Select and Permute: An Improved Online Framework for Scheduling to Minimize Weighted Completion Time *

Samir Khuller[1], Jingling Li[1], Pascal Sturmfels[2], Kevin Sun[3], and Prayaag Venkat[1]

[1] University of Maryland, College Park, MD 20742
samir@cs.umd.edu, jinglingli1024@gmail.com, pkvasv@gmail.com
[2] University of Michigan, Ann Arbor, MI 48109
psturm@umich.edu
[3] Duke University, Durham, NC 27708
ksun@cs.duke.edu

Abstract. In this paper, we introduce a new online scheduling framework for minimizing total weighted completion time in a general setting. The framework is inspired by the work of Hall et al. [10] and Garg et al. [8], who show how to convert an offline approximation to an online scheme. Our framework uses two offline approximation algorithms—one for the simpler problem of scheduling without release times, and another for the *minimum unscheduled weight problem*—to create an online algorithm with provably good competitive ratios.

We illustrate multiple applications of this method that yield improved competitive ratios. Our framework gives algorithms with the best or only-known competitive ratios for the concurrent open shop, coflow, and concurrent cluster models. We also introduce a randomized variant of our framework based on the ideas of Chakrabarti et al. [3] and use it to achieve improved competitive ratios for these same problems.

Keywords: coflow scheduling, concurrent clusters, concurrent open shop, online algorithms.

1 Introduction

Modern computing frameworks such as MapReduce, Spark, and Dataflow have emerged as essential tools for big data processing and cloud computing. In order to exploit large-scale parallelism, these frameworks act in several computation stages, which are interspersed with intermediate data transfer stages. During

* All authors performed this work at the University of Maryland, College Park, under the support of NSF REU Grant CNS 156019. We would also like to thank An Zhu and Google for their support, and the LILAC program at Bryn Mawr College.

data transfer, results from computations must be efficiently scheduled for transfer across clusters so that the next computation stage can begin.

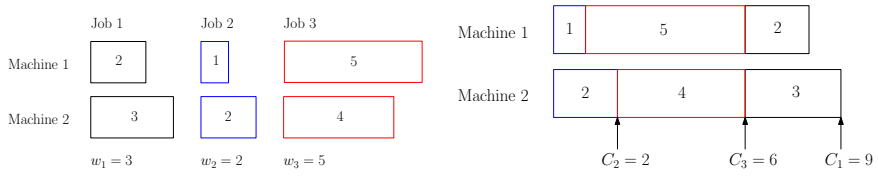
The coflow model [5, 6] and the concurrent cluster model [11, 19] were introduced to capture the distributed processing requirements of jobs across many machines. In these models, the objective of primary theoretical and practical interest is to minimize average job completion time [1, 5, 6, 14, 19, 20]. The *concurrent open shop problem*, a special case of the above models, has emerged as a key subroutine for designing better approximation algorithms [1, 14, 19].

There has been a lot of work studying offline algorithms for these problems (see [1, 14, 20] for the coflow model, [19] for the concurrent cluster model, and [4, 8, 18, 23] for the concurrent open shop model), but in real-world applications, jobs often arrive in an online fashion, so studying online algorithms is critical for accurate modeling of data centers.

Hall et al. [10] proposed a general framework which converts offline scheduling algorithms to online ones. Inspired by this result, we introduce a new online framework that improves upon the online algorithms of Garg et al. [8] for concurrent open shop and also gives the first algorithms with constant competitive ratios for other multiple-machine scheduling settings.

1.1 Formal Problem Statement

In the concurrent open shop setting, the problem is to schedule a set of jobs with machine-dependent components on a set of machines. Let $J = \{1, \ldots, n\}$ denote the set of jobs and $M = \{1, \ldots, m\}$ denote the set of machines. Each job j has one component for each of the m machines. For each job j, we denote the processing time of the component on machine i as p_{ij}, its release time as r_j, and its weight as w_j. The different components of each job can be processed concurrently and in any order, as long as no component of job j is processed before r_j. Job j is complete when all of its components have been processed; we denote its completion time by C_j. Our goal is to specify a schedule of the jobs on the machines that minimizes $\sum_{j \in J} w_j C_j$; see Fig. 1 for an example.



(a) An instance of concurrent open shop. (b) The optimal schedule

Fig. 1: All jobs are released at the same time, and the processing requirement for each job-machine combination is specified inside the blocks.

We follow the 3-field $\alpha \,|\beta|\, \gamma$ notation (see [9]) for scheduling problems, where α denotes the scheduling environment, β denotes the job characteristics, and

γ denotes the objective function. As stated above, we focus on the case where $\gamma = \sum_j w_j C_j$. In accordance with the notation of [10, 18, 19], we let $\alpha = PD$ denote the concurrent open shop setting and $\alpha = CC$ denote the concurrent cluster setting, see below for definitions.

1.2 Related Work

The concurrent open shop model is a relaxation of the well-known open shop model that allows components of the same job to be processed in parallel on different machines. Roemer [21] showed that $PD \,||\, \sum_j w_j C_j$ is NP-hard and after several successive approximation hardness results [2, 18], Sachdeva and Saket [22] showed that it is not approximable within a factor less than 2 unless P = NP, even when job release times are identical. For this model, Wang and Cheng [23] gave a $\frac{16}{3}$-approximation algorithm. This was later improved to a 2-approximation for identical job release times [4, 8, 15, 18], matching the above lower bound, and a 3-approximation for arbitrary job release times [1, 8, 15]. In the preemptive setting, Im and Purohit [12] gave a $(2+\epsilon)$-approximation for arbitrary job release times.

In the online setting, Hall et al. [10] introduced a general framework that improved the best-known approximation guarantees for several well-studied scheduling environments. They showed that the existence of an offline *dual* ρ-approximation yields an online 4ρ-approximation, where a dual ρ-approximation is an algorithm that packs as much weight of jobs into a time interval of length ρD as the optimal algorithm does into an interval of length D. Furthermore, they showed that when $m = 1$, a local greedy ordering of jobs yields further improvements.

While the framework of Hall et al. [10] is entirely deterministic, Chakrabarti et al. [3] gave a randomized variant with an improved competitive ratio guarantee. Specifically, they showed a dual-ρ approximation algorithm can be converted to an expected 2.89ρ-competitive online scheduling algorithm in the same setting, improving upon the 4ρ competitive ratio of Hall et al. [10].

The online version of $PD \,||\, \sum_j w_j C_j$ was first studied by Garg et al. [8]. They noted that applying the framework of [10] was not straightforward, so they focused on minimizing the weight of unscheduled jobs rather than maximizing the weight of scheduled jobs. Using a similar approach to that of Hall et al. [10], they gave an exponential-time 4-competitive algorithm and a polynomial-time 16-competitive algorithm for the online version of $PD \,||\, \sum_j w_j C_j$.

The coflow scheduling model was first introduced as a networking abstraction to model communications in datacenters [5, 6]. In the coflow scheduling problem, the goal is to schedule a set of *coflows* on a non-blocking switch with m input ports and m output ports, where any unused input-output ports can be connected via a path through unused nodes regardless of other existing paths. Each coflow is a collection of parallel flow demands that specify the amount of data that needs to be transferred from an input port to an output port.

For Coflow $|r_j = 0| \sum_j w_j C_j$, Qiu et al. [20] gave deterministic $\frac{64}{3}$ and randomized $(8 + \frac{16\sqrt{2}}{3})$ approximation algorithms. For arbitrary release times, they gave deterministic $\frac{67}{3}$ and randomized $9 + \frac{16\sqrt{2}}{3}$ approximation algorithms.

Khuller and Purohit [14] later improved these deterministic approximations to 8 and 12 for identical and arbitrary release times respectively, and also gave a randomized $(3+2\sqrt{2})$-approximation algorithm for identical release times. Ahmadi et al. [1] gave a deterministic 4-approximation and 5-approximation for identical and arbitrary release times, respectively. Recently, Im and Purohit [12] achieved a tight approximation ratio of $2+\epsilon$ for arbitrary release times. To the best of our knowledge, there are no known constant-factor competitive algorithms for online coflow scheduling, although Li et al. [16] give a $O(m \ln n)$-competitive algorithm when all coflow weights are equal to 1, where m is the number of coflows and n is the number of nodes in the network.

Finally, we mention the *concurrent cluster* model recently introduced by Murray et al. [19]. The concurrent cluster model generalizes the concurrent open shop model by replacing each machine by a cluster of machines, where different machines in the same cluster may have different processing speeds. Each job still has m processing requirements, but this requirement can be fulfilled by any machine in the corresponding cluster. Murray et al. [19] give the first constant-factor approximations for minimizing total weighted completion time via a reduction to concurrent open shop and a list-scheduling subroutine.

1.3 Paper Outline and Results

In Sect. 2, we introduce a general framework for designing online scheduling algorithms for minimizing total weighted completion time. The framework divides time into intervals of geometrically-increasing size, and greedily "packs" jobs into each interval, and then imposes a locally-determined ordering of the jobs within each interval. It is inspired by the framework of Hall et al. [10] and an adaptation by Garg et al. [8].

In Sect. 3, we apply our framework to $PD \,||\, \sum_j w_j C_j$. We show that an offline exponential-time algorithm that optimally solves $PD \,|r_j = 0|\, \sum_j w_j C_j$ yields an exponential-time 3-competitive algorithm for $PD \,||\, \sum_j w_j C_j$. We also combine the algorithms given by Garg et al. [8] and Mastrolilli et al. [18] to create a polynomial-time 10-competitive algorithm for $PD \,||\, \sum_j w_j C_j$. We conclude Sect. 3 by giving a polynomial-time $(3+\epsilon)$-competitive algorithm when the number of machines m is fixed. Details on the subroutines used in this section are provided in the full version of this paper [13].

In Sect. 4, extending the ideas of Sect. 3, we apply our framework to online coflow scheduling to design an exponential-time $(4+\epsilon)$-competitive algorithm, and a polynomial-time $(10+\epsilon)$-competitive algorithm.

Section 5 describes an extension of the techniques of Chakrabarti et al. [3] that produces a randomized variant of our framework that yields better competitive ratio guarantees than the deterministic version. The full version of this paper [13] describes the concurrent cluster model of Murray et al. [19]; we show that extending subroutines used for the concurrent open shop setting yields an online 19-competitive algorithm via our framework.

Table 1: A summary of online approximation guarantees and the best-known previous results, where m denotes the number of machines, ϵ is arbitrarily small, and "-" indicates the absence of a relevant result. The two numbers in each entry of the "Our ratios" column denote the competitive and expected ratio of our deterministic and randomized algorithms, respectively.

Problem	Running time	Our ratios	Previous ratio
$PD \mid\mid \sum_j w_j C_j$	polynomial	10, 7.78	16 [8]
$PD \mid\mid \sum_j w_j C_j$	exponential	3, 2.45	4 [8]
$PD \mid\mid \sum_j w_j C_j$	polynomial, fixed m	$3+\epsilon$, $2.45+\epsilon$	-
Coflow $\mid\mid \sum_j w_j C_j$	polynomial	$10+\epsilon$, $7.78+\epsilon$	-
Coflow $\mid\mid \sum_j w_j C_j$	exponential	$4+\epsilon$, $3.45+\epsilon$	-
$CC \mid\mid \sum_j w_j C_j$	polynomial	19, 14.55	-

2 A Minimization Framework for Online Scheduling

In this section, we introduce our framework for online scheduling problems. To motivate the key ideas of this section, we begin by briefly reviewing the work of Hall et al. [10] and Garg et al. [8].

2.1 The maximization framework of Hall et al. [10]

The framework of Hall et al. [10] divides the online problem into a sequence of offline *maximum scheduled weight* problems, each of which is solved using an offline *dual* approximation algorithm.

Definition 1 (Maximum scheduled weight problem (MSWP) [10]). *Given a set of jobs, a non-negative weight for each job, and a deadline D, construct a schedule that maximizes the total weight of jobs completed by time D.*

Definition 2 (Dual ρ-approximation algorithm [10]). *An algorithm for the MSWP is a* dual ρ-approximation algorithm *if it constructs a schedule of length at most ρD and has total weight at least that of the schedule which maximizes the weight of jobs completed by D.*

Fix a scheduling environment and suppose we have a dual ρ-approximation for the MSWP. We divide time into intervals of geometrically-increasing size by letting $t_0 = 0$ and $t_k = 2^{k-1}$ for $k = 1, \ldots, L$ where L is large enough to cover the entire time horizon. At each time t_k, let $R(t_k)$ denote the set of jobs that have arrived by t_k but have not yet been scheduled. We run the dual ρ-approximation algorithm on $R(t_k)$ with deadline $D = t_{k+1} - t_k = t_k$. In the output schedule, we take only jobs which complete by ρD and schedule them in the interval starting at time ρt_k. Hall et al. [10] show that this framework produces an online 4ρ-competitive algorithm.

2.2 The minimum unscheduled weight problem of Garg et al. [8]

Garg et al. [8] sought to apply the framework of Hall et al. [10] to the concurrent open shop setting. They noted that devising a dual-ρ approximation algorithm for concurrent open shop was difficult, so they instead proposed a variant of the MSWP. The definitions below generalize those used by Garg et al. [8] to arbitrary scheduling problems.

Definition 3 (Minimum unscheduled weight problem (MUWP)). *Given a set of jobs, a non-negative weight for each job, and a deadline D, find a subset of jobs S which can be completed by time D and minimizes the total weight of jobs not in S. We call this quantity the* unscheduled weight*.*

Definition 4 ((α, β)-approximation algorithm). *An algorithm for the MUWP is an* (α, β)-approximation *if it finds a subset of jobs which can be completed by αD and has unscheduled weight at most β times that of the subset of jobs with minimum unscheduled weight that completes by D.*

Note that a dual ρ-approximation for the MSWP is a $(\rho, 1)$-approximation for the MUWP. With these definitions, Garg et al. [8] established constant-factor approximations for $PD \,||\, \sum_j w_j C_j$.

2.3 A minimization framework

We now describe a new framework inspired by the ideas of Hall et al. [10] and Garg et al. [8]. For the settings we consider, previous online algorithms do not impose any particular ordering of jobs within each interval, which can lead to schedules with poor local performance. In our framework, we make use of a γ-approximation to the offline version of the scheduling problem with identical release times to address this issue.

As in the works of Hall et al. [10] and Garg et al. [8], we assume that all processing times are at least 1. This is to avoid the extreme scenario that a single job of size $\epsilon \ll 1$ arrives just after time 0, and our framework waits until time 1 to schedule, thus leading to arbitrarily large competitive ratio.

Let W denote the total weight of all the jobs in J, and let $W_\tau^{\mathcal{A}}$ ($W_\tau^{\mathcal{OPT}}$) denote the total weight of jobs that complete after time τ by our algorithm $\mathcal{A}$ (by the optimal algorithm $\mathcal{OPT}$). Note that $W_\tau^{\mathcal{A}}, W_\tau^{\mathcal{OPT}}$ include the weight of jobs not yet released at time τ. Let $\tau_0 = 0$, and for $k \geq 1$, let $\tau_k = 2^{k-1}$, I_k denote the k^{th} interval $[\tau_k, \tau_{k+1})$, αI_k denote $[\alpha\tau_k, \alpha\tau_{k+1})$, and $R(\tau_k)$ denote the set of jobs released but not yet scheduled before τ_k by $\mathcal{A}$.

Our online algorithm $\mathcal{A}$ works as follows. At each τ_k, run an (α, β)-approximation algorithm on $R(\tau_k)$ with deadline $D = \tau_{k+1} - \tau_k$. Schedule the output set of jobs in αI_k using the offline γ-approximation algorithm.[4]

[4] We make the critical assumption that the offline γ-approximation algorithm does not increase the makespan of the given subset of jobs, so as to ensure that the schedule fits inside of αI_k. For the scheduling models studied in this paper, this assumption

Theorem 1. *Algorithm $\mathcal{A}$ is $(2\alpha\beta+\gamma)$-competitive, with an additive αW term.*

To prove Theorem 1, we first show that at each time step, $\mathcal{A}$ remains competitive with the optimal schedule by incurring a time delay.

Lemma 1. *For any $k \geq 0$, we have $W^{A}_{\alpha\tau_{k+1}} \leq \beta W^{\mathcal{OPT}}_{\tau_k}$.*

Proof. Every job completed by $\mathcal{OPT}$ by τ_k must have been released before τ_k. For each such job j, either our algorithm completed it before time τ_k or $j \in R(\tau_k)$. The set of jobs completed by $\mathcal{OPT}$ by time τ_k gives a feasible solution to the MUWP with deadline $D = \tau_{k+1} - \tau_k = \tau_k$ and its total unscheduled weight is $W^{\mathcal{OPT}}_{\tau_k}$. Therefore, the optimal total unscheduled weight value for the MUWP when considering all $j \in R(\tau_k)$ with deadline D is at most $W^{\mathcal{OPT}}_{\tau_k}$. By the definition of (α, β)-approximation, the claim follows. □

The next lemma states that ordering jobs within each interval further approximates the optimal schedule closely. For a fixed subset S of jobs, let $\mathcal{OPT}(S)$ denote the optimal schedule for S and $C_j^{\mathcal{OPT}(S)}$ denote the completion time of job j in $\mathcal{OPT}(S)$. Also, let $\mathcal{OPT}_0(S)$ denote an optimal schedule that starts at time 0 and ignores all job release times, and let $C_j^{\mathcal{OPT}_0(S)}$ denote the completion time of job j in $\mathcal{OPT}_0(S)$.

Lemma 2. *The weighted completion time for schedule $\mathcal{OPT}_0(S)$ is at most that of schedule $\mathcal{OPT}(S)$; i.e.,*

$$\sum_{j\in S} w_j C_j^{\mathcal{OPT}_0(S)} \leq \sum_{j\in S} w_j C_j^{\mathcal{OPT}(S)} \quad .$$

Proof. The optimal schedule of S with release times defines a valid schedule for S without release times, so the claim follows. □

Recall that at each τ_k, $\mathcal{A}$ uses an (α, β)-approximation on the MUWP to select a subset S_k of $R(\tau_k)$ to schedule within αI_k using a γ-approximation that ignores release times. Let $C_j^{\mathcal{A}}$ denote the completion time of job j in the schedule produced by $\mathcal{A}$, $t(j)$ denote the largest index such that job j begins processing after time $\tau_{t(j)}$, and $\delta_j = C_j^{\mathcal{A}} - \alpha\tau_{t(j)}$ for each job $j \in J$ (see Fig. 2). Let L be the smallest time index such that the optimal schedule finishes by time τ_L, and let S_k denote the set of jobs scheduled independently by $\mathcal{A}$ in the interval αI_k. Then Lemma 2 implies

$$\sum_{j\in S_k} w_j\delta_j \leq \gamma \sum_{j\in S_k} w_j C_j^{\mathcal{OPT}_0(S_k)} \leq \gamma \sum_{j\in S_k} w_j C_j^{\mathcal{OPT}(S_k)} \quad . \tag{1}$$

will indeed hold. In fact, if it can be shown that the γ-approximation algorithm also approximates the makespan criteria within some factor μ, then it is straightforward to incorporate this into the model, at the expense of an additional μ factor in the approximation guarantee. For example, Chakrabarti et al. [3] provide bicriteria approximation algorithms for the total weighted completion time and makespan objective functions.

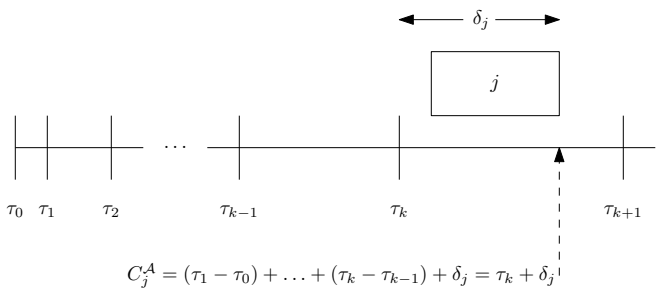


Fig. 2: We let δ_j denote the distance between $C_j^{\mathcal{A}}$ and the beginning of the interval in which job j completes.

Lemma 3. *The weighted sum of the δ_j is at most γ times the optimal weighted completion time; i.e.,*

$$\sum_{j \in J} w_j \delta_j \leq \gamma \sum_{j \in J} w_j C_j^{\mathcal{OPT}} \ .$$

Proof. Recall that $S_1, \ldots, S_L$ partition J, and notice that due to (1), we have

$$\sum_{k=1}^{L} \sum_{j \in S_k} w_j \delta_j \leq \gamma \sum_{k=1}^{L} \sum_{j \in S_k} w_j C_j^{\mathcal{OPT}(S_k)} \leq \gamma \sum_{j \in J} w_j C_j^{\mathcal{OPT}} \ , \tag{2}$$

thus proving the lemma. □

Proof (of Theorem 1). We rewrite the total weighted completion time of the schedule produced by $\mathcal{A}$ to obtain the following.

$$\begin{aligned}
\sum_{j \in J} w_j C_j^{\mathcal{A}} &= \alpha \sum_{k=1}^{L} (\tau_k - \tau_{k-1}) W_{\alpha\tau_k}^{\mathcal{A}} + \sum_{j \in J} w_j \delta_j \\
&= \alpha \sum_{k=2}^{L} (\tau_k - \tau_{k-1}) W_{\alpha\tau_k}^{\mathcal{A}} + \alpha W_{\alpha\tau_1}^{\mathcal{A}} + \sum_{j \in J} w_j \delta_j \\
&\leq 2\alpha \sum_{k=1}^{L} (\tau_k - \tau_{k-1}) W_{\alpha\tau_{k+1}}^{\mathcal{A}} + \alpha W + \sum_{j \in J} w_j \delta_j \\
&\leq 2\alpha\beta \sum_{k=1}^{L} (\tau_k - \tau_{k-1}) W_{\tau_k}^{\mathcal{OPT}} + \alpha W + \sum_{j \in J} w_j \delta_j \\
&\leq (2\alpha\beta + \gamma) \sum_{j \in J} w_j C_j^{\mathcal{OPT}} + \alpha W \ ,
\end{aligned}$$

where the last two inequalities follow from Lemmas 1 and 3, respectively. □

3 Applications to Concurrent Open Shop

Now we apply our minimization framework to $PD \,||\, \sum_j w_j C_j$. In the full version of this paper [13], we give an offline dynamic program that optimally solves $PD \,|r_j = 0|\, \sum_j w_j C_j$ in exponential time, giving $\gamma = 1$ in our framework.

For the MUWP, in exponential time, we can iterate over every subset of jobs to find a feasible schedule that minimizes the total weight of unscheduled jobs, so this is a $(1, 1)$-approximation, giving $\alpha = \beta = 1$. Thus, Theorem 1 yields the following, which improves upon the competitive ratio of 4 from Garg et al. [8].

Corollary 1. *There exists an exponential time 3-competitive algorithm for the concurrent open shop setting.*

In polynomial time, Garg et al. [8] provide a $(2, 2)$-approximation for the MUWP, and Mastrolilli et al. [18] provide a 2-approximation for offline version of $PD \,|r_j = 0|\, \sum_j w_j C_j$. These results with Theorem 1 improve the ratio of 16 by Garg et al. [8]. We note that the additional additive term of αW in Theorem 1 is smaller than the additive $3W$ term in the guarantees of Garg et al. [8], for both the exponential-time and polynomial-time cases.

Corollary 2. *There exists a polynomial-time 10-competitive algorithm for the concurrent open shop setting.*

When the number of machines m is constant, there exists a polynomial-time $(1 + \epsilon, 1)$-approximation algorithm for the MUWP (see full version of this paper [13]) by a reduction to the multidimesional knapsack problem. Furthermore, when m is fixed, Cheng et al. [7] gave a PTAS for the offline $PD \,|r_j = 0|\, \sum_j w_j C_j$. Combining these results with Theorem 1 yields the following.

Corollary 3. *There exists a polynomial time, $(3 + \epsilon)$-competitive algorithm for $PD \,||\, \sum w_j C_j$ when the number of machines is fixed.*

4 Applications to Coflow Scheduling

We now apply our framework to coflow scheduling, introduced by Chowdhury and Stoica [5]. We are given a non-blocking network with m input ports and m output ports. A *coflow* is a collection of parallel flows processed by the network. We represent a coflow j by an $m \times m$ matrix $D^j = (d^j_{io})_{i,o \in [m]}$, where d^j_{io} denotes the integer amount of data to be transferred from input port i to output port o for coflow j. Each port can process at most one unit of data per time unit, and we assume that the transfer of data within the network is instantaneous.

The problem is to schedule a set of n coflows, each with a non-negative weight w_j and release time r_j, that minimizes the sum of weighted completion times, where the completion time of a coflow is the earliest time at which all of its flows have been processed. We denote this problem by Coflow $|| \sum_j w_j C_j$.

As in Sect. 3, in exponential time, we can iterate over all subsets of coflows to optimally solve the MUWP, giving a $(1, 1)$-approximation. Moreover, Im and Purohit [12] proposed a $(2 + \epsilon)$-approximation for offline coflow scheduling[5].

Corollary 4. *There exists an exponential-time* $(4+\epsilon)$*-competitive algorithm for online coflow scheduling.*

Furthermore, we can show that the polynomial-time $(2, 2)$-approximation for the MUWP for $PD \,||\, \sum_j w_j C_j$ of Garg et al. [8] can be applied to coflow scheduling with the same approximation guarantees. Combined with the 4-approximation of Ahmadi et al. [1], our framework yields the following.

Corollary 5. *There exists a polynomial-time* $(10+\epsilon)$*-competitive algorithm for online coflow scheduling.*

To show that the $(2, 2)$-approximation for the MUWP for $PD \,||\, \sum_j w_j C_j$ of Garg et al. [8] can be applied to coflow scheduling with the same approximation guarantees, we recall the reduction from Coflow $||\, \sum_j w_j C_j$ to $PD \,||\, \sum_j w_j C_j$ given by Khuller and Purohit [14]. Given an instance of coflow scheduling $\mathcal{I}$, let $L_i^j = \sum_{o=1}^{m} d_{io}^j$ denote the total amount of data that co-flow j needs to transmit through input port i, and similarly, we let $L_o^j = \sum_{i=1}^{m} d_{io}^j$ or output port o. From this, create a concurrent open shop instance $\mathcal{I}'$ with a set M of $2m$ machines (one for each port) and a set J of n jobs (one for each coflow), with processing times p_{sj} set equal to L_s^j for job j on machine s.

Now, the MUWP on $\mathcal{I}'$ can be formulated by the following integer program of Garg et al. [8].

$$\begin{aligned} \text{minimize} \quad & \sum_{j \in J} w_j (1 - x_j) \\ \text{subject to} \quad & \sum_{j \in J} p_{ij} x_j \leq D \quad \forall i \in M \\ & x_j \in \{0, 1\} \quad \forall j \in J \ . \end{aligned}$$

Let W' denote the optimal unscheduled weight for the MUWP on $\mathcal{I}'$, and define W similarly. The algorithm of Garg et al. [8] solves the linear relaxation of this integer program to obtain an optimal fractional solution $\bar{x}$, and outputs the set of jobs $S' = \{j \in J \mid \bar{x}_j \geq \frac{1}{2}\}$. Letting W^* denote the objective function value of an optimal solution of the LP relaxation, it is straightforward to check that the total processing time of S' on any machine is at most $2D$, the total unscheduled weight is at most $2W^*$, and $W^* \leq W'$. Hence, the algorithm of Garg et al. [8] is indeed a $(2, 2)$-approximation for the MUWP in the concurrent open shop environment.

[5] Since permutation schedules are not necessarily optimal for coflow scheduling [6], even finding a *factorial*-time optimal algorithm is nontrivial. For simplicity, we have chosen to use a polynomial-time algorithm to achieve Corollary 4.

Lemma 4. *The optimal unscheduled weight for the MUWP on $\mathcal{I}'$ is at most that for the MUWP on $\mathcal{I}$; i.e., $W' \leq W$.*

Proof. The proof is essentially identical to that of Lemma 1 in [14]. Let S be the optimal solution to the MUWP for $\mathcal{I}$. Then there exists a schedule of the coflows in S such that every coflow completes by the deadline D. Now consider the set S' of corresponding jobs in $\mathcal{I}'$. Processing job $j \in S'$ on machine s whenever data is being processed for coflow $j \in S$ on port s in the schedule for S gives a schedule for S' in which every job also completes by deadline D. Thus S' is a feasible solution to the MUWP for $\mathcal{I}'$ with objective function value equal to that of the optimal solution S to the MUWP for $\mathcal{I}$, and the claim follows. □

Let S be the set of coflows in $\mathcal{I}$ corresponding to the jobs S' defined above.

Lemma 5. *In polynomial time, we can find a schedule for S that completes by time $2D$, and whose total unscheduled weight is at most $2W$.*

Proof. We know that for any machine s in $\mathcal{I}'$, $\sum_{j \in S'} p_{sj} = \sum_{j \in S} L_s^j \leq 2D$. Thus, if we take any schedule for the coflows S without idle time, every port s finishes processing data by time $\sum_{j \in S} L_s^j \leq 2D$. Since all coflows complete at the same time when all ports have finished processing the data, we get a schedule in which all coflows in S will complete without idle time by time $2D$.

The total unscheduled weight in $\mathcal{I}$ is the same as the total unscheduled weight in $\mathcal{I}'$. By Lemma 4, the total unscheduled weight in $\mathcal{I}$ is at most $2W' \leq 2W$. □

Hence, the $(2, 2)$-approximation for the MUWP for $PD \,||\, \sum_j w_j C_j$ of Garg et al. [8] can be applied to Coflow $||\, \sum_j w_j C_j$ with the same guarantees.

5 A Randomized Online Scheduling Framework

In this section, we show how our ideas can be combined with the randomized framework of Chakrabarti et al. [3] to develop an analogue of the deterministic framework of Sect. 2.

The framework of Chakrabarti et al. [3] modify that of Hall et al. [10] (see Sect. 2.1) by setting $\tau_k = \eta 2^k$, where $\eta \in [\frac{1}{2}, 1)$ is a randomly chosen parameter. After making this choice, the online algorithm proceeds exactly as before, by applying the dual ρ-approximation to the MSWP at each interval.

Let $C_j^{\mathcal{OPT}}$ denote the completion time of job j in an optimal schedule, and let B_j denote the start of the interval $(\tau_{k-1}, \tau_k]$ in which job j completes. Chakrabarti et al. [3] show that if one takes $\eta = 2^{-X}$, where X is chosen uniformly at random from $(0, 1]$, then the following holds.

Lemma 6 ([3]). $E[B_j] = \frac{1}{2 \ln 2} C_j^{\mathcal{OPT}}$.

Hall et al. [10] showed how to produce a schedule of total weighted completion time at most $4\rho \sum_j w_j B_j$. By linearity of expectation, one can apply Lemma 6, so that the schedule produced has total weighted completion at most

$\frac{2}{\ln 2}\rho \sum_j w_j C_j^{\mathcal{OPT}}$, resulting in a randomized $\frac{2}{\ln 2}\rho \leq 2.89\rho$-competitive algorithm.

We can directly adapt this idea of randomly choosing the interval end points in our minimization framework to develop a randomized version of Theorem 1. Specifically, we take $\tau_k = \eta 2^k$, using the same η above, and run the framework described in Sect. 2.3 using this new choice of interval end points. Note that Lemma 1, Lemma 2, and Lemma 3 still hold with our new choice of τ_k.

Let $\mathcal{A}'$ denote this randomized algorithm and using the same notation as in Sect. 2.3, we can achieve the following result.

Theorem 2. *Algorithm $\mathcal{A}'$ is $(\frac{1}{\ln 2}\alpha\beta + \gamma)$-competitive in expectation, with an additive αW term.*

Proof. The same steps as in the proof of Theorem 1 yield

$$\sum_{j \in J} w_j C_j^{\mathcal{A}'} \leq 2\alpha\beta \sum_{k=1}^{L} (\tau_k - \tau_{k-1}) W_{\tau_k}^{\mathcal{OPT}} + \alpha W + \sum_{j \in J} w_j \delta_j \ .$$

By definition of B_j, we notice that

$$\sum_{k=1}^{L} (\tau_k - \tau_{k-1}) W_{\tau_k}^{\mathcal{OPT}} = \sum_{j \in J} w_j B_j \ .$$

By linearity of expectation and Lemma 6, we conclude that

$$E[\sum_{j \in J} w_j C_j^{\mathcal{A}'}] \leq (\frac{1}{\ln 2}\alpha\beta + \gamma) \sum_{j \in J} w_j C_j^{\mathcal{OPT}} + \alpha W \ .$$

□

Using the guarantee of Theorem 2, we can instantiate this framework in various scheduling settings and find values for α, β, γ to achieve improved competitive ratios over our deterministic framework. The results obtained when applying the same subroutines as we did for the framework of Sect. 2.3 are in Table 1.

Acknowledgements. We would like to thank Sungjin Im and Clifford Stein for directing us to [3, 17], and William Gasarch for organizing the REU program.

References

1. Ahmadi, S., Khuller, S., Purohit, M., Yang, S.: On scheduling co-flows. To appear in the proceedings of the 19th Conference on Integer Programming and Combinatorial Optimization (2017)
2. Bansal, N., Khot, S.: Inapproximability of hypergraph vertex cover and applications to scheduling problems. In: ICALP. pp. 250–261. Springer (2010)
3. Chakrabarti, S., Phillips, C.A., Schulz, A.S., Shmoys, D.B., Stein, C., Wein, J.: Improved scheduling algorithms for minsum criteria. In: ICALP. pp. 646–657. Springer (1996)

4. Chen, Z.L., Hall, N.G.: Supply chain scheduling: Assembly systems. Tech. rep., University of Pennsylvania (2000)
5. Chowdhury, M., Stoica, I.: Coflow: A networking abstraction for cluster applications. In: HotNets. pp. 31–36 (2012)
6. Chowdhury, M., Zhong, Y., Stoica, I.: Efficient coflow scheduling with Varys. In: ACM SIGCOMM CCR. vol. 44, pp. 443–454. ACM (2014)
7. Edwin Cheng, T., Nong, Q., Ng, C.T.: Polynomial-time approximation scheme for concurrent open shop scheduling with a fixed number of machines to minimize the total weighted completion time. Naval Research Logistics 58(8), 763–770 (2011)
8. Garg, N., Kumar, A., Pandit, V.: Order scheduling models: hardness and algorithms. In: FSTTCS. pp. 96–107. Springer (2007)
9. Graham, R., Lawler, E., Lenstra, J., Kan, A.R.: Optimization and approximation in deterministic sequencing and scheduling : a survey. Ann. Discrete Math. 5, 287–326 (1979)
10. Hall, L.A., Schulz, A.S., Shmoys, D.B., Wein, J.: Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. Math. Oper. Res. 22(3), 513–544 (1997)
11. Hung, C.C., Golubchik, L., Yu, M.: Scheduling jobs across geo-distributed datacenters. In: SoCC. pp. 111–124. ACM (2015)
12. Im, S., Purohit, M.: A tight approximation for co-flow scheduling for minimizing total weighted completion time. arXiv preprint arXiv:1707.04331 (2017)
13. Khuller, S., Li, J., Sturmfels, P., Sun, K., Venkat, P.: Select and permute: An improved online framework for scheduling to minimize weighted completion time. arXiv preprint arXiv:1704.06677 (2017)
14. Khuller, S., Purohit, M.: Brief announcement: Improved approximation algorithms for scheduling co-flows. In: SPAA. pp. 239–240. ACM (2016)
15. Leung, J.Y.T., Li, H., Pinedo, M.: Scheduling orders for multiple product types to minimize total weighted completion time. Discrete Appl. Math. 155(8), 945–970 (2007)
16. Li, Y., Jiang, S.H.C., Tan, H., Zhang, C., Chen, G., Zhou, J., Lau, F.: Efficient online coflow routing and scheduling. In: Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing. pp. 161–170. ACM (2016)
17. Lübbecke, E., Maurer, O., Megow, N., Wiese, A.: A new approach to online scheduling: Approximating the optimal competitive ratio. ACM Transactions on Algorithms (TALG) 13(1), 15 (2016)
18. Mastrolilli, M., Queyranne, M., Schulz, A.S., Svensson, O., Uhan, N.A.: Minimizing the sum of weighted completion times in a concurrent open shop. Oper. Res. Lett. 38(5), 390–395 (2010)
19. Murray, R., Chao, M., Khuller, S.: Scheduling distributed clusters of parallel machines: Primal-dual and LP-based approximation algorithms. In: ESA (2016)
20. Qiu, Z., Stein, C., Zhong, Y.: Minimizing the total weighted completion time of coflows in datacenter networks. In: SPAA. pp. 294–303. ACM (2015)
21. Roemer, T.A.: A note on the complexity of the concurrent open shop problem. J. Sched. 9(4), 389–396 (2006)
22. Sachdeva, S., Saket, R.: Optimal inapproximability for scheduling problems via structural hardness for hypergraph vertex cover. In: CCC. pp. 219–229. IEEE (2013)
23. Wang, G., Cheng, T.E.: Customer order scheduling to minimize total weighted completion time. Omega 35(5), 623–626 (2007)