Scheduling Under Non-Uniform Job and Machine Delays

Rajmohan Rajaraman
Northeastern University, Boston, MA, USA

David Stalfa
Northeastern University, Boston, MA, USA

Sheng Yang
Shanghai, CN

Abstract

We study the problem of scheduling precedence-constrained jobs on heterogenous machines in the presence of non-uniform job and machine communication delays. We are given a set of n unit size precedence-ordered jobs, and a set of m related machines each with size m_i (machine i can execute at most m_i jobs at any time). Each machine i has an associated in-delay ρ_i^{in} and out-delay ρ_i^{out}. Each job v also has an associated in-delay ρ_v^{in} and out-delay ρ_v^{out}. In a schedule, job v may be executed on machine i at time t if each predecessor u of v is completed on i before time t or on any machine j before time $t - (\rho_i^{\text{in}} + \rho_j^{\text{out}} + \rho_u^{\text{out}} + \rho_v^{\text{in}})$. The objective is to construct a schedule that minimizes makespan, which is the maximum completion time over all jobs.

We consider schedules which allow duplication of jobs as well as schedules which do not. When duplication is allowed, we provide an asymptotic polylog(n)-approximation algorithm. This approximation is further improved in the setting with uniform machine speeds and sizes. Our best approximation for non-uniform delays is provided for the setting with uniform speeds, uniform sizes, and no job delays. For schedules with no duplication, we obtain an asymptotic polylog(n)-approximation for the above model, and a true polylog(n)-approximation for symmetric machine and job delays. These results represent the first polylogarithmic approximation algorithms for scheduling with non-uniform communication delays.

Finally, we consider a more general model, where the delay can be an arbitrary function of the job and the machine executing it: job v can be executed on machine i at time t if all of v's predecessors are executed on i by time $t-1$ or on any machine by time $t - \rho_{v,i}$. We present an approximation-preserving reduction from the Unique Machines Precedence-constrained Scheduling (UMPS) problem, first defined in [15], to this job-machine delay model. The reduction entails logarithmic hardness for this delay setting, as well as polynomial hardness if the conjectured hardness of UMPS holds.

This set of results is among the first steps toward cataloging the rich landscape of problems in non-uniform delay scheduling.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms

Keywords and phrases Scheduling, Approximation Algorithms, Precedence Constraints, Communication Delay, Non-Uniform Delays

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.98

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2207.13121` [42]

Funding *Rajmohan Rajaraman*: Partially supported by NSF grant CCF-1909363.
David Stalfa: Partially supported by NSF grant CCF-1909363.
Sheng Yang: Work done when the author was at Northwestern University, supported by Samir Khuller's funding.

1 Introduction

With the increasing scale and complexity of scientific and data-intensive computations, it is often necessary to process workloads with many dependent jobs on a network of heterogeneous computing devices with varying computing capabilities and communication delays. For instance, the training and evaluation of neural network models, which involves iterations of precedence constrained jobs, is often distributed over diverse devices such as CPUs, GPUs, or other specialized hardware. This process, commonly referred to as *device placement*, has gained significant interest [18, 21, 32, 33]. Similarly, many scientific workflows are best modeled precedence constrained jobs, and the underlying high-performance computing system as a heterogeneous networked distributed system with communication delays [3, 44, 49].

Optimization problems associated with scheduling under communication delays have been studied extensively, but provably good approximation bounds are few and several challenging open problems remain [1, 4, 14, 23, 26, 34, 35, 37, 39, 40, 43]. With a communication delay, scheduling a set of precedence constrained uniform size jobs on identical machines is already NP-hard [40, 43], and several inapproximability results are known [4, 23]. However, the field is still underexplored and scheduling under communication delay was listed as one of the top ten open problems in scheduling surveys [5, 45]. While there has been progress on polylogarthmic-approximation algorithms for the case of uniform communication delays [16, 26, 29, 31], little is known for more general delay models.

This paper considers the problem of scheduling precedence-constrained jobs on machines connected by a network with *non-uniform* communication delays. In general, the delay incurred in communication between two machines could vary with the machines as well as with the data being communicated, which in turn may depend on the jobs being excuted on the machines. For many applications, however, simpler models suffice. For instance, the machine delays model, where the communication between two machines incurs a delay given by the sum of latencies associated with the two machines, is suitable when the bottleneck is primarily at the machine interfaces. On the other hand, job delays model scenarios where the delay incurred in the communication between two jobs running on two different machines is a function primarily of the two jobs. This is suitable when the communication is data-intensive. Recent work in [15] presents a hardness result for a model in which jobs are given as a DAG and any edge of the DAG separating two jobs running on different machines causes a delay, providing preliminary evidence that obtaining sub-polynomial approximation factors for this model may be intractable. Given polylogarithmic approximations for uniform delays, a natural question is which, if any, non-uniform delay models are tractable.

1.1 Overview of our results

A central contribution of this paper is to explore and catalog a rich landscape of problems in non-uniform delay scheduling. We present polylogarithmic approximation algorithms for several models with non-uniform delays, and a hardness result in the mold of [15] for a different non-uniform delay model. Figure 2 organizes various models in this space, with pointers to results in this paper and relevant previous work.

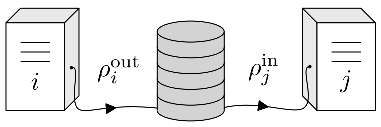

Figure 1 Communicating a result from i to j takes $\rho_i^{\text{out}} + \rho_j^{\text{in}}$ time.

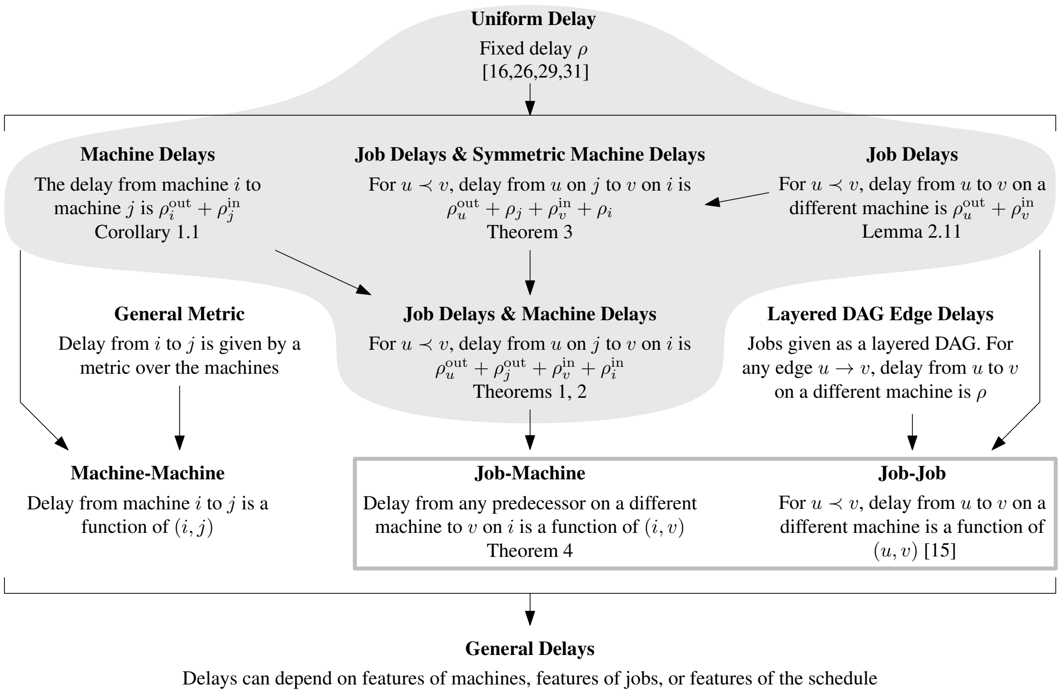
Uniform Delay
Fixed delay ρ
[16,26,29,31]
Machine Delays
The delay from machine i to machine j is $\rho_i^{\text{out}} + \rho_j^{\text{in}}$
Corollary 1.1
Job Delays & Symmetric Machine Delays
For $u \prec v$, delay from u on j to v on i is $\rho_u^{\text{out}} + \rho_j + \rho_v^{\text{in}} + \rho_i$
Theorem 3
Job Delays
For $u \prec v$, delay from u to v on a different machine is $\rho_u^{\text{out}} + \rho_v^{\text{in}}$
Lemma 2.11
General Metric
Delay from i to j is given by a metric over the machines
Job Delays & Machine Delays
For $u \prec v$, delay from u on j to v on i is $\rho_u^{\text{out}} + \rho_j^{\text{out}} + \rho_v^{\text{in}} + \rho_i^{\text{in}}$
Theorems 1, 2
Layered DAG Edge Delays
Jobs given as a layered DAG. For any edge $u \to v$, delay from u to v on a different machine is ρ
Machine-Machine
Delay from machine i to j is a function of (i, j)
Job-Machine
Delay from any predecessor on a different machine to v on i is a function of (i, v)
Theorem 4
Job-Job
For $u \prec v$, delay from u to v on a different machine is a function of (u, v) [15]
General Delays
Delays can depend on features of machines, features of jobs, or features of the schedule

Figure 2 Selection of scheduling models with communication delays. $a \rightarrow b$ indicates that a is a special case of b. We present approximation algorithms for models with machine delays and job delays, and a hardness of approximation result for the job-machine delays model. Theorems and citations point to results in this paper and in previous work, respectively. Those problems backed in gray are ones for which approximation algorithms are known. Those in the gray box are ones for which hardness results have been proven.

Machine delays and job delays (Section 2). We begin with a natural model where the delay incurred in communication from one machine to another is the sum of delays at the two endpoints. Under machine delays, each machine i has an in-delay ρ_i^{in} and out-delay ρ_i^{out}, and the time taken to communicate a result from i to j is $\rho_i^{\text{out}} + \rho_j^{\text{in}}$. This model, illustrated in Figure 1, is especially suitable for environments where data exchange between jobs occurs via the cloud, an increasingly common mode of operation in modern distributed systems [28,30,50]; ρ_i^{in} and ρ_i^{out} represent the cloud download and upload latencies, respectively, for machine i.

The machine delays model does not account for heterogeneity among jobs, where different jobs may be producing or consuming different amounts of data, which may impact the delay between the processing of one job and that of another dependent job on a different machine. To model this, we allow each job u to have an in-delay ρ_u^{in} and an out-delay ρ_u^{out}.

▶ **Definition 1** (Scheduling under Machine Delays and Job Delays). *We are given as input a set of n precedence ordered jobs and a set of m machines. For any jobs u and v with $u \prec v$, machine i, and time t, u is available to v on i at time t if u is completed on i before time t or on any machine j before time $t - (\rho_j^{\text{out}} + \rho_u^{\text{out}} + \rho_i^{\text{in}} + \rho_v^{\text{in}})$. (This model is illustrated in Figure 3.) If job v is scheduled at time t on machine i, then all of its predecessors must be available to v on i at time t. We define $\rho_{\max} = \max_{x \in V \cup M}\{\rho_x^{\text{in}} + \rho_x^{\text{out}}\}$. The objective is to construct a schedule that minimizes makespan.*

Remark. In our model of Definition 1, communication delay is defined over all pairs of precedence ordered jobs. An alternate model defines communication delay only over those pairs that are adjacent in the job DAG. The two settings differ in general but are equivalent

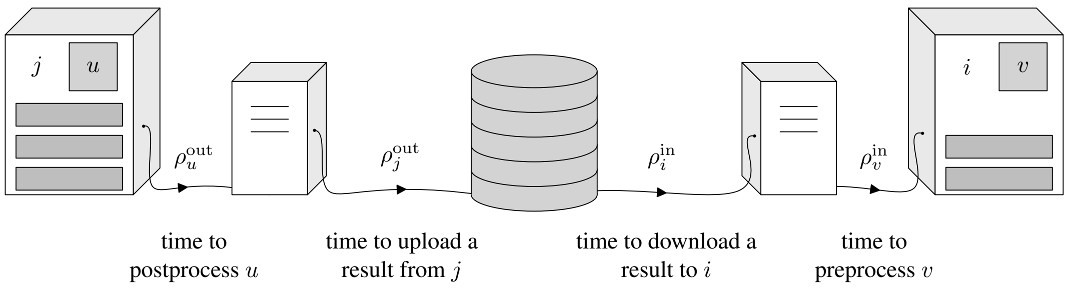

Figure 3 Communicating the result of job u on machine j to execute job v on machine i.

in many scenarios, for instance, when the delays are given by an underlying metric space over the machines, or when communication delays are uniform. The models are equivalent if all delays are machine delays, so our machine delay results hold in the alternate model. The models differ in the presence of general job delays but are equivalent in several special cases, for instance in the setting where the job DAG is transitively closed, which has been extensively studied and proved useful in several important applications [2,19,46]. Transitively closed DAGs capture scenarios where each job may be generating data used by upstream jobs, and an upstream job may need to check the results of any of its predecessors. Examples of such graphs arising in scheduling include interval orders [38], as well as Solution Order Graphs in the context of SAT solvers [8].

We present the first approximation algorithms for scheduling under non-uniform communication delays. In the presence of delays, a natural approach to hide latency and reduce makespan is to duplicate some jobs (for instance, a job that is a predecessor of many other jobs) [1,39]. We consider both schedules that allow duplication (which we assume by default) and those that do not. Our first result is a polylogarithmic asymptotic approximation for scheduling under machine and job delays when duplication is allowed.

▶ **Theorem 1.** There exists a polynomial time algorithm for scheduling unit length, precedence constrained jobs with duplication under machine and job delays, that produces a schedule with makespan $O((\log^9 n)(\textsc{opt} + \rho_{\max}))$.

We emphasize that if the makespan of any schedule includes the delays incurred in distributing the problem instance and collecting the output of the jobs, then the algorithm of Theorem 1 is, in fact, a *true polylogarithmic approximation* for makespan. (From a practical standpoint, in order to account for the time incurred to distribute the jobs and collect the results, it is natural to include in the makespan the in- and out-delays of every machine used in the schedule.)

We note that when delays are uniform and duplication is not allowed, it is easy to check if $\textsc{opt} < \rho$ since any connected component of the job DAG must be placed on the same machine. This is demonstrated in our true approximation without duplication in Theorem 3. In the presence of duplication, the problem is closely related to the Min k-Union problem, for which conditional hardness proofs are known [12]. This motivates the additive $\rho_{\max}$ in our approximation guarantee.

Related machines and multiprocessors. Theorem 1 is based on a new linear programming framework for addressing non-uniform job and machine delays. We demonstrate the power and flexibility of this approach by incorporating two more aspects of heterogeneity: speed and number of processors. Each machine i has a number m_i of processors and a speed s_i at which each processor processes jobs. We generalize Theorem 1 to obtain the following result.

▶ **Theorem 2.** There exists a polynomial time algorithm for scheduling unit length, precedence constrained jobs with duplication on related multiprocessor machines under machine and job delays, that yields a schedule with makespan $\mathrm{polylog}(n)(\textsc{opt} + \rho_{\max})$.

The exact approximation factor obtained depends on the non-uniformity of the particular model. For the most general model we consider in Theorem 2, our proof achieves a $O(\log^{15} n)$ bound. We obtain improved bounds when any of the three defining parameters – size, speed, and delay – are uniform. For instance, we obtain an approximation factor of $O(\log^5 n)$ for scheduling uniform speed and uniform size machines under machine delays alone, i.e., when there are no job delays (Corollary 12 of Section 2). Further, with only job delays and uniform machine delays, we provide a combinatorial asymptotic $O(\log^6 n)$ approximation (Lemma 15 of Section 2) which is improved to an asymptotic $O(\log n)$ approximation if the input contains no out-delays. We note that despite some uniformity, special cases can model certain two-level non-uniform network hierarchies with processors at the leaves, low delays at the first level, and high delays at the second level.

No-duplication schedules. We next consider the problem of designing schedules that do not allow duplication. We obtain a polylogarithmic asymptotic approximation via a reduction to scheduling with duplication. Furthermore, if the delays are symmetric (i.e., $\rho_i^{\mathrm{out}} = \rho_i^{\mathrm{in}}$ for all i, and $\rho_v^{\mathrm{out}} = \rho_v^{\mathrm{in}}$ for all v) we are able to find a *true* polylogarithmic-approximate no-duplication schedule. To achieve this result, we present an approximation algorithm to estimate if the makespan of an optimal no-duplication schedule is at least the delay of any given machine; this enables us to identify machines that cannot communicate in the desired schedule.[1]

▶ **Theorem 3.** There exists a polynomial time algorithm for scheduling unit length, precedence constrained jobs on related multiprocessor machines under machine delays and job delays, which produces a no-duplication schedule with makespan $\mathrm{polylog}(n)(\textsc{opt} + \rho_{\max})$. If $\rho_i^{\mathrm{in}} = \rho_i^{\mathrm{out}}$ for all i, then there exists a polynomial time $\mathrm{polylog}(n)$-approximation algorithm for no-duplication schedules.

Pairwise delays. All of the preceding results concern models where the communication associated with a precedence relation $u \prec v$ when u and v are executed on different machines i and j is an *additive* combination of delays at u, v, i, and j. Additive delays are suitable for capturing independent latencies incurred by various components of the system. A more general class of models considers *pairwise* delays where the delay is an *arbitrary function* of i and j (machine-machine), u and v (job-job), or either job and the machine on which it executes (job-machine). The machine-machine delay model captures classic networking scenarios, where the delay across machines is determined by the network links connecting them. Job-job delays model applications where the data that needs to be communicated from one job to another descendant job depends arbitrarily on the two jobs. The job-machine model is well-suited for applications where the delay incurred for communicating the data consumed or produced by a job executing on a machine is an arbitrary function of the size of

[1] We note that the corresponding problem for duplication schedules is a min-max partitioning variant of the Minimum k-Union problem and related to the Min-Max Hypergraph k-Partitioning problem, both of which have been shown to be Densest-k-Subgraph-hard [9, 11]; this might suggest a similar hardness result for deriving a *true* approximation when duplication is allowed.

the data and the bandwidth of the machine. Recent work in [15] shows that scheduling under job-job delays is as hard as the Unique Machine Precedence Scheduling (UMPS) problem, providing preliminary evidence that obtaining sub-polynomial approximation factors may be intractable. We show that UMPS also reduces to scheduling under job-machine delays, suggesting a similar inapproximability for this model.

▶ **Theorem 4 (umps reduces to scheduling under job-machine delays).** There is a polynomial-time approximation-preserving reduction from UMPS to the scheduling precedence constrained jobs under job-machine delays.

1.2 Overview of our techniques

Our approximation algorithms for scheduling under job delays and machine delays (Theorem 1 proved in Section 2) and the generalization to related machines and multiprocessors (Theorem 2 proved in [42]) rely on a framework composed of a carefully crafted linear programming relaxation and a series of reductions that help successively reduce the level of heterogeneity in the problem. While each individual component of the framework refines established techniques or builds on prior work, taken together they offer a flexible recipe for designing approximation algorithms for scheduling precedence-ordered jobs on a distributed system of heterogeneous machines with non-uniform delays. Given the hardness conjectures of [15] for the job-job delay setting (and for the job-machine setting via Theorem 4), we find it surprising that a fairly general model incorporating both job delays and machine delays on related machines is tractable.

Previous results on scheduling under (uniform) communication delays are based on three different approaches: (a) a purely combinatorial algorithm of [26] that works only for uniform delay machines; (b) an LP-based approach of [31] that handles related machines and uniform delays, assuming jobs can be duplicated, and then extends to no-duplication via a reduction; and (c) an approach of [16] based on a Sherali-Adams hierarchy relaxation followed by a semi-metric clustering, which directly tackles the no-duplication model. At a very high level, our main challenge, which is not addressed in any of the previous studies, is to tackle the *multi-dimensional heterogeneity* of the problem space: in the nature of delays (non-uniform values, in- and out-delays, job delays, machine delays) as well as the machines (delay, speed, and size).

We pursue an LP-based framework, which significantly refines the approach of [31]. Their algorithm organizes the computation in phases, each phase corresponding to a (uniform) delay period, and develops a linear program that includes delay constraints capturing when jobs have to be phase-separated and phase constraints bounding the amount of computation within a phase. In non-uniform delay models, the delay constraints for a job v executing on a machine i depend not only on the predecessors of v, but also on the machines on which they may be scheduled. While there is a natural way to account for non-uniform in-delays in the LP, incorporating out-delays or even symmetric delays poses technical difficulties. We overcome this hurdle by first showing that out-delays can be eliminated by suitably adjusting in-delays, at the expense of a polylogarithmic factor in approximation, thus allowing us to focus on in-delays.

Despite the reduction to in-delays, extending the LP of [31] by replacing the uniform delay parameter by the non-uniform delay parameters of our models fails and yields a high integrality gap. This is because their algorithm crucially relies on an ordering of the machines (on the basis of their speeds), which is exploited both in the LP (in the delay and phase constraints) as well as how jobs get assigned and moved in the computation of the final schedule. Given the multi-dimensional heterogeneity of the problems we study, there is no such natural ordering of the machines. To address the above hurdle, we organize the machines and jobs into groups based on their common characteristics (delay, speed, size), and introduce new variables for assigning jobs to groups without regard to any ordering among them. This necessitates new load and delay constraints and a change in rounding and schedule construction. We now elaborate on these ideas, as we discuss our new framework in more detail.

Reduction to in-delays. The first ingredient of our recipe is an argument that any instance of the problem with machine delays and job delays can be reduced to an instance in which all out-delays are 0, meaning that in the new instance delays depend only on the machine and job receiving the data, at the expense of a polylogarithmic factor in approximation. This reduction is given in Lemma 37 and Algorithm 2 in [42]. To convert from a given schedule with out-delays to one without, we subtract $\rho_i^{\text{out}} + \rho_v^{\text{out}}$ from the execution time of every job v on machine i. However, in order to avoid collisions, we expand the given schedule into phases of different length, organized in particular sequence so that the execution times within each phase may be reduced without colliding with prior phases. This transforms the schedule into one where the in-delay of every machine i is $\rho_i^{\text{in}} + \rho_i^{\text{out}}$ and every job v is $\rho_v^{\text{in}} + \rho_v^{\text{out}}$. This transformation comes at a constant factor cost for machine delays and an $O(\log^2 \rho_{max})$ cost for job delays. A similar procedure converts from an in-delay schedule to one with in- and out-delays, completing the desired reduction.

The linear program (Sections 2.1–2.2). Before setting up the linear program, we partition the machines and the jobs into groups of uniform machines and jobs, respectively; i.e. each machine in a group can be treated as having the same in-delay, speed, and size (to within a constant factor), and each job in a group can be treated as having the same in-delay. The final approximation factor for the most general model grows as K^3 and L, where L is the number of job groups and K is the number of machine groups, which depends on the extent of heterogeneity among the machines. We bound K by $O(\log^3 n)$ in the case when the speeds, sizes, and delays of machines are non-uniform. We emphasize that, even with the machines partitioned in this way, we must carefully design our LP to judiciously distribute jobs among the groups depending on the precedence structure of the jobs and the particular job and machine parameters.

Our LP is inspired by that of [31], though significant changes are necessary to allow for non-uniform delays. The key constraints of each LP are presented below (with the constraints from [31] rewritten to include machine group variables). Here, C^* represents the makespan of the schedule and C_v represents the earliest execution time of job v. $x_{v,k}$ indicates if v is placed on a machine in group $\langle k \rangle$ $(= 1)$ or not $(= 0)$. $z_{u,v,k}$ indicates whether $x_{v,k} = 1$ and $C_v - C_u$ is less the time it takes to communicate the result of u from a different machine. $y_{v,k}$ takes the maximum of $x_{v,k}$ and $\max_u\{z_{v,u,k}\}$ to indicate whether some copy of v is executed on a machine in group $\langle k \rangle$ $(= 1)$ or not $(= 0)$. Other notation used in the linear program is explained in Section 2.

One main difference between our LP and that of [31] is in the constraint that regulates the completion time of precedence ordered jobs in the presence of communication delay.

Delay Constraint in [31] New Delay Constraint

$$C_v \geq C_u + \rho\Big(\sum_{k' \leq k} x_{v,k'} - z_{u,v,k}\Big) \quad \Rightarrow \quad C_v \geq C_u + (\bar{\rho}_k + \bar{\rho}_\ell)(x_{v,k} - z_{u,v,k})$$

$$\forall u, v, k : u \prec v \qquad\qquad \forall u, v, k, \ell : u \prec v \text{ and } v \in [\![\ell]\!]$$

The constraint of [31] states that if $u \prec v$ and v is executed on a machine in speed group k, then the completion time of v is at least ρ greater than the completion time of u unless some duplicate of u is executed on group k. The summation over machine groups orders the groups by increasing speeds (similar to [13]). It turns out that the rounding technique which uses this ordering of machine groups, which is used to eliminate a log factor in [13, 31], does not straightforwardly work in our context. The new constraint has an interpretation similar to that of the delay constraint in [31]: if $u \prec v$ and v is executed on delay group k, then the completion time of v is at least the in-delay of k plus the in-delay of v greater than the completion time of u, unless some duplicate of u is also executed on group k. However, in the new constraint, the summation over machine groups has been replaced by a single machine group assignment variable.

The next change to the linear program regards the constraint which governs how many jobs can be duplicated within a communication phase for a single job.

Phase Constraint in [31] New Phase Constraint

$$\rho \geq \sum_{u \prec v} z_{u,v,k} \qquad \forall v, k \qquad \Rightarrow \qquad (\bar{\rho}_k + \bar{\rho}_\ell) \sum_{u} z_{u,v,k} \qquad \forall v, k, \ell : v \in [\![\ell]\!]$$

Both the old and new constraints state that the amount of duplication that can be performed for a single job within a single communication phase on a given group of machines is at most the length of the phase. The new constraint also incorporates the machine and job in-delays.

The final change is to the constraints which lower bound the makespan of the schedule by the total load placed on a single machine.

Load Constraint in [31] New Load Constraints

$$C^* \cdot |\langle k \rangle| \geq \sum_{v} x_{v,k} \qquad \forall k \qquad \Rightarrow \qquad C^* \cdot |\langle k \rangle| \geq \sum_{v} y_{v,k} \qquad \forall k$$

$$y_{v,k} \geq x_{v,k} \qquad \forall v, k$$

$$y_{u,k} \geq z_{u,v,k} \qquad \forall u, v, k$$

Both constraints state that the makespan is at least the total number of jobs placed on any group divided by the size of the group. The old constraint uses $x_{v,k}$ as the sole indicator of whether or not a job is placed on machine group k, and does not need to account for duplicates because of the optimized rounding scheme which utlizes the ordering of job groups by increasing speed. Because the new constraint cannot rely on this ordering, we use the y-variables to account for all duplicates as well.

In [31], the ordering of the groups was leveraged to construct the final schedule by always placing a job on higher capacity groups than the one to which it is assigned by the LP. Since the LP assigns all jobs to some group, we can infer that the total load over all groups does not increase by more than a constant factor. With multidimensional heterogenous machines, there is no clear ordering of machine groups to achieve a similar property (e.g. one set of jobs may be highly parallelizable, while another requires a single fast machine). Using the new LP, our solution is to place all jobs on those groups to which the LP assigns them, along with any predecessors indicated by the z-variables. However, such a construction could vastly exceed the value of the LP unless the load contributed by the z-variables is counted toward the LP makespan. To this end, we introduce the y-variables and associated constraints, which account for this additional, duplicated load. In the most general setting, we also introduce constraints which govern the amount of duplication possible within a single communication phase. These additional constrains model an optimal schedule of the duplicated jobs on the uniform machines within a single group.

Rounding the LP solution and determining final schedule (Sections 2.3–2.4). The next component rounds an optimal LP solution to an integer solution by placing each job on the group for which the job's LP mass is maximized. We also place duplicate predecessors of each job v on its group according to the z-variables for v's predecessors. This indicates a key difference with [31], where the load contributed by duplicates was handled by the ordering of the machines. A benefit of our simple rounding is that it accommodates many different machine and job properties as long as the number of groups can be kept small. Finally, we construct a schedule using the integer LP solution. This subroutine divides the set of jobs assigned to each group into phases and constructs a schedule for each phase by invoking a schedule for the uniform machines case, appending each schedule to the existing schedule for the entire instance.

No-duplication schedules. The proof of the first part of Theorem 3 extends an asympototic polylogarithmic approximation to no-duplication schedules for machine delays and job delays. The theorem follows from the structure of the schedule designed in Theorem 2 and a general reduction in [31] from duplication to no-duplication schedules in the uniform delay case. Avoiding the additive delay penalty of the first part of Theorem 3 to achieve a true approximation is much more difficult. When delays are symmetric (i.e., in-delays equal out-delays), we can distinguish those machines whose delay is low enough to communicate with other machines from those machines with high delay. One of the central challenges is then to distribute jobs among the high-delay machines. We overcome this difficulty by revising the LP in the framework of Theorem 2 to partition the jobs among low- and high-delay machines, and rounding the corresponding solutions separately.

We then must distinguish between those jobs with delay low enough to communicate with other jobs from those with high delay. We note that any predecessor or successor of a high delay job must be executed on the same machine as that job. We leverage this fact to construct our schedule, first placing all high delay jobs with their predecessors and successors on individual machines. We then run our machine and job delay algorithm with the remaining jobs on the low delay machines. This schedule is placed after the execution of the downward closed high-delay components, and before the upward closed high-delay components, ensuring that the schedule is valid.

We note that the design of no-duplication schedules via a reduction to duplication schedules incurs a loss in approximation factor of an additional polylogarithmic factor. While this may not be desirable in a practical implementation, our results demonstrate the flexibility of the approach and highlight its potential for more general delay models.

Hardness for job-machine delay model. The algorithmic framework outlined above incorporates non-uniform job and machine delays that combine additively. It is natural to ask if the techniques extend to other delay combinations or more broadly to pairwise delay models. In the job-machine delay model we study, when a job u executed on machine i precedes job v executed on machine j, then a delay $\rho_{v,j}$ between the two executions is incurred. Our reduction from UMPS to the job-machine delay problem follows the approach of [15] by introducing new jobs with suitable job-machine delay parameters that essentially force each job to be executed on a particular machine. This reduction does not require the flexibility of assigning different delays for different job-job pairs, but it is unclear if the same technique can be applied to machine-machine delay models. Delineating the boundary between tractable models and those for which polylogarithmic approximations violate conjectured complexity lower bounds is a major problem of interest.

1.3 Related work

Precedence constrained scheduling. The problem of scheduling precedence-constrained jobs was initiated in the classic work of Graham who gave a constant approximation algorithm for uniform machines [20]. Jaffe presented an $O(\sqrt{m})$ makespan approximation for the case with related machines [24]. This was improved upon by Chudak and Shmoys who gave an $O(\log m)$ approximation [13], then used the work of Hall, Schulz, Shmoys, and Wein [22] and Queyranne and Sviridenko [41] to generalize the result to an $O(\log m)$ approximation for weighted completion time. Chekuri and Bender [10] proved the same bound as Chudak and Shmoys using a combinatorial algorithm. In subsequent work, Li improved the approximation factor to $O(\log m/\log\log m)$ [27]. The problem of scheduling precedence-constrained jobs is hard to approximate even for identical machines, where the constant depends on complexity assumptions [6, 25, 47]. Also, Bazzi and Norouzi-Fard [7] showed a close connection between structural hardness for k-partite graph and scheduling with precedence constraints.

Precedence constrained scheduling under communication delays. Scheduling under communication delays has been studied extensively [39, 43, 48]. For unit size jobs, identical machines, and unit delay, a (7/3)-approximation is given in [35], and [23] proves the NP-hardness of achieving better than a 5/4-approxmation. Other hardness results are given in [4, 40, 43]. More recently, Davies, Kulkarni, Rothvoss, Tarnawski, and Zhang [16] give an $O(\log \rho \log m)$ approximation in the identical machine setting using an LP approach based on Sherali-Adams hierarchy, which is extended to include related machines in [17]. Concurrently, Maiti, Rajaraman, Stalfa, Svitkina, and Vijayaraghavan [31] provide a polylogarithmic approximation for uniform communication delay with related machines as a reduction from scheduling with duplication. The algorithm of [31] is combinatorial in the case with identical machines.

Davies, Kulkarni, Rothvoss, Sandeep, Tarnawski, and Zhang [15] consider the problem of scheduling precedence-constrained jobs on uniform machine in the presence of non-uniform, job-pairwise communication delays. That is, if $u \prec v$ and u and v are scheduled on different machines, then the time between their executions is at least $\rho_{u,v}$. The authors reduce to this problem from Unique-Machines Precedence-constrained Scheduling (UMPS) in which there is no communication delay, but for each job there is some particular machine on which that job must be placed. The authors show that UMPS is hard to approximate to within a logarithmic factor by a reduction from job-shop scheduling, and conjecture that UMPS is hard to approximate within a polynomial factor.

Precedence constrained scheduling under communication delays with job duplication. Using duplication with communication delay first studied by Papadimitriou and Yannakakis [39], who give a 2-approximation for DAG scheduling with unbounded processors and fixed delay. Improved bounds for infinite machines are given in [1, 14, 36, 37]. Approximation algorithms are given by Munier and Hanen [34, 35] for special cases in which the fixed delay is very small or very large, or the DAG restricted to a tree. The first bounds for a bounded number of machines are given by Lepere and Rapine [26] who prove an asymptotic $O(\log \rho/\log\log \rho)$ approximation. Recent work has extended their framework to other settings: [31] uses duplication to achieve an $O(\log \rho \log m/\log\log \rho)$ approximation for a bounded number of related machines, and Liu, Purohit, Svitkina, Vee, and Wang [29] improve on the runtime of [26] to a near linear time algorithm with uniform delay and identical machines.

1.4 Discussion and open problems

Our results indicate several directions for further work. First, we conjecture that our results extend easily to the setting with non-uniform job sizes. We believe the only barriers to such a result are the techinical difficulties of tracking the completion times of very large jobs that continue executing long after they are placed on a machine. Also, while our approximation ratios are the first polylogarithmic guarantees for scheduling under non-uniform delays, we have not attempted to optimize logarithmic factors. There are obvious avenues for small reductions in our ratio, e.g. the technique used in [26] to reduce the ratio by a factor of $\log\log\rho$. More substantial reduction, however, may require a novel approach. Additionally, in the setting without duplication, we incur even more logarithmic factors owing to our reduction to scheduling with duplication. These factors may be reduced by using a more direct method, possibly extending the LP-hierarchy style approach taken in [16, 17].

Aside from improvements to our current results, our techniques suggest possible avenues to solve related non-uniform delay scheduling problems. A special case of general machine metrics is a machine hierarchy, where machines are given as leaves in a weighted tree. Our incorporation of parallel processors allows our results to apply to a two-level machine hierarch. We would like to explore extensions of our framework to constant-depth hierarchies and tree metrics. More generally, scheduling under metric and general machine-machine delays remains wide open (see Figure 2).

We also believe there are useful analogs to these machine delay models in the job-pairwise regime. A job v with in-delay ρ_v^{in} and out-delay ρ_v^{out} has the natural interpretation of the data required to execute a job, and the data produced by a job. A job tree hierarchy could model the shared libraries required to execute certain jobs: jobs in different subtrees require different resources to execute, and downloading these additional resources incurs a delay. Given the hardness conjectures of [15] and our hardness result for the job-machine delay model, further refining Figure 2 and exploring the tractability boundary would greatly enhance our understanding of scheduling under non-uniform delays.

Finally, recall that our notion of job delays is defined in terms of the precedence relation over the jobs. Another natural notion of job delay may be to consider a DAG defined over the jobs, with a delay incurred only if there is a directed edge $u \to v$ (rather than $u \prec v$). In this setting, while our results do not hold in the presence of general job delays, they do hold for some significant special cases. These include instances where the job DAG is transitively closed, or where job delays are uniform, or where job delays of predecessors are at most that of their successors (i.e. $u \prec v$ implies $\rho_u^{\text{out}} \le \rho_v^{\text{out}}$ and $\rho_u^{\text{in}} \le \rho_v^{\text{in}}$), or where there are only machine delays. However, resolving the most general case is an interesting open problem since this family of delay models provides an intuitive and important set of problems.

2 Machine Delays and Job Delays

In this section, we present an asymptotic approximation algorithm for scheduling under machine delays and job delays for unit speed and size machines. As discussed in Section 1.2, we can focus on the setting with no out-delays, at the expense of a polylogarithmic factor in approximation; Lemma 37 of [42] presents the reduction to in-delays. Therefore, in this section, we assume that $\rho_i^{\text{out}} = 0$ for all machines i and $\rho_v^{\text{out}} = 0$ for all jobs v. For convenience, we use ρ_i to denote the in-delay ρ_i^{in} of machine i and ρ_v to denote the in-delay ρ_v^{in} of machine v. Let $\rho_{\max} = \max\{\max_v\{\rho_v\}, \max_i\{\rho_i\}\}$.

2.1 Partitioning machines and jobs into groups

In order to simplify our exposition and analysis, we introduce a new set of machines M' with rounded delays. For each $i \in M$, if $2^{k-1} \leq \rho_i < 2^k$, we introduce $i' \in M'$ with $\rho_{i'} = 2^k$. We then partition M' according to machine delays: machine $i \in M'$ is in $\langle k \rangle$ if $\rho_i = 2^k$; we set $\bar{\rho}_k = 2^k$. We also introduce a new set of jobs V' with rounded delays. For each $v \in V$, if $2^{\ell-1} \leq \rho_v < 2^\ell$, we introduce $v' \in V'$ with $\rho_{v'} = 2^\ell$. We then partition V' according to job delays: job $v \in V'$ is in $[\![\ell]\!]$ if $\rho_v = 2^\ell = \bar{\rho}_\ell$. For the remainder of the section, we work with the machine set M' and the job set V', ensuring that all machines or jobs within a group have identical delays. As shown in the following lemma, this partitioning is at the expense of at most a constant factor in approximation.

▶ **Lemma 2.** *The optimal makespan over the machine set V', M' is no more than a factor of 2 greater than the optimal solution over V, M.*

Proof. Consider any schedule σ on the machine set M. We first show that increasing the delay of each machine by a factor of 2 increases the makespan of the schedule by at most a factor of 2. We define the schedule σ' as follows. For every i, t, if $(i, t) \in \sigma(v)$, then $(i, 2t) \in \sigma'(v)$. It is easy to see that σ' maintains the precedence ordering of jobs, and that the time between the executions of any two jobs has been doubled. Therefore, σ' is a valid schedule with all communication delays doubled, and with the makespan doubled. ◀

We can assume that $\max_k\{\bar{\rho}_k\} \leq n$ since if we ever needed to communicate to a machine with delay greater than n we could schedule everything on a single machine in less time. Therefore, we have $K \leq \log n$ machine groups. Similarly, $\max_\ell\{\bar{\rho}_\ell\} \leq n$, implying that we have $L \leq \log n$ job groups.

2.2 The linear program

In this section, we design a linear program LP$_\alpha$ – Equations (1-11) – parametrized by $\alpha \geq 1$, for machine delays. Following Section 2.1, we assume that the machines and jobs are organized in groups, where each group $\langle k \rangle$ (resp., $[\![\ell]\!]$) is composed of machines (resp., jobs) that have identical delay.

$$C^*_\alpha \geq C_v \qquad \forall v \qquad (1)$$

$$C^*_\alpha \cdot |\langle k \rangle| \geq \sum_v y_{v,k} \qquad \forall k \qquad (2)$$

$$C_v \geq C_u + (\bar{\rho}_k + \bar{\rho}_\ell)(x_{v,k} - z_{u,v,k}) \qquad \forall u, v, k, \ell : u \prec v, v \in [\![\ell]\!] \qquad (3)$$

$$C_v \geq C_u + 1 \qquad \forall u, v : u \prec v \qquad (4)$$

$$\alpha(\bar{\rho}_k + \bar{\rho}_\ell) \geq \sum_u z_{u,v,k} \qquad \forall v, k, \ell : v \in [\![\ell]\!] \qquad (5)$$

$$\sum_k x_{v,k} = 1 \qquad \forall v \qquad (6)$$

$$C_v \geq 0 \qquad \forall v \qquad (7)$$

$$x_{v,k} \geq z_{u,v,k} \qquad \forall u, v, k \qquad (8)$$

$$y_{v,k} \geq x_{v,k} \qquad \forall v, k \qquad (9)$$

$$y_{u,k} \geq z_{u,v,k} \qquad \forall u, v, k \qquad (10)$$

$$z_{u,v,k} \geq 0 \qquad \forall u, v, k \qquad (11)$$

Variables. C^*_α represents the makespan of the schedule. For each job v, C_v represents the earliest completion time of v. For each job v and group $\langle k \rangle$, $x_{v,k}$ indicates whether or not v is first executed on a machine in group $\langle k \rangle$. For each $\langle k \rangle$ and pair of jobs u, v such that $u \prec v$ and $v \in [\![\ell]\!]$, $z_{u,v,k}$ indicates whether v is first executed on a machine in group $\langle k \rangle$ and the earliest execution of u is less that $\bar{\rho}_k + \bar{\rho}_\ell$ time before the execution of v. Intuitively, $z_{u,v,k}$ indicates whether there must be a copy of u executed on the same machine that first executes v. For each job v and group $\langle k \rangle$, $y_{v,k}$ indicates whether $x_{v,k} = 1$ or $z_{u,v,k} = 1$ for some u; that is, whether or not some copy of v is placed on group $\langle k \rangle$. Constraints (7 - 11) guarantee that all variables are non-negative.

Makespan (2, 1). Constraint 1 states that the makespan is at least the maximum completion time of any job. Constraint 2 states that the makespan is at least the load on any single group.

Delays (3, 5). Constraint 3 states that the earliest completion time of $v \in [\![\ell]\!]$ must be at least $\bar{\rho}_k + \bar{\rho}_\ell$ after the earliest completion time of any predecessor u if v is first executed on a machine in group $\langle k \rangle$ and no copy of u is duplicated on the same machine as v. Constraint 5 limits the amount of duplication that can be done to improve the completion time of any job: if $v \in [\![\ell]\!]$ first executes on a machine in group $\langle k \rangle$ at time t, then the number of predecessors that may be executed in the $\bar{\rho}_k + \bar{\rho}_\ell$ steps preceding t is at most $\bar{\rho}_k$.

The remaining constraints enforce standard scheduling conditions. Constraint 4 states that the completion time of v is at least the completion time of any of its predecessors, and constraint 6 ensures that every job is executed on some group. Constraints 6 and 8 guarantee that $z_{u,v,k} \leq 1$ for all u, v, k. This is an important feature of the LP, since a large z-value could be used to disproportionately reduce the delay between two jobs in constraint 3.

▶ **Lemma 3** (LP$_1$ is a valid relaxation)**.** *The minimum of C_1^* is at most* OPT.

Proof. Consider an arbitrary schedule σ with makespan C_σ, i.e. $C_\sigma = \max_{v,i,t}\{t : (i,t) \in \sigma(v)\}$.

LP solution. Set $C_1^* = C_\sigma$. For each job v, set C_v to be the earliest completion time of v in σ, i.e. $C_v = \min_{i,t}\{t : (i,t) \in \sigma(v)\}$. Set $x_{v,k} = 1$ if $\langle k \rangle$ is the group that contains the machine on which v first completes (choosing arbitrarily if there is more than one) and 0 otherwise. For u, v, k, set $z_{u,v,k} = 1$ if $u \prec v$, $x_{v,k} = 1$, $v \in [\![\ell]\!]$, and $C_v - C_u < \bar{\rho}_k + \bar{\rho}_\ell$ (0 otherwise). Set $y_{u,k} = \max\{x_{u,k}, \max_v\{z_{u,v,k}\}\}$.

Feasibility. We now establish that the solution defined is feasible. Constraints (1, 7–11) are easy to verify. We now establish constraints (2–5). Consider constraint 2 for fixed group $\langle k \rangle$. $\sum_v y_{v,k}$ is upper bound by the total load Λ on $\langle k \rangle$. The constraint follows from $C_\alpha^* \geq C_\sigma \geq \Lambda/|\langle k \rangle|$.

Consider constraint 3 for fixed u, v, k where $u \prec v$. Let $X = x_{v,k}$ and let $Z = z_{u,v,k}$. If $(X,Z) = (0,0), (0,1)$, or $(1,1)$ then the constraint follows from constraint 4. If $(X,Z) = (1,0)$, then by the assignment of $z_{u,v,k}$ we can infer that $C_v - C_u \geq \bar{\rho}_k + \bar{\rho}_\ell$, which shows the constraint is satisfied.

Consider constraint 5 for fixed v, k. If $x_{v,k} = 0$ then the result follows from the fact that $z_{u,v,k} = 0$ for all u. If $x_{v,k} = 1$, then we can infer that $v \in [\![\ell]\!]$. So, at most $\bar{\rho}_k + \bar{\rho}_\ell$ predecessors of v that can be scheduled in the $\bar{\rho}_k + \bar{\rho}_\ell$ time before C_v, ensuring that the constraint is satisfied. ◀

2.3 Deriving a rounded solution to the linear program

▶ **Definition 4.** *(C, x, y, z) is a* rounded solution *to LP_α if all values of x, y, z are either 0 or 1.*

Let LP_1 be defined over machine groups $\langle 1\rangle, \langle 2\rangle, \ldots, \langle K\rangle$ and job groups $[\![1]\!], [\![2]\!], \ldots, [\![L]\!]$. Given a solution $(\hat{C}, \hat{x}, \hat{y}, \hat{z})$ to LP_1, we construct an integer solution (C, x, y, z) to LP_{2K} as follows. For each v, k, set $x_{v,k} = 1$ if $k = \max_{k'}\{\hat{x}_{v,k'}\}$ (if there is more than one maximizing k, arbitrarily select one); set to 0 otherwise. Set $z_{u,v,k} = 1$ if $x_{v,k} = 1$ and $\hat{z}_{u,v,k} \geq 1/(2K)$; set to 0 otherwise. For all u, k, $y_{u,k} = \max\{x_{u,k}, \max_v\{z_{u,v,k}\}\}$. Set $C_v = 2K \cdot \hat{C}_v$. Set $C^*_{2K} = 2K \cdot \hat{C}^*_1$.

▶ **Lemma 5.** *If $(\hat{C}, \hat{x}, \hat{y}, \hat{z})$ is a valid solution to LP_1, then (C, x, y, z) is a valid solution to LP_{2K}.*

Proof. By constraint (6), $\sum_k \hat{x}_{v,k}$ is at least 1, so $\max_k\{\hat{x}_{v,k}\}$ is at least $1/K$. Therefore, $x_{v,k} \leq K\hat{x}_{v,k}$ for all v and k. Also, $z_{u,v,k} \leq 2K\hat{z}_{u,v,k}$ for any u, v, k by definition. By the setting of C_v for all v, $y_{v,k}$ for all v, k, and C^*_{2K}, it follows that constraints (1, 4-11) of LP_1 imply the respective constraints of LP_{2K}. We first establish constraint (2). For any fixed group $\langle k\rangle$,

$$\begin{aligned} 2K\hat{C}_1 \cdot |\langle k\rangle| &\geq 2K \sum_v \hat{y}_{v,k} = 2K \sum_v \max\{\hat{x}_{v,k}, \max_u\{\hat{z}_{v,u,k}\}\} && \text{by constraints 2, 11 of } \mathrm{LP}_1 \\ &\geq 2K \sum_v \frac{x_{v,k} + \max_u\{z_{v,u,k}\}}{2K} \geq \sum_v y_{v,k} && \text{by definition of } y_{v,k} \end{aligned}$$

which entails constraint (2) by $C^*_{2K} = 2K\hat{C}^*_1$. It remains to establish constraint (3) for fixed u, v, k. We consider two cases. If $x_{v,k} - z_{u,v,k} \leq 0$, then the constraint is trivially satisfied in LP_{2K}. If $x_{v,k} - z_{u,v,k} = 1$, then, by definition of x and z, $\hat{x}_{v,k} - \hat{z}_{u,v,k}$ is at least $1/(2K)$. This entails that $\hat{C}_v \geq \hat{C}_u + ((\bar{\rho}_k + \bar{\rho}_\ell)/2K)$ which establishes constraint (3) of LP_{2K} by definition of C_v and C_u. ◀

▶ **Lemma 6.** *$C_{2K} \leq 4K \cdot$ OPT.*

Proof. Lemma 2 shows that our grouping of machines does not increase the value of the LP by more than a factor of 2. Therefore, by Lemmas 3 and 5, $C_{2K} = 2K \cdot \hat{C}_1 \leq 4K \cdot$ OPT. ◀

2.4 Computing a schedule given an integer solution to the LP

Suppose we are given a partition of M into K groups such that group $\langle k\rangle$ is composed of identical machines (i.e. for all $i, j \in \langle k\rangle$, $\rho_i = \rho_j$). Also, suppose we are given a partition of V into L groups such that group $[\![\ell]\!]$ is composed of jobs with identical in-delay. Finally, we are given a rounded solution (C, x, y, z) to LP_α defined over machine groups $\langle 1\rangle, \ldots, \langle K\rangle$ and job groups $[\![1]\!], \ldots [\![L]\!]$. In this section, we show that we can construct a schedule that achieves an approximation for machine delays in terms of α, K, and L. The combinatorial subroutine that constructs the schedule is defined in Algorithm 1. In the algorithm, we use a subroutine UDPS-Solver for Uniform Delay Precedence-Constrained Scheduling. An $O(\log \rho/\log\log \rho)$-asympototic approximation is given in [26]. For completeness, we use the UDPS-Solver presented and analyzed in [42], which generalizes the algorithm of [26] to incorporate non-uniform machine sizes.

We now describe Algorithm 1 informally. The subroutine takes as input the rounded LP_α solution (C, x, y, z) and initializes an empty schedule σ and global parameters T, θ to 0. For a fixed value of T, we iterate through all machine groups $\langle k\rangle$ and job groups $[\![\ell]\!]$, with decreasing ℓ. For a fixed value of T, k, ℓ, we check if there is some integer d such that $T = d(\bar{\rho}_k + \bar{\rho}_\ell)$. If so, we define $V_{k,\ell,d}$ and $U_{k,\ell,d}$ as in lines 4 and 5. $V_{k,\ell,d}$ represents the set of jobs in $[\![\ell]\!]$ assigned by the LP to machine group $\langle k\rangle$ in a single phase of length $\bar{\rho}_k + \bar{\rho}_\ell$.

Algorithm 1 Machine Delay Scheduling with Duplication.

```
  Init: $\forall v,\ \sigma(v) \leftarrow \varnothing;\ T \leftarrow 0;\ \theta \leftarrow 0$
1 while $T \leq C^*_\alpha$ do
2     forall machine groups $\langle k \rangle$ do
3         for job group $[\![\ell]\!] = [\![L]\!]$ to $[\![1]\!]$: $\exists$ integer $d,\ T = d(\bar\rho_k + \bar\rho_\ell)$ do
4             $V_{k,\ell,d} \leftarrow \{v \in [\![\ell]\!] : x_{v,k} = 1 \text{ and } T \leq C_v < T + \bar\rho_k + \bar\rho_\ell\}$
5             $U_{k,\ell,d} \leftarrow \{u : \exists v \in V_{k,\ell,d},\ u \prec v \text{ and } T \leq C_u < T + \bar\rho_k + \bar\rho_\ell\}$
6             $\sigma' \leftarrow$ UDPS-Solver on $(V_{k,\ell,d} \cup U_{k,\ell,d}, \langle k \rangle, \bar\rho_k + \bar\rho_\ell)$
7             $\forall v, i, t,$ if $(i,t) \in \sigma'(v)$ then $\sigma(v) \leftarrow \sigma(v) \cup \{(i, \theta + \bar\rho_k + \bar\rho_\ell + t)\}$
8             $\theta \leftarrow \theta + 2(\bar\rho_k + \bar\rho_\ell)$
9     $T \leftarrow T + 1$
```

$U_{k,\ell,d}$ represents predecessors of $V_{k,\ell,d}$ whose LP completion times are within $\bar\rho_k + \bar\rho_\ell$ of their successor in $V_{k,\ell,d}$. We then call UDPS-Solver to construct a UDPS schedule σ' on jobs $V_{k,\ell,d} \cup U_{k,\ell,d}$, machines in $\langle k \rangle$, and delay $\bar\rho_k + \bar\rho_\ell$. We then append σ' to σ. Once all values of k, ℓ have been checked, we increment T and repeat until all jobs are scheduled. The structure of the schedule produced by Algorithm 1 is depicted in Figure 4. Lemma 7 (entailed by Lemma 45 of [42]) provides guarantees for the UDPS-Solver subroutine.

▶ **Lemma 7.** *Let U be a set of η jobs such that for any $v \in U$, $|\{u \in U : u \prec v\}| \leq \alpha\delta$. Given input U, a set of μ identical machines, and delay δ, UDPS-Solver produces, in polynomial time, a valid UDPS schedule with makespan at most $3\alpha\delta \log(\alpha\delta) + (2\eta/\mu)$.*

▶ **Lemma 8.** *Algorithm 1 outputs a valid schedule in polynomial time.*

Proof. It is easy to see that the algorithm runs in polynomial time, and Lemma 7 entails that precedence constraints are obeyed on each machine. Consider a fixed v, k, d such that $v \in V_{k,\ell,d}$. By line 7, we insert a communication phase of length $\bar\rho_k + \bar\rho_\ell$ before appending the schedule of any set of jobs $V_{k,\ell,d} \cup U_{k,\ell,d}$ on any machine group $\langle k \rangle$. So, by the time Algorithm 1 executes any job in $V_{k,\ell,d}$, every job u such that $C_u < d(\bar\rho_k + \bar\rho_\ell)$ is available to all machines, including those in group $\langle k \rangle$. So the only predecessors of v left to execute are those jobs in $U_{k,\ell,d}$. Therefore, all communication constraints are satisfied. ◀

▶ **Lemma 9.** *If (C, x, y, z) is a rounded solution to LP_α then Algorithm 1 outputs a schedule with makespan at most $12\alpha \log(\rho_{\max})(KLC^*_\alpha + \rho_{\max}(K + L))$.*

Proof. Fix any schedule σ. Note that the schedule produced by the algorithm executes a single job group on a single machine group at a time. Our proof establishes a bound for the total time spent executing a single job group on a single machine group, then sums this bound over all K machine groups and L job groups.

▷ Claim 10. For any v, u, k, ℓ, d, if $v \in V_{k,\ell,d}$ and $C_v < C_u + (\bar\rho_k + \bar\rho_\ell)$ then $z_{u,v,k,\ell} = 1$.

Proof. Fix u, v, k, ℓ, d such that $v \in V_{k,\ell,d}$ and $C_v < C_u + (\bar\rho_k + \bar\rho_\ell)$. By the definition of $V_{k,\ell,d}$, $x_{v,k}$ is 1. By constraint 3, $C_v \geq C_u + \bar\rho_k(1 - z_{u,v,k})$, implying that $z_{u,v,k}$ cannot equal 0. Since $z_{u,v,k}$ is either 0 or 1, we have $z_{u,v,k} = 1$. ◁

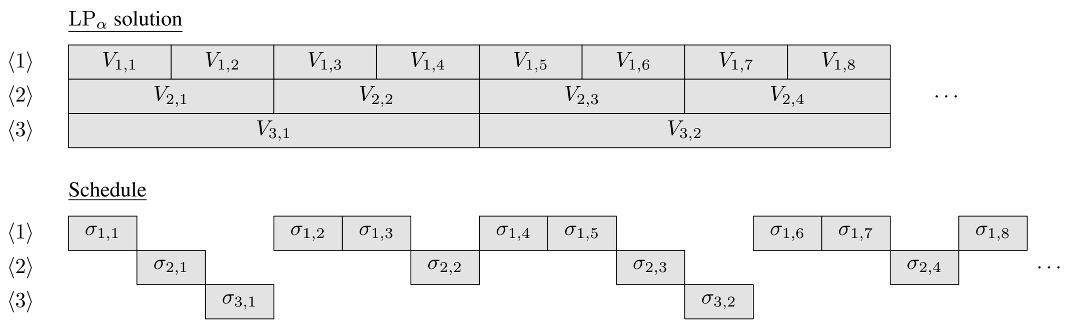

Figure 4 Structure of the schedule produced by Algorithm 1. $\sigma_{k,d}$ denotes a schedule of $V_{k,\ell,d}$ on the machines in group $\langle k\rangle$. The algorithm scans the LP$_\alpha$ solution by increasing time (left to right). At the start of each $V_{k,\ell,d}$, the algorithm constructs a schedule of the set and appends it to the existing schedule.

▷ Claim 11. For any k, ℓ, we show

a $\sum_d |V_{k,\ell,d} \cup U_{k,\ell,d}| \le C^*_\alpha \cdot |\langle k\rangle|$ and

b for any d and $v \in V_{k,\ell,d}$, the number of v's predecessors in $V_{k,\ell,d} \cup U_{k,\ell,d}$ is at most $\alpha(\bar\rho_k + \bar\rho_\ell)$.

Proof. Fix k, ℓ. We first prove a. For any v in $V_{k,\ell,d}$ we have $x_{v,k} = 1$ by the definition of $V_{k,\ell,d}$. Consider any u in $U_{k,\ell,d}$. By definition, there exists a $v' \in V_{k,\ell,d}$ such that $x_{v',k} = 1$ and $C_v < C_u + (\bar\rho_k + \bar\rho_\ell)$; fix such a v'. By claim 10, $z_{u,v',k} = 1$. So, by constraint 10, $y_{v,k} = 1$ for every job $v \in V_{k,\ell,d} \cup U_{k,\ell,d}$. For any $d' \neq d$, $V_{k,\ell,d}$ and $V_{k,\ell,d'}$ are disjoint. So $\sum_d |V_{k,\ell,d} \cup U_{k,\ell,d}|$ is at most the right-hand side of constraint 2, which is at most $C^*_\alpha \cdot |\langle k\rangle|$.

We now prove b. Fix v, d such that $v \in V_{k,\ell,d}$. Consider any u in $V_{k,\ell,d} \cup U_{k,d}$ such that $u \prec v$. By definition of $V_{k,\ell,d}$ and $U_{k,\ell,d}$, $C_v < C_u + (\bar\rho_k + \bar\rho_\ell)$. By Claim 10, $z_{u,v,k} = 1$. The claim then follows from constraint (5). ◁

By Lemma 7 and Claim 11b, the time spent executing jobs in $[\![\ell]\!]$ on machines in $\langle k\rangle$ is at most

$$\sum_d \left(3\alpha(\bar\rho_k + \bar\rho_\ell)\log(\alpha(\bar\rho_k + \bar\rho_\ell)) + \frac{2 \cdot |V_{k,\ell,d} \cup U_{k,\ell,d}|}{|\langle k\rangle|}\right)$$

The summation over the first term is at most $\lceil C^*_\alpha/(\bar\rho_k + \bar\rho_\ell)\rceil\, 3\alpha(\bar\rho_k + \bar\rho_\ell)\log(\alpha(\bar\rho_k + \bar\rho_\ell))$ which is at most $3C^*_\alpha \alpha \log(\alpha(\bar\rho_k + \bar\rho_\ell)) + 3\alpha(\bar\rho_k + \bar\rho_\ell)\log(\alpha(\bar\rho_k + \bar\rho_\ell))$. The summation over the second term is at most $2C^*_\alpha$ by claim 11a. Summing over all K machine groups and L job groups, and considering $K, L \le \log \rho_{\max}$, the total length of the schedule is at most $12\alpha \log(\rho_{\max})(KLC^*_\alpha + \rho_{\max}(K + L))$. ◀

▶ **Theorem 1** (Job Delays and Machine Delays). *There exists a polynomial time algorithm to compute a valid machine delays and job precedence delays schedule with makespan $O((\log n)^9(\text{OPT} + \rho_{\max}))$.*

Proof. Lemma 5 entails that (C, x, y, z) is a valid solution to LP$_{2K}$. Lemma 6 entails that $C^*_{2K} \le 4K \cdot \text{OPT}$. With $\alpha = 2K$, Lemma 9 entails that the makespan of our schedule is at most $12\alpha \log(\rho_{\max})(KLC^*_\alpha + \rho_{\max}(K+L)) = 48(\log \rho_{\max})^5 \text{OPT} + 24(\log \rho_{\max})^3 \rho_{\max}$ for the case with no out-delays. By Lemma 37 of [42], the length of our schedule is $O((\log \rho_{\max})^9(\text{OPT} + \rho_{\max})$ The theorem is entailed by $\rho_{\max} \le n$. This proves the theorem. ◀

▶ **Corollary 12** (Machine Delays). *There exists a polynomial time algorithm to compute a valid machine delays schedule with makespan* $O((\log n)^5 \cdot (\textsc{opt} + \rho))$.

Proof. Lemma 5 entails that (C, x, y, z) is a valid solution to LP_{2K}. Lemma 6 entails that $C^*_{2K} \leq 4K \cdot \textsc{opt}$. With $\alpha = 2K$, Lemma 9 entails that the makespan of our schedule is at most $12\alpha\log(\rho_{\max})(KLC^*_\alpha + \rho_{\max}(K+L)) = 48(\log\rho_{\max})^5\textsc{opt} + 24(\log\rho_{\max})^3\rho_{\max}$ for the case with no out-delays. By Lemma 41 of [42], the length of our schedule is $O((\log\rho_{\max})^5(\textsc{opt} + \rho_{\max})$ The theorem is entailed by $\rho_{\max} \leq n$. ◀

2.5 Combinatorial Algorithm for Uniform Machine Delays

The only noncombinatorial subroutine of our algorithm is solving the linear program. In this section, we describe how to combinatorially construct a rounded solution to LP_1 when machine delays are uniform (i.e. for all i, j, $\rho^{\text{in}}_i = \rho^{\text{out}}_i = \rho^{\text{in}}_j = \rho^{\text{out}}_j$), machine speeds are unit, and machine capacities are unit. We let δ represent the uniform machine delay. By Lemma 37 of [42], we focus on the case where all job out-delays are 0. We let $\rho_v = \delta + \rho^{\text{in}}_v$ for any job v.

Since delays, speeds, and capacities are uniform, there is only one machine group: $\langle 1\rangle$. Set $x_{v,1} = y_{v,1} = 1$ for all v. For each job v, we define C_v as follows. If v has no predecessors, we set $C_v = 0$. Otherwise, we order v's predecessors such that $C_{u_i} \geq C_{u_{i+1}}$. We define $C_v = \max_{1\leq i\leq\rho_v}\{C_{u_i} + i\}$. We set $C^* = \max\{n/m, \max_v\{C_v\}\}$. We set $z_{u,v,1} = 1$ if $u \prec v$ and $C_v - C_u < \rho_v$; and set to 0 otherwise.

▶ **Lemma 13.** $C^* \leq \textsc{opt}$.

Proof. Consider an arbitrary schedule in which t_v is the earliest completion time of any job v. We show that, for any v, $t_v \geq C_v$, which is sufficient to prove the lemma.

We prove the claim by induction on the number of predecessors of v. The claim is trivial if v has no predecessors. Suppose that the claim holds for all of v's predecessors and let $y = \arg\max_{1\leq i\leq\rho_v}\{C_{u_i} + i\}$. Then $C_v = C_{u_y} + y \leq t_{u_y} + y$ (by IH) $= t_y + |\{u_x : 0 \leq x \leq y\}| \leq t_y + \rho_v$. This entails that all jobs $u_1, \ldots u_y$ must be executed on the same machine as v. Now suppose, for the sake of contradiction, that $t_v < C_v$. Then all jobs $u \in \{u_x : 0 \leq x < y\}|$ must be executed serially in the time $t_v - t_{u_y} < C_v - t_{u_y} = |\{u_x : 0 \leq x \leq y\}|$ which gives us our contradiction. ◀

▶ **Lemma 14.** (C, x, y, z) *is a rounded solution to* LP_1.

Proof. It is easy to see that constraints (1, 2, 3, 4, 6, 7, 8, 9, 10 11) are satisfied by the assignment. So we must only show that constraint (5) is satisfied for fixed v. We can see from the definition of C_v, that maximum number of predecessors u such that $C_v - C_u < \rho_v + \rho$ is at most $\rho_v + \rho$. This proves the lemma. ◀

▶ **Lemma 15** (Combinatorial Algorithm for Job Delays). *There exists a purely combinatorial, polynomial time algorithm to compute a schedule for Job Delays with makespan* $O((\log n)^6(\textsc{opt} + \max_v\{\rho_v\}))$.

Proof. Lemma 9 entails that the length of the schedule is at most $12(\log\rho_{\max})^2(\textsc{opt} + \rho_{\max})$ for the problem with job in-delays. By Lemma 37 of [42] we achieve a makespan of $O((\log\rho_{\max})^6(\textsc{opt} + \rho_{\max}))$ for job in- and out-delays. ◀

References

1 Ishfaq Ahmad and Yu-Kwong Kwok. On exploiting task duplication in parallel program scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 9(9):872–892, September 1998. doi:10.1109/71.722221.

2 Adil Amirjanov and Konstantin Sobolev. Scheduling of directed acyclic graphs by a genetic algorithm with a repairing mechanism. *Concurrency and Computation: Practice and Experience*, 29(5):e3954, 2017. e3954 CPE-16-0237.R1. doi:10.1002/cpe.3954.

3 Pau Andrio, Adam Hospital, Javier Conejero, Luis Jordá, Marc Del Pino, Laia Codo, Stian Soiland-Reyes, Carole Goble, Daniele Lezzi, Rosa M Badia, Modesto Orozco, and Josep Gelpi. Bioexcel building blocks, a software library for interoperable biomolecular simulation workflows. *Scientific data*, 6(1):1–8, 2019.

4 Evripidis Bampis, Aristotelis Giannakos, and Jean-Claude König. On the complexity of scheduling with large communication delays. *European Journal of Operational Research*, 94:252–260, 1996.

5 Nikhil Bansal. Scheduling open problems: Old and new. The 13th Workshop on Models and Algorithms for Planning and Scheduling Problems (MAPSP 2017), 2017. URL: http://www.mapsp2017.ma.tum.de/MAPSP2017-Bansal.pdf.

6 Nikhil Bansal and Subhash Khot. Optimal long code test with one free bit. *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 453–462, October 2009. doi:10.1109/focs.2009.23.

7 Abbas Bazzi and Ashkan Norouzi-Fard. Towards tight lower bounds for scheduling problems. *Lecture Notes in Computer Science*, pages 118–129, 2015. doi:10.1007/978-3-662-48350-3_11.

8 Gregor Behnke, Daniel Höller, and Susanne Biundo. Bringing order to chaos – a compact representation of partial order in sat-based htn planning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7520–7529, July 2019. doi:10.1609/aaai.v33i01.33017520.

9 Karthekeyan Chandrasekaran and Chandra Chekuri. Min-max partitioning of hypergraphs and symmetric submodular functions. In *Proceedings of the Thirty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '21, pages 1026–1038, USA, 2021. Society for Industrial and Applied Mathematics.

10 Chandra Chekuri and Michael Bender. An efficient approximation algorithm for minimizing makespan on uniformly related machines. *Journal of Algorithms*, 41(2):212–224, November 2001. doi:10.1006/jagm.2001.1184.

11 Eden Chlamtáč, Michael Dinitz, and Yury Makarychev. Minimizing the union: Tight approximations for small set bipartite vertex expansion. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 881–899. SIAM, 2017.

12 Eden Chlamtáč, Michael Dinitz, and Yury Makarychev. Minimizing the union: Tight approximations for small set bipartite vertex expansion. In *Proceedings of the 2017 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 881–899, 2017.

13 Fabián A. Chudak and David B. Shmoys. Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds. *Journal of Algorithms*, 30(2):323–343, 1999.

14 Sekhar Darbha and Dharma P. Agrawal. Optimal scheduling algorithm for distributed-memory machines. *IEEE Transactions on Parallel and Distributed Systems*, 9:87–95, 1998.

15 Sami Davies, Janardhan Kulkarni, Thomas Rothvoss, Sai Sandeep, Jakub Tarnawski, and Yihao Zhang. On the hardness of scheduling with non-uniform communication delays. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2958–2977, 2022. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611976465.176.

16 Sami Davies, Janardhan Kulkarni, Thomas Rothvoss, Jakub Tarnawski, and Yihao Zhang. Scheduling with communication delays via lp hierarchies and clustering. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 822–833, 2020. doi:10.1109/FOCS46700.2020.00081.

17 Sami Davies, Janardhan Kulkarni, Thomas Rothvoss, Jakub Tarnawski, and Yihao Zhang. Scheduling with communication delays via lp hierarchies and clustering ii: Weighted completion times on related machines. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2958–2977, 2021. `doi:10.1137/1.9781611976465.176`.

18 Yuanxiang Gao, Li Chen, and Baochun Li. Spotlight: Optimizing device placement for training deep neural networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1676–1684. PMLR, 10–15 July 2018. URL: `https://proceedings.mlr.press/v80/gao18a.html`.

19 M. R. Garey and David S. Johnson. Scheduling tasks with nonuniform deadlines on two processors. *J. ACM*, 23(3):461–467, 1976. `doi:10.1145/321958.321967`.

20 Ronald L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17:416–429, 1969.

21 Ubaid Ullah Hafeez, Xiao Sun, Anshul Gandhi, and Zhenhua Liu. Towards optimal placement and scheduling of dnn operations with pesto. In *Proceedings of the 22nd International Middleware Conference*, pages 39–51, 2021.

22 Leslie A. Hall, Andreas S. Schulz, David B. Shmoys, and Joel Wein. Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Mathematics of Operations Research*, 22(3):513–544, August 1997. `doi:10.1287/moor.22.3.513`.

23 J.A. Hoogeveen, Jan Karel Lenstra, and Bart Veltman. Three, four, five, six, or the complexity of scheduling with communication delays. *Operations Research Letters*, 16(3):129–137, 1994. `doi:10.1016/0167-6377(94)90024-8`.

24 Jeffrey M. Jaffe. Efficient scheduling of tasks without full use of processor resources. *Theoretical Computer Science*, 12(1):1–17, September 1980. `doi:10.1016/0304-3975(80)90002-x`.

25 Jan Karel Lenstra and A. H. G. Rinnooy Kan. Complexity of scheduling under precedence constraints. *Operations Research*, 26(1):22–35, 1978. `doi:10.1287/opre.26.1.22`.

26 Renaud Lepere and Christophe Rapine. An asymptotic $\mathcal{O}(\ln \rho / \ln \ln \rho)$-approximation algorithm for the scheduling problem with duplication on large communication delay graphs. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 154–165. Springer, 2002.

27 Shi Li. Scheduling to minimize total weighted completion time via time-indexed linear programming relaxations. In *2017 58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 283–294, 2017. `doi:10.1109/FOCS.2017.34`.

28 Guanfeng Liang and Ulaş C. Kozat. Fast cloud: Pushing the envelope on delay performance of cloud storage with coding. *IEEE/ACM Transactions on Networking*, 22(6):2012–2025, December 2014. `doi:10.1109/TNET.2013.2289382`.

29 Quanquan C. Liu, Manish Purohit, Zoya Svitkina, Erik Vee, and Joshua R. Wang. Scheduling with communication delay in near-linear time. In *STACS*, 2022.

30 Ashraf Mahgoub, Edgardo Barsallo Yi, Karthick Shankar, Eshaan Minocha, Sameh Elnikety, Saurabh Bagchi, and Somali Chaterji. Wisefuse: Workload characterization and dag transformation for serverless workflows. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 6(2), June 2022. `doi:10.1145/3530892`.

31 Biswaroop Maiti, Rajmohan Rajaraman, David Stalfa, Zoya Svitkina, and Aravindan Vijayaraghavan. Scheduling precedence-constrained jobs on related machines with communication delay. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 834–845, 2020. `doi:10.1109/FOCS46700.2020.00082`.

32 Azalia Mirhoseini, Anna Goldie, Hieu Pham, Benoit Steiner, Quoc V. Le, and Jeff Dean. Hierarchical planning for device placement. In *International Conference on Learning Representations*, 2018. URL: `https://openreview.net/pdf?id=Hkc-TeZ0W`.

33 Azalia Mirhoseini, Hieu Pham, Quoc V. Le, Benoit Steiner, Rasmus Larsen, Yuefeng Zhou, Naveen Kumar, Mohammad Norouzi, Samy Bengio, and Jeff Dean. Device placement optimization with reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, pages 2430–2439, 2017. URL: `http://proceedings.mlr.press/v70/mirhoseini17a.html`.

34 Alix Munier. Approximation algorithms for scheduling trees with general communication delays. *Parallel Computing*, 25(1):41–48, 1999.

35 Alix Munier and Claire Hanen. Using duplication for scheduling unitary tasks on m processors with unit communication delays. *Theoretical Computer Science*, 178(1):119–127, 1997. doi:10.1016/S0304-3975(97)88194-7.

36 Alix Munier and Jean-Claude König. A heuristic for a scheduling problem with communication delays. *Operations Research*, 45(1):145–147, 1997.

37 Michael A. Palis, Jing-Chiou Liou, and David S. L. Wei. Task clustering and scheduling for distributed memory parallel architectures. *IEEE Transactions on Parallel and Distributed Systems*, 7(1):46–55, 1996.

38 Christos H. Papadimitriou and Mihalis Yannakakis. Scheduling interval-ordered tasks. *SIAM J. Comput.*, 8(3):405–409, 1979. doi:10.1137/0208031.

39 Christos H. Papadimitriou and Mihalis Yannakakis. Towards an architecture-independent analysis of parallel algorithms. *SIAM journal on computing*, 19(2):322–328, 1990.

40 Christophe Picouleau. *Two new NP-complete scheduling problems with communication delays and unlimited number of processors*. Inst. Blaise Pascal, Univ., 1991.

41 Maurice Queyranne and Maxim Sviridenko. Approximation algorithms for shop scheduling problems with minsum objective. *Journal of Scheduling*, 5(4):287–305, 2002. doi:10.1002/jos.96.

42 Rajmohan Rajaraman, David Stalfa, and Sheng Yang. Scheduling under non-uniform job and machine delays, 2022. arXiv:2207.13121.

43 Victor J Rayward-Smith. UET scheduling with unit interprocessor communication delays. *Discrete Applied Mathematics*, 18(1):55–71, 1987.

44 Juan A. Rico-Gallego, Juan C. Díaz-Martín, Ravi Reddy Manumachu, and Alexey L. Lastovetsky. A survey of communication performance models for high-performance computing. *ACM Computing Surveys*, 51(6), January 2019. doi:10.1145/3284358.

45 Petra Schuurman and Gerhard J. Woeginger. Polynomial time approximation algorithms for machine scheduling: ten open problems. *Journal of Scheduling*, 2(5):203–213, 1999.

46 Bastian Seifert, Chris Wendler, and Markus Püschel. Causal fourier analysis on directed acyclic graphs and posets. *CoRR*, abs/2209.07970, 2022. doi:10.48550/arXiv.2209.07970.

47 Ola Svensson. Conditional hardness of precedence constrained scheduling on identical machines. *Proceedings of the 42nd ACM symposium on Theory of computing - STOC '10*, pages 745–754, 2010. doi:10.1145/1806689.1806791.

48 Bart Veltman, B. J. Lageweg, and Jan Lenstra. Multiprocessor scheduling with communication delays. *Parallel Computing*, 16:173–182, 1990.

49 Laurens Versluis, Erwin Van Eyk, and Alexandru Iosup. An analysis of workflow formalisms for workflows with complex non-functional requirements. In *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*, pages 107–112, 2018.

50 Guangyuan Wu, Fangming Liu, Haowen Tang, Keke Huang, Qixia Zhang, Zhenhua Li, Ben Y. Zhao, and Hai Jin. On the performance of cloud storage applications with global measurement. In *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*, pages 1–10, 2016. doi:10.1109/IWQoS.2016.7590449.