58th Annual IEEE Symposium on Foundations of Computer Science

Scheduling to Minimize Total Weighted Completion Time via Time-Indexed Linear Programming Relaxations

Shi Li
Department of Computer Science and Engineering,
University at Buffalo,
Buffalo, NY, USA
shil@buffalo.edu

***Abstract*—We study approximation algorithms for scheduling problems with the objective of minimizing total weighted completion time, under identical and related machine models with job precedence constraints. We give algorithms that improve upon many previous 15 to 20-year-old state-of-art results. A major theme in these results is the use of time-indexed linear programming relaxations. These are natural relaxations for their respective problems, but surprisingly are not studied in the literature.**

We also consider the scheduling problem of minimizing total weighted completion time on unrelated machines. The recent breakthrough result of [Bansal-Srinivasan-Svensson, STOC 2016] gave a $(1.5-c)$-approximation for the problem, based on some lift-and-project SDP relaxation. Our main result is that a $(1.5-c)$-approximation can also be achieved using a natural and considerably simpler time-indexed LP relaxation for the problem. We hope this relaxation can provide new insights into the problem.

***Keywords*-approximation algorithms, scheduling, time-indexed, weighted completion time**

I. Introduction

Scheduling jobs to minimize total weighted completion time is a well-studied topic in scheduling theory, operations research and approximation algorithms. A systematic study of this objective under many different machine models (e.g, identical, related and unrelated machine models, job shop scheduling, precedence constraints, preemptions) was started in late 1990s and since then it has led to great progress on many fundamental scheduling problems.

In spite of these impressive results, the approximability of many problems is still poorly understood. Many of the state-of-art results that were developed in late 1990s or early 2000s have not been improved since then. Continuing the recent surge of interest on the total weighted completion time objective [1], [2], [3], we give improved approximation algorithms for many scheduling problems under this objective. The machine models we study in this paper include identical machine model with job precedence constraints, with uniform and non-uniform job sizes, related machine model with job precedence constraints and unrelated machine model.

A major theme in our results is the use of time-indexed linear programming relaxations. Given the time aspect of scheduling problems, they are natural relaxations for their respective problems. However, to the best of our knowledge, many of these relaxations were not studied in the literature and thus their power in deriving improved approximation ratios was not well-understood. Compared to other types of relaxations, solutions to these relaxations give fractional scheduling of jobs on machines. Many of our improved results were obtained by using the fractional scheduling to identify the loose analysis in previous results.

A. Definitions of Problems and Our Results

We now formally describe the problems we study in the paper and state our results. In all of these problems, we have a set J of n jobs, a set M of m machines, each job $j \in J$ has a weight $w_j \in \mathbb{Z}_{>0}$, and the objective to minimize is $\sum_{j \in J} w_j C_j$, where C_j is the completion time of the job j. We consider non-preemptive schedules only. So, a job must be processed on a machine without interruption. For simplicity, this global setting will not be repeated when we define problems.

Scheduling on Identical Machines with Job Precedence Constraints: In this problem, each job $j \in J$ has a processing time (or size) $p_j \in \mathbb{Z}_{>0}$. The m machines are identical; each job j must be scheduled on one of the m machines non-preemptively; namely, j must be processed during a time interval of length p_j on some machine. The completion time of j is then the right endpoint of this interval. Each machine at any time can only process at most one job. Moreover, there are precedence constraints given by a partial order "$\prec$", where a constraint $j \prec j'$ requires that job j' can only start after job j is completed. Using the popular three-field notation introduced by Graham et al. [4], this problem is described as $P\big|\mathrm{prec}\big|\sum_j w_j C_j$.

For the related problem $P|\mathrm{prec}|C_{\max}$, i.e, the problem with the same setting but with the makespan objective, the seminal work of Graham [5] gives a 2-approximation algorithm, based on a simple machine-driven list-scheduling algorithm. In the algorithm, the schedule is constructed in real-time. As time goes, each idle machine shall pick any available job to process (a job is available if it is not scheduled but all its predecessors are completed.), if such a job exists; otherwise, it remains idle until some job becomes available. On the negative side, Lenstra and Rinnooy Kan [6] proved a $(4/3 - \epsilon)$-hardness of approximation for $P|\mathrm{prec}|C_{\max}$. Under some stronger version of the Unique Game Conjecture (UGC) introduced by Bansal and Khot [7], Svensson [8] showed that $P|\mathrm{prec}|C_{\max}$ is hard to approximate within a factor of $2 - \epsilon$ for any $\epsilon > 0$.

With precedence constraints, the weighted completion time objective is more general than makespan: one can create a dummy job of size 0 and weight 1 that must be processed after all jobs in J, which have weight 0. Thus, the above negative results carry over to $P\big|\mathrm{prec}\big|\sum_j w_j C_j$. Indeed, Bansal and Khot [7] showed that the problem with even one machine is already hard to approximate within a factor of $2 - \epsilon$, under their stronger version of UGC. However, no better hardness results are known for $P\big|\mathrm{prec}\big|\sum_j w_j C_j$, compared to those for $P|\mathrm{prec}|C_{\max}$.

On the positive side, by combining the list-scheduling algorithm of Graham [5] with a convex programming relaxation for $P\big|\mathrm{prec}\big|\sum_j w_j C_j$, Hall et al. [9] gave a 7-approximation for $P\big|\mathrm{prec}\big|\sum_j w_j C_j$. For the special case $1|\mathrm{prec}|\sum_j w_j C_j$ of the problem where there is only 1 machine, Hall et al. [9] gave a 2-approximation, which matches the $(2 - \epsilon)$-hardness assuming the stronger version of UGC due to [7]. Later, Munier, Queyranne and Schulz ([10], [11]) gave the current best 4-approximation algorithm for $P\big|\mathrm{prec}\big|\sum_j w_j C_j$, using a convex programming relaxation similar to that in Hall et al. [9] and in Charkrabarti et al. [12]. The convex programming gives a completion time vector $(C_j)_{j\in J}$, and the algorithm of [10] runs a *job-driven* list-scheduling algorithm using the order of jobs determined by the values $C_j - p_j/2$. In the algorithm, we schedule jobs j one by one, according to the non-increasing order of $C_j - p_j/2$; at any iteration, we schedule j at an interval $(\widetilde{C}_j - p_j, \widetilde{C}_j]$ with the minimum $\widetilde{C}_j$, subject to the precedence constraints and the m-machine constraint. It has been a long-standing open problem to improve this factor of 4 (see the discussion after Open Problem 9 in [13]).

Munier, Queyranne and Schulz [10] also considered an important special case of the problem, denoted as $P\big|\mathrm{prec}, p_j = 1\big|\sum_j w_j C_j$, in which all jobs have size $p_j = 1$. They showed that the approximation ratio of their algorithm becomes 3 for the special case, which has not been improved since then. On the negative side, the $2-\epsilon$ strong UGC-hardness result of Bansal and Khot also applies to this special case.

In this paper, we improve the long-standing approximation ratios of 4 and 3 for $P\big|\mathrm{prec}\big|\sum_j w_j C_j$ and $P\big|\mathrm{prec}, p_j = 1\big|\sum_j w_j C_j$ due to Munier, Queyranne and Schulz [10], [11]:

Theorem 1. *There is a $2 + 2\ln 2 + \epsilon < (3.387 + \epsilon)$-approximation algorithm for $P\big|\mathrm{prec}\big|\sum_j w_j C_j$, for every $\epsilon > 0$.*

Theorem 2. *There is a $1 + \sqrt{2} < 2.415$-approximation algorithm for $P\big|\mathrm{prec}, p_j = 1\big|\sum_j w_j C_j$.*

Scheduling on Related Machines with Job Precedence Constraints: Then we consider the scheduling problem on related machines. We have all the input parameters in the problem $P|\mathrm{prec}|\sum_j w_j C_j$. Additionally, each machine $i \in M$ is given a speed $s_i > 0$ and the time of processing job j on machine i is p_j/s_i (so the m machines are not identical any more). A job j must be scheduled on some machine i during an interval of length p_j/s_i. Using the three-field notation, the problem is described as $Q|\mathrm{prec}|\sum_j w_j C_j$.

Chudak and Shmoys [14] gave the current best $O(\log m)$ approximation algorithm for the problem, improving upon the previous $O(\sqrt{m})$-approximation due to Jaffe [15]. Using a general framework of Hall et al. [9] and Queyranne and Sviridenko [16], that converts an algorithm for a scheduling problem with makespan objective to an algorithm for the correspondent problem with weighted completion time objective, Chudak and Shmoys reduced the problem $Q|\mathrm{prec}|\sum_j w_j C_j$ to $Q|\mathrm{prec}|C_{\max}$. In their algorithm for $Q|\mathrm{prec}|C_{\max}$, we partition the machines into groups, each containing machines of similar speeds. By solving an LP relaxation, we assign each job to a group of machines. Then we can run a generalization of the Graham's machine-driven list scheduling problem, that respect the job-to-group assignment. The $O(\log m)$-factor comes from the number $O(\log m)$ of machine groups.

On the negative side, all the hardness results for $P|\mathrm{prec}|C_{\max}$ carry over to both $Q|\mathrm{prec}|C_{\max}$ and $Q|\mathrm{prec}|\sum_j w_j C_j$. Recently Bazzi and Norouzi-Fard [17] showed that assuming the hardness of some optimization problem on k-partite graphs, both problems are hard to be approximated within any constant.

In this paper, we give a slightly better approximation ratio than $O(\log m)$ due to Chudak and Shmoys [14], for both $Q|\text{prec}|C_{\max}$ and $Q|\text{prec}|\sum_j w_j C_j$:

Theorem 3. *There are $O(\log m/\log\log m)$-approximation algorithms for both $Q|\text{prec}|C_{\max}$ and $Q|\text{prec}|\sum_j w_j C_j$.*

Scheduling on Unrelated Machines: Finally, we consider the classic scheduling problem to minimize total weighted completion time on unrelated machines (without precedence constraints). In this problem we are given a number $p_{i,j} \in \mathbb{Z}_{>0}$ for every $i \in M, j \in J$, indicating the time needed to process job j on machine i. This problem is denoted as $R||\sum_j w_j C_j$.

For this problem, there are many classic 3/2-approximation algorithms, based on a weak time-indexed LP relaxation [18] and a convex-programming relaxation ([19], [20]). These algorithms are all based on independent rounding. Solving some LP (or convex programming) relaxation gives $y_{i,j}$ values, where each $y_{i,j}$ indicates the fraction of job j that is assigned to machine i. Then the algorithms randomly and independently assign each job j to a machine i, according to the distribution $\{y_{i,j}\}_i$. Under this job-to-machine assignment, the optimum scheduling can be found by applying the Smith rule on individual machines.

Improving the 3/2-approximation ratio had been a long-standing open problem (see Open Problem 8 in [13]). The difficulty of improving the ratio comes from the fact that any independent rounding algorithm can not give a better than a 3/2-approximation for $R||\sum_j w_j C_j$, as shown by Bansal, Srinivasan and Svensson [1]. This lower bound is irrespective of the relaxation used: even if the fractional solution is already a convex combination of optimum integral schedules, independent rounding can only give a 3/2-guarantee. To overcome this barrier, [1] introduced a novel dependence rounding scheme, which guarantees some strong negative correlation between events that jobs are assigned to the same machine i. Combining this with their lifted SDP relaxation for the problem, Bansal, Srinivasan and Svensson gave a $(3/2-c)$-approximation algorithm for the problem $R||\sum_j w_j C_j$, where $c = 1/(108 \times 20000)$. This solves the long-standing open problem in the affirmative.

Besides the slightly improved approximation ratio, our main contribution for this problem is that the $(1.5-c)$-approximation ratio can also be achieved using the following natural time-indexed LP relaxation:

$$\min \quad \sum_j w_j \sum_{i,s} x_{i,j,s}(s+p_{i,j}) \quad \text{s.t.} \quad (\text{LP}_{\text{R}||\text{wC}})$$

$$\sum_{i,s} x_{i,j,s} = 1 \qquad \forall j \tag{1}$$

$$\sum_{j, s\in(t-p_{i,j},t]} x_{i,j,s} \le 1 \qquad \forall i,t \tag{2}$$

$$x_{i,j,s} = 0 \qquad \forall i,j,s > T - p_{i,j} \tag{3}$$

$$x_{i,j,s} \ge 0 \qquad \forall i,j,s \tag{4}$$

In the above LP, T is a trivial upper bound on the makespan of any reasonable schedule ($T = \sum_j \max_{i:p_{i,j}\neq\infty} p_{i,j}$ suffices). i, j, s and t are restricted to elements in $M, J, \{0,1,2,\cdots,T-1\}$ and $[T]$ respectively. $x_{i,j,s}$ indicates whether job j is processed on machine i with starting time s. The objective to minimize is the weighted completion time $\sum_j w_j \sum_{i,s} x_{i,j,s}(s+p_{i,j})$. Constraint (1) requires every job j to be scheduled. Constraint (2) says that on every machine i at any time point t, only one job is being processed. Constraint (3) says that if job j is scheduled on i, then it can not be started after $T - p_{i,j}$. Constraint (4) requires all variables to be nonnegative.

Theorem 4. *The LP relaxation* $(\text{LP}_{\text{R}||\text{wC}})$ *for $R||\sum_j w_j C_j$ has an integrality gap of at most $1.5-c$, where $c = \frac{1}{6000}$. Moreover, there is an algorithm that, given a valid fractional solution x to* $(\text{LP}_{\text{R}||\text{wC}})$*, outputs a random valid schedule with expected cost at most $(1.5-c)\sum_j w_j \sum_{i,s} x_{i,j,s}(s+p_{i,j})$, in time polynomial in the number of non-zero variables of x.*[1]

The above algorithm leads to a $(1.5-c)$-approximation for $R||\sum_j w_j C_j$ immediately if T is polynomially bounded. The case when T is super-polynomial will be handled in the full version of the paper.

B. Our Techniques

A key technique in many of our results is the use of time-indexed LP relaxations. For the identical machine setting, we have variables $x_{j,t}$ indicating whether job j is scheduled in the time-interval $(t-p_j, t]$; we can visualize $x_{j,t}$ as a rectangle of height $x_{j,t}$ with horizontal span $(t-p_j, t]$. With this visualization, it is straightforward to express the objective function, and formulate the machine-capacity constraints and the precedence constraints. For the unrelated machine model, the LP we use is $(\text{LP}_{\text{R}||\text{wC}})$. (We used starting points to index

[1]We assume x is given as a sequence of $(i,j,s,x_{i,j,s})$-tuples with non-zero $x_{i,j,s}$.

intervals, as opposed to ending points; this is only for the simplicity of describing the algorithm.) Each $x_{i,j,s}$ can be viewed as a rectangle of height $x_{i,j,s}$ on machine i with horizontal span $(s, s+p_{i,j}]$. The rectangle structures allow us to recover the previous state-of-art results, and furthermore to derive the improved approximation results by identifying the loose parts in these algorithms and analysis.

$P|\text{prec}|\sum_j w_j C_j$: Let us first consider the scheduling problem on identical machines with job precedence constraints. The 4-approximation algorithm of Munier, Queyranne, and Schulz [10], [11] used a convex programming that only contains the completion time variables $\{C_j\}_{j\in J}$. After obtaining the vector C, we run the job-driven list scheduling algorithm, by considering jobs j in increasing order of $C_j - p_j/2$. To analyze the expected completion time of j^* in the output schedule, focus on the schedule $\mathcal{S}$ constructed by the algorithm at the time j^* was inserted. Then, we consider the total length of busy and idle slots in $\mathcal{S}$ before the completion of j^* separately. The length of busy slots can be bounded by $2C_{j^*}$, using the m-machine constraint. The length of idle slots can also be bounded by $2C_{j^*}$, by identifying a chain of jobs that resulted in the idle slots. More generally, they showed that if jobs are considered in increasing order of $C_j - (1-\theta)p_j$ for $\theta \in [0, 1/2]$ in the list scheduling algorithm, the factor for idle slots can be improved to $1/(1-\theta)$ but the factor for busy slots will be increased to $1/\theta$. Thus, $\theta = 1/2$ gives the best trade-off.

The rectangle structure allows us to exam the tightness of the above factors more closely: though the $1/\theta$ factor for busy slots is tight for every individual $\theta \in [0, 1/2]$, it can not be tight for every such θ. Roughly speaking, the $1/\theta$ factor is tight for a job j^* only when j^* has small p_{j^*}, and all the other jobs j considered before j^* in the list scheduling algorithm has large p_j and $C_j - \theta p_j$ is just smaller than $C_{j^*} - \theta p_{j^*}$. However in this case, if we decrease θ slightly, these jobs j will be considered after j^* and thus the bound can not be tight for all $\theta \in [0, 1/2]$. We show that even if we choose θ uniformly at random from $[0, 1/2]$, the factor for busy time slots remains 2, as opposed to $\int_{\theta=0}^{1/2} \frac{2}{\theta}\mathrm{d}\theta = \infty$. On the other hand, this decreases the factor for idle slots to $\int_{\theta=0}^{1/2} \frac{2}{1-\theta}\mathrm{d}\theta = 2\ln 2$, thus improving the approximation factor to $2 + 2\ln 2$. The idea of choosing a random point for each job j and using them to decide the order in the list-scheduling algorithm has been studied before under the name "α-points" [21], [9], [22], [23]. The novelty of our result is the use of the rectangle structure to relate different θ values. In contrast, solutions to the convex programming of [10] and the weak time-indexed LP relaxation of [18] lack such a structure.

$P|\text{prec}, p_j = 1|\sum_j w_j C_j$: When jobs have uniform length, the approximation ratio of the algorithm of [10] improves to 3. In this case, the θ parameter in the above algorithm becomes useless since all jobs have the same length. Taking the advantage of the uniform job length, the factor for idle time slots improves 1, while the factor for busy slots remains 2. This gives an approximation factor of 3 for the special case.

To improve the factor of 3, we use another randomized procedure to decide the order of jobs in the list scheduling algorithm. For every $\theta \in [0, 1]$, let M_j^θ be the first time when we scheduled θ fraction of job j in the fractional solution. Then we randomly choose $\theta \in [0, 1]$ and consider jobs the increasing order of M_j^θ in the list-scheduling algorithm. This algorithm can recover the factor of 1 for total length of idle slots and 2 for total length of busy slots.

We again use the rectangle structure to discover the loose part in the analysis. With uniform job size, the idle slots before a job j are caused only by the precedence constraints: if the total length of idle slots before the completion time of j is a, then there is a precedence-chain of a jobs ending at j; in other words, j is at depth at least a in the precedence graph. In order for the factor 1 for idle slots to be tight, we need to have $a \approx C_j$. We show that if this happens, the factor for busy time slots shall be much better than 2. Roughly speaking, the factor of 2 for busy time slots is tight only if j is scheduled evenly among $[0, 2C_j]$. However, if j is at depth-a in the dependence graph, it can not be scheduled before time $a \approx C_j$ with any positive fraction. A quantification of this argument allows us to derive the improved approximation ratio $1+\sqrt{2}$ for this special case.

$Q|\text{prec}|\sum_j w_j C_j$: Our $O(\log m/\log\log m)$-approximation for related machine scheduling is a simple one. As mentioned earlier, by losing a constant factor in the approximation ratio, we can convert the problem of minimizing the weighted completion time to that of minimizing the makespan, i.e, the problem $Q|\text{prec}|C_{\max}$. To minimize the makespan, the algorithm of Chudak and Shmoys [14] partitions machines into $O(\log m)$ groups according to their speeds. Based on their LP solution, we assign each job j to a group of machines. Then we run the machine-driven list-scheduling algorithm, subject to the precedence constraint, and the constraint that each job can only be scheduled to a machine in its assigned group. The final approximation ratio is the sum of two

factors: one from grouping machines with different speeds into the same group, which is $O(1)$ in [14], and the other from the number of different groups, which is $O(\log m)$ in [14]. To improve the ratio, we make the speed difference between machines in the same group as large as $\Theta(\log m/\log\log m)$, so that we only have $O(\log m/\log\log m)$ groups. Then, both factors become $O(\log m/\log\log m)$, leading to an $O(\log m/\log\log m)$-approximation for the problem. One remark is that in the algorithm of [14], the machines in the same group can be assumed to have the same speed, since their original speeds only differ by a factor of 2. In our algorithm, we have to keep the original speeds of machines, to avoid a multiplication of the two factors in the approximation ratio.

$R||\sum_j w_jC_j$: Then we sketch how we use our time-indexed LP to recover the $(1.5-c)$-approximation of [1] (with much better constant c), for the scheduling problem on unrelated machines to minimize total weighted completion time, namely $R||\sum_j w_jC_j$.

The dependence rounding procedure of [1] is the key component leading to a better than 1.5 approximation for $R||\sum_j w_jC_j$. It takes as input a grouping scheme: for each machine i, the jobs are partitioned into groups with total fractional assignment on i being at most 1. The jobs in the same group for i will have strong negative correlation towards being assigned to i. To apply the theorem, they first solve the lift-and-project SDP relaxation for the problem, and construct a grouping scheme based on the optimum solution to the SDP relaxation. For each machine i, the grouping algorithm will put jobs with similar Smith-ratios in the same group, as the 1.5-approximation ratio is caused by conflicts between these jobs. With the strong negative correlation, the approximation ratio can be improved to $(1.5-c)$ for a tiny constant $c = 1/(108\times 20000)$.

We show that the natural time-indexed relaxation $(\mathrm{LP}_{\mathrm{R}||\mathrm{wC}})$ for the problem suffices to give a $(1.5-c)$-approximation. To apply the dependence rounding procedure, we need to construct a grouping for every machine i. In our recovered 1.5-approximation algorithm for the problem using $(\mathrm{LP}_{\mathrm{R}||\mathrm{wC}})$, the expected completion time of j is at most $\sum_{i,s} x_{i,j,s}(s+1.5p_{i,j})$, i.e, the average starting time of j plus 1.5 times the average length of j in the LP solution. This suggests that a job j is bad only when its average starting time is very small compared to its average length in the LP solution. Thus, for each machine i, the bad jobs are those with a large weight of scheduling intervals near the beginning of the time horizon. If these bad intervals for two bad jobs j and j' have large overlap, then they are likely to be put into the same group for i. To achieve this, we construct a set of disjoint basic blocks $\{(2^a, 2^{a+1}] : a \geq -2\}$ in the time horizon. A bad job will be assigned to a random basic block contained in its scheduling interval and two bad jobs assigned to the same basic block will likely to be grouped together. Besides the improved approximation ratio, we believe the use of $(\mathrm{LP}_{\mathrm{R}||\mathrm{wC}})$ will shed light on getting an approximation ratio for the problem that is considerably better than 1.5, as it is simpler than the lift-and-project SDP of [1]. Another useful property of our algorithm is that the rounding procedure is oblivious to the weights of the jobs; this may be useful when we consider some variants of the problem.

Finally, we remark that Theorems 1, 2 and 3 can be easily extended to handle job arrival times. However, to deliver the key ideas more efficiently, we chose not to consider arrival times.

C. Other Related Work

There is a vast literature on approximating algorithms for scheduling problems to minimize the total weighted completion time. Here we only discuss the ones that are most relevant to our results; we refer readers to [24] for a more comprehensive overview. When there are no precedence constraints, the problems of minimizing total weighted completion time on identical and related machines ($P||\sum_j w_jC_j$ and $Q||\sum_j w_jC_j$) admit PTASes ([25], [26]). For the problem of scheduling jobs on unrelated machines with job arrival times to minimize weighted completion time ($R|r_j|\sum_j w_jC_j$), many classic results give 2-approximation algorithms ([19], [18], [27]); recently Im and Li [2] gave a 1.8687-approximation for the problem, solving a long-standing open problem. Skutella [3] gave a $\sqrt{e}/(\sqrt{e}-1) \approx 2.542$-approximation algorithm for the single-machine scheduling problem with precedence constraints and job release times, improving upon the previous $e \approx 2.718$-approximation [28].

Makespan is an objective closely related to weighted completion time. As we mentioned, for $P|\mathrm{prec}|C_{\max}$, the Graham's list scheduling algorithm gives a 2-approximation, which is the best possible under a stronger version of UGC [7], [8]. For the special case of the problem $Pm|\mathrm{prec}, p_j = 1|C_{\max}$ where there are constant number of machines and all jobs have unit size, the recent breakthrough result of Levey and Rothvoss [29] gave a $(1+\epsilon)$-approximation with running time $\exp\left(\exp\left(O_{m,\epsilon}(\log^2\log n)\right)\right)$, via the LP hierarchy of the natural LP relaxation for the problem. On the negative side, it is not even known whether $Pm|\mathrm{prec}, p_j =$

$1|C_{\max}$ is NP-hard or not. For the problem $R||C_{\max}$, i.e, the scheduling of jobs on unrelated machines to minimize the makespan, the classic result of Lenstra, Shmoys and Tardos [30] gives a 2-approximation, which remains the best algorithm for the problem. Some efforts have been put on a special case of the problem, where each job j has a size p_j and $p_{i,j} \in \{p_j, \infty\}$ for every $i \in M$ (the model is called restricted assignment model.) [31], [32], [33], [34].

Organization: The proofs of Theorems 1 to 3 are given in Sections II to IV respectively. Due to the space limit, the proof of Theorem 4 is deferred to the full version of the paper. Throughout this paper, we assume the weights, lengths of jobs are integers. Let T be the maximum makespan of any "reasonable" schedule. For problems $P|\text{prec}|\sum_j w_j C_j$ and $R||\sum_j w_j C_j$, we assume T is polynomial in n. By losing a $1+\epsilon$ factor in the approximation ratio, we can handle the case where T is super-polynomial. This and the other omitted proofs can be found in the full version of the paper.

II. Scheduling on Identical Machines with Job Precedence Constraints

In this section we give our $(2 + 2\ln 2 + \epsilon)$-approximation for the problem of scheduling precedence-constrained jobs on identical machines, namely $P|\text{prec}|\sum_j w_j C_j$. We solve ($\text{LP}_{\text{P|prec|wC}}$) and run the job-driven list-scheduling algorithm of [10] with a random order of jobs.

A. Time-Indexed LP Relaxation for $P|\text{prec}|\sum_j w_j C_j$

In the identical machine setting, we do not need to specify which machine each job is assigned to; it suffices to specify a scheduling interval $(t - p_j, t]$ for every job j. A folklore result says that a set of intervals can be scheduled on m machines if and only if their *congestion* is at most m: i.e, the number of intervals covering any time point is at most m. Given such a set of intervals, there is a simple greedy algorithm to produce the assignment of intervals to machines. Thus, in our LP relaxation and in the list-scheduling algorithm, we focus on finding a set of intervals with congestion at most m.

We use ($\text{LP}_{\text{P|prec|wC}}$) for both $P|\text{prec}|\sum_j w_j C_j$ and $P|\text{prec}, p_j = 1|\sum_j w_j C_j$. Let $T = \sum_j p_j$ be a trivial upper bound on the makespan of any reasonable schedule. In the LP relaxation, we have a variable $x_{j,t}$ indicating whether job j is scheduled in $(t - p_j, t]$, for every $j \in J$ and $t \in [T]$. Throughout this and the next section, t and t' are restricted to be integers in $[T]$, and j, j' and j^* are restricted to be jobs in J.

$$\min \quad \sum_j w_j \sum_t x_{j,t} t \quad \text{s.t.} \qquad (\text{LP}_{\text{P|prec|wC}})$$

$$\sum_t x_{j,t} = 1 \qquad \forall j \qquad (5)$$

$$\sum_{j, t \in [t', t'+p_j)} x_{j,t} \le m \qquad \forall t' \qquad (6)$$

$$\sum_{t < t' + p_{j'}} x_{j',t} \le \sum_{t < t'} x_{j,t} \qquad \forall j, j', t' : j \prec j' \qquad (7)$$

$$x_{j,t} = 0 \qquad \forall j, t < p_j \qquad (8)$$

$$x_{j,t} \ge 0 \qquad \forall j, t \qquad (9)$$

The objective function is $\sum_j w_j \sum_t x_{j,t} t$, i.e, the total weighted completion time over all jobs. Constraint (5) requires every job j to be scheduled. Constraint (6) requires that at every time point t', at most m jobs are being processed. Constraint (7) requires that for every $j \prec j'$ and t', j' completes before $t' + p_{j'}$ only if j completes before time t'. A job j can not complete before p_j (Constraint (8)) and all variables are non-negative (Constraint (9)).

We solve ($\text{LP}_{\text{P|prec|wC}}$) to obtain $x \in [0,1]^{J \times [T]}$. Let $C_j = \sum_t x_{j,t} t$ be the completion time of j in the LP solution. Thus, the value of the LP is $\sum_j w_j C_j$. For every $\theta \in [0, 1/2]$, we define $M_j^\theta = C_j - (1-\theta) p_j$. Our algorithm is simply the following: choose θ uniformly at random from $(0, 1/2]$, and output the schedule returned by job-driven-list-scheduling(M^θ) (described in Algorithm 1).

Algorithm 1 job-driven-list-scheduling(M)

Input: a vector $M \in \mathbb{R}^J_{\ge 0}$ used to decide the order of scheduling, s.t. if $j \prec j'$, then $M_j < M_{j'}$

Output: starting and completion time vectors $\widetilde{S}, \widetilde{C} \in \mathbb{R}^J_{\ge 0}$

```
1: for every $j \in J$ in non-decreasing order of $M_j$, breaking ties arbitrarily
2:     let $t \leftarrow \max_{j' \prec j} \widetilde{C}_{j'}$, or $t \leftarrow 0$ if $\{j' \prec j\} = \emptyset$
3:     find the minimum $s \ge t$ such that we can schedule $j$ in interval $(s, s+p_j]$, without increasing the congestion of the schedule to $m+1$
4:     $\widetilde{S}_j \leftarrow s, \widetilde{C}_j \leftarrow s+p_j$, and schedule $j$ in $(\widetilde{S}_j, \widetilde{C}_j]$
5: return $(\widetilde{S}, \widetilde{C})$
```

We first make a simple observation regarding the C vector, which follows from the constraints in the LP.

Claim 5. *For every pair of jobs j, j' such that $j \prec j'$, we have $C_j + p_{j'} \le C_{j'}$.*

Indeed, our analysis does not use the full power of Constraint (7), except for the above claim which is implied by the constraint. Thus, we could simply use $C_j + p_{j'} \leq C_{j'}$ (along with the definitions of C_j's) to replace Constraint (7) in the LP. However, in the algorithm for the problem with unit job lengths (described in Section III), we do need Constraint (7). To have a unified LP for both problems, we chose to use Constraint (7). Our algorithm does not use x-variables, but we need them in the analysis.

B. Analysis

Our analysis is very similar to that in [10]. We fix a job j^* from now on and we shall upper bound $\frac{\mathbb{E}[\widetilde{C}_{j^*}]}{C_{j^*}}$. Notice that once j^* is scheduled by the algorithm, $\widetilde{C}_{j^*}$ is determined and will not be changed later. Thus, we call the schedule at the moment the algorithm just scheduled j^* the *final schedule*.

We can then define idle and busy points and slots w.r.t this final schedule. We say a time point $\tau \in (0, T]$ is *busy* if the congestion of the intervals at τ is m in the schedule (in other words, all the m machines are being used at τ in the schedule); we say τ is *idle* otherwise. We say a left-open-right-closed interval (or slot) $(\tau, \tau']$ (it is possible that $\tau = \tau'$, in which case the interval is empty) is idle (busy, resp.) if all time points in $(\tau, \tau']$ are idle (busy, resp.).

Then we analyze the total length of busy and idle time slots before $\widetilde{C}_{j^*}$ respectively, w.r.t the final schedule. For a specific $\theta \in (0, 1/2]$, the techniques in [10] can bound the total length of idle slots by $\frac{C_{j^*}}{1-\theta}$ and the total length of busy slots by $\frac{C_{j^*}}{\theta}$. Thus choosing $\theta = 1/2$ gives the best 4-approximation, which is the best using this analysis. Our improvement comes from the bound on the total length of busy time slots. We show that the expected length of busy slots before $\widetilde{C}_{j^*}$ is at most $2C_{j^*}$, which is much better than the bound $\mathbb{E}_{\theta\sim_R(0,1/2]} \frac{C_{j^*}}{\theta} = \infty$ given by directly applying the bound for every θ. We remark that the $\frac{C_{j^*}}{\theta}$ bound for each individual θ is tight and thus can not be improved; our improvement comes from considering all possible θ's together.

Bounding the Expected Length of Idle Slots: We first bound the total length of idle slots before $\widetilde{C}_{j^*}$, the completion time of job j^* in the schedule produced by the algorithm. Lemma 6 and 7 are established in [10] and their proofs can be found in the full version of the paper for completeness.

Lemma 6. *Let $j \in J$ be a job in the final schedule with $\widetilde{S}_j > 0$. Then we can find a job j' such that*

- *either $j' \prec j$ and $(\widetilde{C}_{j'}, \widetilde{S}_j]$ is busy,*
- *or $M_{j'} \leq M_j, S_{j'} < S_j$ and $(S_{j'}, \widetilde{S}_j]$ is busy.*

Applying Lemma 6 repeatedly, we can identify a chain of jobs whose scheduling intervals cover all the idle slots before $\widetilde{C}_{j^*}$, which can be used to bound the total length of these slots. This leads to the following lemma from [10]:

Lemma 7. *The total length of idle time slots before $\widetilde{C}_{j^*}$ is at most $\frac{C_{j^*}}{1-\theta}$.*

Thus, the expected length of idle slots before $\widetilde{C}_{j^*}$, over all choices of θ, is at most

$$\int_{\theta=0}^{1/2} \frac{C_{j^*}}{1-\theta} 2\mathrm{d}\theta = \left(2\ln\frac{1}{1-\theta}\bigg|_{\theta=0}^{1/2}\right) C_{j^*} = (2\ln 2)C_{j^*}. \tag{10}$$

Bounding the Expected Length of Busy Slots: We now proceed to bound the total length of busy slots before $\widetilde{C}_{j^*}$. This is the key to our improved approximation ratio. For every $\theta \in [0, 1/2]$, let $J_\theta = \{j : M_j^\theta \leq \widetilde{C}_{j^*}\}$. Thus, if $\theta < \theta'$, we have $J_\theta \supseteq J_{\theta'}$. For every $\theta \in [0, 1/2]$ and $j \in J_0$, define $\theta_j = \sup\{\theta \in [0, 1/2] : j \in J_\theta\}$; this is well-defined since $j \in J_0$. For any subset $J' \subseteq J$ of jobs, we define $p(J') = \sum_{j\in J'} p_j$ to be the total length of all jobs in J'.

Lemma 8. *For a fixed $\theta \in (0, 1/2]$, the total length of busy slots before $\widetilde{C}_{j^*}$ is at most $\frac{1}{m}p(J_\theta)$.*

Proof: The total length of busy time slots in $(0, \widetilde{C}_{j^*}]$ is at most $\frac{1}{m}$ times the total length of jobs scheduled so far, which is at most

$$\frac{1}{m} \sum_{j\in J: M_j^\theta \leq M_{j^*}^\theta} p_j \leq \frac{1}{m} \sum_{j\in J: M_j^\theta \leq C_{j^*}} p_j = \frac{1}{m}p(J_\theta).$$

■

The key lemma for our improved approximation ratio is an upper bound on the above quantity when θ is uniformly selected from $(0, 1/2]$:

Lemma 9. $\displaystyle\int_{\theta=0}^{1/2} p(J_\theta)\mathrm{d}\theta \leq mC_{j^*}.$

Proof: Notice that we have

$$\int_{\theta=0}^{1/2} p(J_\theta)\mathrm{d}\theta = \int_{\theta=0}^{1/2} \sum_{j\in J_0} p_j \mathbf{1}_{j\in J_\theta}\mathrm{d}\theta$$
$$= \sum_{j\in J_0} p_j \int_{\theta=0}^{1/2} \mathbf{1}_{j\in J_\theta}\mathrm{d}\theta = \sum_{j\in J_0} \theta_j p_j.$$

Thus, it suffices to prove that $\sum_{j\in J_0} \theta_j p_j \leq mC_{j^*}$. To achieve this, we construct a set of axis-parallel rectangles. For each $j \in J_0$ and t such that $x_{j,t} > 0$, we place a rectangle with height $x_{j,t}$ and horizontal span $(t-p_j, t-p_j+2\theta_j p_j]$. The total area of all the rectangles for j is exactly $2\theta_j p_j$. Notice that $\theta_j \leq 1/2$ and thus $(t-p_j, t-p_j+2\theta_j p_j] \subseteq (t-p_j, t]$.

Notice that $\sum_t x_{j,t}(t-p_j+\theta_j p_j) = C_j-(1-\theta_j)p_j = M_j^{\theta_j} \leq C_{j^*}, \sum_t x_{j,t} = 1$, and $t-p_j+\theta_j p_j$ is the mass center[2] of the rectangle for (j,t). Thus, the mass center of the union of all rectangles for j is at most C_{j^*}. This in turn implies that the mass center of the union of all rectangles over all $j \in J_0$ and t, is at most C_{j^*}. Notice that for every $t \in (0,T]$, the total height of all rectangles covering t is at most m, by Constraint (6), and the fact that $(t-p_j, t-p_j+2\theta_j p_j] \subseteq (t-p_j, t]$ for every $j \in J_0$ and t. Therefore, the total area of rectangles for all $j \in J_0$ and t is at most $2mC_{j^*}$ (otherwise, the mass center will be larger than C_{j^*}). So, we have $\sum_{j\in J_0} 2\theta_j p_j \leq 2mC_{j^*}$, which finishes the proof of the lemma. ■

Thus, by Lemma 8 and Lemma 9, the expected length of busy time slots before $\widetilde{C}_{j^*}$ is at most

$$\int_{\theta=0}^{1/2} \frac{p(J_\theta)}{m} 2\mathrm{d}\theta = \frac{2}{m}\int_{\theta=0}^{1/2} p(J_\theta)\mathrm{d}\theta \leq 2C_{j^*}. \qquad (11)$$

Thus, by Inequalities (10) and (11), we have

$$\mathbb{E}\left[\widetilde{C}_{j^*}\right] \leq (2\ln 2)C_{j^*} + 2C_{j^*} = (2+2\ln 2)C_{j^*}.$$

Thus, we have proved the $2+2\ln 2+\epsilon \leq (3.387+\epsilon)$-approximation ratio for our algorithm, finishing the proof of Theorem 1.

Remarks: One might wonder if choosing a random θ from $[0,\theta^*]$ for a different θ^* can improve the approximation ratio. For $\theta^* \leq 1/2$, the ratio we can obtain is $\frac{1}{\theta^*} + \frac{1}{\theta^*}\int_{\theta=0}^{\theta^*}\frac{1}{1-\theta}\mathrm{d}\theta = \frac{1}{\theta^*} + \frac{1}{\theta^*}\ln\frac{1}{\theta^*}$; that is, the first factor (for busy time slots) will be increased to $1/\theta^*$ and the second factor (for idle time slots) will be decreased to $\frac{1}{\theta^*}\ln\frac{1}{\theta^*}$. This ratio is minimized when $\theta^* = 1/2$. If $\theta^* > 1/2$, however, the first factor does not improve to $1/\theta^*$, as the proof of Lemma 9 used the fact that $\theta_j \leq 1/2$ for each j. Thus, using our analysis, the best ratio we can get is $2+2\ln 2$.

III. Scheduling Unit-Length Jobs on Identical Machines with Job Precedence Constraints

In this section, we give our $(1+\sqrt{2})$-approximation algorithm for $P\big|\text{prec}, p_j = 1\big|\sum_j w_j C_j$. Again, we solve (LP$_{\text{P|prec|wC}}$) to obtain x; define $C_j = \sum_t x_{j,t}$ for every $j \in J$. For this special case, we define the random M-vector differently. In particular, it depends on the values of x variables. For every $j \in J$ and $\theta \in (0,1]$, define M_j^θ to be the minimum t such that $\sum_{t'=1}^t x_{j,t'} \geq \theta$. Notice that $C_j = \int_{\theta=0}^1 M_j^\theta \mathrm{d}\theta$. Our algorithm for $P\big|\text{prec}, p_j = 1\big|\sum_j w_j C_j$ chooses θ uniformly at random from $(0,1]$, and then call job-driven-list-scheduling(M^θ) and output the returned schedule.

For every $j \in J$, we define a_j to be the largest a such that there exists a sequence of a jobs $j_1 \prec j_2 \prec j_3 \prec \cdots \prec j_a = j$. Thus, a_j is the "depth" of j in the precedence graph.

Claim 10. *For every $j \in J$ and $t < a_j$, we have $x_{j,t} = 0$.*

Again, we fix a job j^* and focus on the schedule at the moment the algorithm just scheduled j^*; we call this schedule the final schedule. We shall bound $\mathbb{E}[\widetilde{C}_{j^*}]/C_{j^*}$ (recall that $\widetilde{C}_{j^*}$ is the completion time of j^* in the schedule we output), by bounding the total length of idle and busy slots in the final schedule before $\widetilde{C}_{j^*}$ separately. Recall that a time point is busy if all the m machines are processing some jobs at that time, and idle otherwise. The next lemma gives this bound for a fixed θ. The first (resp. second) term on the right side bounds the total length of busy (resp. idle) slots before $\widetilde{C}_{j^*}$. The clean bound a_{j^*} on the total length of idle slots comes from the unit-job size property.

Lemma 11. $\widetilde{C}_{j^*} \leq \frac{1}{m}\Big|\left\{j : M_j^\theta \leq M_{j^*}^\theta\right\}\Big| + a_{j^*}$.

We shall use $g(\theta) = M_{j^*}^\theta$ for every $\theta \in (0,1]$. Notice that $C_{j^*} = \int_{\theta=0}^1 g(\theta)\mathrm{d}\theta$. For simplicity, let $g(0) = \lim_{\theta\to 0^+} g(\theta)$; so, $g(0)$ will be the smallest t such that $x_{j^*,t} > 0$. By Claim 10, we have $g(0) \geq a_{j^*}$. For every $j \in J$ and $\theta \in [0,1]$, define $h_j(\theta) := \sum_{t=1}^{g(\theta)} x_{j,t}$. This is the total volume of job j scheduled in $(0, g(\theta)]$. Thus, we have $\sum_{j\in J} h_j(\theta) \leq g(\theta)$. Noticing that $M_j^\theta \leq M_{j^*}^\theta$ if and only if $h_j(\theta) \geq \theta$. So, by Lemma 11, we have $\widetilde{C}_{j^*} \leq g(0) + \frac{1}{m}\sum_{j\in J}\mathbf{1}_{h_j(\theta)\geq\theta}$. Thus, we can bound $\frac{\mathbb{E}[\widetilde{C}_{j^*}]}{C_{j^*}}$ by the superior of

$$\frac{g(0) + \frac{1}{m}\sum_{j\in J}\int_{\theta=0}^1 \mathbf{1}_{h_j(\theta)\geq\theta}\mathrm{d}\theta}{\int_{\theta=0}^1 g(\theta)\mathrm{d}\theta} \qquad (12)$$

subject to

- $g : [0,1] \to [1,\infty)$ is piecewise linear, left-continuous and non-decreasing, (12.1)
- $\forall j \in J$, $h_j : [0,1] \to [0,1]$ is piecewise linear, left-continuous and non-decreasing, (12.2)

[2]Here, we use mass center for the horizontal coordinate of the mass center, since we are not concerned with the vertical positions of rectangles.

- $\sum_{j \in J} h_j(\theta) \leq m g(\theta), \quad \forall \theta \in [0,1].$ (12.3)

To recover the 3-approximation ratio of [10], we know that $g(0) \Big/ \int_{\theta=0}^{1} g(\theta) \mathsf{d}\theta \leq 1$; this corresponds to the fact $a_{j^*} \leq \widetilde{C}_{j^*}$. It is not hard to show $\frac{1}{m} \sum_{j \in J} \int_{\theta=0}^{1} \mathbf{1}_{h_j(\theta) \geq \theta} \mathsf{d}\theta \Big/ \left(\int_{\theta=0}^{1} g(\theta) \mathsf{d}\theta \right) \leq 2$. The tight factor 2 can be achieved when $h_j(\theta) = \theta$ for every $j \in J$ and $\theta \in [0,1]$ and $g(\theta) = n\theta/m$. This corresponds to the following case: by the time θ-fraction of job j^* was completed, exactly θ faction of every job $j \in J$ was completed. However, the two bounds can not be tight simultaneously: the first bound being tight requires g to be a constant function, where the second bound being tight requires g to be linear in θ. This is where we obtain our improved approximation ratio.

Due to the page limit, we shall defer the proof that (12) is at most $1 + \sqrt{2}$ to the full version of the paper. Here we only give the combination of g and $\{h_j\}_{j \in J}$ achieving the bound; the way we prove the upper bound is by showing that this combination is the worst possible. In the worst case, we have $h_j(\theta) = \max\{\alpha, \theta\}$ for every $\theta \in [0,1]$, where $\alpha = \sqrt{2} - 1$. $g(\theta) = \frac{1}{m} \sum_{j \in J} h_j(\theta) = \frac{n}{m} \max\{\alpha, \theta\}$ for every $\theta \in [0,1]$. In this case, the numerator of (12) is $g(0) + \frac{1}{m} \sum_{j \in J} \int_{\theta=0}^{1} \mathbf{1}_{h_j(\theta) \geq \theta} \mathsf{d}\theta = (\alpha+1)\frac{n}{m}$; the denominator is $\int_{\theta=0}^{1} g(\theta) \mathsf{d}\theta = \frac{(1+\alpha^2)n}{2m}$. Thus, (12) in this case is $\frac{\alpha+1}{(1+\alpha^2)/2} = 1 + \sqrt{2}$.

IV. Scheduling on Related Machines with Job Precedence Constraints

In this section, we give our $O(\log m / \log\log m)$-approximation for $Q|\text{prec}|C_{\max}$ and $Q|\text{prec}|\sum_j w_j C_j$, proving Theorem 3. This slightly improves the previous best $O(\log m)$-approximation, due to Chudak and Shmoys [14]. Our improvement comes from a better tradeoff between two contributing factors.

As in [14], we can convert the objective of minimizing total weighted completion time to minimizing makespan, losing a factor of 16. We now describe the LP used in [14] and state the theorem for the reduction. Throughout this section, i is restricted to machines in M, and j and j' are restricted to jobs in J.

$$\min \quad D \qquad (\text{LP}_{\text{Q|prec|Cmax}})$$

$$\sum_i x_{i,j} = 1 \qquad \forall j \qquad (13)$$

$$p_j \sum_i \frac{x_{i,j}}{s_i} \leq C_j \qquad \forall j \qquad (14)$$

$$C_j + p_{j'} \sum_i \frac{x_{i,j'}}{s_i} \leq C_{j'} \qquad \forall j, j', j \prec j' \qquad (15)$$

$$\frac{1}{s_i} \sum_j p_j x_{i,j} \leq D \qquad \forall i \qquad (16)$$

$$C_j \leq D \qquad \forall j \qquad (17)$$

$$x_{i,j}, C_j \geq 0 \qquad \forall j, i \qquad (18)$$

($\text{LP}_{\text{Q|prec|Cmax}}$) is a valid LP relaxation for $Q|\text{prec}|C_{\max}$. In the LP, $x_{i,j}$ indicates whether job j is scheduled on machine i. D is the makespan of the schedule, and C_j is the completion time of j in the schedule. Constraint (13) requires every job j to be scheduled. Constraint (14) says that the completion time of j is at least the processing time of j on the machine it is assigned to. Constraint (15) says that if $j \prec j'$, then $C_{j'}$ is at least C_j plus the processing time of j' on the machine it is assigned to. Constraint (16) says that the makespan D is at least the total processing time of all jobs assigned to i, for every machine i. Constraint (17) says that the makespan D is at least the completion time of any job j. Constraint (18) requires the x and C variables to be non-negative.

The value of ($\text{LP}_{\text{Q|prec|Cmax}}$) provides a lower bound on the makespan of any valid schedule. However, even if we require each $x_{i,j} \in \{0,1\}$, the optimum solution to the integer programming is not necessarily a valid solution to the scheduling problem, since it does not give a scheduling interval for each job j. Nevertheless, we can use the LP relaxation to obtain our $O(\log m / \log\log m)$-approximation for $Q|\text{prec}|C_{\max}$. Using the following theorem from [14], we can extend the result to $Q|\text{prec}|\sum_j w_j C_j$:

Theorem 12 ([14]). *Suppose there is an efficient algorithm $\mathcal{A}$ that can round a fractional solution to ($\text{LP}_{\text{Q|prec|Cmax}}$) to a valid solution to the correspondent $Q|\text{prec}|C_{\max}$ instance, losing only a factor of α. Then there is a 16α-approximation for the problem $Q|\text{prec}|\sum_j w_j C_j$.*

Thus, from now on, we focus on the objective of minimizing the makespan; our goal is to design an efficient rounding algorithm as stated in Theorem 12 with $\alpha = O(\log m / \log\log m)$. We assume that m is big enough. For the given instance of $Q|\text{prec}|C_{\max}$, we shall first pre-processing the instance as in [14] so that it contains only a small number of groups. In the first stage of the pre-processing step, we discard all the machines whose speed is at most $1/m$ times the speed of the fastest machine. Since there are m machines, the total speed for discarded machines is at most the speed of the fastest machine. In essence, the fastest machine can do the work of all the discarded machines; this will increase the makespan by a factor of 2. Formally, let i^* be the machine with the fastest speed. For every discarded machine i and any job j such that $x_{i,j} > 0$, we shall increase $x_{i^*,j}$ by $x_{i,j}$ and change this $x_{i,j}$ to 0. By scaling D by a factor of 2, the LP solution remains feasible. To see this, notice that the modification to the fractional solution can only decrease $p_j \sum_i \frac{x_{i,j}}{s_i}$ for each j. The only constraint we need to check is Constraint (16) for $i = i^*$. Since $\sum_{i \text{ discarded},j} \frac{p_j x_{i,j}}{s_{i^*}} \leq \sum_{i \text{ discarded},j} \frac{p_j x_{i,j}}{m s_i} \leq \sum_{i \text{ discarded}} \frac{D}{m} \leq D$, moving the scheduling of jobs from discarded machines to i^* shall only increase the processing time of jobs on i^* by D. Thus, we assume all machines have speed larger than $1/m$ times the speed of the fastest machine. By scaling speeds of machines uniformly, we assume all machines i have speed $i \in [1, m)$, and $|M| \leq m$.

In the second stage of the pre-processing step, we partition the machines into groups, where each group contains machines with similar speeds. Let $\gamma = \log m / \log\log m$. Then group M_k contains machines with speed in $[\gamma^{k-1}, \gamma^k)$, where $k = 1, 2, \cdots, K := \lceil \log_\gamma m \rceil = O(\log m / \log\log m)$. We remark that that unlike [14], we *can not* round down the speed of each machine i to the nearest power of γ. If we do so, we will lose a factor of $(\log m / \log\log m)$ and finally we can only obtain an $O((\log m / \log\log m)^2)$-approximation. Instead, we keep the speeds of machines unchanged.

We now define some useful notations. For a subset $M' \subseteq M$ of machines, we define $s(M') = \sum_{i \in M'} s_i$ to be the total speed of machines in M'; for $M' \subseteq M$ and $j \in J$, let $x_{M',j} = \sum_{i \in M'} x_{i,j}$ be the total fraction of job j assigned to machines in M'.

For any job j, let ℓ_j be the largest integer ℓ such that $\sum_{k=\ell}^{K} x_{M_k,j} \geq 1/2$. That is, the largest ℓ such that at least $1/2$ fraction of j is assigned to machines in groups ℓ to K. Then, let k_j be the index $k \in [\ell_j, K]$ that maximizes $s(M_k)$. That is, k_j is the index of the group in groups ℓ_j to K with the largest total speed. Later in the machine-driven list scheduling algorithm, we shall constrain that job j can only be assigned to machines in group k_j. The following claim says that the time of processing j on any machine in M_{k_j} is not too large, compared to processing time of j in the LP solution.

Claim 13. *For every $j \in J$, and any machine $i \in M_{k_j}$, we have $\frac{p_j}{s_i} \leq 2\gamma \sum_{i' \in M} \frac{p_j x_{i',j}}{s_{i'}}$.*

Proof: Notice that $\sum_{k=\ell_j+1}^{K} x_{M_k,j} < 1/2$ by our definition of ℓ_j. Thus, $\sum_{k=1}^{\ell_j} x_{M_k,j} > 1/2$. Then, $\sum_{i' \in M} \frac{x_{i',j}}{s_{i'}} \geq \sum_{i' \in \bigcup_{k=1}^{\ell_j} M_k} \frac{x_{i',j}}{s_{i'}} \geq \frac{1}{2} \cdot \gamma^{-\ell_j}$. This is true since $\sum_{i' \in \bigcup_{k=1}^{\ell_j} M_k} x_{i',j} \geq 1/2$ and every i' in the sum has $\frac{1}{s_{i'}} \geq \gamma^{-\ell_j}$.

Since i is in group $k_j \geq \ell_j$, i' has speed at least $\gamma^{\ell_j - 1}$ and thus $\frac{1}{s_i} \leq \gamma^{1-\ell_j}$. Then the claim follows. ■

Claim 14. $\sum_{j \in J} \frac{p_j}{s(M_{k_j})} \leq 2KD.$

Proof: Focus on each job $j \in J$. Noticing that $\sum_{k=\ell_j}^{K} x_{M_k,j} \geq 1/2$, and k_j is the index of the group with the maximum total speed, we have

$$\sum_{k=1}^{K} \frac{x_{M_k,j}}{s(M_k)} \geq \sum_{k=\ell_j}^{K} \frac{x_{M_k,j}}{s(M_k)} \geq \frac{1}{2s(M_{k_j})}.$$

Summing up the above inequality scaled by $2p_j$, over jobs j, we have

$$\sum_{j \in J} \frac{p_j}{s(M_{k_j})} \leq 2 \sum_{j \in J} p_j \sum_{k=1}^{K} \frac{x_{M_k,j}}{s(M_k)}$$
$$= 2 \sum_{k=1}^{K} \frac{1}{s(M_k)} \sum_{j \in J} p_j x_{M_k,j} \leq 2 \sum_{k=1}^{K} D = 2KD.$$

To see the last inequality, we notice that $\sum_{j \in J} p_j x_{M_k,j}$ is the total size of jobs assigned to group k, $s(M_k)$ is the total speed of all machines in M_k and D is the makespan. Thus, we have $\sum_{j \in J} p_j x_{M_k,j} \leq s(M_k)D$. Formally, Constraint (16) says $\sum_{j \in J} p_j x_{i,j} \leq s_i D$ for every $i \in M_k$. Summing up the inequalities over all $i \in M_k$ gives $\sum_{j \in J} p_j x_{M_k,j} \leq s(M_k)D$. ■

With the k_j values, we can run the machine-driven list-scheduling algorithm in [14]. The algorithm constructs the schedule in real time. Whenever a job completes (or at the beginning of the algorithm), for each idle machine i, we attempt to schedule an unprocessed job j on i subject to two constraints: (i) machine i can only pick a job j if $i \in M_{k_j}$ and (ii) all the predecessors of j are completed. If no such job j exists, machine i remains idle until a new job is competed. We use

$\mathcal{S}$ to denote this final schedule; Let $i_j \in M_{k_j}$ be the machine that process j in the schedule constructed by our algorithm.

The following simple observation is the key to prove our $O(\log m/\log\log m)$-approximation. Similar observations were made and used implicitly in [14], and in [5] for the problem on identical machines. However, we think stating the observation in our way makes the analysis cleaner and more intuitive. We say a time point t is critical, if some job starts or ends at t. To avoid ambiguity, we exclude these critical time points from our analysis (we only have finite number of them). At any non-critical time point t in the schedule, we say a job j is minimal if all its predecessors are completed but j itself is not completed yet.

Observation 15. *At any non-critical time point t in $\mathcal{S}$, either all the minimal jobs j are being processed, or there is a group k such that all machines in M_k are busy.*

Proof: All the minimum jobs at t are ready for processing. If some such job j is not processed at t, it must be the case that all machines in M_{k_j} are busy. ■

As time goes in $\mathcal{S}$, we maintain the precedence graph over J', the set of jobs that are not completed yet: we have an edge from $j \in J'$ to $j' \in J'$ if $j \prec j'$. At any time point, the weight of a job j is the time needed to complete the rest of job j on i_j, i.e, the size of the unprocessed part of job j, divided by s_{i_j}. If at t, all minimum jobs are being processed, then the weights of all minimal jobs are being decreased at a rate of 1. Thus, the length of the longest path of in the precedence graph is being decreased at a rate of 1. The total length of the union of these time points is at most length of the longest path in the precedence graph at time 0, which is at most

$$\max_H \sum_{j\in H} \frac{p_j}{s_{i_j}} \le \max_H 2\gamma \sum_{j\in H}\sum_{i\in M} \frac{p_j x_{i,j}}{s_i} \le 2\gamma D,$$

where H is over all precedence chains of jobs. The first inequality is by Claim 13 and the second inequality is by Constraints (15) and (17) in the LP.

If not all the minimal jobs are being processed at time t, then there must be a group k such that all machines in group k are busy, by Observation 15. The total length of the union of all these points is at most

$$\sum_k \frac{\sum_{j:k_j=k} p_j}{s(M_k)} = \sum_{j\in J} \frac{p_j}{s(M_{k_j})} \le 2KD,$$

by Claim 14.

Thus, our schedule has makespan at most $2(\gamma + K)D = O(\log m/\log\log m)D$, leading to an $O(\log m/\log\log m)$-approximation for $Q|\text{prec}|C_{\max}$. Combining this with Theorem 12, we obtain an $O(\log m/\log\log m)$-approximation for $Q|\text{prec}|\sum_j w_jC_j$, finishing the proof of Theorem 3. Indeed, as shown in [14], this factor is tight if we use $(\text{LP}_{\text{Q|prec|Cmax}})$.

Acknowledgment

References

[1] N. Bansal, A. Srinivasan, and O. Svensson, "Lift-and-round to improve weighted completion time on unrelated machines," in *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '16. ACM, 2016, pp. 156–167.

[2] S. Im and S. Li, "Better unrelated machine scheduling for weighted completion time via random offsets from non-uniform distributions," in *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016*, 2016, pp. 138–147.

[3] M. Skutella, "A 2.542-approximation for precedence constrained single machine scheduling with release dates and total weighted completion time objective," *Operations Research Letters*, vol. 44, no. 5, pp. 676 – 679, 2016.

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

[5] R. L. Graham, "Bounds on multiprocessing timing anomalies," *SIAM JOURNAL ON APPLIED MATHEMATICS*, vol. 17, no. 2, pp. 416–429, 1969.

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

[7] N. Bansal and S. Khot, "Optimal long code test with one free bit," in *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, ser. FOCS '09. IEEE Computer Society, 2009, pp. 453–462.

[8] O. Svensson, "Conditional hardness of precedence constrained scheduling on identical machines," in *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, ser. STOC '10. ACM, 2010, pp. 745–754.

[9] L. A. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein, "Scheduling to minimize average completion time: Off-line and on-line approximation algorithms," *Math. Oper. Res.*, vol. 22, no. 3, pp. 513–544, Aug. 1997.

[10] A. Munier, M. Queyranne, and A. S. Schulz, *Approximation Bounds for a General Class of Precedence Constrained Parallel Machine Scheduling Problems*. Springer Berlin Heidelberg, 1998, pp. 367–382.

[11] M. Queyranne and A. S. Schulz, "Approximation bounds for a general class of precedence constrained parallel machine scheduling problems," *SIAM J. Comput.*, vol. 35, no. 5, pp. 1241–1253, May 2006.

[12] S. Chakrabarti, C. A. Phillips, A. S. Schulz, D. B. Shmoys, C. Stein, and J. Wein, *Improved scheduling algorithms for minsum criteria*. Springer Berlin Heidelberg, 1996, pp. 646–657.

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

[14] F. A. Chudak and D. 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*, ser. SODA '97. Society for Industrial and Applied Mathematics, 1997, pp. 581–590.

[15] J. M. Jaffe, "Efficient scheduling of tasks without full use of processor resources," *Theoretical Computer Science*, vol. 12, no. 1, pp. 1 – 17, 1980.

[16] M. Queyranne and M. Sviridenko, "Approximation algorithms for shop scheduling problems with minsum objective," *Journal of Scheduling*, vol. 5, no. 4, pp. 287–305, 2002.

[17] A. Bazzi and A. Norouzi-Fard, *Towards Tight Lower Bounds for Scheduling Problems*. Springer Berlin Heidelberg, 2015, pp. 118–129.

[18] A. S. Schulz and M. Skutella, "Scheduling unrelated machines by randomized rounding," *SIAM J. Discret. Math.*, vol. 15, no. 4, pp. 450–469, Apr. 2002.

[19] M. Skutella, "Convex quadratic and semidefinite programming relaxations in scheduling," *J. ACM*, vol. 48, no. 2, pp. 206–242, Mar. 2001.

[20] J. Sethuraman and M. S. Squillante, "Optimal scheduling of multiclass parallel machines," in *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '99. Society for Industrial and Applied Mathematics, 1999, pp. 963–964.

[21] M. X. Goemans, "Improved approximation algorthims for scheduling with release dates," in *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '97. Society for Industrial and Applied Mathematics, 1997, pp. 591–598.

[22] C. Phillips, C. Stein, and J. Wein, "Minimizing average completion time in the presence of release dates," *Mathematical Programming*, vol. 82, no. 1, pp. 199–223, Jun 1998.

[23] C. Chekuri, R. Motwani, B. Natarajan, and C. Stein, "Approximation techniques for average completion time scheduling," *SIAM J. Comput.*, vol. 31, no. 1, pp. 146–166, Jan. 2002.

[24] C. Chekuri and S. Khanna, "Approximation algorithms for minimizing average weighted completion time," *Handbook of Scheduling: Algorithms, Models, and Performance Analysis. CRC Press, Inc., Boca Raton, FL, USA*, 2004.

[25] M. Skutella and G. J. Woeginger, "A ptas for minimizing the weighted sum of job completion times on parallel machines," in *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, ser. STOC '99. ACM, 1999, pp. 400–407.

[26] C. Chekuri and S. Khanna, *A PTAS for Minimizing Weighted Completion Time on Uniformly Related Machines*. Springer Berlin Heidelberg, 2001, pp. 848–861.

[27] V. A. Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan, "Minimum weighted completion time," in *Encyclopedia of Algorithms*. Springer, 2008, pp. 544–546.

[28] A. S. Schulz and M. Skutella, "Random-based scheduling: New approximations and LP lower bounds," in *Proceedings of the International Workshop on Randomization and Approximation Techniques in Computer Science*, ser. RANDOM '97. Springer-Verlag, 1997, pp. 119–133.

[29] E. Levey and T. Rothvoss, "A (1+epsilon)-approximation for makespan scheduling with precedence constraints using LP hierarchies," in *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '16. ACM, 2016, pp. 168–177.

[30] J. K. Lenstra, D. B. Shmoys, and E. Tardos, "Approximation algorithms for scheduling unrelated parallel machines," *Math. Program.*, vol. 46, no. 3, pp. 259–271, Feb. 1990.

[31] T. Ebenlendr, M. Krčál, and J. Sgall, "Graph balancing: A special case of scheduling unrelated parallel machines," in *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '08. Society for Industrial and Applied Mathematics, 2008, pp. 483–490.

[32] O. Svensson, "Santa claus schedules jobs on unrelated machines," in *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, ser. STOC '11. ACM, 2011, pp. 617–626.

[33] D. Chakrabarty, S. Khanna, and S. Li, "On $(1, \epsilon)$-restricted asgsignment makespan minimization," in *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015)*, 2015.

[34] K. Jansen and L. Rohwedder, "On the configuration-LP of the restricted assignment problem," in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '17. Society for Industrial and Applied Mathematics, 2017, pp. 2670–2678.