Approximation Algorithms for Scheduling with Resource and Precedence Constraints

Gökalp Demirci
Department of Computer Science, University of Chicago
1100 East 58th Street, Chicago, Illinois 60615, USA
demirci@cs.uchicago.edu

Henry Hoffmann
Department of Computer Science, University of Chicago
1100 East 58th Street, Chicago, Illinois 60615, USA
hankhoffmann@cs.uchicago.edu

David H. K. Kim
Department of Computer Science, University of Chicago
1100 East 58th Street, Chicago, Illinois 60615, USA
hongk@cs.uchicago.edu

Abstract

We study non-preemptive scheduling problems on identical parallel machines and uniformly related machines under both resource constraints and general precedence constraints between jobs. Our first result is an $O(\log n)$-approximation algorithm for the objective of minimizing the makespan on parallel identical machines under resource and general precedence constraints. We then use this result as a subroutine to obtain an $O(\log n)$-approximation algorithm for the more general objective of minimizing the total weighted completion time on parallel identical machines under both constraints. Finally, we present an $O(\log m \log n)$-approximation algorithm for scheduling under these constraints on uniformly related machines. We show that these results can all be generalized to include the case where each job has a release time. This is the first upper bound on the approximability of this class of scheduling problems where both resource and general precedence constraints must be satisfied simultaneously.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms

Keywords and phrases Scheduling, Resource, Precedence, Weighted Completion Time

Digital Object Identifier 10.4230/LIPIcs.STACS.2018.25

Related Version The full version of the paper is available as University of Chicago, Department of Computer Science Technical Report with report number TR-2018-01 at `https://newtraell.cs.uchicago.edu/research/publications/techreports/TR-2018-01`.

Funding Funding for this work comes from a DOE Early Career Research Award.

Acknowledgements We thank the anonymous reviewers for their time and feedback. We also thank Janos Simon for his valuable comments on an earlier version of this work.

1 Introduction

Scheduling under resource constraints and scheduling under precedence constraints are both well studied topics in scheduling theory and approximation algorithms. In the former, each job has a resource requirement and there is a finite amount of resource; when jobs run in parallel their total resource requirement must not exceed the given resource capacity. In the latter, a precedence relationship between jobs is given; if a job j has precedence over another, say j', then j must be completed before j' can start. Both of these problems are central to scheduling theory, and their approximability is well understood. They have $(2+\epsilon)$- and $(2-1/m)$-approximation algorithms, respectively, where m is the number of identical machines. However, the approximability of the natural problem with the combination of these constraints is still wide open. We study this problem, namely scheduling on parallel machines under both resource and precedence constraints, and give the first non-trivial approximation algorithms for several important versions of this problem.

The combination of these two constraints naturally models the computation in emerging high performance computing systems. Power consumption is one of the central design considerations for the next generation of *exascale* supercomputers [16]. These future systems will have to run parallel computations within a 20 MW operating budget, but will be built such that if all processors were used at full capacity the budget would be exceeded and the system would fail catastrophically [10]. Therefore, exascale system software must schedule parallel computations (with precedence constraints) on this hardware while respecting the global power budget (a limited shared resource) and minimizing application runtime [18]. This critical problem for emerging supercomputers is a practical example of a scheduling with simultaneous resource and precedence constraints.

We now define the problem formally. We are given a set $\mathcal{J}$ of n jobs to be scheduled on m parallel machines. The schedule needs to be non-preemptive; i.e. each job, once assigned to a machine at some point in time, must run to completion on the same machine without interruption. Each job $j \in \mathcal{J}$ has a processing time $p_j \in \mathbb{Z}_{\geq 0}$ and a resource requirement $s_j \in \mathbb{Z}_{\geq 0}$. There is a global resource capacity $S \in \mathbb{Z}_{\geq 0}$ which any feasible schedule must respect. That is, the sum of the resource requirements of the jobs running on the m machines at any point in time must be at most S. Moreover, we have precedence constraints given by a partial order $\prec$ on the jobs such that if $j \prec j'$ then j must complete before j' can start. We study both *identical* machines and *uniformly related* machines. The difference between the two is the time it takes to process a job—on identical machines, it takes p_j units of time to complete job $j \in \mathcal{J}$, regardless of the machine it runs on. In the case of uniformly related machines, we are given as input a speed f_i $(0 < f_i \leq 1)$ associated with each machine i for $1 \leq i \leq m$, and it takes p_j/f_i units of time to complete job j on machine i.

We also consider two different objectives. The first is minimizing the makespan, or the final completion time of all jobs. This problem on identical machines is denoted as $P|\text{res1}, \text{prec}|C_{max}$ in the standard scheduling notation introduced in [6]. Here P denotes identical parallel machines, as opposed to uniformly related machines which is denoted by Q. Also, the resource constraint, res1, indicates that we have a single resource, and prec stands for the general precedence constraint. C_{max} indicates that the objective is to minimize the makespan. The other objective we consider is the more general goal of minimizing the total weighted completion time. In this setting there is a set of weights $\{w_j\}_{j \in \mathcal{J}}$ associated with the jobs, and the goal is to minimize $\sum_{j \in \mathcal{J}} w_j C_j$ where C_j is the completion time of job j in the final schedule. Using the same notation, we denote this problem on identical machines as $P|\text{res1}, \text{prec}|\sum_j w_j C_j$. Corresponding problems on uniformly related machines are denoted as $Q|\text{res1}, \text{prec}|C_{max}$ and $Q|\text{res1}, \text{prec}|\sum_j w_j C_j$, respectively. Note that the minimum total weighted completion time objective $(\sum_j w_j C_j)$ is more general than the minimum makespan objective (C_{max}) in the presence of precedence constraints as one could transform a C_{max} objective into a $\sum_j w_j C_j$ objective by simply setting all w_j's to be 0 and adding a "last" job j' with $p_{j'} = 0$, $s_{j'} = 0$, and $w_{j'} = 1$ which depends on every other job.

In addition, we consider these problems with release time constraints. In this case, we have a release time $r_j \in \mathbb{Z}_{\geq 0}$ associated with each job in the input such that j can only be started after time r_j. We add r_j in the middle constraint section in the scheduling notation to denote the problems with the additional release time constraint (e.g. $P|\text{res1}, \text{prec}, r_j|C_{max}$).

We have obtained, for the first time, approximation results for all these scheduling problems with provable upper bounds. Our most general result is an $O(\log m \log n)$-approximation for weighted completion time on uniformly related machines under resource and precedence constraints with release times (Theorem 12), which makes substantial use of our new results for more restricted models as well as new specialized linear programming schemes.

Related Work

One important class of problems is scheduling subject to only precedence constraints (e.g. $P|\text{prec}|C_{max}$, $Q|\text{prec}|\sum_j w_jC_j$). Almost all variants of these problems have been studied extensively. For the case of identical parallel machines and the minimum makespan objective ($P|$ $\text{prec}|C_{max}$), Graham's seminal list scheduling algorithm [7] gives a $(2 - 1/m)$-approximation. We will utilize this algorithm and explain some aspects of its analysis in Section 2.1. An almost matching $(2-\epsilon)$-hardness of approximation result is obtained by Svensson [17] assuming a stronger version of the Unique Game Conjecture. For the more general weighted completion time objective ($P|\text{prec}|\sum_j w_jC_j$), Hall et al. [8] gave a 7-approximation which was later improved to a 4-approximation by Munier, Queyranne and Schulz [12]. Very recently, Li [11] obtained the current best ratio of $(2 + 2\ln 2 + \epsilon)$. Of course, all the hardness results for minimizing makespan hold for minimizing the weighted total completion time; however, no stronger hardness results are known for this apparently harder problem. For related machines running at different speeds—$Q|\text{prec}|\sum_j w_jC_j$ and $Q|\text{prec}|C_{max}$—Chudak and Shmoys [3] gave an $O(\log m)$-approximation by grouping the machines into $\log m$ groups according to their speed and treating the machines in each group as identical parallel machines. Li [11] improved this ratio to $O(\log m/\log\log m)$. On the negative side, Bazzi and Norouzi-Fard [2] recently showed the problem is hard to approximate for any constant assuming the hardness of an optimization problem on k-partite graphs.

Another overlapping set of problems is resource-constrained scheduling with one common resource (e.g., $P|\text{res1}|C_{max}$ and $Q|\text{res1}|\sum_j w_jC_j$). Garey and Graham's result in [5] implies a $(3-3/m)$-approximation for $P|\text{res1}|C_{max}$. Later, Niemeier and Wiese [13] gave the algorithm with the current best approximation ratio of $(2 + \epsilon)$. Recently, Jansen et al. [9] provided an AFPTAS for the problem. Since bin packing is a special case of this problem, a simple reduction from the partition problem shows that no approximation with a ratio smaller than 3/2 is possible unless $P = NP$. Note that, in all the problems we consider here, preemption is not allowed and the processing time of a job does not depend on the resource allocated to the job.

There is a clear connection between resource constrained scheduling and packing problems, such as bin packing and strip packing. In the case of unit processing times ($p_j = 1$), for example, $P|\text{res1}, p_j = 1|C_{max}$ is the same as the bin packing problem where we pack different sized items into bins of fixed size and try to minimize the number of bins used. In the strip packing problem, we have a fixed width strip with one open end and a finite set of rectangles. The goal is to fit all the rectangles in the strip minimizing the total height that the rectangles reach. The correspondence between strip packing and resource constrained scheduling is obvious. The width of the strip corresponds to the global resource limit and the widths and the heights of the rectangles correspond to the resource requirement and the processing time of the jobs respectively. In fact, we make use of this correspondence between the two problems in our algorithms. One difference between these two problems is that we can pack as many small width rectangles as the width of the strip allows in the strip packing problem whereas the number of jobs that can run in parallel is bounded by the number of machines in the resource constrained scheduling problem.

Our Results and Techniques

We study identical parallel machines in Section 2. First, in Section 2.1, we consider the objective of minimizing the makespan under both resource and precedence constraints, namely the problem $P|\mathrm{res1}, \mathrm{prec}|C_{max}$. We prove the following.

▶ **Theorem 1.** *There is a $2 + 2\log(n + 1)$-approximation algorithm for $P|res1, prec|C_{max}$.*

To show this, we consider the two constraints separately and present a two-step algorithm for minimizing the makespan. The first step produces an intermediate approximate schedule satisfying only the precedence constraints with a loss of factor of 2 in the approximation. The second step uses a divide and conquer algorithm (inspired by [1]) to stretch the intermediate schedule so that it also satisfies the resource constraint. We generalize this result for the version of the problem with release times.

▶ **Theorem 7.** *There is a $2 + 4\log(n + 1)$-approximation algorithm for $P|res1, prec, r_j|C_{max}$.*

In Section 2.2, we use the algorithm from Section 2.1 as a subroutine to obtain our first main result:

▶ **Theorem 8.** *There is an $O(\log n)$-approximation algorithm for $P|res1, prec, r_j|\sum_j w_jC_j$.*

To obtain this result, we extend the general framework of [8], [15], and [3] with a novel Linear Programming (LP) relaxation for $P|\mathrm{res1}, \mathrm{prec}, r_j|\sum_j w_jC_j$. In this framework, we divide the time horizon into geometrically increasing intervals. Using our new linear program, we obtain the *approximate completion interval* for each job. Then we show that if we consider, for each interval, the set of jobs completing in the same interval as a separate instance of $P|\mathrm{res1}, \mathrm{prec}, r_j|C_{max}$ problem and schedule them in a separate fragment using the makespan minimizing algorithm we obtain in Section 2.1, the concatenation of these fragments also gives a good approximation to the minimum total weighted completion time objective ($\sum_j w_jC_j$) for the original instance. Our core technique is our time-interval-indexed LP where we carefully incorporate the resource requirements into the LP to guarantee a reasonable total resource requirement by the set of jobs completing in any given interval. This allows us to bound the makespan given by our algorithm from Section 2.1 for each $P|\mathrm{res1}, \mathrm{prec}, r_j|C_{max}$ instance.

Finally, in Section 3, we build on our previous techniques to show that they can be modified to work together with the methods in [3] to get an $O(\log m \log n)$-approximation for scheduling on uniformly related machines subject to simultaneous resource, precedence, and release times constraints with weighted completion time objective. In particular, we state a simple generalization of the time-interval-indexed LP used in the previous section without the resource LP constraints) to machines with different speeds. We use the LP solution to get the approximate completion time intervals as we did for identical machines setting as well as job to machine assignments. Then, we argue that the first step of our two-step makespan minimizing algorithm can be replaced by the $O(\log m)$-approximation algorithm for $Q|\mathrm{prec}, r_j|C_{max}$ given in [3]. Moreover, the second step of our algorithm will respect these job to machine assignments. If we begin with an optimal solution to the LP, we show that these integral job to machine assignments are close enough to the fractional assignments given by the LP solution.

▶ **Theorem 12.** *There is an $O(\log m \log n)$-approximation algorithm for $Q|res1,\ prec,\ r_j|\sum_j w_jC_j$.*

These are the first algorithms with non-trivial approximation ratios for this class of scheduling problems where a resource constraint and general precedence constraints need to be satisfied together.

2 Identical Parallel Machines

2.1 The Minimum Makespan Objective

In this section, we present an $O(\log n)$-approximation algorithm for the problem of minimizing the makespan under both resource and precedence constraints.

▶ **Theorem 1.** *There is a $2 + 2\log(n+1)$-approximation algorithm for $P|res1, prec|C_{max}$.*

Given an instance $(\mathcal{J}, m, \{p_j\}_{j\in\mathcal{J}}, S, \{s_j\}_{j\in\mathcal{J}}, \prec)$ of $P|\text{res1}, \text{prec}|C_{max}$, we let OPT denote the makespan of a minimum makespan schedule of this instance. Our algorithm solves the problem in two steps by handling its precedence and resource constraints separately. First, we consider the corresponding $P|\text{prec}|C_{max}$ instance where we drop the resource requirement (i.e. $(\mathcal{J}, m, \{p_j\}_{j\in\mathcal{J}}, \prec)$). In his seminal work, Graham [7] presents an online machine-driven list scheduling algorithm for this problem. His algorithm greedily considers the jobs in some arbitrary extension of the partial order $\prec$ to a total order (i.e. a list). As soon as a machine is idle, the algorithm schedules the next available job (i.e. all its predecessors are finished) in the list on that machine. In the analysis, he uses two standard lower bounds for the value of OPT:

▶ **Lemma 2** (Load Bound). $\frac{1}{m}\sum_{j\in\mathcal{J}} p_j \leq OPT$.

▶ **Lemma 3** (Chain Bound). $\max\limits_{\mathcal{C} \text{ is a chain}} \sum_{j\in\mathcal{C}} p_j \leq OPT$.

The Load Bound is implied by the observation that a perfectly balanced schedule with no idle time would have a makespan of $\frac{1}{m}\sum_{j\in\mathcal{J}} p_j$. Also note that the total processing time of any "chain" of precedence constraints such as $j_1 \prec j_2 \prec \cdots \prec j_k$ is a lower bound on the makespan of any schedule. His analysis charges the time intervals where all the machines are busy on the Load Bound and the time intervals where some machines are idle on the Chain Bound. We run Graham's list scheduling algorithm on our instance after we drop the resource constraints and get an approximate intermediate schedule satisfying the precedence constraints. Let $\mathsf{LS}_{\mathcal{J}}$ denote the makespan of this schedule and $\mathsf{LS}(j)$ denote the completion time of a job $j \in \mathcal{J}$ in it. Graham [7] shows this makespan is bounded by the sum of the Load and the Chain Bounds $\mathsf{LS}_{\mathcal{J}} \leq \frac{1}{m}\sum_{j\in\mathcal{J}} p_j + \max\limits_{\mathcal{C} \text{ is a chain}} \sum_{j\in\mathcal{C}} p_j \leq 2 \cdot OPT$.[1]

The schedule we get by running Graham's list scheduling on the corresponding $P|\text{prec}|C_{max}$ instance may violate the resource constraints of the original instance. In the second step of the algorithm, we run a divide and conquer algorithm on this schedule to make it satisfy the resource constraints without disturbing the precedences already satisfied after our first step. We lose a factor of 2 in the approximation due to Graham's algorithm and an $O(\log n)$-factor in the second step. In the rest of this section, we explain and analyze the second step of the algorithm in detail.

[1] In fact, [7] shows a slightly stronger bound of $(2 - 1/m)\,OPT$.

Algorithm 1 $\mathbf{DS}(J, B, E)$ Divide and Schedule.

```
Input: A subset of jobs $J \subseteq \mathcal{J}$ and the beginning $B$ and the end $E$ times of $J$ in the
    schedule of the first step
Output: A makespan $y$ for a feasible schedule of $J$
 1: if $J = \emptyset$ then
 2:     return 0
 3: end if
 4: $J_{bef} \leftarrow \{j \in J \mid \mathsf{LS}(j) < (B+E)/2\}$
 5: $J_{mid} \leftarrow \{j \in J \mid \mathsf{LS}(j) - p_j < (B+E)/2$ and $\mathsf{LS}(j) \geq (B+E)/2\}$
 6: $J_{aft} \leftarrow \{j \in J \mid \mathsf{LS}(j) - p_j \geq (B+E)/2\}$
 7: $y_{bef} \leftarrow \mathbf{DS}(J_{bef}, \min_{j \in J_{bef}} \mathsf{LS}(j) - p_j, \max_{j \in J_{bef}} \mathsf{LS}(j))$
 8: Schedule $J_{bef}$ according to $\mathbf{DS}(J_{bef}, \min_{j \in J_{bef}} \mathsf{LS}(j) - p_j, \max_{j \in J_{bef}} \mathsf{LS}(j))$.
 9: $y_{mid} \leftarrow \mathbf{NFDH}(J_{mid})$
10: Schedule $J_{mid}$ according to $\mathbf{NFDH}(J_{mid})$ starting at $y_{bef}$.
11: $y_{aft} \leftarrow \mathbf{DS}(J_{aft}, \min_{j \in J_{aft}} \mathsf{LS}(j) - p_j, \max_{j \in J_{aft}} \mathsf{LS}(j))$
12: Schedule $J_{aft}$ according to $\mathbf{DS}(J_{aft}, \min_{j \in J_{aft}} \mathsf{LS}(j) - p_j, \max_{j \in J_{aft}} \mathsf{LS}(j))$ starting at
    $y_{bef} + y_{mid}$.
13: return $y_{bef} + y_{mid} + y_{aft}$
```

Note that the total resource required to complete a job $j \in \mathcal{J}$ is its resource requirement integrated over the time it takes to complete the job: $\int_0^{p_j} s_j \, \mathrm{d}t = s_j p_j$. We denote this value by $\mathsf{RB}(j)$. We define $\mathsf{RB}(J)$ for any subset $J \subseteq \mathcal{J}$ of jobs as the sum of the total resource required for jobs in J (i.e. $\mathsf{RB}(J) = \sum_{j \in J} \mathsf{RB}(j)$). Our algorithm uses another lower bound derived by comparing the resource available in a makespan and total resource required by all jobs.

▶ **Lemma 4** (Resource Bound). $\mathsf{RB}(\mathcal{J})/S \leq OPT$.

To incorporate the resource constraint into the schedule obtained in the first step, we run a divide and conquer algorithm similar to the one employed by Augustine et al. [1] for strip packing. This algorithm (Algorithm 1: **DS**) will need to use a subroutine for scheduling at most m jobs $J \subseteq \mathcal{J}$ with resource constraints that have no precedence constraints among them on m machines. We denote this subroutine by Next-Fit Decreasing-Height: $\mathbf{NFDH}(J)$ and it guarantees a makespan that is bounded by $\mathbf{NFDH}(J) \leq 2\mathsf{RB}(J)/S + \max_{j \in J} p_j$. A simple greedy algorithm that schedules the jobs respecting their resource constraints in the order of non-increasing processing time will have this guarantee. Next-Fit Decreasing-Height algorithm for strip packing analyzed in [4] is one such algorithm when applied to scheduling m jobs on m machines with resource constraints and no precedence constraints.

The algorithm is called with the following initial arguments: all jobs $\mathcal{J}$, beginning time $B = 0$, and end time $E = \mathsf{LS}_{\mathcal{J}}$, the makespan of the schedule obtained in the first step. It takes the set of jobs J_{mid} scheduled to cross the mid time point $\mathsf{LS}_{\mathcal{J}}/2$ in the schedule obtained in the first step. It schedules these jobs in a separate time fragment and concatenates it with schedules obtained recursively on the jobs before, J_{bef}, and the jobs after, J_{aft}. We need the following lemma to justify the initial condition required to call $\mathbf{NFDH}(J_{mid})$. Proofs of all the lemmas in the rest of the paper can be found in the full version of the paper.

▶ **Lemma 5.** *The jobs in J_{mid} have no precedence constraints among them and $1 \leq |J_{mid}| \leq m$.*

In any given level of the recursion tree, the total loss across all recursive calls in this level is at most an additive factor of $O(OPT)$. Since the algorithm terminates in at most $\lceil \log n \rceil$ levels, we get an $O(\log n)$ approximation. Lemma 6 proves this formally and gives a bound on our initial call to Divide and Schedule algorithm: $\mathbf{DS}(\mathcal{J}, 0, \mathsf{LS}_{\mathcal{J}}) \leq 2\mathsf{RB}(\mathcal{J})/S + \mathsf{LS}_{\mathcal{J}} \log(|\mathcal{J}| + 1) \leq (2+2\log(n+1))OPT$. This completes the proof of Theorem 1.

▶ **Lemma 6.** $\boldsymbol{DS}(J, B, E) \leq 2\mathsf{RB}(J)/S + (E - B)\log(|J| + 1)$

The first step of our algorithm can be extended to accommodate release times constraints:

▶ **Theorem 7.** *There is a $2 + 4\log(n + 1)$-approximation algorithm for $P|res1, prec, r_j|C_{max}$.*

In addition to resource and precedence constraints, each job $j \in \mathcal{J}$ now has a release time r_j such that j can only be started after time r_j ($P|\text{res1}, \text{prec}, r_j|C_{max}$). Munier, Queyranne and Schulz [12, 14] provide a 4-approximation for $P|\text{prec}, r_j|C_{max}$. We replace Graham's list scheduling in the first step of our algorithm with their 4-approximation algorithm to get a similar bound for the problem with all three constraints. Note that, after the first step, we obtain a schedule that satisfies the release times and precedence constraints. The second step of our algorithm can easily be made to never schedule a job before its starting time in the schedule from the first step (by forcing J_{mid}'s fragment to start at $\max\{(E - B)/2, y_{bef}\}$).

We note that an $o(\log n)$-approximation is not possible using only the Load, Chain, and Resource Bounds. The bottleneck is in the second step where we use divide-and-conquer. The gap example in [1] for strip packing shows one needs stronger lower bounds on OPT than "the area bound" (Resource Bound in our setting) and "the longest chain of rectangles bound" (Chain Bound in our setting) for an $o(\log n)$ approximation. Their example can easily be translated into a scheduling instance with $m = \log n$ machines. Setting $m = \log n$ allows any solution to their example be interpreted as a scheduling solution because the example has at most $\log n$ parallel precedence chains at any point. Also, the Load Bound of the translated instance is at most $\Theta(1)$, where $OPT = \Omega(\log n)$.[2] Thus, for both the makespan and the weighted completion time objectives, one needs stronger lower bounds on OPT than any combination of the three bounds we use for a better approximation ratio.

2.2 The Minimum Weighted Completion Time Objective

In this section, we generalize our result by giving an $O(\log n)$-approximation algorithm for the minimum total weighted completion time objective. In addition to processing time p_j, resource requirement s_j, and release time r_j, we now have a weight w_j associated with each job $j \in \mathcal{J}$ and our goal is to minimize the total weighted completion time $\sum_j w_j C_j$, where C_j is the completion time of j in the final schedule. This problem is denoted as $P|\text{res1}, \text{prec}, r_j| \sum_j w_j C_j$.

▶ **Theorem 8.** *There is an $O(\log n)$-approximation algorithm for $P|res1, prec, r_j| \sum_j w_j C_j$.*

We first reduce the problem of minimizing weighted completion time ($\sum_j w_j C_j$) to a set of smaller problems with the objective of minimizing the makespan (C_{max}). We then use our $O(\log n)$-approximation algorithm from Section 2.1 for the minimum makespan objective as a subroutine on these problems. In the reduction, we use the general framework of Hall et al.

[2] See [1] and their illustrations for a detailed description of the gap example.

[8] and Queyranne and Sviridenko [15] (see also, for example, [3] for an application of this framework in uniformly related machines setup).

We start with a trivial upper bound 2^L on the length of an optimal schedule, where $L = \lceil \log(n \cdot \max_{j \in \mathcal{J}}(r_j + p_j)) \rceil$, and divide the time horizon into a set of geometrically increasing intervals $[1,2], (2,4], (4,8], \cdots, (2^{L-1}, 2^L]$. For the problem with only precedence and release time constraints ($P|\text{prec}, r_j|\sum_j w_j C_j$), Hall et al. argue that for each job j, if we are given the interval in which it completes in the optimal schedule, or the *completion interval* of j", we can get an approximate schedule based on this information [8]. All the jobs completing in $(2^{l-1}, 2^l]$ in the optimal schedule can be scheduled to start and complete in $(2^{l+1}, 2^{l+2}]$ using a makespan minimizing algorithm on this subset of jobs as a subroutine. This results in an 8-approximation for the weighted completion time objective. Since the completion intervals in an optimal schedule are not known, Hall et al. use an LP relaxation to obtain approximate values for the completion intervals [8]. The makespan of the makespan minimizing algorithm they use for $P|\text{prec}, r_j|C_{max}$ depends only on the Load Bound and the Chain Bound, which are both easy to bound in the same LP where they get the completion intervals. However, in our setting, we need stronger guarantees when obtaining the completion intervals with an LP than prior work provides. In particular, we need the jobs completing in a given interval to have a total resource requirement that is comparable to the resource available up to that interval. In this way, we introduce the following linear programming relaxation for the $P|\text{res1}, \text{prec}, r_j|\sum_j w_j C_j$ problem which ensures that we get a good estimate on the completion interval of each job and simultaneously guarantees that the total resource requirement by jobs completing in some interval is not "too much". We let $[a]$ denote the set $\{1, 2, \cdots, a\}$ for a positive integer a.

$$\min \quad \sum_{j \in \mathcal{J}} w_j C_j \quad \text{s.t.} \tag{LP_P}$$

$$\sum_{i=1}^{m} \sum_{t=1}^{L} x_{ijt} = 1, \qquad \forall j \in \mathcal{J}; \tag{2.1}$$

$$p_j \le C_j - r_j, \qquad \forall j \in \mathcal{J}; \tag{2.2}$$

$$p_j \le C_j - C_{j'}, \qquad \forall j' \prec j; \tag{2.3}$$

$$\sum_{t=1}^{L} 2^{t-1} \sum_{i=1}^{m} x_{ijt} \le C_j, \qquad \forall j \in \mathcal{J}; \tag{2.4}$$

$$\sum_{j \in \mathcal{J}} p_j \sum_{t=1}^{l} x_{ijt} \le 2^l, \qquad \forall i \in [m], l \in [L]; \tag{2.5}$$

$$\sum_{i=1}^{m} \sum_{t=1}^{l} x_{ijt} \le \sum_{i=1}^{m} \sum_{t=1}^{l} x_{ij't}, \qquad \forall j' \prec j, l \in [L]; \tag{2.6}$$

$$\sum_{j \in \mathcal{J}} p_j s_j \sum_{i=1}^{m} \sum_{t=1}^{l} x_{ijt} \le 2^l S, \qquad \forall l \in [L]; \tag{2.7}$$

$$x_{ijt} \ge 0, \qquad \forall i \in [m], j \in \mathcal{J}, t \in [L]; \tag{2.8}$$

In LP_P, we have two sets of variables: a set of real valued variables for the completion times of the jobs $\{C_j\}_{j \in \mathcal{J}}$ and a set of decision variables $\{x_{ijt}\}_{i \in [m], j \in \mathcal{J}, t \in [L]}$ that take 0-1 values in an integral solution with $x_{ijt} = 1$ implying that the job j completed on machine i in the time interval $(2^{t-1}, 2^t]$. Constraint 2.1 says a job needs to complete on some machine and in some interval. Constraints 2.2 and 2.3 make sure that the completion times are not infeasible with respect to release times and the precedence constraints. Constraint 2.4 says that the completion time of a job needs to be later than the time point marking the start of the time interval in which the job completes. Constraint 2.5 ensures that the total processing time of the jobs completing on machine i in the first l intervals is at most the time point marking the end of the lth interval. Constraint 2.6 ensures that a job j depending on another job j' must complete in j''s completion interval or later. Finally, Constraint 2.7 is how we guarantee that the total resource requirement of the jobs completing in the first l intervals is at most the total amount of resource available in these intervals.

Let $\{\tilde{x}_{ijt}\}$ and $\{\tilde{C}_j\}$ be a solution to LP_P. We now describe how to obtain the approximate completion intervals from this fractional LP solution and the smaller problems with minimum makespan objective by partitioning the set of jobs with respect to these completion intervals. Let $\ell_1(j)$ be the first interval l where the sum of job j's variables x_{ijt} across all machines exceed $1/2$ (i.e. minimum l s.t. $\sum_{t=1}^{l}\sum_{i=1}^{m}\tilde{x}_{ijt} \geq 1/2$). Also, define $\ell_2(j)$ to be the interval containing job j's completion time (i.e. minimum l s.t. $C_j \leq 2^l$). We let $\ell(j) = \max\{\ell_1(j), \ell_2(j)\}$. Define, for each $l \in [L]$, $J_l = \{j \in \mathcal{J} : \ell(j) = l\} \subseteq \mathcal{J}$ to be the subset of jobs that "complete" in the lth interval (jobs with $\ell(j) = l$). Note that the sets $J_1, J_2, \cdots, J_L$ are disjoint and they partition the set of jobs $\mathcal{J}$.

Next, we consider each J_l as a separate instance of $P|\text{res1}, \text{prec}|C_{max}$ problem and run our makespan minimizing algorithm from Section 2.1 on the smaller instance $(J_l, m, \{p_j\}_j, S, \{s_j\}_j, \prec_{J_l})$ where $\prec_{J_l}\subseteq\prec$ is the subset of the precedence constraints that are between the jobs in J_l.

▶ **Lemma 9.** *Algorithm 1 in Section 2.1 on the instance* $(J_l, m, \{p_j\}_{j\in J_l}, S, \{s_j\}_{j\in J_l}, \prec_{J_l})$ *returns a feasible schedule of length* $(4 + 3\log(|J_l| + 1))2^l$.

Using Lemma 9, we schedule, for each $l \in [L]$, the jobs in J_l to start after $(4 + 3\log(n + 1))(1 + 2 + 4 \cdots + 2^{l-1})$ and complete before $(4 + 3\log(n + 1))(1 + 2 + 4 \cdots + 2^l)$.

▶ **Lemma 10.** *The resulting schedule satisfies resource, precedence, and release time constraints.*

The next lemma completes the proof of Theorem 8.

▶ **Lemma 11.** *The resulting schedule has weighted completion time within* $32 + 24\log(n + 1)$ *factor of the* LP_P *value* $\sum_{j\in\mathcal{J}} w_j\tilde{C}_j$.

3 Uniformly Related Machines

In this section, we describe our $O(\log n \log m)$-approximation algorithm for the problem of scheduling jobs on uniformly related machines (i.e. machines running at different speeds) under resource, precedence, and release time constraints. Again, the objective is to minimize the more general total weighted completion time. This problem is denoted as $Q|\text{res1}, \text{prec}, r_j|\sum_j w_jC_j$.

▶ **Theorem 12.** *There is an* $O(\log m \log n)$*-approximation algorithm for* $Q|$*res1, prec,* $r_j|\sum_j w_jC_j$.

Now, we have a speed f_i associated with each machine $i \in [m]$ in the input and it takes p_j/f_i amount of time to complete job $j \in \mathcal{J}$ on machine $i \in [m]$. We note that the Load, Chain, and Resource Bounds do not only depend on the set of jobs anymore, but also on the job to machine assignments in a solution (in particular, an optimal solution). This is because the processing time p_j/f_i of a job j now depends on the machine i to which it is assigned. Note that this processing time does not change between machines running at the same speed. Thus we will be more interested in the speed that a job is assigned to than the particular machine. We let K be the number of distinct speeds and $\mathrm{f}_1, \mathrm{f}_2, \cdots, \mathrm{f}_K$ be all the distinct speeds among all m machines. We also let m_k be the number of machines with speed f_k for each $k \in [K]$. We start by solving a time-interval indexed linear program similar to the one we used in Section 2.2 but incorporates the different machine speeds in the LP.

$$\min \quad \sum_{j\in\mathcal{J}} w_j C_j \quad \text{s.t.} \tag{LP_Q}$$

$$\sum_{k=1}^{K}\sum_{t=1}^{L} x_{kjt} = 1, \qquad \forall j \in \mathcal{J}; \tag{3.1}$$

$$\sum_{k=1}^{K}\frac{p_j}{\mathrm{f}_k}\sum_{t=1}^{L} x_{kjt} \le C_j - r_j, \qquad \forall j \in \mathcal{J}; \tag{3.2}$$

$$\sum_{k=1}^{K}\frac{p_j}{\mathrm{f}_k}\sum_{t=1}^{L} x_{kjt} \le C_j - C_{j'}, \qquad \forall j' \prec j; \tag{3.3}$$

$$\sum_{t=1}^{L} 2^{t-1}\sum_{k=1}^{K} x_{kjt} \le C_j, \qquad \forall j \in \mathcal{J}; \tag{3.4}$$

$$\frac{1}{\mathrm{f}_k m_k}\sum_{j\in\mathcal{J}} p_j \sum_{t=1}^{l} x_{kjt} \le 2^l, \qquad \forall k \in [K], l \in [L]; \tag{3.5}$$

$$\sum_{k=1}^{K}\sum_{t=1}^{l} x_{kjt} \le \sum_{k=1}^{K}\sum_{t=1}^{l} x_{kj't}, \qquad \forall j' \prec j, l \in [L]; \tag{3.6}$$

$$\sum_{j\in\mathcal{J}}\sum_{k=1}^{K}\frac{p_j}{\mathrm{f}_k} s_j \sum_{t=1}^{l} x_{kjt} \le 2^l S, \qquad \forall l \in [L]; \tag{3.7}$$

$$x_{kjt} \ge 0, \qquad \forall k \in [K], j \in \mathcal{J}, t \in [L]; \tag{3.8}$$

Constraints and variables of LP_Q are similar to the ones in LP_P of Section 2.2. We still have the real-valued completion time variables $\{C_j\}_{j\in\mathcal{J}}$. However, we are now interested in the distinct speed group a job is assigned to rather than the particular machine. In this way, we modify the decision variables as $\{x_{kjt}\}_{k\in[K],j\in\mathcal{J},t\in[L]}$ where the first index k now indicates the speed group among the machines. In an integral solution, $x_{kjt} = 1$ would stand for the job j completing in the time interval $(2^{t-1}, 2^t]$ on some machine with speed f_k. Since the processing time p_j of a job j now depends on the speed it runs on, we replace p_j in the constraints by $\sum_{k=1}^{K} p_j/\mathrm{f}_k \sum_{t=1}^{L} x_{kjt}$ which essentially is the average processing time of j with respect to fractional speed assignments of j. We make the required change to Constraints 2.2, 2.3, 2.5, and 2.7 to obtain corresponding Constraints 3.2, 3.3, 3.5, and 3.7.

Similar to what we did in the identical machine setting, our algorithm will partition the set of jobs according to approximate completion intervals obtained from LP_Q. Then it will obtain a set of smaller $Q|\text{res1}, \text{prec}, r_j|C_{max}$ instances from the original $Q|\text{res1}, \text{prec}, r_j|\sum_j w_j C_j$ instance. We solve each minimum makespan instance again in two steps by considering the $Q|\text{prec}, r_j|C_{max}$ instance in the first step and handling the resources in the second step. The bound on the makespan minimizing algorithm of Section 2.1 is analyzed in terms of

Load, Chain, and Resource Bounds. However, as we have indicated above, we do not have well-defined Load, Chain, and Resource Bounds in the case of machines running at different speeds. In order to use and analyze a similar two-step algorithm, we use the solution to LP_Q to first fix job to machine speed assignments, then show that the Load, Chain, and Resource Bounds under these assignments are comparable to the LP_Q value. We adapt the speed-based list scheduling algorithm of [3] as our first step for solving $Q|\text{prec}, r_j|C_{max}$ and finally we apply the Divide and Schedule procedure of Section 2.1 as our second step.

Let $\{\tilde{x}_{kjt}\}$ and $\{\tilde{C}_j\}$ be a solution to LP_Q. We obtain "completion intervals" ($\ell(j)$'s) for all the jobs and partition $\mathcal{J}$ by defining J_l for $1 \leq l \leq L$ in the same way as we did in Section 2.2. We let $\ell(j) = \max\{\ell_1(j), \ell_2(j)\}$ where $\ell_1(j)$ is defined as the minimum l s.t. $\sum_{t=1}^{l}\sum_{k=1}^{K} \tilde{x}_{kjt} \geq 1/2$ and $\ell_2(j)$ as the minimum l s.t. $\tilde{C}_j \leq 2^l$. We partition $\mathcal{J}$ by letting $J_l = \{j \in \mathcal{J} : \ell(j) = l\}$ for each $l \in [L]$.

Chudak and Shmoys [3] solve a similar LP without Constraint 3.7 to get an $O(\log m)$ approximation for $Q|\text{prec}, r_j|\sum_j w_j C_j$. They follow a similar framework by first giving a makespan minimizing algorithm for $Q|\text{prec}, r_j|C_{max}$ and then using it on J_l's as a subroutine to get the same result for the weighted completion time objective. We replace Graham's list scheduling for $P|\text{prec}|C_{max}$ with Chudak and Shmoys' speed-based list scheduling algorithm [3] for $Q|\text{prec}, r_j|C_{max}$ in our first step. We now briefly discuss their speed-based list scheduling algorithm, its analysis, and how to integrate it into our setup.

Their speed-based list scheduling algorithm uses a list to process the jobs greedily in the order given by the list similar to Graham's list scheduling. In addition, it uses a function f mapping each job j to a speed $f(j) \in \{\mathrm{f}_1, \mathrm{f}_2, \cdots \mathrm{f}_K\}$. As soon as a job finishes processing, all idle machines are considered in some fixed order one by one. The algorithm considers each machine with speed f_k and schedules the first available job j in the list with $f(j) = \mathrm{f}_k$ on this machine, where a job is available if all its predecessors are completed. They analyze this algorithm by considering the Load Bound of each speed group separately. They use an argument similar to the analysis of Graham's list scheduling by charging the busy time intervals on the Load Bounds and idle intervals on the Chain Bound. They show that the length of the makespan returned by this algorithm is at most the sum of the Load Bounds for each speed group and the Chain Bound:

$$\sum_{k=1}^{K} \frac{1}{m_k} \sum_{j: f(j) = \mathrm{f}_k} \frac{p_j}{f(j)} + \max_{\mathcal{C} \text{ is a chain}} \sum_{j \in \mathcal{C}} \frac{p_j}{f(j)}$$

Note that the bound given above depends heavily on job to speed assignments f used by the algorithm. Given the assignments of an optimal solution, the Load Bound of each speed group and the Chain bound will be bounded by the optimal makespan length. This would put the makespan of the algorithm within a factor $K + 1$ of the optimal solution length. Since we do not have the optimal assignments, we now describe how to get approximate job to speed assignments f by applying standard filtering techniques on the fractional solution $\{\tilde{x}_{kjt}\}$ to LP_Q. Consider a partition J_l and a job $j \in J_l$. Let $\alpha_j = \sum_{i=1}^{K}\sum_{t=1}^{l} \tilde{x}_{kjt}$ and let $\overline{x}_{kj} = \sum_{t=1}^{l} \tilde{x}_{kjt}/\alpha_j$. Note $\alpha_j \geq 1/2$ by definition of $\ell(j) = l$. Let the average processing time of a job j in LP_Q solution be $p_j^{avg} = \sum_{k=1}^{K}(p_j/\mathrm{f}_k)\overline{x}_{kj}$. We define $f(j)$ to be the speed of the maximum capacity speed group among all speeds more than half the average speed that j is assigned to in the fractional LP_Q solution.[3] Formally, let

[3] Equivalently, since p_j is a constant in the evaluation of $f(j)$, we can define it to be the f_k that maximizes $\mathrm{f}_k m_k$.

$$f(j) = \underset{\mathrm{f}_k:\ p_j/\mathrm{f}_k \le 2p_j^{avg}}{\arg\min} \frac{p_j}{\mathrm{f}_k m_k}.$$

Similar to [3], we show that the sum of the Load Bounds of all speed groups under these job to speed assignments given by f is bounded by $4K \cdot 2^l$:

▶ **Lemma 13.**

$$\sum_{k=1}^{K} \frac{1}{m_k} \sum_{j \in J_l:\ f(j)=\mathrm{f}_k} \frac{p_j}{\mathrm{f}_k} \le 4K \cdot 2^l.$$

Now we show that the total processing time of any chain in J_l under the assignments given by f is also bounded:

▶ **Lemma 14.** *For any chain $\mathcal{C}$ of precedence constraints from $\prec_{J_l}$, we have $\sum_{j \in \mathcal{C}} p_j/f(j) \le 4 \cdot 2^l$.*

Lemmas 13 and 14 show that the speed-based list scheduling algorithm we employ in the first step returns a schedule of length $O(K) \cdot 2^l$ on the smaller instance we get from the partition J_l.

After getting the intermediate schedule from the first step of our algorithm, we run our second step (Divide and Schedule) respecting job to machine assignments of the intermediate schedule. Note that J_{mid} in **DS** has only one job per machine because jobs in J_{mid} all run in parallel at some point in the intermediate schedule. Thus, we do not need any modification to the **NFDH** subroutine to ensure the same job to machine assignments in the final schedule.

Next, we show that Resource Bound of the sub instance obtained from the partition J_l is bounded by $4S \cdot 2^l$ under the same job to speed assignments f.

▶ **Lemma 15.** $\mathsf{RB}(J_l) \le 4S \cdot 2^l$.

Our second step returns a schedule of length bounded by $\mathbf{DS}(J, B, E) \le 2\mathsf{RB}(J)/S + (E - B)\log(|J| + 1)$ where $(E - B)$ is the length of the intermediate schedule obtained by the first step. Lemmas 13, 14, and 15 prove that, for any $l \in [L]$, this bound is at most $O(K \log n) \cdot 2^l$ on the sub-instance obtained from the jobs in J_l.

As we did in Section 2.2, we schedule the jobs in J_l for each $l \in [L]$ to start after $O(K \log n)(1 + 2 + 4 \cdots + 2^l)$ and complete before $O(K \log n)(1 + 2 + 4 \cdots + 2^l + 2^{l+1})$. This again gives us a feasible schedule as in Lemma 10 and, given that $2^{l-2} \le \tilde{C}_j$ as in the proof of Lemma 11, we have an $O(K \log n)$ approximation.

Finally using the preprocessing of Chudak and Shmoys [3], we can assume that we only have $K = \log m$ distinct speed groups. This comes with a loss of an additional constant factor on the approximation ratio of our algorithm for the minimum makespan objective. We first discard all the machines with speed less than $1/m$ times the speed of the fastest machine and then round down each remaining speed to the nearest power of $1/2$. Discarding slow machines only increases the optimal makespan by a factor of 2 because we discard no more than m machines each of which is at most as fast as $1/m$ times the speed of the fastest machine. This means the fastest machine can process the jobs of the discarded machines with a loss of factor 2 in the length of the schedule. Similarly, we only lose another factor of 2 by rounding down the speeds of the remaining machines. Since we have at most $\log m$ distinct speeds after the preprocessing, the algorithm described above on each $Q|\mathrm{res1}, \mathrm{prec}, r_j|C_{max}$ sub-instance is an $O(\log m \log n)$-approximation with $K = \log m$.

References

1 John Augustine, Sudarshan Banerjee, and Sandy Irani. Strip packing with precedence constraints and strip packing with release times. In *Proceedings of the Eighteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '06, pages 180–189, New York, NY, USA, 2006. ACM.

2 Abbas Bazzi and Ashkan Norouzi-Fard. Towards tight lower bounds for scheduling problems. In *Proceedings of the 23rd Annual European Symposium on Algorithms, ESA'15*, volume 9294, page 118. Springer, 2015.

3 Fabián A. Chudak and David B. Shmoys. Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '97, pages 581–590, Philadelphia, PA, USA, 1997. Society for Industrial and Applied Mathematics.

4 Jr. E. G. Coffman, M. R. Garey, D. S. Johnson, and R. E. Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9(4):808–826, 1980.

5 M. R. Garey and R. L. Grahams. Bounds for multiprocessor scheduling with resource constraints. *SIAM Journal on Computing*, 4:187–200, 1975.

6 R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G.Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326, 1979. Discrete Optimization II.

7 Ronald L Graham. Bounds for certain multiprocessing anomalies. *Bell Labs Technical Journal*, 45(9):1563–1581, 1966.

8 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, 1997.

9 Klaus Jansen, Marten Maack, and Malin Rau. Approximation schemes for machine scheduling with resource (in-)dependent processing times. In *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 1526–1542, Philadelphia, PA, USA, 2016. Society for Industrial and Applied Mathematics.

10 Peter Kogge, Keren Bergman, Shekhar Borkar, Dan Campbell, William Carlson, William Dally, Monty Denneau, Paul Franzon, William Harrod, Jon Hiller, Sherman Karp, Stephen Keckler, Dean Klein, Robert Lucas, Mark Richards, Al Scarpelli, Steven Scott, Allan Snavely, Thomas Sterling, R. Stanley Williams, Katherine Yelick, Keren Bergman, Shekhar Borkar, Dan Campbell, William Carlson, William Dally, Monty Denneau, Paul Franzon, William Harrod, Jon Hiller, Stephen Keckler, Dean Klein, Peter Kogge, R. Stanley Williams, and Katherine Yelick. Exascale computing study: Technology challenges in achieving exascale systems, 2008.

11 Shi Li. Scheduling to minimize total weighted completion time via time-indexed linear programming relaxations. In *IEEE 57th Annual Symposium on Foundations of Computer Science*, FOCS '17, 2017.

12 Alix Munier, Maurice Queyranne, and Andreas Schulz. Approximation bounds for a general class of precedence constrained parallel machine scheduling problems. In *Integer Programming and Combinatorial Optimization*, IPCO '98, 1998.

13 Martin Niemeier and Andreas Wiese. Scheduling with an orthogonal resource constraint. In *10th Workshop on Approximation and Online Algorithms (WAOA2012)*, number EPFL-CONF-181146, 2012.

14 Maurice Queyranne and Andreas S. Schulz. Approximation bounds for a general class of precedence constrained parallel machine scheduling problems. *SIAM J. Comput.*, 35(5):1241–1253, 2006.

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

16 Vivek Sarkar, Saman Amarasinghe, Dan Campbell, William Carlson, Andrew Chien, William Dally, Elmootazbellah Elnohazy, Mary Hall, Robert Harrison, William Harrod, Kerry Hill, Jon Hiller, Sherman Karp, Charles Koelbel, David Koester, Peter Kogge, John Levesque, Daniel Reed, Robert Schreiber, Mark Richards, Al Scarpelli, John Shalf, Allan Snavely, and Thomas Sterling. Exascale software study: Software challenges in extreme scale systems, 2009. DARPA IPTO Study Report for William Harrod.

17 Ola Svensson. Conditional hardness of precedence constrained scheduling on identical machines. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, pages 745–754, New York, NY, USA, 2010. ACM.

18 ExaOSR Team. Exascale operating systems and runtime software report. Online document, `https://science.energy.gov/~/media/ascr/pdf/research/cs/Exascale%20Workshop/ExaOSR-Report-Final.pdf`.