Lift and Project Algorithms for Precedence Constrained Scheduling to Minimize Completion Time

Shashwat Garg [*] Janardhan Kulkarni [†] Shi Li [‡]

Abstract

We consider the classic problem of scheduling jobs with precedence constraints on a set of identical machines to minimize the weighted completion time objective. Understanding the exact approximability of the problem when job lengths are uniform is a well known open problem in scheduling theory. In this paper, we show an *optimal algorithm* that runs in *polynomial time* and achieves an approximation factor of $(2 + \epsilon)$ for the weighted completion time objective when the number of machines is a constant. The result is obtained by building on the lift and project approach introduced in a breakthrough work by Levey and Rothvoss [15] for the makespan minimization problem.

1 Introduction

A classic problem in scheduling theory is as follows: We are given a set J of n jobs, where each job $j \in J$ has a processing length p_j and a weight w_j. The jobs have *precedence constraints*, which are given by a partial order "$\prec$". A constraint $j \prec j'$ requires that job j' can only start after job j is completed. The jobs need to be scheduled on a set of m *identical* machines. The goal is to schedule jobs *non-preemptively* on machines – that is, each job is assigned to a single time interval of length p_j on a single machine – respecting precedence constraints so as to optimize a certain objective function. One of the most widely studied objective function in the literature is that of minimizing the *sum of weighted completion times of jobs*. In the classic three field notation [1], the problem is denoted by $P|\text{prec}|\sum_j w_jC_j$. The problem has been studied quite extensively in the literature [9, 6, 5, 20, 25, 11, 14, 26, 2, 15, 20, 10, 4, 21] for decades. Despite this, large gaps remain in our understanding of the problem. The influential survey of Schuurman and Woeginger [23] lists the exact approximability and hardness of this problem as one of the top ten open problems in scheduling theory (Problem 9 in the list).

With precedence constraints, the weighted completion time objective is more general than the problem of minimizing *makespan*, which is the problem of minimizing the maximum completion time of jobs. This is true because one can create a dummy job of size 0 and weight 1 that must be processed after all jobs in J, whose weights are set to 0. Already in 1966, Graham [8] showed that any greedy non-idling scheduling is a $2 - 1/m$ approximation to the problem of minimizing makespan with precedence constraints on identical machines. As showed by Svensson [26], the approximation factor of 2 is the best possible even for the special case $P|\text{prec}, p_j = 1|C_{\max}$ of the problem where all jobs have unit size, under a variant of the Unique Game Conjecture (UGC) introduced by Bansal and Khot [2]. As minimizing weighted completion time generalizes the makespan objective function, the lower bound of 2 also extends to the problem $P|\text{prec}, p_j = 1|\sum_j w_jC_j$.

On the positive side, the best known approximation ratios for the problem of minimizing the sum of weighted completion times are $1 + \sqrt{2}$ for $P|\text{prec}, p_j = 1|\sum_j w_jC_j$, and $2 + 2\ln 2 + \epsilon$ for $P|\text{prec}|\sum_j w_jC_j$, both due to Li [18]. Both these results are obtained by rounding a time-indexed LP relaxation for the problem, and build on the ideas in the papers [20, 10, 4]. Although we do not know if the integrality gap of time indexed LP given by Li [18] is more than 2, due to technical reasons it seems difficult round the time indexed LP to obtain the optimal approximation factor of 2, achieving which remains an important open problem in scheduling theory.

In most real world applications, the number of ma-

[*]Eindhoven University of Technology, s.garg@tue.nl. A part of this work was done when the author was visiting MSR, Redmond.

[†]Microsoft Research, Redmond, jakul@microsoft.com

[‡]University at Buffalo, Buffalo, NY, shil@buffalo.edu. Work supported in part by NSF grants CCF-1566356 and CCF-1717134.

[1]In the $\alpha|\beta|\gamma$-notation introduced in [9], α denotes the machine environment. We consider the case where we have m identical machines. We use $\alpha = P$ to denote case where m is a part of the input, $\alpha = Pm$ to denote the case where m is fixed, and $\alpha = 1$ to denote the case $m = 1$. β denotes the set of features in the scheduling problem. In the paper, we always have prec $\in \beta$, indicating we have precedence constraints between jobs. We may have $p_j = 1 \in \beta$, indicating that all jobs have size 1. γ indicates objective value: we use $C_{\max}$ for the makespan objective and $\sum_j w_jC_j$ for the weighted completion time objective.

chines is typically much smaller than the number of jobs. Hence, a natural question is if one can design better approximation algorithms for the case where m is a constant, which is denoted by $Pm|\text{prec}|\sum_j w_jC_j$ in the three field notation. The question has attracted a lot of attention for the makespan objective. For the makespan problem, whether $Pm|\text{prec}, p_j = 1|C_{\max}$ is NP-hard or not for any $m \geq 3$, and whether $Pm|\text{prec}|C_{\max}$ is strongly NP-hard or not for any $m \geq 2$ are long-standing open problems. On the positive side, the recent breakthrough result of Levey and Rothvoss [15] gave an $(1 + \epsilon)$-approximation algorithm for the problem $Pm|\text{prec}, p_j = 1|C_{\max}$, with running time $\exp\left(\exp\left(O_{m,\epsilon}(\log^2 \log n)\right)\right)$, via Sherali-Adams lift of the natural LP relaxation of the problem to $\exp\left(O_{m,\epsilon}(\log^2 \log n)\right)$ levels. Later, Garg [7] made the result strictly quasi-polynomial time.

For the weighted completion time objective, the result of Bansal and Khot [2] showed that even the problem $1|\text{prec}, p_j = 1|\sum_j w_jC_j$, i.e, the problem where there is only 1 machine and all jobs have length 1, can not be approximated better than $2 - \epsilon$ under a variant of Unique Games Conjecture (UGC). On the positive side, a 2-approximation algorithm for $1|\text{prec}|\sum_j w_jC_j$ was given by Hall et al. [10]. However, the algorithm crucially uses the fact that there are no idle times in the schedule when there is only one machine. As soon as one goes beyond the single machine environment this is no longer true; hence, it is not known how they can be extended to the case where we have more (but constant number of) machines, even for the case where all job sizes are 1. In this paper we solve this problem by giving a $(2 + \epsilon)$-approximation algorithm for $Pm|\text{prec}, p_j = 1|\sum_j w_jC_j$, which is almost optimal due to the UGC-hardness of $2 - \epsilon$ for the problem $1|\text{prec}, p_j = 1|\sum_j w_jC_j$. The main result of the paper is the following.

Theorem 1.1. *For any $\epsilon > 0$, there is a $(2 + \epsilon)$-approximation for $Pm|prec, p_j = 1|\sum_j w_jC_j$, i.e, the problem of scheduling precedence constrained unit-length jobs on m identical machines to minimize the total weighted completion time, which runs in time $n^{2^{O((m/\epsilon)\log(1/\epsilon))}}$.*

The technically interesting aspect of the above result is that our algorithm differs from the previous approaches to weighted completion time objective, and is based on the LP-hierarchy approach (also called lift and project method) for the corresponding makespan problem introduced by Levey and Rothvoss [15]. However, we remark that our algorithm runs in polynomial time. In contrast, the algorithm of [15] runs in time slightly more than quasi-polynomial. Similar to [15, 3], our result shows the effectiveness of the LP-hierarchy approach in designing better approximation algorithms for scheduling problems. Further, as noted in [15], even after more than a decade of research on the topic, there are very few problems where LP-hierarchies are truly useful in designing better approximation algorithms. Our result adds to the literature on the topic, and we hope that some of the ideas here will be useful in resolving open problems concerning completion time related objective functions.

2 Our Techniques

Our result is obtained by rounding a LP-hierarchy solution of natural LP for the problem. Our techniques build on the work of [15], so we begin by first giving the high level overview of their algorithm for minimizing the makespan when jobs have unit lengths.

At the heart of analysis of the Levey-Rothvoss algorithm [15] for minimizing makespan is the following simple observation: if the maximum chain length of jobs in the set J is at most ϵT, where T is a guess of the optimal makespan, then Graham's algorithm already gives a $(1 + \epsilon)$-approximation. At a high-level, the algorithm in [15] uses the above observation in the following way: it partitions the input instance J into three sets $J_{\text{top}}, J_{\text{mid}}$ and J_{bot}. The jobs in the set J_{mid} can be ignored (or discarded) as it is mainly required for technical reasons. The partitioning is done, guided by an LP-hierarchy solution, satisfying following two properties: 1) The maximum chain length among J_{top} is small. 2) The precedence constraints across the jobs in the sets J_{top} and J_{bot} are *loose*, and can be easily satisfied. If one can *recursively schedule* the jobs in J_{bot} with $(1 + \epsilon)$ approximation factor, then such a schedule can be easily extended to include J_{top} by using a simple greedy algorithm. This is possible because of the two properties satisfied by our partitioning.

Similar to the Levey-Rothvoss algorithm [15], our algorithm is also based on solving the natural LP relaxation to the problem, lifted to some number r of rounds, partitioning jobs into top, middle and bottom jobs according to their fractional support, and then schedule them separately. However, one big difference between our algorithm and that of [15] is that our algorithm is not recursive: we solve each bottom instance directly. As a result, we only need one level of partitioning of the time horizon, and only condition on constant number of events. This leads to a polynomial time algorithm for $Pm|\text{prec}, p_j = 1|\sum_j w_jC_j$.

In a nutshell, our non-recursive algorithm is obtained by perfectly aligning the two 2-approximation algorithms, for the two special cases of $P|\text{prec}, p_j = 1|\sum_j w_jC_j$: the problem $1|\text{prec}, p_j = 1|\sum_j w_jC_j$ for weighted completion time objective but on a single machine, and the problem $P|\text{prec}, p_j = 1|C_{\max}$ on multiple machines but with the objective of minimizing the makespan. Coincidentally, both special cases admit 2-approximation algorithms [8, 10], and both factors of 2 are tight under some stronger version of UGC [26, 2]. For each bottom instance defined by a bottom interval I_o and the set $J^*_{\text{bot}}(o)$ of bottom jobs assigned to I_o, we solve the problem of scheduling $J^*_{\text{bot}}(o)$ in I_o to minimize the *makespan*, instead of the weighted completion time. We show that the loss caused by considering the makespan objective is small, since the interval I_o is very short. This is where the 2-approximation for $P|\text{prec}, p_j = 1|C_{\max}$ comes into place: using twice many time slots, we can schedule $J^*_{\text{bot}}(o)$ inside I_o.

Then we show that the top jobs can be inserted into the schedule without adding too many slots to each I_o. We used a simple property that the number of jobs with fractional completion time at most C in the LP solution is at most $2mC$, that is, twice the number of jobs that can be scheduled before or at C in an integral schedule. This is indeed the key property used to obtain the 2-approximation for $1|\text{prec}, p_j = 1|\sum_j w_jC_j$ in [10]. In order to insert top jobs, we also need the properties that the maximum chain-length formed by top jobs is small, and that there are no precedence constraints between top jobs and bottom jobs assigned to each interval I_o. The first one can be guaranteed by conditioning, where the second one can be guaranteed using some techniques similar to those in [15]. Similar to [15], we show that the middle jobs can be discarded at first then inserted back without increasing the objective value by too much.

Using the above techniques, we shall have an additive term of $w(J) \cdot \frac{O(1)\cdot T}{2^{\Omega(1/\epsilon)}}$ in the total weighted completion time (where T is the makespan of a "reasonable" schedule of J). This comes from the length of intervals in our partitioning. Our final algorithm is based on decomposing the original instance into many sub instances, each containing a set of jobs with similar fractional completion times. We solve each sub-instance separately; the additive terms from these sub-instances become multiplicative in the end.

Due to some technical reasons, our final algorithm for the original instance J is not based on the Sherali-Adams lift of the natural LP relaxation. Rather, we apply the "round-or-cut" framework that has been used in many recent results [1, 16, 17]. We only solve the non-lifted LP relaxation for the instance J, and given a solution x to the LP, the rounding algorithm either rounds x to an integral schedule of cost at most $(2+\epsilon)$ times the cost of x, or reports that x violates some valid linear constraint. The rounding algorithm works by decomposing the instance J into many sub-instances J^* using x, and solving each instance J^* separately using the Sherali-Adams hierarchy based algorithm. It returns a constraint violated by x if the lifted LP relaxation for some J^* is infeasible.

3 Basics of Sherali-Adams Hierarchy

In this section, we state some basic facts about Sherali-Adams hierarchy that we will need. We refer the reader to [13, 24, 12, 19, 22] for an extensive introduction to hierarchies. Assume we have a linear program without an objective function: $Ax \le b$. The set of feasible solutions is defined as $\mathcal{X} = \{x \in \{0,1\}^n : Ax \le b\}$. It is convenient to think of each $i \in [n]$ as an event, and in a solution $x \in \{0,1\}^n$, x_i indicates whether the event i happens or not.

The idea of Sherali-Adams hierarchy is to strengthen the original LP $Ax \le b$ by adding more variables and constraints. Of course, each $x \in \mathcal{X}$ should still be a feasible solution to the strengthened LP (when extended to a vector in the higher-dimensional space). For some $r \ge 1$, the r-th round of Sherali-Adams lift of the linear program has variables x_S, for every $S \subseteq [n]$ of size at most r. For every solution $x \in \mathcal{X}$, x_S is supposed to indicate whether all the events in S happen or not in the solution x; that is, $x_S = \prod_{i\in S} x_i$. Thus each $x \in \mathcal{X}$ can be naturally extended to a 0/1-vector in the higher-dimensional space defined by all the variables.

To derive the set of constraints, let us focus on the j-th constraint $\sum_{i=1}^n a_{j,i}x_i \le b_j$ in the original linear program [2]. Consider two subsets $S, T \subseteq [n]$ such that $|S| + |T| \le r - 1$. Then the following constraint is valid for $\mathcal{X}$; i.e, all $x \in \mathcal{X}$, the constraint is satisfied:

$$\prod_{i\in S} x_i \prod_{i\in T}(1-x_i)\left(\sum_{i=1}^n a_{j,i}x_i - b_j\right) \le 0.$$

To *linearize* the above constraint, we expand the left side of the above inequality and replace each monomial with the corresponding $x_{S'}$ variable. Then, we obtain the following linear constraint:

(3.1)
$$\sum_{T'\subseteq T} (-1)^{|T'|}\left(\sum_{i=1}^n a_{j,i}x_{S\cup T'\cup\{i\}} - b_j x_{S\cup T'}\right) \le 0.$$

The r-th round of Sherali-Adams lift contains the above constraint for all j, S, T such that $|S|+|T| \le r-1$, and the trivial constraint that $x_\emptyset = 1$. For a linear program $\mathcal{P}$ and an integer $r \ge 1$, we use $\text{SA}(\mathcal{P}, r)$ to denote the r-th round Sherali-Adams lift of $\mathcal{P}$. We

[2]We assume that $x_i \ge 0, x_i \le 1, \forall i \in [n]$ are also in the set $Ax \le b$ of constraints.

also view $\mathcal{P}$ (resp. $\mathrm{SA}(\mathcal{P}, r)$) as the polytope of feasible solutions to the linear program $\mathcal{P}$ (resp. $\mathrm{SA}(\mathcal{P}, r)$). For every $i \in [n]$, we identify the variable x_i in the original LP and $x_{\{i\}}$ in a lifted LP.

A simple observation is that $x_{S_1} \geq x_{S_2}$ if $S_1 \subseteq S_2$, for a valid solution $x \in \mathrm{SA}(\mathcal{P}, r)$ and $|S_1| \leq |S_2| \leq r$. Consider the case where $S_2 = S_1 \cup \{i\}$ for some $i \notin S_1$. Linearizing the constraint $x_i \leq 1$ multiplied by $\sum_{i' \in S_1} x_i$ gives the constraint $x_{S_2} \leq x_{S_1}$. This implies, all the variables have values in $[0,1]$ as $x_\emptyset = 1$.

Conditioning Let $x \in \mathrm{SA}(\mathcal{P}, r)$ for some linear program $\mathcal{P}$ on n variables and $r \geq 2$. Let $i \in [n]$ be an event such that $x_i > 0$; then we can define a solution $x' \in \mathrm{SA}(\mathcal{P}, r-1)$ obtained from x by "conditioning" on the event i. For every $S \subseteq [n]$ of size at most $r-1$, x'_S is defined as

$$x'_S := \frac{x_{S \cup \{i\}}}{x_i}.$$

Observation 3.1. *Let x' be obtained from $x \in \mathrm{SA}(\mathcal{P}, r)$ by conditioning on some event i, for some $r \geq 2$. Then $x' \in \mathrm{SA}(\mathcal{P}, r-1)$ and $x'_i = 1$.*

Proof. By definition of the conditioning operation, we have $x'_i = \frac{x_{\{i\} \cup \{i\}}}{x_i} = \frac{x_i}{x_i} = 1$, and $x'_\emptyset = \frac{x_{\emptyset \cup \{i\}}}{x_i} = \frac{x_i}{x_i} = 1$. The constraint (3.1) on x' for j, S and T is implied by (3.1) on x for $j, S \cup \{i\}$ and T.

Observation 3.2. *Let $x \in \mathrm{SA}(\mathcal{P}, r)$ for some $r \geq 2$ and $x' \in \mathrm{SA}(\mathcal{P}, r-1)$ be obtained from x by conditioning on some event i. Then, if $x_{i'} \in \{0,1\}$ for some $i' \in [n]$, then $x'_{i'} = x_{i'}$.*

Proof. If $x_{i'} = 0$, then $x'_{i'} = \frac{x_{\{i'\} \cup \{i\}}}{x_i} = 0$ since $x_{\{i'\} \cup \{i\}} \leq x_{i'} = 0$. Consider the case $x_{i'} = 1$. Expanding the constraint $(1 - x_i)(1 - x_{i'}) \geq 0$ gives the constraint $1 - x_i - x_{i'} + x_{\{i', i\}} \geq 0$. This implies $x_i = x_{i', i}$. Thus, $x'_{i'} = \frac{x_{\{i, i'\}}}{x_i} = 1$.

The observation says that once an event i' happens with extension 0 or 1 w.r.t lifted solution x, then it will always happen with the same extension (0 or 1) w.r.t any solution x' obtained from x by conditioning. To understand the conditioning operation and the above observations better, it is useful to consider the ideal case where x corresponds to a convex combination of integral solutions in $\mathcal{X}$. Then we can view x as a distribution over $\mathcal{X}$. Then, conditioning on the event i over the solution x corresponds to conditioning on i over the distribution x.

4 Overview of $(2 + \epsilon)$-Approximation for $Pm|\mathbf{prec}, p_j = 1|\sum_j w_j C_j$

In this section we provide an overview of the $(2 + \epsilon)$-approximation algorithm for the weighted completion time problem, give some useful definitions and lemmas and describe the natural LP relaxation. The detailed algorithm will be given in Sections 5 and 6.

Let $k = O(1/\epsilon)$ be large enough and then our goal becomes to obtain a $\left(2 + \frac{O(1)}{k}\right)$-approximation algorithm. Let $J^* \subseteq J$ be the set of jobs we need to schedule (due to the decomposition we shall discuss soon, J^* is not necessarily J). Let T be an upper bound on the makespan of any reasonable schedule of J^*; then we require that all the jobs to be scheduled in $[T]$. Let Φ be a guess of the value of the weighted completion time of the jobs in the optimum schedule of J^*. We first solve Sherali-Adams lift of the natural LP relaxation of the problem to $r = 2^{O(mk \log k)}$ many levels to obtain a solution x. Our main rounding algorithm (described in Section 5) then returns a schedule whose total weighted completion time is at most $\left(2 + \frac{O(1)}{k}\right)\Phi + \frac{O(1) \cdot w(J^*) \cdot T}{2^k}$. This does not immediately lead to a $\left(2 + \frac{O(1)}{k}\right)$-approximation, due to the additive term $\frac{O(1) \cdot w(J^*) \cdot T}{2^k}$. We show (in Section 6) that the additive term can be removed by decomposing the original instance J into many sub-instances J^* and running the main rounding algorithm on each J^*. The main rounding algorithm works as follows.

- By conditioning on a constant number of events in the solution x we arrive at a solution x^*, in which we guarantee that the maximum length of a chain formed by "flexible jobs" is small; these are the jobs that could become top jobs later. From now on, we only need to focus on the solution x^* to the original LP relaxation. (See Section 5.1.)

- We define a *random* partition $\mathcal{I} = \{I_1, I_2, \cdots, I_h\}$ of $[T]$ where all intervals in $\mathcal{I}$ except I_1 and I_h have length L, where $T/L = O_{m,k}(1)$. We then divide J^* into top, middle and bottom jobs (denoted as $J^*_{\text{top}}, J^*_{\text{mid}}$ and J^*_{bot} respectively), according to the number of intervals in $\mathcal{I}$ that the support (under a careful definition of "support") of each job j in x intersects: j is a top job if the number is large, a bottom job if the number is 1 and a middle job otherwise. A top job must be a flexible job, implying that the maximum length of a chain formed by top jobs is small. The probability that a job becomes a middle job is small, allowing us to ignore them most of the time. (See Section 5.2.)

- We create equal-length intervals $\mathcal{I}' = \{I'_1, I'_2, \cdots, I'_h\}$, with length $L' = \left(2 + \frac{O(1)}{k}\right)L$, that is, about twice the length of a typical interval in $\mathcal{I}$. We then assign $J^*_{\text{top}} \cup J^*_{\text{bot}}$ to intervals in $\mathcal{I}'$ and we shall schedule jobs assigned to each I'_o

in I'_o. Let o_j be the index of the interval a job $j \in J^*_{\text{top}} \cup J^*_{\text{bot}}$ is assigned to; we shall use $J^*_{\text{top}}(o)$ and $J^*_{\text{bot}}(o)$ to denote set of top and bottom jobs assigned to I'_o respectively. For a job $j \in J^*_{\text{bot}}$, o_j is defined as the index o such that I_o contains the support of j in x^* (Section 5.3). The o_j's for top jobs are defined to satisfy the following properties. (a) The assignment agrees with the precedence constraints. (b) The interval I_{o_j} is not late compared to time C_j, the completion time of j in the solution x^*; this will guarantee that the completion time of j in the final schedule is at most $\left(2 + \frac{O(1)}{k}\right) C_j$. (c) For each $o \in [h]$, we have $|J^*_{\text{top}}(o) \cup J^*_{\text{bot}}(o)| \leq \left(2 + \frac{O(1)}{k}\right) L$ so that there is enough capacity in the interval I'_o. (d) There are no precedence constraints between $J^*_{\text{top}}(o)$ and $J^*_{\text{bot}}(o)$ for each $o \in [h]$; this will be used when we schedule $J^*_{\text{top}}(o) \cup J^*_{\text{bot}}(o)$ in I'_o. (See Section 5.4.)

- We show that jobs in $J^*_{\text{bot}}(o) \cup J^*_{\text{top}}(o)$ can be scheduled inside I'_o. We construct the schedule in two steps. First we use Graham's list scheduling algorithm to schedule jobs $J^*_{\text{bot}}(o)$ in I'_o; this can be done since the algorithm is a 2-approximation and we have $|I'_o| \geq 2|I_o|$. $J^*_{\text{top}}(o)$ can be greedily inserted so that the makespan of the schedule for $J^*_{\text{top}}(o) \cup J^*_{\text{top}}(o)$ is at most L'. This uses the properties that $|J^*_{\text{top}}(o) \cup J^*_{\text{bot}}(o)| \leq \left(2 + \frac{O(1)}{k}\right) L$, the maximum length of a chain of jobs in $J^*_{\text{top}}(o)$ is small, and there are no precedence constraints between $J^*_{\text{top}}(o)$ and $J^*_{\text{bot}}(o)$. (See Section 5.5.)
- We then define an $o_j \in [h]$ for each job $j \in J^*_{\text{mid}}$ and insert j to I'_{o_j} (Section 5.6). Unlike top and bottom jobs, we may need to increase the length of I'_{o_j} by 1 in order to insert a job $j \in J^*_{\text{mid}}$ to I'_{o_j}. We restrict that each o_j is not too small or too big; it should be consistent with the fractional support of j according to x^*. On one hand, this guarantees that the completion time of a middle job j is at most $2C_j$. On the other hand, the increments of completion times of other jobs due to the insertion of middle jobs are not too large. (See Section 5.7.)

As we mentioned, using the above main algorithm, the expected total weighted completion time of the output schedule has an additive term of $\frac{O(1) \cdot w(J^*) \cdot T}{2^k}$. This comes from the length L of intervals in $\mathcal{I}$. In order to obtain a multiplicative $(2 + \epsilon)$-approximation, we decompose the given instance into many sub instances, and solve each sub instance using the main rounding algorithm. Roughly speaking, we partition J into many groups, where a group ℓ contains the jobs j whose fractional completion time C_j is between $2^{\ell-1}$ and 2^ℓ. Then, we discard one out of every k groups. For each set of (at most) $k-1$ consecutive groups that are not discarded, we create an instance J^* containing jobs in these groups and use the main rounding algorithm to solve the instance J^*. For each discarded group, we also construct a reasonable schedule of jobs in the group. Then, the final schedule is obtained by combining all these sub-schedules carefully. The jobs assigned to a same instance J^* have similar fractional completion times and the additive term can be bounded using multiplicative ones. Moreover, the influence of jobs from instances for lower-indexed groups is small.

For each instance J^* in the decomposition, we need to know the optimum weighted completion time Φ_{J^*} for scheduling J^*, in order to formulate the lifted LP relaxation for J^* using the Sherali-Adams hierarchy. Since we do not know the decomposition of J at the beginning, we cannot not solve the problem using a single lifted LP relaxation for the instance J. Rather, our algorithm is based on the "rounding-or-cut" framework. We solve the non-lifted LP relaxation for J to obtain a solution x, and use x to decide the decomposition of J into sub-instances. Then for each J^* in the decomposition, we define Φ_{J^*} to be the weighted completion time for J^* according to x, and formulate the lifted LP relaxation for J^* using this Φ_{J^*} value. If the lifted LP relaxation is infeasible, then we know that the weighted completion time for scheduling J^* is strictly bigger than Φ_{J^*} and thus we can return a constraint that x violated. Suppose the lifted LP relaxations are feasible for all J^*. Then the rounding algorithm can obtain good schedules for all these J^* and combine them into a final good schedule for J. The decomposition algorithm and its analysis are described in Section 6.

4.1 Useful Definitions and Lemmas

Before formally describing the algorithm, we give some useful definitions and lemmas, and then describe the natural LP relaxation for the problem. Given a subset $J' \subseteq J$ of jobs, we use $\Delta(J')$ to the denote the maximum length of a precedence chain of jobs in J'. We define $w(J') = \sum_{j \in J'} w_j$ as the total weight of jobs in J'.

Reasonable Schedule A schedule for $J' \subseteq J$ is a function $\mathcal{S} \in \mathbb{Z}^{J'}_{>0}$. The makespan of a schedule $\mathcal{S}$ for J' is defined as $\max_{j \in J'} \mathcal{S}_j$. Given a schedule $\mathcal{S}$, and a time slot $t \in \mathbb{Z}_{>0}$, we define $\mathcal{S}^{-1}(t) = \{j \in J' : \mathcal{S}_j = t\}$ to be the set of jobs scheduled at time slot t. A schedule $\mathcal{S}$ is said to be valid if

- (capacity constraints) for every $t \in \mathbb{Z}_{>0}$ we have $|\mathcal{S}^{-1}(t)| \leq m$,
- (precedence constraints) for every pair $j, j' \in J'$ of

jobs such that $j \prec j'$, we have $\mathcal{S}_j < \mathcal{S}_{j'}$.

DEFINITION 4.1. *A valid schedule $\mathcal{S}$ for some $J' \subseteq J$ is said to be reasonable if for every $j \in J'$, any schedule obtained from $\mathcal{S}$ by decreasing $\mathcal{S}_j$ is not a valid one.*

If a schedule $\mathcal{S}$ for J' is not reasonable, then we can decrease the scheduling time of some job $j \in J'$ without violating the validity of the schedule. This does not increase the completion time of any job. Thus, without loss of generality, we can assume that the schedule we are looking for is a reasonable schedule. Graham's result [8] shows that any reasonable schedule for J' gives a 2-approximation for the problem of scheduling J' to minimize the makespan. This analysis is recovered by our Corollary 4.6; indeed, our analysis requires the strong lemma (Lemma 4.4).

DEFINITION 4.2. *For any reasonable schedule $\mathcal{S}$ for some subset $J' \subseteq J$, we say a slot t within the makespan of $\mathcal{S}$ is busy if $|\mathcal{S}^{-1}(t)| = m$ and idle otherwise.*

CLAIM 4.3. *Let $\mathcal{S}$ be a reasonable schedule for some $J' \subseteq J$. Let $j \in J'$ and let t be the latest idle slot that is before $\mathcal{S}_j$ (we assume t exists). Then there is a job $j' \in J'$ such that $\mathcal{S}_{j'} = t$ and $j' \prec j$.*

Proof. Since $\mathcal{S}$ is reasonable, changing $\mathcal{S}_j$ to t will make $\mathcal{S}$ invalid. It must be the case that some precedence constraint is violated since t is idle. So, there is a job $j_1 \in J'$ with $j_1 \prec j$ and $t \le \mathcal{S}_{j_1} < \mathcal{S}_j$. If $\mathcal{S}_{j_1} = t$ then let $j' = j_1$ and we are done. Otherwise, $t < \mathcal{S}_{j_1} < \mathcal{S}_j$. Then changing $\mathcal{S}_{j_1}$ to t will make $\mathcal{S}$ invalid since $\mathcal{S}$ is reasonable. There must be a job $j_2 \in J'$ with $j_2 \prec j_1$ and $t \le \mathcal{S}_{j_2} < \mathcal{S}_{j_1}$. If $\mathcal{S}_{j_2} = t$ then let $j' = j_2$ and we are done. Otherwise, $t < \mathcal{S}_{j_2} < \mathcal{S}_{j_2}$. So, we can repeat this process and eventually we can find a job $j' \in J'$ with $\mathcal{S}_{j'} = t$ and $j' \prec j$. $\square$

LEMMA 4.4. *Let $\mathcal{S}$ be a reasonable schedule for some $J' \subseteq J$. Let $J'' \subseteq J'$ be some subset of jobs such that there are no precedence constraints between J'' and $J' \setminus J''$. Then*

$$\max_{j \in J''} \mathcal{S}_j \le \left\lfloor \frac{|J'|}{m} \right\rfloor + \Delta(J'').$$

Proof. Notice that the total number of busy slots in $\mathcal{S}$ is at most $\left\lfloor \frac{|J'|}{m} \right\rfloor$. Now focus on a job $j \in J''$ and we count the total number of idle slots before or at $\mathcal{S}_j$. Let $t_1, t_2, \cdots, t_g$ be the sequence of idle slots before $\mathcal{S}_j$, with $t_1 < t_2 < \cdots < t_g < \mathcal{S}_j$. Then, by Claim 4.3, there is a job $j_g \in J'$ such that $j_g \prec j$ and $\mathcal{S}_{j_g} = t_g$. Applying the claim again to j_g, there is a job $j_{g-1} \in J'$ such that $j_{g-1} \prec j_g$ and $\mathcal{S}_{j_{g-1}} = t_{g-1}$. Repeatedly applying the claim, we can find a chain of jobs $j_1 \prec j_2 \prec \cdots \prec j_g \prec j$ in J' and $\mathcal{S}_{j_{g'}} = t_{g'}$ for every $g' \in [g]$. Notice that we assumed there are no precedence constraints between J'' and $J' \setminus J''$ and $j \in J''$. Thus, all the jobs in the chain are in J'', implying, $g + 1 \le \Delta(J'')$. Thus, the total number of idle slots before or at t is at most $g + 1 \le \Delta(J'')$. The lemma follows. $\square$

DEFINITION 4.5. *We define $T_{J'} = \left\lfloor \frac{|J'|}{m} \right\rfloor + \Delta(J')$ for every $J' \subseteq J$.*

COROLLARY 4.6. *Any reasonable schedule $\mathcal{S}$ for some $J' \subseteq J$ has makespan at most $T_{J'}$.*

Proof. The corollary follows by applying Lemma 4.4 with $J'' = J'$, since there are no precedence constraints between J'' and $J' \setminus J'' = \emptyset$. $\square$

Notice that both $\left\lfloor \frac{|J'|}{m} \right\rfloor$ and $\Delta(J')$ are lower bounds on the makespan of any valid schedule for J'. Thus, any reasonable schedule gives a 2-approximation for the makespan minimization problem.

4.2 LP Relaxation Consider the problem of scheduling a set $J' \subseteq J$ on m machines. Let Φ be an upper bound on the optimum weighted completion time. Then the natural LP relaxation contains the following constraints:

$$\sum_{t \in [T_{J'}]} x_{j,t} = 1, \qquad \forall j \in J' \tag{4.2}$$

$$\sum_{j \in J'} x_{j,t} \le m, \qquad \forall t \in [T_{J'}] \tag{4.3}$$

$$\sum_{j \in J', t \in [T_{J'}]} w_j x_{j,t} t \le \Phi \tag{4.4}$$

$$\sum_{s \in [t-1]} x_{j,s} \ge \sum_{s \in [t]} x_{j',s}, \quad \forall j \prec j' \in J', t \in [T_{J'}] \tag{4.5}$$

$$x_{j,t} \ge 0, \qquad \forall j \in J', t \in [T_{J'}] \tag{4.6}$$

In the above definition, each variable $x_{j,t}$ indicates whether job j is scheduled at the time slot t. Since we are looking for a reasonable schedule, we can restrict the makespan of our schedule to be $T_{J'}$. We now argue that Constraints (4.2), (4.3), (4.5) and (4.6) are valid and define a valid linear program for the problem of scheduling J'. (4.2) requires all jobs in J' to be scheduled, (4.3) requires that the total number of jobs scheduled at any time slot t is at most m. (4.5) captures the precedence constraints: for any pair of jobs $j \prec j'$ and time slot t, j' is scheduled at or before time t, only if j is scheduled at or before time $t - 1$. (4.6) enforce the non-negativity constraints.

The objective we want to minimize is the weighted completion time $\sum_{j\in J',t\in[T_{J'}]} w_j x_{j,t} t$. As is common in the LP/SDP hierarchy framework, we capture the objective of the optimization problem as an LP constraint, as required by (4.4). So, the LP does not have an objective function and we are only interested in its feasibility. When applying the lift-and-project framework, (4.4) will be lifted together with the other constraints.

For every $J' \subseteq J$ of jobs and $\Phi \geq 0$, we use $\mathcal{P}_{J'}(\Phi)$ to denote the above LP, as well as the polytope of feasible solutions $(x_{j,t})_{j\in J',t\in[T_{J'}]}$ to the LP.

Claim 4.7. *Let $x \in \mathcal{P}_{J'}(\Phi)$ for some $\Phi \geq 0$ and $J' \subseteq J$. Let $C_j = \sum_{t\in[T_J]} x_{j,t} t$ for every $j \in J'$ be the fractional completion time of j according to x. Then $j \prec j'$ implies $C_{j'} \geq C_j + 1$.*

Proof. The proof follows from the constraints of our LP relaxation. We have,

$$\begin{aligned}
C_j &= \sum_{s\in[T_{J'}]} x_{j,s} s = \sum_{s=1}^{T_{J'}}\sum_{t=1}^{s} x_{j,s} \\
&= \sum_{t=1}^{T_{J'}}\sum_{s=t}^{T_{J'}} x_{j,s} = \sum_{t=1}^{T_{J'}}\left(1 - \sum_{s\in[t-1]} x_{j,s}\right) \\
&\leq \sum_{t=1}^{T_{J'}}\left(1 - \sum_{s\in[t]} x_{j',s}\right) = \sum_{t=1}^{T_{J'}}\sum_{s=t+1}^{T_{J'}} x_{j',s} \\
&= \sum_{s=1}^{T_{J'}} (s-1) x_{j',s} = C_{j'} - 1. \qquad \square
\end{aligned}$$

The following lemma bounds $T_{J'}$ using the fractional completion times of jobs in J'. It will be used later in Section 6 to remove the additive term.

Lemma 4.8. *Let $x \in \mathcal{P}_J(\Phi)$ for some $\Phi \geq 0$. Let $C_j = \sum_{t\in[T_J]} x_{j,t} t$ for every $j \in J$. Then, for every subset $J' \subseteq J$, we have $T_{J'} \leq 3 \cdot \max_{j\in J'} C_j$.*

Proof. Recall that $T_{J'} = \left\lfloor \frac{|J'|}{m} \right\rfloor + \Delta(J')$. We upper bound $\frac{|J'|}{m}$ and $\Delta(J')$ using $C_{\max} := \max_{j\in J'} C_j$ respectively. First, for every two jobs $j \prec j'$ we have $C_{j'} \geq C_j + 1$, by Claim 4.7. Thus $C_{\max} \geq \Delta(J')$. Then we shall prove $|J'| \leq 2mC_{\max}$, which implies $\left\lfloor \frac{|J'|}{m} \right\rfloor \leq 2C_{\max}$ and thus the lemma. To see this, for every $j \in J'$ and t, we draw a rectangle with height $x_{j,t}$ and horizontal span $(t-1, t]$ in the 2D plane. Then the total area of rectangles for each $j \in J'$ is exactly 1, and the horizontal position of the mass center for these rectangles is exactly $C_j - 1/2$. Thus overall, the horizontal position of the mass center for all rectangles for jobs in J' is at most $C_{\max}$. Notice that for each real number $\tau > 0$, the total height of rectangles horizontally covering τ is at most m by Constraint (4.3). That means that $|J'| =$ the total area of rectangles for J', and is at most $2mC_{\max}$.

5 The Main Rounding Algorithm

In this section, we give our main rounding algorithm, by proving the following theorem:

Theorem 5.1. (Main Theorem) *Let $k \geq 5$ be an integer, $r = 2^{O(mk\log k)}$ be a big enough integer, $J^* \subseteq J$ and $\Phi > 0$. Let $x \in \mathrm{SA}(\mathcal{P}_{J^*}(\Phi), r)$. Then in running time $n^{O(r)}$, we can find a valid schedule of J^* with total weighted completion time at most $\left(2 + \frac{O(1)}{k}\right)\Phi + \frac{O(1)}{2^k} \cdot w(J^*) \cdot T_{J^*}$.*

Due to the additive term $\frac{O(1)}{2^k} \cdot w(J^*) \cdot T_{J^*}$, we do not obtain a multiplicative-$\left(2 + \frac{O(1)}{k}\right)$-approximation immediately using the above theorem. As we mentioned, in Section 6 we shall achieve this goal by decomposing the set J of jobs into many subsets and applying Theorem 5.1 for each subset J^* in the decomposition.

For notational convenience, we use T for T_{J^*} and $\mathcal{P}(\Phi)$ for $\mathcal{P}_{J^*}(\Phi)$. We define two important global parameters that will be used across the whole section:

- $k_1 = mk(2\lceil \log k \rceil + 3) = \Theta(mk\log k)$;
- $u = \left\lfloor \frac{T}{2^{k_1+k}\cdot k} \right\rfloor$.

We assume that $u \geq k$. Otherwise we have $mT < m \cdot 2^{k_1+k} \cdot k = 2^{O(mk\log k)}$ and we can allow $r > mT \geq n$; then the polytope $\mathrm{SA}(\mathcal{P}(\Phi), r)$ is exact and we can find a valid schedule of J^* with weighted completion time at most Φ.

5.1 Reducing the Maximum Chain Length of Flexible Jobs

In this section, we condition on some events so that there are no long chains of flexible jobs, defined as follows.

Definition 5.2. *Let $x \in \mathrm{SA}(\mathcal{P}(\Phi), r')$ for some $r' \geq 1$. Then for every $j \in J^*$, we define $B^x_j = \min\{t : x_{j,t} > 0\}$ and $E^x_j = \max\{t : x_{j,t} > 0\}$ be the minimum and maximum time slot t in which j is scheduled with a positive fraction according to x. We say j is flexible w.r.t x if $E^x_j - B^x_j \geq k^2 u$.*

The definition is to guarantee that all top jobs (defined later) will be flexible. Then main lemma we prove in the section is the following:

Lemma 5.3. *If the parameter r in Theorem 5.1 is large enough, then in $n^{O(r)}$ time we can find a solution $x^* \in$*

$\mathcal{P}(\Phi)$, *such that*

$$\Delta\big(\{j \in J^* : j \text{ is flexible w.r.t } x^*\}\big) \leq \frac{u}{2^k}. \tag{5.7}$$

Proof. We start from $r' = r$ and $x^* = x \in \mathrm{SA}(\mathcal{P}(\Phi), r')$. While (5.7) does not hold, we apply the following procedure. Take a $\lceil \frac{u}{2^k} \rceil$-length chain Q of flexible jobs w.r.t x^*. Notice for each j in Q, $[B_j^{x^*}, (B_j^{x^*} + E_j^{x^*})/2]$ has length at least $k^2u/2$. Thus there is a real number $\tau \in [1, T]$ and a set $J' \subseteq Q$ of at least $\frac{(k^2u/2)\lceil u/2^k \rceil}{T-1} \geq \frac{k^2u^2}{2^{k+1}T}$ jobs, such that $\tau \in \big[B_j^{x^*}, (B_j^{x^*} + E_j^{x^*})/2\big]$ for every $j \in J'$. Let $j' \in J'$ be the job that appears the last in the chain Q and let $t = B_{j'}^{x^*}$. Notice that by the definition of $B_{j'}^{x^*}$, we have that $x^*_{j',t} > 0$. So we can condition on the event $x^*_{j',t} = 1$. We update x^* to be the solution in $\mathrm{SA}(\mathcal{P}(\Phi), r'-1)$ under this conditioning, and we then update r' to $r' - 1$. After this operation, all jobs $j \in J'$ will have $E_j^{x^*} \leq t \leq \tau$, since all these jobs precede j' and the scheduling time of j' is fixed to t in x^*. Also, notice that the conditioning can only increase $B_j^{x^*}$ since the set of time slots t with $x^*_{j,t} > 0$ can only shrink. Thus, for every job $j \in J'$, $E_j^{x^*} - B_j^{x^*}$ is decreased by at least a factor of 2 by the conditioning. Moreover, for any job $j \notin J'$, the conditioning can only increase $B_j^{x^*}$ and decrease $E_j^{x^*}$ and thus $E_j^{x^*} - B_j^{x^*}$ can only go down.

Initially, $E_j^{x^*} - B_j^{x^*} \leq T - 1$. We can decrease $E_j^{x^*} - B_j^{x^*}$ by at least a factor 2 for at most $\lceil \log(T/k^2u) \rceil$ times before it becomes less than k^2u. Since J' has size at least $\frac{k^2u^2}{2^{k+1}T}$ in each iteration, the number of iterations for the above procedure is at most

$$\frac{|J^*| \cdot \lceil \log(T/k^2u) \rceil}{\frac{k^2u^2}{2^{k+1}T}} \leq 2^{k+1}T \cdot \frac{mT \cdot \lceil \log(T/k^2u) \rceil}{k^2u^2}$$
$$= \frac{2^{k+1}m}{k^2} \cdot \frac{T^2}{u^2} \cdot \left\lceil \log \frac{T}{k^2u} \right\rceil.$$

Since $\frac{T}{u} = O(2^{k_1+k} \cdot k)$ and $k_1 = \Theta(mk \log k)$, the right side of the above sequence is at most $m \cdot 2^{O(k_1)} = 2^{O(mk \log k)}$. Thus, if we make $r = 2^{O(mk \log k)}$ to be large enough, then we can apply the above procedure until (5.7) holds, before we reduce r' to 1. By projecting the x^* to the subspace defined by the level-1 variables, we obtain an solution $x^* \in \mathcal{P}(\Phi)$. The running time of the algorithm is $n^{O(r)}$.

Thus, we have now an $x^* \in \mathcal{P}(\Phi)$ such that (5.7) happens. The proof of Lemma 5.3 is the only place where we use the conditioning. From now on, we shall fix this $x^* \in \mathcal{P}(\Phi)$ and simply use B_j and E_j for $B_j^{x^*}$ and $E_j^{x^*}$. That is, $B_j = \min\{t : x^*_{j,t} > 0\}$ and $E_j = \max\{t : x^*_{j,t} > 0\}$ for every $j \in J^*$.

Definition 5.4. *For every $j \in J^*$, let $C_j = \sum_t x^*_{j,t}t$ be the fractional completion time of j according to x^* and $D_j = \min\{B_j + k(C_j - B_j), E_j\}$.*

Notice that we have $B_j \leq C_j \leq D_j \leq E_j$. For a technical reason, we shall use $[B_j, \lceil D_j \rceil]$ as the support of j when we define top, middle and bottom jobs. Also notice that for two jobs $j \prec j' \in J^*$, we have $B_{j'} \geq B_j + 1, C_{j'} \geq C_j + 1$ and $E_{j'} \geq E_j + 1$; however $D_{j'} \geq D_j + 1$ may not hold.

5.2 Generating a Random Partition $\mathcal{I}$ and Defining Bottom, Middle and Top Jobs To generate the partition $\mathcal{I}$, we first partition $[T]$ into chunks of size u. That is, each chunk is of the form $((i-1)u, iu] \cap [T]$ for some integer i. Let p be an integer chosen uniformly at random between 0 and $k_1 - 1$ and let q be a random integer in $[2^p]$. We shall partition $[T]$ into a set of intervals as follows: the first interval I_1 contains the first q chunks; for each $o = 2, 3, \cdots$, I_o contains the next 2^p chunks, or the remaining chunks if there are less than 2^p of them. So p controls the length of intervals in $\mathcal{I}$ and q is a shifting parameter.

More formally, we let $L = 2^p u$. Let h be the largest integer such that $(h-2)L + qu < T$. Then $I_1 = [qu] \cap [T]$ and $I_o = \big((o-2)L + qu, (o-1)L + qu\big] \cap [T]$ for every $o = 2, 3, \cdots, h$. Let $\mathcal{I} = \{I_1, I_2, \cdots, I_h\}$ be the set of intervals that partition $[T]$. Notice that the intervals $I_2, I_3, \cdots, I_{h-1}$ have size L, while I_1 and I_h may have sizes smaller than L. See Figure 1 for the partitioning of $[T]$ into $\mathcal{I}$.

Definition 5.5. *For every $j \in J^*$, let v_j^B, v_j^C and v_j^D be the indices of the intervals in $\mathcal{I}$ that contain B_j, $\lceil C_j \rceil$ and $\lceil D_j \rceil$ respectively. (Notice that B_j is an integer, but C_j and D_j might be fractional.)*

We show some simple claims regarding v^B, v^C and v^D. Since $B_j \leq \lceil C_j \rceil \leq \lceil D_j \rceil$, we have

Claim 5.6. *$v_j^B \leq v_j^C \leq v_j^D$, for every $j \in J^*$.*

The following simple observation will be useful later.

Claim 5.7. *For two jobs $j, j' \in J^*$ such that $j \prec j'$, we have $v_j^B \leq v_{j'}^B$ and $v_j^C \leq v_{j'}^C$.*

Proof. The claim holds since $B_j \leq B_{j'}$ and $C_j \leq C_{j'}$ if $j \prec j'$. □

We now divide jobs into top, middle and bottom jobs, according to the number $v_j^D - v_j^B + 1$ of intervals in $\mathcal{I}$ that $[B_j, \lceil D_j \rceil]$ intersects for each j:

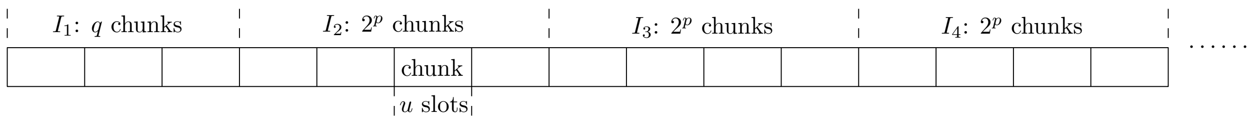

Figure 1: Partitioning of $[T]$ into intervals $\mathcal{I} = \{I_1, I_2, \cdots, I_h\}$.

DEFINITION 5.8. *We say $j \in J^*$ is a top job if $v_j^D - v_j^B + 1 \geq k^2 + 2k + 2$, a middle job if $v_j^D - v_j^B + 1 \in [2, k^2 + 2k + 1]$ and a bottom job if $v_j^D - v_j^B + 1 = 1$ (or equivalently $v_j^B = v_j^C = v_j^D$). Let $J^*_{\text{top}}, J^*_{\text{mid}}, J^*_{\text{bot}}$ denote the set of top, middle and bottom jobs respectively.*

The following claim and two lemmas are regarding the top jobs.

CLAIM 5.9. *For every $j \in J^*_{\text{top}}$, we have $D_j - B_j \geq k(k+2)L$.*

Proof. By the definition of top jobs, $[B_j, \lceil D_j \rceil]$ intersects at least $k^2 + 2k + 2$ intervals in $\mathcal{I}$. Since all intervals in $\{I_2, I_3, \cdots, I_{h-1}\}$ have length exactly L, we have $\lceil D_j \rceil - B_j + 1 \geq (k^2 + 2k)L + 2$, implying that $D_j - B_j \geq k(k+2)L$. □

We defined the flexible jobs in such a way that all the top jobs will be flexible, thus we have:

LEMMA 5.10. $\Delta(J^*_{\text{top}}) \leq \frac{u}{2^k}$.

Proof. Notice that for every job $j \in J^*_{\text{top}}$, we have $E_j - B_j \geq D_j - B_j \geq k(k+2)L = k(k+2)2^p u \geq k^2 u$, by Claim 5.9. Thus, all jobs in J^*_{top} are flexible jobs w.r.t x^*. Since (5.7) holds for x^*, we have $\Delta(J^*_{\text{top}}) \leq \frac{u}{2^k}$.

The following lemma lower bounds the number of intervals in $\mathcal{I}$ that $[B_j, \lceil C_j \rceil]$ intersects, for a top job j:

LEMMA 5.11. *For every $j \in J^*_{\text{top}}$, we have $v_j^C - v_j^B + 1 \geq k + 2$.*

Proof. For every $j \in J^*_{\text{top}}$, we have $C_j - B_j \geq (D_j - B_j)/k \geq (k+2)L$, by Claim 5.9. So, $[B_j, \lceil C_j \rceil]$ intersects at least $k+2$ intervals in $\mathcal{I}$. By the definition of v_j^B and v_j^C, we have that $v_j^C - v_j^B + 1 \geq k + 2$. □

One purpose of selecting the parameters p and q randomly as above is to guarantee that the probability that each job becomes a middle job is small. This is crucial to upper bound the expected completion times of jobs. Formally, we have

LEMMA 5.12. *For any job $j \in J^*$, we have $\Pr[j \in J^*_{\text{mid}}] \leq \frac{1}{km}$, where the probability is over the random choices of p and q.*

Proof. Let z be the number of chunks that $[B_j, \lceil D_j \rceil]$ intersect. Notice that all the intervals in $\mathcal{I}$ other than the first and the last one has exactly 2^p chunks. For $j \in J^*_{\text{mid}}$ to happen, one of the following two events must happen:

1. either $z/(k^2 + 2k + 2) \leq 2^p < z$,
2. or $2^p \geq z$ and $[B_j, \lceil D_j \rceil]$ intersects 2 intervals in $\mathcal{I}$.

Indeed, if both events do not happen, then either $z > 2^p(k^2 + 2k + 2)$, in which case j must be a top job, or $z \leq 2^p$ and $[B_j, \lceil D_j \rceil]$ intersects 1 interval in $\mathcal{I}$, in which case j is a bottom job.

The probability that the first event happens is at most $\frac{\lceil \log(k^2+2k+2) \rceil}{k_1} \leq \frac{2\lceil \log k \rceil + 1}{k_1}$ since there are at most $\lceil \log(k^2 + 2k + 2) \rceil$ different values of p satisfying the condition. Now consider the second event. For fixed p such that $2^p \geq z$, $[B_j, \lceil D_j \rceil]$ will intersect either 1 or 2 intervals in $\mathcal{I}$, and $j \in J^*_{\text{mid}}$ if it intersects 2. Notice that this happens with probability exactly $\frac{z-1}{2^p}$, over all the random choices of q. Considering all such values of p together, we have

$$\Pr[\text{the second event happens}] \leq \sum_{p \in [0, k_1): 2^p \geq z} \frac{1}{k_1} \cdot \frac{z-1}{2^p} < \frac{2}{k_1}.$$

Thus, the probability of $j \in J^*_{\text{mid}}$ is at most $\frac{2\lceil \log k \rceil + 3}{k_1} = \frac{2\lceil \log k \rceil + 3}{km(2\lceil \log k \rceil + 3)} = \frac{1}{km}$. □

To construct our schedule, we build a set $\mathcal{I}'$ of intervals $I'_1, I'_2, \cdots, I'_h$ of size L', where

$$L' = \left\lceil 2\left(1 + \frac{2}{k}\right) L \right\rceil + \left\lfloor \frac{u}{2^k} \right\rfloor \approx 2L. \tag{5.8}$$

More specifically, $I'_o = (oL', (o+1)L'] \cap \mathbb{Z}$ for every $o \in [h]$. With the intervals $\mathcal{I}'$ defined, we shall first assign $J^*_{\text{top}} \cup J^*_{\text{bot}}$ to them and we guarantee that jobs assigned to each I'_o can be scheduled inside I'_o. Then we shall insert J^*_{mid} to the schedule, which may require us to increase the sizes of intervals in $\mathcal{I}'$.

5.3 Assigning Bottom Jobs to Intervals

The assignment of bottom jobs to $\mathcal{I}'$ is straightforward: for each job $j \in J^*_{\text{bot}}$, we define $o_j = v_j^B = v_j^C = v_j^D$ and assign j to I'_{o_j}. Let $J^*_{\text{bot}}(o) = \{j \in J^*_{\text{bot}} : o_j = o\}$ be the set of bottom jobs assigned to I'_o, for every $o \in [h]$. Claim 5.13 and Lemma 5.14 will be used later to show that there is scheduling of $J^*_{\rm bot}(o)$ with small makespan.

CLAIM 5.13. *For every $o \in [h]$, we have $\Delta(J^*_{\rm bot}(o)) \leq L$.*

Proof. For every two jobs $j \prec j'$, we have $B_{j'} \geq B_j + 1$. Also all jobs $j \in J^*_{\rm bot}(o)$ has $B_j \in I_o$. Thus, we have $\Delta(J^*_{\rm bot}(o)) \leq |I_o| \leq L$. □

LEMMA 5.14. *For each $o \in [h]$, we have $|J^*_{\rm bot}(o)| \leq \frac{mL}{1-1/k}$.*

Proof. Notice that for every $j \in J^*_{\rm bot}(o)$, we have $B_j \in I_o$ and $\lceil D_j \rceil \in I_o$. Recall that $D_j = \min\{B_j + k(C_j - B_j), E_j\}$. If $D_j = E_j$, then $\sum_{t=B_j}^{D_j} x^*_{j,t} = 1$. Otherwise, $D_j = B_j + k(C_j - B_j)$. In this case $\sum_{t=B_j}^{\lfloor D_j \rfloor} x^*_{j,t} \geq 1 - 1/k$; otherwise we have $\sum_{t=\lfloor D_j \rfloor+1}^{E_j} x^*_{j,t} > 1/k$, implying $C_j > B_j + \frac{1}{k} \cdot (\lfloor D_j \rfloor + 1 - B_j) \geq B_j + \frac{1}{k}(D_j - B_j) = C_j$, a contradiction. In any case, at least $1 - 1/k$ fraction of the job j is scheduled in $[B_j, \lfloor D_j \rfloor] \cap \mathbb{Z} \subseteq I_o$ in x^*. Then, the size of $J^*_{\rm bot}(o)$ is at most $\frac{m|I_o|}{1-1/k} \leq \frac{mL}{1-1/k}$. □

5.4 Assigning Top Jobs to Intervals $\mathcal{I}'$

In this section, we assign top jobs to $\mathcal{I}'$, by defining an o_j for jobs $j \in J^*_{\rm top}$. This is done by solving a b-matching problem, as in the proof of the following lemma. Similar techniques have been used in [LR16] to remove the dependences between top and bottom jobs.

LEMMA 5.15. *We can find an integer vector $(o_j)_{j \in J^*_{\rm top}}$ such that*

- *for every $j \in J^*_{\rm top}$, we have $o_j \in [v^B_j + 1, v^C_j - 1]$, and*
- *for every $o \in [h]$, we have $\left|J^*_{\rm bot}(o) \cup \{j \in J^*_{\rm top} : o_j = o\}\right| \leq 2m\left(1 + \frac{2}{k}\right) L$.*

Notice that the first property requires o_j to be inside $[v^B_j + 1, v^C_j - 1]$, instead of $[v^B_j, v^C_j]$. This can guarantee that there are no precedence constraints between top jobs and bottom jobs assigned to I'_o. On the other hand, since $v^C_j - v^B_j$ is big for top jobs, we can guarantee this stronger condition, by losing a small factor on the ratio between L' and L.

Proof. [Proof of Lemma 5.15] We formulate the problem of finding the vector $(o_j)_{j \in J^*_{\rm top}}$ as a b-matching problem as follows. The left side of the bipartite graph is $J^*_{\rm top} \cup J^*_{\rm bot}$ and the right side is $[h]$. For every $j \in J^*_{\rm bot}$, there is an edge between j and o_j. For every $j \in J^*_{\rm top}$ and every $o \in [v^B_j + 1, v^C_j - 1]$, there is an edge between j and o. We need to find a matching where every $j \in J^*_{\rm top} \cup J^*_{\rm bot}$ is matched exactly once, and every $o \in [h]$ is matched at most $2m\left(1 + \frac{2}{k}\right) L$ times. By Hall's theorem, the problem is feasible if and only if for every $0 \leq b \leq e \leq h$, we have

$$\left|J^*_{[b,e]}\right| \leq 2m(e - b + 1)\left(1 + \frac{2}{k}\right) L,$$

where $J^*_{[b,e]} = J^*_{\rm top}[b,e] \cup J^*_{\rm bot}[b,e]$, $J^*_{\rm top}[b,e] = \left\{j \in J^*_{\rm top} : b \leq v^B_j + 1 < v^C_j - 1 \leq e\right\}$ and $J^*_{\rm bot}[b,e] = \bigcup_{o=b}^{e} J^*_{\rm bot}(o)$. That is $J^*_{[b,e]}$ is the set of jobs that must be matched to indices in $[b,e]$.

Notice that every job $j \in J^*_{\rm top}$ has $(v^C_j - 1) - (v^B_j + 1) - 1 \geq k$, by Lemma 5.11. Thus, if $e - b + 1 < k$, then $J^*_{\rm top}[b,e] = \emptyset$. The above statement holds since $|J^*_{\rm bot}(o)| \leq \frac{Lm}{1-1/k} \leq 2m\left(1 + \frac{2}{k}\right) L$ for every $o \in [h]$ by Lemma 5.14. So, we assume $e - b + 1 \geq k$. Let $b' = \max\{1, b-1\}$ and $e' = \min\{h, e+1\}$. For every $j \in J^*_{\rm top}[b,e]$, we have $[B_j, \lceil C_j \rceil] \subseteq I_{v^B_j} \cup I_{v^B_j+1} \cup I_{v^B_j+2} \cup \cdots \cup I_{v^C_j} \subseteq I_{b'} \cup I_{b'+1} \cup I_{b'+2} \cup \cdots \cup I_{e'}$. This also holds for every $j \in J^*_{\rm bot}[b,e]$.

Now we focus on the scheduling of jobs $J^*_{[b,e]}$ according to the solution x^*. Let s be the minimum time slot in $I_{b'}$ and t be the maximum time slot in $I_{e'}$. The C_j values of jobs in $J^*_{[b,e]}$ is at most t and all these jobs are scheduled at or after time slot s. Since there are only m machines, the total number of jobs in $J^*_{[b,e]}$ can only be at most $2m(t - s + 1) \leq 2m(e' - b' + 1)L$.

Since $e - b + 1 \geq k$ and $e' - b' + 1 \leq e - b + 1 + 2$, we have that $e' - b' + 1 \leq \left(1 + \frac{2}{k}\right)(e - b + 1)$. Thus, we have that $\left|J^*_{[b,e]}\right| \leq 2m\left(1 + \frac{2}{k}\right)(e - b + 1)L$. This finishes the proof of the lemma. □

We use the above lemma to find the vector $(o_j)_{j \in J^*_{\rm top}}$ satisfying the above properties. W.l.o.g, we can make sure that if $j \prec j'$ for two jobs $j, j' \in J^*_{\rm top}$, then $o_j \leq o_{j'}$. This can be assumed due to Claim 5.7: if $o_j > o_{j'}$, we can simply switch o_j and $o_{j'}$. Let $J^*_{\rm top}(o) = \left\{j \in J^*_{\rm top} : o_j = o\right\}$ for every $o \in [h]$. So the second statement of Lemma 5.15 is $|J^*_{\rm bot}(o) \cup J^*_{\rm top}(o)| \leq 2m\left(1 + \frac{2}{k}\right) L$.

5.5 Scheduling of Top and Bottom Jobs

With the assignment of jobs in $J^*_{\rm bot} \cup J^*_{\rm top}$ to $\mathcal{I}'$, we can now construct a scheduling $\mathcal{S}^1$ of these jobs, where jobs in $J^*_{\rm bot}(o) \cup J^*_{\rm top}(o)$ will be scheduled in I'_o. This will automatically satisfy the precedence constraints between jobs assigned to different intervals:

1. For two jobs $j, j' \in J^*_{\rm bot}$ with $j \prec j'$, we have $o_j = v^B_j \leq v^B_{j'} = o_{j'}$, by Claim 5.7.
2. For two jobs $j, j' \in J^*_{\rm top}$ with $j \prec j'$, we have $o_j \leq o_{j'}$ by the construction.

3. For every $o \in [h]$, there are no precedence constraints between $J^*_{\text{top}}(o)$ and $J^*_{\text{bot}}(o)$.

To see the third statement, focus on two jobs $j \in J^*_{\text{top}}$ and $j' \in J^*_{\text{bot}}$. If $j \prec j'$, then $o_j \le v^C_j - 1 \le v^C_{j'} - 1 < v^C_{j'} = o_{j'}$. If $j' \prec j$, then $o_j \ge v^B_j + 1 \ge v^B_{j'} + 1 > v^B_{j'} = o_{j'}$. Thus, it suffices for us to guarantee the precedence constraints between jobs in $J^*_{\text{top}}(o)$ and between jobs in $J^*_{\text{bot}}(o)$, for every $o \in [h]$.

LEMMA 5.16. *For every $o \in [h]$, we can efficiently find a valid schedule of $J^*_{\text{bot}}(o) \cup J^*_{\text{top}}(o)$ on m machines with makespan at most L'.*

Proof. We build the schedule in two steps. First, we construct any reasonable schedule $\widehat{\mathcal{S}}$ for $J^*_{\text{bot}}(o)$. Then, we insert $J^*_{\text{top}}(o)$ to $\widehat{\mathcal{S}}$, without changing the scheduling times of $J^*_{\text{bot}}(o)$, to form a reasonable schedule $\widehat{\mathcal{S}}'$ for $J^*_{\text{bot}}(o) \cup J^*_{\text{top}}(o)$. Notice that this is possible since there are no precedence constraints between $J^*_{\text{bot}}(o)$ and $J^*_{\text{top}}(o)$. Obviously, this can be done efficiently using the following process. We construct an order of jobs in $J^*_{\text{bot}} \cup J^*_{\text{top}}$ that respects the precedence constraints and puts J^*_{bot} before J^*_{top}. Starting from the empty schedule, for every job in $j \in J^*_{\text{bot}} \cup J^*_{\text{top}}$ according to the order, we update the schedule by inserting j to the earliest slot that maintains the validity of the schedule.

We first consider the reasonable schedule $\widehat{\mathcal{S}}$ for $J^*_{\text{bot}}(o)$. By Corollary 4.6, the makespan of $\widehat{\mathcal{S}}$ is at most $T_{J^*_{\text{bot}}(o)} = \left\lfloor \frac{|J^*_{\text{bot}}(o)|}{m} \right\rfloor + \Delta(J^*_{\text{bot}}(o))$. By Lemma 5.14 and Claim 5.13, this is at most $\left(1 + \frac{1}{1-1/k}\right) L \le 2\left(1+\frac{2}{k}\right) L \le L'$, as $k \ge 5$.

Now, focus on the reasonable schedule $\widehat{\mathcal{S}}'$ of $J^*_{\text{bot}}(o) \cup J^*_{\text{top}}(o)$, extended from $\mathcal{S}'$. We then apply Lemma 4.4 with $J' = J^*_{\text{bot}}(o) \cup J^*_{\text{top}}(o)$ and $J'' = J^*_{\text{top}}(o)$. Notice that there are no precedence constraints between $J'' = J^*_{\text{top}}(o)$ and $J' \setminus J'' = J^*_{\text{bot}}(o)$. Thus, we have

$$\max_{j \in J^*_{\text{top}}(o)} \widehat{\mathcal{S}}'_j \le \frac{|J^*_{\text{bot}}(o) \cup J^*_{\text{top}}(o)|}{m} + \Delta(J^*_{\text{top}}(o))$$
$$\le 2\left(1 + \frac{2}{k}\right) L + \left\lfloor \frac{u}{2^k} \right\rfloor = L',$$

by Lemma 5.15 and 5.10. Thus, overall, the makespan of $\widehat{\mathcal{S}}'$ is at most L'.

We apply the above lemma for every $o \in [h]$ to construct a valid schedule $\mathcal{S}^1$ for $J^*_{\text{bot}} \cup J^*_{\text{top}}$, where jobs in $J^*_{\text{bot}}(o) \cup J^*_{\text{top}}(o)$ are scheduled in I'_o, for every $o \in [h]$.

5.6 Inserting Middle Level Jobs In the final step of the algorithm, we insert jobs in J^*_{mid} to $\mathcal{S}^1$ to form the final schedule $\mathcal{S}^2$. We shall define $o_j \in [v^B_j, v^C_j]$ for every $j \in J^*_{\text{mid}}$. This is done via the following lemma.

LEMMA 5.17. *We can choose an index $o_j \in [v^B_j, v^C_j]$ for each $j \in J^*_{\text{mid}}$, such that for any two jobs $j, j' \in J^*$ with $j \prec j'$, we have $o_j \le o_{j'}$.*

Proof. This is implied by Claim 5.7, and that all jobs $j \in J^*_{\text{top}} \cup J^*_{\text{bot}}$ have $o_j \in [v^B_j, v^C_j]$. Formally, for every $j \in J^*_{\text{mid}}$, define

$$b_j = \max\left\{ v^B_j, \max_{j' \in J^*_{\text{top}} \cup J^*_{\text{bot}} : j' \prec j} o_{j'} \right\}$$
$$\text{and} \quad e_j = \min\left\{ v^C_j, \min_{j' \in J^*_{\text{top}} \cup J^*_{\text{bot}} : j \prec j'} o_{j'} \right\}.$$

Thus, we need $o_j \in [b_j, e_j]$ to guarantee that $o_j \in [v^B_j, v^C_j]$ for every $j \in J^*_{\text{mid}}$ and the precedence constraints between $J^*_{\text{top}} \cup J^*_{\text{bot}}$ and J^*_{mid}. Thus first we have to prove $b_j \le e_j$ for every $j \in J^*_{\text{mid}}$. We prove this by comparing each of the two terms in the "max" operator defining b_j and each of the two terms in the "min" operator defining e_j:

- $v^B_j \le v^C_j$.
- for every $j' \in J^*_{\text{top}} \cup J^*_{\text{bot}}$ such that $j \prec j'$, we have $v^B_j \le v^B_{j'} \le o_{j'}$.
- for every $j' \in J^*_{\text{top}} \cup J^*_{\text{bot}}$ such that $j' \prec j$, we have $o_{j'} \le v^C_{j'} \le v^C_j$.
- for every $j', j'' \in J^*_{\text{top}} \cup J^*_{\text{bot}}$ such that $j' \prec j \prec j''$, we have $o_{j'} \le o_{j''}$.

Thus, it suffices to find a vector $(o_j)_{j \in J^*_{\text{mid}}}$ such that $o_j \in [b_j, e_j]$ for every $j \in J^*_{\text{mid}}$, and for any two jobs $j, j' \in J^*_{\text{mid}}$ such that $j \prec j'$, we have $o_j \le o_{j'}$. Indeed, we can simply define $o_j = b_j$ for every $j \in J^*_{\text{mid}}$: for every $j, j' \in J^*_{\text{mid}}$ with $j \prec j'$, $b_j \le b_{j'}$ is implied by the definitions of b_j and $b_{j'}$, and the facts that $v^B_j \le v^B_{j'}$ and $\prec$ is transitive.

With o_j defined, we can then insert each job $j \in J^*_{\text{mid}}$ to I'_{o_j}. Notice that this will automatically guarantee the precedence constraints between jobs scheduled in different intervals in $\mathcal{I}'$. However, there might be no way to guarantee all the precedence constraints between a $j \in J^*_{\text{mid}}$ and the jobs that have already been scheduled in I'_{o_j} (including the jobs in $J^*_{\text{top}}(o_j) \cup J^*_{\text{bot}}(o_j)$ and all the middle jobs that have been inserted to the interval). To address this issue, we can extend the size of I'_{o_j} by 1 (and shift the intervals after I'_{o_j} accordingly).

5.7 Analyzing the Expected Completion Times We now analyze the expected completion time of each job $j \in J^*$. Let us fix the choices of p and q; thus the partition of J^* into $J^*_{\text{top}}, J^*_{\text{mid}}, J^*_{\text{bot}}$ is fixed. For every job $j \in J^*$, the completion time of j is at most the total size of intervals in $\left\{I'_1, I'_2, \cdots, I'_{o_j}\right\}$, after the insertion of J^*_{mid}, which is at most

$$o_j L' + \left|\{j' \in J^*_{\text{mid}} : o_{j'} \le o_j\}\right|,$$

where $o_j L'$ is the total length of intervals $\left\{I'_1, I'_2, \cdots, I'_{o_j}\right\}$ before the insertion of J^*_{mid}, and $|\{j' \in J^*_{\text{mid}} : o_{j'} \le o_j\}|$ is an upper bound on the increase of the quantity due to insertion of J^*_{mid}.

We focus on the expected value of the second term, over all random choices of p and q. The idea is to show that if some $j' \in J^*_{\text{mid}}$ has $o_{j'} \le o_j$, then $C_{j'}$ is not too big compared to C_j. This uses the fact that a middle job j' has a small $C_{j'} - B_{j'}$. Formally, we have

$$\begin{aligned} C_{j'} &\le B_{j'} + D_{j'} - B_{j'} \le B_{j'} + (k^2+2k+2)L \\ &\le (o_{j'}+1)L + (k^2+2k+2)L \\ &\le (o_{j'} + k^2+2k+3)L \\ &\le (o_j + k^2+2k+3)L \le C_j + (k^2+2k+4)L. \end{aligned}$$

The second inequality holds since $[B_{j'}, \lceil D_{j'} \rceil]$ intersects at most k^2+2k+1 intervals in $\mathcal{I}$. The third inequality is due to $o_{j'} \ge v^B_{j'} = \left\lceil \frac{B_{j'} - qu}{L} \right\rceil \ge \frac{B_{j'}}{L} - 1$, and the last inequality is due to $o_j \le v^C_j = \left\lceil \frac{\lceil C_j \rceil - qu}{L} \right\rceil \le \frac{C_j}{L} + 1$.

For a fixed job $j \in J^*$, the number of jobs j' with $C_{j'} \le C_j + (k^2+2k+4)L$ is at most $2m\left(C_j + (k^2+2k+4)L\right)$. For each such job j', we have $\Pr[j' \in J^*_{\text{mid}}] \le \frac{1}{km}$, by Lemma 5.12. Thus, the expected value of $|\{j' \in J^*_{\text{mid}} : o_{j'} \le o_j\}|$ is at most $\frac{1}{km} \times 2m\left(C_j + (k^2+2k+4)L\right) \le \frac{2C_j}{k} + 4kL$ since $k \ge 5$.

Thus, overall, the expected completion time of any job $j \in J^*$ is at most

$$\begin{aligned} & o_j L' + \frac{2C_j}{k} + 4kL \\ &\le \left(\frac{C_j}{L} + 1\right) \times \left(\left\lceil 2\left(1 + \frac{2}{k}\right)L\right\rceil + \left\lfloor \frac{u}{2^k} \right\rfloor\right) \\ &\qquad + \frac{2C_j}{k} + 4kL \\ &\le \left(\frac{C_j}{L} + 1\right) \times \left(2\left(1 + \frac{2}{k}\right)L + \frac{u}{2^k} + 1\right) \\ &\qquad + \frac{2C_j}{k} + 4kL \\ &\le 2\left(1 + \frac{2}{k}\right)C_j + \frac{C_j}{2^{p+k}} + \frac{C_j}{L} + O(1)\cdot L \\ &\qquad + \frac{2C_j}{k} + 4kL \\ &\le \left(2 + \frac{O(1)}{k}\right)C_j + O(1)\cdot kL. \end{aligned}$$

In the last inequality, we used the assumption that $L \ge u \ge k$, which implies $\frac{C_j}{L} \le \frac{C_j}{k}$.

By the definition of L and u, we have $kL = k2^q u \le k2^{k_1} u \le \frac{T}{2^k}$. Thus, the expected weighted completion time of our schedule is at most $\left(2 + \frac{O(1)}{k}\right)\sum_{j \in J^*} w_j C_j + O(1) \cdot \frac{T}{2^k} \cdot \sum_{j \in J^*} w_j \le \left(2 + \frac{O(1)}{k}\right)\Phi + \frac{O(1)}{2^k} \cdot T \cdot w(J^*)$. We can derandomize the algorithm by enumerating all possible choices of p and q. This finishes the proof of Theorem 5.1.

6 Obtaining Multiplicative Approximation via Decomposition

In this section, we show how to remove the additive term in Theorem 5.1, by decomposing the input instance J into multiple instances J^*. This will finish the proof of Theorem 1.1. The technical lemma we are going to prove is the following:

Lemma 6.1. *Let $k \ge 5$, $\Phi \ge 0$ and $x^* \in \mathcal{P}_J(\Phi)$ be given. Then, in running time $n^{2^{O(mk \log k)}}$, we can either find a subset $J' \subseteq J$ such that any valid schedule $\mathcal{S}$ of J' has weighted completion time strictly larger than $\sum_{j \in J', t \in [T_J]} x^*{}_{j,t} t$, or output a schedule of J with weighted completion time at most $\left(2 + \frac{O(1)}{k}\right)\Phi$.*

We first show how Lemma 6.1 implies Theorem 1.1. By binary search, we assume we are given an upper bound Φ on the weighted completion time of the optimum schedule for J, and our goal is to output a schedule with weighted completion time at most $(2+\epsilon)\Phi$. Let $k = \Theta(1/\epsilon)$ to be large enough.

We run the ellipsoid method using the following separation oracle. Given a vector $x^* \in [0,1]^{J \times [T_J]}$, the separation oracle first checks if $x^* \in \mathcal{P}_J(\Phi)$; if not, it returns a constraint defining $\mathcal{P}_J(\Phi)$ that is violated by x^* and we proceed to the next iteration of the ellipsoid method. Otherwise, the separation oracle runs the algorithm stated in Lemma 6.1. If the outcome of the algorithm is a subset $J' \subseteq J$ such that any schedule $\mathcal{S}$ for J' has weighted completion time strictly larger than $\sum_{j \in J', t \in [T_J]} x^*{}_{j,t} t$, then the separation oracle returns the following valid constraint that is violated by x^*:

$$\sum_{j \in J'} w_j \sum_{t \in [T_J]} x_{j,t} t \ge \left\lfloor \sum_{j \in J'} w_j \sum_{t \in [T_J]} x^*_{j,t} t \right\rfloor + 1.$$

We used the assumption that all weights are integers and thus the weighted completion time of any integral schedule is always an integer. In this case, we also proceed to the next iteration of the ellipsoid method. If the outcome of the algorithm is a schedule of J with weighted completion time at most $\left(2 + \frac{O(1)}{k}\right)\Phi$, then we terminate the ellipsoid method by returning the this schedule. If $k = \Theta(1/\epsilon)$ is large enough, then the weighted completion time of the output schedule is at most $(2+\epsilon)\Phi$. Since there is a schedule of J with weighted completion time at most Φ, the ellipsoid method will terminate in polynomial number of iterations. The running time of the algorithm is the number of iterations times the running time for the algorithm in Lemma 6.1, which is

$$n^{2^{O(mk\log k)}} = n^{2^{O((m/\epsilon)\log(1/\epsilon))}}.$$

Proof. [Proof of Lemma 6.1] In this proof, we shall use T for T_J. For any $j \in J$, let $C_j = \sum_{t\in[T]} t x^*_{j,t}$ be the fractional completion time of j according to x^*. For any integer $p \in [k]$, and integer $\ell \in \mathbb{Z}_{\geq 0}$, we define

$$J^p_\ell = \left\{ j \in J : \log C_j \in \left(\ell k - p, \ell k + k - 1 - p\right]\right\},$$
$$J'^p_\ell = \left\{ j \in J : \log C_j \in \left(\ell k + k - 1 - p, (\ell+1)k - p\right]\right\}.$$

Notice that for any $p \in [k]$, the sets $\{J^p_\ell\}_\ell \cup \{J'^p_\ell\}_\ell$ form a partition of J. (See Figure 2.) For two different sets in the partition, precedence constraints can only go from jobs in the left-side set to jobs in the right-side set, due to Claim 4.7. We define $J^p_{[\ell]} = \bigcup_{\ell'\leq\ell} J^p_{\ell'}$ and $J'^p_{[\ell]} = \bigcup_{\ell'\leq\ell} J'^p_{\ell'}$ for every $\ell \in \mathbb{Z}_{\geq 0}$.

For every $p \in [k]$ and integer $\ell \in \mathbb{Z}_{\geq 0}$, we let $\Phi^p_\ell = \sum_{j\in J^p_\ell} w_j C_j$ be the total weighted completion time of jobs in J^p_ℓ, according to x^*. Let $r = 2^{O(mk\log k)}$ be the parameter satisfying the statement of Theorem 5.1. For every $p \in [k]$ and $\ell \in \mathbb{Z}_{\geq 0}$, we check if $\mathrm{SA}(\mathcal{P}_{J^p_\ell}(\Phi^p_\ell), r)$ is empty or not. If it is empty for some p and ℓ, then we can return the set $J' = J^p_\ell$ and finish the proof of the lemma. Thus, we assume that for every $p \in [k]$ and $\ell \in \mathbb{Z}_{\geq 0}$, we have $\mathrm{SA}(\mathcal{P}_{J^p_\ell}(\Phi^p_\ell), r) \neq \emptyset$.

Now we randomly choose some integer $p \in [k]$; from now on, this p is fixed. For every $\ell \in \mathbb{Z}_{\geq 0}$, we apply Theorem 5.1 to with $J^* = J^p_\ell$ and $\Phi = \Phi^p_\ell$ to obtain a schedule $\mathcal{S}^p_\ell$ of J^p_ℓ with weighted completion time at most $\left(2 + \frac{O(1)}{k}\right)\Phi^p_\ell + \frac{O(1)}{2^k} \cdot T_{J^p_\ell} \cdot w(J^p_\ell)$.

With the schedules $\{\mathcal{S}^p_\ell\}_{\ell\in\mathbb{Z}_{\geq 0}}$ defined, we construct the final schedule $\mathcal{S}$ for J greedily. Start from $\mathcal{S}$ being the empty schedule. For every $\ell = 0, 1, 2, \cdots$, we apply the following procedure. First we append the schedule $\mathcal{S}^p_\ell$ to the end of $\mathcal{S}$. Then we decrease the scheduling times of some jobs in $\mathcal{S}$ to make it reasonable. We construct an *arbitrary* reasonable schedule $\mathcal{S}'^p_\ell$ of J'^p_ℓ and append $\mathcal{S}'^p_\ell$ to the end of $\mathcal{S}$. Again we decrease the scheduling times of some jobs in $\mathcal{S}$ until it becomes reasonable. This finishes the description of the construction of our final $\mathcal{S}$. Notice that the procedure is valid since the order $(J^p_0, J'^p_0, J^p_1, J'^p_1, \cdots)$ agrees with the precedence constraints.

We then analyze the total weighted completion time of all jobs in the final schedule $\mathcal{S}$. We first analyze the contribution of jobs in J^p_ℓ, for some $\ell \in \mathbb{Z}_{\geq 0}$. The contribution is at most

$$\left(2 + \frac{O(1)}{k}\right)\Phi^p_\ell + \frac{O(1)}{2^k} \cdot T_{J^p_\ell} \cdot w(J^p_\ell)$$
$$+ T_{J^p_{[\ell-1]} \cup J'^p_{[\ell-1]}} \cdot w(J^p_\ell).$$

The first two terms come from the weighted completion time of schedule $\mathcal{S}^p_\ell$. Since we appended $\mathcal{S}^p_\ell$ to a reasonable schedule of $J^p_{[\ell-1]} \cup J'^p_{[\ell-1]}$, the completion time of each job $j \in J^p_\ell$ will be increased by at most $T_{J^p_{[\ell-1]} \cup J'^p_{[\ell-1]}}$, which leads to the third term.

By Lemma 4.8, we have $T_{J^p_\ell} \leq 3 \cdot 2^{\ell k + k - 1 - p}$ and $T_{J^p_{[\ell-1]} \cup J'^p_{[\ell-1]}} \leq 3 \cdot 2^{\ell k - p}$. Thus, the weighted completion time of J^p_ℓ in $\mathcal{S}$ is at most

$$\left(2 + \frac{O(1)}{k}\right)\Phi^p_\ell + O(1) \cdot 2^{\ell k - p} \cdot w(J^p_\ell).$$

Now we consider the contribution of J'^p_ℓ, for some $\ell \in \mathbb{Z}_{\geq 0}$. The completion time of any job $j \in J'^p_\ell$ in $\mathcal{S}$ is at most $T_{J^p_{[\ell]} \cup J'^p_{[\ell]}}$, which by Lemma 4.8, is at most $3 \cdot 2^{(\ell+1)k - p}$. So, the contribution of J'^p_ℓ is at most

$$O(1) \cdot w(J'^p_\ell) \cdot 2^{(\ell+1)k-p}.$$

Thus, overall, the total weighted completion time of jobs in J is at most

$$\sum_\ell \left(\left(2 + \frac{O(1)}{k}\right)\Phi^p_\ell + O(1)\cdot 2^{\ell k - p} \cdot w(J^p_\ell)\right)$$
$$+ \sum_\ell O(1) \cdot w(J'^p_\ell) \cdot 2^{(\ell+1)k-p}$$
$$\leq \left(2 + \frac{O(1)}{k}\right)\Phi + O(1)\sum_\ell 2^{\ell k - p} \cdot w(J^p_\ell)$$
$$+ O(1)\sum_\ell 2^{(\ell+1)k-p} \cdot w(J'^p_\ell)$$
$$=: \left(2 + \frac{O(1)}{k}\right)\Phi + Q_1 + Q_2.$$

We now consider the expected contribution of each job j to the terms Q_1, over all random choices of p. Let $\mathcal{E}_j$ denote the event that $j \in \bigcup_\ell J^p_\ell$. Notice that $\Pr[\mathcal{E}_j] = \frac{k-1}{k}$. When $\mathcal{E}_j$ happens, let ℓ be the index such that $j \in J^p_\ell$. Then, it is easy to see that under the condition that $\mathcal{E}_j$ happens, $\lceil \log C_j \rceil - (\ell k - p)$ is uniformly distributed in $[k-1]$. Thus, the expected

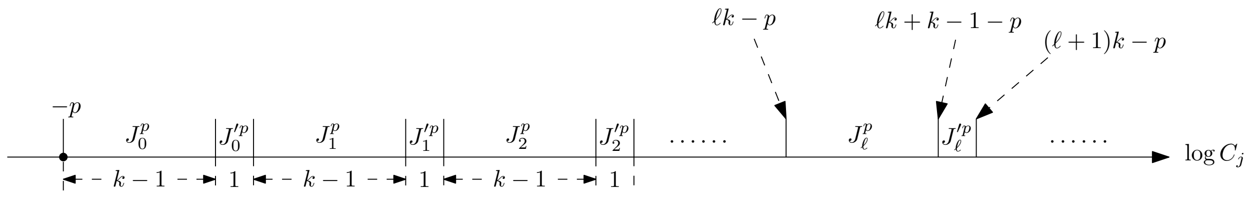

Figure 2: Partitioning of J into $\{J_\ell^p\}_\ell \cup \{J_\ell'^p\}_\ell$ according to $\log C_j$.

contribution of j to Q_1 is at most

$$O(1) \cdot w_j \cdot \mathbb{E}[2^{\ell k - p}] \le O(1) \cdot w_j \cdot \frac{1}{k} \sum_{k'=1}^{k-1} 2^{\lceil \log C_j \rceil - k'}$$

$$\le O(1) \cdot w_j \cdot \frac{1}{k} \sum_{k'=1}^{k-1} \frac{2C_j}{2^{k'}} = \frac{O(1)}{k} w_j C_j.$$

Then we consider the expected contribution of j to Q_2. Notice that j contributes to Q_2 only when $\mathcal{E}_j$ does not happen. Under this condition, its contribution to Q_2 is at most $O(1) \cdot w_j \cdot 2C_j = O(1) \cdot w_j C_j$. Thus the expected contribution of j to Q_2 is at most $\frac{O(1)}{k} w_j C_j$.

Overall, the expected value of $Q_1 + Q_2$ is at most $\frac{O(1)}{k} \sum_{j \in J} w_j C_j$, over all random choices of p. This implies that the expected weighted completion time of the solution output by our algorithm is at most $\left(2 + \frac{O(1)}{k}\right) \sum_{j \in J} w_j C_j = \left(2 + \frac{O(1)}{k}\right) \Phi$. We can derandomize our algorithm by enumerating all possible choices of p. This finishes the proof of Lemma 6.1. □

References

[1] Hyung-Chan An, Mohit Singh, and Ola Svensson. LP-based algorithms for capacitated facility location. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2014*, 2014.

[2] Nikhil Bansal and Subhash Khot. Optimal long code test with one free bit. In *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '09, pages 453–462. IEEE Computer Society, 2009.

[3] Nikhil Bansal, Aravind Srinivasan, and Ola Svensson. Lift-and-round to improve weighted completion time on unrelated machines. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 156–167, 2016.

[4] Soumen Chakrabarti, Cynthia A. Phillips, Andreas S. Schulz, David B. Shmoys, Cliff Stein, and Joel Wein. *Improved scheduling algorithms for minsum criteria*, pages 646–657. Springer Berlin Heidelberg, 1996.

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

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

[7] Shashwat Garg. Quasi-ptas for scheduling with precedences using LP hierarchies. *Accepted to ICALP*, 2018.

[8] R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM JOURNAL ON APPLIED MATHEMATICS*, 17(2):416–429, 1969.

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

[10] Leslie A. Hall, Andreas S. Schulz, David B. Shmoys, and Joel Wein. Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Math. Oper. Res.*, 22(3):513–544, August 1997.

[11] Han Hoogeveen, Petra Schuurman, and Gerhard J. Woeginger. Non-approximability results for scheduling problems with minsum criteria. In *Proceedings of the 6th International IPCO Conference on Integer Programming and Combinatorial Optimization*, pages 353–366. Springer-Verlag, 1998.

[12] Jean B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.

[13] M. Laurent. A comparison of the sherali-adams, lovsz-schrijver and lasserre relaxations for 0-1 programming. *Mathematics of Operations Research*, 28:470–496, 2001.

[14] J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of scheduling under precedence constraints. *Oper. Res.*, 26(1):22–35, February 1978.

[15] Elaine Levey and Thomas Rothvoss. A (1+epsilon)-approximation for makespan scheduling with precedence constraints using LP hierarchies. In *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pages 168–177. ACM, 2016.

[16] Shi Li. On uniform capacitated k-median beyond the natural LP relaxation. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015).*

[17] Shi Li. Approximating capacitated k-median with $(1 + \epsilon)k$ open facilities. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016)*, pages 786–796, 2016.

[18] Shi Li. Scheduling to minimize total weighted completion time via time-indexed linear programming relaxations. In *Proceedings of the 2017 IEEE 58rd Annual Symposium on Foundations of Computer Science*, FOCS '17, 2017.

[19] L. Lovasz and A. Schrijver. Cones of matrices and set-functions and 01 optimization. *SIAM Journal on Optimization*, 1(2):166–190, 1991.

[20] Alix Munier, Maurice Queyranne, and Andreas S. Schulz. *Approximation Bounds for a General Class of Precedence Constrained Parallel Machine Scheduling Problems*, pages 367–382. Springer Berlin Heidelberg, 1998.

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

[22] Thomas Rothvoss. The lasserre hierarchy in approximation algorithms, 2013.

[23] Petra Schuurman and Gerhard J. Woeginger. Polynomial time approximation algorithms for machine scheduling: Ten open problems, 1999.

[24] Hanif Sherali and Warren P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. 3:411–430, 05 1990.

[25] Martin Skutella. A 2.542-approximation for precedence constrained single machine scheduling with release dates and total weighted completion time objective. *Operations Research Letters*, 44(5):676 – 679, 2016.

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