Brief Announcement: Improved Approximation Algorithms for Scheduling Co-Flows*

Samir Khuller
Computer Science Department
University of Maryland, College Park
samir@cs.umd.edu

Manish Purohit
Computer Science Department
University of Maryland, College Park
manishp@cs.umd.edu

ABSTRACT

Co-flow scheduling is a recent networking abstraction introduced to capture application-level communication patterns in datacenters. In this paper, we consider the offline co-flow scheduling problem with release times to minimize the total weighted completion time. Recently, Qiu, Stein and Zhong [8] obtained the first constant approximation algorithms for this problem with a deterministic $\frac{67}{3}$-approximation and a randomized $(9 + \frac{16\sqrt{2}}{3}) \approx 16.54$-approximation. In this paper, we improve upon their algorithm to yield a deterministic 12-approximation algorithm. For the special case when all release times are zero, we obtain a deterministic 8-approximation and a randomized $(3 + 2\sqrt{2}) \approx 5.83$-approximation.

1. INTRODUCTION

Applications designed for data-parallel computation frameworks such as MapReduce, Hadoop, and Spark usually alternate between computation and communication stages. Typically, intermediate data generated by a computation stage needs to be transferred across machines during a communication stage (called "shuffle" in MapReduce) for further processing. Chowdhury and Stoica [2] introduce *co-flows* as a networking abstraction to represent the collective communication requirements of a job. Every job j is associated with a set of flow demands (called as a co-flow) and the job j is said to be satisfied once *all* of its demands are met.

Due to significant potential gains in datacenter throughput, co-flow scheduling has been a topic of active research [3, 4, 8, 10] since its introduction. Although the heuristics developed by Chowdhury et al [4, 3] perform very well in practice, they do not admit provable worst-case guarantees. Even in the offline setting, when all jobs are known in advance, no $O(1)$ approximation algorithm was known until recently. Qiu, Stein and Zhong [8] obtain a deterministic $\frac{67}{3}$ approximation and a randomized $(9 + \frac{16\sqrt{2}}{3})$ approximation for the problem of minimizing the weighted completion time. For the special case when all release times are zero, Qiu et al. [8] demonstrate improved bounds of $\frac{64}{3}$ (deterministic) and $(8 + \frac{16\sqrt{2}}{3})$ (randomized).

*This work is supported by NSF grant CCF 1217890.

SPAA '16 July 11-13, 2016, Pacific Grove, CA, USA

ACM ISBN 978-1-4503-4210-0/16/07.
DOI: http://dx.doi.org/10.1145/2935764.2935809

1.1 Problem Setting

A datacenter is modeled as a single $m \times m$ *non-blocking* switch, i.e., it is comprised of m *input ports* and m *output ports*. For simplicity, we assume that all ports have unit capacity - i.e., at most one unit of data can be transferred through any port at a time.

A co-flow is defined as a collection of parallel flow demands that share a performance goal. Each co-flow j has weight w_j, release time r_j, and is represented as a $m \times m$ integer matrix $D^j = [d^j_{io}]$ where the entry d^j_{io} represents the number of data units that must be transferred from input port i to output port o for co-flow j.

A co-flow j is available to be scheduled at its release time r_j and is said to be completed when all the flows in the matrix D^j have been scheduled. We assume that time is slotted and data transfer within the switch is instantaneous. Since each input port i can transmit at most one unit of data and each output port o can receive at most one unit of data in each time slot, a feasible schedule for a single time slot is described by a matching. Our goal is to find a feasible, integral schedule that minimizes the total, weighted completion time of the co-flows, i.e. minimize $\sum_j w_j C_j$.

1.2 Connection to Concurrent Open Shop

The co-flow scheduling problem as described above generalizes the well-studied concurrent open shop problem [7, 1, 5, 6, 9]. In the concurrent open shop problem, we have a set of m machines and each job j with weight w_j is composed of m tasks $\{t^j_i\}_{i=1}^m$, one on each machine. Let p^j_i denote the processing requirement of task t^j_i. A job j is said to be completed once all its tasks have completed. Any machine can perform at most one unit of processing at a time. The objective is to find a feasible schedule that minimizes the total weighted completion time of jobs. An LP-relaxation using completion time variables yields a 2-approximation algorithm for concurrent open shop scheduling when all release times are zero [1, 5, 6] and a 3-approximation algorithm for arbitrary release times [5, 6]. It can be seen that the concurrent open shop problem is a special case of co-flow scheduling when the demand matrices D^j for all co-flows j are diagonal [4, 8].

1.3 Our Contribution

The main algorithmic contribution of this paper is the following improved approximation guarantee for the offline co-flow scheduling problem.

THEOREM 1. *There exists a deterministic 12-approximation algorithm for co-flow scheduling with release times and a deterministic 8-approximation algorithm for co-flow scheduling without release times.*

THEOREM 2. *There exists a randomized* $(3+2\sqrt{2}) \approx 5.83$*-approximation algorithm for co-flow scheduling without release times.*

2. APPROXIMATION ALGORITHMS

For every co-flow j and input port i, we define the load $L_i^j = \sum_{o=1}^{m} d_{io}^j$ to be the total amount of data that co-flow j needs to transmit through port i. Similarly, we define $L_o^j = \sum_{i=1}^{m} d_{io}^j$ for every co-flow j and output port o. Our algorithm consists of the following two stages.

2.1 Reduction to Concurrent Open Shop:

Let $\mathcal{I}$ denote an instance of the co-flow scheduling problem. We now construct an instance $\mathcal{I}'$ of the concurrent open shop scheduling problem on $2m$ machines (one for each port) and n jobs (one for each co-flow). For a job j, set $p_s^j = L_s^j$, i.e., the processing requirement of job j on a machine s is set to be the load of the co-flow j on the corresponding port. Let $OPT(\mathcal{I})$ denote the cost of an optimal co-flow schedule and $OPT(\mathcal{I}')$ denote the cost of an optimal, preemptive concurrent open shop schedule for the instance $\mathcal{I}'$.

LEMMA 1. $OPT(\mathcal{I}') \leq OPT(\mathcal{I})$

PROOF. Let S^* denote an optimal co-flow schedule for instance $\mathcal{I}$. For a co-flow j and port s, let T_s^j denote the set of time slots when data corresponding to co-flow j is being processed (either input or output) at port s as per schedule S^*. Now processing one unit of the corresponding job j on machine s in the concurrent open shop instance $\mathcal{I}'$ at all times in T_s^j leads to a feasible schedule. □

Let $\bar{C}_j$ denote the completion time of job j in an approximate schedule for the concurrent open shop instance $\mathcal{I}'$. Further, let us assume without loss of generality that the co-flows are ordered so that the following holds.

$$\bar{C}_1 \leq \bar{C}_2 \leq \ldots \leq \bar{C}_n \qquad (1)$$

The following statements now hold from the feasibility of the schedule and Equation (1).

$$\bar{C}_k \geq r_k + \max_s p_s^k, \qquad 1 \leq k \leq n \qquad (2)$$

$$\bar{C}_k \geq \max_s \sum_{j \leq k} p_s^j, \qquad 1 \leq k \leq n \qquad (3)$$

COROLLARY 1. $\sum_j w_j \bar{C}_j \leq 3 \times OPT(\mathcal{I})$*. Further if all release times are zero, then* $\sum_j w_j \bar{C}_j \leq 2 \times OPT(\mathcal{I})$

PROOF. The concurrent open shop scheduling problem with release times has well-known 3-approximation algorithms [6, 5] that also yield a 2-approximation when all release times are zero. We remark that these approximation algorithms also yield guarantees with respect to the optimal *preemptive* schedule. Combining any of these algorithms with Lemma 1 yields the corollary. □

2. Scheduling Co-flows:

The following two lemmas by Qiu et al. [8] show that grouping co-flows in geometrically increasing groups based on the approximate completion times ($\bar{C}_j$) and then scheduling the consolidated co-flows sequentially yields a provably good co-flow schedule.

LEMMA 2 ([8]). *Given a permutation of co-flows that satisfies conditions* (2) *and* (3)*, there exists a deterministic algorithm that yields a feasible co-flow schedule such that for every co-flow* k, $C_k(alg) \leq 4\bar{C}_k$ *where* $C_k(alg)$ *is the completion time of co-flow* k *in the co-flow schedule.*

LEMMA 3 ([8]). *Given a permutation of co-flows that satisfies condition* (3) *and* $r_k = 0$ *for all co-flows* k*, there exists a randomized algorithm that yields a feasible co-flow schedule such that for every co-flow* k, $C_k(alg) \leq (\frac{3}{2} + \sqrt{2})\bar{C}_k$.

Theorems 1 and 2 now follow from Corollary 1 and Lemmas 2 and 3 respectively.

3. REFERENCES

[1] Z.-L. Chen and N. G. Hall. Supply chain scheduling: Conflict and cooperation in assembly systems. *Operations Research*, 55(6):1072–1089, 2007.

[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*, pages 31–36. ACM, 2012.

[3] M. Chowdhury and I. Stoica. Efficient coflow scheduling without prior knowledge. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 393–406. ACM, 2015.

[4] M. Chowdhury, Y. Zhong, and I. Stoica. Efficient coflow scheduling with varys. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 443–454, New York, NY, USA, 2014. ACM.

[5] N. Garg, A. Kumar, and V. Pandit. Order scheduling models: Hardness and algorithms. In *FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science*, pages 96–107. Springer, 2007.

[6] J. Y.-T. Leung, H. Li, and M. Pinedo. Scheduling orders for multiple product types to minimize total weighted completion time. *Discrete Applied Mathematics*, 155(8):945–970, 2007.

[7] 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*, 38(5):390–395, 2010.

[8] 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*, SPAA '15, pages 294–303, New York, NY, USA, 2015. ACM.

[9] G. Wang and T. E. Cheng. Customer order scheduling to minimize total weighted completion time. *Omega*, 35(5):623–626, 2007.

[10] Y. Zhao, K. Chen, W. Bai, M. Yu, C. Tian, Y. Geng, Y. Zhang, D. Li, and S. Wang. Rapier: Integrating routing and scheduling for coflow-aware data center networks. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 424–432. IEEE, 2015.