transmitted in the datacenter network with strict priority over the flows of coflow k'. Sincronia then uses a greedy rate allocation which preserves the σ-order, that is, it is guaranteed that coflow k will be completed before coflow k' (that is $C_k \leq C_{k'}$) if $\sigma(k) < \sigma(k')$.

The method for computing the σ-order is based on the following linear-programming problem:

$$\text{Minimize} \sum_{k \in \mathcal{C}} w_k C_k \qquad \text{(LP-Primal)}$$

$$\text{s.t}$$

$$\sum_{k \in S} p_{\ell,k} C_k \geq f_\ell(S), \quad \ell \in \mathcal{L} \text{ and } S \subseteq \mathcal{C}, \qquad (15)$$

$$C_k \geq 0, \quad k \in \mathcal{C}, \qquad (16)$$

where $f_\ell(S)$ is defined as

$$f_\ell(S) = \frac{1}{2} \sum_{k \in S} (p_{\ell,k})^2 + \frac{1}{2} \left(\sum_{k \in S} p_{\ell,k} \right)^2, \qquad (17)$$

for each port $\ell \in \mathcal{L}$ and each subset $S \subseteq \mathcal{C}$ of coflows. Inequalities (15) correspond to the so-called parallel inequalities [28, 30]. In non-preemptive single-machine scheduling problems, these inequalities define the convex hull of feasible completion time vectors in the absence of precedence constraints. Note that the formulation of problem LP-Primal does not account for the

coupling between the resources of the fabric, that is, it ignores the fact that a flow can be transmitted only if its input and output ports are available. In that respect, problem LP-Primal is a relaxation of the original coflow scheduling problem. It is proven in [1] (see Lemma 1) that the optimal value of LP-Primal provides a lower bound for the coflow scheduling problem, that is, $\sum_{k\in\mathcal{C}} w_k C_k^* \leq \sum_{k\in\mathcal{C}} w_k C_k^{OPT}$, where the C_k^* are the optimal values of the completion times in problem LP-Primal whereas the C_k^{OPT} are the completion times in an optimal schedule.

Sincronia solves problem LP-Primal with a primal-dual algorithm. The dual problem is as follows

$$\text{Maximize} \sum_{\ell\in\mathcal{L}} \sum_{S\subseteq\mathcal{C}} f_\ell(S)\, y_{\ell,S} \tag{LP-Dual}$$

s.t

$$\sum_{S:k\in S} \sum_{\ell\in\mathcal{L}} p_{\ell,k} y_{\ell,S} \leq w_k, \quad \text{for all } k \in \mathcal{C}, \tag{18}$$

$$y_{\ell,S} \geq 0, \quad \text{for all } \ell \in \mathcal{L} \text{ and } S \subseteq \mathcal{C}. \tag{19}$$

The Sincronia primal-dual algorithm is shown in Algorithm 1. The dual variables $y_{\ell,S}$ are initialized to 0 and the set S of unscheduled coflows is initially set to $\mathcal{C}$. At each iteration, the algorithm assigns a priority to one of the unscheduled coflows and then removes it from the set S. Note that the ordering is from last to first. At each iteration t, the algorithm determines the bottleneck port b, that is, the port ℓ for which the total completion time $\sum_{k\in S} p_{\ell,k}$ is maximum. It then determines the coflow k^* with the largest weighted processing time on the bottleneck: $k^* = \text{argmin}_{k\in S}\left(\frac{w_k}{p_{b,k}}\right)$. Coflow k^* is assigned priority t with $\sigma(k^*) = t$ and it is removed from the set of unscheduled coflows S. Before that, the dual variable $y_{b,S}$ is set to $w_{k^*}/p_{b,k^*}$ and the primal variable C_{k^*} is set to the total completion time at port b: $C_{k^*} = \sum_{k\in S} p_{b,k}$. The weights of the other coflows are also updated as follows: $w_k \leftarrow w_k - w_{k^*}\frac{p_{b,k}}{p_{b,k^*}}$. Note that this does not change the weight of a coflow k which does not use the bottleneck port.

It is easy to prove that Algorithm 1 produces a primal feasible solution and a dual feasible solution. It can also be shown that the completion times of coflows produced by Algorithm 1 are such that $C_{\sigma(1)} \leq C_{\sigma(2)} \leq \ldots \leq C_{\sigma(n)}$. These properties are used in [1] to establish the approximation ratio of Sincronia, which is formally stated Theorem 2 below.

Theorem 2. *The cost of the schedule obtained with Algorithm 1 is at most two times the optimal cost. As the Greedy rate allocation is also 2-optimal, this implies that Sincronia is a 4-approximation algorithm.*

Algorithm 1 Sincronia primal-dual algorithm

$y_{\ell,A} \leftarrow 0$ for all $\ell \in \mathcal{L}$ and $A \subseteq \mathcal{C}$.
$S \leftarrow \mathcal{C}$
for $t = n \dots 1$ **do**
  $b \leftarrow \text{argmax}_{\ell \in \mathcal{L}} \sum_{k \in S} p_{\ell,k}$ ▷ Bottleneck port
  $k^* \leftarrow \text{argmin}_{k \in S} \left(\frac{w_k}{p_{b,k}} \right)$ ▷ Coflow with largest weighted proc. time
  $C_{k^*} \leftarrow \sum_{k \in S} p_{b,k}$ ▷ Set primal variable
  $y_{b,S} \leftarrow \frac{w_{k^*}}{p_{b,k^*}}$ ▷ Set dual variable
  $w_k \leftarrow w_k - w_{k^*} \frac{p_{b,k}}{p_{b,k^*}}$ for all $k \in S$ ▷ Update coflow weights
  $\sigma(k^*) \leftarrow t$ ▷ Set priority of coflow k^*
  $S \leftarrow S \setminus \{k^*\}$ ▷ Remove k^* from the set of unscheduled coflows
end for

4 Coflow Scheduling with Predictions

In this section, we consider the setting in which the actual flow sizes are unavailable and only unreliable ML predictions on them can be used for scheduling coflows. We first establish in Section 4.1 the approximation ratio of Sincronia when ran on predictions as a function of the prediction error. We then propose in Section 4.3 a consistent and robust prediction-based algorithm for coflow scheduling.

4.1 Sincronia with Predictions

In this section, we analyze the approximation ratio of Sincronia when, instead of the actual flow sizes $v^{k,j}$ of a coflow $k \in \mathcal{C}$, the algorithm only knows some predictions $\hat{v}^{k,j}$ on the flow sizes. These predictions can be written as $\hat{v}^{k,j} = v^{k,j} + \Delta v^{k,j}$, where $\Delta v^{k,j}$ represents the prediction error on the size of flow $j \in F_k$ and lies in the interval $[-v^{k,j}, \infty)$. The predicted total transmission time of coflow $k \in \mathcal{C}$ on port $\ell \in \mathcal{L}$ is then

$$\hat{p}_{\ell,k} = \sum_{j \in F_k} \frac{\hat{v}^{k,j} x_\ell^{k,j}}{b_\ell} = p_{\ell,k} + \eta_{\ell,k},$$

where $\eta_{\ell,k} = \sum_{j \in F_k} \frac{\Delta v^{k,j} x_\ell^{k,j}}{b_\ell}$ represents the prediction error on the transmission time on port ℓ. Define

$$\mu_{min} = 1 + \min_{\ell,k} \left(\frac{\eta_{\ell,k}}{p_{\ell,k}} \right) \text{ and } \mu_{max} = 1 + \max_{\ell,k} \left(\frac{\eta_{\ell,k}}{p_{\ell,k}} \right),$$

We then have the following inequalities:

$$\mu_{min}\, p_{\ell,k} \leq \hat{p}_{\ell,k} \leq \mu_{max}\, p_{\ell,k}, \tag{20}$$

for all ports $\ell \in \mathcal{L}$ and all coflows $k \in \mathcal{C}$.

When ran on predictions, Sincronia produces a σ-order and compute coflow completion times $\hat{C}_k$ which are a feasible primal solution, that is, $\hat{\mathbf{C}}$ is a feasible solution to problem LP-Primal in which the actual processing times $p_{\ell,k}$ have been replaced by the predicted processing times $\hat{p}_{\ell,k}$. Likewise, the constants $f_\ell(S)$ have been replaced by the constants $\hat{f}_\ell(S)$ obtained from (17) by using the predicted values $\hat{p}_{\ell,k}$ instead of the actual values $p_{\ell,k}$. As a primal-dual algorithm, Sincronia also computes the dual variables $\hat{y}_{\ell,S}$ which provide a feasible solution to dual problem in which predicted processing times are used instead of the actual ones.

We shall assume for simplicity that Sincronia schedule the coflows in the order $1, 2, \ldots, n$, which implies that $\hat{C}_1 \leq \hat{C}_2 \leq \ldots \leq \hat{C}_n$. Our goal is to obtain an upper bound on the approximation ratio obtained by scheduling coflows in that order, as a function of μ_{min} and μ_{max}. Note that $\hat{C}_k$ is only the *estimated completion time* of coflow k. In the following we denote by $\tilde{C}_k$ the actual completion time of coflow $k \in \mathcal{C}$ obtained by scheduling coflows in the order $1, 2, \ldots, n$. We first establish a bound on the predicted weighted sum of completion times.

Lemma 1. *Let $\hat{C}_k$ be the predicted completion time of coflow k when Algorithm 1 is ran over the predictions, and C_k^{OPT} be its completion time in an optimal schedule. Then*

$$\sum_{k \in \mathcal{C}} w_k \hat{C}_k \leq \frac{2\,\mu_{max}^2}{\mu_{min}} \sum_{k \in \mathcal{C}} w_k C_k^{OPT}, \tag{21}$$

Proof. See Appendix A. □

We now use Lemma 1 to establish the approximation ratio of the prediction-based Sincronia algorithm.

Theorem 3. *Scheduling coflows in the order determined by Sincronia with predictions as inputs yields a schedule whose weighted sum of completion times is at most $4 \times \left(\frac{\mu_{max}}{\mu_{min}}\right)^2$ the optimal one.*

Proof. See Appendix B. □

Note that the approximation ratio in Theorem 3 is identical to that of Sincronia for perfect predictions, that is, when $\mu_{max} = \mu_{min} = 1$. More generally, the prediction-based Sincronia algorithm computes the same σ-order as the clairvoyant one when $\mu_{max} = \mu_{min}$. However, the bound in Theorem 3 suggests that Sincronia is not robust to prediction errors, as its approximation ratio grows unboundedly as the ratio μ_{max}/μ_{min} increases. We shall see in Section 4.2 that this is not the case.

To illustrate the above result, consider the scenario where, although unreliable, the relative prediction error $\frac{\Delta v^{k,j}}{v^{k,j}}$ on the flow sizes is guaranteed to lie in the interval $\left[-\frac{1}{2}, \frac{1}{2}\right]$ uniformly for all flows j and all coflow $k \in \mathcal{C}$. In that case, it is easy to see $\mu_{min} \geq \frac{1}{2}$ and $\mu_{max} \leq \frac{3}{2}$, so that Sincronia with predictions as inputs guarantees a $36-$approximation, that is, a worst-case approximation ratio which is $9\times$ that of Sincronia when ran over the actual flow sizes. As we shall see in Section 5, the experimental performance of the prediction-based Sincronia algorithm is far better than this theoretical guarantee.

4.2 Approximation ratio independent of predictions

The bound in Theorem 3 depends upon the quality of predictions. If μ_{min} is close to 0, then the bound goes towards infinity. This happens because the bound makes use of the completion times, $\hat{C}_k$, computed from the predicted processing times, $\hat{p}_{\ell,k}$s, which can highly different from the actual processing times. For example, the predictions can be of the order of ϵ a small quantity while the actual ones could be of the order of 1. The predicted completion times will be of the order of ϵ which is much smaller than the actual completion times which will be of the order of 1.

A bound that does suffer from this drawback can be obtained from the observation that Sincronia is an order-based scheduling algorithm. Its total weighted completion time will be upper bounded by that of the worst-case order, that is:

$$\sum_k w_k \tilde{C}_k \leq \max_\sigma \sum_k w_{\sigma(k)} C_{\sigma(k)}. \tag{22}$$

Beside, the optimal total weighted completion time is at least that of the last coflow in an optimal order:

$$\sum_k w_k C_k^{OPT} \geq w_{\sigma^*(n)} C_{\sigma^*(n)} \tag{23}$$

$$\geq \left(\min_k w_k\right) \max_\ell \sum_k p_{\ell,k} \tag{24}$$

where $\sigma^*(n)$ is the index of the coflow finishing last. The second inequality is a consequence of the fact that the latest completion time cannot be earlier than the time required to empty the most loaded link failing which there will be a coflow that would not have finished.

Combining the above two observations, we get the following bound on the approximation ratio of Sincronia with predictions:

$$\frac{\sum_k w_k \tilde{C}_k}{\sum_k w_k C_k^{OPT}} \leq \frac{\max_\sigma \sum_k w_{\sigma(k)} C_{\sigma(k)}}{(\min_k w_k) \max_\ell \sum_k p_{\ell,k}} \tag{25}$$

Using this inequality, we arrive at the following approximation ratio.

Theorem 4. *Sincronia with predictions is a* $2\frac{\bar{w}}{w_{min}}$ *approximation algorithm, where* $\bar{w} = \sum_k w_k$ *and* $w_{min} = \min_k w_k$.

Proof. See Appendix C. □

Note that Theorem 4 implies that Sincronia is robust to prediction errors. In particular, when all coflow weights are equal, as considered in Section 5, we have $\bar{w} = nw_{min}$ and the average CCT of Sincronia with predictions as inputs cannot be worse than $2n$ the optimal one, independently of the prediction error. Moreover, combining the upper bounds in Theorem 3 and Theorem 4, we obtain our final upper bound on the approximation ratio of the prediction-based algorithm.

Corollary 1. *The approximation ratio of Sincronia with predictions as inputs is upper bounded by* $\min\left\{4 \times \left(\frac{\mu_{max}}{\mu_{min}}\right)^2, 2\frac{\bar{w}}{w_{min}}\right\}$.

Proof. The proof directly follows from Theorem 3 and Theorem 4. □

4.3 A consistent and robust prediction-based algorithm for coflow scheduling

In [27], Kumar, Purohit and Svitnika study online algorithms in which predictions are used. In particular, they propose a prediction-based approach for scheduling jobs of unknown sizes on a single machine so as to minimize the average completion time of the jobs. The idea is to combine a clairvoyant algorithm and a non-clairvoyant one by running them in parallel: the clairvoyant algorithm schedules jobs a fraction λ of the time and the non-clairvoyant one schedules jobs over the remaining $(1 - \lambda)$ fraction of time. It is shown in Lemma 3.1 of [27] that if the worst-case ratio of the clairvoyant (resp. non-clairvoyant) algorithm cost to the offline optimum is α (resp. β), then the resulting combination guarantees an approximation ratio of $\min\left(\frac{\alpha}{\lambda}, \frac{\beta}{1-\lambda}\right)$. The hyperparamer λ controls the trust in the predictions and can be set so as to obtain an algorithm which is both consistent (that is, almost as good as the clairvoyant algorithm when the predictions are perfect) and robust (that is, almost as good as the non-clairvoyant one when the predictions are terrible).

In this section, we extend this approach to coflow scheduling. We combine the Sincronia clairvoyant coflow scheduling algorithm with the RR non-clairvoyant one. Sincronia uses predictions to schedule coflows in the fabric over a fraction λ of time, while RR schedules the coflows the rest of the time. In other words, the rate allocation to flow $j \in F_k$ at time t is

$$r^{k,j}(t) = \lambda\, r^{k,j}_{SP}(t) + (1 - \lambda)\, r^{k,j}_{RR}(t), \tag{26}$$

where $r_{SP}^{k,j}(t)$ is the rate allocated to this flow by Sincronia when ran over predictions. We can obtain the following guarantee on the resulting algorithm.

Theorem 5. *Running in parallel Sincronia with predictions and RR yields an algorithm with competitive ratio*

$$\min\left(\frac{4}{\lambda}\left(\frac{\mu_{max}}{\mu_{min}}\right)^2,\ \frac{2}{\lambda}\frac{\bar{w}}{w_{min}},\ \frac{8\,p}{1-\lambda}\right) \tag{27}$$

In particular, this algorithm is $\min\left\{\frac{2}{\lambda}\frac{\bar{w}}{w_{min}}, \frac{8\,p}{1-\lambda}\right\}$*-robust, that is, its approximation ratio is upper bounded by* $\min\left\{\frac{2}{\lambda}\frac{\bar{w}}{w_{min}}, \frac{8\,p}{1-\lambda}\right\}$ *independently of the prediction error. The algorithm is also* $\frac{4}{\lambda}$*-consistent, that is, its approximation ratio is* $\frac{4}{\lambda}$ *for perfect predictions.*

Proof. It easy to see that Lemma 3.1 in [27] applies directly to coflow scheduling. The proof then directly follows from Theorem 1 and Theorem 1. □

The algorithm combining Sincronia with predictions and RR gives an option to trade-off consistency and robustness. In particular, greater trust in the predictions suggests setting λ close to 1, as this leads to a competitive ratio which is approximately 4 when $\mu_{max} \approx \mu_{min} \approx 1$. In that case, the algorithm performs as well as Sincronia in the worst case. On the other hand, setting λ close to 0 is conservative and guarantees a competitive ratio of $8 \times p$ (as the non-clairvoyant RR algorithm) even for terrible predictions. As the prediction-based Sincronia is robust to prediction errors, this is however relevant only if $\bar{w} > 4pw_{min}$.

5 Numerical results

In this section, we compare the average CCTs obtained with the prediction-based Sincronia algorithm against that of other clairvoyant and non-clairvoyant algorithms on random problem instances. We first describe in Section 5.1 the procedure for generating these instances. In Section 5.2, we present the numerical results obtained on small instances for which the clairvoyant offline optimum can be computed. Section 5.3 provides the results obtained on larger problem instances.

5.1 Random instance generation

We generate random problem instances with an algorithm which works as follows. The algorithm takes as input the number of instances to be generated, the number n of coflows and the number L of ports in each instance.

For each instance, it is assumed the the first $L/2$ ports are input ports, while the other ports are output ports. The capacity of each port is 1. For each coflow k, the weight w_k is 1 and there is a flow between input port ℓ and output port ℓ' with probability p, where p is another input parameter of the algorithm. The number of flows per coflow therefore follows a binomial distribution[1] and the mean number of flows per coflow is $pL^2/4$. The size of each flow is randomly generated from a (truncated) Gaussian distribution with mean m and standard deviation σ, these parameters being also given as inputs to the algorithm.

Given a problem instance, we generate a given number of random predictions as follows. For each coflow $k \in \mathcal{C}$ and each flow $j \in F_k$, we compute the predicted volume as $\hat{v}^{k,j} = u^{k,j} \times v^{k,j}$, where the random variable $u^{k,j}$ follows a uniform distribution in the interval $[1-\delta, 1+\delta]$. We vary the coefficient δ so as to evaluate the impact of the prediction quality on the performance of the scheduling algorithm.

5.2 Comparison against the clairvoyant offline optimum

We first consider small instances for which the mixed-integer linear program presented in Section 3.2.1 can be solved. We use Gurobi as MILP solver, with a time limit of 15 minutes. Two types of instances are considered:

- **Type-1 instances**: We randomly generate $1,000$ instances with $n = 6$ coflows and $L = 6$ ports. We use the parameters $m = 5$, $\sigma = 2$ and $p = \frac{1}{3}$, so that the average number of flows per coflow is 3. The flow sizes are randomly drawn from a normal distribution with $m = 5$ and $\sigma = 2$. The parameter δ takes values in the set $\{0, 0.01, 0.1, \ldots, 0.9, 0.99\}$. For each instance and each value of δ, we generate randomly either $10,000$ or $100,000$ predictions of flow sizes.

- **Type-2 instances**: We randomly generate 100 instances with $n = 8$ coflows, $L = 12$ ports and $p = \frac{1}{4}$, so that there are on average 9 flows per coflow. The other parameters are as above, except that we use only $10,000$ predictions for each value of δ and each instance.

As the upper bound on the approximation ratio of Sincronia with predictions obtained in Theorem 3 is $\min\left\{4\frac{\mu_{max}^2}{\mu_{min}^2}, 2n\right\}$, we first evaluate how this quantity evolves as we vary the prediction error δ. Figure 1 shows the evolution of the empirical minimum, maximum and mean values obtained for this quantity as the prediction error δ varies for one of the type-1 instances (for which, $n = 6$). The results obtained with the other type-1 or type-2 instances are similar. The figure also shows with dots the theoretical minimum value 4 and maximum value $\min\left\{4(1+\delta)^2/(1-\delta)^2, 12\right\}$

[1]However, it is assumed that there is at least one flow per coflow.

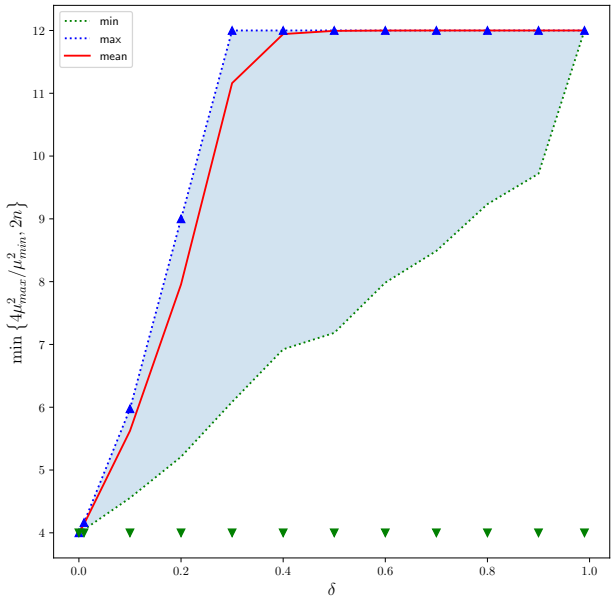

Figure 1: Minimum, maximum and mean values of the approximation ratio $\min\left\{4\frac{\mu_{max}^2}{\mu_{min}^2}, 2n\right\}$ as a function of δ for one of the type-1 instances with $100,000$ predictions.

obtained for each value of δ. We note from Figure 1 that the empirical maximum matches the theoretical value and that the average value is quite close to the maximum value. For $\delta \geq 0.4$, the performance guarantee for the prediction-based Sincronia corresponds to that of the worst σ-order.

We now consider the experimental performance of Sincronia when ran over predictions. Figure 2 shows the minimum, maximum and mean values obtained for the ratio $\frac{SIN_{pred}}{OPT}$ as δ varies for one of the type-1 instances, where SIN_{pred} is the average CCT obtained by Sincronia using predictions whereas OPT is the clairvoyant offline optimum computed with Gurobi. We generated $100,000$ predictions for each value of δ. We also denote by SIN and RR the average CCT obtained with the clairvoyant Sincronia and RR, respectively, and show the ratios $\frac{SIN}{OPT}$ and $\frac{RR}{OPT}$, which are independent of δ, on the figure. Note that for perfect predictions, that is for $\delta = 0$, the experimental performance of Sincronia with predictions matches that of the clairvoyant Sincronia. The minimum value of $\frac{SIN_{pred}}{OPT}$ is never below 1.0, as expected, and its maximum value is at most 1.65, that is, at most a 65% degradation with respect to the clairvoyant offline optimum. However, on average, the performance degradation is fairly low since it is in the order of

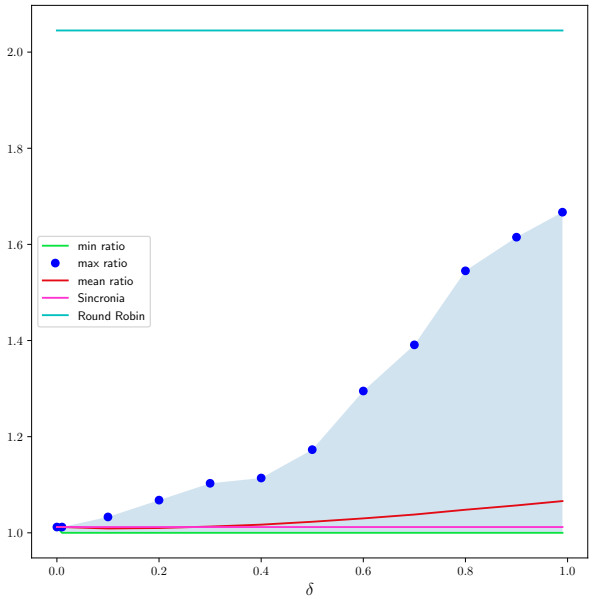

Figure 2: Minimum, maximum and average values of the ratio $\frac{SIN_{pred}}{OPT}$ as a function of δ for one of the type-1 instances.

5% for this instance. We also note that the approximation ratio of RR is significantly greater than what is obtained with Sincronia even for terrible predictions.

Figure 3 summarizes the results obtained over $1,000$ type-1 instances, using $10,000$ predictions. Here, the mean value of $\frac{SIN_{pred}}{OPT}$ is obtained by averaging over the $1,000$ instances and the $10,000$ predictions. Similarly, its maximum (resp. minimum) value corresponds to the greatest (least) ratio obtained over the $1,000$ instances and the $10,000$ predictions. The figure also shows the average values of the ratios $\frac{SIN}{OPT}$ and $\frac{RR}{OPT}$. Note that for $\delta = 0$, even though there is no prediction error, the minimum, maximum, and mean values of $\frac{SIN_{pred}}{OPT}$ do not coincide as they are obtained for different instances. We observe that the minimum value of the ratio remains at 1, which shows that wrong predictions do not necessarily harm Sincronia and can even improve its performance in some cases, which is probably due to its sub-optimality. Interestingly, we observe that on average the performance of the prediction-based Sincronia remains remarkably close to that of the clairvoyant Sincronia, even for terrible predictions. The maximum ratio $\frac{SIN_{pred}}{OPT}$ increases however with δ and slightly surpasses the average value of $\frac{RR}{OPT}$ for $\delta \geq 0.8$, indicating that the prediction-based Sincronia performs

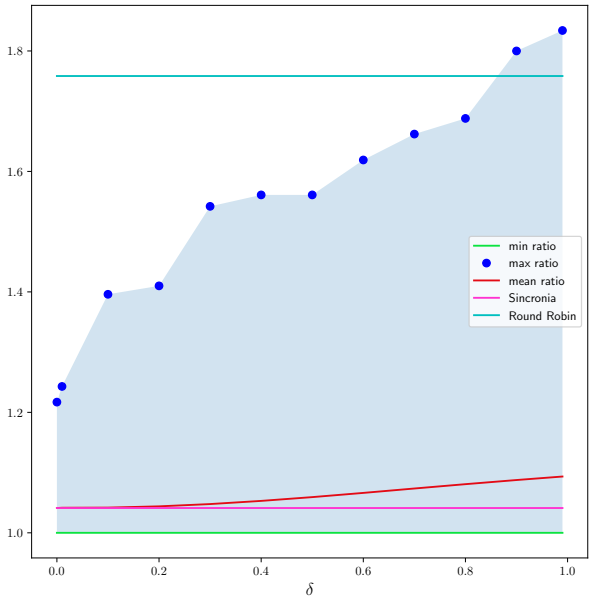

Figure 3: Minimum, maximum and average values of the ratio $\frac{SIN_{pred}}{OPT}$ over $1,000$ type-1 instances as a function of δ.

remarkably well in comparison to RR even with poor predictions.

Figure 4 is similar to Figure 3 and presents the ratios $\frac{SIN_{pred}}{OPT}$ obtained over 100 type-2 instances, using $10,000$ predictions for each one and each value of δ. We note that the minimum value occasionally falls below 1 because for some instances Gurobi was unable to reach the optimal solution within the time limit of 15 minutes. The maximum value does not exceed 1.6. As expected, the average decline in performance tends to rise as δ increases, but remains relatively low, hovering around 10% for $\delta = 0.99$. Moreover the average performance of the prediction-based Sincronia always remains quite close to that of the clairvoyant Sincronia. Another observation is that the average approximation ratio of RR remains close to 2, a significantly higher value than what is achieved by Sincronia, even with extremely poor predictions.

In summary, the above results show that the average performance of Sincronia with predictions is remarkably close to that it obtains in the clairvoyant setting, even with poor predictions, and remains within 10% of the clairvoyant offline optimum. For some bad predictions, Sincronia performance is much worse but remains acceptable and far from the theoretical worst case. Except in some exceptional cases, the prediction-based Sincronia

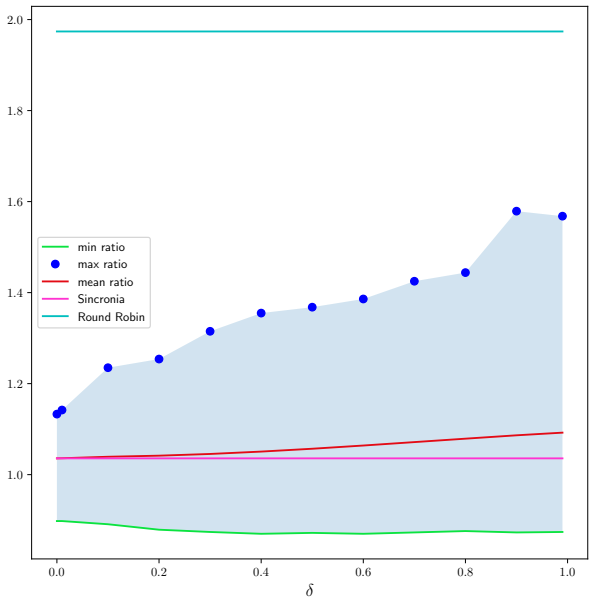

Figure 4: Minimum, maximum and average values of the ratio $\frac{SIN_{pred}}{OPT}$ over 100 type-2 instances as a function of δ.

algorithm outperforms RR.

5.3 Comparison against the clairvoyant Sincronia

We now consider larger problem instances for which the clairvoyant offline optimum cannot be computed. Specifically, we consider four datasets with $n = 10$, 15, 20 and 30 coflows, respectively. Each dataset contains 100 randomly generated instances with $L = 10$ ports and $p = 0.2$ (that is, 5 flows per coflow on average). The flow sizes are randomly drawn from a normal distribution with $m = 5$ and $\sigma = 2$. For each dataset and each instance in this dataset, we vary the parameter δ from 0 to 0.99 and generate $10,000$ random predictions of flow sizes. The number of coflows and the number of ports are kept to relatively low values as $4 \times 100 \times 10,000 \times 12 = 48,000,000$ network simulations are required, which is of course a time-consuming affair.

As the clairvoyant offline optimum is no more available, we choose as reference the clairvoyant Sincronia. Figure 5 plots the minimum, maximum and average values of the ratio $\frac{SIN_{pred}}{SIN}$ as a function of δ. As before, for each value of δ, the mean value is obtained by averaging over the 100 instances and the $10,000$ predictions, whereas the maximum (resp. minimum) value corresponds to the greatest (least) value obtained across all instances and all predictions. We observe that the average performance of Sincronia with predictions is quite close to that it obtains in the clairvoyant setting, even with extremely poor predictions (at most a 5% degradation). The worst-case performance increases with δ, but remains surprinsingly close to that obtained by the clairvoyant Sincronia (at most +42% with respect to the clairvoyant Sincronia). We also observe that on average RR performs poorly as compared to the clairvoyant Sincronia.

Figure 6 plots the results obtained for 30 coflows. The same observations can be made as for Figure 5 . However, a comparison with the latter figure shows that increasing the number of coflows while maintaining the same number of links leads to enhanced worst-case performance for the prediction-based Sincronia (at most +35% for the maximum value). The results obtained with 10 and 20 coflows are similar.

5.4 Running Sincronia with predictions and RR in parallel

From a theoretical point of view, it follows from the analysis in Section 4.3 that running Sincronia with predictions and RR in parallel can yield a smaller worst-case approximation ratio than that of the prediction-based Sincronia provided that $n > 4p$ and $\delta > \frac{\sqrt{2p}-1}{\sqrt{2p}+1}$. The empirical results obtained above show however that Sincronia performs very well and much better than RR even when feed with terrible predictions. In this section, we therefore investigate the performance obtained by running the two algorithms in parallel, focusing on a large value of λ. We choose $\lambda = 0.95$.

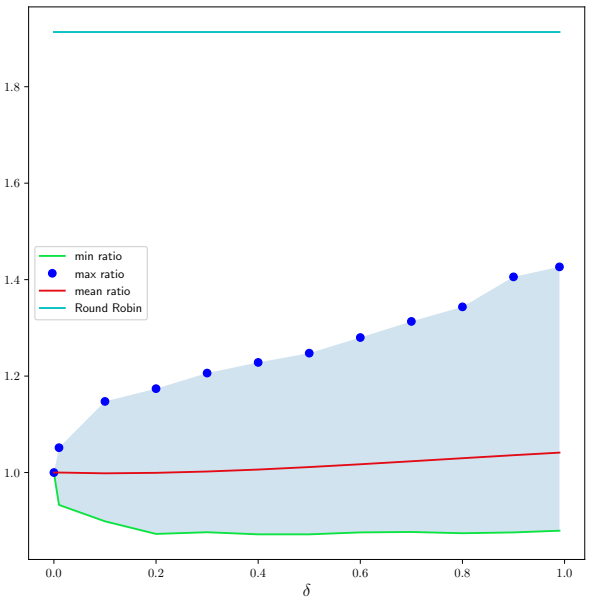

Figure 5: Minimum, maximum and average values of the ratio SIN_{pred}/SIN over 100 instances with 15 coflows and 10 ports as a function of δ.

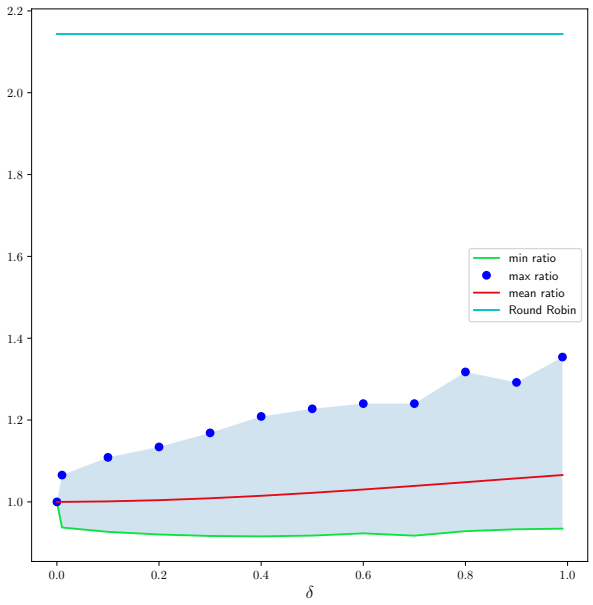

Figure 6: Minimum, maximum and average values of the ratio SIN_{pred}/SIN over 100 instances with 30 coflows and 10 ports as a function of δ.

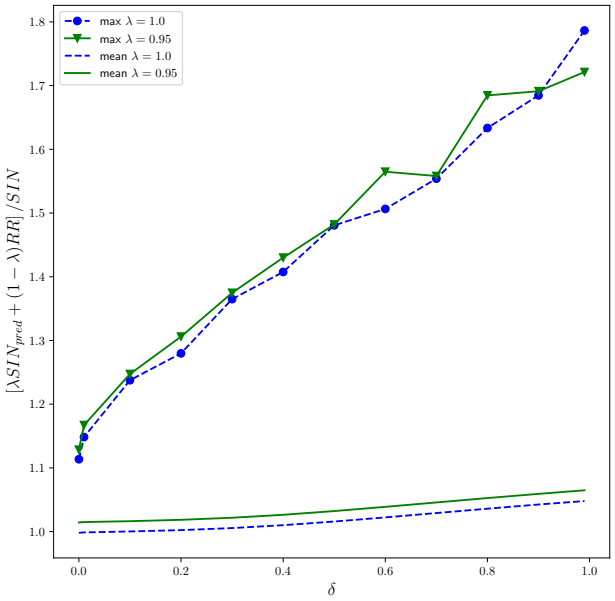

Figure 7: Maximum and average values of the ratio $\frac{\lambda SIN_{pred}+(1-\lambda)RR}{SIN}$ for $\lambda = 0.95$ and $\lambda = 1.0$ over 100 instances with 6 coflows and 6 ports as a function of δ.

Figure 7 shows the maximum and average values of the ratio $\frac{\lambda SIN_{pred}+(1-\lambda)RR}{SIN}$ over 100 instances with 6 coflows and 6 ports as a function of δ for $\lambda = 0.95$ and for $\lambda = 1.0$. For the latter value, only the prediction-based Sincronia is executed. As expected, a slight degradation of the average performance is observed when the weight of RR is increased from 0% to 5%. In terms of worst-case emprical performance, we do not observe any improvement, except for $\delta = 0.99$. We therefore conclude that, given the remarkable performance of the prediction-based Sincronia algorithm, there is no clear benefit of running it in parallel with RR from a practical point of view.

6 Conclusion

Whereas prior works on coflow scheduling focus on the conventional clairvoyant or non-clairvoyant settings, we have explored an alternative one in which the actual flow sizes are unknown, but unreliable predictions on them are available. We have established an upper bound on the approximation ratio of Sincronia with predictions as inputs and proposed to combine the prediction-based Sincronia algorithm with a RR rate allocation to obtain a

consistent and robust coflow scheduling algorithm.

Our numerical results show that Sincronia performs well even when feed with terrible predictions, with average CCT which are quite close on average to those obtained under the clairvoyant setting. The worst performance degradation is limited and tends to decrease as the number of coflows is increased. These results suggest that there is little or no benefit in combining Sincronia with predictions and a RR rate allocation. Moreover, they suggest that operating Sincronia with ML predictions could be an efficient solution in practical scenarios with hundreds of coflows and ports. We emphasize however that these conclusions have been drawn assuming a very specific model for the prediction errors. Confirmation of these conclusions for other models of the prediction errors will require further study and is left for future work.

References

[1] Saksham Agarwal, Rachit Agarwal, Shijin Rajakrishnan, David Shmoys, Akshay Narayan, and Amin Vahdat. Sincronia: Near-Optimal Network Design for Coflows. In *Proc. ACM SIGCOMM*, pages 16–29, 2018.

[2] Saba Ahmadi, Samir Khuller, Manish Purohit, and Sheng Yang. On scheduling coflows. *Algorithmica*, 82:3604 – 3629, 2020.

[3] Eric Balkanski, Vasilis Gkatzelis, and Xizhi Tan. Strategyproof scheduling with predictions. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 11:1–11:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[4] Evripidis Bampis, Konstantinos Dogeas, Alexander V. Kononov, Giorgio Lucarelli, and Fanny Pascual. Scheduling with untrusted predictions. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 4581–4587. ijcai.org, 2022.

[5] Nikhil Bansal and Subhash Khot. Inapproximability of hypergraph vertex cover and applications to scheduling problems. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part I*, volume 6198 of *Lecture Notes in Computer Science*, pages 250–261. Springer, 2010.

[6] Akhil Bhimaraju, Debanuj Nayak, and Rahul Vaze. Non-clairvoyant scheduling of coflows. In *18th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, WiOPT 2020, Volos, Greece, June 15-19, 2020*, pages 81–88. IEEE, 2020.

[7] Li Chen, Wei Cui, Baochun Li, and Bo Li. Optimizing coflow completion times with utility max-min fairness. In *Proc. IEEE INFOCOM*, pages 1–9, 2016.

[8] Woo-Hyung Cho, Shane G. Henderson, and David B. Shmoys. Scheduling with predictions. *CoRR*, abs/2212.10433, 2022.

[9] Mosharaf Chowdhury et al. Near optimal coflow scheduling in networks. In *Proc. ACM SPAA*, pages 123–134, Phoenix, AZ, USA, June 22-24 2019.

[10] Mosharaf Chowdhury and Ion Stoica. Coflow: A networking abstraction for cluster applications. In *Proc. ACM HotNets*, pages 31–36, Redmond, Washington, 2012.

[11] Mosharaf Chowdhury and Ion Stoica. Efficient coflow scheduling without prior knowledge. *SIGCOMM Comput. Commun. Rev.*, 45(4):393–406, August 2015.

[12] Mosharaf Chowdhury, Matei Zaharia, Justin Ma, Michael I. Jordan, and Ion Stoica. Managing data transfers in computer clusters with orchestra. In *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, pages 98–109, New York, NY, USA, 2011. Association for Computing Machinery.

[13] Mosharaf Chowdhury, Yuan Zhong, and Ion Stoica. Efficient Coflow Scheduling with Varys. In *Proc. ACM SIGCOMM*, pages 443–454, 2014.

[14] NM Mosharaf Kabir Chowdhury. *Coflow: A networking abstraction for distributed data-parallel applications.* PhD thesis, University of California, Berkeley, 2015.

[15] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, 2008.

[16] Fahad R. Dogar, Thomas Karagiannis, Hitesh Ballani, and Antony Rowstron. Decentralized task-aware scheduling for data center networks. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 431–442, New York, NY, USA, 2014. Association for Computing Machinery.

[17] Yuanxiang Gao, Hongfang Yu, Shouxi Luo, and Shui Yu. Information-agnostic coflow scheduling with optimal demotion thresholds. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6, 2016.

[18] Sungjin Im. Non-clairvoyant scheduling with predictions. In *Proceedings of the International Symposium on Artificial Intelligence and Mathematics 2022 (ISAIM 2022), Fort Lauderdale, Florida, USA, January 3-5, 2022*, 2022.

[19] Libin Liu, Hong Xu, Chengxi Gao, and Peng Wang. Bottleneck-aware coflow scheduling without prior knowledge. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 50–55, 2020.

[20] Shouxi Luo, Hongfang Yu, and Lemin Li. Decentralized deadline-aware coflow scheduling for datacenter networks. In *Proc. IEEE ICC*, pages 1–6, 2016.

[21] Quang-Trung Luu, Olivier Brun, Rachid El Azouzi, Francesco De Pellegrini, Balakrishna J. Prabhu, and Cédric Richier. Dcoflow: Deadline-aware scheduling algorithm for coflows in datacenter networks. In *IFIP Networking Conference, IFIP Networking 2022, Catania, Italy, June 13-16, 2022*, pages 1–9. IEEE, 2022.

[22] Shiyao Ma, Jingjie Jiang, Bo Li, and Baochun Li. Chronos: Meeting Coflow Deadlines in Data Center Networks. In *Proc. IEEE ICC*, 2016.

[23] Youcef Magnouche, Sébastien Martin, Jérémie Leguay, Francesco De Pellegrini, Rachid Elazouzi, and Cédric Richier. Branch-and-benders-cut algorithm for the weighted coflow completion time minimization problem. In Christina Büsing and Arie M. C. A. Koster, editors, *Proceedings of the 10th International Network Optimization Conference, INOC 2022, Aachen, Germany, June 7-10, 2022*, pages 1–6. OpenProceedings.org, 2022.

[24] Ruijiu Mao, Vaneet Aggarwal, and Mung Chiang. Stochastic non-preemptive co-flow scheduling with time-indexed relaxation. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 385–390, 2018.

[25] Michael Mitzenmacher. Scheduling with predictions and the price of misprediction. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 14:1–14:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[26] Ali Munir, Ting He, Ramya Raghavendra, Franck Le, and Alex X. Liu. Network scheduling aware task placement in datacenters. In *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '16, pages 221–235, New York, NY, USA, 2016. Association for Computing Machinery.

[27] Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ml predictions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[28] M. Queyranne. Structure of a simple scheduling polyhedron. *Mathematical Programming*, 58:263–285, 1993.

[29] Sushant Sachdeva and Rishi Saket. Optimal inapproximability for scheduling problems via structural hardness for hypergraph vertex cover. In *2013 IEEE Conference on Computational Complexity*, pages 219–229, 2013.

[30] Andreas S. Schulz. Scheduling to minimize total weighted completion time: Performance guarantees of lp-based heuristics and lower bounds. In *Conference on Integer Programming and Combinatorial Optimization*, 1996.

[31] Mehrnoosh Shafiee and Javad Ghaderi. An improved bound for minimizing the total weighted completion time of coflows in datacenters. *IEEE/ACM Trans. Netw.*, 26(4):1674–1687, 2018.

[32] Li Shi, Yang Liu, Junwei Zhang, and Thomas Robertazzi. Coflow scheduling in data centers: routing and bandwidth allocation. *IEEE Trans. Parallel Distrib. Syst.*, 32(11):2661–2675, 2021.

[33] Shih-Hao Tseng and Ao Tang. Coflow deadline scheduling via network-aware optimization. In *Proc. Annu. Allert. Conf. Commun. Control Comput.*, pages 829–833, 2018.

[34] Junchao Wang, Huan Zhou, Yang Hu, Cees De Laat, and Zhiming Zhao. Deadline-aware coflow scheduling in a dag. In *Proc. IEEE CloudCom*, pages 341–346, 2017.

[35] Shuo Wang, Jiao Zhang, Tao Huang, Jiang Liu, Tian Pan, and Yunjie Liu. A survey of coflow scheduling schemes for data center networks. *IEEE Commun. Mag.*, 56(6):179–185, 2018.

[36] Zhiliang Wang, Han Zhang, Xingang Shi, Xia Yin, Yahui Li, Haijun Geng, Qianhong Wu, and Jianwei Liu. Efficient scheduling of weighted

coflows in data centers. *IEEE Transactions on Parallel and Distributed Systems*, 30(9):2003–2017, 2019.

[37] Renhai Xu, Wenxin Li, Keqiu Li, and Xiaobo Zhou. Shaping Deadline Coflows to Accelerate Non-Deadline Coflows. In *Proc. EEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. IEEE, 2019.

[38] Renhai Xu, Wenxin Li, Keqiu Li, Xiaobo Zhou, and Heng Qi. Scheduling mix-coflows in datacenter networks. *IEEE Transactions on Network and Service Management*, 18:2002–2015, 2021.

[39] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, Ion Stoica, et al. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.

[40] Hong Zhang, Li Chen, Bairen Yi, Kai Chen, Mosharaf Chowdhury, and Yanhui Geng. Coda: Toward automatically identifying and scheduling coflows in the dark. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, pages 160–173, New York, NY, USA, 2016. Association for Computing Machinery.

[41] Tong Zhang, Fengyuan Ren, Jiakun Bao, Ran Shu, and Wenxue Cheng. Minimizing coflow completion time in optical circuit switched networks. *IEEE Trans. Parallel Distrib. Syst.*, 32(2):457–469, 2021.

[42] Tong Zhang, Fengyuan Ren, Ran Shu, and Bo Wang. Scheduling coflows with incomplete information. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, pages 1–10, 2018.

[43] Tianming Zhao, Wei Li, and Albert Y. Zomaya. Uniform machine scheduling with predictions. In Akshat Kumar, Sylvie Thiébaux, Pradeep Varakantham, and William Yeoh, editors, *Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling, ICAPS 2022, Singapore (virtual), June 13-24, 2022*, pages 413–422. AAAI Press, 2022.

[44] Yangming Zhao, Chen Tian, Jingyuan Fan, Tong Guan, Xiaoning Zhang, and Chunming Qiao. Joint reducer placement and coflow bandwidth scheduling for computing clusters. *IEEE/ACM Transactions on Networking*, 29(1):438–451, 2021.

A Proof of Lemma 1

Proof of Lemma 1. Let $\hat{C}_k$ and $\hat{y}_{\ell,S}$ be the primal and dual variables, respectively, obtained after the execution of Algorithm 1 with the predictions as inputs. We then have for each coflow k

$$\sum_{\ell \in \mathcal{L}} \hat{p}_{\ell,k} \sum_{S \subseteq \mathcal{C}, k \in S} \hat{y}_{\ell,S} = w_k.$$

Consider the dual variables $y_{\ell,S}$ defined as follows: $y_{\ell,S} = \mu_{min}\, \hat{y}_{\ell,S}$ for all $\ell \in \mathcal{L}$ and all $S \subseteq \mathcal{C}$. We then have

$$\begin{aligned}
\sum_{\ell \in \mathcal{L}} p_{\ell,k} \sum_{S \subseteq \mathcal{C}, k \in S} y_{\ell,S} &= \sum_{\ell \in \mathcal{L}} p_{\ell,k} \sum_{S \subseteq \mathcal{C}, k \in S} \mu_{min}\, \hat{y}_{\ell,S}, \\
&\leq \sum_{\ell \in \mathcal{L}} \hat{p}_{\ell,k} \sum_{S \subseteq \mathcal{C}, k \in S} \hat{y}_{\ell,S}, \\
&= w_k,
\end{aligned}$$

which shows that the dual variables $(y_{\ell,S})_{\ell \in \mathcal{L}, S \subseteq \mathcal{C}}$ provide a feasible solution to problem LP-Dual. It also follows from the proof of Theorem 2 in [1] that

$$\begin{aligned}
\sum_{k \in \mathcal{C}} w_k \hat{C}_k &\leq 2 \sum_{t=1}^{n} \hat{f}_{b(t)}\left(S(t)\right)\, \hat{y}_{b(t),S(t)}, \\
&\leq \frac{2}{\mu_{min}} \sum_{t=1}^{n} \hat{f}_{b(t)}\left(S(t)\right)\, y_{b(t),S(t)},
\end{aligned}$$

In addition,

$$\begin{aligned}
\hat{f}_\ell(S) &= \frac{1}{2} \sum_{k \in S} (\hat{p}_{\ell,k})^2 + \frac{1}{2} \left(\sum_{k \in S} \hat{p}_{\ell,k} \right)^2, \\
&\leq \frac{1}{2} \sum_{k \in S} (\mu_{max}\, p_{\ell,k})^2 + \frac{1}{2} \left(\sum_{k \in S} \mu_{max}\, p_{\ell,k} \right)^2, \\
&= \mu_{max}^2 f_\ell(S),
\end{aligned}$$

for all $\ell \in \mathcal{L}$ and all $S \subseteq \mathcal{C}$, so that

$$\begin{aligned}
\sum_{k \in \mathcal{C}} w_k \hat{C}_k &\leq \frac{2}{\mu_{min}} \mu_{max}^2 \sum_{t=1}^{n} f_{b(t)}\left(S(t)\right)\, y_{b(t),S(t)}, \\
&\leq \frac{2\,\mu_{max}^2}{\mu_{min}} \sum_{\ell \in \mathcal{L}} \sum_{S \subseteq \mathcal{C}} f_\ell\left(S\right)\, y^*_{\ell,S}, \\
&\leq \frac{2\,\mu_{max}^2}{\mu_{min}} \sum_{k \in \mathcal{C}} w_k C_k^{OPT},
\end{aligned}$$

where the first inequality follows from the feasibility of $(y_{\ell,S})_{\ell \in \mathcal{L}, S \subseteq \mathcal{C}}$ as a solution to problem LP-Dual, and the last one follows from the weak duality theorem. □

B Proof of Theorem 3

Proof of Theorem 3. We assume that $\hat{C}_1 \leq \hat{C}_2 \leq \ldots \leq \hat{C}_n$, that is, Sincronia with predictions as inputs schedules coflows in the order $1, 2, \ldots, n$. Let $\tilde{C}_k$ be the actual completion time of coflow $k \in \mathcal{C}$ when coflows are scheduled in that order. It follows from Lemma 2 in [2] (see also the corresponding result on the Greedy allocation in [1]) that

$$\tilde{C}_j \leq 2 \max_{\ell \in \mathcal{L}} \sum_{k=1}^{j} p_{\ell,k}, \quad \text{for all } j \in \mathcal{C}.$$

When coflow j is scheduled by Algorithm 1, the set of unscheduled coflows is $\{1, 2, \ldots, j\}$ and the predicted completion time of coflow j is set to

$$\hat{C}_j = \max_{\ell \in \mathcal{L}} \sum_{k=1}^{j} \hat{p}_{\ell,k} \geq \mu_{min} \times \max_{\ell \in \mathcal{L}} \sum_{k=1}^{j} p_{\ell,k}.$$

It thus follows that $\tilde{C}_j \leq 2 \times \hat{C}_j / \mu_{min}$, so that

$$\begin{aligned} \sum_{j \in \mathcal{C}} w_j \tilde{C}_j &\leq \frac{2}{\mu_{min}} \sum_{j \in \mathcal{C}} w_j \hat{C}_j, \\ &\leq 4 \left(\frac{\mu_{max}}{\mu_{min}} \right)^2 \sum_{j \in \mathcal{C}} w_j C_j^{OPT}, \end{aligned}$$

where the last inequality follows from Lemma 1. □

C Proof of Theorem 4

Proof of Theorem 4. For a given order σ, we have the following inequality bounding the completion times of coflows:

$$C_{\sigma(k)} \leq 2 \max_{\ell} \sum_{j=1}^{k} p_{\ell,\sigma(j)} \tag{28}$$

from which we can bound the numerator of the RHS of (25) as

$$\max_{\sigma} \sum_{k} w_{\sigma(k)} C_{\sigma(k)} \leq 2 \max_{\sigma} \sum_{k} w_{\sigma(k)} \max_{\ell} \sum_{j=1}^{k} p_{\ell,\sigma(j)} \tag{29}$$

$$\leq 2 \max_{\sigma} \left(\sum_{k} w_{\sigma(k)} \right) \max_{\ell} \sum_{j=1}^{n} p_{\ell,\sigma(j)} \tag{30}$$

$$= 2 \left(\sum_{k} w_{k} \right) \max_{\ell} \sum_{j=1}^{n} p_{\ell,j} \tag{31}$$

For the second inequality, we have replaced k by n in $\sum_{j=1}^{k} p_{\ell,\sigma(j)}$ which makes this sum independent of k. For the equality in the third line, observe that $\sum_{j=1}^{k} p_{\ell,\sigma(j)}$ as well as the first sum are independent of the order since we are now including all the n coflows and not just the first k.

The claimed ratio now follows by substituting (31) into (25). □