Scheduling with Communication Delays via LP Hierarchies and Clustering

Sami Davies*, Janardhan Kulkarni†, Thomas Rothvoss*, Jakub Tarnawski† and Yihao Zhang*

*

University of Washington, Seattle, USA
Email: {daviess, rothvoss, yihaoz93 }@uw.edu

†

Micorsoft Research, Redmond, USA
Email: {jakul, jatarnaw}@microsoft.com.

***Abstract*—We consider the classic problem of scheduling jobs with precedence constraints on identical machines to minimize makespan, in the presence of *communication delays*. In this setting, denoted by $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$, if two dependent jobs are scheduled on different machines, then at least c units of time must pass between their executions. Despite its relevance to many applications, this model remains one of the most poorly understood in scheduling theory. Even for a special case where an unlimited number of machines is available, the best known approximation ratio is $2/3 \cdot (c+1)$, whereas Graham's greedy list scheduling algorithm already gives a $(c+1)$-approximation in that setting. An outstanding open problem in the top-10 list by Schuurman and Woeginger and its recent update by Bansal asks whether there exists a constant-factor approximation algorithm.**

In this work we give a polynomial-time $O(\log c \cdot \log m)$-approximation algorithm for this problem, where m is the number of machines and c is the communication delay. Our approach is based on a Sherali-Adams lift of a linear programming relaxation and a randomized clustering of the semimetric space induced by this lift.

The full version of this paper is available on arXiv.

***Index Terms*—scheduling; approximation algorithms; metric spaces; linear programming**

I. Introduction

Scheduling jobs with precedence constraints is a fundamental problem in approximation algorithms and combinatorial optimization. In this problem we are given m identical machines and a set J of n jobs, where each job j has a processing length $p_j \in \mathbb{Z}_+$. The jobs have precedence constraints, which are given by a partial order $\prec$. A constraint $j \prec j'$ encodes that job j' can only start after job j is completed. The goal is to find a schedule of jobs that minimizes *makespan*, which is the completion time of the last job. This problem is denoted[1] by $\mathsf{P} \mid \mathsf{prec} \mid C_{\mathsf{max}}$. In a seminal result from 1966, Graham [3] showed that the greedy list scheduling algorithm achieves a $\left(2 - \frac{1}{m}\right)$-approximation. By now, our understanding of the approximability of this basic problem is almost complete: it had been known since the late '70s, due to a result by Lenstra and Rinnooy Kan [4], that it is NP-hard to obtain better than $4/3$-approximation, and in 2010 Svensson [5] showed that, assuming a variant of the Unique Games Conjecture [6], it is NP-hard to get a $(2-\varepsilon)$-approximation for any $\varepsilon > 0$.

The above precedence-constrained scheduling problem models the task of distributing workloads onto multiple processors or servers, which is ubiquitous in computing. This basic setting takes the dependencies between work units into account, but not the data transfer costs between machines, which is critical in applications. A precedence constraint $j \prec j'$ typically implies that the input to j' depends on the output of j. In many real-world scenarios, especially in the context of scheduling in data centers, if j and j' are executed on different machines, then the *communication delay* due to transferring this output to the other machine cannot be ignored. This is an active area of research in applied data center scheduling literature, where several new abstractions have been proposed to deal with communication delays [7], [8], [9], [10], [11], [12], [13]. Another timely example is found in the parallelization of Deep Neural Network training (the machines being accelerator devices such as GPUs, TPUs, or FPGAs). There, when training the network on one sample/minibatch per device in parallel, the communication costs incurred by synchronizing the weight updates in fact dominate the overall running time [14]. Taking these costs into account, it turns out that it is better to split the network onto multiple devices, forming a "model-parallel" computation pipeline [15]. In the resulting *device placement* problem, the optimal split crucially depends on the communication costs between dependent layers/operators.

A classic model that captures the effect of data transfer latency on scheduling decisions is the problem of *scheduling jobs with precedence and communication delay constraints*, introduced by Rayward-Smith [16] and Papadimitriou and Yannakakis [17]. The setting, denoted by $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$, is similar to the makespan minimization problem described earlier, except for one crucial difference. Here we are given a *communication delay parameter* $c \in \mathbb{Z}_{\geq 0}$, and the output

*Thomas Rothvoss is supported by NSF CAREER grant 1651861 and a David & Lucile Packard Foundation Fellowship.

[1]Throughout the paper we use the standard scheduling three-field notation [1], [2]. The respective fields denote: **(1) number of identical machines:** $\mathsf{P}\infty$: unlimited; P: number m of machines given as input; $\mathsf{P}m$: constant number m of machines, **(2) job properties:** prec: precedence constraints; $p_j = 1$: unit-size jobs; c: communication delays of length c (can be c_{jk} if dependent on jobs $j \prec k$); $c-$intervals: see Section III; dup: allowed duplication of jobs, **(3) objective:** C_{max}: minimize makespan; $\sum w_j C_j$: minimize weighted sum of completion times.

2575-8454/20/$31.00 ©2020 IEEE
DOI 10.1109/FOCS46700.2020.00081

schedule must satisfy the property that if $j \prec j'$ and j, j' are scheduled on different machines, then j' can only start executing at least c time units after j had finished. On the other hand, if j and j' are scheduled on the same machine, then j' can start executing immediately after j finishes. In a closely related problem, denoted by $\mathsf{P}\infty \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$, a schedule can use as many machines as desired. The goal is to schedule jobs *non-preemptively* so as to minimize the makespan. In a non-preemptive schedule, each job j needs to be assigned to a single machine and executed during p_j consecutive timeslots. The problems $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$ and $\mathsf{P}\infty \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$ are the focus of this paper.

Despite its theoretical significance and practical relevance, very little is known about the communication delay setting. A direct application of Graham's [3] list scheduling algorithm yields a $(c+2)$-approximation, and no better algorithm is known for the problem. Over the years, the problem has attracted significant attention, but all known results, which we discuss below in Section I-D, concern special settings, small communication delays, or hardness of approximation. To put this in perspective, we note that the current best algorithm for general c [18], which achieves an approximation factor of $2/3 \cdot (c+1)$, only marginally improves on Graham's algorithm while requiring the additional assumptions that the number of machines is unbounded and $p_j = 1$. This is in sharp contrast to the basic problem $\mathsf{P} \mid \mathsf{prec} \mid C_{\mathsf{max}}$ (which would correspond to the case $c = 0$), where the approximability of the problem is completely settled under a variant of the Unique Games Conjecture. This situation hints that incorporating communication delays in scheduling decisions requires fundamentally new algorithmic ideas compared to the no-delay setting. Schuurman and Woeginger [19] placed the quest for getting better algorithms to the problem in their influential list of top-10 open problems in scheduling theory. In a recent MAPSP 2017 survey talk, Bansal [20] highlighted the lack of progress on this model, describing it as "not understood at all; almost completely open", and suggested that this is due to the lack of promising LP/SDP relaxations.

A. Our Contributions

The main result of this paper is the following:

Theorem 1. *There is a randomized $O(\log c \cdot \log m)$-approximation algorithm for $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$ with expected polynomial running time, where $c, p_j \in \mathbb{N}$.*

In any non-preemptive schedule the number m of machines is at most the number n of jobs, so for the easier $\mathsf{P}\infty$ version of the problem, the above theorem implies the following:

Corollary 2. *There is a randomized $O(\log c \cdot \log n)$-approximation algorithm for $\mathsf{P}\infty \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$ with expected polynomial running time, where $c, p_j \in \mathbb{N}$.*

For both problems one can replace either c or m by n, yielding a $O(\log^2 n)$-approximation algorithm. Our results make substantial progress towards resolving one of the questions in "Open Problem 3" in the survey of Schuurman and Woeginger [19], which asks whether a constant-factor approximation algorithm exists for $\mathsf{P}\infty \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$.

Our approach is based on a Sherali-Adams lift of a time-indexed linear programming relaxation for the problem, followed by a randomized clustering of the semimetric space induced by this lift. To our knowledge, this is the first instance of a multiple-machine scheduling problem being viewed via the lens of metric space clustering. We believe that our framework is fairly general and should extend to other problems involving scheduling with communication delays. To demonstrate the broader applicability of our approach, we also consider the objective of minimizing the weighted sum of completion times. Here each job j has a weight w_j, and the goal is to minimize $\sum_j w_j C_j$, where C_j is the completion time of j.

Theorem 3. *There is a randomized $O(\log c \cdot \log n)$-approximation algorithm for $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c \mid \sum_j w_j C_j$ with expected polynomial running time, where $c \in \mathbb{N}$.*

The proof of Theorem 3 can be found in the full version of the paper. No non-trivial approximation ratio was known for this problem prior to our work.

B. Independent work of Maiti et al

In a parallel and independent work, Maiti et al. [21] developed an $O(\log^2 n \log^2 m \log c / \log \log c)$-approximation algorithm for the makespan objective function on *related machines* ($Q \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$). Interestingly, they obtained the results using completely different techniques compared to ours. While our results are based on LP hierarchies and clustering, Maiti et al. [21] developed a novel framework based on *job duplication*. In their framework, they first construct a schedule where a single job can be scheduled on *multiple machines*, which is known to effectively "hide" the communication delay constraints [17]. Quite surprisingly, Maiti et al. [21] showed that one can convert a schedule with duplication to a feasible schedule without duplication, where every job is processed on a single machine, while increasing the makespan by at most an $O(\log^2 n \log m)$ factor.

C. Our Techniques

As we alluded earlier, there is a lack of combinatorial lower bounds for scheduling with communication delays. For example, consider Graham's list scheduling algorithm, which greedily processes jobs on m machines as soon as they become available. One can revisit the analysis of Graham [3] and show that there exists a chain Q of dependent jobs such that the makespan achieved by list scheduling is bounded by

$$\frac{1}{m}\sum_{j \in J} p_j + \sum_{j \in Q} p_j + c \cdot (|Q| - 1).$$

The first two terms are each lower bounds on the optimum — the 3rd term is not. In particular, it is unclear how to certify that the optimal makespan is high because of the communication delays. However, this argument suffices for a $(c+2)$-approximation, since $p_j \geq 1$ for all $j \in J$.

As pointed out by Bansal [20], there is no known promising LP relaxation. To understand the issue let us consider the special case $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c \mid C_{\mathsf{max}}$. Extending, for example, the LP of Munier and König [22], one might choose variables C_j as completion times, as well as decision variables x_{j_1,j_2} denoting whether j_2 is executed in the time window $[C_{j_1}, C_{j_1} + c)$ on the same machine as j_1. Then we can try to enforce communication delays by requiring that $C_{j_2} \geq C_{j_1} + 1 + (c-1) \cdot (1 - x_{j_1,j_2})$ for $j_1 \prec j_2$. Further, we enforce load constraints $\sum_{j_1 \in J} x_{j_1,j_2} \leq c$ for $j_2 \in J$ and $\sum_{j_2 \in J} x_{j_1,j_2} \leq c$ for $j_1 \in J$. To see why this LP fails, note that in any instance where the maximum dependence degree is bounded by c, one could simply set $x_{j_1,j_2} = 1$ and completely avoid paying any communication delay. Moreover, this problem seems to persist when moving to more complicated LPs that incorporate indices for time and machines.

A convenient observation is that, in exchange for a constant-factor loss in the approximation guarantee, it suffices to find an assignment of jobs to length-c intervals such that dependent jobs scheduled in the same length-c interval must be assigned to the same machine. (The latter condition will be enough to satisfy the communication delay constraints as, intuitively, between every two length-c intervals we will insert an empty one.) In order to obtain a stronger LP relaxation, we consider an $O(1)$*-round Sherali-Adams lift* of an inital LP with indices for time and machines. From the lifted LP, we extract a *distance function* $d : J \times J \to [0, 1]$ which satisfies the following properties:

(i) The function d is a *semimetric*.
(ii) $C_{j_1} + d(j_1, j_2) \leq C_{j_2}$ for $j_1 \prec j_2$.
(iii) Any set $U \subseteq J$ with a diameter of at most $\frac{1}{2}$ w.r.t. d, satisfies $|U| \leq 2c$.

Here we have changed the interpretation of C_j to the *index* of the length-c interval in which j will be processed. Intuitively, $d(j_1, j_2)$ can be understood as the probability that jobs j_1, j_2 are *not* being scheduled within the same length-c interval on the same machine. To see why a constant number of Sherali-Adams rounds are helpful, observe that the triangle inequality behind (i) is really a property depending only on *triples* $\{j_1, j_2, j_3\}$ of jobs and an $O(1)$-round Sherali-Adams lift would be locally consistent for every triple of variables.

We will now outline how to round such an LP solution. For jobs whose LP completion times are sufficiently different, say $C_{j_1} + \Theta(\frac{1}{\log(n)}) \leq C_{j_2}$, we can afford to deterministically schedule j_1 and j_2 at least c time units apart while only paying a $O(\log n)$-factor more than the LP. Hence the critical case is to sequence a set of jobs $J^* = \{j \in J \mid C^* \leq C_j \leq C^* + \Theta(\frac{1}{\log(n)})\}$ whose LP completion times are very close to each other. Note that by property (ii), we know that any dependent jobs $j_1, j_2 \in J^*$ must have $d(j_1, j_2) \leq \Theta(\frac{1}{\log(n)})$. As d is a semimetric, we can make use of the rich toolset from the theory of metric spaces. In particular, we use an algorithm by Calinescu, Karloff and Rabani [23]: For a parameter $\Delta > 0$, one can partition a semimetric space into *random clusters* so that the diameter of every cluster is bounded by Δ and each δ-neighborhood around a node is separated, meaning contains jobs assigned to different clusters, with probability at most $O(\log(n)) \cdot \frac{\delta}{\Delta}$. Setting $\delta := \Theta(\frac{1}{\log(n)})$ and $\Delta := \Theta(1)$ one can then show that a fixed job $j \in J^*$ will be in the same cluster as *all* its ancestors in J^* with probability at least $\frac{1}{2}$, while all clusters have diameter at most $\frac{1}{2}$. By (iii), each cluster will contain at most $2c$ many (unit-length) jobs, and consequently we can schedule all the clusters in parallel, where we drop any job that got separated from any ancestor. Repeating the sampling $O(\log n)$ times then schedules all jobs in J^*. This reasoning results in a $O(\log^2 n)$-approximation for this problem, which we call $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c\text{−}\mathsf{intervals} \mid C_{\mathsf{max}}$. With a bit of care the approximation factor can be improved to $O(\log c \cdot \log m)$.

Finally, the promised $O(\log c \cdot \log m)$-approximation for the more general problem $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$ follows from a reduction to the described special case $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c\text{−}\mathsf{intervals} \mid C_{\mathsf{max}}$.

D. History of the Problem

Precedence-constrained scheduling problems of minimizing the makespan and sum of completion times objectives have been extensively studied for many decades in various settings. We refer the reader to [24], [25], [26], [27], [28] for more details. Below, we only discuss results directly related to the communication delay problem in the offline setting.

a) Approximation algorithms.: As mentioned earlier, Graham's [3] list scheduling algorithm yields a $(c + 2)$-approximation for $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$, and a $(c + 1)$-approximation for the $\mathsf{P}\infty$ variant. For unit-size jobs and $c \geq 2$, Giroudeau, König, Moulai and Palaysi [18] improved the latter ($\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c \geq 2 \mid C_{\mathsf{max}}$) to a $\frac{2}{3}(c+1)$-approximation. For unit-size jobs and $c = 1$, Munier and König [22] obtained a $4/3$-approximation via LP rounding ($\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c = 1 \mid C_{\mathsf{max}}$); for the P variant, Hanen and Munier [29] gave an easy reduction from the $\mathsf{P}\infty$ variant that loses an additive term of 1 in the approximation ratio, thus yielding a $7/3$-approximation. Thurimella and Yesha [30] gave a reduction that, given an α-approximation algorithm for $\mathsf{P}\infty \mid \mathsf{prec}, c, p_j = 1 \mid C_{\mathsf{max}}$, would yield a $(1 + 2\alpha)$-approximation algorithm for $\mathsf{P} \mid \mathsf{prec}, c, p_j = 1 \mid C_{\mathsf{max}}$.

For a constant number of machines, a hierarchy-based approach of Levey and Rothvoss [31] for the no-delay setting ($\mathsf{P}m \mid \mathsf{prec}, p_j = 1 \mid C_{\mathsf{max}}$) was generalized by Kulkarni, Li, Tarnawski and Ye [32] to allow for communication delays that are also bounded by a constant. For any $\varepsilon > 0$ and $\hat{c} \in \mathbb{Z}_{\geq 0}$, they give a nearly quasi-polynomial-time $(1 + \varepsilon)$-approximation algorithm for $\mathsf{P}m \mid \mathsf{prec}, p_j = 1, c_{jk} \leq \hat{c} \mid C_{\mathsf{max}}$. The result also applies to arbitrary job sizes, under the assumption that preemption of jobs is allowed, but migration is not.

b) Hardness.: Hoogeveen, Lenstra and Veltman [33] showed that even the special case $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c = 1 \mid C_{\mathsf{max}}$ is NP-hard to approximate to a factor better than $7/6$. For the case with bounded number of machines (the P variant) they show 5/4-hardness. These two results can be generalized for $c \geq 2$ to $(1+1/(c+4))$-hardness [18] and $(1+1/(c+3))$-hardness [34], respectively.[2]

c) Duplication model.: The communication delay problem has also been studied in a setting where jobs can be duplicated (replicated), i.e., executed on more than one machine, in order to avoid communication delays. This assumption seems to significantly simplify the problem, especially when we are also given an unbounded number of machines: already in 1990, Papadimitriou and Yannakakis [17] gave a rather simple 2-approximation algorithm for $\mathsf{P}\infty \mid \mathsf{prec}, p_j, c_{jk}, \mathsf{dup} \mid C_{\max}$. Observe that this result holds even when communication delays are unrelated. The only non-trivial approximation algorithm for arbitrary c and a bounded number of machines is due to Lepere and Rapine [35], who gave an asymptotic $O(\log c/\log\log c)$-approximation for $\mathsf{P} \mid \mathsf{prec}, p_j = 1, c, \mathsf{dup} \mid C_{\max}$. On the hardness side, Papadimitriou and Yannakakis [17] showed NP-hardness of $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c, \mathsf{dup} \mid C_{\max}$ (using a large delay $c = \Theta(n^{2/3})$).

Many further references can be found in [2], [18], [36], [37], [38], [39], [40].

II. Preliminaries

A. The Sherali-Adams Hierarchy for LPs with Assignment Constraints

We review the *Sherali-Adams hierarchy*, which provides an automatic strengthening of linear relaxations for 0/1 optimization problems. The authorative reference is Laurent [41], and we adapt the notation from Friggstad et al. [42]. Consider a set of variable indices $[n] = \{1, \dots, n\}$ and let $U_1, \dots, U_N \subseteq [n]$ be subsets of variable indices. We consider a polytope

$$K = \Big\{x \in \mathbb{R}^n \mid \tilde{A}x \geq \tilde{b}, \\ \sum_{i \in U_k} x_i = 1 \ \ \forall k \in [N], \ \ 0 \leq x_i \leq 1 \ \ \forall i \in [n]\Big\},$$

which we write compactly as $K = \{x \in \mathbb{R}^n \mid Ax \geq b\}$ with $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. We explicitly included the "box constraints" $0 \leq x_i \leq 1$ for all variables i. Moreover, the constraint matrix contains *assignment constraints* of the form $\sum_{i \in U_k} x_i = 1$; this aspect of our presentation is non-standard.

The general goal is to obtain a strong relaxation for the integer hull $\mathrm{conv}(K \cap \{0,1\}^n)$. Observe that any point $x \in \mathrm{conv}(K \cap \{0,1\}^n)$ can be interpreted as a *probability distribution* X over points $K \cap \{0,1\}^n$. We know that any distribution can be described by the 2^n many values $y_I = \Pr[\bigwedge_{i \in I}(X_i = 1)]$ for $I \subseteq [n]$ — in fact, the probability of any other event can be reconstructed using the *inclusion-exclusion formula*. This is an exact, yet inefficient approach. In order to obtain a polynomial-size LP, we only work with variables y_I where $|I| \leq O(1)$. For $r \geq 0$, we denote $\mathcal{P}_r([n]) := \{S \subseteq [n] \mid |S| \leq r\}$ as all the index sets of size at most r.

Definition 4. *Let $SA_r(K)$ be the set of vectors $y \in \mathbb{R}^{\mathcal{P}_{r+1}([n])}$ satisfying $y_\emptyset = 1$ and for all $I, J \subseteq [n]$ with $|I| + |J| \leq r$*

$$\sum_{H \subseteq J} (-1)^{|H|} \cdot \Big(\sum_{i=1}^{n} A_{\ell,i} y_{I \cup H \cup \{i\}} - b_\ell y_{I \cup H}\Big) \geq 0 \quad \forall \ell \in [m].$$

The parameter r is usually called the *rank* or *number of rounds* of the Sherali-Adams lift. The reader can verify that for $I = J = \emptyset$, the constraint simplifies to $\sum_{i=1}^n A_{\ell,i} y_{\{i\}} \geq b_\ell y_\emptyset = b_\ell$, implying that $(y_{\{1\}}, \dots, y_{\{n\}}) \in K$. Additionally, for any feasible integral solution $x \in K \cap \{0,1\}^n$ one can set $y_I := \prod_{i \in I} x_i$ to obtain a vector $y \in SA_r(K)$.

Theorem 5 (Properties of Sherali-Adams)**.** *Let $y \in SA_r(K)$ for some $r \geq 0$. Then the following holds:*

(a) For $J \in \mathcal{P}_r([n])$ with $y_J > 0$, the vector $\tilde{y} \in \mathbb{R}^{\mathcal{P}_{r+1-|J|}([n])}$ defined by $\tilde{y}_I := \frac{y_{I \cup J}}{y_J}$ satisfies $\tilde{y} \in SA_{r-|J|}(K)$.
(b) One has $0 \leq y_I \leq y_J \leq 1$ for $J \subseteq I$ and $|I| \leq r+1$.
(c) If $|J| \leq r+1$ and $y_i \in \{0,1\}$ $\forall i \in J$, then $y_I = y_{I \setminus J} \cdot \prod_{i \in I \cap J} y_i$ for all $|I| \leq r+1$.
(d) For $J \subseteq [n]$ with $|J| \leq r$ there exists a distribution over vectors $\tilde{y}$ such that (i) $\tilde{y} \in SA_{r-|J|}(K)$, (ii) $\tilde{y}_i \in \{0,1\}$ for $i \in J$, (iii) $y_I = \mathbb{E}[\tilde{y}_I]$ for all $I \subseteq [n]$ with $|I \cup J| \leq r+1$ (this includes in particular all $I \in \mathcal{P}_{r+1-|J|}([n])$).
(e) For $I \subseteq [n]$ with $|I| \leq r$ and $k \in [N]$ one has $y_I = \sum_{i \in U_k} y_{I \cup \{i\}}$.
(f) Take $H \subseteq [N]$ with $|H| \leq r$ and set $J := \bigcup_{k \in H} U_k$. Then there exists a distribution over vectors $\tilde{y}$ such that (i) $\tilde{y} \in SA_{r-|H|}(K)$, (ii) $\tilde{y}_i \in \{0,1\}$ for $i \in J$, (iii) $y_I = \mathbb{E}[\tilde{y}_I]$ for all $I \in \mathcal{P}_{r+1-|H|}([n])$.

Proof. For (a)-(d), we refer to Laurent [41]. Our proofs of (e) and (f) are custom-tailored to LPs with assignment constraints:

(e) Fix $I \subseteq [n]$ with $|I| \leq r$. We apply (d) to obtain a distribution over $\tilde{y}$ with $\tilde{y} \in SA_{r-|I|}(K)$ so that $\tilde{y}_i \in \{0,1\}$ for $i \in I$. Then using the fact that $\sum_{i \in U_k} \tilde{y}_i = 1$

$$\sum_{i \in U_k} y_{I \cup \{i\}} \stackrel{\text{linearity}}{=} \mathbb{E}\Big[\sum_{i \in U_k} \tilde{y}_{I \cup \{i\}}\Big] \stackrel{(c)}{=} \mathbb{E}\Big[\tilde{y}_I \cdot \sum_{i \in U_k} \tilde{y}_i\Big] \\ = \mathbb{E}[\tilde{y}_I] = y_I.$$

We apply (c) for index sets $I \cup \{i\}$ where variables in $J := I$ are integral. Note that $|I \cup (I \cup \{i\})| \leq r+1$.

(f) By an inductive argument it suffices to consider the case of $|H| = 1$. Let $H = \{k\}$ and set $U := U_k$, i.e. the constraints for polytope P contain the assignment constraint $\sum_{i \in U} x_i = 1$ and we want to make all variables in U integral while only losing a *single* round in the hierarchy. Abbreviate $U^+ := \{i \in U \mid y_{\{i\}} > 0\}$. For $i \in U^+$, define $y^{(i)} \in \mathbb{R}^{\mathcal{P}_r([n])}$ to be the vector with $y_I^{(i)} := \frac{y_{I \cup \{i\}}}{y_i}$. By (a) we know that $y^{(i)} \in SA_{r-1}(K)$. Moreover $y_{\{i\}}^{(i)} = \frac{y_{\{i\}}}{y_{\{i\}}} = 1$. Then the assignment constraint of the LP forces that $y_{\{i'\}}^{(i)} = 0$ for $i' \in U \setminus \{i\}$. Now we define

[2]Papadimitriou and Yannakakis [17] claim a 2-hardness for $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c \mid C_{\max}$, but give no proof. Schuurman and Woeginger [19] remark that "it would be nice to have a proof for this claim".

a probability distribution over vectors $\tilde{y}$ as follows: for $i \in U^+$, with probability y_i we set $\tilde{y} := y^{(i)}$. Then (i) and (ii) hold for $\tilde{y}$ as discussed. Property (iii) follows from

$$\begin{aligned}\mathbb{E}[\tilde{y}_I] &= \sum_{i \in U^+} y_i y_I^{(i)} = \sum_{i \in U^+} y_i \frac{y_{I \cup \{i\}}}{y_i} \\ &= \sum_{i \in U^+} y_{I \cup \{i\}} \stackrel{(b)}{=} \sum_{i \in U} y_{I \cup \{i\}} \stackrel{(e)}{=} y_I.\end{aligned}$$

□

It is known that Theorem 5.(f) holds in a stronger form for the SDP-based *Lasserre hierarchy*. Karlin, Mathieu and Nguyen [43] proved a result that can be paraphrased as follows: *if one has any set $J \subseteq [n]$ of variables with the property that there is no LP solution with more than k ones in J, then one can make all variables of J integral while losing only k rounds*. Interestingly, Karlin, Mathieu and Nguyen prove that this is completely false for Sherali-Adams. In particular, for a Knapsack instance with unit size items and capacity $2 - \varepsilon$, the integrality gap is still $2 - 2\varepsilon$ after $\Theta_\varepsilon(n)$ rounds of Sherali-Adams. In a different setting, Friggstad et al. [42] realized that given a "tree constraint", a Sherali-Adams lift can provide the same guarantees that Rothvoss [44] derived from Lasserre. While Friggstad et al. did not state their insight in the generality that we need here, our Lemma 5.(e)+(f) are inspired by their work.

B. Semimetric Spaces

A *semimetric space* is a pair (V, d) where V is a finite set (we denote $n := |V|$) and $d : V \times V \to \mathbb{R}_{\geq 0}$ is a *semimetric*, i.e.

- $d(u, u) = 0$ for all $u \in U$.
- Symmetry: $d(u, v) = d(v, u)$ for all $u, v \in V$.
- Triangle inequality: $d(u, v) + d(v, w) \geq d(u, w)$ for all $u, v, w \in V$.

Recall that the more common notion is that of a *metric*, which additionally requires that $d(u, v) > 0$ for $u \neq v$. For a set $U \subseteq V$ we denote the *diameter* as $\text{diam}(U) := \max_{u,v \in U} d(u, v)$. Our goal is to find a partition $V = V_1 \dot{\cup} \dots \dot{\cup} V_k$ such that the diameter of every cluster V_i is bounded by some parameter Δ. We denote $d(w, U) := \min\{d(w, u) : u \in U\}$ as the distance to the set U. Moreover, for $r \geq 0$ and $U \subseteq V$, let $N(U, r) := \{v \in V \mid d(v, U) \leq r\}$ be the *distance r-neighborhood* of U.

We use a very influential clustering algorithm due to Calinescu, Karloff and Rabani [23], which assigns each $v \in V$ to a random cluster center $c \in V$ such that $d(u, c) \leq \beta\Delta$. Nodes assigned to the same cluster center form one block V_i in the partition. Formally the algorithm is as follows:

CKR CLUSTERING ALGORITHM

Input: Semimetric space (V, d), $V = \{v_1, \dots, v_n\}$, $\Delta > 0$
Output: Clustering $V = V_1 \dot{\cup} \dots \dot{\cup} V_k$ for some k.

(1) Pick a uniform random $\beta \in [\frac{1}{4}, \frac{1}{2}]$
(2) Pick a random ordering $\pi : V \to \{1, \dots, n\}$
(3) For each $v \in V$ set $\sigma(v) := v_\ell$ so that $d(v, v_\ell) \leq \beta \cdot \Delta$ and $\pi(v_\ell)$ is minimal
(4) Denote $v \in V$ with $\sigma^{-1}(v) \neq \emptyset$ by $c_1, \dots, c_k \in V$. Return clusters $V_i := \sigma^{-1}(c_i)$ for $i = 1, \dots, k$

Note that the algorithm has two sources of randomness: it picks a random parameter β, and independently it picks a random ordering π. Here the ordering is to be understood such that element v_ℓ with $\pi(v_\ell) = 1$ is the "highest priority" element. The original work of Calinescu, Karloff and Rabani [23] only provided an upper bound on the probability that a short edge (u, v) is separated. Mendel and Naor [45] note that the same clustering provides the guarantee of $\Pr[N(u, t) \text{ separated}] \leq 1 - O(\frac{t}{\Delta} \cdot \ln(\frac{|N(u,\Delta)|}{|N(u,\Delta/8)|}))$ for all $u \in V$ and $0 \leq t < \frac{\Delta}{8}$. Mendel and Naor attribute this to Fakcharoenphol, Rao and Talwar [46] (while Fakcharoenphol, Rao and Talwar [46] do not state it explicitly in this form and focus on the "local growth ratio" aspect). Instead of the algorithm by Calinescu, Karloff and Rabani [23], one could also cluster using the techniques of Leighton and Rao [47] or those of Garg, Vazirani and Yannakakis [48].

We state the formal claim in a form that will be convenient for us. For the sake of completeness, a proof can be found in the Appendix of the full version.

Theorem 6 (Analysis of CKR). *Let $V = V_1 \dot{\cup} \dots \dot{\cup} V_k$ be the random partition of the CKR algorithm. The following holds:*

(a) The blocks have $diam(V_i) \leq \Delta$ for $i = 1, \dots, k$.
(b) Let $U \subseteq V$ be a subset of points. Then

$$\begin{aligned}&\Pr[U \text{ is separated by clustering}] \\ &\leq \ln\left(2\left|N\left(U, \Delta/2\right)\right|\right) \cdot \frac{4diam(U)}{\Delta} \leq \ln(2n) \cdot \frac{4diam(U)}{\Delta}.\end{aligned}$$

In the above, *separated* means that there is more than one index i with $V_i \cap U \neq \emptyset$.

III. AN APPROXIMATION FOR $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c\mathsf{-intervals} \mid C_{\max}$

In this section, we provide an approximation algorithm for scheduling n unit-length jobs J with communication delay $c \in \mathbb{N}$ on an unbounded number of machines so that precedence constraints given by a partial order $\prec$ are satisfied. Instead of working with $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c \mid C_{\max}$ directly, it will be more convenient to consider a slight variant that we call $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c\mathsf{-intervals} \mid C_{\max}$. This problem variant has the same input but the time horizon is partitioned into time *intervals* of length c, say $I_s = [sc, (s+1)c)$ for $s \in \mathbb{Z}_{\geq 0}$. The goal is to assign jobs to intervals and machines. We require that if $j_1 \prec j_2$ then either j_1 is scheduled in an earlier interval than j_2 or j_1 and j_2 are scheduled in the same interval on the same machine. Other than that, there are no further communication delays. The objective function is to minimize the number of intervals used to process the jobs. In fact we do not need to decide the order of jobs within intervals as any topological order will work. In a more mathematical notation, the problem asks to find a partition $J = \dot{\bigcup}_{s\in\{0,\dots,S-1\},i\in\mathbb{N}} J_{s,i}$ with $|J_{s,i}| \leq c$ such that S is minimized and for every $j_1 \prec j_2$ with $j_1 \in J_{s_1,i_1}$ and $j_2 \in J_{s_2,i_2}$ one has either $s_1 < s_2$ or $(s_1, i_1) = (s_2, i_2)$. See Figure 1 for an illustration.

It is rather straightforward to give reductions between $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c \mid C_{\max}$ and $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c\text{−intervals} \mid C_{\max}$ that only lose a small constant factor in both directions. The only subtle point to consider here is that when the optimum makespan for $\mathsf{P}\infty \mid \mathsf{prec}, c \mid C_{\max}$ is less than c, the problem admits a PTAS; we refer to Section IV for details.

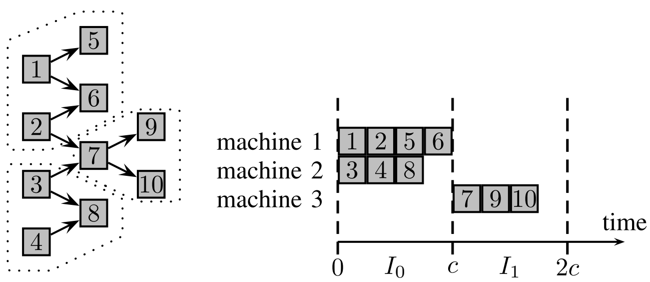

Fig. 1. Left: example of an instance of $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c\text{−intervals} \mid C_{\max}$ with $c = 4$ (where the partial order $\prec$ is the transitive closure of the depicted digraph). Right: a valid schedule in 2 intervals.

A. The Linear Program

Let $m \in \mathbb{N}$ be a parameter defining the number of machines that we admit for the LP. Moreover, let $S \in \mathbb{N}$ be the number of intervals that we allow for the time horizon. To obtain an approximation for $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c\text{−intervals} \mid C_{\max}$ one can set $m := n$ and perform a binary search to find the minimal S for which the LP is feasible. But we prefer to keep the approach general.

We construct the LP in two steps. First consider the variables

$$x_{j,i,s} = \begin{cases} 1 & \text{if } j \text{ is scheduled on machine } i \text{ in interval } I_s \\ 0 & \text{otherwise} \end{cases}$$
$$\forall j \in J, i \in [m], s \in \{0,\dots,S-1\}$$

Let K be the set of fractional solutions to the following linear system

$$\begin{aligned} \sum_{i\in[m]}\sum_{s\geq 0} x_{j,i,s} &= 1 \quad \forall j \in J \\ \sum_{j\in J} x_{j,i,s} &\leq c \quad \forall i \in [m] \ \ \forall s \in \{0,\dots,S-1\} \\ 0 \leq x_{j,i,s} &\leq 1 \quad \forall j \in J, i \in [m], s \in \{0,\dots,S-1\} \end{aligned}$$

So far, K simply assigns jobs (fractionally) to intervals and machines without taking any precedence constraints into account. Next, we will use a lift $x \in \mathrm{SA}_r(K)$ containing variables $x_{(j_1,i_1,s_1),(j_2,i_2,s_2)}$, which provide the probability for the event that j_1 is scheduled in interval s_1 on machine i_1 and j_2 is scheduled in interval s_2 on machine i_2. We introduce two more types of decision variables:

$$\begin{aligned} y_{j_1,j_2} &= \begin{cases} 1 & j_1, j_2 \text{ on the same machine and interval} \\ 0 & \text{otherwise} \end{cases} \\ C_j &= \text{index of interval where } j \text{ is processed} \end{aligned}$$

Let $Q(r)$ be the set of vectors (x, y, C) that satisfy

$$\begin{aligned} y_{j_1,j_2} &= \sum_{s\in\{0,\dots,S-1\}}\sum_{i\in[m]} x_{(j_1,i,s),(j_2,i,s)} \\ C_{j_2} &\geq C_{j_1} + (1 - y_{j_1,j_2}) \quad \forall j_1 \prec j_2 \\ C_j &\geq 0 \quad \forall j \in J \\ x &\in SA_r(K) \end{aligned}$$

The analysis of our algorithm will work for all $r \geq 5$ while solving the LP takes time $n^{O(r)}$. Here we make no attempt at optimizing the constant r. The main technical contribution of this section is the following rounding result:

Theorem 7. *Consider an instance with unit-length jobs J, a partial order $\prec$, and parameters $c, S, m \in \mathbb{N}$ such that $Q(r)$ is feasible for $r := 5$. Then there is a randomized algorithm with expected polynomial running time that finds a schedule for $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c\text{−intervals} \mid C_{\max}$ using at most $O(\log m \cdot \log c) \cdot S$ intervals.*

We would like to emphasize that we require $\prec$ to be a partial order, which implies that it is transitive. While replacing any acyclic digraph with its transitive closure does not change the set of feasible integral schedules and hence can be done in a preprocessing step, it corresponds to adding constraints to the LP that we rely on in the algorithm and in its analysis.

We will now discuss some properties that are implied by the Sherali-Adams lift:

Lemma 8. *Let $(x, y, C) \in Q(r)$ with $r \geq 2$. Then for any set $\tilde{J} \subseteq J$ of $|\tilde{J}| \leq r - 2$ jobs, there exists a distribution $\mathcal{D}(\tilde{J})$ over pairs $(\tilde{x}, \tilde{y})$ such that*

(A) $\tilde{x}_{j,i,s} \in \{0, 1\}$ *for all* $j \in \tilde{J}$, *all* $i \in [m]$ *and* $s \geq 0$.
(B) $\tilde{y}_{j_1,j_2} = \sum_{s\geq 0}\sum_{i\in[m]} \tilde{x}_{j_1,i,s} \cdot \tilde{x}_{j_2,i,s}$ *if* $|\{j_1, j_2\} \cap \tilde{J}| \geq 1$.
(C) $\tilde{x} \in K$, $\tilde{y}_{j_1,j_2} = \sum_{s\in\{0,\dots,S-1\}}\sum_{i\in[m]} \tilde{x}_{(j_1,i,s),(j_2,i,s)}$ *for all* $j_1, j_2 \in J$.
(D) $\mathbb{E}[\tilde{x}_{j,i,s}] = x_{j,i,s}$ *and* $\mathbb{E}[\tilde{y}_{j_1,j_2}] = y_{j_1,j_2}$ *for all* j, j_1, j_2, i, s.

Proof. By Theorem 5.(f), there is a distribution over $\tilde{x} \in SA_2(K)$ which satisfies (A) and has $\tilde{x} \in K$, $\mathbb{E}[\tilde{x}_{j,i,s}] = x_{j,i,s}$ and $\mathbb{E}[\tilde{x}_{(j_1,i_1,s_1),(j_2,i_2,s_2)}] = x_{(j_1,i_1,s_1),(j_2,i_2,s_2)}$, and additionally is integral on variables involving only jobs from $\tilde{J}$, where $|\tilde{J}| \leq r - 2$. Here, we crucially use that every job $j \in \tilde{J}$ is part of an assignment constraint $\sum_{i\in[m]}\sum_{s\geq 0} x_{j,i,s} = 1$, hence making these variables integral results in the loss of only one round per job. Then, the y-variables are just linear functions depending on the x-variables, so we can define

$$\tilde{y}_{j_1,j_2} := \sum_{s\in\{0,\dots,S-1\}}\sum_{i\in[m]} \tilde{x}_{(j_1,i,s),(j_2,i,s)}$$

and the claim follows. $\square$

From the LP solution, we define a semimetric d. Here the intuitive interpretation is that a small distance $d(j_1, j_2)$ means that the LP schedules j_1 and j_2 mostly on the same machine and in the same interval.

Lemma 9. *Let $(x, y, C) \in Q(r)$ be a solution to the LP with $r \geq 5$. Then $d(j_1, j_2) := 1 - y_{j_1,j_2}$ is a semimetric.*

Proof. The first two properties from the definition of a semimetric (see Section II-B) are clearly satisfied. We verify the triangle inequality. Consider three jobs $j_1, j_2, j_3 \in J$. We apply Lemma 8 with $\tilde{J} := \{j_1, j_2, j_3\}$ and consider the distribution $(\tilde{x}, \tilde{y}) \sim \mathcal{D}(\tilde{J})$. For $j \in \tilde{J}$, define $Z(j) = (\tilde{s}(j), \tilde{i}(s))$ as the random variable that gives the unique pair of indices such that $\tilde{x}_{j,\tilde{i}(j),\tilde{s}(j)} = 1$. Then for $j', j'' \in \tilde{J}$ one has

$$\begin{aligned} d(j', j'') &= \Pr[Z(j') \neq Z(j'')] \\ &= \Pr\left[\left(\tilde{s}(j), \tilde{i}(j')\right) \neq \left(\tilde{s}(j''), \tilde{i}(j'')\right)\right] \end{aligned}$$

Then indeed

$$\begin{aligned} d(j_1, j_3) &= \Pr[Z(j_1) \neq Z(j_3)] \\ &\leq \Pr[Z(j_1) \neq Z(j_2) \vee Z(j_2) \neq Z(j_3)] \\ &\overset{\text{union bound}}{\leq} \Pr[Z(j_1) \neq Z(j_2)] + \Pr[Z(j_2) \neq Z(j_3)] \\ &= d(j_1, j_2) + d(j_2, j_3). \end{aligned}$$

$\square$

Lemma 10. *For every $j_1 \in J$ one has $\sum_{j_2 \in J} y_{j_1,j_2} \leq c$.*

Proof. Consider the distribution $(\tilde{x}, \tilde{y}) \sim \mathcal{D}(\{j_1\})$. From Lemma 8.(B) we know that $\mathbb{E}[\tilde{y}_{j_1,j_2}] = y_{j_1,j_2}$ and $\tilde{y}_{j_1,j_2} = \sum_{s \in \{0,\ldots,S-1\}} \sum_{i \in [m]} \tilde{x}_{j_1,i,s} \cdot \tilde{x}_{j_2,i,s}$. By linearity it suffices to prove that $\sum_{j_2 \in J} \tilde{y}_{j_1,j_2} \leq c$ always. Fix a pair $(\tilde{x}, \tilde{y})$. There is a unique pair of indices (i_1, s_1) with $\tilde{x}_{j_1,i_1,s_1} = 1$. Then

$$\begin{aligned} \sum_{j_2 \in J} \tilde{y}_{j_1,j_2} &= \sum_{s \in \{0,\ldots,S-1\}} \sum_{j_2 \in J} \sum_{i \in [m]} \underbrace{\tilde{x}_{j_1,i,s}}_{0 \text{ if } i \neq i_1 \text{ or } s \neq s_1} \cdot \tilde{x}_{j_2,i,s} \\ &= \sum_{j_2 \in J} \tilde{x}_{j_2,i_1,s_1} \leq c. \end{aligned}$$

$\square$

A crucial insight is that for any job j^*, few jobs are very close to j^* with respect to d.

Lemma 11. *Fix $j^* \in J$ and abbreviate $U := \{j \in J \mid d(j, j^*) \leq \beta\}$ for $0 < \beta < 1$. Then $|U| \leq \frac{c}{1-\beta}$.*

Proof. For each $j \in U$ we have $y_{j,j^*} = 1 - d(j, j^*) \geq 1 - \beta$. Combining with the last lemma we have $(1 - \beta)|U| \leq \sum_{j \in J} y_{j,j^*} \leq c$. $\square$

B. Scheduling a Single Batch of Jobs

We now come to the main building block of our algorithm. We consider a subset J^* of jobs whose LP completion times C_j are very close (within a $\Theta(\frac{1}{\log(c)})$ term of each other) and show we can schedule half of these jobs in a single length-$2c$ interval. The following lemma is the main technical contribution of the paper.

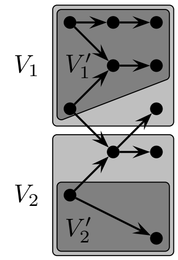

Fig. 2. Visualization of the partition $V = V_1 \dot\cup \ldots \dot\cup V_k$ and the induced sets $V_\ell' \subseteq V_\ell$. Here $\prec$ is the transitive closure of the depicted digraph.

Lemma 12. *Let $(x, y, C) \in Q(r)$ with $r \geq 5$ and let $0 < \delta \leq \frac{1}{64 \log(4c)}$ be a parameter. Let $C^* \geq 0$ and set $J^* \subseteq \{j \in J \mid C^* \leq C_j \leq C^* + \delta\}$. Then there is a randomized rounding procedure that finds a schedule for a subset $J^{**} \subseteq J^*$ in a single interval of length at most $2c$ such that every job $j \in J^*$ is scheduled with probability at least $1 - 32\log(4c) \cdot \delta \geq \frac{1}{2}$.*

We denote $\Gamma^-(j)$ as the predecessors of j and $\Gamma^+(j)$ as the successors, and similary $\Gamma^{-/+}(J') = \{j \in J : \exists j' \in J' \text{ s.t. } j \in \Gamma^{-/+}(j')\}$. Again, recall that we assume $\prec$ to be transitive. The rounding algorithm is the following:

SCHEDULING A SINGLE BATCH
(1) Run CKR clustering on (J^*, d) with $\Delta := \frac{1}{4}$. Let $V_1, \ldots, V_k$ be the clusters.
(2) Let $V_\ell' := \{j \in V_\ell \mid \Gamma^-(j) \cap J^* \subseteq V_\ell\}$ for $\ell = 1, \ldots, k$.
(3) Schedule V_ℓ' on one machine for all $\ell = 1, \ldots, k$.

We now discuss the analysis. First we show that no cluster is more than a constant factor too large.

Lemma 13. *One has $|V_\ell'| \leq 2c$ for all $\ell = 1, \ldots, k$.*

Proof. We know by Theorem 6 that $\text{diam}(V_\ell') \leq \text{diam}(V_\ell) \leq \Delta < \frac{1}{2}$ where the diameter is with respect to d. Fix a job $j^* \in V_\ell'$. Then we know by Lemma 11 that there are at most $2c$ jobs j with $d(j, j^*) \leq \frac{1}{2}$ and the claim follows. $\square$

Next, we see that the clusters respect the precedence constraints.

Lemma 14. *The solution $V_1', \ldots, V_k'$ is feasible in the sense that jobs on different machines do not have precedence constraints.*

Proof. Consider jobs processed on different machines, say (after reindexing) $j_1 \in V_1'$ and $j_2 \in V_2'$. If $j_1 \prec j_2$ then we did *not* have $\Gamma^-(j_2) \subseteq V_2'$. This contradicts the definition of the sets V_ℓ'. $\square$

A crucial property that makes the algorithm work is that predecessors of some job $j \in J^*$ must be very close in d distance.

Lemma 15. *For every $j_1, j_2 \in J^*$ with $j_1 \prec j_2$ one has $d(j_1, j_2) \leq \delta$.*

Proof. We know that

$$C^* \overset{j_1 \in J^*}{\leq} C_{j_1} \leq C_{j_1} + \underbrace{(1 - y_{j_1,j_2})}_{=d(j_1,j_2)} \overset{LP}{\leq} C_{j_2} \overset{j_2 \in J^*}{\leq} C^* + \delta$$

and so $d(j_1, j_2) \leq \delta$. $\square$

We will use the three statements above together with Theorem 6 to prove Lemma 12.

Proof of Lemma 12. We have already proven that the scheduled blocks have size $|V'_\ell| \leq 2c$ and that there are no dependent jobs in different sets of $V'_1, \ldots, V'_k$. To finish the analysis, we need to prove that a fixed job $j^* \in J^*$ is scheduled with good probability. Consider the set $U := \{j^*\} \cup (\Gamma^-(j^*) \cap J^*)$ of j^* and its ancestors in J^*.

Since the diameter of U is at most 2δ by Lemma 15, we can use Lemma 11 to see that $|N(U, \Delta/2)| \leq \frac{c}{1-2\delta-\Delta}$. For our choice of $\Delta = 1/4$ and $\delta \leq \frac{1}{64\log(4c)}$, $|N(U, 1/8)| \leq 2c$. From Theorem 6, the cluster is separated with probability at most $\log(4c) \cdot \frac{8\delta}{\Delta} \leq \frac{1}{2}$. $\square$

To schedule all jobs in J^*, we repeat the clustering procedure $O(\log m)$ times and simply schedule the remaining jobs on one machine.

Lemma 16. *Let $(x, y, C) \in Q(r)$ with $r \geq 5$. Let $C^* \geq 0$ and set $J^* \subseteq \{j \in J \mid C^* \leq C_j < C^* + \delta\}$. Assume that all jobs in $\Gamma^-(J^*) \setminus J^*$ have been scheduled respecting precedence constraints. Then there is an algorithm with expected polynomial running time that schedules all jobs in J^* using at most $O(\log m) + \frac{|J^*|}{mc}$ many intervals.*

Proof. Our algorithm in Lemma 12 schedules each $j \in J^*$ in an interval of length $2c$ with probability at least $1/2$. We run the algorithm for $2\log m$ iterations, where input to iteration $k+1$ is the subset of jobs that are not scheduled in the first k iterations. For $k \in \{1, 2, \ldots, 2\log m\}$, let J_k^{**} denote the subset of jobs that are scheduled in the k^{th} iteration, and let $J^*_{k+1} := J^* \setminus \{\bigcup_{k'=1}^{k} J^{**}_{k'}\}$. In this notation, $J^*_1 := J^*$. Let $\mathcal{S}(J^{**}_k)$ denote the schedule of jobs J^{**}_k given by Lemma 12. We schedule $\mathcal{S}(J^{**}_1)$ first, then for $k = 2, \ldots, 2\log m$, we append the schedule $\mathcal{S}(J^{**}_k)$ after $\mathcal{S}(J^{**}_{k-1})$. Let $J' := J^*_{2\log m+1}$ denote the set of jobs that were not scheduled in the $2\log m$ iterations. We schedule all jobs in J' consecutively on a single machine after the completion of $\mathcal{S}(J^{**}_{2\log m})$.

From our construction, the length of a schedule for J^*, which is a random variable, is at most $O(\log m) + \lceil\frac{|J'|}{c}\rceil$ many intervals. For $k \in \{1, 2, \ldots, 2\log m\}$, Lemma 12 guarantees that each job $j \in J^*_k$ gets scheduled in the k^{th} iteration with probability at least $1/2$. Therefore, the probability that $j \in J'$, i.e., it does not get scheduled in the first $2\log m$ iterations, is at most $\frac{1}{2m}$. This implies that $\mathbb{E}[|J'|] \leq \frac{|J^*|}{2m}$. By Markov's inequality $\Pr[|J'| > \frac{|J^*|}{m}] \leq \Pr[|J'| > 2 \cdot \mathbb{E}[|J'|]] \leq 1/2$. Hence we can repeat the described procedure until indeed we have a successful run with $|J'| \leq \frac{|J^*|}{m}$ which results in the claimed expected polynomial running time.

Let us now argue that the schedule of J^* is feasible. For $k \in \{1, 2, \ldots, 2\log m\}$ and any two jobs $j, j' \in \mathcal{S}(J^{**}_k)$, Lemma 12 guarantees that precedence and communication constraints are satisfied. Furthermore, Lemma 12 also ensures that there cannot be jobs j, j' such that $j \in \mathcal{S}(J^{**}_k)$, $j' \in \mathcal{S}(J^{**}_{k'})$ and $j' \prec j$ and $k' > k$. Finally note that every length-$2c$ interval can be split into 2 length-c intervals. The claim follows. $\square$

C. The Complete Algorithm for $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c\text{−}\mathsf{intervals} \mid C_{\max}$

Now we have all the pieces to put the rounding algorithm together and prove its correctness. We partition the jobs into batches, where each batch consists of subset of jobs that have C_j very close to each other in the LP solution. The complete algorithm is given below.

THE COMPLETE ALGORITHM

(1) Solve the LP and let $(x, y, C) \in Q(r)$ with $r \geq 5$.
(2) For $\delta = \frac{1}{64\log(4c)}$ and $k \in \{0, 1, 2 \ldots \frac{S-1}{\delta}\}$, define $J_k = \{j \in J : k \cdot \delta \leq C_j < (k+1) \cdot \delta\}$
(3) FOR $k = 0$ TO $\frac{S-1}{\delta}$ DO
  (4) Schedule J_k using the algorithm in Subsection III-B.

Now we finish the analysis of the rounding algorithm.

Proof of Theorem 7. Let us quickly verify that the schedule constructed by our algorithm is feasible. For jobs $j_1 \prec j_2$ with $j_1 \in J_{k_1}$ and $j_2 \in J_{k_2}$, the LP implies that $C_{j_1} \leq C_{j_2}$ and so $k_1 \leq k_2$. If $k_1 < k_2$, then j_1 will be scheduled in an earlier interval than j_2. If $k_1 = k_2 = k$, then Lemma 16 guarantees that precedence constraints are satisfied.

It remains to bound the makespan of our algorithm. Lemma 16 guarantees that for $k \in \{0, 1, 2 \ldots \frac{S-1}{\delta}\}$, the jobs in J_k are scheduled using at most $O(\log m) + \frac{|J_k|}{cm}$ many intervals. Then the total number of intervals required by the algorithm is bounded by

$$\begin{aligned}\frac{S}{\delta} \cdot O(\log m) + \sum_{k=0}^{\frac{S-1}{\delta}} \frac{|J_k|}{cm} &= O(\log m \cdot \log c) \cdot S + \frac{|J|}{cm} \\ &\leq O(\log m \cdot \log c) \cdot S.\end{aligned}$$

Here we use that $|J| \leq S \cdot cm$ is implied by the constraints defining K. $\square$

Remark 17. *We note that it is possible to reverse-engineer our solution and write a more compact LP for the problem, enforcing only the necessary constraints such as those given by Lemmas 9 and 10. Such an LP would be simpler and could be solved more efficiently. However, we feel that the Sherali-Adams hierarchy gives a more principled and intuitive way to tackle the problem and explain how the LP arises, and hence we choose to present it that way.*

IV. Reductions

We now justify our earlier claim: the special case $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c\mathsf{-intervals} \mid C_{\max}$ indeed captures the full computational difficulty of the more general problem $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\max}$. The main result for this section will be the following reduction:

Theorem 18. *Suppose there is a polynomial time algorithm that takes a solution for the LP $Q(r)$ with parameters $m, c, S \in \mathbb{N}$ and $r \geq 5$ and transforms it into a schedule for $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c\mathsf{-intervals} \mid C_{\max}$ using at most $\alpha \cdot S$ intervals. Then there is a polynomial time $O(\alpha)$-approximation for $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\max}$.*

For the reduction we will make use of the very well known list scheduling algorithm by Graham [3] that can be easily extended to the setting with communication delays. Here the notation $\sigma(j) = ([t, t+p_j), i)$ means that the job j is processed in the time interval $[t, t+p_j)$ on machine $i \in [m]$.

Graham's List Scheduling

(1) Set $\sigma(j) := \emptyset$ for all $j \in J$
(2) FOR $t = 0$ TO ∞ DO FOR $i = 1$ TO m DO
  (3) Select any job $j \in J$ with $\sigma(j) = \emptyset$ where every $j' \prec j$ satisfies the following:
  - If j' is scheduled on machine i then j' is finished at time $\leq t$
  - If j' is schedule on machine $i' \neq i$ then j' finished at time $\leq t - c$

  (4) Set $\sigma(j) := ([t, t+p_j), i)$ (if there was such a job)

For example, for the problem $\mathsf{P} \mid \mathsf{prec} \mid C_{\max}$, Graham's algorithm gives a 2-approximation. The analysis works by proving that there is a chain of jobs covering all time units where not all machines are busy. Graham's algorithm does *not* give a constant factor approximation for our problem with communication delays, but it will still be useful for our reduction.

Recall that a set of jobs $\{j_1, \ldots, j_\ell\} \subseteq J$ with $j_\ell \prec j_{\ell-1} \prec \ldots \prec j_1$ is called a *chain*. We denote $\mathcal{Q}(J)$ as the set of all chains in J w.r.t. precedence order $\prec$.

Lemma 19. *Graham's list scheduling on an instance of $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\max}$ results in a schedule with makespan at most $\frac{1}{m}\sum_{j \in J} p_j + \max_{Q \in \mathcal{Q}(J)}\{\sum_{j \in Q} p_j + c \cdot (|Q| - 1)\}$.*

Proof. We will show how to construct the chain Q that makes the inequality hold. Let j_1 be the job which finishes last in the schedule produced by Graham's algorithm and let t_{j_1} be its start time. Let j_2 be the predecessor of j_1 that finishes last. More generally in step i, we denote j_{i+1} as the predecessor of j_i that finishes last. The construction finishes with a job j_ℓ without predecessors. Now let Q be the chain of jobs $j_\ell \prec j_{\ell-1} \prec \ldots \prec j_1$. The crucial observation is that for any $i \in \{1, \ldots, \ell-1\}$, either all machines are busy in the time interval $[t_{j_{i+1}} + p_{j_{i+1}} + c, t_{j_i})$ or this interval is empty. The reason is that Graham's algorithm does not leave unnecessary idle time

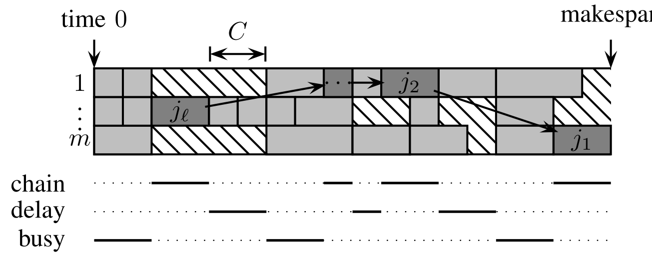

Fig. 3. Analysis of Graham's algorithm with communication delay c.

and would have otherwise processed j_i earlier. It is also true that all m machines are busy in the time interval $[0, t_{j_\ell})$. The total amount of work processed in these busy time intervals is

$$L := m \cdot \Big(t_{j_\ell} + \sum_{i=1}^{\ell-1} \max\{t_{j_i} - (c + p_{j_{i+1}} + t_{j_{i+1}}), 0\}\Big)$$
$$\leq \sum_{j \in J} p_j - \sum_{i=1}^{\ell} p_{j_i}.$$

Then any time between 0 and the makespan falls into at least one of the following categories: (a) the busy time periods described above, (b) the times that a job of the chain Q is processed, (c) the interval of length c following a job in the chain Q. Thus, we see that the makespan from Graham's list scheduling is at most

$$t_{j_1} + p_{t_{j_1}} \leq \frac{L}{m} + \sum_{j \in Q} p_j + c \cdot (|Q| - 1)$$
$$\leq \frac{1}{m}\sum_{j \in J} p_j + \Big(1 - \frac{1}{m}\Big)\sum_{j \in Q} p_j + c \cdot (|Q| - 1).$$

□

It will also be helpful to note that the case of very small optimum makespan can be well approximated:

Lemma 20. *Any instance for $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\max}$ with optimum objective function value at most c admits a PTAS.*

Proof. Let J be the jobs in the instance and let $\mathrm{OPT}_m \leq c$ be the optimum value. Consider the undirected graph $G = (J, E)$ with $\{j_1, j_2\} \in E \Leftrightarrow ((j_1 \prec j_2)$ or $(j_2 \prec j_1))$. Let $J = J_1 \dot\cup \ldots \dot\cup J_N$ be the partition of jobs into connected components w.r.t. graph G. We abbreviate $p(J') := \sum_{j \in J'} p_j$. The assumption guarantees that the optimum solution cannot afford to pay the communication delay and hence there is a length-c schedule that assigns all jobs of the same connected component to the same machine. If we think of a connected component J_ℓ as an "item" of size $p(J_\ell)$, then for any fixed $\varepsilon > 0$ we can use a PTAS for $\mathsf{P} \mid\mid C_{\max}$ (i.e. makespan minimization without precedence constraints) to find a partition of "items" as $[N] = I_1 \dot\cup \ldots \dot\cup I_m$ with $\sum_{\ell \in I_i} p(J_\ell) \leq (1 + \varepsilon) \cdot \mathrm{OPT}_m$ in polynomial time [49]. Arranging the jobs $\bigcup_{\ell \in I_i} J_\ell$ on machine i in any topological order finishes the argument. □

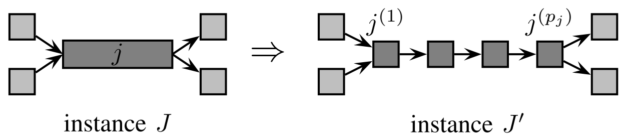

Fig. 4. Splitting jobs into chains of unit-length jobs.

Additionally, it is a standard argument to convert an instance with arbitrary p_j to an instance where all $p_j \leq n/\varepsilon$, while only losing a factor of $(1+2\varepsilon)$ in the approximation. For $p_{\max} := \max_j p_j$, we simply scale the job lengths and communication delay down by a factor of $\frac{n}{\varepsilon p_{\max}}$ then round them to the nearest larger integer. This results in at most a 2ε fraction of the optimal makespan being rounded up and all job sizes are integral and at most n/ε.

Now we can show the main reduction:

Proof of Theorem 18. Consider an instance of $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\max}$ with $p_j, c \in \mathbb{N}$. Let J denote its job set with precedence constraints, and $\mathrm{OPT}_m(J)$ denote its optimal value where m is the number of available machines. By the previous argument, we may assume that $p_j \leq 2n$ for all $j \in J$. Moreover, by Lemma 20 we only need to focus on the case where $\mathrm{OPT}_m(J) > c$. We may guess the optimum value of $\mathrm{OPT}_m(J)$ as $\mathrm{OPT}_m(J) \in \{1, \ldots, 2n^2\}$.

Let J' denote the job set obtained by splitting each job $j \in J$ into a chain of p_j unit sub-jobs $j^{(1)} \prec \cdots \prec j^{(p_j)}$. Moreover, precedence constraints in J are preserved in J' as we set all predecessors of j to be predecessors of $j^{(1)}$ and all successors of j to be successors of $j^{(p_j)}$, see Figure 4. We note that $\mathrm{OPT}_m(J') \leq \mathrm{OPT}_m(J)$ as splitting does not increase the value of the optimum. Let $\mathcal{S}_m(J')$ be a schedule achieving the value of $\mathrm{OPT}_m(J')$. Next, observe that $\mathcal{S}_m(J')$ implies an integral solution for $Q(r)$ with parameters m, c, S where $S := \lceil \mathrm{OPT}_m(J)/c \rceil$ and $r := 5$. In particular here we use that if jobs $j_1 \prec j_2$ are scheduled on different machines by $\mathcal{S}_m(J')$, then their starting times differ by at least $c+1$ and hence they are assigned to different length-c intervals.

Now we execute the assumed α-approximate rounding algorithm and obtain a schedule $\mathcal{S}_{\infty,\text{int}}(J')$ that uses $T \leq \alpha S$ many intervals. We will use this solution $\mathcal{S}_{\infty,\text{int}}(J')$ to construct a schedule $\mathcal{S}_\infty(J)$ for $\mathsf{P}\infty \mid \mathsf{prec}, c \mid C_{\max}$ with job set J by running split sub-jobs consecutively on the same processor. This will use $4T$ time intervals in total. Recall that I_s denotes the time interval $[sc, (s+1)c)$. The rescheduling process is as follows:

For a fixed job $j \in J$, let I_{s_1} be the time interval where $j^{(1)}$ is scheduled in $\mathcal{S}_{\infty,\text{int}}(J')$. Then all other sub-jobs of j should be either scheduled in I_{s_1} or the time intervals after I_{s_1}.

- **Case 1: Some sub-job of j is not scheduled in I_{s_1}.** Schedule job j at the beginning of time interval I_{4s_1+1} on a new machine. If j is a short job, then it will finish running by the end of the interval. Otherwise j is a long job. Let I_{s_2} be the last time interval where a sub-job of j is scheduled in $\mathcal{S}_{\infty,\text{int}}(J')$. Then, the length satisfies $p_j \leq c \cdot (s_2 - s_1 + 1)$, which implies that the job finishes by time $c \cdot (4s_1+1) + p_j \leq c \cdot (4s_2 - 1)$.
- **Case 2: All sub-jobs of j are scheduled in I_{s_1}.** Simply schedule job j during time interval I_{4s_1} on the same machine as in $\mathcal{S}_{\infty,\text{int}}(J')$.

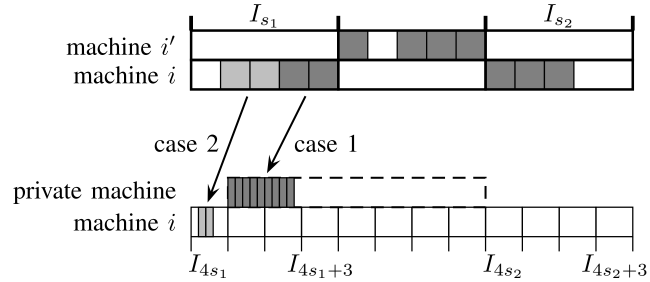

Fig. 5. Transformation of the schedule $\mathcal{S}_{\infty,\text{int}}(J')$ (top) to $\mathcal{S}_\infty(J)$ (bottom), where $\mathcal{S}_\infty(J)$ is compressed by a factor of 4. Here a "private" machine for a job j means the machine never processes any job other than j.

See Figure 5 for a visualization. Then $\mathcal{S}_\infty(J)$ is a valid schedule for $\mathsf{P}\infty \mid \mathsf{prec}, c \mid C_{\max}$, with makespan $\leq 4c \cdot T$. Moreover, $\mathcal{S}_\infty(J)$ satisfies the following:

(a) A short job is fully contained in some interval I_s.
(b) A long job's start time is at the beginning of some interval I_s.

For $\mathcal{S}_\infty(J)$, define a new job set H. Every long job j becomes an element of H with its original running time p_j. Meanwhile, every set of short jobs that are assigned to the same machine in one time interval becomes an element of H, with running time equal to the sum of running times of the short jobs merged. To summarize, a new job $h \in H$ corresponds to a set $h \subseteq J$ and $p_h = \sum_{j \in h} p_j$.

We define the partial order $\tilde{\prec}$ on H with $h_1 \tilde{\prec} h_2$ if and only if there are $j_1 \in h_1$ and $j_2 \in h_2$ with $j_1 \prec j_2$. One can check that this partial order is well defined. Moreover, by the fact that jobs assigned to the same interval but different machines do not have precedence constraints, the length of the longest chain in $(H, \tilde{\prec})$ in terms of the number of elements is bounded by the number of intervals that are used, which is at most $4T$.

Now run Graham's list scheduling on the new job set H with order $\tilde{\prec}$ and m machines. By Lemma 19, the makespan of the list scheduling is bounded by $\frac{1}{m}\sum_{h \in H} p_h + \max_{Q \in \mathcal{Q}(H)} \{\sum_{h \in Q} p_h + c \cdot |Q|\}$. As the total sum of the processing times does not change from J to H, we see that $\frac{1}{m}\sum_{h \in H} p_h \leq \mathrm{OPT}_m(J)$. Moreover, for any chain $Q \in \mathcal{Q}(H)$, $\sum_{h \in Q} p_h$ is no greater than the makespan of $\mathcal{S}_\infty(J)$, which is $4cT$. Finally, as argued earlier, the chain has

$|Q| \leq 4T$ elements. Above all,

$$\begin{aligned}
\frac{1}{m} \sum_{h \in H} p_h + \max_{Q \in \mathcal{Q}(H)} \Big\{ \sum_{h \in Q} p_h + c \cdot |Q| \Big\} \\
&\leq \mathrm{OPT}_m(J) + 4cT + 4cT \\
&\leq \mathrm{OPT}_m(J) + 8\alpha \cdot cS \\
&\leq \mathrm{OPT}_m(J) + 16\alpha \cdot \mathrm{OPT}_m(J) \\
&= O(\alpha) \cdot \mathrm{OPT}_m(J).
\end{aligned}$$

□

Discussion and Open Problems

We gave a new framework for scheduling jobs with precedence constraints and communication delays based on metric space clustering. Our results take the first step towards resolving several important problems in this area. One immediate open question is to understand whether our approach can yield a constant-factor approximation for $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$. A more challenging problem is to handle *non-uniform* communication delays in the problem $\mathsf{P} \mid \mathsf{prec}, c_{jk} \mid C_{\mathsf{max}}$, where c_{jk} is the communication delay between jobs $j \prec k$.

References

[1] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Ann. Discrete Math.*, vol. 4, pp. 287–326, 1979.

[2] B. Veltman, B. Lageweg, and J. Lenstra, "Multiprocessor scheduling with communication delays," *Parallel Computing*, vol. 16, no. 2, pp. 173 – 182, 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/016781919090056F

[3] R. L. Graham, "Bounds for certain multiprocessing anomalies," *Bell System Technical Journal*, vol. 45, no. 9, pp. 1563–1581, 1966. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1538-7305.1966.tb01709.x

[4] J. K. Lenstra and A. H. G. Rinnooy Kan, "Complexity of scheduling under precedence constraints," *Oper. Res.*, vol. 26, no. 1, pp. 22–35, Feb. 1978.

[5] O. Svensson, "Conditional hardness of precedence constrained scheduling on identical machines," in *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, ser. STOC '10. New York, NY, USA: Association for Computing Machinery, 2010, pp. 745–754. [Online]. Available: https://doi.org/10.1145/1806689.1806791

[6] N. Bansal and S. Khot, "Inapproximability of hypergraph vertex cover and applications to scheduling problems," in *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part I*, ser. Lecture Notes in Computer Science, vol. 6198. Springer, 2010, pp. 250–261.

[7] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," in *Proceedings of the ACM SIGCOMM 2011 Conference*, ser. SIGCOMM ?11. Association for Computing Machinery, 2011, p. 98?109.

[8] Z. Guo, X. Fan, R. Chen, J. Zhang, H. Zhou, S. McDirmid, C. Liu, W. Lin, J. Zhou, and L. Zhou, "Spotting code optimizations in data-parallel pipelines through periscope," in *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, 2012, pp. 121–133.

[9] C. Hong, M. Caesar, and P. Godfrey, "Finishing flows quickly with preemptive scheduling," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 127–138, 2012.

[10] A. Shymyrbay, A. Zhanbolatov, A. Amankhan, A. Bakambekova, and I. Ukaegbu, "Meeting deadlines in datacenter networks: An analysis on deadline-aware transport layer protocols," in *2018 International Conference on Computing and Network Communications (CoCoNet)*. IEEE, 2018, pp. 152–158.

[11] J. Zhang, H. Zhou, R. Chen, X. Fan, Z. Guo, H. Lin, J. Li, W. Lin, J. Zhou, and L. Zhou, "Optimizing data shuffling in data-parallel computation by understanding user-defined functions," in *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, 2012, pp. 295–308.

[12] 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 *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 424–432.

[13] S. Luo, H. Yu, Y. Zhao, S. Wang, S. Yu, and L. Li, "Towards practical and near-optimal coflow scheduling for data center networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 11, pp. 3366–3380, 2016.

[14] D. Narayanan, A. Harlap, A. Phanishayee, V. Seshadri, N. Devanur, G. Ganger, P. Gibbons, and M. Zaharia, "Pipedream: Generalized pipeline parallelism for dnn training," in *Proc. 27th ACM Symposium on Operating Systems Principles (SOSP)*, Huntsville, ON, Canada, Oct. 2019.

[15] Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu, and Z. Chen, "Gpipe: Efficient training of giant neural networks using pipeline parallelism," in *Advances in Neural Information Processing Systems*, 2019.

[16] V. Rayward-Smith, "Uet scheduling with unit interprocessor communication delays," *Discrete Applied Mathematics*, vol. 18, no. 1, pp. 55 – 71, 1987. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0166218X87900424

[17] C. H. Papadimitriou and M. Yannakakis, "Towards an architecture-independent analysis of parallel algorithms," *SIAM J. Comput.*, vol. 19, no. 2, p. 322328, Apr. 1990. [Online]. Available: https://doi.org/10.1137/0219021

[18] R. Giroudeau, J. König, F. K. Moulai, and J. Palaysi, "Complexity and approximation for precedence constrained scheduling problems with large communication delays," *Theoretical Computer Science*, vol. 401, no. 1, pp. 107 – 119, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304397508002387

[19] P. Schuurman and G. J. Woeginger, "Polynomial time approximation algorithms for machine scheduling: Ten open problems," 1999.

[20] N. Bansal, "Scheduling open problems: Old and new." ser. MAPSP 2017. http://www.mapsp2017.ma.tum.de/MAPSP2017-Bansal.pdf, 2017.

[21] B. Maiti, R. Rajaraman, D. Stalfa, Z. Svitkina, and A. Vijayaraghavan, "Scheduling precedence-constrained jobs on related machines with communication delay," *To appear at 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS 2020)*, 2020.

[22] A. Munier and J. König, "A heuristic for a scheduling problem with communication delays," *Operations Research*, vol. 45, no. 1, pp. 145–147, 1997. [Online]. Available: https://doi.org/10.1287/opre.45.1.145

[23] G. Călinescu, H. Karloff, and Y. Rabani, "Approximation algorithms for the 0-extension problem," *SIAM J. Comput.*, vol. 34, no. 2, pp. 358–372, 2004. [Online]. Available: https://doi.org/10.1137/S0097539701395978

[24] M. Pinedo, *Scheduling: theory, algorithms, and systems*. Springer, 2018.

[25] E. Lawler, J. Lenstra, A. Rinnooy Kan, and D. Shmoys, "Sequencing and scheduling: Algorithms and complexity," *Handbooks in operations research and management science*, vol. 4, pp. 445–522, 1993.

[26] K. Pruhs, J. Sgall, and E. Torng, "Online scheduling," in *Handbook of Scheduling - Algorithms, Models, and Performance Analysis*, J. Y. Leung, Ed. Chapman and Hall/CRC, 2004. [Online]. Available: http://www.crcnetbase.com/doi/abs/10.1201/9780203489802.ch15

[27] C. Ambühl, M. Mastrolilli, N. Mutsanas, and O. Svensson, "Precedence constraint scheduling and connections to dimension theory of partial orders," in *Bulletin of the European Association for Theoretical Computer Science (EATCS*. Citeseer, 2008.

[28] O. Svensson, "Approximability of some classical graph and scheduling problems," Ph.D. dissertation, Università della Svizzera italiana, 2009.

[29] C. Hanen and A. Munier, "An approximation algorithm for scheduling dependent tasks on m processors with small communication delays," *Discrete Applied Mathematics*, vol. 108, no. 3, pp. 239 – 257, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166218X00001797

[30] R. THURIMELLA, "A scheduling principle for precedence graphs with communication delay," *International Conference on Parallel Processing*, vol. 3, pp. 229–236, 1992. [Online]. Available: https://ci.nii.ac.jp/naid/10010438009/en/

[31] E. Levey and T. Rothvoss, "A (1+epsilon)-approximation for makespan scheduling with precedence constraints using LP hierarchies," in *Pro-*

ceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016, 2016, pp. 168–177.

[32] J. Kulkarni, S. Li, J. Tarnawski, and M. Ye, *Hierarchy-Based Algorithms for Minimizing Makespan under Precedence and Communication Constraints*, 2020, pp. 2770–2789. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611975994.169

[33] J. Hoogeveen, J. Lenstra, and B. Veltman, "Three, four, five, six, or the complexity of scheduling with communication delays," *Operations Research Letters*, vol. 16, no. 3, pp. 129–137, 1994.

[34] E. Bampis, A. Giannakos, and J. König, "On the complexity of scheduling with large communication delays," *European Journal of Operational Research*, vol. 94, no. 2, pp. 252 – 260, 1996. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0377221796001245

[35] R. Lepere and C. Rapine, "An asymptotic o(ln ρ/ ln ln ρ)-approximation algorithm for the scheduling problem with duplication on large communication delay graphs," in *STACS 2002*, H. Alt and A. Ferreira, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 154–165.

[36] M. Drozdowski, *Scheduling with Communication Delays*. London: Springer London, 2009, pp. 209–299. [Online]. Available: https://doi.org/10.1007/978-1-84882-310-5_6

[37] R. Giroudeau and J. König, "Scheduling with communication delays," in *Multiprocessor Scheduling*, E. Levner, Ed. Rijeka: IntechOpen, 2007, ch. 4. [Online]. Available: https://www.intechopen.com/books/multiprocessor_scheduling_theory_and_applications/scheduling_with_communication_delays

[38] J. Y. Colin and P. Chrétienne, "C.p.m. scheduling with small communication delays and task duplication," *Operations Research*, vol. 39, no. 4, pp. 680–684, 1991. [Online]. Available: https://doi.org/10.1287/opre.39.4.680

[39] H. Jung, L. Kirousis, and P. Spirakis, "Lower bounds and efficient algorithms for multiprocessor scheduling of directed acyclic graphs with communication delays," *Information and Computation*, vol. 105, no. 1, pp. 94 – 104, 1993. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0890540183710412

[40] A. Munier and C. Hanen, "Using duplication for scheduling unitary tasks on m processors with unit communication delays," *Theoretical Computer Science*, vol. 178, no. 1, pp. 119 – 127, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304397597881947

[41] M. Laurent, "A comparison of the sherali-adams, lovász-schrijver, and lasserre relaxations for 0-1 programming," *Math. Oper. Res.*, vol. 28, no. 3, pp. 470–496, 2003. [Online]. Available: https://doi.org/10.1287/moor.28.3.470.16391

[42] Z. Friggstad, J. Könemann, Y. Kun-Ko, A. Louis, M. Shadravan, and M. Tulsiani, "Linear programming hierarchies suffice for directed steiner tree," in *Integer Programming and Combinatorial Optimization - 17th International Conference, IPCO 2014, Bonn, Germany, June 23-25, 2014. Proceedings*, 2014, pp. 285–296. [Online]. Available: https://doi.org/10.1007/978-3-319-07557-0_24

[43] A. Karlin, C. Mathieu, and C. T. Nguyen, "Integrality gaps of linear and semi-definite programming relaxations for knapsack," in *Integer Programming and Combinatoral Optimization - 15th International Conference, IPCO 2011, New York, NY, USA, June 15-17, 2011. Proceedings*, 2011, pp. 301–314. [Online]. Available: https://doi.org/10.1007/978-3-642-20807-2_24

[44] T. Rothvoß, "Directed steiner tree and the lasserre hierarchy," *CoRR*, vol. abs/1111.5473, 2011. [Online]. Available: http://arxiv.org/abs/1111.5473

[45] M. Mendel and A. Naor, "Ramsey partitions and proximity data structures," in *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, 2006, pp. 109–118. [Online]. Available: https://doi.org/10.1109/FOCS.2006.65

[46] J. Fakcharoenphol, S. Rao, and K. Talwar, "A tight bound on approximating arbitrary metrics by tree metrics," *J. Comput. Syst. Sci.*, vol. 69, no. 3, pp. 485–497, 2004. [Online]. Available: https://doi.org/10.1016/j.jcss.2004.04.011

[47] T. Leighton and S. Rao, "Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms," *J. ACM*, vol. 46, no. 6, p. 787832, Nov. 1999. [Online]. Available: https://doi.org/10.1145/331524.331526

[48] N. Garg, V. V. Vazirani, and M. Yannakakis, "Approximate max-flow min-(multi)cut theorems and their applications," *SIAM Journal on Computing*, vol. 25, pp. 698–707, 1993.

[49] D. S. Hochbaum and D. B. Shmoys, "Using dual approximation algorithms for scheduling problems theoretical and practical results," *J. ACM*, vol. 34, no. 1, pp. 144–162, 1987. [Online]. Available: https://doi.org/10.1145/7531.7535