Scheduling Coflows in Datacenter Networks: Improved Bound for Total Weighted Completion Time

Mehrnoosh Shafiee and Javad Ghaderi

***Abstract*—*Coflow* is a recently proposed networking abstraction to capture communication patterns in data-parallel computing frameworks. We consider the problem of efficiently scheduling coflows with release dates in a shared datacenter network so as to minimize the total weighted completion time of coflows. Specifically, we propose a randomized algorithm with approximation ratio of $3e \approx 8.155$, which improves the prior best known ratio of $9 + 16\sqrt{2}/3 \approx 16.542$. For the special case when all coflows are released at time zero, we obtain a randomized algorithm with approximation ratio of $2e \approx 5.436$ which improves the prior best known ratio of $3 + 2\sqrt{2} \approx 5.828$. Simulation result using a real traffic trace is presented that shows improvement over the prior approaches.**

***Index Terms*—Scheduling Algorithms, Approximation Algorithms, Coflow, Datacenter Network**

I. Introduction

Many data-parallel computing applications (e.g. MapReduce [1]) consist of multiple computation stages. Intermediate parallel data is often produced at various stages which needs to be transferred among servers in the datacenter network. A stage often cannot start or be completed unless all the required data pieces from the preceding stage are received. Hence, the collective effect of the data flows between the stages is more important for the performance (e.g. latency, job completion time) than that of any of the individual flows. Recently Chowdhury and Stoica [2] have introduced the *coflow* abstraction to capture these communication patters. *A coflow is defined as a collection of parallel flows whose completion time is determined by the completion time of the last flow in the collection.*

A Smallest-Effective-Bottleneck-First heuristic was introduced in Varys [3] for the problem of coflow scheduling in order to minimize the weighted sum of completion times of coflows in the system. This problem has been shown [4], [5] to be NP-complete through its connection with the concurrent open shop problem [3], [4]. Table I summarizes the best known approximation ratio results for this problem [4], [5]. We propose a deterministic algorithm that gives the same bound as the prior best known bound, however, our approach improves the best known bound for the randomized algorithm in both cases of with and without release dates.

2-page abstract of this work is published in SIGMETRICS 2017.
This work is supported by NSF Grants CNS-1652115 and CNS-1565774.

TABLE I: Performance Guarantees (Approximation Ratios)

Case	Best known	This paper
Deterministic, without release dates	8 [5]	8
Deterministic, with release dates	12 [5]	12
Randomized, without release dates	$3+2\sqrt{2}$ [5]	$2e$
Randomized, with release dates	$9+16\sqrt{2}/3$ [4]	$3e$

II. Model and Problem Statement

Similar to [3], [4], we abstract out the datacenter network as one giant $N \times N$ non-blocking switch, with N input links connected to N source servers and N output links connected to N destination servers. We assume all the link capacities are equal and normalized to one.

We assume there is a set of K coflows denoted by $\mathcal{K}$. Coflow $k \in \mathcal{K}$ can be denoted as an $N \times N$ matrix $D^{(k)}$ which is released (arrives) at time r_k that means it can only be scheduled after time r_k. Every flow is a triple (i, j, k), where $i \in \mathcal{I}$ is its source node, $j \in \mathcal{J}$ is its destination node, and k is the coflow to which it belongs. The size of flow (i, j, k) is denoted by d_{ij}^k, which is the (i, j)-th element of the matrix $D^{(k)}$. For a source node i and a coflow $k \in \mathcal{K}$, we define $d_i^k = \sum_{1 \le j \le N} d_{ij}^k$, which is the aggregate flow that node i needs to transmit for coflow k. d_j^k is defined similarly for destination node $j \in \mathcal{J}$ and coflow $k \in \mathcal{K}$.

We use f_k to denote the completion time of coflow k, which, by definition of coflow, is the time when all its flows have finished processing. In other words, for every coflow $k \in \mathcal{K}$, $f_k = \max_{1 \le i,j \le N} f_{ij}^k$, where f_{ij}^k is the completion time of flow (i, j, k).

For given positive weights w_k, $k \in \mathcal{K}$, the goal is to minimize the weighted sum of coflow completion times, i.e., $\sum_{k \in \mathcal{K}} w_k f_k$, subject to capacity and release dates constraints.

III. Linear Programing Relaxation

In this section, we use *linear ordering variables* (see, e.g., [6]–[9]) to present a relaxed linear program of coflow scheduling problem. For any two coflows k, k', we introduce a binary variable $\delta_{kk'} \in \{0, 1\}$ such that $\delta_{kk'} = 1$ if coflow k is finished before coflow k', and it is 0 otherwise. In the linear program relaxation, we allow the ordering variables to be fractional. This yields the following relaxed LP:

$$(\mathbf{LP})\ \min \sum_{k=1}^{K} w_k f_k \tag{1a}$$

$$\text{subject to: } f_k \geq d_i^k + \sum_{k' \in \mathcal{K}} d_i^{k'} \delta_{k'k} \quad 1 \leq i \leq N, k \in \mathcal{K} \tag{1b}$$

$$f_k \geq d_j^k + \sum_{k' \in \mathcal{K}} d_j^{k'} \delta_{k'k} \quad 1 \leq j \leq N, k \in \mathcal{K} \tag{1c}$$

$$f_k \geq W(k) + r_k \quad k \in \mathcal{K} \tag{1d}$$

$$\delta_{kk'} + \delta_{k'k} = 1 \quad k, k' \in \mathcal{K} \tag{1e}$$

$$\delta_{kk'} \in [0,1] \quad k, k' \in \mathcal{K}. \tag{1f}$$

Where, $W(k) = \max\{\max_{1 \leq i \leq N} d_i^k, \max_{1 \leq j \leq N} d_j^k\}$ is defined as the effective size of coflow k. The constraint (1b) (similarly (1c)) follows from the definition of ordering variables and the fact that flows incident to a source node i (a destination node j) are processed by a single link of unit capacity. The fact that each coflow cannot be completed before its release date plus its effective size is captured by constraint (1d). The next constraint (1e) indicates that for each two incident coflows, one precedes the other.

We use $\tilde{f}_k$ to denote the optimal solution to this LP for the completion time of coflow k. Also we use $\widetilde{\text{OPT}} = \sum_k w_k \tilde{f}_k$ to denote the corresponding objective value. Similarly we use $f_k^\star$ to denote the completion time of coflow k in some optimal scheduling, and use OPT $= \sum_k w_k f_k^\star$ to denote its optimal objective value. The following lemma establishes a relation between $\widetilde{\text{OPT}}$ and OPT.

Lemma 1. $\sum_{k=1}^{K} w_k \tilde{f}_k \leq \sum_{k=1}^{K} w_k f_k^\star$.

Proof. Consider an optimal solution. We set ordering variables so as $\delta_{kk'} = 1$ if coflow k precedes coflow k' in this solution, and $\delta_{kk'} = 0$, otherwise. If both coflows finish their corresponding incident flows at the same time, we set either one to 1 and the other one to 0. We note that this set of ordering variables and coflow completion times satisfies constraints (1b) and (1c) (as a result of capacity constraint on communication links) and also constraint (1d). Furthermore, the rest of (LP) constraints are satisfied by the construction of ordering variables. Therefore, optimal solution can be converted to a feasible solution to (LP). Hence, the optimal value of LP, $\widetilde{OPT}$, is at most equal to OPT. □

IV. Algorithm Description and Main Results

Both deterministic and randomized algorithms have three steps: (i) solve the relaxed LP (1), (ii) use the solution of the relaxed LP to partition coflows into disjoint subsets $C_0, C_1, \cdots, C_L$ for some L to be determined, and (iii) treat each subset C_l, $l = 0, \cdots, L$, as a single coflow and schedule its flows in a way that optimizes its completion time.

Partition Rule: We define $\gamma = \min_{i,j,k} d_{ij}^k$ and $T = \max_k r_k + \sum_k \sum_i \sum_j d_{ij}^k$. Given $\beta \geq 1$ (to be optimized), we choose L to be the smallest integer such that $\gamma \beta^{L+\alpha} \geq T$, where in the deterministic algorithm, $\alpha = 0$ and in the randomized algorithm α is a number chosen uniformly at random from $[0, 1)$. Consequently, define $a_l = \gamma \beta^l$, for $l = -1, 0, 1, \cdots, L$. Then the l-th partition is defined as the interval $(a_{l-1}, a_l]$ in both algorithms and the set C_l is defined as the subset of coflows whose completion times $\tilde{f}_k$ (as the result of solving the LP (1)) fall within the l-th partition, i.e., $C_l = \{k \in \mathcal{K} : \tilde{f}_k \in (a_{l-1}, a_l]\};\ l = 0, 1, \cdots, L$.

Scheduling Each Subset: To schedule coflows of each subset C_l, $l = 1, \cdots, L$, we construct a single coflow by aggregating all the coflows of C_l, and then schedule its flows to optimize the completion time of this aggregate coflow.

Suppose a single coflow $D = (d_{ij})_{i,j=1}^N$ is given. Let $W(D)$ denote its effective size. We assign transmission rate $x_{ij} = d_{ij}/W(D)$ to flow (i, j) until it is completed. Note that this rate assignment respects the capacity constraints, since for all source nodes $i \in \mathcal{I}$:

$$\sum_{1 \leq j \leq N} x_{ij}(t) = \sum_{1 \leq j \leq N} \frac{d_{ij}}{W(D)} \leq \frac{W(D)}{W(D)} = 1,$$

and similarly for all destination nodes $j \in \mathcal{J}$. Also, by this rate assignment, all flows of D finish in $W(D)$ amount of time.

Now we state the main results in the following theorems.

Theorem 1. *When $\beta = 2$, the described deterministic algorithm is a 12-approximation algorithm for the problem of total weighted completion time minimization of coflows with release dates. Without release dates, it is an 8-approximation algorithm.*

Theorem 2. *When $\beta = e$, the described randomized algorithm is a ($3e$)-approximation algorithm. If all coflows are released at time 0, then it is a ($2e$)-approximation algorithm.*

V. Algorithm Analysis

To prove Theorems 1 and 2, we need the following lemma that characterizes the time to complete all the coflows in C_l.

Lemma 2. *Let τ_{C_l} be the amount of time spent on processing all the coflows in C_l. Then*

$$\tau_{C_l} \leq \max_{k: k \in C_l} r_k + W(C_l), \tag{2}$$

where, $W(C_l)$ is the effective size of the aggregate coflow constructed from coflows in partition C_l.

Proof. In order for all coflows of the set C_l to be released, we wait for at most $\max_{k:k \in C_l} r_k$ amount of time. Also, it is obvious that all corresponding flows are processed in $W(C_l)$ amount of time as a result of rate allocation of both deterministic and randomized algorithms. Thus, $\tau_{C_l} \leq \max_{k:k \in C_l} r_k + W(C_l)$. □

The following lemma demonstrates a relationship between completion time of coflows in C_l obtained from (LP) and $W(C_l)$.

Lemma 3. $\max_{k:k \in C_l} \tilde{f}_k \geq 1/2 W(C_l)$

Proof. Variant versions of this lemma were used in other scheduling problems (see e.g., [6]–[9]). We refer to these work for the proof. □

We note that $W(C_l)$ is a lower bound on the time that all coflows of partition C_l complete in a feasible scheduling (as a result of the capacity constraints). Lemma 3 asserts that by allowing ordering variables to be fractional, completion time of coflows in partition C_l obtained from (LP) is still lower bounded by half of $W(C_l)$. Combining this with Lemma 2, we get the following inequality regarding the completion time of each partition which helps us bound the completion time of coflows under our algorithms.

Corollary 1. *Let τ_{C_l} be the amount of time spent on processing all the coflows in C_l, and $\tilde{f}_k$ be the optimal solution to the LP (1). Then*

$$\tau_{C_l} \leq 3 \max_{k:k\in C_l} \tilde{f}_k.$$

Proof. Using Lemma 2, we have

$$\tau_{C_l} \leq \max_{k:k\in C_l} r_k + W(C_l).$$

Constraint (1d) ensures that $\max_{k:k\in C_l} r_k \leq \max_{k:k\in C_l} \tilde{f}_k$. Combining this with Lemma 3 gives the bound in the lemma's statement. □

A. Proof of Theorem 1

Proof of Theorem 1. Let $\hat{f}_k$ denote the actual completion time of coflow k under our deterministic algorithm. Also, let l_k be the index of the partition into which coflow k falls based on $\tilde{f}_k$. Then

$$\hat{f}_k \overset{(a)}{\leq} \sum_{p=0}^{l_k} \tau_{C_p} \overset{(b)}{\leq} \sum_{p=0}^{l_k} 3 \max_{s:s\in C_p} \tilde{f}_s \overset{(c)}{\leq} 3\sum_{p=0}^{l_k} a_p \qquad (3)$$

Inequality (a) bounds the completion time of coflow k with summation of the time the algorithm spends on previous partitions plus the time it spends on C_{l_k}. Corollary 1 implies inequality (b), and inequality (c) follows from the fact that $\tilde{f}_s$ falls in $(a_{p-1}, a_p]$ for every coflow $s \in C_p$. By the partitioning rule,

$$\sum_{p=0}^{l_k} a_p = \gamma \sum_{p=0}^{l_k} \beta^p = \gamma \frac{\beta^{(l_k+1)} - 1}{\beta - 1} < \frac{\beta}{\beta-1} a_{l_k} \qquad (4)$$

Combining (3) with (4),

$$\hat{f}_k < 3\frac{\beta}{\beta-1} a_{l_k} = 3\frac{\beta^2}{\beta-1} a_{l_k-1} \overset{(d)}{<} 3\frac{\beta^2}{\beta-1}\tilde{f}_k \qquad (5)$$

Where (d) is because $\tilde{f}_k$ falls in $(a_{l_k-1}, a_{l_k}]$. This inequality implies that

$$\sum_k w_k \hat{f}_k < 3\frac{\beta^2}{\beta-1}\sum_k w_k \tilde{f}_k = 3\frac{\beta^2}{\beta-1}\widetilde{OPT} \leq 3\frac{\beta^2}{\beta-1} OPT$$

The last inequality follows from Lemma 1. $\beta = 2$ gives 12-approximation ratio. When all coflows are released at time 0, $\tau_{C_l} = W(C_l) \leq 2\max_{k:k\in C_l} \tilde{f}_k$ in Corollary 1. This only changes the inequality (b) in (3) and the rest of the argument is similar. □

B. Proof of Theorem 2

Now, we analyze performance of our randomized algorithm. Note that, in this case, $\tilde{f}_k$ is a random variable, because as a result of the random partitioning of the time horizon, l_k, the partition that coflow k belongs to, is a random variable. Also a_{l_k} is a random variable.

Lemma 4. *Denote by l_k the partition that coflow k belongs to under random partitioning (α is uniformly distributed over $[0,1)$), then $\mathbb{E}[a_{l_k}] = \frac{\beta-1}{\ln\beta}\tilde{f}_k$.*

Proof. A similar result was proved and used in [8] for some other scheduling problem. We present an alternative proof here. From the partitioning rule described in Section IV, $a_{l_k-1} < \tilde{f}_k \leq a_{l_k}$, where $a_{l_k} = \gamma\beta^{l_k+\alpha}$ which implies that $l_k = \lceil \log_\beta(\tilde{f}_k/\gamma) - \alpha \rceil$. We define $y = \log_\beta(\tilde{f}_k/\gamma) - \lfloor \log_\beta(\tilde{f}_k/\gamma) \rfloor$ and assume $y > 0$. Therefore,

$$l_k = \begin{cases} \lceil \log_\beta(\tilde{f}_k/\gamma) \rceil & \text{if } \alpha \leq y \\ \lceil \log_\beta(\tilde{f}_k/\gamma) \rceil - 1 & \text{otherwise} \end{cases}$$

Hence,

$$\begin{aligned} \mathbb{E}[a_{l_k}] &= \int_0^y \gamma\beta^{\alpha + \lceil \log_\beta(\tilde{f}_k/\gamma)\rceil} d\alpha + \int_y^1 \gamma\beta^{\alpha+\lceil \log_\beta(\tilde{f}_k/\gamma)\rceil - 1} d\alpha \\ &= \gamma\beta^{\lceil \log_\beta(\tilde{f}_k/\gamma)\rceil} \int_{y-1}^{y} \beta^\alpha d\alpha \\ &= \gamma\frac{\beta-1}{\ln\beta} \times \beta^{\lceil \log_\beta(\tilde{f}_k/\gamma)\rceil + y - 1} \\ &\overset{*}{=} \gamma\frac{\beta-1}{\ln\beta} \times \frac{\tilde{f}_k}{\gamma} \\ &= \frac{\beta-1}{\ln\beta}\tilde{f}_k \end{aligned}$$

Equality (*) is implied by definition of y and its positivity. When $y = 0$, the argument is similar. □

Proof of Theorem 2. From the first part of inequality (5), we have the following bound on $\hat{f}_k$.

$$\hat{f}_k < 3\frac{\beta}{\beta-1} a_{l_k}$$

Lemma (4) together with linearity of expectation imply that,

$$\sum_k w_k \mathbb{E}\left[\hat{f}_k\right] \leq 3\frac{\beta}{\ln\beta}\sum_k w_k \tilde{f}_k \leq 3\frac{\beta}{\ln\beta} OPT \qquad (6)$$

The choice $\beta = e$ gives $3e$-approximation for the randomized algorithm. For the case that all coflows arrives at time 0, $\tau_{C_l} = W(C_l) \leq 2\max_{k:k\in C_l}\tilde{f}_k$, and the result is followed by replacing 3 with 2 in inequality (6). □

VI. Empirical Evaluation

Now, we present our simulation result to evaluate the performance of our algorithms.

Workload: The workload is based on a Hive/MapReduce trace at Facebook that was collected from a 3000-machine cluster with 150 racks and was also used in [3], [4]. Similar to [4], we filter the coflows based on *the number of their non-zero flows which we denote by M*. Apart from considering all coflows ($M \geq 1$), we consider three coflow collections filtered by the conditions $M \geq 10$, $M \geq 30$, and $M \geq 50$. Furthermore, the original cluster had a $10 : 1$ core-to-rack oversubscription ratio with a total bisection bandwidth of 300 Gbps. Hence, each link has a capacity of 128 MBps. To obtain the same traffic intensity offered to our network (without oversubscription), for the case of scheduling coflows with release dates, we need to scale down the arrival times of coflows by 10. For the case of without release dates, we assume that all coflows arrive at time 0.

Algorithms: We implement our deterministic and randomized algorithms with and without backfilling policy. The backfilling strategy works as follows: After assigning rates to aggregated coflow, we increase x_{ij}, rate of transmitting flows from source node i to destination node j, until either capacity of link i or link j is fully utilized. We continue until for any node, either source or destination node, the summation of rates sum to one. We also transmit flows respecting coflow order inside of each partition C_l. When there is no more service requirement on the pair of input i and output j for coflows of current partition, we backfill (transmit) in order from the flows on the same pair of ports from the subsequent coflows. We refer to this algorithm as 'LP-OV-BR', where OV stands for ordering variables, and BR indicates that scheduling is done by continuous rate control combined with backfilling. When there is no backfilling, the algorithm is referred to 'LP-OV-R'. We also simulate the algorithm in [4] combined with backfilling strategy as described in [4] and refer to it as 'LP-II-B'. Varys [3] is the last algorithm simulated for the means of performance comparison.

A. *Performance of Our Algorithms*

We report the ratios of the total weighted completion time obtained from our deterministic algorithm, with and without backfilling, and the optimal value of relaxed linear program (1), to verify Theorems 1. We also simulate our randomized algorithm with 10 random values of α for different cases, and present average ratio of the total weighted completion time divided by the optimal value of relaxed linear program (1) to verify Theorems 2. We also report the best performance ratio among 10 runs for each case. We only present results of the simulation in where we consider all coflows of the real traffic trace with equal weights and random weights. The results are more or less similar for other collections and all are in consistence with our theoretical results.

Table II shows the performance ratio of the deterministic algorithm for the case that all coflows are released at time 0. All performances are consistent with our theoretical result indicating that the approximation ratio is at most 8. In fact, the ratios are much smaller than 8 and is at most 1.94, when no backfilling is used and coflows have random weights. We note that the effect of backfilling is less than 8%.

Tables III shows the results for the case of release dates. All the ratios are less that 12 which verifies Theorem 1. The effect of backfilling is more profound in this case and is in the range of 10% to 18%.

TABLE II: Performance Ratio of Deterministic Algorithm, Without Release Dates

Algorithm	Equal weights	Random weights
LP-OV-R	1.91	1.94
LP-OV-BR	1.79	1.80

TABLE III: Performance Ratio of Deterministic Algorithm, With Release Dates

Algorithm	Equal weights	Random weights
LP-OV-R	1.53	1.59
LP-OV-BR	1.35	1.44

Average and the best performance ratio of our randomized algorithm over 10 random choices of α for the case when all coflows are released at time 0 and the case of general release dates are presented in Tables IV and V, respectively. In each cell, the first number is average performance ratio and the second number is the best one. We note that, although the best performance ratio (best choice of α) is always smaller than performance ratio of the deterministic algorithm, this is not true for the average ratio for all the cases.

B. *Performance Comparison with Other Algorithms*

Now, we compare performance of deterministic algorithm combined with backfilling with LP-II-B and Varys. We ran simulations for the four collections of coflows described. We set all the weights of coflows to be equal to one, and normalize the total completion time under each algorithm by the total completion time under LP-OV-BR.

Figure 1 shows the performance of different algorithms for different collections of coflows when all coflows are released at time 0. LP-OV-BR outperforms Varys for $28-32\%$ in different collections. It also constantly outperforms LP-II-B for about $6-7\%$. As we see appropriate ordering and grouping of coflows are paramount in decreasing the completion times. Figure 2 shows the performance of different algorithms for different collection of coflows for the case of release dates. LP-OV-BR outperforms Varys for about 5%, 7%, 16%, and 9% for all coflows, $M \geq 10$, $M \geq 30$, $M \geq 50$, respectively. It also outperforms LP-II-B for 9% when we consider all coflows and about 4% in other cases.

TABLE IV: The Average and The Best Performance Ratio of Randomized Algorithm, Without Release Dates

Algorithm	Equal weights	Random weights
LP-OV-R	1.91/1.86	1.91/1.76
LP-OV-BR	1.83/1.74	1.79/1.73

TABLE V: The Average and The Best Performance Ratio of Randomized Algorithm, With Release Dates

Algorithm	Equal weights	Random weights
LP-OV-R	1.48/1.39	1.54/1.43
LP-OV-BR	1.42/1.34	1.48/1.33

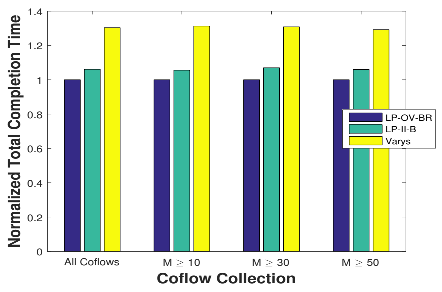

Fig. 1: Performance of LP-OV-BR, LP-II-B, and Varys when all coflows release at time 0, normalized with the performance of LP-OV-BR, under real traffic trace.

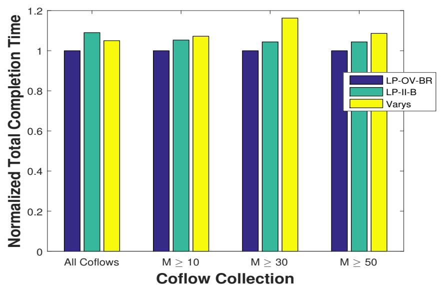

Fig. 2: Performance of LP-OV-BR, LP-II-B, and Varys in the case of release dates, normalized with the performance of LP-OV-BR, under real traffic trace.

VII. Concluding Remarks

In this paper, we studied the problem of scheduling of coflows with release dates to minimize their total weighted completion time, and proposed algorithms with improved approximation ratio. We also conducted extensive experiments to evaluate the performance of our algorithms, compared with two algorithms proposed before, using a real traffic trace. Our experimental results show that our algorithm combined with backfilling strategy yields improvement over the prior approaches.

As future work, other realistic constraints such as precedence requirement or deadline constraints need to be considered. Also, theoretical and experimental evaluation of the performance of the proposed online algorithm is left for future work. While we modeled the datacenter network as a giant non-blocking switch (thus focusing on rate allocation), the routing of coflows in the datacenter network is also of great importance for achieving the quality of service.

References

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

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

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

[4] Z. Qiu, C. Stein, and Y. Zhong, “Minimizing the total weighted completion time of coflows in datacenter networks,” in *Proceedings of the 27th ACM symposium on Parallelism in Algorithms and Architectures*. ACM, 2015, pp. 294–303.

[5] S. Khuller and M. Purohit, “Brief announcement: Improved approximation algorithms for scheduling co-flows,” in *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures*. ACM, 2016, pp. 239–240.

[6] C. Potts, “An algorithm for the single machine sequencing problem with precedence constraints,” in *Combinatorial Optimization II*. Springer, 1980, pp. 78–87.

[7] L. A. Hall, D. B. Shmoys, and J. Wein, “Scheduling to minimize average completion time: Off-line and on-line algorithms,” in *SODA*, vol. 96, 1996, pp. 142–151.

[8] R. Gandhi, M. M. Halldórsson, G. Kortsarz, and H. Shachnai, “Improved bounds for scheduling conflicting jobs with minsum criteria,” *ACM Transactions on Algorithms (TALG)*, vol. 4, no. 1, p. 11, 2008.

[9] M. Mastrolilli, M. Queyranne, A. S. Schulz, O. Svensson, and N. A. Uhan, “Minimizing the sum of weighted completion times in a concurrent open shop,” *Operations Research Letters*, vol. 38, no. 5, pp. 390–395, 2010.