Scheduling with Communication Delays via LP Hierarchies and Clustering II: Weighted Completion Times on Related Machines

Sami Davies* Janardhan Kulkarni† Thomas Rothvoss*
Jakub Tarnawski† Yihao Zhang*

Wednesday 15th July, 2020

Abstract

We consider the problem of scheduling jobs with precedence constraints on related machines to minimize the weighted sum of completion times, in the presence of *communication delays*. In this setting, denoted by $\mathsf{Q} \mid \mathsf{prec}, c \mid \sum w_j C_j$, if two dependent jobs are scheduled on different machines, then at least c units of communication delay time must pass between their executions. Our main result is an $O(\log^4 n)$-approximation algorithm for the problem. As a corollary of our result, we also obtain an $O(\log^3 n)$-approximation algorithm for the problem of minimizing makespan $\mathsf{Q} \mid \mathsf{prec}, c \mid C_{\max}$, which improves upon the $O(\log^5 n / \log\log n)$-approximation algorithm due to a recent work of Maiti et al. [MRS+20].

1 Introduction

We consider the problem of scheduling jobs with precedence and communication delay constraints on *related machines.* This classic model was first introduced by Rayward-Smith [RS87] and Papadimitriou and Yannakakis [PY90]. In this problem we are given a set J of n jobs, where each job j has a processing length $p_j \in \mathbb{Z}_+$ and a weight $w_j \in \mathbb{Z}_+$. The jobs need to be scheduled on m related machines, where machine $i \in [m]$ has speed $s_i \in \mathbb{Z}_+$. If a job j with processing length p_j is scheduled on the machine i, then it requires p_j/s_i time units to complete. In addition, we are given a *communication delay parameter* $c \in \mathbb{Z}_{\geq 0}$. The jobs have precedence and communication delay constraints, which are given by a partial order $\prec$. A constraint $j \prec j'$ encodes that job j' can only start after job j is completed. Moreover, if $j \prec j'$ and j, j' are scheduled on different machines, then j' can only start executing at least c time units after j had finished. On the other hand, if j and j' are scheduled on the same machine, then j' can start executing immediately after j finishes. The goal is to schedule jobs *non-preemptively* so as to minimize a certain objective function. In a non-preemptive schedule, each job j needs to be assigned to a single machine i and executed during a contiguous time interval of length p_j/s_i.

We focus on two widely studied objective functions: (1) minimizing the weighted sum of completion times of jobs, and (2) minimizing the makespan. In the standard 3-field notation[1], these

*University of Washington, Seattle. Email: {daviess,rothvoss,yihaoz93}@uw.edu. Thomas Rothvoss is supported by NSF CAREER grant 1651861 and a David & Lucile Packard Foundation Fellowship.

†Microsoft Research, Redmond. Email: {jakul,jatarnaw}@microsoft.com.

[1]We adopt the convention of [GLLK79, VLL90], where the respective fields denote: **(1) machine environment:**

problems are denoted by $Q \mid \mathsf{prec}, c \mid \sum w_j C_j$ and $Q \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$, respectively. In the presence of precedence and communication delay constraints, the weighted completion time objective is more general than the makespan, in the sense that one can use an approximation algorithm for the completion time objective to obtain a comparable result for the makespan. Our main motivation to study these problems is twofold:

- The problems of scheduling jobs with communication delays are some of the most notorious open questions in approximation algorithms and scheduling theory, which have resisted progress for a long time. For this reason, the well-known survey by Schuurman and Woeginger [SW99] and its recent update by Bansal [Ban17] list understanding the approximability of the problems in this model as one of the top-10 open questions in scheduling theory.

- These problems arise in many real-world applications, especially in the context of scheduling in data centers. A precedence constraint $j \prec j'$ typically implies that the input to j' depends on the output of j. Hence, if j and j' are scheduled on different machines, then the *communication delay* due to transferring this output to the other machine often becomes the bottleneck. The problem has received significant attention in applied data center scheduling literature; see [CZM+11, GFC+12, HCG12, SZA+18, ZZC+12, ZCB+15, LYZ+16] for more details. Another timely example is in the parallelization of Deep Neural Network training. When DNNs are trained on multiple clusters, the communication costs incurred for synchronizing the weight updates in fact dominate the overall execution time [NHP+19]. The resulting *device placement* problem [MPL+17, GCL18, JZA19, TPD+20] is indeed a variant of scheduling with communication delays.

Scheduling jobs with precedence and communication delays has been studied extensively over many years [RS87, PY90, MK97, HM01, TY92, HLV94, PY90, GKMP08]. Yet, very little was known in terms of the approximation algorithms for problem until the recent work by Maiti et al. [MRS+20] and us [DKR+20]. Previously, even for the *identical machines* case, where all machines have the same speed, the best algorithm for general c achieved an approximation factor of $2/3 \cdot (c+1)$ [GKMP08], which only marginally improves on Graham's list scheduling algorithm that obtains a $(c+1)$-approximation, while requiring the assumption that $p_j = 1$. Moreover, no results were known for the related machines case.

In a very recent work, we [DKR+20] designed an $O(\log m \log c)$-approximation algorithm for minimizing makespan in the *identical machines case.* We also showed an $O(\log n \log c)$-approximation algorithm for the weighted sum of completion times objective, in a special case where $p_j = 1$ and we have an *unlimited* number of machines. In a parallel and independent work, Maiti et al. [MRS+20] developed an $O(\log^2 n \log^2 m \log c / \log\log n)$-approximation algorithm for the makespan objective function on *related machines.* Interestingly, these two results were obtained using rather different techniques. While our results were based on LP hierarchies and clustering, Maiti et al. [MRS+20] developed a novel framework based on *job duplication.* In their framework, they first construct a schedule where a single job can be scheduled on *multiple machines*, which is known to effectively "hide" the communication delay constraints [PY90]. Quite surprisingly, Maiti et al. [MRS+20] showed that one can convert a schedule with duplication to a feasible schedule without duplication, where every job is processed on a single machine, while increasing the makespan by at most an $O(\log^2 n \log m)$ factor. Unfortunately, their framework does not seem to extend to the weighted

Q for related machines, P for identical machines, **(2) job properties:** prec for precedence constraints; c for communication delays of length c; $p_j = 1$ for unit length case, **(3) objective:** C_{max} for minimizing makespan, $\sum w_j C_j$ for minimizing weighted sum of completion times.

sum of completion times objective; we discuss this further in Section 1.2. Moreover, our previous results [DKR+20] also do not imply any approximation guarantees for the related machines case. Specifically, in the presence of communication delay constraints there are no known reductions between the sum of weighted completion time and makespan objective functions in a spirit similar to [CS99].

Finally, in another recent work, Su et al. [SRVW20] studied the objective of minimizing makespan and sum of weighted completion times on related machines. They showed that a Generalized Earliest Time First (GETF) algorithm achieves a makespan of $O(\log m/\log\log m)\cdot OPT+C$, where C is the total communication delay in the longest chain in the input graph. They show a similar bound on the schedule produced by GETF for the weighted sum of completion times objective. However, both of these bounds do not give any multiplicative approximation guarantees, as the additive terms can be substantially higher than the optimal solution. The additive terms in above bounds are precisely the reason why the scheduling with communication delays problem is significantly more challenging than the case when $c = 0$.

1.1 Our Contributions

Let M denote the number of machines types or equivalently speed classes. By standard arguments we can assume that $M = \log(s_{\max}/s_{\min}) \leq O(\log m)$ while only losing a constant factor in the approximation guarantee, where $s_{\max}$ and $s_{\min}$ denote the fastest and slowest speeds of machines in the input instance.

The main result of this paper is the following:

Theorem 1. *There is a randomized $O(M^2 \cdot \log^2 n)$-approximation algorithm for $Q \mid \mathsf{prec}, c \mid \sum_j w_j C_j$ with expected polynomial running time. When jobs have unit processing lengths, $Q \mid \mathsf{prec}, c, p_j = 1 \mid \sum_j w_j C_j$, the approximation factor of the algorithm improves to $O(M \cdot \log^2 n)$.*

As $M \leq O(\log n)$, our result gives an $O(\log^4 n)$-approximation algorithm to the general problem and an $O(\log^3 n)$-approximation algorithm for the case with unit processing lengths. As a byproduct of our result, we also obtain an improved approximation for the makespan objective function.

Theorem 2. *There is a randomized $O(M \cdot \log m \cdot \log n)$-approximation algorithm for $Q \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$ with expected polynomial running time. When jobs have unit processing lengths, $Q \mid \mathsf{prec}, c, p_j = 1 \mid C_{\mathsf{max}}$, the approximation factor of the algorithm improves to $O(\log m \cdot \log n)$.*

So our result gives an $O(\log^3 n)$-approximation algorithm for the problem of minimizing makespan (and an $O(\log^2 n)$-approximation algorithm for the case with unit processing lengths). This improves the $O(\log^5 n/\log\log n)$-approximation algorithm due to Maiti et al. [MRS+20].

1.2 Technical Challenges

Absent the communication delay constraints, one can use an approximation for the makespan objective to obtain an approximation algorithm for the weighted sum of completion times objective, with some (negligible) loss in the approximation quality [HSSW97, QS02, Li17]. Namely, a standard approach is to first solve an LP for the weighted sum of completion times problem, and then geometrically partition jobs according to completion times in the LP solution into buckets of the form $[2^t, 2^{t+1}]$. Next, use an α-approximation algorithm for makespan to schedule these jobs in an interval of length $O(\alpha \cdot 2^t)$. Finally, concatenate the schedules of jobs belonging to different partitions. This approach gives an $O(\alpha)$-approximation for identical machines, and an extension of this idea to related machines is given by [CS99].

However, this approach fails in the presence of communication delay constraints, even for the identical machines case. In particular, the main technical difficulty arises when in the LP solution a large number of jobs have very small completion times, say $O(c/\log^2 n)$. We need to schedule these jobs so that no precedence and communication delay constraints are violated, while at the same time achieving completion times comparable to the LP. This requires very precise control over job completion times, which cannot be achieved by the existing algorithms for the makespan. The problem becomes further more complex if machines have different speeds and jobs have different weights. This is also the main technical hurdle if one tries to adapt the framework of Maiti et al. [MRS$^+$20]. In fact, [MRS$^+$20] gives a schedule whose cost can be as large as $O\left((\log^5 n/\log\log n)\cdot OPT + c\cdot\sum_j w_j\right)$, which only implies an approximation factor of $\max\{c, \log^5 n/\log\log n\}$.

1.3 Our Techniques and Algorithm Overview

We obtain our results by generalizing the LP hierarchy framework introduced by [DKR$^+$20] to handle processors of varying speed and the more general objective of minimizing the weighted sum of completion times. In this section we outline our main techniques and explain our innovations compared to [DKR$^+$20]. In particular, as alluded above, the weighted sum of completion times objective requires a finer control over the *time slots where jobs are scheduled* compared to minimizing the makespan. Moreover, allowing different machine types introduces an *assignment* aspect to the problem, namely assigning jobs to a machine type, which is not present in the identical machines case. Tackling these two challenges simultaneously requires an extended LP as well as more structural insights about the solution produced by the Sherali-Adams hierarchy compared to [DKR$^+$20].

To keep the notation simple we will use $\log n$ instead of the potentially smaller quantities $\log m$ and M. We begin our discussion with the problem $\mathsf{Q} \mid \mathsf{prec}, p_j = 1, c \mid \sum w_j C_j$, where jobs have unit lengths. We will see later that we can reduce the version with general processing times to the case $p_j = 1$ while losing a factor of $O(\log n)$. A helpful view that already emerged in preceeding work is to partitition the time horizon into intervals of length c or larger. Then if one can assign each job j_1 so that any predecessor j_2 is either executed on the same machine or in a later interval, then inserting a waiting time of c after each interval will result in a valid schedule while completion times increase by at most a factor of 2.

We start by designing an LP with *time-indexed variables* $x_{j,i,t}$ that specify whether job j is to be scheduled in time slot t on machine i. To accommodate the different speeds, a machine in class k has s_k slots available per unit time interval. Similarly to [DKR$^+$20], we take the *Sherali-Adams lift* with a constant number of rounds and use it to extract variables y_{j_1,j_2}, which tell us whether the jobs are scheduled on the same machine within an interval of length c, as well as variables C_j, which denote the *LP completion times*. Our main LP rounding result is that we can generate a schedule at random so that the completion time of every job is at most $O(\log^3 n)\cdot C_j$ in expectation.

Using the Sherali-Adams properties one can prove that the function $d : J \times J \to \mathbb{R}_{\geq 0}$ with $d(j_1, j_2) := 1 - y_{j_1,j_2}$ is a *semimetric*. One important ingredient of our algorithm is a clustering procedure due to Calinescu, Karloff and Rabani [CKR04] which, for such a semimetric space (J, d) and a parameter $\Delta > 0$, finds a random partition $J = V_1 \dot\cup \ldots \dot\cup V_q$ into blocks of diameter at most Δ such that any set $U \subseteq J$ has a probability of at most $O(\log(n)\cdot\frac{\mathrm{diam}(U)}{\Delta})$ of being separated. We will use two different lines of arguments for scheduling jobs with very small LP completion time and for the remaining jobs.

Case I: Scheduling jobs with very small LP completion time. Consider the jobs $J_0 := \{j \in J \mid 0 \leq C_j \leq \Theta(\frac{c}{\log^2 n})\}$ whose LP completion time C_j is much smaller than the communication delay. Jobs with tiny C_j have to be scheduled in the first interval with sufficiently high probability, and moreover the position within the first interval has to be proportional to C_j as well. To achieve this, we run the CKR clustering with a rather small parameter $\Delta := \Theta(\frac{1}{\log n})$ and denote $J_0 = V_1 \dot\cup \ldots \dot\cup V_q$ as the obtained partition. Clustering with such small diameter parameter allows us to prove structural insights about clusters that are new and were not required for the setting of [DKR+20]. If we consider a job $j \in J_0$, one can prove that all its predecessors $j' \prec j$ have a distance of $d(j, j') \leq \frac{C_j}{c}$. In particular, the probability that j gets separated from any of its ancestors is bounded by $O(\log^2(n) \cdot \frac{C_j}{c})$. Let us denote $V'_\ell \subseteq V_\ell$ as the jobs that are not separated from any ancestor and let $J'_0 := \bigcup_{\ell=1}^{q} V'_\ell$ be their union. Note that any set V'_ℓ could be processed on a single machine starting at time 0 without the danger of violating any precedence or communication delay constraints. Consequently, we will schedule the jobs in J'_0 right at the beginning of the schedule; after all of them are completed, we will schedule the jobs in $J_0 \setminus J'_0$.

- **Scheduling J'_0.** This is the most delicate part of the algorithm, as for the jobs in J'_0 even the relative order of the jobs within the sets V'_ℓ has to be decided. Moreover, there is no obvious bijection between the $V'_1, \ldots, V'_q$ and the machines, and we do not have a uniform upper bound on the size of the sets V'_ℓ. But we can prove a novel structural lemma that for each V'_ℓ there is a machine type k such that $|V'_\ell| \leq O(s_k c)$ and the LP solution schedules *every* job $j \in V'_\ell$ to an extent of $\Omega(\frac{1}{\log n})$ on machines of class k. Then we assign V'_ℓ to a random machine in that speed class k. Now consider the situation of one machine i of class k and let $J'_0(i,k)$ be the union of jobs assigned to this machine. The order we choose for sequencing the jobs in $J'_0(i,k)$ is the increasing order of *α-points*, which for us is the time when the LP has processed a $(1 - \Theta(\frac{1}{\log n}))$-fraction of job j. Again using properties implied by the Sherali-Adams lift combined with a Chernoff bound we can prove that for every θ, the number of jobs in $J'_0(i,k)$ with α-point at most θ is at most $O(\log^2 n) \cdot s_k \cdot \theta$. We can then conclude that every job in J'_0 is indeed completed by time $O(\log^3 n) \cdot C_j$.
- **Scheduling $J''_0 := J_0 \setminus J'_0$.** As the probability for a job $j \in J_0$ to end up in this case is small enough, it suffices to schedule all the jobs J''_0 in a time horizon of $O(\log n) \cdot c$ without caring about their particular order. In fact, we can simply use the same procedure that we are about to describe for Case II.

Overall we can see that adding up the contributions from both cases, the expected completion time of a job $j \in J_0$ will be $O(\log^3(n) \cdot C_j) + O(\log^2 n \cdot \frac{C_j}{c}) \cdot O(\log n \cdot c) \leq O(\log^3(n) \cdot C_j)$ as required.

Case II: Scheduling jobs with lower-bounded LP completion time. The main result to handle this case is that any subset J_T of jobs with LP completion times $C_j \leq T$ for all $j \in J_T$ can be scheduled with makespan $O(\log^2 n) \cdot T + O(\log n) \cdot c$. The complete algorithm can then be easily obtained by running the argument for all $T \geq \Theta(\frac{c}{\log^2 n})$ that are powers of 2 and concatenating the obtained schedules. We can break the problem further down to scheduling jobs in a narrow band of the form $J^* := \{j \in J_T \mid C^* \leq C_j \leq C^* + \Theta(\frac{c}{\log n})\}$, i.e., jobs that have close LP completion times. Again we run the CKR clustering procedure, but this time with a larger parameter $\Delta := \frac{1}{4}$ (which will result in saving a $\log n$ factor in the approximation guarantee compared to choosing $\Delta = \Theta(\frac{1}{\log n})$). Again we denote the random partition by $J^* = V_1 \dot\cup \ldots \dot\cup V_q$ and define $V'_\ell \subseteq V_\ell$ to be the jobs that were not separated from any ancestor in J^*. Unfortunately in this regime of $\Delta = \Theta(1)$, the mentioned structural claim from Case I does not hold anymore. However we can

prove a weaker result that for every block V'_ℓ there is a speed class k with $|V'_\ell| \leq O(s_k c)$ such that the LP solution schedules jobs in V'_ℓ on *average* to an extend of $\Omega(\frac{1}{\log n})$ on class-k machines. Since in Case II, we do not have to decide an order within the blocks V'_ℓ, this weaker property will be sufficient. Repeating the random clustering $\log n$ times will schedule all jobs in J^*. Finally we use a load argument to conclude the bound on the makespan of J_T.

Conclusion and reduction to general processing times. Overall this line of arguments gives an $O(\log^3 n)$-approximation for $\mathsf{Q} \mid \mathsf{prec}, p_j = 1, c \mid \sum w_j C_j$. Moreover for the problem $\mathsf{Q} \mid \mathsf{prec}, p_j = 1, c \mid C_{\max}$ it suffices to consider instances with optimum value at least c and hence we can find an $O(\log^2 n)$-approximation for that setting.

Finally let us outline how to obtain an approximation to $\mathsf{Q} \mid \mathsf{prec}, c \mid \sum w_j C_j$ while losing at most another $O(\log n)$ factor. Consider jobs J that now have arbitrary integer processing times p_j. We perform the natural reduction and split each job into a chain of p_j many unit-length jobs. Then we run the $O(\log^3 n)$-approximation algorithm for $\mathsf{Q} \mid \mathsf{prec}, p_j = 1, c \mid \sum w_j C_j$. The obtained schedule can be interpreted as a schedule for the original length-p_j jobs that respects precedence constraints and communication delays, but uses preemption and migration. By standard arguments it suffices to schedule the jobs with completion time at most T so that the makespan is $O(\log n) \cdot T$. We use the information from the migratory schedule and create a new instance $(\tilde{J}, \tilde{\prec})$ where jobs that are scheduled within a length-c timeframe are merged and we have precedence constraints $j_1 \tilde{\prec} j_2$ whenever a job j_1 was finished before j_2 in the migratory schedule. We can prove that chains in the new partial order $\tilde{\prec}$ contain at most $O(\frac{T}{c})$ many jobs. This implies that a small modification of the Speed-based List Scheduling by Chudak and Shmoys [CS99] can be used to schedule $(\tilde{J}, \tilde{\prec})$ with makespan $O(\log n) \cdot T$. In particular, the penalty for taking communication delays into account is bounded by $c \cdot (|C| - 1) \leq O(T)$, where $C \subseteq \tilde{J}$ is the chain that defines the bottleneck in the algorithm. That concludes the $O(\log^4 n)$-approximation for $\mathsf{Q} \mid \mathsf{prec}, c \mid \sum w_j C_j$.

1.4 Outline

The rest of this paper is organized as follows. In Section 2 we give preliminaries on hierarchies, semimetric spaces, and a useful version of the Chernoff bound. In Section 3 we discuss our linear program and its properties. In Sections 4 and 5 we give our algorithm for the weighted sum of completion times objective in the special case of unit-size jobs. Then, in Section 6, we show how to reduce the general processing time case to the unit-size case. Together, Sections 4–6 constitute the proof of Theorem 1. Finally, in Section 7 we solve the makespan minimization version while saving one $O(\log n)$ factor, thus proving Theorem 2.

2 Preliminaries

2.1 The Sherali-Adams Hierarchy for LPs with Assignment Constraints

The *Sherali-Adams hierarchy* systematically strengthens linear programs. Our notation for the Sherali-Adams hierarchy is adapted from that of Friggstad et al. [FKK+14]. Let $[n] = \{1, \ldots, n\}$ be a set of indices of variables and let $U_1, \ldots, U_N \subseteq [n]$ be subsets of them. Given these subsets of variable indices, we study the polytope

$$K = \left\{ x \in \mathbb{R}^n \mid \tilde{A}x \geq \tilde{b},\ \sum_{i \in U_h} x_i = 1 \ \ \forall h \in [N],\ \ 0 \leq x_i \leq 1 \ \ \forall i \in [n] \right\}$$

or more compactly $K = \{x \in \mathbb{R}^n \mid Ax \geq b\}$ for $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. A non-standard piece of our definition is that the constraint matrix contains *assignment constraints* $\sum_{i \in U_h} x_i = 1$.

Overall, we want a strong relaxation for the integer hull $\text{conv}(K \cap \{0,1\}^n)$. We motivate how we can obtain this with a Sherali-Adams lift through a probabilistic lens. The set of points $x \in \text{conv}(K \cap \{0,1\}^n)$ can be viewed as a *probability distribution* X defined by the 2^n values $y_I = \Pr[\bigwedge_{i \in I}(X_i = 1)]$ for $I \subseteq [n]$. Note that from the *inclusion-exclusion formula* we can, albeit inefficiently, define the probability of any event; for example $\Pr[X_1 = 1 \text{ and } X_2 = 0] = y_{\{1\}} - y_{\{1,2\}}$. In order to keep the size of the lifted LP polynomial, we only enforce constraints such that $K \cap \{0,1\}^n$ looks locally like a probability distribution, giving rise only to variables y_I for $|I| \leq O(1)$.

Definition 3. *Let $SA_r(K)$ be the set of vectors $y \in \mathbb{R}^{\mathcal{P}_{r+1}([n])}$ satisfying $y_\emptyset = 1$ and*

$$\sum_{H \subseteq J} (-1)^{|H|} \cdot \Big(\sum_{i=1}^{n} A_{\ell,i} y_{I \cup H \cup \{i\}} - b_\ell y_{I \cup H} \Big) \geq 0 \quad \forall \ell \in [m]$$

for all $I, J \subseteq [n]$ with $|I| + |J| \leq r$.

We call the parameter r above the *rank* or *number of rounds* of the Sherali-Adams lift. Observe that one can set $I = J = \emptyset$ to see that $(y_{\{1\}}, \ldots, y_{\{n\}}) \in K$ and for any $x \in K \cap \{0,1\}^n$ one can set $y_I := \prod_{i \in I} x_i$ to obtain a vector $y \in SA_r(K)$.

More on hierarchies can be found in the work by Laurent [Lau03]. In the properties that follows, for $r \geq 0$ we let $\mathcal{P}_r([n]) := \{S \subseteq [n] \mid |S| \leq r\}$ be the set of index sets with size at most r.

Theorem 4 (Properties of Sherali-Adams)**.** *Let $y \in SA_r(K)$ for some $r \geq 0$. Then the following holds:*

(a) For $J \in \mathcal{P}_r([n])$ with $y_J > 0$, the vector $\tilde{y} \in \mathbb{R}^{\mathcal{P}_{r+1-|J|}([n])}$ defined by $\tilde{y}_I := \frac{y_{I \cup J}}{y_J}$ satisfies $\tilde{y} \in SA_{r-|J|}(K)$.

(b) One has $0 \leq y_I \leq y_J \leq 1$ for $J \subseteq I$ and $|I| \leq r+1$.

(c) If $|J| \leq r+1$ and $y_i \in \{0,1\}$ $\forall i \in J$, then $y_I = y_{I \setminus J} \cdot \prod_{i \in I \cap J} y_i$ for all $|I| \leq r+1$.

(d) For $J \subseteq [n]$ with $|J| \leq r$ there exists a distribution over vectors $\tilde{y}$ such that (i) $\tilde{y} \in SA_{r-|J|}(K)$, (ii) $\tilde{y}_i \in \{0,1\}$ for $i \in J$, (iii) $y_I = \mathbb{E}[\tilde{y}_I]$ for all $I \subseteq [n]$ with $|I \cup J| \leq r+1$ (this includes in particular all $I \in \mathcal{P}_{r+1-|J|}([n])$).

(e) For $I \subseteq [n]$ with $|I| \leq r$ and $h \in [N]$ one has $y_I = \sum_{i \in U_h} y_{I \cup \{i\}}$.

(f) Take $H \subseteq [N]$ with $|H| \leq r$ and set $J := \bigcup_{h \in H} U_h$. Then there exists a distribution over vectors $\tilde{y}$ such that (i) $\tilde{y} \in SA_{r-|H|}(K)$, (ii) $\tilde{y}_i \in \{0,1\}$ for $i \in J$, (iii) $y_I = \mathbb{E}[\tilde{y}_I]$ for all $I \in \mathcal{P}_{r+1-|H|}([n])$.

Proofs of properties (a)-(d) can be found in Laurent [Lau03], while proof of properties (e) and (f) can be found in [DKR+20]. For a Sherali-Adams lift $y \in SA_r(K)$, it will be convenient later to use the notation $\tilde{y} \sim \mathcal{D}_y(H)$ with $|H| \leq r$ for the distribution described in Theorem 4.(f). Often we will omit the vector y if the Sherali-Adams lift is clear from the context. If we consider indices $J \subseteq U_h$ for some $h \in [N]$ then we can interpret these as an event $\mathcal{E} = \text{“}\{\exists i \in J : x_i = 1\}''$ and the probability of this event is $\Pr_{\tilde{y} \sim \mathcal{D}_y(h)}[\mathcal{E} \text{ holds in } \tilde{y}] = \sum_{i \in J} y_i$. Assuming that the probability of this event is positive, we can obtain a new Sherali-Adams lift conditioned on that event to happen while losing at most one round in the rank of the solution. We summarize the properties that we require later:

Lemma 5. *Let $y \in SA_r(K)$ for some $r \geq 0$. Fix an index $h \in [N]$ and fix $J \subseteq U_h$ with $\sum_{i \in J} y_i > 0$. Define $\bar{y} \in \mathbb{R}^{\mathcal{P}_r([n])}$ with $\bar{y}_I := \frac{\sum_{i \in J} y_{I \cup \{i\}}}{\sum_{i \in J} y_i}$ for $I \in \mathcal{P}_r([n])$. Then $\bar{y} \in SA_{r-1}(K)$ and moreover $\bar{y}_I \leq \frac{y_I}{\sum_{i \in J} y_i}$.*

Proof. Abbreviate $\gamma := \sum_{i \in J} y_i > 0$ and $J_+ := \{i \in J \mid y_i > 0\}$. For $i \in J_+$ we denote $y^{(i)} \in \mathcal{P}_r([n])$ as the vector with $y^{(i)}_I := \frac{y_{I \cup \{i\}}}{y_i}$. By Theorem 4.(a) we know that $y^{(i)} \in SA_{r-1}(K)$. Then y is a convex combination of the vectors $y^{(i)}$ as one can see from $\bar{y}_I = \sum_{i \in J_+} \frac{y_i}{\gamma} \cdot y^{(i)}_I$. By convexity of $SA_{r-1}(K)$, this implies that $\bar{y} \in SA_{r-1}(K)$. The moreover part follows from $\sum_{i \in J} y_{I \cup \{i\}} \leq \sum_{i \in U_h} y_{I \cup \{i\}} = y_I$, using Theorem 4.(e). □

Suppose that $\bar{y} \in SA_{r-1}(K)$ is the vector obtained by conditioning on the event $\mathcal{E} = \text{“}\{\exists i \in J : x_i = 1\}''$ according to Lemma 5 with $J \subseteq U_h$. Then we will also abbreviate the distribution $\mathcal{D}_{\bar{y}}(h')$ more conveniently as the conditional distribution $\mathcal{D}_y(h' \mid \mathcal{E})$.

2.2 Semimetric Spaces

A *metric space* is a pair (V, d) where V is a finite set (we denote $n := |V|$) and $d : V \times V \to \mathbb{R}_{\geq 0}$ is a *metric*, i.e.,

(I) $d(u, v) > 0 \Leftrightarrow (u \neq v)$ for all $u, v \in V$.

(II) Symmetry: $d(u, v) = d(v, u)$ for all $u, v \in V$.

(III) Triangle inequality: $d(u, v) + d(v, w) \geq d(u, w)$ for all $u, v, w \in V$.

Throughout this paper we will be working with a slight generalization of a *semi-metric* d where (I) is replaced by the weaker condition $d(u, u) = 0$ (meaning that it is possible that $d(u, v) = 0$ for $u \neq v$). For a set $U \subseteq V$ we denote the *diameter* as $\mathrm{diam}(U) := \max_{u,v \in U} d(u, v)$. Our goal is to find a random partition $V = V_1 \dot{\cup} \ldots \dot{\cup} V_q$ such that the diameter of every cluster V_i is bounded by some parameter Δ. We say that a set U is *separated* by such a clustering if there is more than one index i with $V_i \cap U \neq \emptyset$. Moreover, we denote $d(w, U) := \min\{d(w, u) : u \in U\}$ as the distance to the set U.

We use the very influential clustering algorithm due to Calinescu, Karloff and Rabani [CKR04], which assigns each node $v \in V$ to a random cluster center $c \in V$ such that $d(u, c) \leq \beta\Delta$, for a random parameter β. Nodes assigned to the same cluster center form one block V_i in the partition.

CKR Clustering algorithm

Input: Semimetric space (V, d) with $V = \{v_1, \ldots, v_n\}$, parameter $\Delta > 0$.
Output: Clustering $V = V_1 \dot{\cup} \ldots \dot{\cup} V_q$ for some q.

(1) Pick a uniform random $\beta \in [\frac{1}{4}, \frac{1}{2}]$.
(2) Pick a random ordering $\pi : V \to \{1, \ldots, n\}$.
(3) For each $v \in V$ set $\sigma(v) := v_\ell$ so that $d(v, v_\ell) \leq \beta \cdot \Delta$ and $\pi(v_\ell)$ is minimal.
(4) Denote the points $v \in V$ with $\sigma^{-1}(v) \neq \emptyset$ by $c_1, \ldots, c_q \in V$ and return clusters $V_i := \sigma^{-1}(c_i)$ for $i = 1, \ldots, q$.

A key trick for the analysis are the two sources of randomness: the algorithm picks a random parameter β, and independently selects a random ordering π. Here the ordering is to be understood so that element v_ℓ with $\pi(v_\ell) = 1$ is the “highest priority” element.

The original work of Calinescu, Karloff and Rabani [CKR04] only provided an upper bound on the probability that an edge (u, v) is separated. Mendel and Naor [MN06] note that the same clustering provides the guarantee of $\Pr[N(u,t) \text{ separated}] \leq 1 - O(\frac{t}{\Delta} \cdot \ln(\frac{|N(u,\Delta)|}{|N(u,\Delta/8)|}))$ for all $u \in V$ and $0 \leq t < \frac{\Delta}{8}$. Here, for $r \geq 0$ and $U \subseteq V$, $N(U, r) := \{v \in V \mid d(v, U) \leq r\}$ denotes the *distance r-neighborhood* of U. Mendel and Naor attribute this to Fakcharoenphol, Rao and Talwar [FRT04] (while Fakcharoenphol, Rao and Talwar [FRT04] do not state it explicitly in this form and focus on the "local growth ratio" aspect).

We state the formal claim in a form that will be convenient for us. For a self-contained proof see for example the appendix of [DKR+20].

Theorem 6 (Analysis of CKR)**.** *Let $V = V_1 \dot\cup \ldots \dot\cup V_q$ be the random partition of the CKR algorithm. The following holds:*

(a) The blocks have $diam(V_i) \leq \Delta$ for $i = 1, \ldots, q$.
(b) Let $U \subseteq V$ be a subset of points. Then

$$\Pr[U \text{ is separated by clustering}] \leq \ln\big(2\big|N\big(U, \Delta/2\big)\big|\big) \cdot \frac{4diam(U)}{\Delta} \leq \ln(2n) \cdot \frac{4diam(U)}{\Delta}.$$

It should be pointed out that it was not absolutely necessary to use the algorithm by Calinescu, Karloff and Rabani [CKR04]. One could have extracted a suitable clustering also using the *region growing technique*, see Leighton and Rao [LR99] or Garg, Vazirani and Yannakakis [GVY93].

2.3 Concentration

We will also make use of the following version of the Chernoff bound. See for example the textbook of Dubhashi and Panconesi [DP09].

Lemma 7. *There is a universal constant $C > 0$ such that the following holds. Let $X := X_1 + \ldots + X_n$ be a sum of independent random variables with $0 \leq X_\ell \leq \alpha$ for all $\ell \in [n]$ and $\mathbb{E}[X] \leq \alpha$ for some $\alpha > 0$. Then for any $N \geq 4$ one has $\Pr[X > C\frac{\log N}{\log\log N} \cdot \alpha] \leq \frac{1}{N}$.*

3 The Linear Program and Its Properties

Let J be a set of $|J| = n$ jobs, each with a *processing time* $p_j = 1$ and a weight $w_j \geq 0$. Let $\{1, \ldots, m\}$ be the indices of the available machines where we assume to have m_k machines of speed $s_k \in \mathbb{N}$. We let M be the number of different machine *types*, also referred to as speed classes. By the standard argument of geometrically grouping machines with speeds within a constant factor of each other [Li17, MRS+20], we can assume that $M \leq O(\log(\frac{s_{\max}}{s_{\min}})) \leq O(\log m)$ while we lose a constant factor in the approximation. The goal is to find a schedule such that jobs $j_1 \prec j_2$ are either scheduled on the same machine (with j_1 finishing before j_2 is started) or they are scheduled on different machines and j_2 starts at least c time units after j_1 is finished. By a slight abuse of notation we denote $[m_k]$ as the indices of the machines of speed s_k.

3.1 The Linear Program

It will be convenient to partition the time horizon into *intervals* $I_0, I_1, \ldots, I_{L-1}$ of c time units each. As machines have different speeds and hence can handle different numbers of jobs per interval, we abbreviate the discrete set of time slots that a speed-s_k machine has in I_ℓ as $I_{\ell,k} := \{\ell \cdot c + \frac{t}{s_k} : t =$

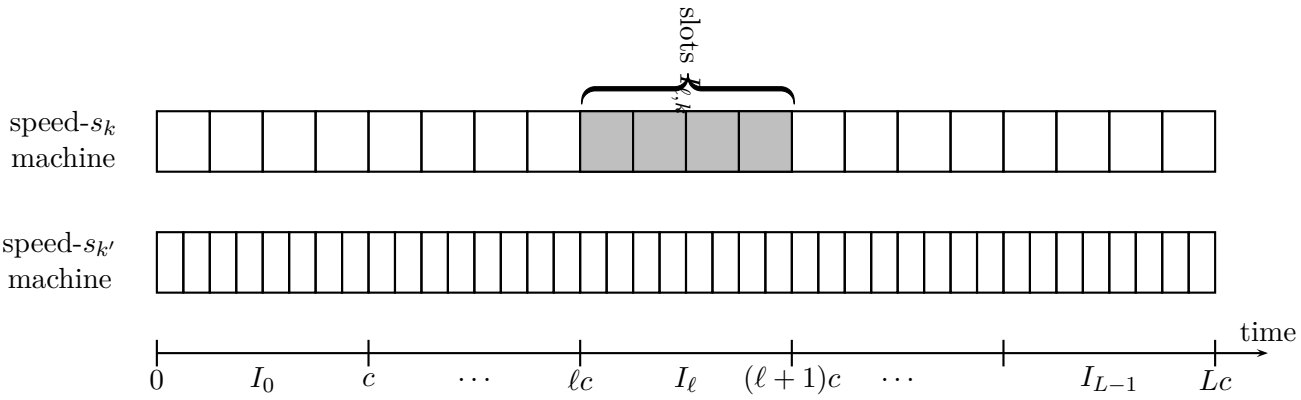

Figure 1: Slots for different machine types.

$1, \ldots, s_k c\}$. We note that indeed $|I_{\ell,k}| = s_k c$. Moreover we set $I_{*,k} := \bigcup_{\ell=0}^{L-1} I_{\ell,k}$ as all time slots where a speed-s_k machine can schedule jobs.

We construct the LP in two steps. First consider the variables

$$x_{j,i,t} = \begin{cases} 1 & \text{if } j \text{ is scheduled on machine } i \text{ in time slot } t \in I_{*,k}\,, \\ 0 & \text{otherwise} \end{cases}$$

for all $j \in J$, $k \in [M]$, $i \in [m_k]$, and $t \in I_{*,k}$. Let K be the set of fractional solutions to the following linear system:

$$\begin{aligned} \sum_{k\in[M], i\in[m_k]} \sum_{t\in I_{*,k}} x_{j,i,t} &= 1 \quad \forall j \in J \\ \sum_{j\in J} x_{j,i,t} &\leq 1 \quad \forall k \in [M]\ \ \forall i \in [m_k]\ \ \forall t \in I_{*,k} \\ 0 \leq x_{j,i,t} &\leq 1 \quad \forall j \in J\ \forall k \in [M]\ \ \forall i \in [m_k]\ \forall t \in I_{*,k} \end{aligned}$$

Next, we will use a lift $x \in \mathrm{SA}_r(K)$, which contains in particular the variables $x_{(j_1,i_1,t_1),(j_2,i_2,t_2)}$ that provide the probability for the event that j_1 is scheduled at time t_1 on machine i_1 and j_2 is scheduled at time t_2 on machine i_2. We introduce more types of decision variables:

$$\begin{aligned} y_{j_1,j_2,k} &= \begin{cases} 1 & j_1 \text{ and } j_2 \text{ are scheduled on the same machine of type } k \text{ in the same interval,} \\ 0 & \text{otherwise,} \end{cases} \\ y_{j_1,j_2} &= \begin{cases} 1 & j_1 \text{ and } j_2 \text{ are scheduled on the same machine in the same interval,} \\ 0 & \text{otherwise,} \end{cases} \\ C_j &= \text{completion time of job } j. \end{aligned}$$

The LP relaxation is then as follows:

$$\begin{aligned}
\text{Minimize} \quad & \sum_{j\in J} w_j \cdot C_j \quad (\mathsf{LP}) \\
y_{j_1,j_2,k} &= \sum_{\ell\in\{0,\dots,L-1\}} \sum_{i\in[m_k]} \sum_{t_1\in I_{\ell,k}} \sum_{t_2\in I_{\ell,k}} x_{(j_1,i,t_1),(j_2,i,t_2)} && \forall j_1,j_2\in J\ \forall k\in[M] \\
y_{j_1,j_2} &= \sum_k y_{j_1,j_2,k} && \forall j_1,j_2\in J \\
C_{j_2} &\geq C_{j_1} + (1-y_{j_1,j_2})\cdot c && \forall j_1 \prec j_2 \\
C_j &= \sum_{k\in[M],i\in[m_k]} \sum_{t\in I_{*,k}} x_{j,i,t}\cdot t && \forall j\in J \\
x &\in SA_r(K)
\end{aligned}$$

Following the discussion in Section 2.1, we know that for every job j, there is a distribution that we denote as $(\tilde{x},\tilde{y})\sim\mathcal{D}(j^*)$ such that $\mathbb{E}[\tilde{x}_{j,i,t}]=x_{j,i,t}$ and $\mathbb{E}[\tilde{y}_{j_1,j_2}]=y_{j_1,j_2}$ with $\tilde{x}\in SA_{r-1}(K)$ where job j^* is integrally assigned. This is immediate for the x-part as $x\in SA_r(K)$, and follows for the y-variables as these linear in the x-variables. If $\mathcal{E}$ is an event, then we write $(\tilde{x},\tilde{y})\sim\mathcal{D}(j^*\mid\mathcal{E})$ as the conditional distribution (conditioning on the event $\mathcal{E}$ occurring), see again Section 2.1 for details.

3.2 Properties of the LP

We will now discuss some properties that are implied by the Sherali-Adams lift. The properties proved in this section, which we use crucially in our rounding algorithms, are the main technical contributions of this paper.

Lemma 8. *Let (x,y,C) be a solution to (LP) with $r\geq 5$. Then $d(j_1,j_2):=1-y_{j_1,j_2}$ is a semimetric.*

This property was proven in [DKR$^+$20] for a special case of $s_k=1$ and is not hard to extend to our more general LP. For the sake of completeness, we give the proof in Appendix A. From now on, the symbol d as well as the quantity $\operatorname{diam}(\cdot)$ will always refer to this particular semimetric. For the special case of $s_k=1$ for all k, one can also find a proof in [DKR$^+$20] that any set $U\subseteq J$ with $\operatorname{diam}(U)\leq\frac{1}{2}$ has size $|U|\leq 2c$. While this is a obviously false for arbitrary speeds s_k, we can prove a similar claim:

Lemma 9. *For $k\in[M]$ and $j_1\in J$ one has $\sum_{j_2\in J} y_{j_1,j_2,k}\leq s_k\cdot c\cdot\sum_{i\in[m_k]}\sum_t x_{j_1,i,t}\leq s_k\cdot c$.*

Proof. The second inequality is trivial as $\sum_{i\in[m_k]}\sum_t x_{j_1,i,t}\leq 1$, so we only justify the first inequality. As we are proving a linear inequality, it suffices to show this for a fixed outcome $(\tilde{x},\tilde{y})\sim\mathcal{D}(j_1)$ where job j_1 is assigned integrally. If in $(\tilde{x},\tilde{y})$ the job j_1 is not assigned to a machine in $[m_k]$, then both sides are 0. So suppose that j_1 is assigned to a machine $i_1\in[m_k]$ and to interval ℓ_1. Then indeed

$$\sum_{j_2\in J}\tilde{y}_{j_1,j_2,k}=\sum_{j_2\in J}\sum_{t\in I_{\ell_1,k}}\tilde{x}_{j_2,i_1,t}\leq s_k c=s_k c\underbrace{\sum_{i\in[m_k]}\sum_t\tilde{x}_{j_1,i,t}}_{=1}.$$

□

Lemmas 10 and 11 are key technical insights behind the algorithms for the related machines setting. They say that if one considers a set of jobs which are close to each other with respect to

d, then there exists a type k^* such that we can schedule all the jobs in an interval of length $O(c)$ on a single machine of type k^*. Moreover, the LP also schedules a good fraction of these jobs on the same machine type. These lemmas are important in assigning jobs to machine types in our algorithms.

Lemma 10. *Let $U \subseteq J$ be a non-empty subset of jobs with $diam(U) \leq \frac{1}{4}$. Then there exists a $k^* \in [M]$ such that*

(i) $|U| \leq O(1) \cdot s_{k^*} \cdot c$, *and*
(ii) $\sum_{j \in U} \sum_{i \in [m_{k^*}]} \sum_t x_{j,i,t} \geq \Omega(\frac{1}{M}) \cdot |U|$.

Proof. Let us sort the speed classes $[M]$ so that $s_1 \geq \ldots \geq s_M$ and abbreviate $z_{j,k} := \sum_{i \in [m_k]} \sum_t x_{j,i,t}$ as the fraction of job j scheduled on class k. Moreover let $\rho_k := \sum_{j \in U} z_{j,k}$ be the load of cluster U on class k. We fix an arbitrary job $j^* \in U$. Let $k_{\text{median}} \in [M]$ be the median speed class for job j^*, meaning that $\sum_{k \leq k_{\text{median}}} z_{j^*,k} \geq \frac{1}{2}$ and $\sum_{k \geq k_{\text{median}}} z_{j^*,k} \geq \frac{1}{2}$. We split the remaining proof into two separate claims.

Claim I. *One has $|U| \leq 4 \cdot s_{k_{median}} \cdot c$.*

Proof of Claim I. First we observe that for every $j \in U$ we have $\sum_{k \geq k_{\text{median}}} y_{j^*,j,k} \geq \frac{1}{2} - d(j^*, j) \geq \frac{1}{4}$. We use this to estimate

$$\begin{aligned} \frac{|U|}{4} &\leq \sum_{k \geq k_{\text{median}}} \sum_{j \in U} y_{j^*,j,k} \overset{\text{Lem 9}}{\leq} \sum_{k \geq k_{\text{median}}} s_k c \sum_{i \in [m_k]} \sum_t x_{j^*,i,t} \\ &\leq s_{k_{\text{median}}} c \underbrace{\sum_{k \geq k_{\text{median}}} \sum_{i \in [m_k]} \sum_t x_{j^*,i,t}}_{\leq 1} \leq s_{k_{\text{median}}} c \end{aligned}$$

Rearranging gives $|U| \leq 4 s_{k_{\text{median}}} c$ as claimed. □

Claim II. *There is a class $k^* \in \{1, \ldots, k_{median}\}$ where $\rho_{k^*} \geq \frac{|U|}{4M}$.*

Proof of Claim II. For any job $j \in U$ we have $\sum_{k \leq k_{\text{median}}} z_{j,k} \geq (\sum_{k \leq k_{\text{median}}} z_{j^*,k}) - d(j, j^*) \geq \frac{1}{2} - \frac{1}{4} = \frac{1}{4}$. Hence $\sum_{k \leq k_{\text{median}}} \rho_k \geq \frac{|U|}{4}$. Then at least one index $k^* \in \{1, \ldots, k_{\text{median}}\}$ will have $\rho_{k^*} \geq \frac{|U|}{4M}$. □

While the above lemma is enough to obtain our approximation algorithm for makespan on related machines, for the weighted completion time objective we need the following strengthening: if one considers a set of jobs which are *very* close to each other, then there exists a type k^* such that we can schedule all the jobs in an interval of length $O(c)$ on a single machine of type k^*. Moreover, the LP solution also schedules at least $O(1/M)$ fraction of *every one of these jobs* on the same machine type. Recall that Lemma 10 only satisfied this condition on average.

Lemma 11. *Let $U \subseteq J$ be a non-empty set with $diam(U) \leq \frac{1}{8M}$ and abbreviate $z_{j,k} := \sum_{i \in [m_k]} \sum_t x_{j,i,t}$. Then*

(i) $|z_{j_1,k} - z_{j_2,k}| \leq diam(U) \leq \frac{1}{8M}$ *for all $j_1, j_2 \in U$ and $k \in [M]$.*

Moreover there are indices $\pi(U) \subseteq [M]$ such that

(ii) $z_{j,k} \geq \frac{1}{4M}$ *for all $j \in U$ and $k \in \pi(U)$*

(iii) $\sum_{k \in \pi(U)} \min\{z_{j,k} : j \in U\} \geq \frac{1}{2}$

(iv) $|U| \le 2 \cdot s_k \cdot c$ *for all* $k \in \pi(U)$.

Proof. We prove the points in order.

(i) Actually we claim the stronger property of $|z_{j_1,k} - z_{j_2,k}| \le d(j_1, j_2)$. Sample a distribution $(\tilde{x}, \tilde{y}) \sim \mathcal{D}(j_1, j_2)$ and let $\sigma(j_1) \in [m]$ be the random variable that denotes the machine index with $\sum_t \tilde{x}_{j_1,\sigma(j_1),t} = 1$ (similarly we define $\sigma(j_2)$). Then we can see that

$$|z_{j_1,k} - z_{j_2,k}| = |\Pr[\sigma(j_1) \in [m_k]] - \Pr[\sigma(j_2) \in [m_k]]| \le |\Pr[\sigma(j_1) \neq \sigma(j_2)]| \le d(j_1, j_2).$$

Now we fix any job $j^* \in U$ and set $\pi(U) := \{k \in [M] : z_{j^*,k} \ge \frac{1}{4M}\}$. Then we continue the proof:

(ii) By (i) we know that for each $j \in U$ and $k \in \pi(U)$ we have $z_{j,k} \ge z_{j^*,k} - \frac{1}{8M} \ge \frac{1}{4M}$.

(iii) We have

$$\begin{aligned}\sum_{k\in\pi(U)} \min\{z_{j,k} : j \in U\} &\ge \sum_{k\in\pi(U)} \left(z_{j^*,k} - \frac{1}{8M}\right) \\ &\ge \underbrace{\sum_{k\in[M]} z_{j^*,k}}_{=1} - \sum_{k\in[M]\setminus\pi(U)} \underbrace{z_{j^*,k}}_{\le 1/(4M)} - \frac{1}{8M}\underbrace{|\pi(U)|}_{\le M} \ge \frac{1}{2}\end{aligned}$$

(iv) Fix a machine type $k \in \pi(U)$ and consider the event $\mathcal{E} := \text{“}\sum_{i\in[m_k]} \sum_t \tilde{x}_{j^*,i,t} = 1\text{”}$ (meaning the event that j^* is assigned to a machine in class k). Note that $\Pr_{(\tilde{x},\tilde{y})\sim\mathcal{D}(j^*)}[\mathcal{E}] = z_{j^*,k} \ge \frac{1}{4M}$. Now, let $(\bar{x}, \bar{y})$ be the Sherali-Adams lift (see Lemma 5) conditioned on the event $\mathcal{E}$. Then trivially $\sum_{i\in[m_k]} \sum_t \bar{x}_{j^*,i,t} = 1$. As we have conditioned on an event with probability $z_{j^*,k} \ge \frac{1}{4M}$, the chance of other events cannot increase by more than a factor of $4M$ (see again Lemma 5). In particular for $j \in U$ we have $1 - \bar{y}_{j^*,j} \le 4M \cdot (1 - y_{j^*,j}) = 4M \cdot d(j^*, j) \le \frac{1}{2}$. As $\bar{y}_{j^*,j,k'} = 0$ for all $k' \neq k$ and $j \in U$ we can conclude that $\bar{y}_{j^*,j,k} \ge \frac{1}{2}$ for all $j \in U$. Finally by Lemma 9 we know that $\sum_{j\in J} \bar{y}_{j^*,j,k} \le s_k c$ which then gives $|U| \le 2 s_k c$.

□

4 The Rounding Algorithm for $\mathsf{Q} \mid \mathsf{prec}, p_j = 1, c \mid \sum w_j C_j$

In this section, we describe the rounding algorithm which proves our main technical result, Theorem 12. We let C_j^A denote the algorithm's completion time of job j. We denote $\Gamma^-(j)$ as the predecessors of j and $\Gamma^+(j)$ as the successors, and similarly $\Gamma^{-/+}(J') = \{j \in J : \exists j' \in J' \text{ s.t. } j \in \Gamma^{-/+}(j')\}$.

Theorem 12. *There is a polynomial-time randomized algorithm that, given a solution* (x, y, C) *to (LP) for* $r \ge 5$*, produces a schedule with completion times* $\{C_j^A\}_{j\in J}$ *such that*

$$\mathbb{E}[C_j^A] \le O(M \cdot \log^2 n) \cdot C_j \le O(\log^3 n) \cdot C_j.$$

Before we describe our rounding algorithm, we set up some notation. Let (x, y, C) be an optimal solution to the LP with $r \ge 5$. We partition the jobs based on their fractional completion times in the LP solution. For any constant $a > 128$ set $\delta = \frac{1}{a \cdot M \cdot \log 2n}$. Consider

$$J_0 := \{j : C_j < \delta \cdot c\} \tag{1}$$

and for $\gamma \geq 1$ define

$$J_\gamma := \left\{ j : 2^{\gamma-1} \cdot \delta \cdot c \leq C_j < 2^\gamma \cdot \delta \cdot c \right\} \tag{2}$$

Our rounding algorithm has the following steps:

- Schedule J_0 via the first batch scheduling algorithm given in Section 4.1
- For $\gamma = 1, 2, \ldots$, schedule J_γ via the intermediate scheduling algorithm given in Section 4.2.
- Concatenate the schedules of $J_0, J_1, J_2, \ldots$.

We show that the expected completion time of every job in the above schedule satisfies Theorem 12. We need the following scheduling subroutine for the rounding steps above.

Theorem 13. *Let (x, y, C) be a solution to the LP with $r \geq 5$ and let $T^* \in \mathbb{N}$. For some subset $J' \subset J$, suppose in the LP solution completion time $C_j \leq T^*$ for every job $j \in J'$. Then there is a randomized polynomial time algorithm that schedules all jobs such that $C_j^A \leq O(\log m \cdot \log n \cdot T^*) + O(\log m \cdot c)$ for every job $j \in J'$.*

Note that the above theorem immediately implies our result for the makespan minimization problem on related machines. We prove this theorem in Section 5.

4.1 First Batch Scheduling

Here we give an algorithm for scheduling jobs in J_0. We define the *α-point* of job j as the earliest time t_j^* when the LP solution has completed an α-fraction of j. Formally,

$$t_j^* := \min \left\{ t' \in [T] : \sum_{i=1}^{m} \sum_{t \leq t'} x_{j,i,t} \geq \alpha \right\}. \tag{3}$$

For $\beta \leq 1/4M$, consider any subset $U \subseteq J$ with $\mathrm{diam}(U) \leq \beta$. Let $\pi(U) \subseteq [M]$ denote the indices of machine types that satisfy the conditions of Lemma 11. We use the same semimetric $d(j_1, j_2) := 1 - y_{j_1,j_2}$ and schedule jobs in J_0 using the following algorithm.

FIRST BATCH SCHEDULING

(1) Run a CKR clustering on the semimetric space (J_0, d) with parameter $\Delta := \frac{1}{100M}$ and let $V_1, \ldots, V_q$ be the clusters.
(2) Let $V'_\ell := \{j \in V_\ell \mid (\Gamma^-(j) \cap J_0) \subseteq V_\ell\}$ for $\ell = 1, \ldots, q$, and $J'_0 := \cup_{\ell=1}^{q} V'_\ell$.
(3) FOR $\ell = 1$ TO q DO
    (4) Sample a machine type k^* from the set $\pi(V'_\ell)$ with probability $\frac{\min\{z_{jk^*} : j \in V'_\ell\}}{\sum_{k \in \pi(V'_\ell)} \min\{z_{jk} : j \in V'_\ell\}}$.
    (5) Assign V'_ℓ to a machine $i \in [m_{k^*}]$ with probability $\frac{1}{m_{k^*}}$.
(6) For a machine $i \in [m_k]$ of type k, let $J'_0(i,k)$ denote the set of jobs assigned to machine i.
(7) For all k and for all $i \in [m_k]$, schedule $J'_0(i,k)$ in the increasing order of their α-points for $\alpha = (1 - \frac{1}{100M})$ as defined in Eq. (3).
(8) Insert a gap of c time slots.
(9) Let $J''_0 := J_0 \setminus J'_0$ be the set of jobs that did not get scheduled in steps (1) - (5). Use Theorem 13 to schedule J''_0.

We now argue that expected completion time of a job j in above algorithm is comparable to its LP cost. First we focus on bounding the completion time of jobs in J'_0. We need the following crucial lemma.

Lemma 14. *Let $U \subseteq J$ be a set of jobs with $diam(U) \leq \frac{1}{100M}$ w.r.t. distance d. For every job $j \in U$, define t^*_j as in Eq (3). For $\theta > 0$, consider the set of jobs $U^* := \{j \in U : t^*_j < \theta\}$ with α-point less than θ. Then $|U^*| \leq 2 \cdot s_{k^*} \cdot \theta$ for any $k^* \in \pi(U)$.*

The lemma formalizes the intuition that as the jobs in U^* are very close to each other, it must be the case that the LP schedules all of them on the same machine. As a good fraction of each of these jobs are scheduled in the interval of length θ there cannot be too many of them.

Proof. We prove the lemma by contradiction. Consider a set U^* that violates the conditions in the lemma and fix a job $j^* \in U^*$. Let $k^* \in \pi(U)$. Then we have

$$(A)\ \sum_{j\in U^*}\sum_{i\in[m]}\sum_{t>\theta} x_{j,i,t} < \frac{1}{100M}\cdot|U^*|, \quad (B)\ \sum_{j\in U^*} y_{j,j^*} \geq \Big(1-\frac{1}{100M}\Big)|U^*|, \quad (C)\ \sum_{i\in[m_{k^*}]}\sum_{t\leq\theta} x_{j^*,i,t} \geq \frac{1}{5M}.$$

where (A) follows from the definition of α-point of jobs, (C) follows from $\sum_{i\in[m_{k^*}]}\sum_{t=1}^{\theta} x_{j^*,i,t} \geq 1/4M - 1/100M > 1/5M$, and (B) follows from $\mathrm{diam}(U^*) \leq \mathrm{diam}(U) \leq \frac{1}{100M}$. Now we appeal to the properties of the Sherali-Adams hierarchy. Consider the event $\mathcal{E} := "\sum_{i\in[m_{k^*}]}\sum_{t\leq\theta} x_{j^*,i,t} = 1"$ and note that from (C) we know that the probability of the event $\mathcal{E}$ is at least $\frac{1}{5M}$. Now let $(\bar{x}, \bar{y})$ be the Sherali-Adams lift with $\bar{x} \in SA_{r-1}(K)$ conditioned on this event, see Lemma 5 for details. In particular one has $\sum_{i\in[m_{k^*}]}\sum_{t\leq\theta} \bar{x}_{j^*,i,t} = 1$, meaning that the LP solution $(\bar{x}, \bar{y})$ schedules j^* fully on machines of class k^* and until time θ.

As we have conditioned on an event of probability at least $\frac{1}{5M}$, the probabilities of other events cannot increase by more than a factor of $5M$ (see the "moreover" part in Lemma 5). In particular for $j \in U$ we have $1-\bar{y}_{j^*,j} \leq 5M\cdot(1-y_{j^*,j}) = 5M\cdot d(j^*,j) \leq \frac{1}{10}$. Similarly, $\sum_{j\in U^*}\sum_{i\in[m]}\sum_{t>\theta}\bar{x}_{j,i,t} < 5M\cdot(\sum_{j\in U^*}\sum_{i\in[m]}\sum_{t>\theta} x_{j,i,t}) \leq 5M\cdot\frac{1}{100M}\cdot|U^*| \leq \frac{|U^*|}{10}$. Therefore we have

$$(A')\ \sum_{j\in U^*}\sum_{i\in[m_{k^*}]}\sum_{t>\theta}\bar{x}_{j,i,t} \leq \frac{|U^*|}{10}, \quad (B')\ \sum_{j\in U^*}\bar{y}_{j,j^*} \geq \frac{9}{10}\cdot|U^*|, \quad (C')\ \sum_{i\in[m_{k^*}]}\sum_{t\leq\theta}\bar{x}_{j^*,i,t} = 1.$$

Then by Markov's inequality there exists an outcome $(\tilde{x}, \tilde{y}) \sim \mathcal{D}_{\bar{y}}(j^*)$ where the job j^* is integrally assigned and

$$(A'')\ \sum_{j\in U^*}\sum_{i\in[m_{k^*}]}\sum_{t>\theta}\tilde{x}_{j,i,t} \leq \frac{|U^*|}{5}, \quad (B'')\ \sum_{j\in U^*}\tilde{y}_{j,j^*} \geq \frac{4}{5}\cdot|U^*|, \quad (C'')\ \sum_{i\in[m_{k^*}]}\sum_{t\leq\theta}\tilde{x}_{j^*,i,t} = 1.$$

We fix that outcome and let $i^* \in [m_{k^*}], t^* \in \{1,\dots,\theta\}$ be the indices with $\tilde{x}_{j^*,i^*,t^*} = 1$. Let ℓ^* be the interval index such that $t^* \in I_{\ell^*,k^*}$. Then

$$\begin{aligned}
\frac{4}{5}\cdot|U^*| &\overset{(B'')}{\leq} \sum_{j\in U^*}\tilde{y}_{j,j^*} \overset{LP}{=} \sum_{j\in U^*}\sum_{\ell\in\{0,\dots,L-1\}}\sum_{k}\sum_{i\in[m_k]}\sum_{t_1\in I_{\ell,k}}\sum_{t_2\in I_{\ell,k}}\tilde{x}_{(j,i,t_1),(j^*,i,t_2)} \\
&\leq \sum_{j\in U^*}\sum_{t\in I_{\ell^*,k^*}}\tilde{x}_{j,i^*,t} = \underbrace{\sum_{j\in U^*}\sum_{t\in I_{\ell^*,k^*}:t\leq\theta}\tilde{x}_{j,i^*,t}}_{\leq s_{k^*}\cdot\theta \text{ by LP}} + \underbrace{\sum_{j\in U^*}\sum_{t\in I_{\ell^*,k^*}:t>\theta^*}\tilde{x}_{j,i^*t}}_{\leq\frac{1}{5}\cdot|U^*| \text{ by } (A'')} \\
&\leq s_{k^*}\cdot\theta + \frac{|U^*|}{5}
\end{aligned}$$

Rearranging gives $|U^*| \leq \frac{5}{3} \cdot s_{k^*} \cdot \theta^*$, which is a contradiction. □

Lemma 15. *Let $\theta > 0$ and i be any machine of type k. Then with high probability $1 - 1/n^{100}$ it holds that*

$$|\{j \in J_0'(i,k) : t_j^* \leq \theta\}| \leq O\Big(\frac{\log n}{\log\log n} \cdot s_k \cdot \theta\Big)$$

Proof. Fix $\theta^* > 0$ and any machine i^* of type k. Consider any outcome of the clustering itself. The randomness needed for the lemma is only over the assignment in step (3)-(5) of the algorithm. We define the following random variables. Let

$$X_\ell = \begin{cases} |\{j \in V_\ell' : t_j^* \leq \theta^*\}| & \text{if } V_\ell' \text{ is assigned to machine } i^* \text{ of type } k \text{ by our algorithm,} \\ 0 & \text{otherwise} \end{cases}$$

and abbreviate $X = \sum_{\ell=1}^{q} X_\ell$. Note the random variables $X_1, \ldots, X_q$ are independent and $X = |\{j \in J_0'(i^*,k) : t_j^* \leq \theta^*\}|$. From Lemma 14 we know that $X_\ell \leq 2s_k \cdot \theta^*$ for all ℓ.

Next, we estimate $\mathbb{E}[X]$. Recall that the algorithm uses the indices $\pi(V_\ell') \subseteq [M]$ from Lemma 11 which in particular satisfy $z_{j,k} \geq \frac{1}{4M}$ and $|z_{j,k} - z_{j',k}| \leq \frac{1}{8M}$ whenever $j, j' \in V_\ell'$ with $k \in \pi(V_\ell')$. Abbreviate

$$\mu_{\ell,k} := \begin{cases} \min\{z_{j,k} : j \in V_\ell'\} & \text{if } k \in \pi(V_\ell') \\ 0 & \text{otherwise.} \end{cases}$$

Recall that for every ℓ we have $\sum_{k\in[M]} \mu_{\ell,k} \geq 1/2$. We prove two technical claims that we need for our analysis:

Claim I. *For any ℓ and k and $j \in V_\ell'$ one has $\mu_{\ell,k} \leq 2z_{j,k}$.*

Proof of Claim. If $k \notin \pi(V_\ell')$ then the left hand side is 0, so suppose $k \in \pi(V_\ell')$. Then $\mu_{\ell,k} = \min\{z_{j',k} : j' \in V_\ell'\} \leq z_{j,k} + \frac{1}{8M} \leq 2z_{j,k}$. □

Claim II. *For $j \in V_\ell'$ with $t_j^* \leq \theta^*$ and $k \in \pi(V_\ell')$ one has $z_{j,k} \leq 2\sum_{i\in[m_k]} \sum_{t\in I_{*,k}:t\leq\theta^*} x_{j,i,t}$.*

Proof of Claim II. Since $k \in \pi(V_\ell')$ we know that $z_{j,k} \geq \frac{1}{4M}$. We consider the distribution $(\tilde{x}, \tilde{y}) \sim \mathcal{D}(j)$ and denote $\tilde{i}$ and $\tilde{t}$ as the random indices so that $\tilde{x}_{j,\tilde{i},\tilde{t}} = 1$. Then $\Pr[\tilde{i} \in [m_k]] = z_{j,k}$ and $\Pr[\tilde{t} > \theta^*] = \sum_{k\in[M]} \sum_{i\in[m_k]} \sum_{t\in I_{*,k}:t>\theta^*} x_{j,i,t} \leq \frac{1}{100M}$ by the definition of α-points and the assumption $t_j^* \leq \theta^*$. Then

$$\begin{aligned} \sum_{i\in[m_k]} \sum_{t\in I_{*,k}:t\leq\theta^*} x_{j,i,t} &= \Pr\left[\tilde{i} \in [m_k] \text{ and } \tilde{t} \leq \theta^*\right] \\ &\geq \Pr\left[\tilde{i} \in [m_k]\right] - \Pr\left[\tilde{t} > \theta^*\right] \\ &\geq z_{j,k} - \frac{1}{100M} \geq \frac{1}{2} z_{j,k} \end{aligned}$$ □

Now we can bound the expectation of our random variable as

$$\begin{aligned}
\mathbb{E}[X] &= \frac{1}{m_k}\sum_{\ell=1}^{q}\Pr\left[V'_\ell \text{ assigned to class } k\right]\cdot|\{j\in V'_\ell : t^*_j\le\theta^*\}| \\
&= \frac{1}{m_k}\sum_{\ell=1}^{q}\frac{\mu_{\ell,k}}{\sum_{k'}\mu_{\ell,k'}}\cdot|\{j\in V'_\ell : t^*_j\le\theta^*\}| \\
&\overset{\text{Claim I}}{\le} \frac{4}{m_k}\sum_{\ell:k\in\pi(V'_\ell)}\sum_{j\in V'_\ell: t^*_j\le\theta^*} z_{j,k} \\
&\overset{\text{Claim II}}{\le} \frac{8}{m_k}\sum_{\ell:k\in\pi(V'_\ell)}\sum_{j\in V'_\ell: t^*_j\le\theta^*}\sum_{i\in[m_k]}\sum_{t\in I_{*,k}: t\le\theta^*} x_{j,i,t} \\
&\le \frac{8}{m_k}\sum_{i\in[m_k]}\sum_{t\in I_{*,k}: t\le\theta^*}\underbrace{\sum_{j\in J'_0} x_{j,i,t}}_{\le 1 \text{ by } (\mathsf{LP})} \le 8\cdot|\{t\in I_{*,k} : t\le\theta^*\}| \le 8s_k\cdot\theta^*
\end{aligned}$$

Here we also have used that $\sum_{k'\in[M]}\mu_{\ell,k'}\ge\frac{1}{2}$. As X is sum of independent random variables X_ℓ each of which is bounded by $O(s_k\cdot\theta^*)$, we apply the Chernoff bound from Lemma 7 and obtain that $\Pr[X > C'\cdot\frac{\log n}{\log\log n}\cdot s_k\theta^*]\le 1/n^{1000}$ for some constant $C'>0$. To complete the lemma, we simply do a union bound over all possible values of θ^* and i. The number of machines is $m\le n$ and the number of relevant θ^*'s in the time horizon is bounded by $\sum_{k\in[M]}|I_{*,k}|\le M\cdot 2n$. □

Lemma 16. *For every job $j\in J'_0$, with high probability, the completion time of j in our schedule $C^A_j\le O(M\cdot\frac{\log n}{\log\log n}\cdot C_j)$.*

Proof. Fix a job $j^*\in J'_0$, and suppose job j^* is scheduled on machine i of type k in our schedule. From Lemma 15, there are at most $O(\frac{\log n}{\log\log n}\cdot s_k\cdot t^*_{j^*})$ jobs whose $t^*_j\le t^*_{j^*}$. Therefore, the completion time is $C^A_{j^*}\le O(\frac{\log n}{\log\log n}\cdot t^*_{j^*})$. Note that in the LP solution at least an $\frac{1}{100M}$-fraction of j^* was scheduled at or after $t^*_{j^*}$. Hence, $C_j\ge\frac{1}{100M}\cdot t^*_{j^*}$. Putting things together we conclude $C^A_j\le O(\frac{\log n}{\log\log n}\cdot t^*_{j^*})\le O(M\cdot\frac{\log n}{\log\log n}\cdot C_j)$. □

Now we focus on bounding the completion time of jobs in J''_0.

Lemma 17. *For every job $j\in J''_0$, the completion time of j in our schedule $C^A_j\le O(\log n\cdot c)$.*

Proof. From the definition of set J_0, for all $j\in J''_0$ the completion time $C_j\le c\cdot\delta$ in the LP solution. This implies that at least a 1/2-fraction of every job $j\in J''_0$ is scheduled in the interval $[0,2c\cdot\delta]$ in the LP solution. We take the LP solution restricted to J''_0 in the interval $[0,c\cdot\delta]$, and invoke Theorem 13 to construct a schedule of jobs J''_0. Theorem 13 guarantees that every job $j\in J''_0$ is scheduled in an interval of length at most $2c\cdot\delta\cdot(\log m\cdot\log n+M^2)+O(\log m\cdot c)=O(\log m\cdot c)$ from our choice of δ. To complete the proof, it remains to account for the increase in completion time caused by jobs in J'_0 as our algorithm schedules J''_0 after scheduling jobs in J'_0. However, from Lemma 16 every job in J'_0 has completion time at most $O(\log n\cdot c)$. Thus the lemma follows. □

Lemma 18. *For any job $j_1\in J_0$, the probability that $j_1\in J''_0$ is at most $O(M\cdot\log n\cdot\frac{C_{j_1}}{c})$.*

Proof. Consider the set $U:=\{j_1\}\cup(\Gamma^-(j_1)\cap J_0)$ of j_1 and its ancestors. If $j_0\prec j_1$, then $0\le C_{j_0}+c\cdot d(j_0,j_1)\le C_{j_1}$ by the LP constraints and so $d(j_0,j_1)\le\frac{C_{j_1}}{c}$. Then the diameter of U with

respect to semimetric d is bounded by $2C_{j_1}/c$ and hence by Theorem 6.(b) the probability that U is separated is bounded by $\ln(2n) \cdot \frac{4\text{diam}(U)}{\Delta} \leq O(M \cdot \log n \cdot \frac{C_{j_1}}{c})$. □

We can now bound the expected completion time of every job in J_0.

Lemma 19. *For every job $j \in J_0, \mathbb{E}[C_j^A] \leq O(M \cdot \log^2 n) \cdot C_j$.*

Proof. Fix a job $j_1 \in J_0$ and consider

$$\begin{aligned}
\mathbb{E}[C_{j_1}^A] &= \mathbb{E}[C_{j_1}^A | (j_1 \in J_0')] \cdot \Pr[(j_1 \in J_0')] + \mathbb{E}[C_{j_1}^A | (j_1 \in J_0'')] \cdot \Pr[(j_1 \in J_0'')] \\
&\overset{(*)}{\leq} O\Big(M \cdot \frac{\log n}{\log\log n} \cdot C_{j_1}\Big) + O\Big(M \cdot \log n \cdot \frac{C_{j_1}}{c}\Big) \cdot O(\log n \cdot c) \\
&\leq O(M \cdot \log^2 n) \cdot C_{j_1},
\end{aligned}$$

where (*) follows from Lemmas 16, 17, 18. □

We also note the following simple consequence of previous lemmas:

Lemma 20. *For every job $j \in J_0$, $C_j \leq O(\log n \cdot c)$. In other words, the makespan of our schedule for J_0 is at most $O(\log n \cdot c)$.*

4.2 Intermediate Batch Scheduling

Now we describe our algorithm to schedule jobs in the set J_γ for $\gamma > 0$, which is similar to that of scheduling jobs in J_0''. Recall that from the definition of set J_γ in Eq. (2), $2^{\gamma-1}\delta \cdot c < C_j \leq 2^\gamma \delta \cdot c$ in the LP solution. We take the LP solution restricted to J_γ in the interval $[0, 2^\gamma \delta \cdot c]$, and invoke Theorem 13 to construct a schedule of jobs J_γ. Theorem 13 ensures that every job $j \in J_\gamma$ is scheduled in an interval of length at most $O(2^\gamma \delta \cdot c \cdot \log m \cdot \log n) + O(\log n \cdot c)$. As a consequence, we obtain the following lemma.

Lemma 21. *Fix any $\gamma^* \geq 0$. For every $j \in J_{\gamma^*}$, $C_j^A \leq O(2^{\gamma^*} \cdot \delta \cdot \log m \cdot \log n) + O(\gamma^* \cdot \log n \cdot c)$.*

Proof. For $\gamma = 0, 1, \ldots$, let $\mathcal{S}(J_\gamma)$ denote the schedule of jobs in the set J_γ. Our final schedule $\mathcal{S}$ is simply a concatenation of schedules $\mathcal{S}(J_0), \mathcal{S}(J_1), \ldots$, while inserting c empty time slots between $\mathcal{S}(J_\gamma)$ and $\mathcal{S}(J_{\gamma+1})$. Let T_γ denote the makespan of $\mathcal{S}(J_\gamma)$. From Lemma 20 and Theorem 13, $T_\gamma \leq O(2^\gamma \cdot \delta \cdot c \cdot \log m \cdot \log n) + O(\log n \cdot c)$. Fix the job $j^* \in J_{\gamma^*}$ with the highest completion time in $\mathcal{S}$. We can bound the completion time of j^* as

$$\begin{aligned}
C_{j^*}^A &\leq \sum_{\gamma=0}^{\gamma^*} T_\gamma \leq O(\gamma^* \cdot \log n \cdot c) + \sum_{\gamma=1}^{\gamma^*} O(2^\gamma \cdot \delta \cdot c \cdot \log m \cdot \log n) \\
&\leq O(\gamma^* \cdot \log n \cdot c) + O(2^{\gamma^*} \cdot \delta \cdot c \cdot \log m \cdot \log n)
\end{aligned}$$

which completes the proof. □

We finish the proof of main theorem for the weighted sum of completion times of jobs.

Proof of Theorem 12. For every job $j \in J_0$, from Lemma 19, we have $\mathbb{E}[C_j^A] \leq O(M \cdot \log^2 n) \cdot C_j$. From the previous Lemma 21, for $\gamma > 0$ and $j \in J_\gamma$ we have, $C_j^A \leq O(\gamma \cdot \log n \cdot c) + O(2^\gamma \cdot \delta \cdot c \cdot \log m \cdot \log n)$. On the other hand, for every $j \in J_\gamma$, by definition, $C_j > 2^\gamma \cdot \delta \cdot c$. Thus, $C_j^A \leq O(M \cdot \log^2 n) \cdot C_j$.

□

5 Proof of Theorem 13

Recall that we are given a subset of jobs $J' \subseteq J$, and a solution (x, y, C) to the LP with $r \geq 5$ and $T \in \mathbb{N}$. It is promised that in (x, y, C), the completion times satisfy $C_j \leq T$ for every job $j \in J'$. Then there is a randomized polynomial time algorithm that schedules all jobs such that $C_j^A \leq O(\log m \cdot \log n \cdot T) + O(\log m \cdot c)$ for every job $j \in J'$.

5.1 Main Scheduling Subroutine

The following lemma is the main scheduling subroutine towards proving Theorem 13.

Lemma 22. *Let $C^* \geq 0$ and $\delta = 1/64 \log(2n)$. Consider the set $J^* \subseteq \{j \in J' \mid C^* \leq C_j \leq C^* + \delta \cdot c\}$. Then there is a randomized rounding procedure that finds a subset $J^{**} \subseteq J^*$, a partition of J^{**} into $J^{**} = \cup_{k=1}^{M} J^{**(k)}$ where $J^{**(k)}$ are jobs assigned to machines of type k, and a schedule for J^{**} in an interval of length at most $5c + \sum_k \frac{|J^{**(k)}|}{s_k \cdot m_k}$ such that every job $j \in J^*$ is scheduled with probability at least $\frac{1}{2}$.*

The rounding algorithm to prove the above lemma is the following:

Scheduling a Single Batch on Related Machines

(1) Run a CKR clustering on the semimetric space (J^*, d) with parameter $\Delta := \frac{1}{4}$ and let $V_1, \ldots, V_q$ be the clusters.
(2) Let $V'_\ell := \{j \in V_\ell \mid \Gamma^-(j) \cap J^* \subseteq V_\ell\}$ for $\ell = 1, \ldots, q$.
(3) For all $\ell = 1, \ldots, q$, assign the jobs in V'_ℓ to type k^* satisfying the conditions in Lemma 10.
(4) For all $\ell = 1, \ldots, q$, if V'_ℓ is assigned to type k^*, assign all jobs in V'_ℓ to the machine of type k^* which has the least load (breaking ties arbitrarily).

We prove few lemmas that will be helpful in proving the desired properties of the algorithm. The first lemma shows that the clusters produced by the algorithm are not too large.

Lemma 23. *For all $\ell = 1, \ldots, q$, one has $|V'_\ell| \leq O(s_{k^*} c)$. Here, k^* is the machine type that V'_ℓ is assigned in our algorithm. Thus, it takes $O(c)$ time to process all jobs in V'_ℓ.*

Proof. We know by Theorem 6 that $\operatorname{diam}(V'_\ell) \leq \operatorname{diam}(V_\ell) \leq \Delta \leq \frac{1}{4}$. The lemma follows by Lemma 10 and by our choice of k^*. □

Further it is easy to see that the clusters respect precedence constraints.

Lemma 24. *The solution $V'_1, \ldots, V'_q$ is feasible in the sense that jobs on different machines do not have precedence constraints.*

Proof. Consider jobs processed on different machines, say (after reindexing) $j_1 \in V'_1$ and $j_2 \in V'_2$. If $j_1 \prec j_2$ then we did *not* have $\Gamma^-(j_2) \subseteq V'_2$. This contradicts the definition of the sets V'_ℓ. □

We will use the two statements above together with Theorem 6 to prove Lemma 22.

Proof of Lemma 22. Call the schedule produced by the above algorithm $\mathcal{S}(J^*)$. By Lemma 23, for all $\ell = 1, \ldots, q$, the processing time of V'_ℓ is at most $O(c)$. By Lemma 24, there are no dependent jobs in different sets of $V'_1, \ldots, V'_q$.

The interval in which the jobs in J^* are scheduled can be divided into c-length time intervals $I_1, \cdots, I_\ell$. To prove the lemma, it suffices to consider the case where $\ell \geq 6$. We say that a machine of type k is *full* in a c-length interval if it is processing $c \cdot s_k$ jobs in $\mathcal{S}(J^*)$. From our greedy packing and the bound on processing time of V'_ℓ, it follows that there exists a machine type $k \in [M]$ such that every machine $i \in [m_k]$ is full in $I_1, \cdots I_{\ell-5}$. Therefore,

$$c \cdot (\ell - 5) \leq \max_k \frac{|J^{**(k)}|}{s_k \cdot m_k} \leq \sum_k \frac{|J^{**(k)}|}{s_k \cdot m_k}.$$

Thus, the total length of the interval used in a single batch scheduling is bounded by

$$c \cdot \ell \leq 5c + \sum_k \frac{|J^{**(k)}|}{s_k \cdot m_k}.$$

It remains to prove that a fixed job $j^* \in J^*$ is scheduled with good probability. Consider the set $U := \{j^*\} \cup (\Gamma^-(j^*) \cap J^*)$ of j^* and its ancestors in J^*. By the LP constraint $C_{j_1} + d(j_1, j_2) \cdot c \leq C_{j_2}$, rearranging we see that $d(j_1, j_2) \leq \delta$ for every $j_1, j_2 \in U$. So the diameter of U is at most 2δ. Now we appeal to Lemma 10. For our choice of $\Delta = 1/4$ and $\delta \leq \frac{1}{64 \log(2n)}$, we apply Theorem 6 to see that the cluster is separated with probability at most $\log(2n) \cdot \frac{8\delta}{\Delta} \leq \frac{1}{2}$. □

To schedule all jobs in J^*, we repeat the clustering procedure $O(\log m)$ times and simply schedule the remaining jobs on the fastest machine.

Lemma 25. *Let $C^* \geq 0$ and $\delta = 1/64 \log(2n)$. Consider the set $J^* \subseteq \{j \in J' \mid C^* \leq C_j \leq C^* + \delta \cdot c\}$. Then there is an algorithm with expected polynomial running time that finds disjoint subsets $J^{*(k)} \subset J^*$, where $J^{*(k)}$ are jobs assigned to machines of type k, and schedules all jobs in J^* using at most $O(\log m \cdot c) + \frac{|J^*|}{m \cdot s_{\max}} + \sum_k \frac{|J^{*(k)}|}{s_k \cdot m_k}$ many time slots.*

Proof. We run the above algorithm for $2 \log m$ iterations. Input to iteration $h+1$ is the set of jobs that are not scheduled in the first h iterations. For $h \in \{1, 2, \ldots, 2 \log m\}$, let J^{**}_h be the set of jobs scheduled in iteration h, and let $J^*_{h+1} := J^* \setminus \{\bigcup_{i=1}^h J^{**}_i\}$. In this notation, $J^*_1 := J^*$.

Let $\mathcal{S}(J^{**}_h)$ be the schedule of jobs J^{**}_h obtained from by Lemma 22. We schedule $\mathcal{S}(J^{**}_1)$ first, then append schedule $\mathcal{S}(J^{**}_h)$ after $\mathcal{S}(J^{**}_{h-1})$ while inserting c empty time slots between them, for $h = 2, \ldots, 2 \log m$. Let $\hat{J} := J^*_{2 \log m + 1}$ be the set of jobs that were not scheduled in these $2 \log m$ iterations. We schedule all jobs in $\hat{J}$ consecutively on a single machine with the *fastest speed* after the completion of $\mathcal{S}(J^{**}_{2 \log m})$.

From our construction, the length of a schedule for J^*, which is a random variable, is at most $O(\log m \cdot c) + \lceil \frac{|\hat{J}|}{s_{\max}} + \sum_k \frac{|J^{*(k)}|}{s_k \cdot m_k} \rceil$. For $h \in \{1, 2, \ldots, 2 \log m\}$, Lemma 22 guarantees that each job $j \in J^*_h$ gets scheduled in the h^{th} iteration with probability at least $1/2$. Therefore, the probability that $j \in \hat{J}$, i.e. it does not get scheduled in the first $2 \log m$ iterations, is at most $\frac{1}{2m}$. This implies that $\mathbb{E}[|\hat{J}|] \leq \frac{|J^*|}{2m}$. The claimed expected polynomial running time bound now simply follows from appealing to Markov's inequality.

Finally, by our construction precedence and communication delay constraints are satisfied. □

5.2 The Complete Algorithm

To schedule all jobs in J' we do the following.

THE COMPLETE ALGORITHM

(1) Let (x, y, C) be a solution to (LP) with $r \geq 5$ for J'.
(2) For $\delta = \frac{1}{64 \log(2n)}$ and $h \in \{0, 1, 2 \ldots \frac{T-1}{c \cdot \delta}\}$, define $J_h = \{j \in J : h \cdot \delta \cdot c \leq C_j < (h+1) \cdot \delta \cdot c\}$
(3) FOR $h = 0$ TO $\frac{T-1}{c \cdot \delta}$ DO
    (4) Schedule the jobs in J_h using the algorithm in Subsection 5.1.
    (5) Insert c new empty idle slots.

Proof of Theorem 13. First consider the case when $T > \delta \cdot c$. The total number of time slots used by our algorithm can be upper bounded by

$$\frac{T}{c \cdot \delta} \cdot O(\log m \cdot c) + \sum_{h=0}^{\frac{T-1}{c \cdot \delta}} \frac{|J_h|}{m \cdot s_{\max}} + \sum_{h,k} \frac{|J_h^{(k)}|}{s_k \cdot m_k} = O(\log m \cdot \log n) \cdot T + \frac{|J'|}{m \cdot s_{\max}} + \sum_k \frac{|J^{(k)}|}{s_k \cdot m_k}.$$

Here, $J_h^{(k)}$ is the set of jobs assigned on machines of type k when scheduling J_h, introduced in lemma 25, and $J^{(k)} = \cup_{h=0}^{\frac{T-1}{c \cdot \delta}} J_h^{(k)}$. From lemma 10 and the choice of machine type in the algorithm, $\sum_{j \in J^{(k)}} \sum_t \sum_{i \in [m_k]} x_{j,i,t} \geq \Omega(1/M) \cdot |J^{(k)}|$. On the other hand, by the constraints of the LP, we have

$$\sum_{j \in J^{(k)}} \sum_{t=0}^{T} \sum_{i \in [m_k]} x_{j,i,t} \leq \sum_{j \in J} \sum_t \sum_{i \in [m_k]} x_{j,i,t} \leq m_k \cdot s_k \cdot T.$$

Thus, $\frac{|J^{(k)}|}{s_k \cdot m_k} \leq O(M \cdot T)$, and $\sum_k \frac{|J^{(k)}|}{s_k \cdot m_k} \leq O(M^2 \cdot T)$. It is also implied from the same constraints of the LP that $|J'| \leq T \cdot m \cdot s_{\max}$. As $M^2 \leq O(\log m \cdot \log n)$, the proof follows assuming $T > \delta \cdot c$.

Now if $T \leq \delta \cdot c$, then $J' = J^*$ and our algorithm does only one iteration. Thus from Lemma 25 the total number of time slots used by our algorithm is at most

$$O(\log m \cdot c) + \frac{|J'|}{m \cdot s_{\max}} + \sum_k \frac{|J'^{(k)}|}{s_k \cdot m_k} \leq O(\log m \cdot c) + O(M^2 \cdot T).$$

This completes the proof. □

Finally, we conclude this section by noting that above proof in fact gives an $O(\log^2 n)$ approximation for the makespan objective on related machines for the unit length case.

6 A Reduction from $\mathsf{Q} \mid \mathsf{prec}, c \mid \sum w_j C_j$ to $\mathsf{Q} \mid \mathsf{prec}, p_j = 1, c \mid \sum w_j C_j$

An extension of the classic *list scheduling* algorithm of Graham [Gra66] is the *speed based list scheduling* algorithm by Chudak and Shmoys [CS99]. In this setting, one has a set of jobs J with precedence constraints and an assignment that determines on what machine type a job will be executed. Then we process the jobs greedily in the sense that any available job is scheduled as early as possible on one of the machines belonging to its assigned speed class. The makespan can then be upper bounded by the maximum chain length plus the sum of the loads (if summed over all speed classes).

As in previous sections, we will consider a set J of jobs with processing time p_j and m machines, partitioned into machine types $(m_k, s_k)_{k \in [M]}$, meaning that there are m_k machines of speed s_k available. We design a slight extension of speed based list scheduling that allows us to take communication delays into account.

SPEED-BASED LIST SCHEDULING (WITH COMMUNICATION DELAYS)

Input: Jobs J with $p_j \geq 0$, machine types $(m_k, s_k)_{k\in[M]}$; assignment $\pi : J \to [M]$; communication delay c

Output: Feasible non-preemptive schedule

(1) Set $\sigma(j) := \emptyset$ for all $j \in J$

(2) FOR $t = 0$ TO ∞ DO FOR $i = 1$ TO m DO

(3) IF i is idle at time t THEN select any job $j \in J$ satisfying the following

- $\sigma(j) = \emptyset$
- Every $j' \prec j$ has been completed. Moreover if $j' \prec j$ was scheduled on machine $i' \neq i$, then j' must have finished by time $t - c$
- Machine i is of class $\pi(j)$

(4) Set $\sigma(j) := ([t, t + \frac{p_j}{s_{\pi(j)}}), i)$ (if there was such a job)

We provide an analysis which follows closely Chudak and Shmoys [CS99] as well as Graham [Gra66].

Lemma 26. *Suppose we are given jobs J with processing time p_j and a partial order $\prec$ as well as machine types $(m_k, s_k)_{k\in[M]}$ and an assignment $\pi : J \to [M]$. Denote $D_k := \sum_{j\in J:\pi(j)=k} \frac{p_j}{m_k s_k}$ as the* load *on type k. Then the* SPEED-BASED LIST SCHEDULING *algorithm produces a schedule of length*

$$\sum_{k=1}^{M} D_k + \max\left\{\left(\sum_{j\in C} \frac{p_j}{s_{\pi(j)}}\right) + (|C| - 1) \cdot c \mid C \subseteq J \text{ is a chain}\right\} \tag{4}$$

Proof. Consider the schedule produced by SPEED-BASED LIST SCHEDULING. Let j_1 denote the last job that finishes. For $\ell \geq 1$ suppose we have already constructed the sequence $j_1, \ldots, j_\ell$. Then we choose $j_{\ell+1}$ as that job with $j_{\ell+1} \prec j_\ell$ which is finished last in the schedule. We terminate the procedure when we reach a chain $j_q \prec \ldots \prec j_2 \prec j_1$ where j_q does not have any predecessors. Let $i_{j_\ell} \in [m]$ be the machine index where j_ℓ is scheduled and let $[t_{j_\ell}, t_{j_\ell} + \frac{p_{j_\ell}}{s_{\pi(j_\ell)}})$ be the time interval in which j_ℓ is scheduled. We can make the observation that by the time $t_{j_{\ell+1}} + c$, all predecessors of j_ℓ have been completed and moreover a time of c has passed. Hence if the period between $t_{j_{\ell+1}} + c$ and t_{j_ℓ} is non-empty, then all machines in class $\pi(j_\ell)$ had to be busy during that period. We can make the more general conclusion that for the constructed chain $C := \{j_q, \ldots, j_1\}$ it is true that for every time unit in $[0, t_{j_1} + \frac{p_{j_1}}{s_{\pi(j_1)}})$ one of the following holds:

(a) a job from C is processed at time t
(b) a job from C has finished less than c time units ago
(c) in at least one class k, all machines are busy at t

We note that the duration of time that can fall into one of the categories $(a), (b), (c)$ is indeed upper bounded by (4). The claim is then proven. □

This can be combined with an assignment argument:

Lemma 27. *Let J be a set of jobs with processing times p_j, a partial order $\prec$ and machine types $(m_k, s_k)_{k\in[M]}$. Suppose there is a pre-emptive migratory schedule with makespan T^*. Then there is an assignment $\pi : J \to [M]$ such that each load $D_k := \sum_{j\in J:\pi(j)=k} \frac{p_j}{m_k s_k}$ satisfies $D_k \leq 4T^*$ and the maximum chain length is $\Delta \leq 2T^*$.*

Proof. After scaling the time horizon we may assume that the job lengths p_j are multiples of 2 and the individual parts in the schedule have unit length. Let T^* be the makespan of the schedule. Let $x_{jk} := \frac{\#\text{job parts of } j \text{ assigned to class } k}{p_j}$ we see that we have a fractional solution to the assignment LP

$$\begin{aligned}
\sum_{k=1}^{M} x_{jk} &= 1 \quad \forall j \in J \qquad \text{(Assignment-LP-I)} \\
\sum_{j \in J} x_{jk} \frac{p_j}{s_k m_k} &\leq T^* \quad \forall k \in [M] \\
\sum_{j \in C} \sum_{k=1}^{M} x_{jk} \frac{p_j}{s_k} &\leq T^* \quad \forall \text{ chain } C \subseteq J \\
0 \leq x_{jk} &\leq 1 \quad \forall j \in J \, \forall k \in [M]
\end{aligned}$$

We now perform a standard procedure in approximation algorithms that is usually called *filtering*. Consider a job j and delete the $1/2$ of its parts that are on the slowest machines and scale the fractional solution by a factor of 2 on the remaining parts. Then we obtain a fractional solution satisfying

$$\begin{aligned}
\sum_{k=1}^{M} y_{jk} &= 1 \quad \forall j \in J \qquad \text{(Assignment-LP-II)} \\
\sum_{j \in J} y_{jk} \frac{p_j}{s_k m_k} &\leq 2T^* \quad \forall k \in [M] \\
0 \leq y_{jk} &\leq 1 \quad \forall j \in J \, \forall k \in [M]
\end{aligned}$$

Moreover due to the filtering, any assignment $\pi : J \to [M]$ with $y_{j,\pi(j)} > 0$ for all $j \in J$, satisfies the following two properties:

(A) One has $\frac{p_j}{s_{\pi(j)}} \leq 2T^*$.

(B) The maximum chain length w.r.t. π is $\Delta \leq 2T^*$.

If we imagine $q_{jk} := \frac{p_j}{s_k m_k}$ as item sizes, then the seminal rounding argument of Shmoys-Tardos shows that any fractional solution to (Assignment-LP-II) can be rounded to an integral solution where the right hand side is exceeded by at most $\max\{q_{jk} : y_{jk} > 0\} \leq 2T^*$. The obtained integral solution is then the desired assignment π. □

We use this for the main reduction:

Theorem 28. *Suppose there is an α-approximation to $\mathsf{Q} \mid \mathsf{prec}, p_j = 1, c \mid \sum w_j C_j$. Then there is an $O(M\alpha)$-approximation for $\mathsf{Q} \mid \mathsf{prec}, c \mid \sum w_j C_j$.*

Proof. Let J be the set of jobs with processing times $p_j \in \mathbb{N}$ and weights w_j with original precedence constraints $\prec$. Let OPT be the minimum weighted sum of completion times over all feasible schedules. We run the α-approximation algorithm for $\mathsf{Q} \mid \mathsf{prec}, p_j = 1, c \mid \sum w_j C_j$ treating each job as p_j many unit length *tasks* that form a chain where the last unit length task receives a weight of w_j and the other tasks receive weight 0. The algorithm will provide us with a schedule σ which satisfies communication delays but it spreads tasks of the same job over different machines (and even different machine types). Let C_j be the completion time of the last task of job j. Then $\sum_{j \in J} w_j C_j \leq \alpha OPT$ by construction.

We abbreviate $J^*(0) := \{j \in J \mid C_j \leq c\}$ and $J^*(r) := \{j \in J \mid c2^{r-1} < C_j \leq c2^r\}$ for $r \in \mathbb{Z}_{\geq 1}$. We will create a schedule that schedules first $J^*(0)$, then $J^*(1), J^*(2), \ldots$ where we start the schedule for $J^*(r)$ exactly c time units after the last job of $J^*(r-1)$ is finished. Note that we can re-sort the tasks of jobs in $J^*(0)$ so that (i) the tasks of the same job are scheduled consecutively; (ii) for every job j the last tasks is still completed by time C_j, (iii) the precedence constraints and communication constraints (within $J^*(0)$) are still satisfied. Note that this re-sorting can be done by sorting the jobs in non-increasing order of C_j's.

So we fix a value of $r \geq 1$ and set $T^* := c2^{r-1}$ and $J^* := J^*(r)$. It remains to show how to schedule the jobs J^* within a time interval of $O(T^*)$.

We partition the time horizon into intervals of length c each. Consider the jobs J^* and let σ^* be the schedule with makespan T^* which satisfies precedence constraints and communication delays but is potentially preemptive and migratory. We construct a modified set $\tilde{J}$ by merging jobs whose tasks are scheduled on the same machine in one such interval $[\beta c, (\beta+1)c[$ with $\beta \in \mathbb{Z}_{\geq 0}$ (we call these jobs *compact*); all other jobs are inherited without a change (we call such jobs *spread out*).

We denote the inherited migratory schedule for $\tilde{J}$ by $\tilde{\sigma}$; the completion time of a job is denoted by $\tilde{C}_j$ and the start time is denoted by $\tilde{S}_j$. Note that $\tilde{C}_j - \tilde{S}_j \geq p_j$ but because of the preemption this inequality might be strict. We create a new partial order $\tilde{\prec}$ such that one has $j_1 \tilde{\prec} j_2$ for $j_1, j_2 \in \tilde{J}$ if one of the following conditions is satisfied:

1. j_1, j_2 are assigned to the same machine in $\tilde{\sigma}$ and $\tilde{C}_{j_1} \leq \tilde{S}_{j_2}$.
2. j_1, j_2 are assigned to different machines in $\tilde{\sigma}$ and j_1 finishes in an interval before j_2 is started.

Note that in particular this precedence order $\tilde{\prec}$ implies $\prec$. The maximum chain length is trivially bounded by T^*. But it is important to note that the maximum *number* of jobs in any chain of $\tilde{\prec}$ is bounded by $O(T^*/c)$ as every job is either spread out and hence crosses an interval boundary or it is the unique compact job included in an interval (on that machine). Then Lemma 27 provides us with an assignment $\pi : \tilde{J} \to [M]$ such that the loads are $D_k \leq 4T^*$ and the maximum chain length is $\Delta \leq 2T^*$. We use the Speed-based List Scheduling algorithm to find a non-preemptive schedule respecting precedence constraints $\tilde{\prec}$ and communication delays while the the makespan is bounded by

$$\sum_{k=1}^{M} D_k + \sum_{j \in C} \frac{p_j}{s_{\pi(j)}} + (|C|-1) \cdot c \leq 4MT^* + 2T^* + O\Big(\frac{T^*}{c}\Big) \cdot c \leq O(MT^*)$$

for some chain $C \subseteq \tilde{J}$. □

7 Makespan Minimization on Related Machines

We now prove our result on makespan minimization on related machines, Theorem 2. The proof essentially follows from first using Theorem 13 to obtain a schedule of cost $O(\log^2 n)$ for the unit length case and then appealing to the reduction in Theorem 28, which increases the makespan at most by a factor of $O(\log n)$. These two steps together give an $O(\log^3 n)$-approximation for $\mathsf{Q} \mid \mathsf{prec}, c \mid C_{\max}$. Note a small technicality here. Both Theorem 13 and Theorem 28 were proved for the weighted completion time objective and not the makespan objective function. However, it is not hard to see that proofs of both the theorems directly extend to the makespan problem.

For technical completeness, we give a formal proof of our result for makespan. Our proof also serves the purpose of explaining how an algorithm for the sum of weighted completion times can

be used to obtain an algorithm for makespan. In our proof below, we first reduce the problem of minimizing makespan to a special case of the problem of minimizing the weighted sum of completion times. But we *do not* use the result on weighted completion time as a black box – that would only imply an $O(\log^4 n)$-approximation. Instead, we tweak our rounding a bit, and use Theorem 13 and Theorem 28 directly to obtain an $O(\log^3 n)$-approximation for makespan.

Proof. Consider an input instance $\mathcal{I} := (J, \prec)$ of $\mathsf{Q} \mid \mathsf{prec}, c \mid C_{\max}$, and let $T(*)$ denote the optimal makespan for any instance $(*)$ of the problem. We consider two cases:

- Case $T(\mathcal{I}) < c$: In this case, we find all connected components, $G_1, G_2, \ldots G_q$, in the underlying (undirected) graph of the input instance. Since $T(\mathcal{I}) < c$, for $v \in [q]$ all jobs in G_v are scheduled on the same machine by an optimal solution. For every $v \in [q]$, we create a meta-job j_v with processing time equal to the sum of processing times of jobs in G_v. Then we appeal to the $O(1)$-approximation algorithm for makespan minimization on related machines without precedence or communication delay constraints.

- Case $T(\mathcal{I}) \geq c$. Here we reduce $\mathcal{I}$ to an instance of $\mathcal{I}' := (J', \prec')$ of $\mathsf{Q} \mid \mathsf{prec}, c \mid \sum w_j C_j$. We create a new dummy job j_D with $p_{j_D} = 1$ and weight $w_{j_D} = 1$. We define $J' := J \cup \{j_D\}$ and set weights $w_j = 0$ for every job $j \in J' \setminus \{j_D\}$. Moreover, we define the new precedence constraints $\prec'$ by augmenting the precedence constraints of the instance $\mathcal{I}$ with precedence constraints $j \prec j_D$ from every job $j \in J' \setminus \{j_D\}$.

  Suppose $\mathcal{S}$ is a feasible schedule for the instance $\mathcal{I}'$. From our construction, the makespan of $\mathcal{S}$ is (exactly) equal to the total weighted completion times of jobs in $\mathcal{I}'$. This follows because only the dummy job j_D has non-zero weight and it has to be scheduled at the last.

  We now give an algorithm for solving the instance $\mathcal{I}'$. Similar to our weighted completion time result, we solve the problem via a two step procedure. We treat each job j with processing time p_j as a chain of p_j unit length jobs. Using standard arguments [DKR+20], we can assume that this reduction is both of polynomial size and time. After this reduction, we get a new input instance $\mathcal{I}'' := (J'', \prec')$ of $\mathsf{Q} \mid \mathsf{prec}, p_j = 1, c \mid \sum w_j C_j$.

  We observe that $T(\mathcal{I}'') \in [T(\mathcal{I}), 2T(\mathcal{I})]$. This follows because i) the weight of all jobs except j_D is zero in our instance; ii) j_D can be scheduled only after completing all the other jobs, which requires at least $T(\mathcal{I})$ time steps; iii) $c \leq T(\mathcal{I})$.

  We solve our LP for the weighted completion time problem on the instance $\mathcal{I}''$. Let (x, y, C) be the optimal solution for the LP. We note that all jobs in the LP solution are scheduled in the interval $[0, 2T(\mathcal{I})]$. Here we used the facts that $c \leq T(\mathcal{I})$ and that the LP can indeed schedule all jobs in $\mathcal{I}''$ within an interval of length $c + T(\mathcal{I})$. We invoke Theorem 13 to obtain a schedule $\mathcal{S}$ with cost at most $O(\log m \cdot \log n) \cdot 2T(\mathcal{I})$. Finally we use Theorem 28 to convert this schedule $\mathcal{S}$ to a schedule $\mathcal{S}'$ where every job is scheduled on one machine in consecutive time steps. From the guarantees of Theorem 28, we know the cost of $\mathcal{S}'$ is at most $O(M \cdot \log m \cdot \log n) \cdot T(\mathcal{I})$.

  As noted earlier, the makespan of the schedule $\mathcal{S}'$ is (exactly) equal to the total weighted completion time of jobs in $\mathcal{I}'$. Thus

$$\text{Makespan of } (\mathcal{S}') \leq O(M \cdot \log m \cdot \log n) \cdot T(\mathcal{I})$$

  which completes the proof.

□

References

[Ban17] N. Bansal. Scheduling open problems: Old and new. MAPSP 2017. http://www.mapsp2017.ma.tum.de/MAPSP2017-Bansal.pdf, 2017.

[CKR04] G. Călinescu, H. Karloff, and Y. Rabani. Approximation algorithms for the 0-extension problem. *SIAM J. Comput.*, 34(2):358–372, 2004.

[CS99] Fabián A. Chudak and David B. Shmoys. Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds. *J. Algorithms*, 30(2):323–343, 1999.

[CZM+11] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica. Managing data transfers in computer clusters with orchestra. In *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM11, page 98109. Association for Computing Machinery, 2011.

[DKR+20] Sami Davies, Janardhan Kulkarni, Thomas Rothvoss, Jakub Tarnawski, and Yihao Zhang. Scheduling with communication delays via LP hierarchies and clustering. *To appear at 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS 2020), 16-19 November 2006, Durham, North Carolina, USA, Proceedings*, 2020.

[DP09] Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[FKK+14] Z. Friggstad, J. Könemann, Y. Kun-Ko, A. Louis, M. Shadravan, and M. Tulsiani. Linear programming hierarchies suffice for directed steiner tree. In *Integer Programming and Combinatorial Optimization - 17th International Conference, IPCO 2014, Bonn, Germany, June 23-25, 2014. Proceedings*, pages 285–296, 2014.

[FRT04] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, 2004.

[GCL18] Yuanxiang Gao, Li Chen, and Baochun Li. Spotlight: Optimizing device placement for training deep neural networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1676–1684, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[GFC+12] Zhenyu Guo, Xuepeng Fan, Rishan Chen, Jiaxing Zhang, Hucheng Zhou, Sean McDirmid, Chang Liu, Wei Lin, Jingren Zhou, and Lidong Zhou. Spotting code optimizations in data-parallel pipelines through periscope. In *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, pages 121–133, 2012.

[GKMP08] R. Giroudeau, J.C. König, F. K. Moulai, and J. Palaysi. Complexity and approximation for precedence constrained scheduling problems with large communication delays. *Theoretical Computer Science*, 401(1):107 – 119, 2008.

[GLLK79] 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.

[Gra66] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45(9):1563–1581, 1966.

[GVY93] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM Journal on Computing*, 25:698–707, 1993.

[HCG12] Chi-Yao Hong, Matthew Caesar, and P Brighten Godfrey. Finishing flows quickly with preemptive scheduling. *ACM SIGCOMM Computer Communication Review*, 42(4):127–138, 2012.

[HLV94] J.A. Hoogeveen, J.K. Lenstra, and B. Veltman. Three, four, five, six, or the complexity of scheduling with communication delays. *Operations Research Letters*, 16(3):129–137, 1994.

[HM01] C. Hanen and A. Munier. An approximation algorithm for scheduling dependent tasks on m processors with small communication delays. *Discrete Applied Mathematics*, 108(3):239 – 257, 2001.

[HSSW97] Leslie A Hall, Andreas S Schulz, David B Shmoys, and Joel Wein. Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Mathematics of operations research*, 22(3):513–544, 1997.

[JZA19] Zhihao Jia, Matei Zaharia, and Alex Aiken. Beyond data and model parallelism for deep neural networks. In *Proceedings of the 2nd SysML Conference, SysML '19*, Palo Alto, CA, USA, 2019.

[Lau03] M. Laurent. A comparison of the sherali-adams, lovász-schrijver, and lasserre relaxations for 0-1 programming. *Math. Oper. Res.*, 28(3):470–496, 2003.

[Li17] Shi Li. Scheduling to minimize total weighted completion time via time-indexed linear programming relaxations. *SIAM Journal on Computing*, (0):FOCS17–409, 2017.

[LR99] Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787832, November 1999.

[LYZ+16] Shouxi Luo, Hongfang Yu, Yangming Zhao, Sheng Wang, Shui Yu, and Lemin Li. Towards practical and near-optimal coflow scheduling for data center networks. *IEEE Transactions on Parallel and Distributed Systems*, 27(11):3366–3380, 2016.

[MK97] A. Munier and J.C. König. A heuristic for a scheduling problem with communication delays. *Operations Research*, 45(1):145–147, 1997.

[MN06] M. Mendel and A. Naor. Ramsey partitions and proximity data structures. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 109–118, 2006.

[MPL+17] Azalia Mirhoseini, Hieu Pham, Quoc V Le, Benoit Steiner, Rasmus Larsen, Yuefeng Zhou, Naveen Kumar, Mohammad Norouzi, Samy Bengio, and Jeff Dean. Device placement optimization with reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2430–2439. JMLR. org, 2017.

[MRS+20] Biswaroop Maiti, Rajmohan Rajaraman, David Stalfa, Zoya Svitkina, and Aravindan Vijayaraghavan. Scheduling precedence-constrained jobs on related machines with communication delay. *To appear at 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS 2020)*, 2020.

[NHP+19] D. Narayanan, A. Harlap, A. Phanishayee, V. Seshadri, N. Devanur, G. Ganger, P. Gibbons, and M. Zaharia. Pipedream: Generalized pipeline parallelism for dnn training. In *Proc. 27th ACM Symposium on Operating Systems Principles (SOSP)*, Huntsville, ON, Canada, October 2019.

[PY90] C. H. Papadimitriou and M. Yannakakis. Towards an architecture-independent analysis of parallel algorithms. *SIAM J. Comput.*, 19(2):322328, April 1990.

[QS02] Maurice Queyranne and Maxim Sviridenko. Approximation algorithms for shop scheduling problems with minsum objective. *Journal of Scheduling*, 5(4):287–305, 2002.

[RS87] V.J. Rayward-Smith. Uet scheduling with unit interprocessor communication delays. *Discrete Applied Mathematics*, 18(1):55 – 71, 1987.

[SRVW20] Yu Su, Xiaoqi Ren, Shai Vardi, and Adam Wierman. Communication-aware scheduling of precedence-constrained tasks on related machines. *CoRR*, abs/2004.14639, 2020.

[SW99] P. Schuurman and G. J. Woeginger. Polynomial time approximation algorithms for machine scheduling: Ten open problems, 1999.

[SZA+18] Ayan Shymyrbay, Arshyn Zhanbolatov, Assilkhan Amankhan, Adilya Bakambekova, and Ikechi A Ukaegbu. Meeting deadlines in datacenter networks: An analysis on deadline-aware transport layer protocols. In *2018 International Conference on Computing and Network Communications (CoCoNet)*, pages 152–158. IEEE, 2018.

[TPD+20] Jakub Tarnawski, Amar Phanishayee, Nikhil R. Devanur, Divya Mahajan, and Fanny Nina Paravecino. Efficient algorithms for device placement of dnn graph operators, 2020.

[TY92] R. Thurimella and Y. Yesha. A scheduling principle for precedence graphs with communication delay. *International Conference on Parallel Processing*, 3:229–236, 1992.

[VLL90] B. Veltman, B.J. Lageweg, and J.K. Lenstra. Multiprocessor scheduling with communication delays. *Parallel Computing*, 16(2):173 – 182, 1990.

[ZCB+15] Yangming Zhao, Kai Chen, Wei Bai, Minlan Yu, Chen Tian, Yanhui Geng, Yiming Zhang, Dan Li, and Sheng Wang. Rapier: Integrating routing and scheduling for coflow-aware data center networks. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 424–432. IEEE, 2015.

[ZZC+12] Jiaxing Zhang, Hucheng Zhou, Rishan Chen, Xuepeng Fan, Zhenyu Guo, Haoxiang Lin, Jack Y Li, Wei Lin, Jingren Zhou, and Lidong Zhou. Optimizing data shuffling in data-parallel computation by understanding user-defined functions. In *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pages 295–308, 2012.

A Missing proofs on LP properties

We now give the postponed proof of Lemma 8.

Lemma. *Let (x, y, C) be a solution to* (LP) *with $r \geq 5$. Then $d(j_1, j_2) := 1 - y_{j_1,j_2}$ is a semimetric.*

Proof. The first two properties from the definition of a semimetric are clearly satisfied. We verify the triangle inequality. Consider three jobs $j_1, j_2, j_3 \in J$. We set $\tilde{J} := \{j_1, j_2, j_3\}$ and consider the distribution $(\tilde{x}, \tilde{y}) \sim \mathcal{D}(\tilde{J})$. For $j \in \tilde{J}$, define $Z(j) = (\tilde{s}(j), \tilde{i}(s))$ as the random variable that gives the unique pair of indices such that $\tilde{x}_{j,\tilde{i}(j),\tilde{s}(j)} = 1$. Then for $j', j'' \in \tilde{J}$ one has

$$d(j', j'') = \Pr[Z(j') \neq Z(j'')] = \Pr\left[\left(\tilde{s}(j), \tilde{i}(j')\right) \neq \left(\tilde{s}(j''), \tilde{i}(j'')\right)\right]$$

Then indeed

$$\begin{aligned} d(j_1, j_3) &= \Pr[Z(j_1) \neq Z(j_3)] \leq \Pr[Z(j_1) \neq Z(j_2) \vee Z(j_2) \neq Z(j_3)] \\ &\overset{\text{union bound}}{\leq} \Pr[Z(j_1) \neq Z(j_2)] + \Pr[Z(j_2) \neq Z(j_3)] = d(j_1, j_2) + d(j_2, j_3). \end{aligned}$$

□