

3.415-Approximation for Coflow Scheduling via Iterated Rounding

Lars Rohwedder*

Leander Schnaars[†]

Abstract

We provide an algorithm giving a $\frac{140}{41} (< 3.415)$ -approximation for Coflow Scheduling and a 4.36-approximation for Coflow Scheduling with release dates. This improves upon the best known 4- and respectively 5-approximations and addresses an open question posed by [Aga+18; Fuk22] and others. We additionally show that in an asymptotic setting, the algorithm achieves a $(2 + \epsilon)$ -approximation, which is essentially optimal under $\mathbb{P} \neq \mathbb{NP}$. The improvements are achieved using a novel edge allocation scheme using iterated LP rounding together with a framework which enables establishing strong bounds for combinations of several edge allocation algorithms.

1 Introduction

Coflow Scheduling models the problem of data exchange between various nodes in a shared network. It has been proven indispensable in improving the performance of common data exchange and distributed computing frameworks such as MapReduce [DG08], Spark [Zah+10], and Hadoop [Shv+10]. These routines form an integral part for large scale computations commonly found in applications such as bioinformatics, deep learning, and large language models [Guo+18; Mos+20; Gup+17]. The problem has enjoyed attention both from the theory community as well as the application side, with many works spanning the bridge between theory and practice.

Formally, a coflow instance is given by some bipartite set of vertices $V := U_1 \cup U_2$ and a set of coflows E_1, \dots, E_n , where each coflow E_j is a subset of bipartite edges on V , possibly containing duplicates. Additionally, each coflow E_j has some associated weight $\omega_j \in \mathbb{R}^+$. This models for example a set of input and output ports in a shared network, where each edge inside a coflow represents some data transmission requirement. During each discrete step in time, we are allowed to schedule a set of edges on the graph for which no vertex has more than one adjacent edge, so a matching. This represents the requirement that ports only send to and receive from one other port during each discrete step in time. A coflow finishes at time-step t if all of its edges have been scheduled on or before time t , with at least one edge being scheduled during t . We call C_j^* the finishing time of coflow E_j and wish to minimize the weighted sum of completion times $\sum_{j \in [n]} \omega_j C_j^*$.

Coflow Scheduling was first introduced by Chowdhury et al. [CZS14], though the closely related problem of scheduling on network switches has been studied earlier under different names by various authors [BGW91; Gup96]. Coflow Scheduling can be seen as an extension with a

*University of Southern Denmark, Odense, Denmark. Supported by Dutch Research Council (NWO) project “The Twilight Zone of Efficiency: Optimality of Quasi-Polynomial Time Algorithms” [grant number OCEN.W.21.268]

[†]Technical University of Munich, Munich, Germany. Supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - GRK 2201/2 - Projektnummer 277991500

combinatorial structure of a problem called Concurrent Open Shop Scheduling (COSS) with preemption. In COSS, there is a set of machines and jobs, where every job has some demand on each machine which can be fulfilled concurrently, and jobs finish when they are completed on every machine. For COSS, several 2-approximation algorithms are known [GKP07; LLP07; Mas+10] and the problem is known to be NP-hard to approximate within $2 - \epsilon$, for any $\epsilon > 0$ [SS13]. This hardness result extends to Coflow Scheduling. On the side of approximation algorithms for Coflow Scheduling, there is still a gap to the lower bound. For the case of no release dates, multiple authors have given 4-approximation algorithms [Aga+18; Ahm+17; Fuk22], which extend to 5-approximations in the case of release dates. Fukunaga [Fuk22] shows that in the case of release dates the integrality gap of the linear program used in the algorithm is at most 4, though his proof is non-constructive. Several authors have claimed $(2 + \epsilon)$ -approximations, but all were later shown to be incorrect, see [IP18; Ahm+17] for discussion. For the setting in which the simultaneously schedulable flows have to be independent sets of a matroid instead of matchings, Im et al. [Im+19] provided an algorithm with a $(2 + \epsilon)$ -approximation guarantee. Khuller et al. [Khu+19] showed a framework which provides guarantees in an online setting using offline approximation algorithms, leading to a 12-approximation for online Coflow Scheduling. Extensions to general graphs [JKR17] and so called path-based Coflow Scheduling [Eck+20] have been studied.

Whether the 4- and respectively 5-approximation can be improved has been a major open question raised by most previous works [Aga+18; Fuk22], especially in light of the $2 - \epsilon$ lower bound. We address this question and show that both bounds can be beaten, with a tight result in an asymptotic setting.

1.1 Our Contribution

We present the first polynomial time algorithm which achieves a better than 4-approximation for Coflow Scheduling without release dates and the first algorithm which achieves a better than 5-approximation with release dates. More specifically, we show the following theorems.

Theorem 1. *There is a polynomial time algorithm achieving a $\frac{140}{41} (< 3.415)$ -approximation for Coflow Scheduling without release dates.*

Theorem 2. *There is a polynomial time algorithm achieving a 4.36-approximation for Coflow Scheduling with release dates.*

Using a more technical construction, the guarantee of Theorem 1 can be slightly improved, see Section 6.6 for details. We additionally prove that in a certain asymptotic setting, roughly when most coflows have large finishing times in any optimum solution, we can achieve a $(2 + \epsilon)$ -approximation, which is optimal. This result holds even in the case with release dates.

Theorem 3. *For any $\epsilon > 0$, there exists an $\hat{\epsilon} > 0$ such that there is a $(2 + \epsilon)$ -approximation algorithm for all coflow instances \mathcal{I} fulfilling*

$$\sum_{j \in [|\mathcal{I}|]} \omega_j \leq \hat{\epsilon} \cdot \text{OPT}(\mathcal{I}).$$

Note that using a framework by Khuller et al. [Khu+19], any improvement in the approximation ratio for Coflow Scheduling without release dates directly gives an improvement for the best known approximation for the online setting of Coflow Scheduling. The following table provides an overview over the state of the art of approximation algorithms for various variants of Coflow Scheduling and our respective improvements.

Case	Best Known		This work
No Release Dates	4	[*]	3.415
Release Dates	5	[*]	4.36
Release Dates (integrality gap)	4	[Fuk22]	3.893
Asymptotic + No Release Dates	4	[*]	$2 + \epsilon$
Asymptotic + Release Dates	5	[*]	$2 + \epsilon$
Online	12	[Khu+19]	11.415

Table 1: Best known previous approximations and our results. The sources marked [*] are [Ahm+17; Aga+18; Fuk22].

The main technical contributions are a novel allocation and rounding scheme for the individual edges of each coflow, inspired by techniques used in proving the Beck-Fiala Theorem from discrepancy theory and a framework for establishing the approximation ratio for combinations of several such edge scheduling algorithms.

Our algorithm follows a two phase approach. This has been done either implicitly or explicitly in most approaches found in the literature. In the first phase, for each coflow and its associated flows, deadlines are determined through an LP based approach combined with a randomized rounding procedure. These deadlines are equipped with a special structure which we exploit in the algorithm. They specifically provide a 2-approximation cost guarantee with respect to some optimum solution. In the second phase, the goal is to find a valid allocation for the coflows, or more precisely for the individual edges of each coflow, to time-slots. In previous works, this was achieved by a simple greedy allocation procedure, which in the case without release dates achieves a deadline violation of at most a factor 2 for each coflow, yielding a 4-approximation. We use a combination of two algorithms, both the greedy allocation rule and a novel iterated rounding scheme. The greedy allocation performs well for small deadlines, but converges to a 2-approximation for larger deadlines, while the rounding procedure can schedule large deadlines arbitrarily well, but has worse results for small ones. Note that these factors only capture the completion time delay and do not take into account the additional loss of factor 2 from the deadline construction. By running both algorithms and picking the better result, we are able to improve upon the factor 4.

To show the improved approximation ratio, we establish a general framework which can be used to bound the coflow scheduling approximation ratio for any collection of edge allocation algorithms.

1.2 Organization

The next chapter introduces important notation and discusses results from LP and graph theory which form an important part of several subroutines. Section 3 provides a complete overview of the entire algorithm and the most important theorems for Coflow Scheduling without release dates. Full proofs and additional details can be found in the following Section 4. Section 5 proves the $2 + \epsilon$ guarantee in the asymptotic case. Appendix 6 provides additional details and discusses some further related theory, such as results regarding the structure and complexity of the used LP, extending the algorithm to release dates and high edge multiplicities, the improved LP integrality gap, and shows how to achieve a slight improvement over the approximation guarantee from Theorem 7.

2 Preliminaries

For $n \in \mathbb{N}$ we define $[n] := \{1, \dots, n\}$. Graphs $G = (V, E)$ are defined by a set of vertices V and edges $E \subseteq V \times V$. We slightly abuse notation and allow E to contain multiple copies of the same edge. We use $\Delta(G)$ to denote the maximum degree of a graph. For some set of edges E , $\Delta(E)$ refers to the maximum degree of the canonical graph induced by this edge set.

A coflow instance is given by a collection of bipartite edge multi-sets E_1, \dots, E_n on some set of vertices, together with weights $\omega_1, \dots, \omega_n \in \mathbb{R}_+$. We usually only refer to the edge sets and let the vertices of the underlying graph be implicitly described by them. We define $E := \cup_{j \in [n]} E_j$. In the case of release dates, for every coflow $j \in [n]$ there is a release date $r_j \in \mathbb{N}$. A release date r_j means that edges from coflow E_j can be scheduled the earliest in time slot $r_j + 1$. We use the term flow to refer to a collection of identical edges within one coflow, so essentially an edge with its multiplicity. In the main body of this work we assume that edge multiplicities are encoded in this explicit way, meaning that multiplicities are represented by multiple copies. We discuss the case where the multiplicities are instead encoded as an integer in Section 6.4. A valid schedule is a mapping of all edges to time slots, such that in every time slot the assigned edges form a matching. In the case of release dates, no edge from any coflow is allowed to be scheduled in a time slot smaller than or equal to the respective release date. The finishing time of a coflow is the latest time slot to which one of its edges is assigned. We call C_j^* the finishing time of coflow E_j and wish to minimize the weighted sum of completion times $\sum_{j \in [n]} \omega_j C_j^*$.

As we use an iterated LP rounding scheme, we give a brief overview of the most important relevant theory here. For more details see for example [Ban14; Sch86]. Let $A \in \mathbb{R}^{m \times n}$ be a matrix and $b \in \mathbb{R}^m, c \in \mathbb{R}^n$ be vectors. We consider the polytope $\mathcal{P} = \{x \in [0, 1] \mid Ax \leq b\}$ and the associated LP $\min_{x \in \mathcal{P}} c^T x$. From standard LP theory we know that there always exists an LP optimum solution at a vertex of \mathcal{P} which can be found in strongly polynomial time, as long as all values in A are polynomially bounded [Tar86]. One key fact we use is that additionally if $m < n$, one can find such a vertex solution in which at least $n - m$ entries are in $\{0, 1\}$. This theorem can be extended to work when every entry x_i is constrained to some interval $[0, n_i]$, for $n_i \in \mathbb{N}$.

The Coflow Scheduling problem is closely related to several well studied graph and coloring problems. As we use some of these results directly and implicitly in our work, we briefly review them here. Given some set of edges E on a bipartite graph $G = (V, E)$, the question whether they can be partitioned into some number of matchings k is equivalent to asking whether there is a proper k -edge-coloring of E . Clearly at least $\Delta(G)$, i.e. the maximum degree of the graph, colors are needed. König's Theorem, and to an extent also Vizing's Theorem, give a strong result for bipartite graphs:

Theorem 4 ([Kön16]). *Any bipartite graph G can be properly edge-colored with $\Delta(G)$ colors.*

As a set of matchings is equivalent to a set of scheduled flows in the coflow setting, the question whether a set of edges can be scheduled during some collection of time points is equivalently answered by this. Any set of flows for which the induced graph has maximum degree d can be scheduled within d time slots. This result is essential to our analysis, as it shows that degree bounds for some set of selected flows are sufficient to ensure their schedulability. Note that the proof of Theorem 4 can be done in a constructive way, leading to a polynomial time algorithm producing such an allocation. This algorithm can be extended to work even for edges with possibly superpolynomial multiplicities, for details see [QSZ15].

3 Algorithmic Framework

This section provides a complete overview of the algorithmic framework used to establish the improved approximation ratios. We focus on the case without release dates here, the necessary modifications for release dates are described in Section 6.3.

3.1 Coflow Deadlines

Given some coflow instance, we aim to determine a deadline for each of its coflows. These deadlines might not necessarily be strict in the sense that constructed schedules have to adhere to them, but they are rather used to both guide edge allocation procedures and to then bound their resulting costs. Most existing coflow approximation algorithms use a similar strategy.

We take a structural approach, where we first define structure which we want our deadlines to obey and then describe how such deadlines can be found. We capture the structural constraints in the following LP. It has been implicitly used in analysis by [Im+19; Fuk22] and others. Let $C_1 \leq C_2 \leq \dots \leq C_n$ be deadlines for the coflows and for easier notation define $C_0 := 0$.

$$\sum_{s \in [n]} x_{s,e} = 1 \quad \forall e \in E \quad (I)$$

$$\sum_{e: v \in e} x_{s,e} \leq C_s - C_{s-1} \quad \forall s \in [n], \forall v \in V \quad (II) \quad (\text{LP I})$$

$$\begin{aligned} x_{s,e} &= 0 & \forall j \in [n], \forall e \in E_j, \forall s > j \\ x_{s,e} &\geq 0 \end{aligned} \quad (III)$$

Instead of enforcing the necessary constraints for Coflow Scheduling for each individual time slot, **LP I** groups the slots into blocks in between the coflow deadlines. The variable $x_{s,e}$ describes the assignment of edge e to block s . Constraint (I) ensures that every edge from every coflow is fully scheduled. The block between C_{s-1} and C_s has size $C_s - C_{s-1}$, so constraint (II) ensures that in every such block for every vertex the amount of adjacent edges does not exceed the block size. Constraint (III) forces edges to be zero for blocks after the respective deadline. The step from individual time slot degree bounds to block degree bounds is justified by König's Theorem (Theorem 4), as it guarantees that any bipartite graph with some maximum degree Δ can be decomposed into Δ matchings. This is equivalent to saying that any set of bipartite edges E for which the induced graph has maximum degree Δ can be scheduled in Δ time slots.

Assuming the deadlines $C_1 \leq C_2 \leq \dots \leq C_n$ are set as the coflow finishing times from some optimal schedule for the underlying coflow instance, the edge assignment directly induces a feasible point inside **LP I**. For each edge e which is scheduled in some time slot t in the optimal schedule, set $x_{s,e} := 1$, for s such that $t \in (C_{s-1}, C_s]$. Set all other variables to 0. As by definition of a valid solution, each edge is scheduled before its coflow's deadline, constraint (I) is fulfilled and constraint (III) cannot be violated. As the edges in each time slot form a matching, constraint (II) is also fulfilled.

Conversely, an integral solution to the LP corresponds to a valid solution for Coflow Scheduling. However, in order to be able to solve the LP in polynomial time, we cannot enforce integrality. Hence the constraints only enforce that the variable assignment corresponds to a fractional matching. There are instances and deadlines for which **LP I** is feasible, but no feasible integral point and therefore also no feasible integral schedule exists. In fact, determining whether an integral points exists is an NP -hard problem, for details see Section 6.1 in the appendix.

Finding some set of deadlines for which [LP I](#) is feasible is easy, as one can simply choose large enough values to guarantee feasibility. However, for the purpose of constructing good approximation algorithms for Coflow Scheduling, we require that the deadlines fulfill some cost guarantees with respect to an optimal coflow schedule.

There is an LP based approach which returns deadlines for which [LP I](#) is feasible and certain strong guarantees hold. This technique has been used by [\[Im+19; Fuk22\]](#) and others. They use a randomized rounding scheme on another LP formulation to determine integral deadlines C'_1, \dots, C'_n . We slightly modify their algorithm and leave out the final step in which they round up the deadlines and obtain C_1, \dots, C_n . These deadlines are thus potentially fractional. By slightly modifying their proof, the following bound can be shown.

Lemma 5 ([\[Im+19\]](#)). *There is a polynomial time randomized algorithm determining deadlines C_1, \dots, C_n for which [LP I](#) is feasible and for which the following cost bound holds:*

$$\sum_{j \in [n]} \omega_j \mathbb{E}[C_j] \leq 2 \cdot \text{OPT} - \sum_{j \in [n]} \omega_j$$

More details about the procedure used by [\[Im+19\]](#) to determine such deadlines can be found in the appendix in Section 6.2. They only implicitly work with [LP I](#), so we provide additional details on the connection. Note that the procedure can be de-randomized to obtain a fully deterministic algorithm.

The multiplicative factor of 2 in Lemma 5 is optimal assuming $\mathbb{P} \neq \text{NP}$. This follows from the factor $(2 - \epsilon)$ -approximation hardness of Concurrent Open Shop Scheduling, as Coflow Scheduling can be seen as a generalization of this problem [\[SS13\]](#).

3.2 Integral Edge Assignments with Guarantees

Let C_1, \dots, C_n be deadlines for which [LP I](#) is feasible and Lemma 5 holds. Using the result of the lemma, we immediately obtain that if we are able to find an allocation such that all edges from each coflow are scheduled by their respective deadlines, we have achieved a 2-approximation for Coflow Scheduling. In the same way, if for some $\alpha \geq 1$ we are able to schedule each coflow j by time $\alpha \cdot C_j$, we obtain a $2 \cdot \alpha$ approximation algorithm.

We analyze two edge allocation algorithms which provide different guarantees for the finishing times of the coflows. The first algorithm Greedy is a simple greedy allocation scheme. This procedure was used by previous authors to derive 4-approximation algorithms for Coflow Scheduling. Let $\text{Greedy}(C_j)$ denote the finishing time of coflow E_j in the schedule produced by Greedy.

Lemma 6. *For given deadlines C_1, \dots, C_n for which [LP I](#) is feasible there is an algorithm Greedy returning a valid coflow schedule such that the following holds.*

$$\text{Greedy}(C_j) \leq 2C_j - 1$$

The second algorithm CBF^τ is a novel allocation scheme using a form of iterated rounding inspired by the Beck-Fiala Theorem from discrepancy theory [\[BF81\]](#). The algorithm is parameterized by $\tau \in \mathbb{N}_{\geq 2}$ and allocates the coflow deadlines to blocks, where each block's size is some integer multiple of τ . An edge assignment is then determined which only slightly violates the size of each block. This leads to the following completion time guarantees.

Theorem 7. *For given deadlines C_1, \dots, C_n for which [LP I](#) is feasible, weights $\omega_1, \dots, \omega_n$, and a parameter $\tau \in \mathbb{N}_{\geq 2}$, there is an algorithm CBF^τ returning a valid coflow schedule such that the following holds.*

$$\sum_{j \in [n]} \omega_j \cdot \text{CBF}^\tau(C_j) \leq \sum_{j \in [n]} \omega_j \left(\frac{\tau + 2}{\tau} C_j + \frac{\tau}{2} + 2.5 - \frac{2}{\tau} \right)$$

Note that the approximation guarantees of both algorithms are quite different. Greedy achieves rather strong approximation for small deadlines, while for large deadlines, through an appropriate choice of τ , CBF^τ gives good guarantees. In fact, under certain assumptions on the provided deadlines, CBF^τ can achieve approximations arbitrarily close to the optimum of 1, see Section 5.

Our coflow algorithm aims to achieve guarantees for both small and large deadlines by combining both algorithms in some way. The procedure is straightforward. It obtains the deadlines C_1, \dots, C_n through the procedure explained in Section 3.1 and then runs both Greedy and CBF^6 independently, returning the schedule with lower cost. Analyzing the cost of the returned solution requires some care, as we need a uniform bound over any possible input instance.

3.3 Combining Algorithmic Guarantees

By the definition of the coflow algorithm, for each possible instance \mathcal{I} , its cost $C_{\text{ALG}}(\mathcal{I})$ is given as the minimum of the costs $C_G(\mathcal{I})$ and $C_{\text{CBF}}(\mathcal{I})$ of the Greedy and respectively CBF^6 algorithm.

We show a general proof framework for such algorithms which provide deadline guarantees, which gives sharp bounds for taking minimums over several algorithms' costs. The derivation is not difficult, but the framework offers a surprisingly simple and strong method to establish bounds for large classes of algorithms. It only requires bounds for the delay guarantees of the algorithms, which are usually relatively simple to establish.

For this purpose, let C_1, \dots, C_n be deadlines for which **LP 1** is feasible and let $\text{ALG}_1, \dots, \text{ALG}_k$ be algorithms producing valid coflow schedules from such deadlines, with $\text{ALG}_i(C_j)$ being the finishing time of coflow E_j in the schedule produced by ALG_i . For $j \in [k]$, let f_j be a function capturing a bound on the maximum weighted deadline delay of ALG_j . This means that f_j is such that $\sum_{j \in [n]} \omega_j \cdot \text{ALG}_j(C_j) \leq \sum_{j \in [n]} \omega_j \cdot f_j(C_j)$. Such functions might stem from bounds on individual deadlines like in the case of Greedy, but can also come from bounds which are already given as a weighted sum over all deadlines like for CBF^τ .

Lemma 8. *Let $\lambda_1, \dots, \lambda_k \geq 0$ with $\sum_{i \in [k]} \lambda_i = 1$ and $\alpha \in \mathbb{R}^+$. If for all $x \geq 1$*

$$\sum_{i \in [k]} \lambda_i f_i(x) \leq \alpha(x + 1),$$

then for all coflow instances \mathcal{I} :

$$C_{\text{ALG}}(\mathcal{I}) = \min\{C_{\text{ALG}_1}(\mathcal{I}), \dots, C_{\text{ALG}_k}(\mathcal{I})\} \leq 2\alpha \cdot \text{OPT}(\mathcal{I})$$

Proof. **TOPROVE 0** □

3.4 Main Theorem

Using the previous lemmata and theorems, we prove Theorem 1. The proof follows by application of the framework from Theorem 8 to selected edge allocation algorithms.

Theorem 1. *There is a polynomial time algorithm achieving a $\frac{140}{41} (< 3.415)$ -approximation for Coflow Scheduling without release dates.*

Proof. **TOPROVE 1** □

4 Integral Edge Assignments with Guarantees

In this section we introduce and analyze algorithms which allocate edges of coflows to time-slots. They work on coflow deadlines fulfilling certain structural properties and their goal is to provide feasible schedules together with guarantees on the average delay each coflow experiences. Similar strategies are also used in most of the previous 4-approximations for Coflow Scheduling. We introduce the algorithms Greedy and CBF^τ and show their guarantees in Lemma 6 and Theorem 7.

4.1 Greedy Scheduling

We start by introducing and analyzing a greedy allocation algorithm Greedy, which is one of the edge allocation procedures used in previous works to achieve a 4-approximation. Let $C_1 \leq C_2 \leq \dots \leq C_n$ be deadlines for which LP I is feasible. Greedy schedules all coflows consecutively, starting with E_1 up to E_n . Each edge is simply scheduled in a work-conserving way, meaning that it is scheduled in the earliest possible time-slot in which both its vertices are free. By doing this for all edges, a schedule is obtained. For $j \in [n]$, let $\text{Greedy}(C_j)$ be the finishing time of coflow E_j in the schedule obtained from this procedure.

Lemma 6. *For given deadlines C_1, \dots, C_n for which LP I is feasible there is an algorithm Greedy returning a valid coflow schedule such that the following holds.*

$$\text{Greedy}(C_j) \leq 2C_j - 1$$

Proof. **TOPROVE 2** □

Greedy provides a strict deadline guarantee for each coflow, meaning that in the schedule produced every coflow finishes the latest at the provided bound. This also implies that the same guarantee holds when taking weighted sums over the finishing times. Note that $2C_j - 1 = (2 - \frac{1}{C_j})C_j$, so for small C_j this gives a tangible improvement over the factor 2.

4.2 Iterated Rounding using Beck-Fiala

This section gives a proof of Theorem 7. We start by providing a full description of the algorithm, then we give a preliminary analysis and subsequently strengthen the guarantee through further refinements.

Procedure Idea

Given some deadlines $C_1 \leq C_2 \leq \dots \leq C_n$ for which LP I is feasible, the core idea is to round these deadlines to the next integer multiple of some parameter $\tau \in \mathbb{N}_{\geq 2}$ and to then form blocks between consecutive rounded deadlines. With these blocks and associated deadlines, we show that it is possible to allocate all edges to blocks while only violating the block size by a small additive constant. Given such an allocation, using the guarantee provided by König's Theorem (Theorem 4) there exists a feasible schedule containing all assigned edges within the maximum vertex load of each block. Through the rounding and the increase in blocks' sizes the finishing times of the coflows are delayed with respect to their deadlines. We are however able to show strong bounds on this delay.

We call this algorithm CBF^τ due to its close association with the proof of the Beck-Fiala Theorem [BF81].

Edge-to-Block Allocation LP

We use a rounding technique inspired by the proof of the Beck-Fiala Theorem. Let $\tau \in \mathbb{N}_{\geq 2}$ be a fixed constant. For $j \in [n]$ let \bar{C}_j be the deadline C_j rounded up to the next integer multiple of τ . The blocks' sizes are defined by the distance between two non-equal consecutive deadlines. So for $\bar{C}_j \neq \bar{C}_{j-1}$, block j has size $\bar{C}_j - \bar{C}_{j-1}$. We define the following LP, which models the allocation of coflow edges to blocks. One can assume without loss of generality that all rounded deadlines are distinct and that there are n of them, as coflows whose rounded deadlines are equal can be joined in this step.

$$\sum_{b \in [n]} x_{e,b} = 1 \quad \forall e \in E \quad (I)$$

$$\sum_{e: v \in e} x_{e,b} \leq \bar{C}_b - \bar{C}_{b-1} \quad \forall v \in V, \forall b \in [n] \quad (II) \quad (\text{LP CBF})$$

$$\begin{aligned} x_{e,b} &= 0 & \forall j \in [n], \forall e \in E_j, \forall b > j \\ x_{e,b} &\geq 0 \end{aligned}$$

The structure of **LP CBF** is identical to **LP I**, though the special form of the rounded deadlines induces some additional properties. By definition $\bar{C}_j \geq C_j$ for all $j \in [n]$. Additionally, with $\bar{C}_0 = 0$, for each $b \in [n]$ we have $\bar{C}_b - \bar{C}_{b-1} = k_b \cdot \tau$, for some $k_b \in \mathbb{N}$. We identify each edge and vertex in the coflow instance with the respective variable in the LP and use both terms interchangeably.

We claim that feasibility of **LP I** for C_1, \dots, C_n directly implies feasibility of **LP CBF** for $\bar{C}_1, \dots, \bar{C}_n$. For **LP I**, increasing the value of any deadline without violating the total order can only increase the feasible region, as the equal zero constraints get less restrictive and any possible excess assignment can be shifted between the two adjacent blocks whose size changes. Therefore, as all deadlines can only increase, **LP CBF** also has to be feasible.

LP Rounding

We describe a procedure which finds an integral edge-to-block assignment violating the block size constraint (II) by a constant amount. To achieve this, we start with an initial solution to the LP and then successively refine and resolve the LP, until we have obtained an integral solution fulfilling certain strong properties.

From now on we assume that we started with deadlines C_1, \dots, C_n for which **LP I** is feasible, so we know that for $\bar{C}_1, \dots, \bar{C}_n$ **LP CBF** is feasible. After obtaining an initial solution to **LP CBF**, we take two steps. We first fix all integral edges and remove their respective variables from the LP and modify the right hand side of (II) accordingly and then we delete all constraints from (II) with at most $k - 1$ fractional variables remaining, for some fixed number $k \in \mathbb{N}$. This corresponds to dropping the degree constraint on the respective vertex if at most $k - 1$ adjacent edges are still fractional. Let \mathcal{E}_b be the set of all fractional edges contained in block b and let \mathcal{V}_b be the vertices in block b with at least k fractional adjacent edges. Let $S_{v,b}$ be the set of already fixed edges adjacent to v in block b . This gives rise to the following resulting LP:

$$\sum_{b \in [n]} x_{e,b} = 1 \quad \forall e \in \bigcup_{b \in [n]} \mathcal{E}_b \quad (I)$$

$$\sum_{e \in \mathcal{E}_b: v \in e} x_{e,b} \leq k_b \cdot \tau - |S_{v,b}| \quad \forall b \in [n], v \in \mathcal{V}_b \quad (II)$$

$$x_{e,b} \geq 0 \quad \forall b \in [n], e \in \mathcal{E}_b$$

If we can show that this LP always contains strictly more variables than it contains constraints in (I) and (II), by considering a basic feasible solution, we obtain at least one more integral variable, so repeating the fixing variables and removing constraints step leads to at least one more fixed variable. Therefore in a polynomial number of steps the procedure must terminate and we obtain an integral solution. The step in which we drop constraints means that this integral solution is most likely not feasible for the original LP, but we later show that the amount of violation cannot be very large, which yields the desired approximation behaviour.

Constraints and Variables

We want to show that the number of constraints is strictly smaller than the number of variables. The total number of variables in the LP is equal to $\sum_{b \in [n]} |\mathcal{E}_b|$. The number of constraints in (I) \cup (II) is equal to $|\bigcup_{b \in [n]} \mathcal{E}_b| + \sum_{b \in [n]} |\mathcal{V}_b|$. We show two bounds which enable us to establish the desired inequality.

Lemma 9. *For all $b \in [n]$: $|\mathcal{V}_b| \leq \frac{2}{k} |\mathcal{E}_b|$*

Proof. **TOPROVE 3** □

Lemma 10. *For all $b \in [n]$: $|\bigcup_{b \in [n]} \mathcal{E}_b| \leq \frac{1}{2} \sum_{b \in [n]} |\mathcal{E}_b|$*

Proof. **TOPROVE 4** □

Combining the two lemmata, we obtain:

$$|\text{Cons}| = \left| \bigcup_{b \in [n]} \mathcal{E}_b \right| + \sum_{b \in [n]} |\mathcal{V}_b| \leq \frac{1}{2} \sum_{b \in [n]} |\mathcal{E}_b| + \frac{2}{k} \sum_{b \in [n]} |\mathcal{E}_b| = \left(\frac{1}{2} + \frac{2}{k} \right) |\text{Vars}|$$

So for all $k > 4$ a strict inequality follows. For $k = 4$ we have the inequality $|\text{Cons}| \leq |\text{Vars}|$. We can however still achieve a strict inequality for this case by slightly modifying the LP. In its current form, the LP is given without an objective function. We can thus remove one constraint from (II) and shift it to the objective function instead. This reduces the number of constraints by one without changing the number of variables. If $b, v \in \mathcal{V}_b$ are the parameters corresponding to the chosen inequality, the added objective function is $\min \sum_{e \in \mathcal{E}_b : v \in e} x_{e,b}$. From the minimization objective it follows that feasible optimal points of the modified LP are feasible for the original LP, as the removed constraint cannot be violated.

Delay Bound

Looking at the integral assignment, we can show that the violation of constraints of **LP CBF** is small. Note that the following statement only requires integrality of the deadlines and not the special structure of the rounded deadlines.

Lemma 11. *Given integral deadlines C_1, \dots, C_n for which **LP CBF** is feasible and a parameter $\tau \in \mathbb{N}_{\geq 2}$, in polynomial time we can find an integral point such that:*

- a) *All constraints (II) in **LP CBF** are exceeded by at most 2.*
- b) *All other constraints in **LP CBF** are fulfilled.*

Proof. **TOPROVE 5** □

Using this result, we can show an upper bound on the maximum delay for each deadline when applying CBF^τ . In total, we obtain the following lemma, which at this point is slightly weaker than required for Theorem 7, as there is an additive constant of $\tau + 2$ instead of $\tau/2 + 2.5 - \frac{2}{\tau}$. Nevertheless, the theorem in this form would already be sufficient to gain a significant improvement over the factor 4-approximation. Like in the case of Greedy, $\text{CBF}^\tau(C_j)$ is the finishing time of coflow E_j in the schedule created by CBF^τ .

Lemma 12. *For given deadlines C_1, \dots, C_n for which [LP I](#) is feasible and a parameter $\tau \in \mathbb{N}_{\geq 2}$, there is an algorithm CBF^τ returning a valid coflow schedule such that the following holds for all $j \in [n]$.*

$$\text{CBF}^\tau(C_j) \leq \frac{\tau + 2}{\tau} C_j + \tau + 2$$

Proof. [TOPROVE 6](#) □

Reducing the Additive Constant

The additive constant τ in Lemma 12 assumes the worst case for each deadline, meaning that every deadline gets shifted from the very start to the very end of a block. We show that an averaging argument can be used to reduce the average amount of shift to $\tau/2 + \frac{1}{2} - \frac{2}{\tau}$. This requires that we show the bound across the weighted sum over all deadlines, unlike the previous proofs which established hard upper bounds for each individual deadline.

For $\lambda \in \mathbb{N}$, we consider a variant CBF_λ^τ of CBF^τ where an additional first block of fixed size λ is inserted. This is equivalent to rounding the deadlines to the next larger term in the sequence $\{\lambda + i \cdot \tau\}_{i \in \mathbb{N}}$. Note that by simple modification of the arguments, the feasibility statements for [LP CBF](#) still apply. Lemma 11 is thus also applicable.

This change to the deadline rounding step can change the finishing time of deadlines in our procedure. On the one hand, the delay of some deadlines might increase, as the last time slot of their respective blocks gets increased. On the other hand, the delay of some deadlines might decrease, as they now get included in an earlier block. We show the following.

Theorem 7. *For given deadlines C_1, \dots, C_n for which [LP I](#) is feasible, weights $\omega_1, \dots, \omega_n$, and a parameter $\tau \in \mathbb{N}_{\geq 2}$, there is an algorithm CBF^τ returning a valid coflow schedule such that the following holds.*

$$\sum_{j \in [n]} \omega_j \cdot \text{CBF}^\tau(C_j) \leq \sum_{j \in [n]} \omega_j \left(\frac{\tau + 2}{\tau} C_j + \frac{\tau}{2} + 2.5 - \frac{2}{\tau} \right)$$

Proof. [TOPROVE 7](#) □

5 Asymptotic $2 + \epsilon$ Approximation

Theorem 1 establishes that there is an algorithm returning a 3.415-approximation for any given coflow input instance. In this section we show a stronger, asymptotically optimal, approximation for input instances with a certain structure. This result does not depend on the approximation framework but rather follows directly from the bounds established in Lemma 17. Note that to show $(2 - \epsilon)$ -approximation hardness in [SS13], they construct a sequence of instances for which the ratio between the sum over all weights and the optimum grows arbitrarily large, which shows that the asymptotic result in Theorem 3 is essentially optimal.

Theorem 3. *For any $\epsilon > 0$, there exists an $\hat{\epsilon} > 0$ such that there is a $(2 + \epsilon)$ -approximation algorithm for all coflow instances \mathcal{I} fulfilling*

$$\sum_{j \in [|\mathcal{I}|]} \omega_j \leq \hat{\epsilon} \cdot \text{OPT}(\mathcal{I}).$$

Proof. **TOPROVE 8** □

Note that while the requirements in Theorem 3 are rather technical, it implies several strong results for natural classes of coflow instances, such as instances where all coflows have large maximum degree.

Corollary 13. *For any $\epsilon > 0$, there is $D \in \mathbb{N}$ such that there is a $(2 + \epsilon)$ -approximation algorithm for Coflow Scheduling without release dates for instances \mathcal{I} fulfilling*

$$\forall E_j \in \mathcal{I} : \Delta(E_j) \geq D.$$

Proof. **TOPROVE 9** □

References

- [Aga+18] Saksham Agarwal, Shijin Rajakrishnan, Akshay Narayan, Rachit Agarwal, David Shmoys, and Amin Vahdat. “Sincronia: near-optimal network design for coflows”. In: *Proceedings of SIGCOMM*. ACM, 2018, pp. 16–29. DOI: [10.1145/3230543.3230569](https://doi.org/10.1145/3230543.3230569).
- [AH00] Ron Aharoni and Penny Haxell. “Hall’s theorem for hypergraphs”. In: *Journal of Graph Theory* 35.2 (2000), pp. 83–88. DOI: [10.1002/1097-0118\(200010\)35:2<83::aid-jgt2>3.0.co;2-v](https://doi.org/10.1002/1097-0118(200010)35:2<83::aid-jgt2>3.0.co;2-v).
- [Ahm+17] Saba Ahmadi, Samir Khuller, Manish Purohit, and Sheng Yang. “On Scheduling Coflows”. In: *Proceedings of IPCO*. Vol. 10328. Springer International Publishing, 2017, pp. 13–24. DOI: [10.1007/978-3-319-59250-3_2](https://doi.org/10.1007/978-3-319-59250-3_2).
- [Ban14] Nikhil Bansal. “New Developments in Iterated Rounding”. In: *Proceedings of FSTTCS*. Vol. 29. LIPIcs. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014, pp. 1–10. DOI: [10.4230/LIPIcs.FSTTCS.2014.1](https://doi.org/10.4230/LIPIcs.FSTTCS.2014.1).
- [BC01] Maurizio Bonuccelli and Maria Claudia Clo. “Scheduling of real-time messages in optical broadcast-and-select networks”. In: *IEEE/ACM Transactions on Networking* 9.5 (2001), pp. 541–552. DOI: [10.1109/90.958324](https://doi.org/10.1109/90.958324).
- [BF81] József Beck and Tibor Fiala. ““Integer-making” theorems”. In: *Discrete Applied Mathematics* 3.1 (1981), pp. 1–8. DOI: [10.1016/0166-218x\(81\)90022-6](https://doi.org/10.1016/0166-218x(81)90022-6).
- [BGW91] Maurizio Bonuccelli, Inder Gopal, and Chak-Kuen Wong. “Incremental time-slot assignment in SS/TDMA satellite systems”. In: *IEEE Transactions on Communications* 39.7 (1991), pp. 1147–1156. DOI: [10.1109/26.87220](https://doi.org/10.1109/26.87220).
- [CZS14] Mosharaf Chowdhury, Yuan Zhong, and Ion Stoica. “Efficient coflow scheduling with Varys”. In: *Proceedings of SIGCOMM*. ACM, 2014, pp. 443–454. DOI: [10.1145/2619239.2626315](https://doi.org/10.1145/2619239.2626315).
- [DG08] Jeffrey Dean and Sanjay Ghemawat. “MapReduce: simplified data processing on large clusters”. In: *Communications of the ACM* 51.1 (2008), pp. 107–113. DOI: [10.1145/1327452.1327492](https://doi.org/10.1145/1327452.1327492).
- [Eck+20] Alexander Eckl, Luisa Peter, Maximilian Schiffer, and Susanne Albers. “Minimization of Weighted Completion Times in Path-based Coflow Scheduling”. In: *arXiv:1911.13085 [cs]* (2020). DOI: [10.48550/arXiv.1911.13085](https://doi.org/10.48550/arXiv.1911.13085).

- [EIS75] Shimon Even, Alon Itai, and Adi Shamir. “On the complexity of time table and multi-commodity flow problems”. In: *Proceedings of FOCS*. 1975, pp. 184–193. DOI: [10.1109/SFCS.1975.21](https://doi.org/10.1109/SFCS.1975.21).
- [Fuk22] Takuro Fukunaga. “Integrality Gap of Time-Indexed Linear Programming Relaxation for Coflow Scheduling”. In: *Proceedings of APPROX/RANDOM*. Vol. 245. LIPIcs. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 36:1–36:13. DOI: [10.4230/LIPIcs.APPROX/RANDOM.2022.36](https://doi.org/10.4230/LIPIcs.APPROX/RANDOM.2022.36).
- [GKP07] Naveen Garg, Amit Kumar, and Vinayaka Pandit. “Order Scheduling Models: Hardness and Algorithms”. In: *Proceedings of FSTTCS*. Vol. 4855. Springer Berlin Heidelberg, 2007, pp. 96–107. DOI: [10.1007/978-3-540-77050-3_8](https://doi.org/10.1007/978-3-540-77050-3_8).
- [Guo+18] Runxin Guo, Yi Zhao, Quan Zou, Xiaodong Fang, and Shaoliang Peng. “Bioinformatics applications on Apache Spark”. In: *GigaScience* (2018). DOI: [10.1093/gigascience/giy098](https://doi.org/10.1093/gigascience/giy098).
- [Gup+17] Anand Gupta, Hardeo Kumar Thakur, Ritvik Shrivastava, Pulkit Kumar, and Sreyashi Nag. “A Big Data Analysis Framework Using Apache Spark and Deep Learning”. In: *Proceedings of ICDMW*. IEEE, 2017, pp. 9–16. DOI: [10.1109/icdmw.2017.9](https://doi.org/10.1109/icdmw.2017.9).
- [Gup96] Pankaj Gupta. “Scheduling in input queued switches: A survey”. In: *Technical document, Stanford University* (1996).
- [Im+19] Sungjin Im, Benjamin Moseley, Kirk Pruhs, and Manish Purohit. “Matroid Coflow Scheduling”. In: *Proceedings of ICALP*. Vol. 132. LIPIcs. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 145:1–145:14. DOI: [10.4230/LIPIcs.ICALP.2019.145](https://doi.org/10.4230/LIPIcs.ICALP.2019.145).
- [IP18] Sungjin Im and Manish Purohit. *A Tight Approximation for Co-flow Scheduling for Minimizing Total Weighted Completion Time*. 2018. arXiv: [1707.04331](https://arxiv.org/abs/1707.04331) [cs.DS].
- [JKR17] Hamidreza Jahanjou, Erez Kantor, and Rajmohan Rajaraman. “Asymptotically Optimal Approximation Algorithms for Coflow Scheduling”. In: *Proceedings of SPAA*. ACM, 2017. DOI: [10.1145/3087556.3087567](https://doi.org/10.1145/3087556.3087567).
- [Khu+19] Samir Khuller, Jingling Li, Pascal Sturfels, Kevin Sun, and Prayaag Venkat. “Select and permute: An improved online framework for scheduling to minimize weighted completion time”. In: *Theoretical Computer Science* 795 (2019), pp. 420–431. DOI: [10.1016/j.tcs.2019.07.026](https://doi.org/10.1016/j.tcs.2019.07.026).
- [Kön16] Dénes König. “Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre”. In: *Mathematische Annalen* 77 (1916), pp. 453–465. DOI: <https://doi.org/10.1007/BF01456961>.
- [LLP07] Joseph Y.-T. Leung, Haibing Li, and Michael Pinedo. “Scheduling orders for multiple product types to minimize total weighted completion time”. In: *Discrete Applied Mathematics* 155.8 (2007), pp. 945–970. DOI: [10.1016/j.dam.2006.09.012](https://doi.org/10.1016/j.dam.2006.09.012).
- [Mas+10] Monaldo Mastrolilli, Maurice Queyranne, Andreas S. Schulz, Ola Svensson, and Nelson A. Uhan. “Minimizing the sum of weighted completion times in a concurrent open shop”. In: *Operations Research Letters* 38.5 (2010), pp. 390–395. DOI: [10.1016/j.orl.2010.04.011](https://doi.org/10.1016/j.orl.2010.04.011).
- [Mos+20] Ali Mostafaeipour, Amir Jahangard Rafsanjani, Mohammad Ahmadi, and Joshua Arockia Dhanraj. “Investigating the performance of Hadoop and Spark platforms on machine learning algorithms”. In: *The Journal of Supercomputing* 77.2 (2020), pp. 1273–1300. DOI: [10.1007/s11227-020-03328-5](https://doi.org/10.1007/s11227-020-03328-5).

- [QSZ15] Zhen Qiu, Cliff Stein, and Yuan Zhong. “Minimizing the Total Weighted Completion Time of Coflows in Datacenter Networks”. In: *Proceedings of SPAA*. ACM, 2015, pp. 294–303. DOI: [10.1145/2755573.2755592](https://doi.org/10.1145/2755573.2755592).
- [Sch21] Leander Schnaars. “Concurrent Open Shop and Coflow Scheduling”. MA thesis. Munich, Germany: Technical University of Munich, 2021.
- [Sch86] Alexander Schrijver. *Theory of linear and integer programming*. USA: John Wiley & Sons, Inc., 1986.
- [Shv+10] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. “The Hadoop Distributed File System”. In: *Proceedings of IEEE MSST*. 2010, pp. 1–10. DOI: [10.1109/MSST.2010.5496972](https://doi.org/10.1109/MSST.2010.5496972).
- [SS13] Sushant Sachdeva and Rishi Saket. “Optimal Inapproximability for Scheduling Problems via Structural Hardness for Hypergraph Vertex Cover”. In: *2013 IEEE Conference on Computational Complexity*. IEEE, 2013, pp. 219–229. DOI: [10.1109/cc.2013.30](https://doi.org/10.1109/cc.2013.30).
- [Tar86] Éva Tardos. “A Strongly Polynomial Algorithm to Solve Combinatorial Linear Programs”. In: *Operations Research* 34.2 (1986), pp. 250–256. DOI: [10.1287/opre.34.2.250](https://doi.org/10.1287/opre.34.2.250).
- [Zah+10] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. “Spark: cluster computing with working sets”. In: *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*. Boston, MA: USENIX Association, 2010, p. 10.

6 Appendix

6.1 Allocation LP Integrality and Complexity

Non-Integrality

We provide a set of coflows and associated deadlines such that [LP I](#) contains a feasible fractional point but no integral point. The instance is defined on a vertex set $U \cup V$, where both U and V contain exactly 7 vertices. It uses four coflows E_1, E_2, E_3, E_4 with associated deadlines $C_1 = 1, C_2 = 2, C_3 = 3, C_4 = 3$. The first three coflows act as a gadget construction which blocks certain vertices in V from being used by edges in E_4 . The edge sets of the gadget coflows are $E_1 = \{(6, 2), (7, 7)\}$, $E_2 = \{(6, 4), (7, 5)\}$, $E_3 = \{(6, 1), (7, 3)\}$. Any valid integral schedule has to schedule E_1 in the first time slot, E_2 in the second time slot and E_3 in the third time slot, which essentially implies that coflow E_4 cannot use V vertices $\{2\}, \{4, 5\}, \{1, 3\}$ in the first, second, and respectively third time slot. For easier argumentation we separate the edges of E_4 into two sets $E_4^1 = \{(1, 1), (2, 1), (3, 3), (4, 3), (1, 4), (3, 4), (2, 5), (4, 5)\}$ and $E_4^2 = \{(2, 2), (3, 2)\}$. There is a feasible half-integral allocation for this instance displayed in [Figure 1](#).

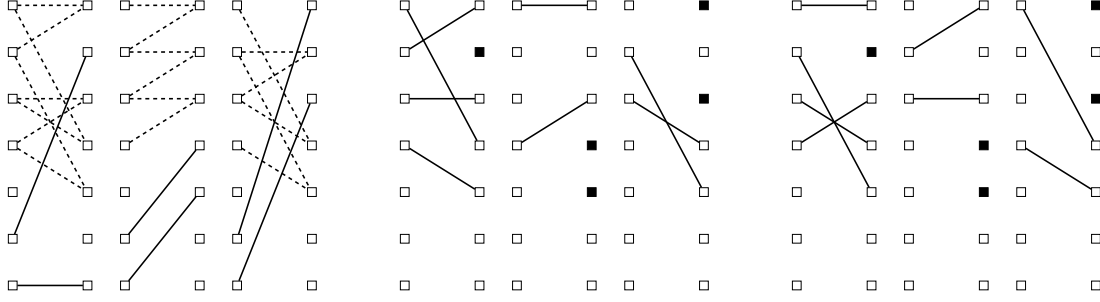


Figure 1: The first picture shows a feasible fractional allocation, where dashed lines indicate values of $\frac{1}{2}$. The second and third picture show the two allocations for $E_1 \cup E_2 \cup E_3 \cup E_4^1$, with the relevant V vertices which are blocked by gadget edges replaced by filled vertices for easier readability.

By simple enumeration one can check that there are only two feasible integral allocations containing all edges in $E_1 \cup E_2 \cup E_3 \cup E_4^1$. The choice whether to schedule $(1, 1)$ in the first slot and $(2, 1)$ in the second slot or vice versa already fully determines a unique maximal matching in each case. However, for both of these allocations it is impossible to further include both edges in E_4^2 , as vertex 2 in V is blocked in the first time slot and in both cases in either the second or the third time slot both vertices 2 and 3 are blocked in U . Therefore there is no integral matching containing all edges in $E_1 \cup E_2 \cup E_3 \cup E_4$.

NP-Hardness

For [LP I](#) it is possible to show that it is NP-hard to decide whether an integral point exists. This is equivalent to showing that it is NP-hard to determine whether a feasible integral schedule exists for some given coflow deadlines. The hardness holds even in a very restricted setting with just 3 coflows and all deadlines in $\{1, 2, 3\}$.

Lemma 14. *Given some bipartite multi-graph $G = (U \cup V, E)$, a disjoint partition of the edges $E = E_1 \cup \dots \cup E_n$ and deadlines $C_1, \dots, C_n \in \mathbb{N}$, it is NP-complete to decide whether there exists a proper coloring $c : E \rightarrow \mathbb{N}$ such that for all $j \in [n], e \in E_j : c(e) \leq d_j$. This holds even for $n = 3$ and $C_j \in \{1, 2, 3\}$.*

Proof. [TOPROVE 10](#) □

6.2 Coflow Deadline Linear Program

This section gives a short overview of the linear programming approach used to obtain deadlines for the coflows. All of the constructions and results are due to [\[Fuk22\]](#) and [\[Im+19\]](#).

Linear Program

We use the linear program [LP D](#) as given in [\[Fuk22\]](#), adapted to our notation. For this purpose, let $E = \bigcup_{j \in [n]} E_j$ be the set of all edges and let $T = \max_{j \in [n]} r_j + 2 \cdot \Delta(E)$ be an upper bound on the number of required time slots. Remember that we use the term flow to refer to an edge together with some possibly fractional multiplicity. The LP has one variable $x_{t,e}$ for each $t \in [T]$ and $e \in E$, which models the processing of flow e during time step t . Additionally, for each coflow $j \in [n]$, there is a finishing time variable c_j .

$$\begin{aligned}
\min \quad & \sum_{j \in [n]} \omega_j c_j \\
s.t. \quad & \sum_{t \in [T]} t \cdot x_{t,e} \leq c_j & \forall j \in [n], \forall e \in E_j \quad (1) \\
& \sum_{e \in \delta_E(v)} x_{t,e} \leq 1 & \forall t \in [T], \forall v \in V \quad (2) \\
& \sum_{t \in [T]} x_{t,e} = 1 & \forall j \in [n], \forall e \in E_j \quad (3) \\
& x_{t,e} = 0 & \forall j \in [n], \forall e \in E_j, \forall t \in [r_j] \quad (4) \\
& x_{t,e} \geq 0 & \forall t \in [T], \forall e \in E \quad (5)
\end{aligned} \tag{LP D}$$

Constraint (1) models that the completion time of each coflow is at least the time spent on scheduling each edge in said coflow. Constraint (2) models the matching constraints. Constraint (3) models that each edge of all coflows has to be fully scheduled.

Any valid solution for a coflow instance corresponds to a feasible point inside the polytope, so clearly we have $\text{Cost}(\text{LP}) \leq \text{OPT}$, where OPT is the optimal cost of the coflow instance. Note that any feasible solution to the LP forms a sequence of fractional matchings in the underlying graph. Instead of as a discrete fractional assignments in each time slot, we can also view them as continuous assignments during intervals. So if some edge e during time t has $x_{t,e} > 0$ flow assigned, we can view this as a continuous scheduling of $x_{t,e}$ flow amount during time $[t, t+1)$. In the same manner, given such a continuous assignment, by discretizing into unit length intervals, one can obtain a discrete fractional assignment. This view is both used in the coming rounding argument but also useful in general to analyze and understand the problem structure.

Obtaining Coflow Deadlines

The finishing time variables c_j capture the notion of time spent processing each edge, but they do not correspond cleanly to deadlines for each coflow. If we for example set $C_j = \max\{t \in [T] : \exists e \in E_j : x_{e,t} > 0\}$, then the weighted sum over these deadlines might far exceed the cost of the LP, as it is possible that due to the fractionality of the matchings, some fraction of an edge is scheduled very late, even though the majority of the coflow is scheduled much earlier. To obtain deadlines for which there are strong approximation guarantees, we use a randomized rounding procedure as employed by [Fuk22; Im+19] and others. For this purpose, for $j \in [n]$ and $\theta \in [0, 1]$ let $C_j(\theta)$ be the smallest point in time at which all flows $e \in E_j$ have completed by at least a θ fraction in the continuous view of the fractional assignment. This means that we view each flow as being continuously assigned during the respective time slots and we determine the smallest point in time at which this continuous schedule reaches θ completion for all respective flows. The authors in the aforementioned works obtain deadlines by randomly selecting θ according to the probability distribution $f(x) = 2x$ and setting $C'_j := \lceil C_j(\theta)/\theta \rceil$. As we do not require integrality in the next steps of our algorithm, we set $C_j := C_j(\theta)/\theta$, leaving out the rounding step.

It can be shown that the following holds for the rounded deadlines:

Lemma 15 ([Im+19]). *There is a polynomial time randomized algorithm determining deadlines C'_1, \dots, C'_n for which LP I is feasible and for which for all $j \in [n]$:*

$$\mathbb{E}[C'_j] \leq 2c_j$$

Slightly modifying their proof to account for not rounding the deadlines, we obtain:

Lemma 5 ([Im+19]). *There is a polynomial time randomized algorithm determining deadlines C_1, \dots, C_n for which LP I is feasible and for which the following cost bound holds:*

$$\sum_{j \in [n]} \omega_j \mathbb{E}[C_j] \leq 2 \cdot \text{OPT} - \sum_{j \in [n]} \omega_j$$

Proof. **TOPROVE 11** □

Note that this procedure can be de-randomized to obtain a fully deterministic algorithm, for details see [Im+19].

The choice of θ and subsequent setting of $C_j := C_j(\theta)/\theta$ can be viewed as stretching the continuous schedule obtained for LP D. This means that if some amount of flow $x_{e,t}$ is scheduled during time $[t, t+1)$, after the stretching the same amount of flow is scheduled during $[\frac{t}{\theta}, \frac{t+1}{\theta})$. To not exceed the flow requirements, the schedule is cut off when the total amount of scheduled flow reaches the requirement. By definition of $C_j(\theta)$ and the matching constraints in LP D, in the stretched schedule the matching constraints are still fulfilled and every coflow E_j finishes by time C_j . By partitioning the continuous schedule into intervals between consecutive deadlines, the connection to LP I becomes clear. The maximum allocation constraints are fulfilled as the stretched schedule fulfills the matching constraints and by definition of C_j , for every edge enough flow is allocated before the deadline. This argument highlights that we can explicitly construct a feasible point for which LP I is feasible and the cost guarantees from Lemma 5 hold, by performing the stretching operation on the schedule obtained for LP D and then discretizing the assignment with respect to each block.

6.3 Coflow Scheduling with Release Dates

In this section we show how the scheduling framework can be extended to work for the case with release dates. Theorem 2 gives an extension of the guarantee provided by Theorem 1 to the case with release dates, though this comes at the cost of a worse approximation ratio.

Theorem 2. *There is a polynomial time algorithm achieving a 4.36-approximation for Coflow Scheduling with release dates.*

The overall proof structure is very similar to the case without release dates, just with tweaks at every step to account for the additional constraints. We provide the general outline here and omit some minor details which follow from modifications to the original arguments.

Coflow Deadlines

Like in the case of no release dates, we want to obtain deadlines for the coflows which obey some structural constraints. LP I does not contain release dates, but with some minor modifications we obtain a suitable LP. For this purpose, for some $\kappa \in [2n]$, define a sequence $D_1 \leq D_2 \leq \dots \leq D_\kappa$ containing exactly all deadlines and release dates. For some edge $e \in E$, let $r(e)$ be the index of the release date associated to e in the chain of points in D and respectively $d(e)$ the index of the deadline. Then we construct the following LP.

$$\begin{aligned} \sum_{s \in [\kappa]} x_{s,e} &= 1 & \forall e \in E \\ \sum_{e: v \in e} x_{s,e} &\leq D_s - D_{s-1} & \forall s \in [\kappa], \forall v \in V \\ x_{s,e} &= 0 & \forall j \in [n], \forall e \in E_j, \forall s \notin \{r(e) + 1, \dots, d(e)\} \\ x_{s,e} &\geq 0 \end{aligned} \tag{LP R}$$

The structure of **LP R** is very similar to **LP I**, just with added block separators for each release date and modification of the constraints to prevent edges from being scheduled in blocks before their respective release dates.

The same procedure by [Im+19] used in Section 3.1 can be employed to obtain deadlines C_1, \dots, C_n for which **LP R** is feasible and for which the same cost bound from Lemma 5 holds.

Edge Allocation

Given such a set of deadlines for which **LP R** is feasible, we again describe two algorithms Greedy_R and CBF_R^τ which provide feasible allocations for all edges. Their guarantees are slightly worse due to the added release date constraints.

Like previously, $\text{Greedy}_R(C_j)$ and $\text{CBF}_R^\tau(C_j)$ will be used to denote the finishing time of coflow E_j in the schedule provided by the respective algorithm.

Lemma 16. *For given deadlines C_1, \dots, C_n for which **LP R** is feasible there is an algorithm Greedy_R returning a valid coflow schedule such that the following holds for all $j \in [n]$.*

$$\text{Greedy}_R(C_j, r_j) \leq r_j + 2C_j - 1$$

Proof. **TOPROVE 12** □

For the allocation procedure CBF_R^τ , the guarantee worsens by an additive $\tau + 2$.

Lemma 17. *For given deadlines C_1, \dots, C_n for which **LP R** is feasible, weights $\omega_1, \dots, \omega_n$, and a parameter $\tau \in \mathbb{N}_{\geq 2}$, there is an algorithm CBF_R^τ returning a valid coflow schedule such that the following holds.*

$$\sum_{j \in [n]} \omega_j \cdot \text{CBF}_R^\tau(C_j) \leq \sum_{j \in [n]} \omega_j \left(\frac{\tau + 2}{\tau} C_j + \frac{3}{2} \tau + 4.5 - \frac{2}{\tau} \right)$$

Proof. **TOPROVE 13** □

Framework and Approximation Bound

The algorithm for Coflow Scheduling with release dates again works by obtaining deadlines and then running several edge allocation algorithms on these and returning the cheapest solution among them. To bound the cost, a framework very similar to the one described in Lemma 8 is used, though an additional bound on the distance to the optimum cost is needed due to the presence of the r_j summand in the guarantee provided by Greedy_R . As in any optimal solution the finishing time OPT_j of coflow E_j has to be after r_j , we have $r_j \leq \text{OPT}_j - 1$.

In the following lemma, like in Lemma 8, let f_1, \dots, f_k be some functions capturing the edge allocation guarantees provided by some collection of algorithms $\text{ALG}_1, \dots, \text{ALG}_k$. In this case the functions additionally depend on a parameter $r_j \in \mathbb{R}_{\geq 0}$, which like in the case for Greedy captures the dependency on release dates.

Lemma 18. *Let $\lambda_1, \dots, \lambda_k \geq 0$ with $\sum_{i \in [k]} \lambda_i = 1$ and $a, b \in \mathbb{R}$. If for all possible pairs $x \geq 1, r_x \in [0, x - 1]$*

$$\sum_{i \in [k]} \lambda_i f_i(x, r_x) \leq a(x + 1) + b(r_x + 1),$$

then for all coflow instances \mathcal{I} :

$$C_{\text{ALG}}(\mathcal{I}) = \min\{C_{\text{ALG}_1}(\mathcal{I}), \dots, C_{\text{ALG}_k}(\mathcal{I})\} \leq (2a + b) \cdot \text{OPT}(\mathcal{I})$$

Proof. TOPROVE 14 □

We can now apply this modified framework to show Theorem 2.

Theorem 2. *There is a polynomial time algorithm achieving a 4.36-approximation for Coflow Scheduling with release dates.*

Proof. TOPROVE 15 □

6.4 High Edge Multiplicities

In the main body of this work we have assumed that in each coflow, each edge is given explicitly, meaning that multiple copies of said edge are included if the respective flow demand is greater than 1. In this chapter we show how to extend the algorithms to the setting where instead each edge $e \in E$ has some associated flow requirement $p_e \in \mathbb{N}_+$. To ensure polynomial runtime, we need dependency on $\mathcal{O}(\log(p_e))$ instead of $\mathcal{O}(p_e)$. We individually highlight the changes required in each step. Extending the algorithms to this setting increases the cost by a multiplicative factor of $(1 + \epsilon)$, for an arbitrarily small $\epsilon > 0$.

Coflow Deadlines

Obtaining coflow deadlines in this modified setting requires some additional care, as the procedure uses a time-indexed LP, which could potentially require a super-polynomial amount of time slots. We adapt the procedure described in [Im+19] to our case. The basic idea is to change the time-indices to interval indices, for a polynomially sized set of geometrically increasing intervals. We focus on the case without release dates, the case with them follows analogously.

Let T be some upper bound on the number of required time slots, for example the sum over all multiplicities. For some $\epsilon > 0$, define the set of points $\{\lfloor (1 + \epsilon)^i \rfloor\}_{i \in [\lceil \log_{1+\epsilon} T \rceil]}$. For some $K \in \mathbb{N}$, let these timepoints be $1 = t_1 \leq t_2 \leq \dots \leq t_K$. For convenience, set $t_0 = 0$. The modified LP is as follows.

$$\begin{aligned}
 \min \quad & \sum_{j \in [n]} \omega_j c_j \\
 \text{s.t.} \quad & \sum_{i \in [K]} (t_{i+1} - 1) \cdot x_{i,e} \leq c_j & \forall j \in [n], \forall e \in E_j & (1) \\
 & \sum_{e \in \delta_E(v)} x_{t,e} \leq t_i - t_{i-1} & \forall i \in [K], \forall v \in V & (2) & (\text{LP D}') \\
 & \sum_{i \in [K]} x_{i,e} = p_e & \forall j \in [n], \forall e \in E_j & (3) \\
 & x_{i,e} \geq 0 & \forall i \in [K], \forall e \in E & (4)
 \end{aligned}$$

In this LP, the timepoints are replaced by time intervals of size $t_{i+1} - t_i$. The right hand sides of constraints (2) and (3) are respectively adjusted to account for this. If an interval contains some amount of flow $x_{i,e}$, it is assumed that this is scheduled instantaneously at the latest possible time point $t_{i+1} - 1$ in the interval, which gives the term in (1). This leads to an overestimation of the cost by a factor of at most $(1 + \epsilon)$. The rounding procedure to determine deadlines C_1, \dots, C_n and the associated results for LP feasibility remain unchanged.

Greedy Flow Allocation

For the greedy algorithm, we cannot simply do one iteration for each edge and multiplicity, as this could require super-polynomially many steps. The required adaption to the procedure was described in a similar form in [QSZ15].

We create n sets of edges iteratively, one associated to each coflow. Assume that E_1, \dots, E_n are the sets of edges ordered without loss of generality ascendingly by the deadlines returned in the first step. The first set initially only contains the edges in E_1 . When creating the j -th set, we iterate over all edges $e \in E_j$. If there exists some set with index $i < j$ such that e could be added to the set without increasing its maximum vertex degree, we add a copy \hat{e} of e to this set with as much flow demand $p_{\hat{e}}$ as possible without increasing the maximum degree and remove this flow demand from p_e . After doing this procedure for all coflows and edges, we have n sets of edges, some of which might be empty. Each of these blocks can now be scheduled consecutively in polynomial time using Theorem 4 (König's Theorem). It is easy to check that for each edge the greedy property is fulfilled, meaning it cannot be scheduled earlier without changing other allocation. Therefore the bounds guaranteed by Lemma 6 hold. In the case of release dates, the sets have to be subdivided to prevent crossing any release date and during the flow shifting only sets have to be considered for which the lowest possible time slot is larger than the respective release date.

Coflow Beck-Fiala

To obtain a polynomial algorithm for the CBF^τ allocation procedure we only need to do a minor modification to LP CBF. Initially, in constraint (I), we change the right hand side to p_e instead of 1. After having obtained a solution to this modified LP, for every edge and every block, fix the integral part of the respective flow assigned to this block. Replace each edge e by \hat{p}_e copies and change the right hand side to 1 again, where \hat{p}_e is the sum over the remaining fractional assignments. As there are at most $2n$ blocks, this amount is upper bounded by $2n$, hence strongly polynomial in the input size. Solve the remaining instance with the unmodified iterated rounding process.

6.5 LP Integrality Gap

In his work, using a non-constructive result from hypergraph matching theory, Fukunaga [Fuk22] shows that the integrality gap of LP D is at most 4, even in the case of release dates. By using the framework from Lemma 8 on a slightly refined version of the non-constructive hypergraph result together with one of the edge allocation algorithms from this work, we establish a stronger bound.

Lemma 19. *The integrality gap of LP D is at most $\frac{109}{28} (< 3.893)$.*

To show this, we use the following bound which is obtained by refining a result from [Fuk22].

Lemma 20. *Given deadlines C_1, \dots, C_n and release dates r_1, \dots, r_n for which LP R is feasible, there exists a valid coflow schedule such that for the finishing times C_j^* the following holds.*

$$C_j^* \leq 2C_j + 1$$

Proof. TOPROVE 16 □

Combining this with the guarantees obtained from Lemma 17, Lemma 19 can be shown.

Proof. TOPROVE 17 □

6.6 Approximation Improvements

Using the framework from Lemma 8 together with a new edge allocation function, we can achieve a slight improvement upon the 3.415-approximation from Theorem 1.

Lemma 21. *Given deadlines C_1, \dots, C_n for which [LP CBF](#) is feasible, weights $\omega_1, \dots, \omega_n$ and a parameter $\tau \in \mathbb{N}_{\geq 2}$ and a parameter $b \in \mathbb{N}_{\geq 1}$, there is an algorithm $\text{CKBF}^{(\tau, b)}$ returning a valid coflow schedule such that the following holds.*

$$\sum_{j \in [n]} \omega_j \cdot \text{CKBF}^{(\tau, b)}(C_j) \leq b \cdot \sum_{j: C_j < b+1} \omega_j + \sum_{j: C_j \geq b+1} \omega_j \left(\frac{\tau+2}{\tau} C_j + \frac{\tau}{2} + 2.5 + b - \frac{2}{\tau} \right)$$

Proof. [TOPROVE 18](#) □

Combining this algorithm with the edge allocation algorithms previously used, the following stronger bound can be derived. The difference between the two bounds is slightly more than $\frac{1}{100}$.

Lemma 22. *There is a polynomial time algorithm achieving a $\frac{497}{146} (< 3.4042)$ -approximation for Coflow Scheduling without release dates.*

Proof. [TOPROVE 19](#) □

Note that the same approach does not give an improvement for neither Coflow Scheduling with release dates nor the integrality gap from Section 6.5. We surmise that similar improvements for these cases and also further tiny improvements for the case without release dates might be possible using more involved combinations of (possibly new) edge allocation functions.