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

Shi Li *

Abstract

We study the approximation of scheduling problems with the objective of minimizing total weighted completion time, under many different machine models: identical machine model with job precedence constraints, with uniform and non-uniform job sizes, open shop model and related machine model with job precedence constraints. For these problems, we give approximation algorithms that improve upon the 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 LP relaxations for their respective problems, but are not studied explicitly 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 the use of this time-indexed LP relaxation can provide new insights into this problem.

1 Introduction

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

Despite of these impressive results, the approximability of many problems are still poorly understood. Many of the state-of-art results were developed in late 1990s or early 2000s and have not been improved since. Continuing the recent surge of interest on the total weighted completion time objective [2, 11, 22], 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, open shop model, 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 nature of scheduling problems, these are natural LP relaxations for their respective problems. Though some of these LPs were studied in the literature (e.g. [17, 9]), the power of this technique in deriving improved approximation ratios is not well-exploited. Compared to other types of LPs, a solution to a time-indexed LP relaxation gives how the jobs are scheduled the machines fractionally. For many problems we study in this paper, this information allows us to identify the loose parts in previous state-of-art results and thus give the improved algorithms.

---

*Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY, USA
We now formally describe the problems we study in the paper and discuss the known 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 $j$. For simplicity, we do not repeat this global setting when defining 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. Needless to say, 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. [8], this problem is described as $P|\text{prec}|\sum_j w_j C_j$.

For the problem, the current best approximation algorithm is a 4-approximation algorithm due to Munier, Queyranne and Schulz ([14], [15]). The algorithm solves some LP relaxation to obtain a completion time vector $C \in \mathbb{R}_{\geq 0}$ and then run a job-driven list-scheduling algorithm that considers jobs one by one, in some order defined by the vector $C$. Since then it has remained an open problem to improve this factor of 4 (see the discussion after Open Problem 9 in [19]). They also considered an important special case of the problem, denoted as $P|\text{prec}, p_j = 1|\sum_j w_j C_j$, 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 also has not been improved.

On the negative side, Lenstra and Rinnooy Kan [12] proved a $(4/3 - \epsilon)$-hardness of approximation for the problem, with the objective of minimizing the makespan of the schedule (denoted as $P|\text{prec}|C_{\text{max}}$). With precedence constraints, the objective of minimizing weighted completion time is more general.\(^1\) Thus, $P|\text{prec}|\sum_j w_j C_j$ is also NP-hard to approximate within a factor of $4/3 - \epsilon$ for any $\epsilon > 0$. Better hardness result can be obtained with stronger complexity assumptions. Bansal and Khot [1] showed the special case of the problem where there is only one machine (denoted as $1|\text{prec}|\sum_j w_j C_j$) can not be approximated within a factor of $2 - \epsilon$, assuming some stronger version of the Unique Game Conjecture. Svensson [24] used the result to show that $P|\text{prec}|C_{\text{max}}$ is hard to approximate within a factor of $2 - \epsilon$, with the same assumption. The hardness results carry over to $P|\text{prec}|\sum_j w_j C_j$, since it is a generalization of both $1|\text{prec}|\sum_j w_j C_j$ and $P|\text{prec}|C_{\text{max}}$. However, no better hardness results are known for $P|\text{prec}|\sum_j w_j C_j$, with or without the stronger assumption.

**Open Shop Scheduling**  Then, we consider the open shop scheduling problem. In this problem, each job $j \in J$ consists of $m$ operations $\{O_{i,j}\}_{i \in M}$, where operation $O_{i,j}$ for the job $j$ has size $p_{i,j} \in \mathbb{Z}_{\geq 0}$ and has to be processed on machine $i$. Any any time point $t$, at most one operation can be processed on any machine $i$, and at most one operation for any job $j$ can be processed across all machines. Again, we consider non-preemptive schedules, in which each operation $O_{i,j}$ must be processed during a time interval of length $p_{i,j}$ on machine $i$. Given a valid schedule, the completion time $C_j$ of a job $j$ is the time when all operations for $j$ are completed. The problem is denoted as $O||\sum_j w_j C_j$ using the three-field notation.

The current best approximation algorithm for the problem is given by Queyranne and Sviridenko [16], which achieves a $3 + 2\sqrt{2} \approx 5.83$ approximation. Their algorithm first solves an LP relaxation for the problem, manually defines a set of precedence constraint over $J$ and then run a machine-driven list-scheduling algorithm respecting the defined precedence constraints. On the negative side, only APX-hardness result is known for this problem [10].

\(^1\)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.
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
$P|\text{prec}|\sum_j w_j C_j$. In addition, each machine $i \in M$ is given a speed $s_i > 0$ and the time of
processing job $j$ on machine $i$ is $p_j/s_i$ (thus the $m$ machines are not identical any more). Again
we do not allow preemptions: 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 $Q|\text{prec}|\sum_j w_j C_j$.
The best approximation algorithm for the problem has been the $O(\log m)$-approximation due to
Chudak and Shmoys [6]. They first reduce the problem of minimizing total weighted completion
time to that of minimizing makespan, with a constant loss in the approximation ratio. The
algorithm for minimizing the makespan solves an LP relaxation and then decides the group
of machines that each job assigned to. Then it runs a machine-based list scheduling problem
respecting the job-to-group assignment. On the negative side, all the hardness results for
$P|\text{prec}|\sum_j w_j C_j$ carry over to this problem. Recently Bazzi and Norouzi-Fard [3] showed that
assuming the hardness of some optimization problem on $k$-partite graphs, the problem is hard
to approximate within any constant.

Scheduling on Unrelated Machines  Finally, we consider the classic scheduling problem
to minimize total weighted completion time on unrelated machines (without precedence con-
straints). In this problem we are given a number $p_{i,j} \in \mathbb{Z}_{\geq 0}$ for every $i \in M, j \in J$, indicating
the time of processing job $j$ on machine $i$. To schedule a job $j$, we need to process it on some
machine $i$ in an interval of length $p_{i,j}$. This problem is described as $R||\sum_j w_j C_j$.
For this problem, many classic results gave $3/2$-approximation algorithms, based on time-
indexed LP relaxation [17] and convex-programming relaxation ([21], [20]). Improving the
3/2-approximation ratio had been a long-standing open problem ([19]), until Bansal, Srinivasan
and Svensson [2] recently gives a breakthrough result of $(3/2 - c)$-approximation for the problem,
for some tiny constant $c > 0$. Their algorithm is based on a novel dependence rounding scheme
and a lift-and-project SDP relaxation for the problem.

1.1 Our Results

We give improved approximation algorithms for $P|\text{prec}|\sum_j w_j C_j$, $P|\text{prec}, p_j = 1|\sum_j w_j C_j$,
$O||\sum_j w_j C_j$ and $Q|\text{prec}|\sum_j w_j C_j$. The previous best results for these problems have stood for
15 to 20 years.

For the scheduling problem on identical machines with precedence constraints, we improve
the long-standing approximation ratio of 4 by Munier, Queyranne and Schulz [14, 15]:

**Theorem 1.1.** There is a $2+2 \ln 2 + \epsilon < (3.387+\epsilon)$-approximation algorithm for $P|\text{prec}|\sum_j w_j C_j$.

For the special case of the problem in which all jobs have size 1, we improve their 3-
approximation ratio:

**Theorem 1.2.** There is a $1+\sqrt{2} < 2.415$-approximation algorithm for $P|\text{prec}, p_j = 1|\sum_j w_j C_j$.

For open shop scheduling, we improve upon the $3+2\sqrt{2} \leq 5.83$-approximation algorithm of
Queyranne and Sviridenko [16]:

**Theorem 1.3.** There is a $(5.102 + \epsilon)$-approximation algorithm for $O||\sum_j w_j C_j$.

For the related machine scheduling in presence of precedence constraints, we give a slightly
better approximation ratio than $O(\log m)$ due to Chudak and Shmoys [6]. As in [6], we first
reduce the objective of minimizing total weighted completion time to minimizing makespan.
Thus, we have improved algorithms for both problems:
Theorem 1.4. There are $O(|\lg m/\lg \lg m|)$-approximation algorithms for both $Q\{\text{prec}\}|C_{\text{max}}$ and $Q\{\text{prec}\}\sum |w_jC_j|$. For the classic problem $R||\sum |w_jC_j|$ of minimizing total weighted completion time on unrelated-machines, besides the slightly improved approximation ratio (the constant $c$ in [2] was $1/(108 \times 20000)$), our main result is that the $(1.5-c)$-approximation algorithm of [2] can also be achieved using a natural time-indexed LP relaxation:

Theorem 1.5. The LP relaxation $(\text{LP}_{R||wC})$ for $R||\sum |w_jC_j|$ has an integrality gap of at most $1.5-c$, where $c = \frac{1}{6000}$. Moreover, there is an efficient randomized algorithm that outputs a valid schedule whose expected cost is at most $(1.5-c)$ times the value of the LP relaxation.

1.2 Our Techniques

As we mentioned, a common key technique in many of our results is the use of time-indexed LP relaxations. Compared to many other types of LP relaxations, time-indexed LP relaxations give more information on how the jobs are actually scheduled on machines. We have variables $x_{i,j,t}$ to indicate whether job $j$ is scheduled on machine $i$ during the time-interval $(t-p_{i,j}, t]$, where $p_{i,j}$ is the processing time of $j$ on machine $i$. With these variables, it is straightforward to express the objective function, and formulate the machine-capacity and precedence constraints. We can visualize each $x_{i,j,t}$ as a rectangle with horizontal span $(t-p_{i,j}, t]$ and height $x_{i,j,t}$ for machine $j$ on job $i$. Then the completion times, the machine-capacity and precedence constraints all have intuitive meanings under the visualization. The structures of rectangles allow us to not only recover the previous state-of-art results, but also derive the improved approximation results by identifying the loose parts in these algorithms and analysis.

$P\{\text{prec}\}\sum |w_jC_j|$. Let us first consider the scheduling problem on identical machines with job precedence constraints. The 4-approximation algorithm of Munier, Queyranne, and Schulz [14, 15] used an LP that only contains the completion time variables $\{C_j\}_{j \in I}$. After solving the LP, they run a job-driven list scheduling algorithm, by considering jobs $j$ in increasing order of $C_j - p_j/2$. To analyze for expected completion time of $j^*$ in the output schedule, they consider two quantities separately: the total length of busy slots (i.e, the time slots when all the $m$ machines are used), and the total length of idle slots before the completion of $j^*$. They show that the total length of busy time slots can be bounded by $2C_{j^*}$, using the machine-capacity constraints in the LP. The total 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 structure of rectangles obtained from solving our time-indexed LP allows us to examine the tightness of the above factors more carefully: though the $1/\theta$ factor for the busy slots is tight for every individual $\theta \in [0, 1/2]$, it can not be tight for every such $\theta$. Roughly speaking, the $1/\theta$ factor is tight for job $j^*$ only when job $j^*$ 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 slightly smaller than $C_{j^*} - \theta p_{j^*}$. However in this case, if we decrease $\theta$ slightly, these jobs $j$ will be considered after $j^*$. With this observation, we show that even if we choose $\theta$ uniformly at random from $[0, 1/2]$, the factor for busy time slots remains 2, as opposed to $\int_{\theta=0}^{1/2} \frac{2}{-\theta} d\theta = \infty$. This will decrease the factor for idle slots to $\int_{\theta=0}^{1/2} \frac{2}{-\theta} d\theta = 2 \ln 2$, thus improving the approximation factor to $2 + 2 \ln 2$. Our analysis needs to use the our time-indexed LP relaxation (which is at least as strong as the LP of [14]). The structure of rectangles given by the LP solution enables us to connect the algorithms for all $\theta$ values. In contrast, a solution to the LP of [14] lacks this structure.
When jobs have uniform length, the approximation ratio of the algorithm of [14] improves to 3. In this case, the parameter in the above algorithm becomes useless since all jobs have the same \( p_j \). Taking the advantage of the uniform job length, the factor for idle time slots becomes 1. The factor for busy slots remains 2 and thus this gives an approximation factor of 3 for the special case.

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

We again use the structure of rectangles to discover the loose parts in the analysis. With uniform job size, the idle slots before a job \( j \) are caused by the precedence constraints. That is, if the total length of idle slots before the completion of \( j \) is \( a \), then there is a precedence-chain of \( a \) jobs, with the last one being \( j \); 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 of 2 for busy time slots can not be tight. Roughly speaking, the factor of 2 for busy time slots is tight only when the scheduling of job \( j \) is uniformly distributed in \([0,2C_j]\). However, if \( j \) is at depth-\( a \approx C_j \) in the dependence graph, it can not be scheduled with a positive fraction before time \( a \). With this observation, we are able to derive the improved approximation ratio \( 1 + \sqrt{2} \) for this special case.

Our improved \( O(1) \)-approximation for related machine scheduling problem. Queyranne and Sviridenko [16] gave a \((3 + 2\sqrt{2})\)-approximation based on some LP relaxation that contains only completion time variables. In the algorithm, they first manually define a set of precedence constraints, based on the completion times given by the LP solution. A job \( j \) is a precedent of another job \( j' \) if job \( j \) has much smaller completion time than job \( j' \); more specifically, \( C_j \leq C_j'/\gamma \) for some \( \gamma > 1 \). Then they run a machine-driven list-scheduling algorithm, which constructs the schedule in real-time. The analysis for the completion time of each job \( j^* \) is broken into two separate bounds. Assuming \( O_{i,j^*} \) is the operation for \( j^* \) that is completed the last, then they upper bound the total lengths of idle and busy slots on machine \( i \) before the completion of \( O_{i,j^*} \) separately. The total length of idle slots can be bounded by \( \frac{1}{\gamma} C_j \), and the total length of busy slots can be bounded by \( 2\gamma C_j \). This gives a \((\frac{1}{\gamma} + 2\gamma)\)-approximation for the problem. By setting \( \gamma = 1 + \sqrt{2}/2 \), the ratio is \( 3 + 2\sqrt{2} \).

Let \( L_j = \sum_i p_{i,j} \) be the total length of all operations for job \( j \). We notice that for jobs \( j \) with smaller \( L_j/C_j \), we can afford to make more jobs to be predecessors of \( j \). Thus, our new manual definition of precedence constraints takes the job lengths into account: we define \( j \prec j' \) if \( C_j \leq C_j'/\theta L_{j'} \) for some \( \theta = (0,1) \). The definition coincides with the definition of [16] when \( \theta = 1 - 1/\gamma \) and \( L_{j'} = C_j' \). As \( L_{j'} \) gets smaller, jobs are more likely to be the predecessors of \( j' \). With this new definition, we can still prove the \( \frac{1}{\gamma - 1} = \frac{1}{\theta} \) factor on the total length idle slots before completion of \( O_{i,j^*} \); on the other hand, more jobs will become successors of \( j^* \), allowing us to reduce the \( 2\gamma \) factor for the total length of busy slots. A recovery of the \( 2\gamma = \frac{2}{1-\theta} \) factor using our time-indexed LP shows that the factor is tight only when many jobs \( j \) with \( C_j \approx C_j'(1-\theta) \) and \( L_j/C_j \approx 0 \) are scheduled before \( j^* \). With our new definition of precedence constraints, these jobs \( j \) will be successors of \( j^* \) and thus can not be scheduled before \( j^* \).

Our improved \( O(\log m/\log\log m) \)-approximation for related machine scheduling in presence of job precedence constraints is the simplest one; it is achieved by a better selection of parameters. The \( O(\log m) \)-approximation algorithm of Chudak and Shmoys [6] works as
follows. By losing a constant factor in the approximation ratio, we can assume our objective is to minimize the makespan. Then the algorithm solves an LP relaxation, partitions machines into $O(\lg m)$ groups according to their speeds, and then decides the machine group each job can be assigned to. Then it runs a machine-driven list-scheduling algorithm to schedule the jobs, subject to the precedence constraint, and the job-to-group assignment. The final approximation ratio is the sum of two factors: one coming from grouping machines with different speeds into the same group, which is $O(1)$ in [6], and the other coming from the precedence constraints between jobs assigned to different groups, which is $O(\lg m)$ in [6]. To improve the ratio, we make the speed difference between machines in the same group as large as $O(\lg m/\lg \lg m)$, so that we only have $O(\lg m/\lg \lg m)$ groups. This way, both factors become $O(\lg m/\lg \lg m)$-approximation algorithm for the problem. One minor remark is that in the algorithm of [6], 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 case, we can not make this assumption, since otherwise it would only lead to an $O((\lg / \lg \lg m)^2)$-approximation. In our algorithm, we keep the original speeds for all machines; by doing so, we can make sure that the final ratio is the sum, instead of the product, of the 2 factors of $O(\lg m/\lg \lg m)$.

$R||\sum_j w_jC_j$ Then we sketch how we use our time-indexed LP to recover the $(1.5−c)$-approximation of [2] (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 previous 1.5-approximation algorithms ([20, 21, 17]) are all based on independent rounding: the LP relaxations have $y_{i,j}$ variables, which indicate the fraction of job $j$ that is assigned to machine $i$. Then the algorithms will randomly and independently assign each job $j$ to a machine $i$, according to the distribution $\{y_{i,j}\}$.

However, as shown in [2], independent rounding algorithms can not give a better than 1.5 approximation, even if the fractional solution is already a convex combination of optimum integral schedules. To overcome this barrier, [2] introduced a novel dependence rounding scheme, which guarantees some strong negative correlation between events that jobs are assigned to the same machine $i$. More specifically, the rounding scheme 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 a natural time-indexed interval-LP relaxation for the problem suffices to give a $(1.5−c)$-approximation. Our time-indexed LP relaxation has a variable on each interval that a job $j$ can be assigned to: $x_{i,j,s}$ indicates whether job $j$ is assigned to machine $i$ during the time interval $(s, s+p_{i,j})$. For this time interval, we can use $s+p_{i,j}$ as the completion time. This makes our LP stronger than the time-indexed LP in [17] that is used to give an $(1.5+\epsilon)$-approximation for this problem. Though being a natural LP, this time-indexed interval LP relaxation for this problem was never studied before. In a recent and related result, Im and Li [11] used the LP (with consideration of job arrival times) to give a 1.8687-approximation for $R||\sum_j w_jC_j$, the generalization of $R||\sum_j w_jC_j$ to the case where jobs have arrival times.

We use the dependence rounding scheme of [2] as a black box. To apply the scheme, we need to construct a grouping for every machine $i$. In our recovered 1.5-approximation algorithm for the problem using our time-indexed interval-LP, we note that the expected completion time of $j$ is 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. Thus, for each machine \( i \), the bad jobs are those with a large weight of scheduling intervals starting 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 in 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. Roughly speaking, 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 be grouped together. Besides the improved approximation ratio, we believe our algorithm will provide more insights into this problem, as our natural time-indexed LP relaxation is considerably simpler than the lift-and-project SDP of [2]. Another property that our rounding algorithm has is that it is oblivious to the weights of the jobs. This may be useful when we extend the algorithm to other variants of the problem.

Finally, we remark that all of our results except the one for \( R || \sum_j w_j C_j \) can be easily extended to the case when jobs have arrival times. However, to deliver our ideas more efficiently, we chose not to consider arrival times.

### 1.3 Other Related Work

There is a vast literature on approximating algorithms for scheduling problems to minimize the total weighted completion time. We can only discuss the ones that are most relevant to our results; we refer readers to [4] for a more comprehensive overview. For the problem of \( 1|\text{prec}||\sum_j w_j C_j \), i.e., the special case of \( P|\text{prec}||\sum_j w_j C_j \) where there is only 1 machine, there is a 2-approximation algorithm [9]. As we already mentioned, assuming a stronger version of the Unique Game Conjecture, the problem is NP-hard to approximate within a factor of \( 2 - \epsilon \) ([24]). When there are no precedence constraints, the problems of minimizing total weighted completion time on identical and related machines admit PTASes ([23, 5]).

Makespan minimization is a closely related objective to the objective of minimizing the total weighted completion time. The makespan, denoted as \( C_{\text{max}} \) is the maximum completion time over all jobs. For the problem \( P|\text{prec}|C_{\text{max}} \), i.e, the problem of makespan minimization on identical machines with job precedence constraints, the classic machine-driven list-scheduling algorithm of Graham [7] gives a 2-approximation. As we mentioned, there is a 4/3-hardness of approximation for the problem [12]; assuming a stronger version of the unique game conjecture, the problem is NP-hard to be approximated within a factor of \( 2 - \epsilon \) [1]. On related machines, the result of [6] shows that \( Q|\text{prec}||\sum_j w_j C_j \) and \( Q|\text{prec}|C_{\text{max}} \) are equivalent, up to a constant factor loss in the approximation ratio, which can be ignored with current best approximation ratios for both problems standing at \( O(\lg m / \lg \lg m) \) (our result). For the problem \( R||C_{\text{max}} \), i.e, the scheduling of jobs on unrelated machines to minimize the makespan, the classic result of Lenstra, Shmoys and Tardos [13] gives a 2-approximation, which remains the best algorithm for the problem. For the problem of minimizing makespan in open shop scheduling, i.e, \( O||C_{\text{max}} \), the best approximation algorithm is given by Schuurman and Woeginger [18], which is a 2-approximation.

### 2 Preliminaries

Throughout this paper, we assume the weights, lengths of jobs (operations) are integers. Let \( T \) be the maximum makespan of any “reasonable” schedule; \( T \) will be polynomially bounded if all job lengths are polynomially bounded. For the sake of simplicity, we first assume that \( T \) is polynomially bounded for all the problems. We shall show how to handle super-polynomial \( T \) for related problems in Section 8, by losing a factor of \( 1 + \epsilon \) in the approximation ratio.
List-Scheduling Algorithms When handling job-precedence constraints, there are two types of list-scheduling algorithms studied in the literature: machine-driven and job-driven list-scheduling algorithms.

In a job-driven list-scheduling algorithm, we construct the final schedule job-by-job, using some specified order of jobs. When considering a job \( j \), we add it to the existing schedule, in a greedy manner; once scheduled, it will not be affected by jobs scheduled after it. In this case, it is possible that a job added later by the algorithm may complete earlier. In general, job-driven list-scheduling algorithms are suitable for the objective of total weighted completion time. To analyze the completion time of a particular job \( j^* \), we can focus on the schedule at the moment the algorithm just scheduled \( j^* \). For example, the 2-approximation for \( 1|\text{prec}|\sum w_j C_j \) in [9] and the 4-approximation algorithm of [14] for \( P|\text{prec}|\sum w_j C_j \) use job-driven list-scheduling algorithms.

In a machine-driven list-scheduling algorithm, we construct a schedule in real-time. As time goes, each idle machine will pick the first valid job in a pre-specified list of jobs to schedule, if it exists. If no such job exists, it remains idle until the next event happens. In general, machine-driven list-scheduling algorithms are suitable for the max-span objective. The classic 2-approximation of Graham [7] for \( P|\text{prec}|C_{\text{max}} \) uses a machine-driven list-scheduling algorithm. With some modifications, machine-driven list-scheduling algorithms can also be used to handle the weighted completion time objective. For example, the \( 3 + 2\sqrt{2} \)-approximation algorithm of [16] for open shop scheduling uses a machine-driven list-scheduling algorithm that respects some manually-defined precedence constraints. The \( O(\log m) \)-approximation of [6] for \( Q|\text{prec}|\sum w_j C_j \) first reduces the \( \sum w_j C_j \) objective to the \( C_{\text{max}} \) objective, and then use a machine-driven list-scheduling algorithm for \( Q|\text{prec}|C_{\text{max}} \).

Organization The rest of the paper is organized as follows. The proofs of Theorems 1.1 to 1.5 are given in Sections 3 to 7 respectively. In Section 8, we show how to handle the case when \( T \) is super-polynomial in \( n \), for related problems.

3 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 on identical machines with job precedence constraints, namely \( P|\text{prec}|\sum w_j C_j \). We first solve our time-index LP relaxation (LP\( P|\text{prec}|\sum w_j C_j \)) and then run the job-driven list-scheduling algorithm of [14] with a random order of jobs.

3.1 Time-Indexed LP Relaxation for \( P|\text{prec}|\sum w_j C_j \)

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 \( j \). It is a folklore result 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 (LP\( P|\text{prec}|\sum w_j C_j \)) for both \( P|\text{prec}|\sum w_j C_j \) and \( P|\text{prec}, p_j = 1|\sum w_j C_j \). Let \( T \) be the makespan of any reasonable schedule. In the LP relaxation, we have a variable \( x_{j,t} \) indicating whether job \( j \) is scheduled in \((t, t - p_j]\), for every \( j \in J \) and \( t \in [T] \). Throughout this section, \( t \)
and \( t' \) are always integers in \([T]\), and \( j, j' \) and \( j^* \) are jobs in \( J \).

\[
\min \sum_j w_j \sum_t x_{j,t} t \quad \text{s.t.} \quad (LP_{\text{prec\&wc}})
\]

\[
\sum_t x_{j,t} = 1 \quad \forall j \quad (1)
\]

\[
\sum_{t < t' + p_{j'}} x_{j',t} \leq \sum_{t < t'} x_{j,t} \quad \forall j, j', t' : j < j' \quad (3)
\]

\[
\sum_{j,t \in [t', t'+p_j]} x_{j,t} \leq m \quad \forall t' \quad (2)
\]

\[
x_{j,t} = 0 \quad \forall j, t < p_j \quad (4)
\]

\[
x_{j,t} \geq 0 \quad \forall j, t \quad (5)
\]

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 (1) requires every job \( j \) of \( J \) to be scheduled. Constraint (2) requires that at every point \( t' \), at most \( m \) jobs are being processed. Constraint (3) requires that for every \( j < j' \) 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 (4)). We require all variables \( x_{j,t} \) to be non-negative (Constraint (5)).

We solve \((LP_{\text{prec\&wc}})\) to obtain \( x \in [0, 1]^{J \times [T]} \). Let \( C_j = \sum_t x_{j,t} \) 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 \( M^\theta_j = C_j - (1 - \theta)p_j \). As we mentioned in the introduction, our algorithm is simply the following: choose \( \theta \) uniformly at random from \((0, 1/2]\), and output the schedule returned by job-driven-list-scheduling\((M^\theta)\) (described in Algorithm 1).

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

**Input:** a vector \( M \in \mathbb{R}_{\geq 0}^J \) used to decide the order of scheduling, s.t. if \( j < j' \), then \( M_j < M_{j'} \)

**Output:** starting and completion time vector \( S, \tilde{C} \in \mathbb{R}_{\geq 0}^J \)

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

We first make a simple observation regarding the \( C \) vector obtained from solving \((LP_{\text{prec\&wc}})\). It follows from the constraints in the LP:

**Claim 3.1.** For every pair of jobs \( j, j' \) such that \( j < j' \), we have \( C_j + p_j \leq C_{j'} \).

**Proof.** \[
C_j + p_j = \sum_{t' < t} x_{j,t'} t' + p_{j'} = \sum_{t', t \leq t'} x_{j,t'} + p_{j'} = \sum_t \left( 1 - \sum_{t' < t} x_{j,t'} \right) + p_{j'} \leq \sum_t \left( 1 - \sum_{t' < t + p_{j'}} x_{j,t'} \right) + p_{j'} = \sum_{t' \geq p_{j'}} (t' - p_{j'}) x_{j,t'} + p_{j'} = \sum_{t} t' x_{j',t} - p_{j'} + p_{j'} = C_{j'}.
\]

The inequality used Constraint (3); some of the equalities used Constraint (1) and (4) and the definitions of \( C_j \) and \( C_{j'} \). \( \square \)

In particular, our analysis does not use the full power of Constraint (3), except for the above claim that 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 (3) for this problem. However, in the algorithm
for the problem with uniform job length (described in Section 4), we do need Constraint (3). To have a unified LP for both problems, we chose to use Constraint (3). Though our algorithm does not use \( x \)-variables, we need them in our analysis (more specifically, in Lemma 3.5).

### 3.2 Analysis

Our analysis is very similar to that in [14]. We fix a job \( j^* \) from now on and we shall upper bound \( \frac{E[\tilde{C}_{j^*}]}{C_{j^*}} \). Notice that once \( j^* \) is scheduled by the algorithm, \( \tilde{C}_{j^*} \) 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 \( t \in (0, T] \) is busy if the congestion of the intervals at \( t \) is \( \theta \) in the schedule (in other words, all the \( m \) machines are being used at \( t \) in the schedule); we say \( t \) 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 \( \tilde{C}_{j^*} \) respectively, w.r.t the final schedule. For a specific \( \theta \in (0, 1/2] \), the techniques in [14] can bound the total length of idle slots by \( \frac{1}{\theta^2} \) and the total length of busy slots by \( \frac{2}{\theta^2} \). 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 \( \tilde{C}_{j^*} \) is at most \( 2\tilde{C}_{j^*} \), which is much better than the bound \( \frac{1}{\theta} \tilde{C}_{j^*} = \infty \) given by directly applying the bound for every \( \theta \). We remark that the \( \frac{2\tilde{C}_{j^*}}{\theta} \) bound for each individual \( \theta \) is tight and thus can not be improved; our improvement comes from considering all \( \theta \)’s together.

**Bounding the Expected Length of Idle Slots** We first bound the total length of busy slots before \( \tilde{C}_{j^*} \), the completion time of job \( j^* \) in the schedule produced by the algorithm. Lemma 3.2 and 3.3 are established in [14] and we include their proofs for completeness.

**Lemma 3.2.** Let \( j \in J \) be a job scheduled before \( j^* \) in the list-scheduling algorithm with \( S_j > 0 \). Then we can find a job \( j' \) such that

- either \( j' < j \) and \( (\tilde{C}_{j'}, S_j] \) is busy,
- or \( M_{j'} \leq M_j, S_{j'} < S_j \) and \((S_{j'}, S_j] \) is busy.

**Proof.** Recall that busy and idle slots are defined w.r.t the final schedule, i.e, the schedule at the moment we just scheduled \( j^* \). We first assume \((S_j - 1, S_j] \) is idle. Then before the iteration for \( j \), the interval \((S_j - 1, S_j + p_j] \) is available for scheduling and thus scheduling \( j \) at \((S_j - 1, S_j + p_j] \) will not violate the congestion constraint. Therefore, it must be the case that \( S_j = \max_{j' < j} \tilde{C}_{j'} \). So, there is a job \( j' < j \) such that \( \tilde{C}_{j'} = S_j \). Since \((\tilde{C}_{j'}, S_j] \) is idle, the first property holds.

If we can assume \((S_j - 1, S_j] \) is busy. Let \((\tau, \tau'] \) be the maximal busy slot that contains \((S_j - 1, S_j] \). If there is a job \( j' < j \) such that \( \tilde{C}_{j'} \geq \tau \), then the first property holds since \((\tilde{C}_{j'}, S_j] \) is busy. So, we can assume that no such \( j' \) exists.

As \( \tau \) is the starting point of a maximal busy slot, there is a job \( j_1 \) with \( S_{j_1} = \tau < S_j \). If \( \tilde{C}_{j_1} < S_j \), then there is a job \( j_2 \) with \( \tilde{C}_{j_1} = S_{j_2} < S_j \), as \((C_{j_1}, C_{j_1} + 1] \) is busy. If \( \tilde{C}_{j_2} < S_j \) then there is a job \( j_3 \) with \( \tilde{C}_{j_2} = S_{j_3} < S_j \). We can repeat this process to find a job \( j_k \) such that \( \tau \leq S_{j_k} < S_j \leq \tilde{C}_{j_k} \). Let \( j' = j_k \); we claim that \( M_{j'} \leq M_j \). Otherwise, \( j \) is considered before \( j' \) in the list scheduling algorithm. At the iteration for \( j \), the interval \((S_{j'}, \tilde{C}_{j}] \) is available. Since we assumed that there are no jobs \( j'' < j \) such that \( \tilde{C}_{j''} \geq \tau \), \( j \) should be started at \( S_{j'} \), a contradiction. Thus, \( M_{j'} \leq M_j \) and \( j' \) satisfies the second property of the lemma.

\( \square \)
Applying Lemma 3.2, we can identify a chain of jobs whose scheduling intervals cover all the idle slots before $j^*$; this can be used to bound the total length of these slots. The following lemma from [14] captures this idea:

**Lemma 3.3.** The total number of idle time slots before $\tilde{C}_{j^*}$ is at most $\frac{C_{j^*}}{1-\theta}$.

**Proof.** Let $j_0 = j^*$; if $S_{j_0} > 0$, we apply Lemma 3.2 for $j = j_0$ to find a job $j'$ and let $j_1 = j'$. If $S_{j_1} > 0$, then we apply the lemma again for $j = j_1$ to find a job $j'$ and let $j_2 = j'$. We can repeat the process until we reach a job $j_k$ with $S_{j_k} = 0$. For convenience, we shall revert the sequence so that $j^* = j_k$ and $S_{j_0} = 0$. Thus, we have found a sequence $j_0, j_1, \cdots, j_k = j^*$ of jobs such that $S_{j_0} = 0$, and for each $\ell = 0, 1, 2, \cdots, k-1$, either (i) $j_{\ell} < j_{\ell+1}$ and $(\tilde{C}_{j_{\ell}}, S_{j_{\ell+1}}]$ is busy, or (ii) $M_{j_{\ell}} \leq M_{j_{\ell+1}}$, $S_{j_{\ell}} < S_{j_{\ell+1}}$ and $(S_{j_{\ell}}, S_{j_{\ell+1}}]$ is busy.

We say $\ell$ is of type-1 if $\ell$ satisfies (i); otherwise, we say $\ell$ is of type-2 ($\ell$ must satisfy (ii)). The interval $(0, S_{j^*}]$ can be broken into $k$ intervals: $(S_{j_0}, S_{j_1}], (S_{j_1}, S_{j_2}], \cdots, (S_{j_{k-1}}, S_{j_k}]$. If some $\ell$ is of type-2, then $(S_{j_{\ell}}, S_{j_{\ell+1}}]$ is busy; if $\ell$ is of type-1, then $(\tilde{C}_{j_{\ell}}, S_{j_{\ell+1}}]$ is busy. Let $L$ be the set of type-1 indices $\ell \in [0, k-1]$. Then, all the idle slots in $(0, S_{j^*}]$ are contained in $\bigcup_{\ell \in L} (S_{j_{\ell}}, \tilde{C}_{j_{\ell}}]$. With this observation, we can bound the total length of idle slots in $(0, S_{j^*}]$ by

$$\sum_{\ell \in L} p_{j_{\ell}} \leq \sum_{\ell \in L} \frac{1}{1-\theta}(M_{j_{\ell+1}}^\theta - M_{j_{\ell}}^\theta) \leq \frac{1}{1-\theta} \sum_{\ell=0}^{k-1} (M_{j_{\ell+1}}^\theta - M_{j_{\ell}}^\theta) \leq \frac{M^\theta_{j^*}}{1-\theta} = \frac{C_{j^*}}{1-\theta} - p_{j^*}.$$  

The first inequality holds since $p_{j_{\ell}} = \frac{1}{1-\theta}(C_{j_{\ell}} - M_{j_{\ell}}^\theta) \leq \frac{1}{1-\theta}(C_{j_{\ell+1}} - p_{j_{\ell+1}} - M_{j_{\ell}}^\theta) \leq \frac{1}{1-\theta}(M_{j_{\ell+1}}^\theta - M_{j_{\ell}}^\theta)$, due to Claim 3.1. The second inequality is by the fact that $M_{j_{\ell}} \leq M_{j_{\ell+1}}$ for every $\ell \in [0, k-1]$ and the third inequality is by $j_k = j^*$ and $M_{j_0}^\theta \geq 0$. The equality is by the definition of $M_{j^*}^\theta$. Thus, the total length of idle slots in $(0, \tilde{C}_{j^*}]$ is at most $\frac{C_{j^*}}{1-\theta}$.

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

$$\int_{\theta=0}^{1/2} \frac{C_{j^*}}{1-\theta} 2d\theta = \left(2 \ln \frac{1}{1-\theta}\right)_{\theta=0}^{1/2} C_{j^*} = (2 \ln 2) C_{j^*}. \tag{6}$$

**Bounding the Expected Length of Busy Slots** We now proceed to bound the total length of busy slots before $\tilde{C}_{j^*}$. This is the key leading to our improved approximation ratio. For every $\theta \in [0, 1/2)$, let $J_\theta = \{ j : M_j^\theta \leq \tilde{C}_{j^*} \}$. Thus, if $\theta < \theta'$, we have $J_\theta \supseteq J_{\theta'}$. For every $\theta \in [0, 1/2]$ and $j \in J_\theta$, define $\theta_j = \sup \{ \theta \in [0, 1/2] : j \in J_\theta \}$; this is well-defined since $j \in J_\theta$. 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 3.4.** For a fixed $\theta \in (0, 1/2]$, the total length of busy slots before $\tilde{C}_{j^*}$ is at most $\frac{1}{m} p(J_\theta)$.

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

$$\frac{1}{m} \sum_{j : M_j^\theta \leq \tilde{C}_{j^*}} p_j \leq \frac{1}{m} \sum_{j : M_j^\theta \leq \tilde{C}_{j^*}} p_j = \frac{1}{m} p(J_\theta).$$

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

**Lemma 3.5.** $\int_{\theta=0}^{1/2} p(J_\theta) d\theta \leq mC_{j^*}$. 

11
Proof. Notice that we have
\[\int_{\theta=0}^{1/2} p(J_\theta) d\theta = \int_{\theta=0}^{1/2} \sum_{j \in J_\theta} p_j 1_{j \in J_\theta} d\theta = \sum_{j \in J_\theta} p_j \int_{\theta=0}^{1/2} 1_{j \in J_\theta} d\theta = \sum_{j \in J_\theta} \theta_j p_j.\]
Thus, it suffices to prove that \(\sum_{j \in J_\theta} \theta_j p_j \leq mC_{j^*}\). 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 \(\sum_{j} x_{j,t}(t - p_j + \theta_j p_j) = C_j - (1 - \theta_j)p_j = M_{\theta_j}^j \leq C_{j^*}\), \(\sum_{j} x_{j,t} = 1\), 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 \((0, T)\), the total height of all rectangles covering \(t\) is at most \(m\), by Constraint (2), 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 3.4 and Lemma 3.5, the expected length of busy time slots before \(\tilde{C}_{j^*}\) is at most
\[\int_{\theta=0}^{1/2} \frac{p(J_\theta)}{m} 2d\theta = \frac{2}{m} \int_{\theta=0}^{1/2} p(J_\theta) d\theta \leq 2C_{j^*}.\] (7)

Thus, by Inequalities (6) and (7), we have
\[E[\tilde{C}_{j^*}] \leq (2 + 2\ln 2)C_{j^*} + 2C_{j^*} = (2 + 2\ln 2)C_{j^*}.\]
Thus, we have proved the \(2 + 2\ln 2 + \epsilon \leq (3.387 + \epsilon)\)-approximation ratio for our algorithm, finishing the proof of Theorem 1.1.

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

In this section, we give our algorithm for \(P[\text{prec}, p_j = 1] \sum w_j C_j\). Again, we solve (LP\(_{P[\text{prec}, wC]}\)) to obtain \(x\); define \(C_j = \sum x_{j,t}\) for every \(j \in J\). For this special case, we define the random \(M\)-vector differently. In particular, it will depend on the values of \(x\) variables. For every \(j \in J\) and \(\theta \in (0, 1]\), define \(M_\theta^j\) to be the minimum \(t\) such that \(\sum_{t=1}^t x_{j,t} \geq \theta\). Notice that \(C_j = \int_{\theta=0}^1 M_\theta^j d\theta\). Our algorithm for \(P[\text{prec}, p_j = 1] \sum w_j C_j\) chooses \(\theta\) uniformly at random from \((0, 1]\), and then call job-driven-list-scheduling(\(M_\theta^j\)) 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 \(j_1 < j_2 < j_3 < \cdots < j_a = j\). Thus, \(a_j\) is the “depth” of \(j\) in the dependence graph. Then,

Claim 4.1. For every \(j \in J\) and \(t < a_j\), we have \(x_{j,t} = 0\).

Proof. By the definition of \(a_j\), there is a sequence of \(a := a_j\) jobs \(j_1 < j_2 < \cdots < j_a\) such that \(j_a = j\). Let \(t_a = t\). If \(x_{j_a,t_a} > 0\), then by Constraint (3), there is an \(t_{a-1} \leq t_a - 1\) such that \(x_{j_{a-1}, t_{a-1}} > 0\). Then, there is an \(t_{a-2} \leq t_{a-1} - 1 \leq t_a - 2\) such that \(x_{j_{a-2}, t_{a-2}} > 0\). Repeating this process, there will be an \(t_1 \leq t_a - (a - 1)\) such that \(x_{j_1,t_1} > 0\). This contradicts the fact that \(t_a = t \leq a - 1\). \(\square\)

\(^2\)Here, we use mass center for the horizontal coordinate of the mass center, since we are not concerned with the vertical positions of rectangles.
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 $\mathbb{E}[\tilde{C}_{j^*}]/C_{j^*}$ (recall that $\tilde{C}_{j^*}$ 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 $\tilde{C}_{j^*}$ 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 $\theta$. In particular, since all jobs have uniform length, we can have an $a_{j^*}$ bound on the total length of idle slots before $\tilde{C}_{j^*}$.

**Lemma 4.2.** \( \tilde{C}_{j^*} \leq \frac{1}{m} \left| \left\{ j : M_j^\theta \leq M_{j^*}^\theta \right\} \right| + a_{j^*}. \)

**Proof.** The total length of busy slots in \((0, \tilde{C}_{j^*})\) is at most \(1/m\) times the total number of jobs scheduled so far, which is at most \(1/m \left| \left\{ j : M_j^\theta \leq M_{j^*}^\theta \right\} \right| \).

We now bound the total length of idle slots in \((0, \tilde{C}_{j^*})\). We start from $j_1 = j^*$. For every $\ell = 1, 2, 3 \cdots$, let $j_{\ell+1} \prec j_\ell$ be the job that is scheduled the last in the final schedule; if $j_{\ell+1}$ does not exist, then we let $a = \ell$ and break the loop. Thus, we constructed a chain $j_a \prec j_{a-1} \prec j_{a-2} \cdots \prec j_1 = j^*$ of jobs. By the definition of $a_{j^*}$, we have $a \leq a_{j^*}$.

We shall show that for every idle unit slot \((t-1, t]\) in \((0, \tilde{C}_{j^*})\), some job in the chain is scheduled in \((t-1, t]\). Assume otherwise; let $\ell \in [a]$ be the largest index such that $\tilde{C}_{j_\ell} > t$ (\(\ell\) exists since $j_1 = j^*$ is scheduled in \((\tilde{C}_{j^*} - 1, \tilde{C}_{j^*}]\) and $t < \tilde{C}_{j^*}$). Then either $j_{\ell+1}$ does not exist, or $\tilde{C}_{j_{\ell+1}} \leq t - 1$. Consider the iteration when $j_\ell$ is considered in the list scheduling algorithm. Before the iteration, \((t-1, t]\) is available for scheduling. Since $j_\ell$ is scheduled after $t$, there must be a job $j' \prec j_\ell$ such that $\tilde{C}_{j'} \geq t$. Thus, we would have set $j_{\ell+1} = j'$, a contradiction.

Thus, the idle slots before $\tilde{C}_{j^*}$ are covered by the scheduling intervals of jobs in the chain. So, the total number of idle slots before $\tilde{C}_{j^*}$ is at most $a \leq a_{j^*}$. Overall, we have $\tilde{C}_{j^*} \leq \frac{1}{m} \left| \left\{ j : M_j^\theta \leq M_{j^*}^\theta \right\} \right| + a_{j^*}. \quad \Box$

We shall use $g(\theta) = M_j^\theta$, for every $\theta \in (0, 1]$. Notice that $C_{j^*} = \int_{\theta=0}^1 g(\theta)d\theta$. 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 4.1, we have $g(0) \geq a_{j^*}$. For every $j \in J$ and $\theta \in [0, 1]$, define $h_j(\theta) := \sum_{t=1}^\infty x_{j,t}$. 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 4.2, we have $\tilde{C}_{j^*} \leq g(0) + \frac{1}{m} \sum_{j \in J} 1_{h_j(\theta) \geq \theta}$. Thus, we can bound $\mathbb{E}[\tilde{C}_{j^*}]/C_{j^*}$ by the superior of

$$\frac{g(0) + \frac{1}{m} \sum_{j \in J} \int_{\theta=0}^1 1_{h_j(\theta) \geq \theta}d\theta}{\int_{\theta=0}^1 g(\theta)d\theta} \quad (8)$$

subject to

- $g : [0, 1] \to [0, \infty)$ is piecewise linear, left-continuous and non-decreasing, \quad \( (8.1) \)
- $\forall j \in J, h_j : [0, 1] \to [0, 1]$ is piecewise linear, left-continuous and non-decreasing, \quad \( (8.2) \)
- $\sum_{j \in J} h_j(\theta) \leq mg(\theta), \ \forall \theta \in [0, 1].$ \quad \( (8.3) \)
To recover the 3-approximation, we know that \( g(0)/\int_{\theta=0}^{1} g(\theta) d\theta \leq 1 \); this corresponds to the fact \( a_{j^*} \leq \overline{C}_{j^*} \). As can be seen from our analysis later, we have \( \frac{1}{m} \sum_{j \in J} \int_{\theta=0}^{1} 1_{h_j(\theta) \geq \theta} d\theta / \left( \int_{\theta=0}^{1} g(\theta) d\theta \right) \leq 2 \). The tight factor 2 can be achieved when \( h_j(\theta) = \theta \) for every \( j \in J \) and \( \theta \in [0, 1] \) and \( g(\theta) = \theta \). This corresponds to the following case: by the time \( \theta \)-fraction of job \( j^* \) has been finished, exactly \( \theta \) fraction of every job \( j \in J \) has been finished. 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 \( \theta \). This is where we obtain our improved approximation ratio.

**Computing the superior of** (8) **We now compute the superior of** (8) **subject to Constraints (8.1), (8.2) and (8.3). The functions** \( g \) **and** \( h_j \)'s **defined may have other stronger properties (e.g., they are piecewise constant functions): however in the process, we only focus on the properties described above. In the following,** \( j \) **in a summation is over all jobs in** \( J \). **First, we can add the following constraint without changing the superior of** (8). **Let** \( g' \) **be a constant function over** \([0, \infty)\) **piecewise-linear, left-continuous and non-decreasing.**

\[
\exists \theta^* \in [0, 1], \text{ such that } g(\theta) = g(0), \forall \theta \in [0, \theta^*], \text{ and } g(\theta) = \frac{1}{m} \sum_{j} h_j(\theta), \forall \theta \in (\theta^*, 1].
\]

To see this, we take any \( g \) and \( \{h_j\}_j \) satisfying Constraints (8.1) to (8.3). Let \( g'(\theta) = \max \left\{ g(0), \frac{1}{m} \sum_{j} h_j(\theta) \right\} \) for every \( \theta \in [0, 1] \). Then \( g' : [0, 1] \rightarrow [0, \infty) \) is piecewise-linear, left-continuous and non-decreasing because of Property (8.2); so \( g' \) satisfies Property (8.1). Obviously \( g' \) satisfies Property (8.3). We define \( \theta^* = \sup \{ \theta \in [0, 1] : g'(\theta) = g'(0) \} \). Since \( g' \) is left-continuous and monotone non-decreasing, we have \( g'(\theta) = g'(0) \geq \frac{1}{m} \sum_{j} h_j(\theta) \) for every \( \theta \in [0, \theta^*] \) and \( g'(\theta) = \frac{1}{m} \sum_{j} h_j(\theta) > g'(0) \) for every \( \theta \in (\theta^*, 1] \). Thus, \( g' \) satisfies Constraint (9). Changing \( g \) to \( g' \) will not decrease (8): the numerator does not change and the denominator can only decrease since \( g(\theta) \geq \max \left\{ g(0), \frac{1}{m} \sum_{j} h_j(\theta) \right\} = g'(\theta) \) for every \( \theta \in [0, 1] \). Thus, we can impose Constraint (9), without changing the superior of (8).

Then, we can make the following constraint on \( \{h_j\}_j \):

\[
\text{For every } j \in J, \text{ } h_j \text{ is a constant over } [0, \theta^*].
\]

For every \( j \), we define \( h'_j(\theta) = h_j(\theta^*) \) if \( \theta \in [0, \theta^*] \) and \( h'_j(\theta) = h_j(\theta) \) if \( \theta \in (\theta^*, 1] \). Then each \( h'_j \) satisfies Constraint (8.2). Moreover, \( \{h'_j\}_j \) satisfies Constraint (8.3) since \( \sum_{j} h'_j(\theta) = \sum_{j} h_j(\theta^*) \leq mg(\theta^*) = mg(\theta) \) for every \( \theta \in [0, \theta^*] \). Moreover, if we change \( h_j \) to \( h'_j \) for every \( j \in J \), the denominator of (8) does not change and the numerator can only increase.

Finally, we can assume

\[
\sum_{j} h_j(\theta) = mg(\theta), \quad \forall \theta \in [0, \theta^*].
\]

Notice that the functions \( \{h_j\}_j \) and \( g \) are constant functions over \([0, \theta^*]\) and \( \sum_{j} h_j(\theta) = mg(\theta) \) for \( \theta \in (\theta^*, 1] \). If the constraint is not satisfied, we can find some \( j \) such that \( h_j \) is not right-continuous at \( \theta^* \). Then, we increase \( h_j(\theta) \) simultaneously for all \( \theta \in [0, \theta^*] \) until the constraint is satisfied or \( h_j \) becomes right-continuous at \( \theta^* \). This process can be repeated until the constraint becomes satisfied.

Thus, with all the constraints, we have \( g(\theta) = \frac{1}{m} \sum_{j} h_j(\theta) \) for every \( \theta \in [0, 1] \). So, (8) becomes

\[
\frac{1}{m} \sum_{j} h_j(0) + \frac{1}{m} \sum_{j} \int_{\theta=0}^{1} 1_{h_j(\theta) \geq \theta} d\theta = \frac{\sum_{j} \left( h_j(0) + \int_{\theta=0}^{1} 1_{h_j(\theta) \geq \theta} d\theta \right)}{\sum_{j} \int_{\theta=0}^{1} h_j(\theta) d\theta}.
\]
To bound the quantity, it suffices to upper bound
\[
\sup_h \frac{h(0) + \int_{0}^{1} 1_{h(\theta) \geq \theta} \, d\theta}{\int_{0}^{1} h(\theta) \, d\theta},
\]
(12)
where the superior is over all piecewise-linear, left-continuous and monotone non-decreasing functions \( h : [0, 1] \to [0, 1] \).

**Claim 4.3.** Let \( \alpha = h(0) \) and \( \beta = \int_{0}^{1} 1_{h(\theta) \geq \theta} \, d\theta \). Then \( 0 \leq \alpha \leq \beta \leq 1 \) and
\[
\int_{0}^{1} h(\theta) \, d\theta \geq \frac{\alpha^2}{2} + \beta - \frac{\beta^2}{2}.
\]

**Proof.** \( \beta \geq \alpha \) because \( h(\theta) \geq \theta \) holds for every \( \theta \in [0, \alpha] \). To prove the second part, we can assume that \( h(\theta) = \alpha \) if \( \theta \in [0, \alpha] \); otherwise, we can change \( h(\theta) \) to \( \alpha \) for every \( \theta \in (0, \alpha] \); this does not change \( \int_{0}^{1} 1_{h(\theta) \geq \theta} \, d\theta \) and can only decrease \( \int_{0}^{1} h(\theta) \, d\theta \). Also, we can assume that \( h(\theta) \leq \theta \) for every \( \theta \in [\alpha, 1] \). Otherwise, for every \( \theta \) such that \( h(\theta) > \theta \), we decrease \( h(\theta) \) to \( \theta \). This does not change \( \int_{0}^{1} 1_{h(\theta) \geq \theta} \, d\theta \) and can only decrease \( \int_{0}^{1} h(\theta) \, d\theta \).

\[
\int_{0}^{1} h(\theta) \, d\theta = \int_{0}^{1} \left( \theta - (\theta - h(\theta)) \right) \, d\theta = \frac{1}{2} + \frac{\alpha^2}{2} - \int_{\theta=0}^{1} (\theta - h(\theta)) \, d\theta.
\]

For interval \([a, b] \subseteq [\theta^*, 1]\) such that \( h(\alpha) = a \), we have that \( \int_{\theta=a}^{b} (\theta - h(\theta)) \, d\theta \leq \frac{(b-a)^2}{2} \). Since \( \int_{\theta=\alpha}^{1} 1_{h(\theta) < \theta} \leq 1 - \beta \), we have that \( \int_{\theta=\alpha}^{1} (\theta - h(\theta)) \leq (1 - \beta)^2 \). Thus, the above quantity is at least
\[
\frac{1}{2} + \frac{\alpha^2}{2} - (1 - \beta)^2 = \frac{\alpha^2}{2} + \beta - \frac{\beta^2}{2}.
\]

Thus, (12) is at most \( \sup_{0 \leq \alpha \leq \beta \leq 1} \frac{\alpha + \beta}{\alpha^2 + \beta - \frac{\beta^2}{2}} = \sup_{0 \leq \alpha \leq \beta \leq 1} \frac{2(1+\alpha)}{2\alpha - (\alpha + \beta)(\beta - \alpha)} \). Scaling both \( \alpha \) and \( \beta \) up can only increase \( \frac{2(1+\alpha)}{2\alpha - (\alpha + \beta)(\beta - \alpha)} \). Thus, we can assume \( \beta = 1 \) and (12) becomes \( \sup_{\alpha \in [0, 1]} \frac{2(1+\alpha)}{1+2\alpha^2} \). For \( \alpha \in [0, 1] \), \( \frac{2(1+\alpha)}{1+2\alpha^2} \) is maximized at \( \alpha^* = \sqrt{2} - 1 \) and the maximum value is \( \frac{2(1+\sqrt{2}-1)}{1+2(\sqrt{2}-1)^2} = \sqrt{2} + 1 \). This finishes the proof of the \( (\sqrt{2} + 1) \)-approximation for \( P_{\text{prec}, p_j = 1}[\sum_j w_j C_j] \) (Theorem 1.2).

## 5 Open Job Shop Scheduling

In this section, we give our improved approximation algorithm for \( O||\sum_j w_j C_j \). Our algorithm is similar to that of Queyranne and Sviridenko [16], in which we solve some LP relaxation, manually define some precedence constraints according to the LP solution, and then run a machine-driven list scheduling algorithm that respects the precedence constraints. We apply a job-by-job analysis. If the last operation for a job \( j^* \) in the constructed schedule, then we analyze separately the total lengths of busy and idle slots on machine \( i \) before the completion of \( O_{i,j^*} \). To improve the approximation ratio, we use a different definition of precedence constraints, such that if a job \( j \) has shorter total length of operations, it is more likely to be preceded by other jobs. This will not hurt job \( j \) itself but will reduce the total length of busy slots for other jobs. Our analysis requires our stronger time-indexed LP relaxation which we describe now.
5.1 Time-Indexed LP for $O \parallel \sum_j w_j C_j$

Let $L_j = \sum_{i \in M} p_{i,j}$ be the total length of all operations for job $j$. Let $T = \sum_{j \in J} L_j$ be the trivial upper bound on the makespan of any reasonable schedule. We use $i$ to index machines, $j$ to index jobs and $t$ and $t'$ to index integers in $[T]$. We use $x_{i,j,t}$ to indicate whether $O_{i,j}$ is scheduled in $(t - p_{i,j}, t]$ or not. $y_{j,t}$ will indicate whether the job $j$ ends at time $t$ or not. The LP relaxation is given in (LP$_{O \parallel |wC}$).

$$\begin{align*}
\text{min} & \quad \sum_j w_j \sum_t y_{j,t}t \\
\text{s.t.} & \quad \sum_t x_{i,j,t} = 1 \quad \forall i, j \quad \text{(13)} \\
& \quad \sum_{t \leq t'} x_{i,j,t} \leq 1 \quad \forall i, t' \quad \text{(14)} \\
& \quad \sum_{i, t \in [t', t' + p_{i,j}]} x_{i,j,t} \leq 1 \quad \forall j, t' \quad \text{(15)} \\
& \quad \sum_t y_{j,t} = 1 \quad \forall j \quad \text{(16)} \\
& \quad \sum_{t \leq t'} y_{j,t} \leq \sum_{t \leq t'} x_{i,j,t} \quad \forall i, j, t' \quad \text{(17)} \\
& \quad y_{j,t} = 0 \quad \forall j, t < L_j \quad \text{(18)} \\
& \quad x_{i,j,t}, y_{j,t} \geq 0 \quad \forall i, j, t \quad \text{(19)}
\end{align*}$$

In the above LP relaxation, the objective to minimize is the total weighted completion time of all jobs. Constraint (13) requires operation $O_{i,j}$ to be scheduled. Constraint (14) requires that for every machine $i$ at any time point $t'$, at most 1 operation is being processed. Constraint (15) requires that for every job $j$ at any time point $t'$, at most 1 operation for $j$ is being processed. Constraint (16) says that every job $j$ must be completed at some time $t$. Constraint (17) says that job $j$ completes at or before time $t'$ only if $O_{i,j}$ completes at or before time $t'$. Constraint (18) says that job $j$ can not complete before time $L_j$. Constraint (19) requires all variables to be non-negative.

We solve (LP$_{O \parallel |wC}$) to obtain an optimum solution $(x, y)$ to the LP. Define $C_j = \sum_t y_{j,t}t$ to be the completion time of $j$ indicated by the LP solution $(x, y)$. Thus, the value of the LP is $\sum_j w_j C_j$.

5.2 Rounding the LP Solution

Let $\theta < 1$ be a parameter to be decided later. Notice that there are no precedence constraints in the definition of the open shop scheduling problem. As in [16], we shall manually define some precedence constraints according to the completion time vector $C$ given by the LP solution. Different from the one in [16], our definition takes the job lengths $\{L_j\}_j$ into account:

**Definition 5.1.** Define $j < j'$ if $C_j < C_{j'} - \theta L_{j'}$; define $j' > j$ iff $j < j'$.

If job $j'$ has $L_{j'} = C_{j'}$, then the condition in the definition becomes $C_j < (1 - \theta)C_{j'}$, which coincides with the definition in [16] with $\gamma = 1/(1 - \theta)$. If $L_{j'}$ is smaller, $j'$ will have more predecessors according to our definition. For each $j^*$, we analyze the completion time of $O_{i,j^*}$, where $O_{i,j^*}$ is the operation of $j^*$ that is completed the last. We bound the total lengths of idle and busy time slots on $i$ before $O_{i,j^*}$ separately. With our definition of precedence constraints, one can give the same bound on the total length of idle slots as in [16] (see Lemma 5.2). On the other hand, the richer family of precedence constraints make more jobs become successors of $j^*$, allowing us to give smaller factor on the total length of busy slots. This analysis requires our stronger time-indexed LP relaxation.

With the definition of the precedence constraints, we can now run the machine-driven list-scheduling algorithm in [16]. The algorithm constructs the schedule in real time. At time 0
or when an operation is completed, for every idle machine $i$, we will attempt to schedule an unprocessed operation $O_{i,j}$ on $i$ subject to two constraints: (i) for every predecessor $j'$ of $j$, $O_{i,j'}$ has been completed and (ii) no other operation for $j$ is being processed. If no such operation $O_{i,j}$ exists, machine $i$ remains idle until the next operation completes. The algorithm terminates when all operations are competed. For every $j \in J$, let $\bar{C}_j$ be the completion time of $j$, in the schedule generated by the algorithm.

### 5.3 Analysis of Rounding Algorithm

Fix a job $j^*$ and let $O_{i,j^*}$ be the operation for job $j^*$ that is completed the last in the output schedule. Our analysis is job-by-job: for every $j^* \in J$, we shall bound $\bar{C}_j^*/C_j^*$. Let $B$ and $I$ respectively be the lengths of total busy and idle slots on machine $i$ before $\bar{C}_j^*$; so $\bar{C}_j^* = B + I$. The upper bound on $I$ was established in [16] for their definition of precedence constraints; we show that the bound still holds with our new definition of precedence constraints.

**Lemma 5.2.** $I \leq C_j^*/\theta$.

**Proof.** We define a sequence $j_0 = j^*, j_1, j_2, \cdots, j_k$ of jobs such that $j_k \prec j_{k-1} \prec j_{k-2} \cdots \prec j_0$ as follows. Let $j_0 = j^*$. For each $\ell = 0, 1, 2, 3, \cdots$, let $j_{\ell+1}$ be the job $j \prec j_{\ell}$ such that $O_{i,j}$ is scheduled the latest; if there is no such job, we let $k = \ell$ and compete the definition of the sequence.

We claim that the total length of idle slots between the execution of $O_{i,j\ell+1}$ and $O_{i,j\ell}$ is at most $L_{j\ell}$. By the way we select $j_{\ell+1}$, there is no $O_{i,j}$ with $j \prec j_{\ell}$ that is scheduled between $O_{i,j\ell+1}$ and $O_{i,j\ell}$. Thus, after we completed the operation $O_{i,j\ell+1}$ in the list-scheduling algorithm, there are no precedence constraints preventing the scheduling of $O_{i,j\ell}$. Thus, at any idle time point between the completion time of $O_{i,j\ell+1}$ and starting time of $O_{i,j\ell}$, the job $j_{\ell}$ is being processed on some other machine. Thus, the total length of idle slots between the completion time of $O_{i,j\ell+1}$ and the starting time of $O_{i,j\ell}$ is at most $\sum_{i' \neq i} p_{i',j\ell} \leq L_{j\ell}$. Similarly, the total length of idle slots before $O_{i,jk}$ is at Most $L_{jk}$. Thus, the total length $I$ of idle slots on machine $i$ before $\bar{C}_j^*$ is at most $\sum_{\ell=0}^{k} L_{j\ell}$.

Recall that $j_{\ell+1} \prec j_{\ell}$ for every $\ell = 0, 1, \cdots, k - 1$; i.e, $C_{j\ell+1} < C_{j\ell} - \theta L_{j\ell}$. This implies $L_{j\ell} < \frac{1}{\theta}(C_{j\ell} - C_{j\ell+1})$ for every $\ell = 0, 1, 2, \cdots, k - 1$.

$$I \leq \sum_{\ell=0}^{k} L_{j\ell} \leq \sum_{\ell=0}^{k-1} \frac{1}{\theta}(C_{j\ell} - C_{j\ell+1}) + \frac{1}{\theta} C_{j_0} = \frac{1}{\theta} C_j^*.$$

### Intuition for Improved Bound on $B$

Now we proceed to bound the total length $B$ of busy time slots on $i$ before the completion of $O_{i,j^*}$. To deliver the idea behind our improvement, we first recover the $2C_j^*/(1 - \theta)$ bound of [16], using our time-indexed LP relaxation. We view the $x_{i,j,t}$ variable as a rectangle of height $x_{i,j,t}$ with horizontal span being $(t - p_{i,j}, t]$.

Any operation $O_{i,j}$ scheduled before $\bar{C}_j^*$ must have $j \neq j^*$. The total length of all busy time slots on $i$ before the completion of $O_{i,j^*}$ is at most $\sum_{j \neq j^*} p_{i,j}$. For any job $j \neq j^*$, we have $C_j - \theta L_j \leq C_j^*$, implying $C_j \leq C_j^*/(1 - \theta)$. (Notice that $L_j \leq C_j$.) This means $\sum_{t} b_{j,t} t \leq C_j^*/(1 - \theta)$, which implies $\sum_{t} x_{i,j,t} t \leq C_j^*/(1 - \theta)$ by Constraints from the LP. Thus, the mass center\(^3\) of all rectangles for operation $O_{i,j}$ is at most $C_j^*/(1 - \theta)$. Considering all such jobs $j$ together, the mass center of all rectangles for all operations $\{O_{i,j}\}_{j \neq j^*}$ is at most $C_j^*/(1 - \theta)$. Since the total height of rectangles on machine $i$ covering any time point is at least $1$, the total area of these rectangles is at most $2C_j^*/(1 - \theta)$.

\(^3\)Again, we use mass center for the horizontal position of the mass center.
To improve the $2/(1 - \theta)$ factor, we notice that the above analysis is tight only when most jobs $j \not\approx j^*$ has $L_j \approx C_j \approx C_{j^*}/(1 - \theta)$, $(y_{j,t})_t \approx (x_{i,j,t})_t$ and $p_{i,j} \ll C_{j^*}$. Moreover, the rectangles for the jobs $j \not\approx j^*$ should occupy most space in $(0, 2C_{j^*}/(1 - \theta))]$. Then Constraint (18) plays an important role in deriving a contradiction: most jobs $j \not\approx j^*$ can not complete before $L_j \approx C_{j^*}/(1 - \theta)$ and thus they can not efficiently utilize the space before time $C_{j^*}/(1 - \theta)$.

Formally, we shall give every time point $t$ a resource scalar $f'(t)$, for some monotone non-increasing function $f'$. If a job $j$ completes at $t$ with $y_{j,t}$ fraction, then it uses $y_{i,j}p_{i,j}f'(t)$ units of resource. One can then show that the total resource used by all clients $j$ is at most $\sum_t f'(t)$. We define $f'$ such that for every $j \not\approx j^*$, the total resource used by $j$ is at least $p_{i,j}$. Then, this will bound $B$ by $\sum_t f'(t)$. We remark that to recover the $2C_{j^*}/(1 - \theta)$-upper bound on $B$ using this framework, we can define $f'(t) = \max\left\{0, 2 - \frac{(1 - \theta)f(t)}{C_{j^*}}\right\}$. We give the improved approximation ratio by defining a better $f'$. The proof of the next lemma incorporates this idea; in the lemma, $f'$ is the continuous version of $f$, with the domain scaled by $1/C_{j^*}$.

**Lemma 5.3.** Let $\theta \in [0, 1)$ and $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ be a monotone non-increasing function such that,

\[(5.3a)\text{ for every two real numbers } a, b \text{ with } 0 < a \leq \frac{1}{1 - \theta} < b \text{ and } \alpha = \frac{b - a - 1}{b - a}, \text{ we have } \alpha f(a) + (1 - \alpha)f(b) \geq 1.\]

Then, we have

\[
\sum_{j : C_j - \theta L_j \leq C_{j^*}} p_{i,j} \leq C_{j^*} \int_{\tau = 0}^{\infty} f(\tau)d\tau. \tag{20}
\]

**Proof.** Notice that $a \leq \frac{1}{1 - \theta} < b$ implies $a \leq \theta a + 1 = \theta a + (1 - \theta)\frac{1}{1 - \theta} < \theta b + (1 - \theta)b = b$, which implies $\alpha \in [0, 1]$.

For simplicity, for every integer $t \in [T]$, we let $f'(t) = f\left(\frac{t}{C_{j^*}}\right)$; let $f'(T + 1) = 0$. Obviously $f'$ is monotone non-increasing. We define the resource used by any job $j \in J$ to be $p_{i,j} \sum_t y_{j,t}f'(t)$. Then the total resource used by all jobs $j \in J$ is

\[
\sum_j p_{i,j} \sum_t y_{j,t}f'(t) = \sum_j p_{i,j} \sum_{t' \leq t} (f'(t') - f'(t' + 1)) \sum_{t' \leq t} y_{j,t} \\
\leq \sum_j p_{i,j} \sum_{t' \leq t} (f'(t') - f'(t' + 1)) \sum_{t' \leq t} x_{i,j,t} = \sum_j p_{i,j} \sum_{t' \leq t} x_{i,j,t}f'(t) \\
\leq \sum_j p_{i,j} \sum_{t' \leq t} x_{i,j,t}f'(t) = \sum_{t' \leq t} f'(t') \sum_j p_{i,j} \sum_{t' \leq t} x_{i,j,t}f'(t') \\
\leq \sum_{t'} f'(t') \sum_j p_{i,j} \sum_{t' \leq t} x_{i,j,t}f'(t') = \sum_{t' \leq t} f'(t') \sum_j p_{i,j} \sum_{t' \leq t} x_{i,j,t}f'(t') \\
\leq \sum_{t'} f'(t') \sum_j p_{i,j} \sum_{t' \leq t} x_{i,j,t}f'(t') = \int_{\tau = 0}^{\infty} f\left(\frac{\tau}{C_{j^*}}\right) d\tau = C_{j^*} \int_{\tau = 0}^{\infty} f(\tau)d\tau. \tag{21}
\]

The first inequality is by Constraint (17) and monotonicity of $f'$, the second inequality is by monotonicity of $f'$, and the third inequality is by Constraint (14) and the last one is by monotonicity of $f$.

The right side of (21), which is exactly that of (20), is an upper bound on the total resource of all jobs. Thus, it suffices to prove for any job $j$ with $C_j - \theta L_j \leq C_{j^*}$, $j$ uses at least $p_{i,j}$ amount of resource: if this holds, the left-side of (20) is at most the total resource used by all jobs $j$, which is at most $C_{j^*} \int_{\tau = 0}^{\infty} f(\tau)d\tau$.

Thus from now on we fix a job $j$ with $C_j - \theta L_j \leq C_{j^*}$. It suffices to prove that $\sum_t y_{j,t}f'(t) \geq 1$. We consider the following LP with variables $\{z_t\}_{t \geq L_j}$:

\[
\min \sum_{t \geq L_j} z_tf'(t), \quad \text{s.t.}
\]
By Lemma 5.3, we have
\[ \sum_{t \geq L_j} z_t = 1; \quad \sum_{t \geq L_j} z_t t = C_j; \quad z_t \geq 0, \quad \forall t \geq L_j. \]

By letting \( z_t = y_{j,t} \) for every \( t \geq L_j \), all the constraints of the LP are satisfied: the first constraint holds due to Constraints (16) and (18) and the second constraint follows by the definition of \( C_j \) and Constraint (18).

Thus, the value of the LP is at most \( \sum_t y_{j,t} f'(t) \). Any basic solution of the LP has at most 2 non-zero \( z \) values. Let \( a' \) and \( b' \), \( L_j \leq a' < b' \), be the two indices with positive \( z \) values. (If there is only one \( a' \) with \( z_{a'} = 1 \), then we let \( b' > a' \) be arbitrary.) Let \( \alpha' = z_{a'} \); thus \( \alpha' a' + (1 - \alpha') b' = C_j \) and \( \alpha' f'(a') + (1 - \alpha') f'(b') \leq \sum_t y_{j,t} f'(t) \). Since \( C_j - \theta L_j \leq C_{j^*} \), we have \( \alpha' a' + (1 - \alpha') b' \leq C_{j^*} + \theta L_j \leq C_{j^*} + \theta a' \). This implies \( a' \leq C_{j^*} + \theta a' \), which is \( a' \leq \frac{1}{1 - \theta} C_{j^*} \); this also implies \( \alpha' \geq \frac{b' - \theta a' - C_{j^*}}{b - a} \).

If \( b' > \frac{1}{1 - \theta} C_{j^*} \), then we let \( a = \frac{a'}{C_{j^*}}, b = \frac{b'}{C_{j^*}}, \) and \( \alpha = \frac{b' - \theta a' - C_{j^*}}{b - a} \). So, we have \( a \leq \frac{1}{1 - \theta} < b \) and \( \alpha' \geq \alpha \in [0, 1) \). Thus, by Property (5.3a) and the fact that \( f' \) is monotone non-increasing, we have \( \sum_t y_{j,t} f'(t) \geq \alpha' f'(a') + (1 - \alpha') f'(b') \geq \alpha f'(a') + (1 - \alpha) f'(b') = \alpha f(a) + (1 - \alpha) f(b) \geq 1 \).

If \( b' \leq \frac{1}{1 - \theta} C_{j^*} \), then we let \( a = \frac{1}{1 - \theta}, b > \frac{1}{1 - \theta} \) be arbitrary and \( \alpha = \frac{b - \theta a - 1}{b - a} = 1 \). By Property (5.3a), we have \( f \left( \frac{1}{1 - \theta} \right) \geq 1 \). Thus, \( \sum_t y_{j,t} f'(t) \geq \alpha' f'(a') + (1 - \alpha') f'(b') = \alpha' f \left( \frac{a'}{C_{j^*}} \right) + (1 - \alpha') f \left( \frac{b'}{C_{j^*}} \right) \geq \alpha' f \left( \frac{1}{1 - \theta} \right) + (1 - \alpha') f \left( \frac{1}{1 - \theta} \right) = f \left( \frac{1}{1 - \theta} \right) \geq 1 \). This finishes the proof of Lemma 5.3.

Summing up inequalities in Lemma 5.2 and 5.3 and taking the sum of the bound over all clients \( j^* \in J \), we have:

**Lemma 5.4.** Let \( f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0} \) be a monotone non-increasing function satisfying Property (5.3a) for some \( \theta \in (0, 1) \). Then the approximation ratio of our algorithm is at most
\[ \int_{\tau=0}^{\infty} f(\tau) d\tau + \frac{1}{\theta}. \]  

**Proof.** By Lemma 5.3, we have
\[ B \leq \sum_{j \neq j^*} \sum_{j : C_j - \theta L_j \leq C_{j^*}} \frac{p_{i,j}}{C_j} \leq C_{j^*} \int_{\tau=0}^{\infty} f(\tau) d\tau. \]

Combing the above inequality with Lemma 5.2, we have
\[ \tilde{C}_{j^*} = B + 1 \leq C_{j^*} \int_{\tau=0}^{\infty} f(\tau) d\tau + \frac{1}{\theta} C_{j^*} = \left( \int_{\tau=0}^{\infty} f(\tau) d\tau + \frac{1}{\theta} \right) C_{j^*}. \]

The lemma follows by multiplying the above inequality by \( w_{j^*} \) and sum up it over all \( j^* \in J \).

Thus, we now focus on finding the parameter \( \theta \in (0, 1) \) and the monotone function \( f \) satisfying Property (5.3a) for this \( \theta \), so as to minimize (22).
Obtaining Final Approximation Ratio by Giving $f$ and $\theta$  
To obtain a purely analytical approximation ratio, we further restrict that $f$ to be concave over $[0, \tau^*]$ and $f(\tau) = 0$ if $\tau \geq \tau^*$, for some $\tau^* > 1/(1 - \theta)$. In this way, the hardest constraints in Property (5.3a) are those with $b = \tau^*$. The function $f$ is defined in the following lemma:

**Lemma 5.5.** Let $\theta \in (0, 1)$ and $\tau^* > 1/(1 - \theta)$ be some real number. The following function $f$ satisfies Property (5.3a):

$$f(\tau) = \begin{cases} \frac{\tau^* - \tau}{\tau^* - \theta \tau - 1} & \tau \in [0, \frac{1}{1-\theta}] \\ \frac{\tau^* - \tau}{\tau^* - 1/(1 - \theta)} & \tau \in \left(\frac{1}{1-\theta}, \tau^*\right] \\ 0 & \tau > \tau^* \end{cases}$$

**Proof.** Focus on $f$ on the domain $[0, \frac{1}{1-\theta}]$. The derivative of $f$ over $\tau$ in this domain is

$$-\frac{1(\tau^* - \theta \tau - 1) - (\tau^* - \tau)(-\theta)}{(\tau^* - \theta \tau - 1)^2} = \frac{1 - (1 - \theta)\tau^*}{(\tau^* - \theta \tau - 1)^2} < 0.$$ 

Thus, $f$ is decreasing over $[0, \frac{1}{1-\theta}]$. Moreover, $\tau^* - \theta \tau - 1 \geq \tau^* - \frac{\theta}{1-\theta} - 1 = \tau^* - \frac{1}{1-\theta} > 0$. Thus, as $\tau$ increases, the derivative decreases. So, $f$ is concave over $[0, \frac{1}{1-\theta}]$. Moreover,

$$f\left(\frac{1}{1-\theta}\right) = \frac{\tau^* - 1/(1 - \theta)}{\tau^* - \theta/(1 - \theta) - 1} = 1 = \lim_{\tau \to (\frac{1}{1-\theta})^+} f(\tau),$$

and

$$\frac{d f}{d \tau}|_{\tau \to (\frac{1}{1-\theta})^-} = \frac{1 - (1 - \theta)\tau^*}{(\tau^* - \theta/(1 - \theta) - 1)^2} = \frac{1 - (1 - \theta)\tau^*}{(\tau^* - 1/(1 - \theta))^2} = \frac{-(1 - \theta)}{\tau^* - 1/(1 - \theta)} \geq \frac{-1}{\tau^* - 1/(1 - \theta)} = \frac{d f}{d \tau}|_{\tau \to (\frac{1}{1-\theta})^+}.$$

So, $f$ is continuous at $\frac{1}{1-\theta}$ and the left derivative of $f$ at $\frac{1}{1-\theta}$ is larger than the right derivative. Moreover $f$ is a linear function on $(\frac{1}{1-\theta}, \tau^*)$. Thus, $f$ is concave over $[0, \tau^*]$. $f(\tau) = 0$ for every $\tau \geq \tau^*$.

We now prove that $f$ satisfies Property (5.3a). Focus on any $a \leq \frac{1}{1-\theta} < b$ and $\alpha = \frac{b - \theta a - 1}{b - a} \in [0, 1]$; we need to prove (*): $\alpha f(a) + (1 - \alpha)f(b) \geq 1$. If $b > \tau^*$, we can change $b$ to $\tau^*$, this will decrease $\alpha$ without changing $f(b)$ and thus make (*) harder to satisfy. If $b < \tau^*$, then we can also change $b$ to $\tau^*$ so that (*) becomes harder to satisfy. To see this, we fix $a \in \left[0, \frac{1}{1-\theta}\right]$. Then $\alpha$ is the number such that $\alpha a + (1 - \alpha)b = \theta a + 1$. Since $f$ is concave over $[0, \tau^*]$, $\alpha f(a) + (1 - \alpha)f(b)$ is minimized when $b = \tau^*$. Thus, it suffices to show that for every $a \in \left[0, \frac{1}{1-\theta}\right]$, we have $\frac{\tau^* - \theta a - 1}{\tau^* - a} f(a) \geq 1$. We only need $f(a)$ to be at least $\frac{\tau^* - \theta a - 1}{\tau^* - a}$, which is exactly the definition of $f(a)$ when $a \leq \frac{1}{1-\theta}$.

For the $f$ given in Lemma 5.5, we have
\[
\int_{\tau=0}^{\infty} f(\tau) d\tau = \int_{\tau=0}^{\tau_1} \left( \frac{\tau^* - \tau}{\tau^* - \theta \tau - 1} \right) d\tau + \int_{\tau=\tau_1}^{\tau*} \left( \frac{\tau^* - \tau}{\tau^* - 1/(1 - \theta)} \right) d\tau \\
= \left( \frac{\tau}{\theta} - \frac{(\tau^* - \tau + 1) \ln(\tau^* - \theta \tau - 1)}{\theta^2} \right) \bigg|_{\tau=0}^{\tau_1} + \frac{1}{2} \left( \tau^* - 1 \right) - \frac{1}{1 - \theta} \\
= \frac{1}{\theta(1 - \theta)} + \frac{(\tau^* - \tau + 1)}{\theta^2} \ln \left( \frac{\tau^* - 1}{\tau^* - 1/(1 - \theta)} \right) + \frac{1}{2} \left( \tau^* - 1 \right) - \frac{1}{1 - \theta} . \tag{23}
\]

The approximation ratio is at most the above quantity plus \(\frac{1}{\theta} \). Let \(\theta = 0.493\) and \(\tau^* = 2.791 > \frac{1}{1 - \theta}\), the ratio is at most 5.102. Thus, we obtain a 5.102-approximation algorithm for \(O||\sum_j w_j C_j\).

We also attempted to use a computer program to find the best \(f\) (without assuming \(f\) is concave over its support). We discretize the \([0, 2/(1 - \theta)]\) into intervals of length 0.005 and assume that \(f\) is a constant over each interval. It is not hard to show that the hardest constraints in Property (5.3a) are those with \(a\) and \(b\) being the left endpoints of intervals (assuming the \(f\) value of a breaking point is equal to the right limit of the point). Then for a fixed \(\theta\), finding the best \(f'\) of this type becomes a linear programming. We used \texttt{lp.solve} to solve the LPs. After a few attempts, we saw that by setting \(\theta \approx 0.5\) gives a ratio of 5.005. However, we chose to use the approximation ratio 5.102 obtained from the pure analytical proof in the statement of Theorem 1.3, as it can be verified by hand and it is not much worse than the one obtained from a computer program.

6 Scheduling on Related Machines with Job Precedence Constraints

In this section, we give our \(O(\lg m / \lg \lg m)\)-approximation for \(Q|\text{prec}|C_{\text{max}}\) and \(Q|\text{prec}|\sum_j w_j C_j\), the problems of minimizing the makespan and the total weighted completion time on related machines, with job precedence constraints. This slightly improves the previous best \(O(\lg m)\)-approximation, due to Chudak and Shmoys [6]. As we mentioned, our improved result is by a better tradeoff between two contributing factors to the approximation ratio.

As in [6], we can convert the objective of minimizing total weighted completion time to minimizing makespan, losing a factor of 16. This reduction requires that the algorithm for \(Q|\text{prec}|C_{\text{max}}\) to be based on certain type of LPs. Since we are using the same LP as in [6], we describe the LP and then state the theorem for the reduction.

\begin{align*}
\min & \quad D & (\text{LP}_{Q|\text{prec}|C_{\text{max}}}) \\
\sum_{i \in M} x_{i,j} = 1 & \quad \forall j \in J \quad (24) & \quad \frac{1}{s_i} \sum_{j \in J} p_j x_{i,j} \leq D \quad \forall i \in M \quad (27) \\
p_j \sum_{i \in M} \frac{x_{i,j}}{s_i} \leq C_j & \quad \forall j \in J \quad (25) \quad C_j \leq D \quad \forall j \in J \quad (28) \\
C_j + p_j \sum_{i \in M} \frac{x_{i,j'}}{s_i} \leq C_{j'} & \quad \forall j, j' \in J, j \prec j' \quad (26) \quad x_{i,j}, C_j \geq 0 \quad \forall j \in J, i \in M \quad (29)
\end{align*}

(\text{LP}_{Q|\text{prec}|C_{\text{max}}}) is a valid LP relaxation for \(Q|\text{prec}|C_{\text{max}}\). In the LP, \(x_{i,j}, i \in M, j \in J\) indicates whether job \(j\) is scheduled on machine \(i\). \(D\) is the makespan of the schedule, and \(C_j, j \in J\), is the completion time of \(j\) in the schedule. Constraint (24) requires every job \(j\)
to be scheduled. Constraint (25) says that the completion time of $j$ is at least the processing time of $j$ on the machine it is assigned to. Constraint (26) says that if $j < j'$, then $C_{j'}$ is at least $C_j$ plus the processing time of $j'$ on the machine it is assigned to. Constraint (27) says that the makespan $D$ is at least the total processing time of all jobs assigned to $i$, for every machine $i$. Constraint (28) says that the makespan $D$ is at least the completion time of any job $j$. Constraint (29) requires the $x$ and $C$ variables to be non-negative.

The value of (LP$_Q$|prec|C$_\text{max}$) 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 $Q|$prec|C$_\text{max}$. Using the following theorem from [6], we can extend the result to $Q|$prec|\sum_j w_jC_j:

**Theorem 6.1** ([6]). Suppose there is an algorithm $A$ that can round a fractional solution to (LP$_Q$|prec|C$_\text{max}$) to a valid solution to the correspondent $Q|$prec|C$_\text{max}$ instance, losing only a factor of $\alpha$. Then there is a $16\alpha$-approximation for the problem $Q|$prec|\sum_j w_jC_j.

Thus, from now on, we focus on the objective of minimizing the makespan and the LP relaxation (LP$_Q$|prec|C$_\text{max}$). We also assume that $m$ is big enough. For the given instance of $Q|$prec|C$_\text{max}$, we shall first pre-processing the instance as in [6] 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 $D$ to 2$D$, the LP solution remains feasible. Thus, we assume all machines have speed larger than $1/m$ times the speed of the fastest machine.

In the second stage of the pre-processing step, we partition the machines into groups, where each group contains machines with similar speeds. More specifically, by scaling the machine speeds uniformly, we can assume each machine $i$ has speed $s_i \in [1,m]$. Let $\gamma = \lg m / \lg \lg m$. Then, then group $M_k$ contains machines with speed in $[\gamma^{k-1}, \gamma^k)$, where $k = 1,2,\cdots,K := \lfloor \lg m \rfloor = O(\log m / \log \log m)$. We remark that that unlike [6], we can not round down the speed of each machine $i$ to the nearest power of $\gamma$. 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 $\ell_j$ be the largest integer $\ell$ such that $\sum_{k=\ell}^K x_{M_k,j} \geq 1/2$. That is, the largest $\ell$ such that at least $1/2$ fraction of $j$ is assigned to machines in groups $\ell$ to $K$. Then, let $k_j$ be the index $k \in [\ell,K]$ that maximizes $s(M_k)$. That is, $k_j$ is the index of the group in groups $\ell$ 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 $k_j$.

**Claim 6.2.** For every $j \in J$, and any machine $i \in M_{k_j}$, we have $\frac{p_{j,i}}{s_i} \leq 2\gamma \sum_{i' \in M} \frac{p_{j,i'}}{s_{i'}}$.

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

22
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. 

\[ \sum_{j \in J} \frac{p_j}{s(M_{k_j})} \leq 2KD. \]

Claim 6.3.

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

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

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

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

To see the last inequality, we notice that $\sum_{j \in J} x_{M_k,j}$ 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 (27) says $\sum_{j \in J} p_j x_{M_k,j} \leq s_i D$ 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 [6]. 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 $S$ 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 $O(\lg m/\lg \lg m)$-approximation. Similar observations were made and used implicitly in [6], and in [7] 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 caused by boundary cases, 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 6.4. At any non-critical time point $t$ in $S$, 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 that time, it must be the case that all machines in $M_{k_j}$ are busy. 

As time goes in $S$, we maintain a 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 $s_{ij}$. 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

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

where $H$ is over all precedence chains of jobs. The first inequality is by Claim 6.2 and the second inequality is by Constraints (26) and (28) 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 6.4. The total length of the union of all these points is at most

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

by Claim 6.3.

Thus, our schedule has makespan at most $2(\gamma + K)D = O(\log m / \log \log m)D$. This shows that our algorithm is an $O(\log m / \log \log m)$-approximation for $Q|\text{prec}|C_{\max}$. Combining this with Theorem 6.1, that reduces the $\sum_j w_j C_j$ objective to the $C_{\max}$ objective, we obtain an $O(\log m / \log \log m)$-approximation for $Q|\text{prec}|\sum_j w_j C_j$, finishing the proof of Theorem 1.4.

7 (1.5−$c$)-Approximation Algorithm for Unrelated Machine Scheduling Based on Time-Indexed LP

In this section, we give our (1.5−$c$)-approximation algorithm for the classic problem of scheduling on unrelated machines to minimize total weighted completion time, via a natural time-indexed LP relaxation. Our algorithm uses the dependence rounding scheme of [2] as a blackbox. We first describe our LP relaxation and the rounding algorithm, and then give our analysis.

7.1 LP Relaxation and Rounding Algorithm

Our time-indexed LP for $R||\sum_j w_j C_j$ is described in (LP$_{R||wC}$). For this problem, we find it more convenient to use starting times to index variables. Throughout this section, $s$ is always restricted to be an integer in $\{0, 1, 2, \cdots, T-1\}$, where $T$ is an upper bound on the makespan of any reasonable schedule. Again we first assume $T$ is polynomial in $n$ and left the case where $T$ is not polynomial to Section 8. $t$ is restricted to be an integer in $[T]$.

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

$$\sum_{i,s} x_{i,j,s} = 1 \quad \forall j \quad (30) \quad x_{i,j,s} = 0 \quad \forall i, j, s > T - p_{i,j} \quad (32)$$

$$\sum_{j,s \in [t-p_{i,j}, t]} x_{i,j,s} \leq 1 \quad \forall i, t \quad (31) \quad x_{i,j,s} \geq 0 \quad \forall i, j, s \quad (33)$$

In the LP, $x_{i,j,s}$ 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 (30) requires every job $j$ to be scheduled. Constraint (31) says that on every machine $i$ at any time point $t$, only one job is being processed. Constraint (32) says that if job $j$ is scheduled on $i$, then it can not be started after $T - p_{i,j}$. Constraint (33) requires all variables to be nonnegative.
We solve \((\text{LP}_{R||wC})\) to obtain the optimum solution \(x\) to the LP. Again, we use \(C_j = \sum_{i,s} x_{i,j,s}(s + p_{i,j})\) to denote the completion time of \(j\) in the LP solution; thus the value of the LP is \(\sum_j w_j C_j\). Let \(y_{i,j} = \sum_s x_{i,j,s}\) for every pair \(i, j\) that is scheduled on machine \(i\); so \(\sum_i y_{i,j} = 1\) for every \(j\). It is convenient to define a rectangle \(R_{i,j,s}\) for every \(x_{i,j,s} > 0\): \(R_{i,j,s}\) has height \(x_{i,j,s}\), with horizontal span being \((s, s + p_{i,j})\). We say \(R_{i,j,s}\) is the rectangle for \(j\) on machine \(i\) at time \(s\). We say a rectangle covers a time point (or a time interval), if its horizontal span covers the time point (or the time interval), if its horizontal span covers the time point (or the time interval).

We can recover the classic 1.5-approximation for \(R||\sum_j w_j C_j\) using \((\text{LP}_{R||wC})\). For each job \(j \in J\), randomly choose a rectangle for \(j\): the probability of choosing \(R_{i,j,s}\) is \(x_{i,j,s}\), i.e., its height. We shall assign job \(j\) to \(i\) and let \(\tau_j\) be a random number in \((s, s + p_{i,j})\). Then, all jobs \(j\) assigned to machine \(i\) will be scheduled in increasing order of their \(\tau_j\) values. To see this is a 1.5-approximation, fix a job \(j^* \in J\) and conditioned on the event that \(j^* \rightarrow i\) (indicating the event that job \(j^*\) is assigned to machine \(i\)) and \(\tau_j\). Let \(\hat{C}_j\) be the completion time of \(j^*\) in the schedule returned by the algorithm. Notice that \(E[\hat{C}_j \mid j^* \rightarrow i, \tau_j] \leq \sum_{j \neq j^*} \tau_j \leq \tau_j^* p_{i,j} + p_{i,j^*} \). We consider the contribution of each rectangle \(R_{i,j,s}, j \neq j^*\) to the expectation. The probability that we choose the rectangle \(R_{i,j,s}\) for \(j\) is \(x_{i,j,s}\). Under this condition, the probability that \(\tau_j \leq \tau_j^*\) is exactly fraction of the area in \(R_{i,j,s}\) that is before \(\tau_j^*\). When this happens, the contribution made by this \(R_{i,j,s}\) is exactly \(p_{i,j}\). Thus, the expected contribution of \(R_{i,j,s}\) is the area of the portion of \(R_{i,j,s}\) before \(\tau_j^*\). Since the total height of rectangles on \(i\) covering any time point is at most 1, the total contribution from all rectangles on \(i\) is at most \(\tau_j^*\). Thus \(E(C_j \mid j^* \rightarrow i, \tau_j^*) \leq \sum_{j \neq j^*} \tau_j \leq \tau_j^* p_{i,j} + p_{i,j^*} \leq \tau_j^* + \tau_j^*\). Notice that conditioned on choosing rectangle \(R_{i,j,s}\) for \(j^*\), the expected value of \(\tau_j^*\) is \(s + p_{i,j^*}/2\). Thus, \(E(C_j) \leq \sum_{i,s} x_{i,j,s}(s + p_{i,j^*}/2 + p_{i,j^*}) = \sum_{i,s} x_{i,j,s} (s + 1.5 p_{i,j^*}) \leq 1.5 \sum_{i,s} x_{i,j,s} (s + p_{i,j^*}) = 1.5 C_j^*\).

As [2] already showed, we can not beat 1.5 if the assignments of jobs to machines are independent. This lower bound is irrespective of the LP we use: even if the fractional solution is the LP is satisfying \(\text{edges}\). Let \(\eta = 0, 1\) \(E^i_1, E^i_2, \ldots, E^i_{\kappa_i} \subseteq \delta(i)\) subsets of edges incident to \(i\) such that \(y(E^i_j) \leq 1\) for \(\ell = 1, \ldots, \kappa_i\).

Then, there exists a randomized polynomial-time algorithm that outputs a random subset of the edges \(E^* \subseteq E\) satisfying

(a) For every \(j \in J\), we have \(|E^* \cap \delta(j)| = 1\) with probability 1;

(b) For every \(e \in E\), \(\Pr[e \in E^*] = y_e\);

(c) For every \(i \in M\) and all \(e \neq e' \in \delta(i)\):

\[
\Pr\left[e \in E^*, e' \in E^*\right] \leq \begin{cases} 
(1 - \zeta) \cdot y_e y_{e'} & \text{if } e, e' \in E^i_\ell \text{ for some } \ell \in \{1, 2, \ldots, \kappa_i\}, \\
y_e y_{e'} & \text{otherwise}.
\end{cases}
\]

In the theorem, \(\delta(u)\) is the set of edges incident to the vertex \(u\) in \(G\), and \(y(E') = \sum_{e \in E'} y_e\) for every \(E' \subseteq E\). We shall apply the theorem with \(G = (M \cup J, E)\), with \(E = \{(i, j) : y_{i,j} > 0\}\) and \(y\) values being our \(y\) values. In the theorem, we can specify a grouping for edges incident to every machine \(i \in M\) subject to the constraint that the total \(y\)-value of all edges in a group
is at most 1. The theorem says that we can select a subset \( E^* \subseteq E \) of edges respecting the marginal probabilities \( \{y_{i,j}\}_{(i,j) \in E} \), and satisfying the property that exactly one edge incident to any job \( j \) is selected, and the negative correlation. The key to the improved approximation ratio in [2] is that for two distinct edges \( e, e' \) in the same group for \( i \), their correlation should be “sufficiently negative”.

The key to apply Theorem 7.1 is to define a grouping for each machine \( i \). Notice that our analysis for the independent rounding algorithm suggests that the expected completion time for \( j^* \) is at most \( \sum_{i,a} x_{i,j^*,s} (s + 1.5p_{i,j^*}) \); that is, there is no 1.5-factor before \( s \). Thus, for the 1.5-approximation ratio to be tight, for most rectangles \( R_{i,j^*,s} \), \( s \) should be very small compared to \( p_{i,j^*} \). Intuitively, we define our random groupings such that, if such rectangles for \( j \) and for \( j' \) on machine \( i \) overlap a lot, then \( j \) and \( j' \) will have a decent probability to be grouped together in the grouping for \( i \).

**Defining the Groupings for Machines** Our groupings for the machines are random and will depend on the \( \tau \) values of jobs. We shall choose a \( \tau_{i,j} \) value for every job \( j \) on every machine \( i \) such that \( y_{i,j} > 0 \) (as opposed to choosing only one \( \tau_j \) value for a job \( j \) ). Choose \( s_{i,j} \) at random such that \( \Pr[s_{i,j} = s] = x_{i,j,s}/y_{i,j} \); this is well-defined since \( \sum_s x_{i,j,s} = y_{i,j} \). Then we let \( \tau_{i,j} \) be a random real number in \( [s_{i,j}, s_{i,j} + p_{i,j}] \).

Recall that the expected completion time of \( j \) in the schedule given by the independence rounding is \( \sum_{s} x_{i,j,s}(s + 1.5p_{i,j}) \). Thus, if some rectangle \((i,j,s)\) has a large \( s \), it will save some factor in the coefficient for completion time of \( j \). With this in mind, we can afford to “shift” such a rectangle to the right side by some distance. On one hand, we define the shifting parameter such that \( j \) itself will not be hurt. On the other hand, the shifting of rectangles for \( j \) will benefit the other jobs scheduled on \( i \). Formally, for every machine \( i \) and job \( j \) with \( y_{i,j} > 0 \), we define

\[
\phi_{i,j} = \frac{1}{y_{i,j}} \sum_s x_{i,j,s} s \quad \text{and} \quad \theta_{i,j} = 0.2(s_{i,j} + \phi_{i,j}) + 0.4y_{i,j}p_{i,j}.
\]

\( \phi_{i,j} \) is the average starting time of rectangles for job \( j \) on machine \( i \). \( \theta_{i,j} \) will be the shifting parameter for job \( j \) on \( i \); namely, we shall use the values of \( \{\tau_{i,j} + \theta_{i,j}\}_{i,j} \) to decide the order of scheduling jobs.

We shall distinguish between good jobs and bad jobs. Informally, we say a job \( j \) is good on a machine \( i \), if using the independent rounding algorithm, we can prove a better than 1.5-factor on the completion time of \( j \), conditioned on that \( j \) is assigned to \( i \). Formally,

**Definition 7.2.** Given a job \( j \) and machine \( i \) with \( y_{i,j} > 0 \), we say \( j \) is good on \( i \) if

\[
\phi_{i,j} + y_{i,j}p_{i,j} \geq 0.01p_{i,j}.
\]

Otherwise, we say job \( j \) is bad on \( i \).

Now we are ready to define the groupings \( \{E_i^j\}_{i \in M, j \in [K_i]} \) in Theorem 7.1. Till this end, we fix a machine \( i \) and show how to construct the grouping for \( i \). If a job \( j \) is good on \( i \), then \((i,j)\) is not in any group. Thus, it suffices to focus on bad jobs on \( i \).

**Definition 7.3.** A basic block is a time interval \((2^a, 2^{a+1}] \subseteq (0, T)\), where \( a \geq -2 \) is an integer.

**Definition 7.4.** For a bad job \( j \) on machine \( i \) with \( y_{i,j} > 0 \), we say the edge \((i,j)\) is assigned to a basic block \((2^a, 2^{a+1})\), denoted as \( j \to a \), if

(7.4a) \((2^a, 2^{a+1}] \subseteq (10\phi_{i,j}, p_{i,j}]\), and
(7.4b) \(s_{i,j} + \theta_{i,j} \leq 2^a\), and
Property (7.4a) requires the block to be inside \((10\phi_{i,j}, p_{i,j})\) and Property (7.4c) requires \(\tau_{i,j}\) to be inside the block. Property (7.4b) requires that for the rectangle for \(j\) starting at \(s_{i,j}\), after we shift it by \(\theta_{i,j}\) distance, it still contains \((2^a, 2^{a+1})\). With Property (7.4a) and the definition of \(\phi_{i,j}\), we can prove the following lemma:

**Lemma 7.5.** For every machine \(i\) and a basic block \((2^a, 2^{a+1})\), we have \(\sum_{j \sim a} y_{i,j} \leq 10/9\).

*Proof.* Focus on a job \(j\) that is bad on \(i\). We have \(\sum_{s \leq 10\phi_{i,j}} x_{i,j,s} \geq 9y_{i,j}/10\), since otherwise we shall have \(\sum_{s} x_{i,j,s} > 10\phi_{i,j} \cdot (y_{i,j}/10) = \phi_{i,j}y_{i,j}\), contradicting the definition of \(\phi_{i,j}\). Thus, \(y_{i,j} \leq (10/9)\sum_{s \leq 10\phi_{i,j}} x_{i,j,s}\). If \(j \sim a\), then \((2^a, 2^{a+1}) \subseteq (10\phi_{i,j}, p_{i,j})\) by Property (7.4a). Thus, \((2^a, 2^{a+1})\) is covered by \((s, s + p_{i,j})\) for every \(s \leq 10\phi_{i,j}\). Thus, we have \(\sum_{j \sim a} y_{i,j} \leq (10/9)\sum_{j \sim a, s \leq 10\phi_{i,j}} x_{i,j,s} \leq 10/9\). The second inequality used the fact that the total height of rectangles covering \((2^a, 2^{a+1})\) on machine \(i\) is at most 1.

For every basic block \((2^a, 2^{a+1})\), we partition the set of edges assigned to \((2^a, 2^{a+1})\) into at most 10 sets, each set containing edges with total weight at most 1/8. This is possible since the total weight of all edges assigned to \((2^a, 2^{a+1})\) is at most 10/9 by Lemma 7.5, and every bad job \(j\) on \(i\) has \(y_{i,j} < 0.01\): we can keep adding edges to a set until the total weight is at least 1/8 and then we start constructing the next set; the number of sets we constructed is at most \((10/9)/(1/9) = 10\) and each set has a total weight of at most 1/9 + 0.01 ≤ 1/8. Now, we can randomly drop at most 2 sets so that we have at most 8 sets remaining. Then we create a group for \(i\) containing the edges in the remaining sets. Thus, the total \(y\)-value of the edges in this group is at most 1.

So, we have defined the grouping for \(i\); recall that for each basic block \((2^a, 2^{a+1}) \subseteq (10\phi_{i,j}, p_{i,j})\), we may create a group. For a bad job \(j\) on \(i\), \((i, j)\) may not be assigned to any group. This may happen if \((i, j)\) is not assigned to any basic block, or if \((i, j)\) is assigned to a basic block, but we dropped the set containing \((i, j)\) when constructing the group for the basic block.

**Obtaining the Final Schedule** With the groupings for the machines, we can now apply Theorem 7.1 to obtain a set \(E^*\) of edges that satisfies the properties of the theorem. If \((i, j) \in E^*\), then we assign \(j\) to \(i\); \(j\) is assigned to exactly one machine since \(E^*\) contains exactly one edge incident to \(j\). For all the jobs \(j\) assigned to \(i\), we schedule them according to the increasing order of \(\tau_{i,j} + \theta_{i,j}\); with probability 1, no two jobs \(j\) have the same \(\tau_{i,j} + \theta_{i,j}\) value. Let \(\tilde{C}_j\) be the completion time of \(j\) in this final schedule.

### 7.2 Analysis of Algorithm

**Notations and Simple Observations** From now on, we use \(\sim^i\) to denote the event that \((i, j)\) is assigned to the machine \(i\). Recall that \(\sim^i\) indicates the event that \((i, j)\) is assigned to the basic block \((2^a, 2^{a+1})\). From the way we construct the groups, the following observation is immediate:

**Observation 7.6.** Let \((2^a, 2^{a+1})\) be a basic block, \(j \neq j'\) be bad jobs on \(i\). We have

\[
\Pr\left[\sim^i, j \sim^i, j' \sim^i\right] \geq 0.8.
\]

The following observation will be used in the analysis:
Observation 7.7. Let \( j \neq j' \) be bad jobs on \( i \). Then

\[
\Pr \left[ \tau_{i,j} + \theta_{i,j} < \tau_{i,j'} + \theta_{i,j'} | j \sim j' \right] = 1/2.
\]

Proof. Fix a basic block \((2^a,2^{a+1})\) such that \((2^a,2^{a+1}) \in (10\phi_{i,j'},p_{i,j'}) \) and \((2^a,2^{a+1}) \in (10\phi_{i,j'},p_{i,j'})\) (i.e., Property (7.4a) for both \( j \) and \( j' \)).

Fix a \( s_{i,j} \) such that \( s_{i,j} + \theta_{i,j} \leq 2^a \) (i.e., Property (7.4b)). Condition on this \( s_{i,j}, \tau_{i,j} \) is uniformly distributed in \((s_{i,j},s_{i,j} + p_{i,j})\), and thus \( \tau_{i,j} + \theta_{i,j} \) is uniformly distributed in \((s_{i,j} + \theta_{i,j}, s_{i,j} + p_{i,j} + \theta_{i,j}) \supseteq (2^a,2^{a+1})\). Thus, conditioned on \( s_{i,j} \) and \( j \sim a \), \( \tau_{i,j} + \theta_{i,j} \) is uniformly distributed in \((2^a,2^{a+1})\). This holds even if we only condition on \( j \sim a \), since the statement holds for any \( s_{i,j} \) satisfying \( s_{i,j} + \theta_{i,j} \leq 2^a \).

The same holds for \((i,j')\). For simplicity, denote by \( e \) the event \( \tau_{i,j} + \theta_{i,j} < \tau_{i,j'} + \theta_{i,j'} \). Thus, \( \Pr \left[ e | j \sim a, j' \sim a \right] = 1/2 \) (notice that the events \( j \sim a \) and \( j' \sim a \) are independent). Conditioned on \( j \sim a, j' \sim a \), the event \( j \sim j' \) is independent of the event \( e \). Thus, \( \Pr \left[ e | j \sim j', j \sim a, j' \sim a \right] = 1/2 \). This holds for every \( a \); thus, \( \Pr \left[ e | j \sim j' \right] = 1/2 \). \( \square \)

We define \( h_{i,j}(\tau) = \sum_{s \in \tau - p_{i,j}} x_{i,j,s} \) to be the total height of rectangles for \( j \) on \( i \) that cover \( \tau \). It is easy to see that \( \frac{h_{i,j}}{y_{i,j}} \) is the probability density function for \( \tau_{i,j} \). Let \( A_{i,j}(\tau) = \int_{\tau}^\infty h_{i,j}(\tau') d\tau' \) be the total area of the parts of rectangles \( R_{i,j,s} \) that are before \( \tau \); we simply use \( A_{i,j}(\tau) \) for \( A_{i,j}(\tau) \) for every \( i \) and \( \tau \in [0,T] \).

It is also convenient to define a set of “shifted” rectangles. For every \( i, j, s \), we define \( R'_{i,j,s} \) to be the rectangle \( R_{i,j,s} \) shifted by \( 0.2(s + \phi_{i,j}) + 0.4y_{i,j}p_{i,j} \) units of time to the right. That is, \( R'_{i,j,s} \) is the rectangle of height \( x_{i,j,s} \) with horizontal span \((1.2s + 0.2\phi_{i,j} + 0.4y_{i,j}p_{i,j}, 1.2s + 0.2\phi_{i,j} + 0.4y_{i,j}p_{i,j} + 1.2s + 0.2\phi_{i,j} + 0.4y_{i,j}p_{i,j})\). Notice that \( 0.2(s + \phi_{i,j}) + 0.4y_{i,j}p_{i,j} \) is the definition of \( \theta_{i,j} \) if \( s_{i,j} = s \). To distinguish these rectangles from the original rectangles, we call the new rectangles \( R' \)-rectangles and the original ones \( R \)-rectangles.

Similarly, we define \( h'_{i,j}(\tau) \) to be the total height of \( R' \)-rectangles for \( j \) on \( i \) that covers \( \tau \). Thus, \( \frac{h'_{i,j}}{y_{i,j}p_{i,j}} \) is the PDF for the random variable \( \tau_{i,j} + \theta_{i,j} \). Let \( A_{i,j}'(\tau) = \int_{\tau}^\infty h'_{i,j}(\tau') d\tau' \) to be the total area of the parts of rectangles \( R'_{i,j,s} \) that are before \( \tau \). Notice that for every \( i \in M \) and \( \tau \in [0,T] \), \( A_{i,j}'(\tau) \leq A_{i,j}(\tau) \leq \tau \) since we only shift rectangles to the right.

For two functions \( f : [0,T] \rightarrow \mathbb{R}_{\geq 0} \) and \( F : [0,T] \rightarrow \mathbb{R}_{\geq 0} \), define

\[
f \circ F = \int_0^T f(\tau) F(\tau) d\tau.
\]

Bounding Expected Completion Time Job by Job To analyze the approximation ratio of the algorithm, we fix a job \( j^* \) and a machine \( i \) such that \( y_{i,j^*} > 0 \). We shall bound \( \mathbb{E} [ C_{j^*} | j^* \rightarrow i ] \). It suffices to bound it by \((1.5 - \frac{1}{6000}) \sum_s \frac{\tau_{i,j^*}}{y_{i,j^*}} (s + p_{i,j^*}) = (1.5 - \frac{1}{6000}) (\phi_{i,j^*} + p_{i,j^*})\). The follow lemma gives a comprehensive upper bound that takes all parameters into account:

Lemma 7.8. Let \( I : [0,T] \rightarrow [0,T] \) be the identity function. Then \( \mathbb{E} [ C_{j^*} | j^* \rightarrow i ] \) is at most

\[
1.4\phi_{i,j^*} + (1.5 - 0.1y_{i,j^*}) p_{i,j^*} - \frac{h_{i,j^*}}{y_{i,j^*}p_{i,j^*}} \quad \circ \quad (I - A_{i,j^*}') - \frac{\zeta}{2} \sum_{j \neq j^*} \Pr \left[ j \sim j^* \right] y_{i,j}p_{i,j}. \tag{34}
\]

\(^4\)We slightly increase \( T \) so that even after shifting, all rectangles are inside the interval \((0,T)\).
If we throw away all the negative terms, then we get an $1.4\phi_{i,j^*} + 1.5p_{i,j^*}$ upper bound on $E[\tilde{C}_j^*|j^* \rightarrow i]$, which is at most $1.5(\phi_{i,j^*} + p_{i,j^*})$. If either $\phi_{i,j^*}$ or $y_{i,j^*}$ is large, then we can prove a better than 1.5 factor; this coincides with our definition of good jobs. For a bad job $j^*$ on $i$, we shall show that the absolute value of the third and fourth term in (34) is large. The third term is large if many rectangles are shifted by a large amount. The fourth term is large if many rectangles are shifted by a large amount. The fourth term is where we use the strong negative correlation of Theorem 7.1 from [2] to reduce the final approximation ratio.

**Proof of Lemma 7.8.** For notational convenience, let $e_j$ denote the event that $\tau_{i,j} + \theta_{i,j} \leq \tau_{i,j^*} + \theta_{i,j^*}$, for every $j \in J$.

\[
E[\tilde{C}_j^*|j^* \rightarrow i] = \frac{1}{\Pr[j^* \rightarrow i]} E[1_{j^* \rightarrow i} \times \tilde{C}_j^*] = \frac{1}{y_{i,j^*}} \sum_j \Pr[j^* \rightarrow i, j \rightarrow i, e_j] \times p_{i,j}
\]

\[
= \frac{1}{y_{i,j^*}} \sum_{j \neq j^*} \left( \Pr [e_j, j \sim j^*] \times \Pr [j^* \rightarrow i, j \rightarrow i, e_j, j \sim j^*] \right)
+ \Pr [e_j, j \sim j^*] \times \Pr [j^* \rightarrow i, j \rightarrow i, e_j, j \sim j^*] \times p_{i,j} + p_{i,j^*}
\]

\[
\leq \frac{1}{y_{i,j^*}} \sum_{j \neq j^*} \left( \Pr [e_j, j \sim j^*] (1 - \zeta) y_{i,j^*} y_{i,j} + \Pr [e_j, j \sim j^*] y_{i,j^*} y_{i,j} \right) p_{i,j} + p_{i,j^*}
\]

\[
= \sum_{j \neq j^*} \Pr [e_j] y_{i,j} p_{i,j} - \zeta \sum_{j \neq j^*} \Pr [e_j, j \sim j^*] y_{i,j} p_{i,j} + p_{i,j^*}
\]

\[
= \sum_{j \neq j^*} \Pr [e_j] y_{i,j} p_{i,j} - \frac{\zeta}{2} \sum_{j \neq j^*} \Pr [j \sim j^*] y_{i,j} p_{i,j} + p_{i,j^*}.
\]  

(35)

The second equality is due to the fact that $\tilde{C}_j^* = \sum_{j \rightarrow i; e_j} p_{i,j}$, conditioned on $j^* \rightarrow i$. The only inequality is due to the third property of $E^*$ in Theorem 7.1: conditioned on $j \sim j^*$ ($j \neq j^*$ resp.), the probability that $j^* \rightarrow i, j \rightarrow i$ is at most $(1 - \zeta) y_{i,j^*} y_{i,j}$ ($y_{i,j^*} y_{i,j}$ resp.), independent of the $\tau$ and $\theta$ values (thus, independent of $e_j$). The last equality is due to Observation 7.7.

We focus on the first term of (35):

\[
\sum_{j \neq j^*} \Pr [e_j] y_{i,j} p_{i,j} = \sum_{j \neq j^*} \Pr [\tau_{i,j} + \theta_{i,j} \leq \tau_{i,j^*} + \theta_{i,j^*}] y_{i,j} p_{i,j} = \frac{h'_{i,j^*}}{y_{i,j^*} p_{i,j^*}} \otimes A'_{i,j^*} \left( \begin{array}{c}
I - \frac{h'_{i,j^*}}{y_{i,j^*} p_{i,j^*}} \otimes (I - A'_{i,j}) - \frac{h'_{i,j^*}}{y_{i,j^*} p_{i,j^*}} \otimes A'_{i,j^*}.
\end{array} \right)
\]  

(36)

To see the second equality, we notice that $\tau_{i,j} + \theta_{i,j}$ has PDF $\frac{h'_{i,j^*}}{y_{i,j^*} p_{i,j^*}}$, $\tau_{i,j} + \theta_{i,j}$ has PDF $\frac{h'_{i,j^*}}{y_{i,j^*} p_{i,j^*}}$ and the two random quantities are independent if $j \neq j^*$. Thus, for a fixed $\tau_{i,j^*} + \theta_{i,j^*} = \tau$, the probability that $\tau_{i,j} + \theta_{i,j} \leq \tau$ is exactly $A'_{i,j^*}$. Thus contribution of $j$ is exactly $A'_{i,j^*}$. Summing up over all $j \neq j^*$ gives the equality. The third equality is by $A'_{i,j^*} \equiv I - (I - A'_{i,j}) - A'_{i,j^*}$.
The first term of (36) is
\[
\frac{h'_{i,j^*}}{y_{i,j^*}p_{i,j^*}} \otimes I = \sum_s \frac{x_{i,j^*,s}}{y_{i,j^*}} \int_{\tau=s}^{s+p_{i,j^*}} \left( \tau + 0.2(s + \phi_{i,j^*}) + 0.4y_{i,j^*}p_{i,j^*} \right) d\tau \]
\[
= \sum_s \frac{x_{i,j^*,s}}{y_{i,j^*}}(s + 0.5p_{i,j^*} + 0.2(s + \phi_{i,j^*}) + 0.4y_{i,j^*}p_{i,j^*}) = 1.4\phi_{i,j^*} + 0.5p_{i,j^*} + 0.4y_{i,j^*}p_{i,j^*}.
\]

Now focus on the third term of (36):
\[
\frac{h'_{i,j^*}}{y_{i,j^*}p_{i,j^*}} \otimes A'_{i,j^*} = y_{i,j^*}p_{i,j^*} \int_{\tau=0}^T \frac{h'_{i,j^*}(\tau)}{y_{i,j^*}p_{i,j^*}} \int_{\tau'=0}^T \frac{h'_{i,j^*}(\tau')}{y_{i,j^*}p_{i,j^*}} 1_{\tau' < \tau} d\tau d\tau' = \frac{y_{i,j^*}p_{i,j^*}}{2}.
\]

The first equality holds since \( A'_{i,j^*} \) is the integral of \( h'_{i,j^*} \). The second equality holds since the probability that \( \tau' < \tau \), where \( \tau \) and \( \tau' \) are i.i.d random variables and are not equal almost surely, is 1/2.

Thus, by applying the above two equalities to (36), we have
\[
\sum_{j \neq j^*} \Pr[e_j] y_{i,j^*}p_{i,j^*} = 1.4\phi_{i,j^*} + 0.5p_{i,j^*} - \frac{h'_{i,j^*}}{y_{i,j^*}p_{i,j^*}} \otimes (I - A'_{i,j^*}) - 0.1y_{i,j^*}p_{i,j^*}.
\]

Applying the above equality to (35), we that \( \mathbb{E}[\tilde{C}_{j^*} | j^* \rightarrow i] \) is at most
\[
1.4\phi_{i,j^*} + (1.5 - 0.1y_{i,j^*}) p_{i,j^*} - \frac{h'_{i,j^*}}{y_{i,j^*}p_{i,j^*}} \otimes (I - A'_{i,j^*}) - \frac{\zeta}{2} \sum_{j \neq j^*} \Pr[j^* \rightarrow j] y_{i,j^*}p_{i,j^*}.
\]

This is exactly (34). \( \square \)

With the lemma, we shall analyze good jobs and bad jobs separately. For good jobs, the bound follows directly from the definition:

**Lemma 7.9.** If \( j^* \) is good on \( i \), then \( \mathbb{E}[\tilde{C}_{j^*} | j^* \rightarrow i] \leq 1.4991(\phi_{i,j^*} + p_{i,j^*}). \)

**Proof.** We have \( \mathbb{E}[\tilde{C}_{j^*} | j^* \rightarrow i] \leq 1.4\phi_{i,j^*} + (1.5 - 0.1y_{i,j^*}) p_{i,j^*} \), by throwing the two negative terms in (34). Since \( j^* \) is good on \( i \), we have \( \phi_{i,j^*} + y_{i,j^*}p_{i,j^*} \geq 0.01p_{i,j^*} \).

\[
\mathbb{E}[\tilde{C}_{j^*} | j^* \rightarrow i] \leq 1.5(\phi_{i,j^*} + p_{i,j^*}) - 0.1\phi_{i,j^*} + 0.1y_{i,j^*}p_{i,j^*} \\
\leq 1.5(\phi_{i,j^*} + p_{i,j^*}) - 0.01\phi_{i,j^*} - 0.09(\phi_{i,j^*} + y_{i,j^*}p_{i,j^*}) \\
\leq 1.5(\phi_{i,j^*} + p_{i,j^*}) - 0.01\phi_{i,j^*} - 0.09 \times 0.01p_{i,j^*} \leq 1.4991(\phi_{i,j^*} + p_{i,j^*}). \quad \square
\]

Thus, it remains to consider the case where \( j^* \) is bad on \( i \). The rest of the section is devoted to the proof of the following lemma:

**Lemma 7.10.** If \( j^* \) is bad on \( i \), then \( \mathbb{E}[\tilde{C}_{j^*} | j^* \rightarrow i] \leq (1.5 - \frac{1}{6000}) (\phi_{i,j^*} + p_{i,j^*}). \)

It suffices to give a lower bound on the sum of the absolute values of negative terms in (34):
\[
0.1y_{i,j^*}p_{i,j^*} + \frac{h'_{i,j^*}}{y_{i,j^*}p_{i,j^*}} \otimes (I - A'_{i,j^*}) + \frac{\zeta}{2} \sum_{j \neq j^*} \Pr[j^* \rightarrow j] y_{i,j^*}p_{i,j^*} \geq \frac{p_{i,j^*}}{6000}.
\]
If the above inequality holds, then we have \( \mathbb{E} \left[ \tilde{C}_{j^*} | j^* \rightarrow i \right] \leq 1.4\phi_{i,j^*} + (1.5 - \frac{1}{6000}) p_{i,j^*} \leq (1.5 - \frac{1}{6000}) (\phi_{i,j^*} + p_{i,j^*}) \), implying Lemma 7.10.

To prove (38), we construct a set of configurations with total weight 1, and lower bound the left-side contribution by configuration. We define a configuration \( U \) to be a set of pairs in \( J \times \{0, 1, 2, \ldots, T - 1\} \) such that for every two distinct pairs \( (j, s), (j', s') \in U \), the two intervals \( (s, s + p_{i,j}) \) and \( (s', s' + p_{i,j'}) \) are disjoint.\(^5\) For the sake of the description, we also view \((j, s)\) as the interval \((s, s + p_{i,j})\), associated with the job \(j\).

Recall that the total height of all \( R \)-rectangles on \( i \) covering any time point is at most 1. It is a folklore result that we can find a set of configurations, each configuration \( U \) with a \( z_U > 0 \), such that \( \sum_U z_U = 1 \) and \( \sum_{U \ni (s, j)} z_U = x_{i,j,s} \) for every \( j \) and \( s \).

With the decomposition of the \( R \)-rectangles on \( i \) into a convex combination of configurations, we can now analyze the contribution of each configuration to the left of (38). For any configuration \( U \), we define a function \( \ell_U : [0, T] \rightarrow \mathbb{R}_{\geq 0} \) as follows:

\[
\tilde{\ell}_U(\tau) = \sum_{(j,s) \in U: 1.2s + 0.2p_{i,j} + 0.4y_{i,j}p_{i,j} \leq \tau} \min \{ \tau - (1.2s + 0.2\phi_{i,j} + 0.4y_{i,j}p_{i,j}), p_{i,j} \}.
\]

The definition comes from the following process. Focus on the intervals \( \{(s, s + p_{i,j}) : (j, s) \in U\} \). We then shift each interval \((s, s + p_{i,j})\) to the right by \(0.2s + 0.2\phi_{i,j} + 0.4y_{i,j}p_{i,j}\); notice that this is exactly the definition of \( \theta_{i,j} \) when \( s_{i,j} = s \). Then \( \ell_U(\tau) \) is exactly the total length of the sub-intervals of the shifted intervals before time point \( \tau \). Recalling that \( A_{i,j}'(\tau) \) is the total area of the parts of the \( R' \)-rectangles on \( i \) before time point \( \tau \), and each \( R_{i,j,s}' \) is obtained by shifting \( R_{i,j,s} \) by \(0.2s + 0.2\phi_{i,j} + 0.4y_{i,j}p_{i,j} \) to the right, the following holds:

\[
A_{i,j}' = \sum_U z_U \ell_U.
\]  

(39)

Let \( c_U(j) = |\{s : (j, s) \in U\}| \) be the number of pairs in \( U \) for the job \( j \). Now, we can define the contribution of \( U \) to the bound to be

\[
D_U := z_U \left( 0.1p_{i,j^*}c_U(j^*) + \frac{h_{i,j^*}}{y_{i,j^*}p_{i,j^*}} \otimes (I - \ell_U) + \frac{\zeta}{2} \sum_{j \neq j^*} \Pr[j \sim j^*]p_{i,j}c_U(j) \right).
\]

Claim 7.11. The left side of (38) is exactly \( \sum_U D_U \).

Proof. Indeed, the three terms in (38) is respectively the sum over all \( U \) of each of the three terms in the definition of \( D_U \). For the first and the third term, the equality comes from \( y_{i,j} = \sum_U z_U c_U(j) \) for every \( j \). The equality for the second term comes from \( \sum_U z_U (I - \ell_U) = I - A_{i,j}' \). \( \square \)

Lower Bound the Contribution of Each \( U \) \( \) Now we fix some configuration \( U \) such that \( z_U > 0 \). Let \( a \) be the largest integer such that \( 2^a + 1 \leq 0.9p_{i,j^*} \); thus \( a \geq -2 \). We can focus on the basic block \((2^a, 2^{a+1}]\). Notice that the length of the basic block is \( 2^a \geq 0.9p_{i,j^*}/4 \), by the definition of \( a \).

The following simple observations are useful in establishing our bounds.

Observation 7.12. If \( s \leq 18\phi_{i,j^*} \), then \( R_{i,j^*} \) will cover \((2^a, 2^{a+1}]\).

\(^5\) Notice that this configuration does not necessarily correspond to a valid scheduling on \( i \), since it may contain two pairs with the same \( j \).
Proof. The rectangle \( R'_{i,j,s} \) will cover \( (2^a, p_{i,j}) \) if \( s + 0.2(s + \phi_{i,j}) + 0.4y_{i,j}p_{i,j} \leq 2^a \), which is \( s \leq (2^a - 0.2\phi_{i,j} - 0.4y_{i,j}p_{i,j})/1.2 \). Since \( j^* \) is bad on \( i \), we have \( 0.2\phi_{i,j} + 0.4y_{i,j}p_{i,j} \leq 0.4 \times 0.01p_{i,j} = 0.004p_{i,j} \). Thus, \( (2^a - 0.2\phi_{i,j} - 0.4y_{i,j}p_{i,j})/1.2 \geq (2^a - 0.004p_{i,j})/1.2 \geq (0.9/4 - 0.004)p_{i,j}/1.2 \geq \frac{0.9/4 - 0.004}{1.2} \times \phi_{i,j} \geq 18\phi_{i,j}^* \). Thus, if \( s \leq 18\phi_{i,j}^* \), we have \( s + 0.2(s + \phi_{i,j}^*) + 0.4y_{i,j}p_{i,j} \leq 2^a \).

\[ \Box \]

**Observation 7.13.** For every \( \tau \in (2^a, p_{i,j}) \), we have \( h'_{i,j}(\tau) \geq \frac{17}{18} y_{i,j}^* \).

Proof. This comes from Observation 7.12. \( R'_{i,j,s} \) will cover \( \tau \) if \( s \leq 18\phi_{i,j}^* \). By the definition of \( \phi_{i,j} \) and Markov inequality, the sum of \( x_{i,j,s} \) over all such \( s \) is at least \( \frac{17}{18} y_{i,j}^* \).

\[ \Box \]

**Observation 7.14.** If \( \tau' \) is not contained in the interior of any interval in \( U \) and \( \tau \geq \tau' \), then \( \tau - \ell_U(\tau) \geq \min \{0.2\tau', \tau - \tau'\} \).

Proof. Consider the definition of \( \ell_U \) via the shifting of intervals. Since \( \tau' \) is not contained in the interior of any interval in \( U \), an interval in \( U \) is either to the left of \( \tau' \) or to the right of \( \tau' \). By the way we shifting intervals, any interval to the right of \( \tau' \) will be shifted by at least 0.2 \( \tau' \) distance to the right. Thus, the sub-intervals of the intervals in \( U \) from max \( \{\tau', \tau - 0.2\tau'\} \) to \( \tau \) will be shifted to the right of \( \tau \). The observation follows.

\[ \Box \]

**Observation 7.15.** If \( \tau \) is covered by some interval \( (s, s + p_{i,j}) \) in \( U \), then \( \tau - \ell_U(\tau) \geq \min \{0.2(s + \phi_{i,j}) + 0.4y_{i,j}p_{i,j}, \tau - s\} \).

Proof. The interval \( (s, s + p_{i,j}) \) will be shifted by \( 0.2(s + \phi_{i,j}) + 0.4y_{i,j}p_{i,j} \) distance to the right. So the sub-interval \( (\max \{\tau - 0.2(s + \phi_{i,j}) - 0.4p_{i,j}, s\}, \tau) \) will be shifted to the right of \( \tau \). The observation follows.

\[ \Box \]

Equipped with these observations, we can analyze the contribution of \( U \) case by case:

**Case 1:** \( (2^a, 0.92p_{i,j}^*) \) is not a sub-interval of any interval in \( U \). In this case, there is \( \tau' \in (2^a, 0.92p_{i,j}^*) \) that is not in the interior of any interval in \( U \). By Observation 7.14, any point in \( \tau \in (0.95p_{i,j}^*, \phi_{i,j}^*) \) has \( \tau - \ell_U(\tau) \geq \min \{\tau - \tau', 0.2\tau'\} \geq \min \{0.95p_{i,j}^* - 0.92p_{i,j}^*, 0.2 \times 2^a\} \geq \min \{0.03p_{i,j}^* - 0.2 \times 0.9p_{i,j}^*/4\} = 0.03p_{i,j}^* \). By Observation 7.13, any such \( \tau \) has \( h' \) value at least \( \frac{17}{18} y_{i,j}^* \). Thus, the contribution of \( U \) is \( D_U \geq z_U \times \frac{h'_{i,j}^*}{y_{i,j}^* \times p_{i,j}^*} \otimes (I - \ell_U) \geq z_U \times \frac{17y_{i,j}^*}{18y_{i,j}^* \times p_{i,j}^*} \times \frac{0.03p_{i,j}^*}{(0.05p_{i,j}^*)} \geq 0.0014z_U p_{i,j}^* \geq \frac{2p_{i,j}^*}{6000} \).

**Case 2:** \( (2^a, 0.92p_{i,j}^*) \) is covered by some interval \( (s, s + p_{i,j}) \) in \( U \), and \( 0.2(\phi_{i,j} + s) + 0.4y_{i,j}p_{i,j} \geq 0.02 \times 2^a \). Focus on any \( \tau \in (1.01 \times 2^a, 0.92p_{i,j}^*) \subseteq (2^a, p_{i,j}) \). By Observation 7.13, we have \( h'(\tau) \geq \frac{17}{18} y_{i,j}^* \). By Observation 7.15, we have \( \tau - \ell_U(\tau) \geq \min \{0.2(\phi_{i,j} + s) + 0.4y_{i,j}p_{i,j}, \tau - s\} \geq \min \{0.002 \times 2^a, 0.01 \times 2^a\} = 0.002 \times 2^a \). Thus, the contribution of \( U \) is \( \frac{D_U \geq z_U \times \frac{h'_{i,j}^*}{y_{i,j}^* \times p_{i,j}^*} \otimes (I - \ell_U) \geq z_U \times \frac{17y_{i,j}^*}{18y_{i,j}^* \times p_{i,j}^*} \times (0.002 \times 2^a) \times (0.92p_{i,j}^* - 1.01 \times 2^a) \geq (\frac{17}{18} \times 0.002 \times 0.9/4 \times (0.92 - 1.01 \times 0.9/2)) z_U p_{i,j}^* \geq 0.00019z_U p_{i,j}^* \geq \frac{2p_{i,j}^*}{6000} \) \), where we used 0.9 \( p_{i,j}^* \)/4 \( \leq 2^a \leq 0.9p_{i,j}^*/2 \).

**Case 3:** \( (2^a, 0.92p_{i,j}^*) \) is covered by some interval \( (s, s + p_{i,j}) \) in \( U \), and \( 0.2(\phi_{i,j} + s) + 0.4y_{i,j}p_{i,j} \leq 0.02 \times 2^a \). If \( j = j^* \), then \( D_U \geq z_U \times 0.1p_{i,j}^* \cdot c_U(j^*) \geq z_U \times 0.1p_{i,j}^* \). So, we assume \( j \neq j^* \). This is where we use the strong negative correlation between \( j \) and \( j^* \). We shall lower bound \( \Pr[j = j^*|a] \) and \( \Pr[j \neq j^*|a] \) separately.

Notice that \( 2^a \geq 0.9p_{i,j}^*/4 \geq 0.9 \times \frac{1}{6000} \phi_{i,j}^*/4 \geq 10\phi_{i,j}^* \), by the fact that \( j^* \) is bad on \( i \). Thus, \( (2^a, 2^{a+1}) \subseteq (10\phi_{i,j}^*, p_{i,j}^*) \), implying Property (7.4a) for \( j^* \). If \( s_{i,j} = s \) and \( R'_{i,j,s} \) covers
(2^a, 2^{a+1})$, then Property (7.4b) holds. By Observation 7.12, this happens with probability at least $\frac{17}{18}$. Thus, we have

$$\Pr\left[j^* \sim i \sim a\right] \geq \frac{17}{18} \cdot \frac{2^a}{p_{i,j}^*} \geq \frac{17}{18} \cdot \frac{0.9}{4} \geq 0.21.$$ 

Now, we continue to bound $\Pr\left[j^* \sim i \sim a\right]$. To do this, we need to first prove that $j$ is bad on $i$. Indeed, $\phi_{i,j} + y_{i,j}p_{i,j} \leq 5(0.2(\phi_{i,j} + s) + 0.4y_{i,j}p_{i,j}) < 5 \times 0.002 \times 2^a \leq 0.01p_{i,j}$, since $(s, s + p_{i,j}) \supseteq (2^a, 0.92p_{i,j}^*) \supseteq (2^a, 2^{a+1})$, which is of length at least $2^a$. This implies that $j$ is bad on $i$.

Then, $s \leq 0.002 \times 2^a/0.2 = 0.01 \times 2^a$ and $s + p_{i,j} > 0.92p_{i,j}^* \geq 0.92 \times 2^{a+1} > 2^{a+1} + 0.01 \times 2^a$. This implies that $p_{i,j} \geq 2^{a+1}$. Also, $2^a \geq 0.2f_{i,j}/0.002 = 100f_{i,j} \geq 10f_{i,j}$. Thus, Property (7.4a) holds.

For every $s' \leq 50f_{i,j}$, we have $1.2s' + 0.2\phi_{i,j} + 0.4y_{i,j}p_{i,j} \leq 60.2\phi_{i,j} + 0.4y_{i,j}p_{i,j} \leq \frac{60.2}{0.2} \times (0.2(\phi_{i,j} + s) + 0.4y_{i,j}p_{i,j}) \leq \frac{60.2}{0.2} \times 0.002 \times 2^a \leq 2^a$. Thus, Property (7.4b) holds.

Overall, $\Pr\left[j^* \sim i \sim a\right] \geq 0.98 \times \frac{0.9p_{i,j}^*}{4p_{i,j}} \geq 0.22p_{i,j}^* / p_{i,j}$. Thus, by Observation 7.6,

$$\Pr\left[j^* \sim i \sim a\right] \geq 0.8 \Pr\left[j^* \sim i \sim a\right] \Pr\left[j^* \sim i \sim a\right] \geq 0.8 \times 0.21 \times 0.22p_{i,j}^*/p_{i,j} \geq 0.036p_{i,j}^*/p_{i,j}.$$ 

Thus, the contribution of $U$ is $D_U \geq sz^U \frac{2}{2} \Pr\left[j^* \sim i \sim a\right]p_{i,j} \geq 0.018\zeta z^U p_{i,j}^* = \frac{z^U p_{i,j}^*}{6000}$. Thus, the left side of (38) is $\sum_U D_U \geq \frac{z^U p_{i,j}^*}{6000} = \frac{p_{i,j}^*}{6000}$. This finishes the proof of Lemma 7.10.

So, we always have $\mathbb{E}\left[C_{i,j^*}|i^* \sim j\right] \leq (1.5 - \frac{1}{6000}) (\phi_{i,j^*} + p_{i,j^*})$. Deconditioning on $j^* \sim i$, we have $\mathbb{E}\left[C_{i,j^*}\right] \leq (1.5 - \frac{1}{6000}) \sum_i y_{i,j^*} (\phi_{i,j^*} + p_{i,j^*}) = (1.5 - \frac{1}{6000}) C_j$. This finishes the proof of Theorem 1.5.

8 Handling Super-Polynomial $T$

In this section, we show how to handle the case when $T$ is super-polynomial in $n$ for the problems we studied. For $Q[p_{i,j} \sim j] w_j C_j$, the algorithm in Section 6 runs in polynomial time ($T$ is not defined). For $P[p_{i,j} = 1| j] w_j C_j$, $T$ is always polynomially bounded since all jobs have $p_j = 1$.

For $P[p_{i,j} \sim j] w_j C_j$, $R[p_{i,j} \sim j] w_j C_j$ and $O| \sum_j w_j C_j$, the way we handle super-polynomial $T$ is the same as that in [11]. [11] considers the problem $R|r_j|\sum_j w_j C_j$, i.e., the unrelated machine job scheduling with job arrival times. They showed how to efficiently obtain a $(1 + \epsilon)$ approximate LP solution that only contains polynomial number of non-zero variables. Since the problem they considered is more general than $R|\sum_j w_j C_j$ and their LP is also a generalization of our (LP|R||wC), their technique can be directly applied to our algorithm for $R|\sum_j w_j C_j$. Due to the precedence constraints, and the conflicts between operations of the same job, $P[p_{i,j} \sim j] w_j C_j$ and $O| \sum_j w_j C_j$ are incomparable to $R|\sum_j w_j C_j$. However, with a trivial modification, the technique in [11] can handle the constraints in the two problems as well. We omit the detail here since the analysis will be almost identical to that in [11].
References


