# Approximation Algorithms for Scheduling and Fair Allocations

Yihao Zhang

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2022

Reading Committee:
Thomas Rothvoss, Chair
Dmitriy Drusvyatskiy
Isabella Novik

Program Authorized to Offer Degree:
Mathematics

University of Washington

**Abstract**

Approximation Algorithms for Scheduling and Fair Allocations

Yihao Zhang

Chair of the Supervisory Committee:
Thomas Rothvoss
Department of Mathematics

In this thesis, we will have discussions on two main topics, max-min allocation and scheduling jobs with precedent constraints on machines with communication delays. New approximation algorithms are given in Chapter 2, 4 and 5, where linear programming plays a fairly important role on algorithm designs, while Chapter 3 contains partial results on the general max-min allocation.

The Santa Claus problem is also known as the restricted max-min fair allocation. In this problem, Santa Claus has a set of gifts, and he wants to distribute them among a set of children so that the least happy child is made as happy as possible. Here, the value that a child $i$ has for a present $j$ is of the form $p_{ij} \in \{0, p_j\}$. Based on a modification of Haxell's hypergraph matching argument, a polynomial time algorithm by Annamalai et al. gives a 12.33-approximation.

In joint work with Sami Davies and Thomas Rothvoss, a *matroid* version of the Santa Claus problem is introduced. The algorithm is based on Haxell's augmenting tree, but with the introduction of the matroid structure. Our result can then be used as a blackbox to obtain a $(4 + \varepsilon)$-approximation for Santa Claus, comparing against a natural, compact LP. A recent work of Cheng and Mao [CM19] also gets the factor $(4 + \varepsilon)$.

On the second half, we first consider the classic problem of scheduling jobs with precedence constraints on identical machines to minimize makespan, in the presence of *communication*

*delays*. In this setting, denoted by $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$, if two dependent jobs are scheduled on different machines, then at least $c$ units of time must pass between their executions. Despite its relevance to many applications, the best known approximation ratio was $O(c)$, whereas Graham's greedy list scheduling algorithm already gives a $(c+1)$-approximation in that setting. An outstanding open problem in the top-10 list by Schuurman and Woeginger and its recent update by Bansal asks whether there exists a constant-factor approximation algorithm.

In joint work with Sami Davies, Janardhan Kulkarni, Thomas Rothvoss and Jakub Tarnawski, we give a polynomial-time $O(\log c \cdot \log m)$-approximation algorithm for this problem, where $m$ is the number of machines and $c$ is the communication delay. Our approach is based on a Sherali-Adams lift of a linear programming relaxation and a randomized clustering of the semimetric space induced by this lift.

Finally, a more general version of this problem is considered in Chapter 5, to minimize the weighted sum of completion times on related machines, denoted by $\mathsf{Q} \mid \mathsf{prec}, c \mid \sum w_j C_j$. Our main result is an $O(\log^4 n)$-approximation algorithm for the problem. As a byproduct 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].

# Contents

# Acknowledgments

First, I wish to express my sincere appreciation to my advisor Thomas. He is not only an excellent advisor, being kind, supportive and knowledgeable, but also a role model of a researcher, with his diligence, enthusiam and brilliance. His enormous collection of ideas stored as tex files truly illustrates how great work is produced. Moreover, his humility and sense of humor make it even enjoyable for my career.

I would like to extend my appreciations to my co-authors, Sami Davies, Janardhan Kulkarni and Jakub Tarnawski, from whom I have learned a lot.

I am grateful to Dmitriy Drusvyatskiy, Isabella Novik and Archis Gate for serving on my supervisory committee. I also want to thank Dima and Isabella for their guidance in my early career.

Furthermore, I wish to express my thankfulness to the math department of UW, its faculty and staff, for offering such a precious opportunity to me of gaining knowledge and conducting research. I would also like to thank my friends and colleagues for their company.

Finally, I want to express my gratitude to my family, for their support, trust and endless love.

# Dedication

to my dear wife, Fuchen Liu

# Chapter 1

# Introduction and Preliminaries

## *1.1 Introduction*

Since many combinatorial optimization problems can not be effectively solved to find an exact solution, researchers tend to seek for a substitutional approximate one that can be computed effectively. Meanwhile, algorithm designs and analysis is a core part of computer science. Then, it becomes an interesting field in theoretical computer science to design and analyze approximation algorithms for discrete optimization problems, applying mathematical tools like linear programming and probability.

To be formal, the class **P** denotes the problems that can be solved in polynomial complexity of time, while a large family of problems, denoted by **NP**-hard, are believed to be unsolvable in polynomial time. Then, it would be necessary to find approximation algorithms solved in polynomial time for those **NP**-hard problems. Famous **NP**-hard problems include problems like traveling salesman problem (TSP), knapsack problem and minimum makespan scheduling. A formal definition of an approximation algorithm and its factor will be available in Section 1.3.1.

When an **NP**-hard discrete optimization problem can be formulated as an integer programming, a standard way of designing an approximation algorithm is to use the linear programming relaxtion, by removing integer constraints. In such a way, the key part is to

design an LP relaxation and an algorithm to construct a feasible solution to the integer programming from a feasible solution to the LP relaxatioin. Details about the scheme will be illustrated in Section 1.3.2. On the other hand, linear programming can be solved efficiently, within polynomial time of its size. Even though some constructed LPs have an exponential number of constraints, it may still be possible to solve those in polynomial time with respect to the size of the original problem, by employing certain techniques such as the ellipsoid method.

In this thesis, we apply LP relaxations to design approximation algorithms to four problems in two main topics: max-min fair allocation and scheduling jobs with precedent constraints on machines with communication delays. As the reader will see, these problems are all natural to be asked, having nice formulation. However, they are **NP**-hard, which means that approximation algorithms are indeed required to solve them.

### *1.2 Our Contributions*

Chapter 2 is based on a joint work with Sami Davies and Thomas Rothvoss, which was published in SODA 2020 [DRZ20]. In that paper, we study the Santa Claus problem, also known as restricted max-min fair allocation problem. In this problem, Santa Claus has a set of gifts to distribute among a set of children, so that the least happy child is made as happy as possible. Here, the value that a child $i$ has for a present $j$ is of the form $p_{ij} \in \{0, p_j\}$.

By introducing a matroid version of the Santa Claus problem, we design an algorithm based on Haxell's augmenting tree method. Finally, we obtain a $(4+\varepsilon)$-approximation for Santa Claus, which is currently the best known ratio. Also, note that the configuration LP is usually used to solve this problem, which has an exponential size, while our LP is a compact one.

Chapter 3 consists of a joint work with Sami Davies, Janardhan Kulkarni and Thomas Rothvoss, which is currently in progress. In this chapter, we work on the general max-min fair allocation problem, where $p_{ij}$, the value that a child $i$ has for a present $j$, is a nonnegative real number, no longer restricted in $\{0, p_j\}$. We propose a candidate algorithm, which may

provide a better factor in the future than an $O(\log^{10}(n))$-approximation by [CCK09].

Chapter 4 and 5 are based on joint works with Sami Davies, Janardhan Kulkarni, Thomas Rothvoss and Jakub Tarnawski, which were published in FOCS 2020 [DKR+20] and SODA 2021 [DKR+21] respectively. These papers both work on scheduling jobs with precedent constraints on machines with communication delays. The former one is about a simpler case, where machines are identical and the objective is to minimize the makespan, while the latter one deals with a general case, to minimize the weighted sum of completion time on related machines. Here, related machines means that they have different efficiency, i.e. the processing time of job $j$ on machine $i$ can be represented as $p_j/s_i$. Since for the weighted completed time version, a dummy job can be created to have all other jobs as predecessors and full weight in objective, makespan minimization is indeed a special case of it.

Our main result in Chapter 4 gives a polynomial-time $O(\log c \cdot \log m)$-approximation algorithm, where $m$ is the number of machines and $c$ is the communication delay, while only an $O(c)$-approximation is known before. It is based on a Sherali-Adams lift of a linear programming relaxation and a randomized clustering of the semimetric space induced by this lift. For the case of related machines in Chapter 5, we obtain an $O(\log^4 n)$-approximation for minimizing a weighted sum of completion time and an $O(\log^3 n)$-approximation for makespan minimization, by a generalization of the methods in Chapter 4. Both of the factors are currently the best.

## *1.3 Preliminaries*

### *1.3.1 Approximation Algorithms*

As mentioned above, approximation algorithms are efficient algorithms that find approximate solutions to optimization problems in polynomial time, especially for **NP**-hard problems. The factor of approximation algorithms is defined as follows.

**Definition 1.** A **$\rho$-approximation** ($\rho \geq 1$) algorithm for an (**NP**-hard) optimization problem is an algorithm, such that for any valid instance $x$, the objective value $f(x)$ of the

approximate solution that the algorithm returns would satisfy

$$\mathrm{OPT}(x) \leq f(x) \leq \rho \cdot \mathrm{OPT}(x)$$

for minimization problems and

$$\rho^{-1} \cdot \mathrm{OPT}(x) \leq f(x) \leq \mathrm{OPT}(x)$$

for maximization problems, where $\mathrm{OPT}(x)$ is the optimal value of the objective function for instance $x$.

For convenience, we will shorten $\mathrm{OPT}(x)$ as OPT when there is no ambiguity. Sometimes, it may also be denoted as $\frac{1}{\rho}$-approximation for maximization problems in some references.

Another definition worth mentioning is polynomial time approximation schemes (PTAS).

**Definition 2.** A **PTAS** for an optimization problem is an algorithm, such that $\forall$ parameter $\epsilon > 0$, the algorithm produces a solution that is within a factor $1 + \epsilon$ of being optimal, in time of polynomial complexity (in the problem size).

Here, the time complexity only need to be polynomial in the problem size for every fixed $\epsilon$, and it can be different for different $\epsilon$. For example, running in $O(n^{\frac{1}{\epsilon}})$ or even $O(n^{\exp(\frac{1}{\epsilon})})$ still counts as PTAS. A similar definition can be derived for a $(c + \epsilon)$-approximation, where $c > 1$ is a constant.

### *1.3.2 LP Relaxation and Integrality Gap*

The LP relaxation of an integer linear program is the LP that arises by removing the integrality constraint of each variable. It is mentioned in Section 1.1 that LP relaxations is a standard method to design approximation algorithms.

Now we consider a minimization problem formulated by an integer programming with an instance. Actually, it would be similar to deal with a maximization problem, so we just consider minimization problems in this section for convenience. Let OPT and $\mathrm{OPT}_{frac}$ be the optimal value of the integer programming and the LP relaxation respectively, then it is

clear that $\text{OPT}_{frac} \leq \text{OPT}$. Suppose that an algorithm can be designed to derive a feasible solution for the original integer programming from the optimal solution of the LP relaxation, with a guarantee that the objective value of the rounded solution $\leq \rho \cdot \text{OPT}_{frac}$, then this objective value $\leq \rho \cdot \text{OPT}$, meaning that a $\rho$-approximation is obtained.

This shows a typical way of applying an LP relaxation. One related concept worth mentioning is the integrality gap between the original integer programming and the LP relaxation.

**Definition 3.** In an instance $x$ of a minimization problem, if the real minimum (the minimum of the integer problem) is $\text{OPT}(x)$, and the relaxed minimum (the minimum of the LP relaxation) is $\text{OPT}_{frac}(x)$, then the **integrality gap** of the LP relaxation is $\sup_x \frac{\text{OPT}(x)}{\text{OPT}_{frac}(x)}$, which is the maximal of this ratio over all the instances.

Actually, the integrality gap is important from two perspectives. Firstly, it provides a view of how good an approximation factor one can obtain based on this LP relaxation. Suppose that the integrality gap is greater than $\rho$, meaning that $\exists$ instance $x_0$, s.t. $\text{OPT}(x_0) > \rho \cdot \text{OPT}_{frac}(x_0)$. Then, it is impossible to get a rounded integer solution for this instance with objective value $\leq \rho \cdot \text{OPT}_{frac}(x_0)$, otherwise it is less than $\text{OPT}(x_0)$, which contradicts the fact that $\text{OPT}(x_0)$ is the minimum for integer solutions. Thus, by the standard procedure mentioned above, it could be impossible to get an approximation factor better than the integrality gap. On the other hand, knowing the integrality gap could provide a bound to the optimal value of the integer programming by solving the linear programming, i.e. giving an approximation without returning a solution.

Thus, LP designing is indeed crucial to get a better approximation factor. Sometimes an LP that researchers can naturally come up with is too weak, which could even have an infinity integrality gap. To overcome the weakness, surely one way is to find a new LP, like changing the definition of variables. One example is the configuation LP in the max-min fair allocation problem, for which references can be found in Section 2.1.1.

A canonical way of strengthening the existing LP is to apply the lift to an LP relaxation,

which provides an automatic strengthening. In this thesis, the Sherali-Adams lift is employed through Chapter 3 to 5, and a detialed introduction is available in Section 4.2.1. The basic idea of LP lifting is that, without any rounds of lift, the original LP relaxation of the integer programming could be too weak. While by lifting it to the final round, the feasible domain becomes the convex integer hull of the orignal integer programming, but it is hard to solve. For a trade-off, a constant rounds of lift is typically applied, which provides the strengthening of the LP relaxation, while still being solvable in polynomial time.

# Chapter 2

# The Santa Claus Problem

## *2.1 Introduction*

### *2.1.1 The Santa Claus Problem*

Formally, the *Santa Claus* problem takes as input a set $M$ of children, a set $J$ of gifts, and values $p_{ij} \in \{0, p_j\}$ for all $i \in M$ and $j \in J$. In other words, a child is only interested in a particular subset of the gifts, but then its value only depends on the gift itself. The goal is to find an assignment $\sigma : J \to M$ of gifts to children so that $\min_{i \in M} \sum_{j \in \sigma^{-1}(i)} p_{ij}$ is maximized. Actually, it is **NP**-hard to compute a solution better than a 2-approximation [BD05].

The first major progress on this problem is due to Bansal and Sviridenko [BS06], who showed an $O(\log \log n / \log \log \log n)$-approximation based on rounding a *configuration LP*. Bansal and Sviridenko also realized that in order to obtain an $O(1)$-approximation, it suffices to solve a purely combinatorial problem: show that in a uniform bipartite hypergraph with equal degrees on all sides, there is a left-perfect matching that selects a constant fraction of nodes from the original edges. This problem was solved by Feige [Fei08] who proved a large unspecified constant using the Lovász Local Lemma repeatedly, and then Haeupler, Saha and Srinivasan [HSS11] made it constructive, giving the first constant-factor approximation. Also, Asadpour, Feige and Saberi [AFS08] showed that one can also solve the problem posed by Bansal and Sviridenko non-constructively, meaning with a potentially exponential time

argument, by using a beautiful theorem on hypergraph matchings due to Haxell [Hax95]; their bound[1] of 4 has been slightly improved to 3.84 by Jansen and Rohwedder [JR18c] and then to 3.808 by Cheng and Mao [CM19]. Recently, Jansen and Rohwedder [JR18a] also showed (still non-constructively) that it suffices to compare to a linear program with as few as $O(n^3)$ many variables and constraints, in contrast to the exponential size configuration LP.

A *hypergraph* $\mathcal{H} = (X \dot\cup W, \mathcal{E})$ is called *bipartite* if $|e \cap X| = 1$ for all hyperedges $e \in \mathcal{E}$. A *(left-) perfect matching* is a set of hyperedges $F \subseteq \mathcal{E}$ that are disjoint but cover each node in $X$. In general, finding perfect matchings in even bipartite hypergraphs is **NP**-hard, but there is an intriguing sufficient condition:

**Theorem 1** (Haxell [Hax95])**.** *Let $\mathcal{H} = (X \dot\cup W, \mathcal{E})$ be a bipartite hypergraph with $|e| \leq r$ for all $e \in \mathcal{E}$. Then either $\mathcal{H}$ contains a left-perfect matching or there is a subset $C \subseteq X$ and a subset $U \subseteq W$ so that all hyperedges incident to $C$ intersect $U$ and $|U| \leq (2r-3) \cdot (|C|-1)$.*

It is instructive to consider a "standard" bipartite graph with $r = 2$. In this case, if there is no perfect matching, then there is a set $C \subseteq X$ with at most $|C|-1$ many neighbors — so Haxell's condition generalizes *Hall's Theorem*. Unlike Hall's Theorem, Haxell's proof is non-constructive and based on a possibly exponential time augmentation argument. Only very recently and with a lot of care, Annamalai [Ann16] managed to make the argument polynomial. This was accomplished by introducing some slack into the condition and assuming the parameter $r$ is a constant. Preceding [Ann16], Annamalai, Kalaitzis and Svensson [AKS15] gave a non-trivially modified version of Haxell's argument for Santa Claus, which runs in polynomial time and gives a 12.33-approximation[2]. Recently, Cheng and Mao altered their algorithm to improve the approximation to $6 + \varepsilon$, for any constant $\varepsilon > 0$ [CM18]. Our algorithm will also borrow a lot from [AKS15]. However, through a much cleaner argument

[1]The conference version of [AFS08] provides a factor of 5, which in the journal version [AFS12] has been improved to 4.

[2]To be precise they obtain a $(6 + 2\sqrt{10} + \varepsilon)$-approximation in time $n^{O(\frac{1}{\varepsilon^2}\log(\frac{1}{\varepsilon}))}$.

we obtain a result that works in a more general matroid setting and implies a better approximation of $4 + \varepsilon$ for Santa Claus. An independent work of Cheng and Mao [CM19] also get the same factor by improving their previous work.

### *2.1.2 General Max-Min Fair Allocation*

It should not go without mention that the version of the Santa Claus problem with arbitrary $p_{ij}$ has also been studied before under the name *Max-Min Fair Allocation*, which we will further discuss in the next chapter.

The Max-Min Allocation problem was first studied as a machine scheduling problem where the minimum completion time is maximized. Woeginger [Woe97] and Epstein and Sgall [ES99] gave polynomial time approximation schemes (PTAS) for the cases when all agents (machines) have identical utilities for the items. Woeginger [Woe00] also gave an FPTAS for the case when the number of agents, $m$, is a constant. The first non-trivial approximation algorithm for the general MaxMin Allocation problem is due to Bezakova and Dani [BD05] who gave an $(n - m + 1)$-approximation algorithm. They also showed the problem is **NP**-hard to approximate up to any factor smaller than 2.

Interestingly, the integrality gap of the configuration LP is at least $\Omega(\sqrt{n})$ [BS06]. Then, Asadpour and Saberi [AS10] gave an $O(\sqrt{m}\log^3 m)$ approximation for the problem using the same LP relaxation. Meanwhile, Chakrabarty, Chuzhoy and Khanna [CCK09] found an (rather complicated) $O(\log^{10}(n))$-approximation algorithm in $n^{O(\log n)}$ time[3].

### *2.1.3 Minimum Makespan Scheduling*

The general Max-Min Allocation problem has a very well studied "dual" minmax problem. Usually it is phrased as *Makespan Scheduling* with *machines* $i \in M$ and *jobs* $j \in J$. Then we have a running time $p_{ij}$ of job $j$ on machine $i$, and the goal is to assign jobs to machines so that the maximum load of any machine is minimized. More precisely, it is called "Scheduling

[3]The factor is $n^{\varepsilon}$ if only polynomial time is allowed, where $\varepsilon > 0$ is arbitrary but fixed.

on Unrelated Parallel Machines", where "unrelated" means that there is no relation between the processing times of a job on the different mahcines. Also, the Santa Claus problem may be viewed as a "dual" problem of restricted minimum makespan scheduling, where $p_{ij} \in \{p_j, \infty\}$.

In the general setting, the seminal algorithm of Lenstra, Shmoys and Tardos [LST87] gives a 2-approximation — with no further improvement since then. In fact, a $(\frac{3}{2} - \varepsilon)$-approximation is **NP**-hard [LST87], and the configuration LP has an integrality gap of 2 [VW11]. In the restricted assignment setting, the breakthrough of Svensson [Sve11] provides a non-constructive 1.942-bound on the integrality gap of the configuration LP using a custom-tailored Haxell-type search method. Recently, this was improved by Jansen and Rohwedder [JR17] to 1.834. In an even more restricted variant called *Graph Balancing*, each job is admissable on exactly 2 machines. In this setting Ebenlendr, Krcál and Sgall [EKS08] gave a 1.75-approximation based on an LP-rounding approach, which has again been improved by Jansen and Rohwedder [JR18b] to 1.749 using a local search argument.

A more special case of the makespan scheduling is to schedule on identical machines, i.e. $p_{ij} = p_j$. Graham [Gra66a] showed that a greedy algorithm would get a 2-approximation. Few decades later, Hochbaum and Shmoys [HS87a] obtained a polynomial-time approximation scheme (PTAS), which is the best possible result for **NP**-hard problems.

The problems we solved in Chapter 4 and 5 are specific ones in the family of problems on minimum makespan scheduling, with precedence contraints on jobs and communication delays among machines.

### *2.2 Our contributions*

From now on, we focus on the Santa Claus problem in this chapter. Let $\mathcal{M} = (X, \mathcal{I})$ be a *matroid* with *groundset* $X$ and a family of *independent sets* $\mathcal{I} \subseteq 2^X$. Recall that a matroid is characterized by three properties:

(i) *Non-emptyness*: $\emptyset \in \mathcal{I}$;

(ii) *Monotonicity*: For $Y \in \mathcal{I}$ and $Z \subseteq Y$ one has $Z \in \mathcal{I}$;

(iii) *Exchange property*: For all $Y, Z \in \mathcal{I}$ with $|Y| < |Z|$ there is an element $z \in Z \setminus Y$ so that $Y \cup \{z\} \in \mathcal{I}$.

The *bases* $\mathcal{B}(\mathcal{M})$ of the matroid are all inclusion-wise maximal independent sets. The cardinalities of all bases are identical, with size denoted as $\operatorname{rank}(\mathcal{M})$. The convex hull of all bases is called the *base polytope*, that is $P_{\mathcal{B}(\mathcal{M})} := \operatorname{conv}\{\chi(S) \in \{0,1\}^X \mid S \text{ is basis}\}$, where $\chi(S)$ is the *characteristic vector* of $S$.

Now consider a bipartite graph $G = (X \dot\cup W, E)$ with the ground set $X$ on one side and a set of *resources* $W$ on the other side; each resource $w \in W$ has a *size* $p_w \geq 0$. In a problem that we call *Matroid Max-Min Allocation*, the goal is to find a basis $S \in \mathcal{B}(\mathcal{M})$ and an assignment $\sigma : W \to S$ with $(\sigma(w), w) \in E$ so that $\min_{i \in S} \sum_{w \in \sigma^{-1}(i)} p_w$ is maximized. To the best of our knowledge, this problem has not been studied before. In particular if $T \geq 0$ is the target objective function value, then we can define a linear programming relaxation $Q(T)$ as the set of vectors $(x, y) \in \mathbb{R}^X_{\geq 0} \times \mathbb{R}^E_{\geq 0}$ satisfying the constraints

$$x \in P_{\mathcal{B}(\mathcal{M})}; \quad \sum_{w \in N(i)} p_w y_{iw} \geq T \cdot x_i \; \forall i \in X; \quad y(\delta(w)) \leq 1 \; \forall w \in W; \quad y_{iw} \leq x_i \; \forall (i,w) \in E.$$

Here, the decision variable $x_i$ expresses whether element $i$ should be part of the basis, and $y_{iw}$ expresses whether resource $w$ should be assigned to element $i$. We abbreviate $N(i)$ as the neighborhood of $i$ and $y(\delta(w))$ is shorthand for $\sum_{i:(i,w) \in E} y_{iw}$. Then our main technical result is:

**Theorem 2.** *Suppose $Q(T) \neq \emptyset$. Then for any $\varepsilon > 0$ one can find*

$$(x, y) \in Q\left(\left(\frac{1}{3} - \varepsilon\right) \cdot T - \frac{1}{3} \cdot \max_{w \in W} p_w\right)$$

*with both $x$ and $y$ integral in time $n^{\Theta_\varepsilon(1)}$, where $n := |X| + |W|$. This assumes that membership in the matroid can be tested in time polynomial in $n$.*

Previously this result was not even known with non-constructive methods. We see that Matroid Max-Min Allocation is a useful framework by applying it to the Santa Claus problem:

**Theorem 3.** *The Santa Claus problem admits a $(4+\varepsilon)$-approximation algorithm in time $n^{\Theta_\varepsilon(1)}$.*

For a suitable threshold $0 < \delta < 1$, call a gift $j$ *small* if $p_j \leq \delta \cdot \mathrm{OPT}$ and *large* otherwise. Then the family of sets of children that can get assigned large gifts forms a *matchable set matroid*. We apply Theorem 2 to the *co-matroid* of the matchable set matroid. Then we obtain a basis $S := \{i \in M \mid x_i = 1\}$, which contains the children *not* receiving a large gift. These children can receive small gifts of total value $(\frac{1}{3} - \frac{\delta}{3} - \varepsilon) \cdot \mathrm{OPT}$. The remaining children receive a large gift with value at least $\delta \cdot \mathrm{OPT}$. Setting $\delta := \frac{1}{4}$ implies the claim. Note the approximation factor $4 + \varepsilon$ will be with respect to a natural, compact linear program with $O(n^2)$ many variables and constraints. The smallest LP that was previously known to have a constant integrality gap was the $O(n^3)$-size LP of [JR18a].

### *2.3 An algorithm for Matroid Max-Min Allocation*

In this section we provide an algorithm that proves Theorem 2.

#### *2.3.1 Intuition for the algorithm*

We provide some insight by starting with an informal overview of our algorithm. Let $G = (X \cup W, E)$ be the bipartite graph defined in Section 2.2. If $U \subset W$ and $i \in X$ with $(i, j) \in E$ for all $j \in U$, we can consider the pair $(i, U)$ to be a hyperedge. Then for $0 < \nu < 1$ and $\mathrm{val}(\cdot)$ the function summing the value in a hyperedge's resources, we say that $(i, U)$ is a *$\nu$-edge* if it a hyperedge with minimal (inclusion wise) resources such that $\mathrm{val}(U) := \sum_{w \in U} p_w \geq \nu T$. By $\mathcal{E}_{\nu T}$ we denote the set of $\nu$-edges.

Fix constants $0 < \beta < \alpha < 1$ and $0 < \delta < 1$, to be chosen later. The goal of the algorithm is to find a basis $S \in \mathcal{B}(\mathcal{M})$ and a hypergraph matching $M \subseteq \mathcal{E}_{\beta T}$ covering $S$. The algorithm is initialized with $S := \{i_0\}$, for any node $i_0 \in X$, and $M := \emptyset$. We perform $\mathrm{rank}(\mathcal{M})$ many phases, where in each phase we find a larger matching, and the set it covers in $X$ is independent with respect to the matroid. In an intermediate phase, we begin with

$S \in \mathcal{I}$ and $M \subseteq \mathcal{E}_{\beta T}$ a hypergraph matching covering $S \setminus \{i_0\}$ with one exposed node $i_0 \in X$. At the end of a phase, the algorithm produces an updated matching covering an independent set $S'$, with $|S'| = |S|$. For $|S'| < \text{rank}(\mathcal{M})$, there exists $i'_0 \in X \setminus S'$ such that $S' \cup \{i'_0\} \in \mathcal{I}$. Repeating this $\text{rank}(\mathcal{M})$ times, we end with a basis which is well-covered by $\beta$-edges.

The algorithm generalizes the notion of an augmenting path used to find a maximum matchings in bipartite graphs to an *augmenting tree*. Though instead of swapping every other edge in an augmenting path, as is the case for a bipartite graph, the algorithm swaps sets of edges in the augmenting tree to find more space in the hypergraph. During a phase, the edges are swapped in such a way that the underlying set in $X$ covered by the matching is always independent with respect to the matroid. The edges which are candidates for being swapped into the matching are called *adding edges* and denoted by $A$, while those which are candidates for being swapped out of the matching are called *blocking edges* and denoted by $B$. It is helpful to discuss the nodes covered by adding and blocking edges in each part, and so for hyperedges $H \subseteq \mathcal{E}_{\nu T}$ we define $H_X$ and $H_W$ as the nodes covered by $H$ in $X$ and $W$, respectively. The algorithm gives some slack by allowing the adding edges to be slightly larger than the blocking edges.

The parameters $\alpha$ and $\beta$ determine the value of the adding and blocking edges, respectively, so the adding edges are a subset of $\mathcal{E}_{\alpha T}$ while the blocking edges are a subset of $\mathcal{E}_{\beta T}$. Set $\delta := \max_w p_w/T$, so that all elements in the basis receive resources with value at most $\delta T$. The following observations follow from minimality of the hyperedges:

1. A $\nu$-edge has value less than $(\nu + \delta)T$. This implies that an add edge has value less than $(\alpha + \delta)T$ and a blocking edge has value less than $(\beta + \delta)T$.

2. Every blocking edge has value at most $\beta \cdot T$ not covered by an add edge.

To build the augmenting tree, the algorithm starts from the node in $S$ uncovered by $M$, $i_0$, and chooses an edge $e \in \mathcal{E}_{\alpha T}$ covering $i_0$ which is added to $A$. If there is a large enough hyperedge $e' \in \mathcal{E}_{\beta T}$ such that $e' \subset e$ and $e'$ is disjoint from $M$, then there is enough available

resources that we simply update $M$ by adding $e'$ to it. Otherwise, $e$ does not contain a set of resources with total value $\beta T$ free from $M$. The edges of $M$ intersecting $e$ are added to the set of blocking edges, $B$. Nodes in $C = \{i_0\} \cup B_X$ are called *discovered* nodes, as they are the nodes covered by the hypermatching $M$ which appear in the augmenting tree.

Continuing to build the augmenting tree in later iterations, the algorithm uses an *Expansion Lemma* to find a large set of disjoint hyperedges, $H \subset \mathcal{E}_{\alpha T}$, that cover a subset which can be swapped into $S$ in place of some subset of $C$ while maintaining independence in the matroid. The set of hyperedges $H$ either $(i)$ intersects many edges of $M$ or $(ii)$ has a constant fraction of edges which contain a hyperedge from $\mathcal{E}_{\beta T}$ that is disjoint from $M$.

In the first case, a subset of $H$ which intersects $M$, denoted $A_{\ell+1}$, is added to $A$, and the edges of $M$ intersecting $A_{\ell+1}$, denoted $B_{\ell+1}$, are added to $B$, for $\ell$ the index of the iteration. Note we naturally obtain *layers* which partition the adding and blocking edges in our augmenting tree. The layers for the adding and blocking edges respectively are denoted as $A_\ell$ and $B_\ell$, with

$$A_{\leq \ell} := \bigcup_{i=0}^{\ell} A_i \qquad \text{and} \qquad B_{\leq \ell} := \bigcup_{i=0}^{\ell} B_i.$$

The layer indices are tracked because they are useful in proving the algorithm's runtime. In the second case, for the set of edges $H' \subset \mathcal{E}_{\alpha T}$ that have a hyperedge from $\mathcal{E}_{\beta T}$ disjoint from $M$, the algorithm finds a layer which has a large number of discovered nodes that can be swapped out for a subset of nodes which $H'$ covers.

### *2.3.2 A detailed procedure*

Recall, we fixed $\delta = \max_{w \in W} p_w/T$. Then, we set $\beta = \frac{1}{3} - \frac{\delta}{3} - \varepsilon$ and $\alpha = \frac{1}{3} - \frac{\delta}{3} - \frac{\varepsilon}{2}$, for $0 < \varepsilon < (1-\delta)/3$. Here lies the subtle but crucial difference to previous work. In [AKS15] the authors have to use adding edges that are a large constant factor bigger than blocking edges. In our setup we can allow adding edges that are only marginally larger than the blocking edges. This results in an improved approximation factor of $4 + \varepsilon$ for Santa Claus compared to the 12.33 factor by [AKS15].

The algorithm is described in Figure 2.1. For later reference, the constant from Lemma 7 is $\frac{1-2\alpha-\beta-\delta}{1+\delta} = \frac{2\varepsilon}{1+\delta} \geq \varepsilon$, and the constant from Lemma 8 is $\frac{\alpha-\beta}{\delta+\alpha} = \frac{\varepsilon}{2(1+\delta)} \geq \varepsilon/4$. We use Lemma 9, with constant $c = \frac{1-2\alpha-\beta-\delta}{1+\delta} \cdot \frac{\alpha-\beta}{\delta+\alpha}$. Our bounds for constants do not use a specific choice of $\delta$, and instead they only use the fact that $0 < \delta < 1$. Both cases in the algorithm are visualized in Figure 2.2 and Figure 2.3.

### *2.3.3 Correctness of the algorithm*

Here, we prove several lemmas used in the algorithm which implies Theorem 2. We begin by building up to our *Expansion Lemma*, Lemma 7. Our algorithm takes a fixed independent set, $S$, and swaps $C \subset S$ out of $S$ for a set of nodes $D$ in order to construct a new independent set of the same size. This is possible by Lemma 7.

Recall a variant of the so-called *Exchange Lemma*. For independent sets $Y, Z \in \mathcal{I}$, let $H_{\mathcal{M}}(Y, Z)$ denote the bipartite graph on parts $Y$ and $Z$ (if $Y \cap Z \neq \emptyset$, then have one copy of the intersection on the left and one on the right). For $i \in Y \setminus Z$ and $j \in Z \setminus Y$ we insert an edge $(i, j)$ in $H_{\mathcal{M}}(Y, Z)$ if $Y \setminus \{i\} \cup \{j\} \in \mathcal{I}$. Otherwise, for $i \in Y \cap Z$, there is an edge between the left and right copies of $i$, and this is the only edge for both copies of $i$.

**Lemma 4** (Exchange Lemma)**.** *For any matroid $\mathcal{M} = (X, \mathcal{I})$ and independent set $Y, Z \in \mathcal{I}$ with $|Y| \leq |Z|$, the exchange graph $H_{\mathcal{M}}(Y, Z)$ contains a left perfect matching.*

Next, we prove several lemmas about vectors in the base polytope with respect to sets containing swappable elements. Lemma 7 relies on a *Swapping Lemma*, Lemma 6, for which the next lemma serves as a helper function.

**Lemma 5** (Weak Swapping Lemma)**.** *Let $\mathcal{M} = (X, \mathcal{I})$ be a matroid with an independent set $S \in \mathcal{I}$. For $C \subseteq S$, define*

$$U := \{i \in (X \setminus S) \cup C \mid (S \setminus C) \cup \{i\} \in \mathcal{I}\}.$$

*Then for any vector $x \in P_{\mathcal{B}(\mathcal{M})}$ in the base polytope one has $\sum_{i \in U} x_i \geq |C|$.*

**Input:** Node $i_0$ and set $S \in \mathcal{I}$ with $i_0 \in S$. Matching $M \subseteq \mathcal{E}_{\beta T}$ with $M_X = S \setminus \{i_0\}$.
Initialize: $A = A_0 = \emptyset$, $B = B_0 = \emptyset$, $C = \{i_0\}$, $\ell = 0$.

**while** TRUE **do**

Find disjoint $H \subseteq \mathcal{E}_{\alpha T}$ covering $D \subseteq (X \setminus S) \cup C$, s.t. $|D| \geq \varepsilon \cdot |C|$, $(S \setminus C) \cup D \in \mathcal{I}$, *and $H_W$ is disjoint from $A_W \cup B_W$.

*// Build the next layer in the augmenting tree*

**if** $H$ intersects at least $\frac{\varepsilon}{4} \cdot |H| \geq \frac{\varepsilon^2}{4} \cdot |C|$ many edges $M$ on $W$-side **then**

$B \leftarrow B \cup B_{\ell+1}$, $B_{\ell+1} = \{e \in M : e \cap H \neq \emptyset\}$

*// Find subset of $H$ to add to $A$*

**for** $b \in B_{\ell+1}$ **do**

Choose one edge $h_b \in H$ such that $h_b \cap b \neq \emptyset$

$A_{\ell+1} \leftarrow A_{\ell+1} \cup \{h_b\}$

**end for**

$C \leftarrow B_X \cup \{i_0\}$

$\ell \leftarrow \ell + 1$

*// Swap sets and collapse layers*

**else** $H' = \{e \in H : \text{val}(e_W \setminus M_W) \geq \beta T\}$ has size at least $\frac{\varepsilon}{4} \cdot |H|$

For all $e \in H'$, choose one $e' \subset e$ with $e' \in \mathcal{E}_{\beta T}$ and $e'_W \cap M_W = \emptyset$. Replace $e$ for $e'$ in $H'$.

*// Find a set to swap in, $\tilde{D}$, and a set to swap out, $\tilde{C}$*

$D' \subseteq D$ are the nodes covered by $H'$

$C' \subseteq C$ is such that $|C'| = |D'|$ and $S \setminus C' \cup D' \in \mathcal{I}$

**if** $i_0 \in C'$ **then**

Let $i_1 \in D'$ so that $S' := S \setminus \{i_0\} \cup \{i_1\} \in \mathcal{I}$ and let $e_1 \in H'$ be edge covering $i_1$.

Return $M' := M \cup \{e_1\}$ covering all of $S'$ and **terminate**.

**end if**

Layer $\tilde{\ell} \leq \ell$ contains $\tilde{C} \subset C' \cap (B_{\tilde{\ell}})_X$, with $|\tilde{C}| \geq \gamma |C'|$. **

Let $\tilde{D} \subset D'$ be such that $|\tilde{C}| = |\tilde{D}|$ and $S' := S \setminus \tilde{C} \cup \tilde{D} \in \mathcal{I}$.

$\tilde{M} \subset M$ covers $\tilde{C}$ and $\tilde{H} \subset H'$ covers $\tilde{D}$.

$M \leftarrow M \setminus \tilde{M} \cup \tilde{H}$, and $S \leftarrow S'$

$A \leftarrow A_{\leq \tilde{\ell}}$, $B \leftarrow B_{\leq \tilde{\ell}} \setminus \tilde{M}$, $C \leftarrow B_X \cup \{i_0\}$

$\ell \leftarrow \tilde{\ell}$

**end if*****

**end while**

* Possible by Lemma 7 with $W' := A_W \cup B_W$.

** By Lemma 9, such a $\tilde{C}$ exists.

*** One of the conditionals occurs by Lemma 8.

Figure 2.1: Main algorithm

*Proof.* Note that in particular $C \subseteq U$. Moreover, an equivalent definition of $U$ is

$$U = \{i \in (X \setminus S) \cup C \mid \exists j \in C : (S \setminus \{j\}) \cup \{i\} \in \mathcal{I}\}.$$

Due to the integrality of the base polytope, there is a basis $B \in \mathcal{I}$ with $\sum_{i \in U} x_i \geq \sum_{i \in U} (\chi(B))_i = |U \cap B|$, where $\chi(B) \in \{0,1\}^X$ is the characteristic vector of $B$. As $S$ and $B$ are independent sets with $|S| \leq |B|$, from Lemma 4 there is a left-perfect matching in the exchange graph $H_{\mathcal{M}}(S, B)$. The neighborhood of $C$ in $H_{\mathcal{M}}(S, B)$ is $U \cap B$. As there is a left-perfect matching, $|B \cap U|$ is least $|C|$ and hence $\sum_{i \in U} x_i \geq |U \cap B| \geq |C|$. □

Next, we derive a more general form of the Swapping Lemma (which coincides with the previous Lemma 5 if $D = \emptyset$):

**Lemma 6** (Strong Swapping Lemma)**.** *Let $\mathcal{M} = (X, \mathcal{I})$ be a matroid with an independent set $S \in \mathcal{I}$. Let $C \subseteq S$ and $D \subseteq (X \setminus S) \cup C$ with $|D| \leq |C|$ and $S \setminus C \cup D \in \mathcal{I}$. Define*

$$U := \{i \in ((X \setminus S) \cup C) \setminus D \mid S \setminus C \cup D \cup \{i\} \in \mathcal{I}\}.$$

*Then for any vector $x \in P_{\mathcal{B}(\mathcal{M})}$ in the base polytope one has $\sum_{i \in U} x_i \geq |C| - |D|$.*

*Proof.* Partition $C = C_1 \dot\cup C_2$ so that $C \cap D \subseteq C_1$, $|C_1| = |D|$ and $S' := S \setminus C_1 \cup D \in \mathcal{I}$. Then note that

$$\begin{aligned} U &= \Big\{i \in X \setminus \underbrace{(S \setminus C \cup D)}_{=S' \setminus C_2} \mid \underbrace{S \setminus C \cup D}_{=S' \setminus C_2} \cup \{i\} \in \mathcal{I}\Big\} \\ &= \{i \in (X \setminus S') \cup C_2 \mid S' \setminus C_2 \cup \{i\} \in \mathcal{I}\}. \end{aligned}$$

Then applying Lemma 5 gives

$$\sum_{i \in U} x_i \geq |C_2| = |C| - |D|.$$

□

Having proved our swapping lemma, we are equipped to prove the Expansion Lemma. Note that in our algorithm, layers are built to ensure that $|A_{\ell+1}| \leq |B_{\ell+1}|$. Due to this and the minimality of the edges in $\mathcal{E}_{\alpha T}$ and $\mathcal{E}_{\beta T}$, $W' := A_W \cup B_W$ has $\mathrm{val}(W') \leq (\alpha + \beta + \delta)T \cdot |C|$.

**Lemma 7** (Expansion Lemma)**.** *Let $C \subseteq S \in \mathcal{I}$, $W' \subseteq W$ with val$(W') \leq (\alpha+\beta+\delta)T \cdot |C|$. Further, let $\mu := \frac{1-2\alpha-\beta-\delta}{1+\delta} > 0$ and assume that there exists $(x, y) \in Q(T)$. Then there is a set $D \subseteq (X \setminus S) \cup C$ of size $|D| \geq \lceil \mu \cdot |C| \rceil$ covered by a matching $H \subseteq \mathcal{E}_{\alpha T}$ so that $H_W \cap W' = \emptyset$ and $(S \setminus C) \cup D \in \mathcal{I}$.*

*Proof.* Note that $D$ may contain elements from $C$. Greedily choose $D$ and the matching $H$ with $|D| = |H|$ one node/edge after the other. Suppose the greedy procedure gets stuck — no edge can be added without intersecting $W' \cup H_W$. For the sake of contradiction assume this happens when $|D| < \mu|C|$. First, let

$$U := \{i \in ((X \setminus S) \cup C) \setminus D \mid (S \setminus C) \cup D \cup \{i\} \in \mathcal{I}\}$$

be the nodes which could be added to $D$ while preserving independence. Then for our fixed $x \in P_{\mathcal{B}(\mathcal{M})}$, by Lemma 6 one has

$$\sum_{i \in U} x_i \geq |C| - |D| > (1-\mu) \cdot |C|.$$

Let $W'' := W' \cup H_W$ be the right hand side resources that are being covered by the augmenting tree. Here, we let $W' = A_W \cup B_W$. Using the minimality of the adding and blocking edges,

$$\text{val}(W'') \leq \mu|C|(\alpha+\delta)T + |C|(\beta+\alpha+\delta)T = |C|T(\mu(\alpha+\delta)+\beta+\alpha+\delta).$$

By the assumption that the greedy procedure is stuck, there is no edge $e \in \mathcal{E}_{\alpha T}$ with $e_X \in U$ and $e \cap W'' = \emptyset$. If $N(i)$ denotes the neighborhood of $i \in X$ in the bipartite graph $G$, then this means that val$(N(i) \setminus W'') < \alpha T$ for all $i \in U$. For every fixed $i \in U$ we can then lower bound the $y$-weight going into $W''$ as

$$\sum_{(i,w) \in E : w \in W''} p_w y_{i,w} = \underbrace{\sum_{w \in \delta(i)} p_w y_{i,w}}_{\geq T x_i} - \sum_{(i,w) \in E : w \notin W''} p_w \underbrace{y_{i,w}}_{\leq x_i} \geq T x_i - x_i \underbrace{\Big( \sum_{(i,w) \in E : w \notin W''} p_w \Big)}_{< \alpha T} \geq T \cdot x_i \cdot (1-\alpha) \quad (*)$$

Then double counting the $y$-weight running between $U$ and $W''$ with a lower and upper bound shows that

$$(1-\alpha)T \underbrace{\sum_{i \in U} x_i}_{\geq (1-\mu)|C|} \leq \sum_{(i,w) \in E : i \in U, w \in W''} p_w y_{i,w} \leq \sum_{w \in W''} p_w \underbrace{y(\delta(w))}_{\leq 1} \leq \text{val}(W'')$$

Simplifying the above,

$$(1-\alpha) \cdot (1-\mu) \cdot T|C| < (\mu(\alpha+\delta) + \beta + \alpha + \delta) \cdot T|C| \quad \Rightarrow \quad \frac{1 - 2\alpha - \beta - \delta}{1+\delta} < \mu.$$

Thus we reach a contradiction for our choice of $\mu$. □

The algorithm relies on the fact that from the set of hyperedges, $H$, guaranteed by the Expansion Lemma, there is either some constant fraction of $H$ to swap into the matching, or a constant fraction of $H$ is blocked by edges in the current matching. In the former, significant space is found in $W$ for $S$. In the latter, enough edges of the matching are intersected to guarantee the next layer in the augmenting tree is large. The following lemma proves at least one of these conditions occurs.

**Lemma 8.** *Set $\mu := \frac{\alpha - \beta}{\delta + \alpha} > 0$. Let $M \subseteq \mathcal{E}_{\beta T}$ and $F \subseteq \mathcal{E}_{\alpha T}$ both be hypergraph matchings. Further, let*

$$H := \{e \in F \mid \textit{val}(e_W \setminus M_W) \geq \beta T\}$$

*be the edges in $F$ that still have value $\beta T$ after overlap with $M$ is removed. Then either (i) $|H| \geq \mu |F|$ or (ii) $F$ intersects at least $\mu |F|$ edges of $M$.*

*Proof.* Let $W' := M_W \cap F_W$ be the right hand side nodes where the hypermatchings overlap and suppose for the sake of contradiction that neither of the two cases occur. Then double counting the value of $W'$ gives

$$\mu \cdot (\beta+\delta) \cdot T \cdot |F| > (\beta+\delta) T \cdot \underbrace{(\#\text{edges in } M \text{ intersecting } W')}_{\mu |F|>} \geq \text{val}(W') \geq \underbrace{|F \setminus H|}_{\geq (1-\mu)\cdot |F|} \cdot (\alpha - \beta) \cdot T.$$

Rearranging and simplifying, the above implies $\mu > \frac{\alpha - \beta}{\delta + \alpha}$. Thus we contradict our choice of $\mu$. □

Our last lemma will show that a constant fraction of the nodes which could be swapped out of the augmenting tree come from the same *layer* in the tree. This allows us to swap out enough nodes from the same layer to make substantial progress with each iteration. Here $C'$ and $\tilde{C}$ are labelled the same as in the algorithm.

**Lemma 9.** *Let sets $C'$ and $\{B_i\}_{i=0}^{\ell}$ be such that $C' \subset (B_{\leq \ell})_X$. Further, suppose there exists constant $c > 0$ such that $|C'| \geq c \cdot |B_{\leq \ell}|$ and $|B_{i+1}| \geq c \cdot |B_{\leq i}|$ for $i = 0, \ldots, \ell - 1$. Then, there exists a layer $0 \leq \tilde{\ell} \leq \ell$ and constant $\gamma := \gamma(c) > 0$, such that $\tilde{C} := C' \cap (B_{\tilde{\ell}})_X$ has size $|\tilde{C}| \geq \gamma \cdot |C'|$.*

*Proof.* By induction, $|B_{\leq \ell}|$ can be written in terms of lower indexed sets as

$$|B_{\leq \ell}| \geq (1 + c)^k \cdot |B_{\leq \ell - k}|,$$

for $k = 0, \ldots, \ell$. Therefore, the size of $C'$ can be written as $|C'| \geq c(1 + c)^k \cdot |B_{\leq \ell - k}|$. As $c$ is a constant, take $k$ large enough so $c(1 + c)^k \geq 2$, namely $k \geq \frac{\log(\frac{2}{c})}{\log(1+c)}$. Then the collection of sets $(B_{\ell - i})_X$ for $i = 0, \ldots, k$ contain at least half of $C'$, so one of them must contain at least $\gamma = \frac{1}{2(k+1)}$ of $C'$. □

### *2.3.4 Termination and runtime*

As seen in Lemma 9,

$$|X| \geq |B_{\leq \ell}| \geq \left(1 + \frac{\varepsilon^2}{4}\right)^{\ell} |B_0|,$$

and solving for $\ell$ shows $\frac{\log(|X|)}{\log\left(1 + \frac{\varepsilon^2}{4}\right)} \geq \ell$. Thus the total number of layers at any step in the algorithm is $O(\log |X|)$. Note after each collapse of the layers, the matching $M$ and possibly the independent set $S$ are updated. However, the fixed exposed node $i_0$ will remain in $S$ until the very last iteration in which the algorithm finds an edge $e_1$ that augments the matching. Before we begin discussing the proof guaranteeing our algorithm terminates, we need a lemma to compare the number of blocking edges after a layer is collapsed to the number of blocking edges at the beginning of the iteration.

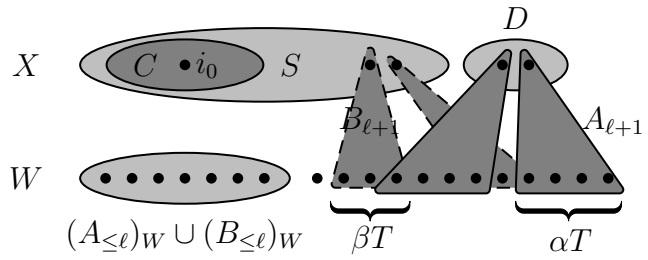



Figure 2.2: Case 1 of the algorithm, where a set $A_{\ell+1} \subseteq \mathcal{E}_{\alpha T}$ of hyperedges is found that intersects many new edges $B_{\ell+1} \subseteq (M \setminus B_{\leq \ell})$. In particular $|B_{\ell+1}| \geq \Omega_\varepsilon(|C|)$. Note that $D$ might contain nodes from $C$.

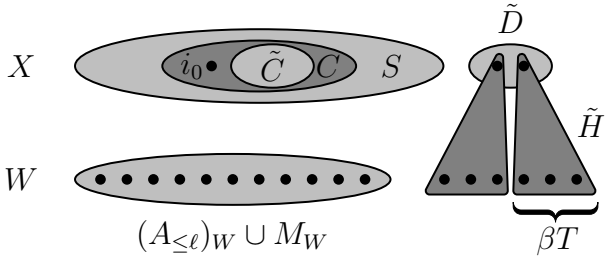



Figure 2.3: Case 2 of the algorithm, where $\tilde{H} \subseteq \mathcal{E}_{\beta T}$ of size $|\tilde{H}| \geq \Omega_\varepsilon(|C|)$ is found so that $(i)$ $\tilde{H}$ is disjoint on the $W$-side to the matching $M$ and the adding edges in the augmenting tree, $(ii)$ $\tilde{H}$ covers a set $\tilde{D}$ with $S \setminus \tilde{C} \cup \tilde{D} \in \mathcal{I}$, and $(iii)$ $\tilde{C}$ is from one layer of the augmenting tree. Here $\tilde{D}$ and $\tilde{C}$ do not have to be disjoint.

**Lemma 10.** *Let $\tilde{\ell}$ be the index of the collapsed layer and let $B'$ be the updated blocking edges after a collapse step. Then, $|B'_{\leq\tilde{\ell}}| \leq |B_{\leq\tilde{\ell}}| \cdot (1 - \frac{\varepsilon^2}{4} \cdot \gamma)$.*

*Proof.* Recall $B'_{\tilde{\ell}} = B_{\tilde{\ell}} \setminus F$ for $F$ the edges of $M$ covering $\tilde{C}$. Further, the blocking edges in layers indexed less than $\tilde{\ell}$ are not effected in the iteration. Hence

$$|B'_{\leq\tilde{\ell}}| \;=\; |B'_{\leq\tilde{\ell}-1}| + |B'_{\tilde{\ell}}| = |B_{\leq\tilde{\ell}-1}| + |B'_{\tilde{\ell}}|$$

From Lemmas 7 and 8, $|B_{\ell+1}| \geq \frac{\varepsilon^2}{4}|B_{\leq\ell}|$. Then examining the collapsed layer by itself, we see

$$|B'_{\tilde{\ell}}| \;=\; |B_{\tilde{\ell}}| - |F| \leq |B_{\tilde{\ell}}| - \frac{\varepsilon^2}{4} \cdot \gamma |B_{\leq\tilde{\ell}}|.$$

Substituting back into $|B'_{\leq\tilde{\ell}}|$, we find that

$$\begin{aligned} |B'_{\leq\tilde{\ell}}| \;&\leq\; |B_{\leq\tilde{\ell}-1}| + |B_{\tilde{\ell}}| - \frac{\varepsilon^2}{4} \cdot \gamma |B_{\leq\tilde{\ell}}| \\ &=\; |B_{\leq\tilde{\ell}}| - \frac{\varepsilon^2}{4} \cdot \gamma |B_{\leq\tilde{\ell}}| = |B_{\leq\tilde{\ell}}| \cdot \big(1 - \frac{\varepsilon^2}{4} \cdot \gamma\big). \end{aligned}$$

□

To prove the algorithm terminates in polynomial time, we consider a signature vector $s = (s_0, s_1, \ldots, s_\ell, \infty)$, where $s_j = \lfloor \log_c |B_{\leq j}| \rfloor$ for $c = \frac{1}{1-\frac{\varepsilon^2}{4}\cdot\gamma}$. The signature vector and proof that the algorithm terminates is inspired by [AKS15], but it is subtly different.

**Lemma 11.** *The signature vector decreases lexicographically after each iterative loop in the algorithm.*

*Proof.* Let $s = (s_0, \ldots, s_\ell, \infty)$ be a signature vector at the beginning of a step in the algorithm, and let $s'$ be the result of $s$ through one iteration of the algorithm. For $\ell+1$ denoting the newest built layer in the algorithm, if the newest set of hyperedges found intersects at least $\frac{\varepsilon^2}{4}|C|$ many edges of $M$, then another layer in the augmenting tree is built and no layer is collapsed. Then $s' = (s_0, \ldots, s_\ell, s_{\ell+1}, \infty)$ is lexicographically smaller than $s$.

Otherwise, layer $0 \leq \tilde{\ell} \leq \ell$ is collapsed. All finite coordinates above $s_{\tilde{\ell}}$ are deleted from the signature vector, and all coordinates before $s_{\tilde{\ell}}$ are unaffected. So it suffices to check that $s'_{\tilde{\ell}} < s_{\tilde{\ell}}$. Again, let $B'$ be the updated blocking edges after a collapse step. As $B_{\tilde{\ell}}$ is the only set of blocking edges in $B_{\leq \tilde{\ell}}$ affected by the collapse, by Lemma 10 one has $|B'_{\leq \tilde{\ell}}| \leq |B_{\leq \tilde{\ell}}|(1 - \frac{\varepsilon^2}{4} \cdot \gamma)$. Taking a log we compare the coordinates

$$s'_{\tilde{\ell}} = \left\lfloor \log_c \left( \left| B'_{\leq \tilde{\ell}} \right| \right) \right\rfloor \leq \left\lfloor \log_c \left( \left| B_{\leq \tilde{\ell}} \right| \left( 1 - \frac{\varepsilon^2}{4} \cdot \gamma \right) \right) \right\rfloor = \left\lfloor \log_c \left( \left| B_{\leq \tilde{\ell}} \right| \right) \right\rfloor - 1 = s_{\tilde{\ell}} - 1.$$

□

Choose the infinite coordinate to be some integer larger than $\log |X|$. Since for every layer $\ell$, we have $|B_{\leq \ell}| \leq |X|$, then every coordinate of the signature vector is upper bounded by $U = O(\log |X|)$. Recall the number of layers, and thus the number of coordinates in the signature vector, is also upper bounded by $U$. Together, these imply that the sum of the coordinates of the signature vector is at most $U^2$.

As the signature vector has non-decreasing order, each signature vector corresponds to a partition of an integer $z \leq U^2$. On the other hand, every partition of some $z \leq U^2$ has a corresponding signature vector. Thus we apply a result of Hardy and Ramanujan to find the total number of signature vectors is $\sum_{k \leq U^2} e^{O(\sqrt{k})} = |X|^{O(1)}$. Since each iteration of the algorithm can be done in polynomial time and the signature vector decreases lexicographically after each iteration, the algorithm terminates after a total time of $n^{\Theta_\varepsilon(1)}$.

### *2.4 Application to Santa Claus*

In this section, we show a polynomial time $(4 + \varepsilon)$-approximation algorithm for the Santa Claus problem. Recall that for a given set of children $M$, and a set of presents $J$, the Santa Claus problem asks how Santa should distribute presents to children in order to maximize the minimum happiness of any child[4]. Here, present $j$ is only wanted by some subset of children that we denote by $A_j \subseteq M$, and present $j$ has value $p_j$ to child $i \in A_j$. The happiness of

[4]We assume Santa to be an equitable man– not one influenced by bribery, social status, etc.

child $i$ is the sum of all $p_j$ for presents $j$ assigned to child $i$. We assume w.l.o.g. to know the integral objective function value $T$ of the optimum solution, otherwise $T$ can be found by binary search.

We partition gifts into two sets: *large* gifts $J_L := \{j \in J \mid p_j > \delta_2 T\}$ and *small* gifts $J_S := \{j \in J \mid p_j \leq \delta_1 T\}$, for parameters $0 < \delta_1 \leq \delta_2 < 1$ such that all gifts have values in $[0, \delta_1 T] \cup (\delta_2 T, T]$. Let $P(T, \delta_1, \delta_2)$ be the set of vectors $z \in \mathbb{R}_{\geq 0}^{J \times M}$ satisfying

$$\begin{aligned}
\sum_{j \in J_S : i \in A_j} p_j z_{ij} &\geq T \cdot \Big(1 - \sum_{j \in J_L : i \in A_j} z_{ij}\Big) && \forall i \in M \\
\sum_{i \in A_j} z_{ij} &\leq 1 && \forall j \in J \\
z_{ij} &\leq 1 - \sum_{j' \in J_L : i \in A_{j'}} z_{ij'} && \forall j \in J_S \; \forall i \in A_j
\end{aligned}$$

If $n = |J| + |M|$, then this LP has $O(n^2)$ many variables and $O(n^2)$ many constraints. To see that this is indeed a relaxation, take any feasible assignment $\sigma : J \to M$ with $\sum_{j \in \sigma^{-1}(i)} p_j \geq T$ for all $i \in M$. Now let $\sigma : J \to M \cup \{\emptyset\}$ be a modified assignment where we set $\sigma(j) = \emptyset$ for gifts that we decide to drop. For each child $i \in M$ that receives at least one large gift we drop all small gifts and all but one large gift. Then a feasible solution $z \in P(T, \delta_1, \delta_2)$ is obtained by letting

$$z_{ij} := \begin{cases} 1 & \text{if } \sigma(j) = i \\ 0 & \text{otherwise.} \end{cases}$$

We will show that given a feasible solution $z \in P(T, \delta_1, \delta_2)$, there exists a feasible solution $(x^*, y^*)$ to $Q(T)$. To do this, we will exploit two underlying matroids in the Santa Claus problem, allowing us to apply Theorem 2. Let

$$\mathcal{I} = \{M_L \subseteq M \mid \exists \text{ left-perfect matching between } M_L \text{ and } J_L \text{ using edges } (i,j) : i \in A_j\},$$

be a family of independent sets. Then $\mathcal{M} = (M, \mathcal{I})$ constitutes a *matchable set matroid*. We denote the *co-matroid* of $\mathcal{M}$ by $\mathcal{M}^* = (M, \mathcal{I}^*)$. Recall that the independent sets of the

co-matroid are given by

$$\mathcal{I}^* = \{M_S \subseteq M |\ \exists M_L \in \mathcal{B}(\mathcal{M}) : M_S \cap M_L = \emptyset\}.$$

We can define a vector $x \in \mathbb{R}^M$ with $x_i = \sum_{j \in J_L : i \in A_j} z_{ij}$ that lies in the matroid polytope of $\mathcal{M}$. This fact follows easily from the integrality of the fractional matching polytope in bipartite graphs. It is instructive to think of $x_i$ as the decision variable telling whether child $i \in M$ should receive a large present.

Unfortunately, $x$ does not have to lie in the base polytope — in fact the sum $\sum_{i \in M} x_i$ might not even be integral. However, there always exists a vector $x'$ in the base polytope that covers every child just as well with large presents as $x$ does. This observation can be stated for general matroids:

**Lemma 12.** *Let $\mathcal{M} = (X, \mathcal{I})$ be any matroid and let $x$ be a point in its matroid polytope. Then in polynomial time one can find a point $x'$ in the base polytope so that $x' \geq x$ coordinate-wise.*

In fact the algorithm behind this claim is rather trivial: as long as $x \in P_{\mathcal{M}}$ is not in the base polytope, there is always a coordinate $i$ and a $\mu > 0$ so that $x + \mu e_i \in P_{\mathcal{M}}$.

With the new vector $x' \in P_{\mathcal{B}(\mathcal{M})}$ at hand, we can redefine the $z$-assignments by letting

$$z'_{ij} = \begin{cases} z_{ij} & x_i = 1 \\ \frac{1-x'_i}{1-x_i} z_{ij} & x_i \neq 1. \end{cases}$$

for $j \in J_S$; the new values $z'_{ij}$ for $j \in J_L$ can be obtained from the fractional matching that corresponds to $x'_i$. Note that $0 \leq z'_{ij} \leq z_{ij}$ for $j \in J_S$. The reader should be convinced that still $z' \in P(T, \delta_1, \delta_2)$, just that the corresponding vector $x'$ now lies in $P_{\mathcal{B}(\mathcal{M})}$[5].

[5]There is an alternative proof without the need to replace $x$ by $x'$. Add the constraint $\sum_{j \in J_L, i \in A_j} z_{ij} = \text{rank}(\mathcal{M})$ to $P(T, \delta_1, \delta_2)$. There is always a feasible integral solution satisfying this constraint. Then for any fractional solution $z \in P(T, \delta_1, \delta_2)$, the corresponding vector $x$ will immediately lie in the base polytope.

It is well known in matroid theory that the complementary vector $x^* := \mathbf{1} - x'$ lies in $P_{\mathcal{B}(\mathcal{M}^*)}$. Again, it is instructive to think of $x^*_i$ as the decision variable whether child $i$ has to be satisfied with small gifts. Finally, the assignments $y^*$ are simply the restriction of $z'$ on the coordinates $(i,j) \in M \times J_S$. The obtained pair $(x^*, y^*)$ lies in $Q(T)$, where the matroid in the definition of $Q(T)$ is $\mathcal{M}^*$.

As $Q(T) \neq \emptyset$, we can apply Theorem 2 which results in a subset $M_S \in \mathcal{B}(\mathcal{M}^*)$ of the children and an assignment $\sigma : J_S \to M_S$, where each child in $M_S$ receives happiness at least $\left(\frac{1}{3} - \frac{\delta_1}{3} - \varepsilon\right) \cdot T$ from the assignment of small gifts. Implicitly due to the choice of the matroid $\mathcal{M}^*$, we know that the remaining children $M \setminus M_S = M_L$ can all receive one large gift and this assignment can be computed in polynomial time using a matching algorithm. Overall, each child receives either one large present of value at least $\delta_2 \cdot T$ or small presents of total value at least $(\frac{1}{3} - \frac{\delta_1}{3} - \varepsilon) \cdot T$. Therefore each child receives value at least

$$\min\left\{\left(\frac{1}{3} - \frac{\delta_1}{3} - \varepsilon\right) \cdot T, \delta_2 \cdot T\right\} \geq \left(\frac{1}{4} - \varepsilon\right) \cdot T \tag{2.1}$$

for a choice of $\delta_2 = \delta_1 = \frac{1}{4}$. In some instances of Santa Claus, we can do better. Set $\delta_1$ so that $\delta_1 \cdot T$ is the largest gift value that is at most $\frac{1}{4}T$, and set $\delta_2$ so that $\delta_2 \cdot T$ is the smallest gift value that is at $\frac{1}{4}T$. Then the algorithm guarantees that each child receives value at least as in the left hand side of Equation 2.1. When $\delta_1$ and $\delta_2$ are bounded away from $1/4$, then the approximation improves. For example, when $\delta_2 \geq 1/3$ and $\delta_1 T$ is close to 0, such as in the case where all gifts have value either $T$ or 1, we approach a $(3+\varepsilon)$-approximation.

# Chapter 3

# General Max-min Fair Allocation

### *3.1 Introduction*

In this document we are discussion a new and simpler approach to approximate the *Max Min Fair Allocation problem*:

| MaxMinFairAllocation |
| --- |
| **Input:** Agents $i \in [m]$ and items $j \in [n]$ with utilities $p_{ij} \geq 0$. |
| **Ouput:** Assignment $\sigma : [n] \to [m]$ that maximizes $\min_{i \in [m]} \sum_{j \in \sigma^{-1}(i)} p_{ij}$ |

A review of the previous work is available in Section 2.1.2. To recap, the problem is **NP**-hard to approximate up to any factor smaller than 2 by Bezakova and Dani [BD05], who also gave an $(n - m + 1)$-approximation algorithm. The integrality gap of the configuration LP is at least $\Omega(\sqrt{n})$ [BS06], and then, Asadpour and Saberi [AS10] gave an $O(\sqrt{m} \log^3 m)$ approximation for the problem using the same LP relaxation. Meanwhile, a rather complicated $O(\log^{10} n)$-approximation in quasi-polynomial time was given by Chakrabarty, Chuzoy and Khanna [CCK09] from FOCS 2009. Their algorithm can also achieve an $O(n^{\varepsilon})$ approximation in polynomial time for any fixed $\varepsilon > 0$.

### *3.1.1 Our Contributions*

This is actually a work in progress, where we propose an approach that looks promising but only have partial results currently. In Theorem 13, we will see a rather clean reduction from the general max-min fair allocation problem to the hypergraph matching problem defined in the next section by losing an $O(\log n)$ factor. Then, a candidate algorithm is proposed in Section 3.3.

To prove the correctness of the candidate algorithm, proving Conjecture 14 would be sufficient. A natural way is to apply the Hall's condition. However, a naive union bound approach would not work since there are exponentially many events. Although we do not succeed to analyze the original random experiment, if instead of sampling hyperedges one would sample singletons, then we can prove that the random experiment is successful and a fractional matching would exist. Section 3.4 to 3.6 provides a successful analysis of the single vertex failure random experiment.

## *3.2 Hypergraph matchings*

First we want to reduce the problem to a simpler looking hypergraph matching problem while only loosing a logarithmic factor in the approximation.

Recall that a *hypergraph* $\mathcal{H} = (A \dot\cup B, \mathcal{E})$ is called *bipartite* if $|e \cap A| = 1$ for all edges $e \in \mathcal{E}$. For a parameter $0 \leq \delta \leq 1$ we define

$$\mathcal{E}_{\geq\delta} := \left\{ e' \mid \exists e \in \mathcal{E} \text{ with } e' \subseteq e \text{ and } |e' \cap A| = 1 \text{ and } |e' \cap B| \geq \lceil \delta \cdot |e \cap B| \rceil \right\}$$

as all the sub-edges that have at least a $\delta$-fraction of right hand side nodes. A set $M \subseteq \mathcal{E}$ is called an *A-perfect matching* if (i) $\#(e \in M \text{ with } a \in e) = 1 \ \forall a \in A$ and (ii) $\#(e \in M \text{ with } b \in e) \leq 1 \ \forall b \in B$. We consider the following problem:

| HypergraphMatching |
| --- |
| **Input:** A bipartite hypergraph $\mathcal{H} = (A \dot\cup B, \mathcal{E})$ that is guaranteed to contain an $A$-perfect matching. |
| **Output:** An $A$-perfect matching $M \subseteq \mathcal{E}_{\geq \delta}$. The objective is to maximize $0 \leq \delta \leq 1$. |

Note that for $e \in \mathcal{E}$ with $|e \cap B| = 1$ one cannot take any subedge – one has to take the whole edge.

**Theorem 13.** *Let $\alpha \geq 1$. If there is an $\alpha$-approximation algorithm for* Hypergraph Matching*, then there is an $O(\log n) \cdot \alpha$-approximation algorithm for* MaxMin Fair Allocation.

*Proof.* Take a MaxMin Fair Allocation instance and assume that the value OPT [1] of the optimum solution is known. Then by losing a constant factor we may assume that for every $i, j$ one has either $p_{ij} = 0$ or one has $p_{ij} = 2^{-k}\text{OPT}$ with $k \in \{0, \ldots, \log(n)\}$. For every agent $i \in [m]$ we have $\log(n)+1$ many "buckets" $B_{ik} = \{j \in [n] \mid p_{ij} = 2^{-k}\text{OPT}\}$ of items that have the same utility. Now, we create a bipartite hypergraph $\mathcal{H} = ([m] \dot\cup [n], \mathcal{E})$ with agents on one side and items on the other side. For every agent $i \in [m]$ and every minimal set $S \subseteq B_{ik}$ with $|S| \cdot 2^{-k} \geq \frac{1}{\log(n)+1}$, we create one hyper edge $\{i\} \cup S$. Then an $[m]$-perfect matching $M \subseteq \mathcal{E}_{\geq 1/\alpha}$ corresponds to a MaxMin FairAllocation solution of value $\Theta(\frac{1}{\alpha \log(n)}) \cdot \text{OPT}$. □

Note that in principle, the defined hypergraph can have an exponential number of edges, and the reduction is not necessarily doable in polynomial time. But one can expect that an algorithm for Hypergraph Matching would not need to do the enumeration of all subsets explicitly.

[1] Actually, we do not need to know the exact value of OPT. The reduction still works by setting this threshold within a constant factor, then a standard search procedure guarantees it.

### *3.3 A candidate algorithm for HypergraphMatching*

If we have an undirected bipartite graph $G = (V \dot{\cup} U, E)$, then a *demand function* is a function of the form $d : V \to \mathbb{Z}_{\geq 0}$, that means every left hand side node $v \in V$ has some demand $d(v)$. We say that a set of edges $F \subseteq E$ is a *d-matching* if $|\delta_F(v)| = d(v)$ for $v \in V$ and $|\delta_F(u)| \leq 1$ for all $u \in U$. Here $\delta_F(v)$ are the edges of $F$ that are incident to $v$. If we have a subset hyperedges $\mathcal{F} \subseteq \mathcal{E}$ in a hypergraph $\mathcal{H} = (A \dot{\cup} B, \mathcal{E})$, then in a natural way these induce a 2-uniform graph $G = (A \dot{\cup} B, \{(i, j) : i \in A, j \in B, \{i, j\} \subseteq e \text{ for some } e \in \mathcal{F}\})$. We will now consider a candidate approximation algorithm for Hypergraph Matching.

RANDOMIZED ROUNDING ALGORITHM FOR HYPERGRAPH MATCHING

**Input:** A bipartite hypergraph $\mathcal{H} = (A \dot\cup B, \mathcal{E})$ and the promise that there is an $A$-perfect maching $M \subseteq \mathcal{E}$

**Output:** An $A$-perfect matching $M \subseteq \mathcal{E}_{\geq 1/(C\log(n))^2}$

(1) Partition $\mathcal{E} = \mathcal{E}_{\text{small}} \dot\cup \mathcal{E}_{\text{large}}$ with $\mathcal{E}_{\text{small}} = \{e \in \mathcal{E} : |e \cap B| = 1\}$ and $\mathcal{E}_{\text{large}} = \{e \in \mathcal{E} : |e \cap B| \geq (C\log(n))^2\}$ (we can w.l.o.g. replace edges with $1 < |e \cap B| < (C\log(n))^2$ by subedges of size 2).

(2) Compute a 1-round Sherali-Adams[a] $\boldsymbol{x}^* \in \mathrm{SA}_1(K)$, where $K$ is the set of vectors $\boldsymbol{x} \in \mathbb{R}^{\mathcal{E}}$ satisfying

$$\begin{aligned} \sum_{e \in \delta_{\mathcal{E}}(a)} x_e &= 1 \quad \forall a \in A \\ \sum_{e \in \delta_{\mathcal{E}}(b)} x_e &\leq 1 \quad \forall b \in B \\ 0 \leq x_e &\leq 1 \quad e \in \mathcal{E} \end{aligned}$$

(3) For every $a \in A$ make an independent random decision and with probability $\min\{C\log(n) \cdot \boldsymbol{x}^*(\delta_{\mathcal{E}_{\text{large}}}(a)), 1\}$, add $a$ to $A_{\text{large}}$; otherwise add $a$ to $A_{\text{small}}$.

(4) For every $a \in A_{\text{large}}$, pick a random edge $e_a \in \delta_{\mathcal{E}_{\text{large}}}(a)$ proportional to the value $x^*_{e_a}$.

(5) Define a demand function $d : A \to \mathbb{Z}_{\geq 1}$ and set

$$d(a) := \begin{cases} 1 & \text{if } a \in A_{\text{small}} \\ \left\lfloor \frac{|e_a|}{(C\log(n))^2} \right\rfloor & \text{if } a \in A_{\text{large}} \end{cases}$$

(6) Define a bipartite graph $G = (A \dot\cup B, E)$ that consists of all the size-2 subedges of $\mathcal{E}_{\text{small}} \cup \{e_a : a \in A_{\text{large}}\}$.

(7) Compute a $d$-matching $F \subseteq E$ in the graph $G$. Return $M := \{\{a\} \cup \{b \in B : (a,b) \in F\} \mid a \in A\}$

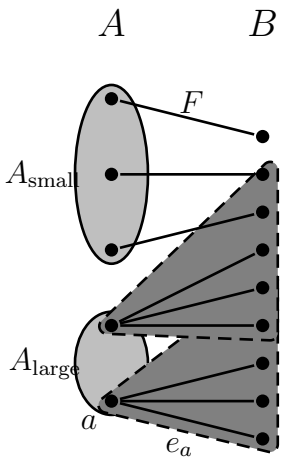
[a] see Section 4.2.1 for a detailed description of the Sherali-Adams lift

An informal description of the algorithm is as follows: We have a hypergraph with *small* hyperedges with $|e| = 2$ and large hyperedges with $|e| \geq (C\log(n))^2 + 1$. We solve the 1-round Sherali Adams lift of the standard hypergraph matching LP and denote the solution by $\boldsymbol{x}^*$. Then we sample every large hyperedge with probability $C\log(n) \cdot x^*(e)$ (with the modification that we need at most one hyperedge covering a node $a$, so we cap the sum of the probabilities at 1). Here $e_a$ denotes the sampled large edge incident to $a \in A_{\text{large}}$. The nodes of $A$ that are covered by such hyperedges are called $A_{\text{large}}$, the uncovered nodes are called $A_{\text{small}}$. Then in polynomial time we compute a $d$-matching that assigns every node in $A_{\text{small}}$ a small edge and every node in $A_{\text{large}}$ receives a $\frac{1}{(C\log(n))^2}$-fraction of nodes from $e_a$.

We want to avoid using the floor operator in (5), hence let us assume for the sake of simplicity that $\frac{|e|}{(C\log(n))^2} \in \mathbb{Z}$ for every large edge.

The crucial ingredient would be proving the following:

**Conjecture 14.** *With high probability, in step (7), the $d$-matching $F$ exists.*




We would like to point out that at this point we do cannot even analyze the simpler case where each node in $A$ is contained either only in large hyperedges or only in small hyperedges (meaning that there is no randomness in forming the partition $A = A_{\text{large}} \dot{\cup} A_{\text{small}}$).

### 3.4 Analysis of the single vertex failure random experiment via Supermodularity

Since we did not ultimately succeed to analyze the original random experiment, a simpler case is considered where sampling hyperedges are replaced with sampling singletons. In the following sections, we will prove the following general statement which in particular provides successful analysis of the *single vertex failure random experiment*:

**Theorem 15.** *For $\alpha := C\log(n)$ with a large enough constant $C > 0$ the following holds. Let $F : 2^{[n]} \to \mathbb{Z}_{\geq 0}$ be a non-negative monotone integral supermodular function and let $0 < p_j \leq \frac{1}{\alpha}$ be values for $j \in [n]$. Let $X_1, \ldots, X_n \in \{1 - \frac{1}{\alpha}, 1\}$ be independent random variables with $\Pr[X_j = 1 - \frac{1}{\alpha}] = p_j$ for each $j \in [n]$. Assume that*

$$F(S) \leq |S| - \alpha \cdot \sum_{j \in S} p_j \quad \forall \emptyset \subseteq S \subseteq [n]$$

*Then*

$$\Pr\left[F(S) \leq \sum_{j \in S} X_j \ \ \forall S \subseteq [n]\right] \geq 1 - \frac{1}{n^{100}}$$

Recall that a function $F : 2^{[n]} \to \mathbb{R}$ is called *supermodular* if

$$F(A \cup \{j\}) - F(A) \leq F(B \cup \{j\}) - F(B) \quad \forall A \subseteq B \subseteq [n] \quad \forall j \in [n] \setminus B$$

One can think of this as an *increasing returns* property. An equivalent characterization of supermodular function is to require that

$$F(A) + F(B) \leq F(A \cap B) + F(A \cup B) \quad \forall A, B \subseteq [n]$$

In order to prove Theorem 15, we will first derive several structural properties that allow us to find a small *net*. Then in a later section we will provide a probabilistic analysis which works by taking a union bound over the constructed net.

## *3.5 Structural properties*

Consider a monotone supermodular function $F : 2^{[n]} \to \mathbb{Z}_{\geq 0}$. We say that a capacity vector $\boldsymbol{X} \in \mathbb{R}^n_{\geq 0}$ is called *feasible* for $F$, if $F(S) \leq \sum_{j \in S} X_j \; \forall S \subseteq [n]$. Otherwise the vector is called *infeasible*.

### *3.5.1 The deficit function and closedness of the set of maximizers*

For $I \subseteq [n]$, we define a function $D_I : 2^{[n]} \to \mathbb{Z}$ so that $D_I(S)$ gives the "deficit" of $S$ with respect to the capacity vector $\mathbf{1} - \mathbf{1}_I$. More formally we define

$$D_I(S) := F(S) - \sum_{j \in S} (\mathbf{1} - \mathbf{1}_I)_j = F(S) - |S \setminus I| = F(S) - |S| + |S \cap I|$$

Since $F(S)$ is supermodular and $\sum_{j \in S} (\mathbf{1} - \mathbf{1}_I)_j$ is linear, the resulting difference $D_I(S)$ is supermodular as well (though $D_I$ is in general not monotone). Recall the following:

**Lemma 16.** *The family $\mathcal{F}_I$ of maximizers of $D_I$ is closed under taking union and intersection. In particular the set $S^*_I := \bigcup_{S \in \mathcal{F}_I} S$ is contained in $\mathcal{F}_I$.*

*Proof.* Take two sets $A, B \in \mathcal{F}_I$, meaning that $D_I(A) = D_I(B) = \alpha$ for some $\alpha \in \mathbb{R}$. Then

$$2\alpha \overset{\text{maximality}}{\geq} \underbrace{D_I(A \cup B)}_{\leq \alpha} + \underbrace{D_I(A \cap B)}_{\leq \alpha} \overset{\text{supermod}}{\geq} D_I(A) + D_I(B) \overset{\text{maximality}}{=} 2\alpha$$

We can see that indeed all inequalities have to be equalities and $D_I(A \cup B) = \alpha$ and $D_I(A \cap B) = \alpha$. Hence $\mathcal{F}_I$ is indeed closed under taking unions and intersections. The other claim immediately follows. □

### *3.5.2 Properties of $S^*_I$*

It will be convinient to denote $d(I) := D_I(S^*_I)$ as the maximum deficit for capacity vector $\mathbf{1} - \mathbf{1}_I$. deficit. Next, we prove the following:

**Lemma 17.** *Let $F : 2^{[n]} \to \mathbb{Z}_{\geq 0}$ be a monotone integral supermodular function satisfying $F(S) \leq |S|$ for all $S \subseteq [n]$. Then for any $I \subseteq [n]$ where the capacity vector $\mathbf{1} - \mathbf{1}_I$ is infeasible one has $|S_I^*| \geq F(S_I^*) \geq |S_I^*| - |I|$.*

*Proof.* The first inequality $F(S_I^*) \leq |S_I^*|$ is by assumption. Note that the capacities $\mathbf{1} - \mathbf{1}_I$ are infeasible if and only if $D_I(S_I^*) > 0$ since $S_I^*$ is the maximizer of $D_I$. Now assume that this is indeed the case. Then

$$0 < D_I(S_I^*) = \underbrace{F(S_I^*)}_{\leq |S_I^*|} - \underbrace{|S_I^* \setminus I|}_{\geq |S_I^*| - |I|} \leq F(S_I^*) - (|S_I^*| - |I|)$$

This can be rearranged to the desired $F(S_I^*) \geq |S_I^*| - |I|$. □

We also prove a variant saying that the optimizer $S_I^*$ always has a small slack:

**Lemma 18.** *For any $I \subseteq [n]$ one has $0 \leq |S_I^*| - F(S_I^*) \leq |I|$.*

*Proof.* We have

$$|F(S_I^*)| - |S_I^*| + |I| \geq |F(S_I^*)| - \underbrace{|S_I^* \setminus I|}_{\geq |S_I^*| - |I|} = d(I) \geq D_I(\emptyset) = \underbrace{F(\emptyset)}_{=0} - 0 = 0$$

Then rearranging gives the claim. □

It is not hard to argue that $I$ is included in $S_I^*$:

**Lemma 19.** *Let $I \subseteq [n]$. Then $I \subseteq S_I^*$.*

*Proof.* Assume for the sake of contradiction that there is an index $j^* \in I \setminus S_I^*$. Then

$$D_I(S_I^* \cup \{j^*\}) = \underbrace{F(S_I^* \cup \{j^*\})}_{\geq F(S_I^*)} - \underbrace{|(S_I^* \cup \{j^*\}) \setminus I|}_{= |S_I^*/I|} \geq D_I(S_I^*) = d(I)$$

which is a contradiction to the maximality of $S_I^*$. □

In fact, we can prove the structural property that removing capacity will increase $S_I^*$ monotonically.

**Lemma 20.** *Let $I \subseteq [n]$ and $j^* \in [n] \setminus I$, then $S_I^* \subseteq S_{I\cup\{j^*\}}^*$. Moreover, we have equality if and only if $d(I \cup \{j^*)\} = d(I) + 1$.*

*Proof.* Note that for any $S \subseteq [n]$ one has $D_{I\cup\{j^*\}}(S) - D_I(S) = \mathbf{1}_{j^*\in S} \in \{0, 1\}$. As $S_I^*$ is the maximizer of $D_I$, we know that $d(I \cup \{j^*\}) \geq D_{I\cup\{j^*\}}(S_I^*) \geq D_I(S_I^*) = d(I)$.

On the other hand also $S_I^*$ is a maximizer of $D_I$ and so $d(I \cup \{j^*\}) \leq D_I(S_{I\cup\{j^*\}}^*) + 1 \leq D_I(S_I^*) + 1 = d(I) + 1$. So really we only have two cases:

- **Case:** $d(I \cup \{j^*\}) = d(I) + 1$. First, note that this means $j^* \in S_{I\cup\{j^*\}}^*$. In this case $S_{I\cup\{j^*\}}^*$ is also a maximizer for $D_I$. By maximality we definitely have $S_{I\cup\{j^*\}}^* \subseteq S_I^*$. Moreover we claim that this actually has to be an equality. Since $j^* \in S_I^*$ we know that $D_{I\cup\{j^*\}}(S_I^*) = d(I) + 1$ as well. That means $S_I^*$ is a maximizer for $D_{I\cup\{j^*\}}$ and also $S_I^* \subseteq S_{I\cup\{j^*\}}^*$. Hence $S_I^* = S_{I\cup\{j^*\}}^*$.

- **Case:** $d(I \cup \{j^*\}) = d(I)$. In this case also $S_I^*$ is a maximizer for $D_{I\cup\{j^*\}}$ and so $S_I^* \subseteq S_{I\cup\{j^*\}}^*$. We also know that $j^* \notin S_I^*$ and $j^* \in S_{I\cup\{j^*\}}^*$ (by Lemma 19) since otherwise we could not be in this case. Then $S_I^* \subset S_{I\cup\{j^*\}}^*$.

□

**Lemma 21.** *Let $I \subseteq [n]$ and $j^* \in [n] \setminus S_I^*$. Then the following holds:*

*(a) One has $d(I \cup \{j^*\}) = d(I)$.*

*(b) One has $\{j^*\} \cup S_I^* \subseteq S_{I\cup\{j^*\}}^*$.*

*Proof.* We know by Lemma 20 that $S_{I\cup\{j^*\}}^* \supseteq S_I^*$ and we also know that $j^* \in S_{I\cup\{j^*\}}^*$. Then in particular we must have a case of strict inclusion, i.e. $S_{I\cup\{j^*\}}^* \supset S_I^*$. Then from the "moreover" part of Lemma 20 we know that $d(I) = d(I \cup \{j^*\})$. □

### *3.5.3 Any set $S$ is contained in some maximizer $S_I^*$*

The crucial point is proving that any set $S$ is fully contained in $S_I^*$ where $I$ is small.

**Lemma 22.** *Let $F : 2^{[n]} \to \mathbb{Z}_{\geq 0}$ be an integral monotone supermodular function with $F(S) \leq |S|$ for all $S \subseteq [n]$. Consider any set $S \subseteq [n]$ and set $k := |S| - F(S) \geq 0$. Then one can select a set $I \subseteq S$ with $|I| \leq k$ so that $S \subseteq S^*_I$.*

*Proof.* Fix a set $S \subseteq [n]$. We construct a set $I \subseteq S$, iteratively starting at $I := \emptyset$. In each iteration we select an index $j^* \in S \setminus S^*_I$ and add it to $I$. Note that by Lemma 21 we have $d(I \cup \{j^*\}) = d(I)$. We terminate when there is no such index anymore. When we terminate, we have $I \subseteq S \subseteq S^*_I$ by construction and by a simple induction argument we know that $d(I) = d(\emptyset)$. Then

$$|I| - k = \underbrace{F(S) - |S|}_{=-k} + |I| \overset{I \subseteq S}{=} F(S) - |S \setminus I| = D_I(S) \overset{\text{maximality}}{\leq} d(I) = d(\emptyset) \leq 0$$

and so $|I| \leq k$. □

### *3.6 Probabilistic analysis*

#### *3.6.1 Bounding the probability to violate a fixed set*

First we discuss the probability that a fixed set $S$ is violated. Recall the following standard tool:

**Theorem 23** (Bernstein Inequality)**.** *Let $X_1, \ldots, X_N$ be independent random variables with $X_i \in [a_i, a_i + M]$ for some $a_i \in \mathbb{R}$ and $M > 0$. Set $Z := \sum_{i=1}^N X_i$. Then for any $t > 0$ one has*

$$\Pr[|Z - \mathbb{E}[Z]| \geq t] \leq 2 \exp\left( - \frac{\frac{1}{2}t^2}{\sum_{i=1}^N \mathsf{Var}[X_i] + \frac{1}{3}Mt} \right) \leq 2 \exp\Big( - \min\Big\{ \frac{t^2}{4\mathsf{Var}[Z]}, \frac{t}{2M} \Big\} \Big)$$

We use this to prove:

**Lemma 24.** *Let $F$ be as described in Theorem 15. Consider a set $S \subseteq [n]$ and let $k := |S| - F(S) \geq 0$. Then*

$$\Pr\Big[ \sum_{j \in S} (1 - X_j) > k/2 \Big] \leq 2 \exp(-\alpha k/8)$$

*Proof.* Note that the assumption on $F$ in Theorem 15 gives that $F(S) \leq |S| - \alpha \sum_{j \in S} p_j$ which can be rearranged to $\alpha \sum_{j \in S} p_j \leq k$. Let us denote $Y_j := 1 - X_j$ and $Y := \sum_{j \in S} Y_j$. Then $Y_j \in \{0, \frac{1}{\alpha}\}$ with $\Pr[Y_j = \frac{1}{\alpha}] = p_j$. Note that $\mathbb{E}[Y] = \frac{1}{\alpha} \sum_{j \in S} p_j \leq \frac{k}{\alpha^2}$. Moreover we can verify that the variance is

$$\begin{aligned} \mathsf{Var}[Y_j] &= \mathbb{E}\left[(Y_j - \mathbb{E}[Y_j])^2\right] = \underbrace{\Pr[Y_j = 0]}_{\leq 1} \cdot \underbrace{(0 - \mathbb{E}[Y_j])^2}_{\leq (p_j/\alpha)^2} + \underbrace{\Pr\left[Y_j = \frac{1}{\alpha}\right]}_{= p_j} \cdot \underbrace{\left(\frac{1}{\alpha} - \mathbb{E}[Y_j]\right)^2}_{\leq (1/\alpha)^2} \\ &\leq \left(\frac{p_j}{\alpha}\right)^2 + \frac{p_j}{\alpha^2} \leq \frac{2p_j}{\alpha^2} \end{aligned}$$

and so by independence $\mathsf{Var}[Y] \leq \frac{2}{\alpha^2} \sum_{j \in S} p_j \leq \frac{2k}{\alpha^3}$. The goal is to prove that $\Pr[Y > k/2] \leq n^{-100k}$. And indeed, applying Bernstein's Inequality with $t := k/4$ and $M := \frac{1}{\alpha}$ gives

$$\begin{aligned} \Pr\left[Y \geq \frac{k}{2}\right] &\leq \Pr\left[Y \geq \mathbb{E}[Y] + \frac{k}{4}\right] \overset{\text{Bernstein}}{\leq} 2\exp\Big(-\min\Big\{\frac{t^2}{4\mathsf{Var}[Y]}, \frac{t}{2M}\Big\}\Big) \\ &= 2\exp\Big(-\min\Big\{\frac{(k/4)^2}{(2k/\alpha^3)}, \frac{k/4}{2/\alpha}\Big\}\Big) = 2\exp(-\alpha k/8) \end{aligned}$$

□

### *3.6.2 The union bound over all sets $S_I^*$*

Next, we prove that with high probability in each set of the form $S_I^*$, the amount of "lost" capacity $\sum_{j \in S_I^*}(1 - X_j)$ is at most $|I|/2$. For an event $\mathcal{E}$ we will use $\bar{\mathcal{E}}$ to denote the complementary event.

**Lemma 25.** *Consider the event $\mathcal{E} := \bigwedge_{k=1}^{n} \mathcal{E}_k$ where*

$$\mathcal{E}_k := \bigwedge_{I \in \binom{[n]}{k}} \Big(\sum_{j \in S_I^*}(1 - X_j) \leq k/2\Big)$$

*If $\alpha = C\log(n)$ with a large enough constant $C > 0$, then $\Pr[\mathcal{E}] \geq 1 - \frac{1}{n^{100}}$.*

*Proof.* First, fix a value of $k$. From Lemma 18 we know that for any index set $I \in \binom{[n]}{k}$ one

has $|S_I^*| - |F(S_I^*)| \le |I| = k$. Then applying the union bound with Lemma 24 gives

$$\begin{aligned}\Pr[\bar{\mathcal{E}}_k] &= \Pr\Big[\bigvee_{I\in\binom{[n]}{k}} \Big(\sum_{j\in S_I^*}(1-X_j) > \frac{k}{2}\Big)\Big] \le \sum_{I\in\binom{[n]}{k}} \Pr\Big[\sum_{j\in S_I^*}(1-X_j) > \frac{k}{2}\Big] \\ &\overset{\text{Lem 24}}{\le} n^k \cdot 2\exp(-\alpha k/8) \le n^{-1000k}\end{aligned}$$

if $\alpha = C\log(n)$ with $C > 0$ large enough. Then applying again the union bound over all $k$ we get

$$\Pr[\bar{\mathcal{E}}] \le \Pr\Big[\bigvee_{k=1,\dots,n} \bar{\mathcal{E}}_k\Big] \le \sum_{k=1}^{n} \Pr[\bar{\mathcal{E}}_k] \le n^{-100}$$

□

### *3.6.3 The conclusion*

Now we can wrap up the proof of the main result.

**Theorem 26.** *Assume event $\mathcal{E}$ from Lemma 25 happens. Then $F(S) \le \sum_{j\in S} X_j$ for all $S \subseteq [n]$.*

*Proof.* Fix a set $S \subseteq [n]$ that we want to verify and set $k := |S| - F(S) \ge 0$. Then by Lemma 22, there is an index set $I$ with $|I| = k$ so that $I \subseteq S \subseteq S_I^*$. Assuming event $\mathcal{E}$ happens, we know that $\sum_{j\in S_I^*}(1-X_j) \le k/2$. Then by monotonicity $\sum_{j\in S}(1-X_j) \le k/2$. Finally $F(S) = |S| - k \le |S| - \frac{k}{2} \le \sum_{j\in S} X_j$ as claimed. □

Since $\Pr[\mathcal{E}] \ge 1 - \frac{1}{n^{100}}$ and this event implies that $F(S) \le \sum_{j\in S} X_j$, overall Theorem 15 is proven.

# Chapter 4

# Scheduling with Communication delays on Identical Machines

### *4.1 Introduction*

Scheduling jobs with precedence constraints is a fundamental problem in approximation algorithms and combinatorial optimization. In this problem we are given $m$ identical machines and a set $J$ of $n$ jobs, where each job $j$ has a processing length $p_j \in \mathbb{Z}_+$. The jobs have precedence 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. The goal is to find a schedule of jobs that minimizes *makespan*, which is the completion time of the last job. This problem is denoted[1] by $\mathsf{P} \mid \mathsf{prec} \mid C_{\mathsf{max}}$.

In a seminal result from 1966, Graham [Gra66b] showed that the greedy list scheduling algorithm achieves a $\left(2 - \frac{1}{m}\right)$-approximation. By now, our understanding of the approximability of this basic problem is almost complete: it had been known since the late '70s,

[1]Throughout this chapter and the next, we use the standard scheduling three-field notation [GLLK79, VLL90]. The respective fields denote: **(1) number of identical machines:** $\mathsf{P}\infty$: unlimited; $\mathsf{P}$: number $m$ of machines given as input; $\mathsf{P}m$: constant number $m$ of machines, **(2) job properties:** $\mathsf{prec}$: precedence constraints; $p_j = 1$: unit-size jobs; $c$: communication delays of length $c$ (can be $c_{jk}$ if dependent on jobs $j \prec k$); $c-\mathsf{intervals}$: see Section 4.3; $\mathsf{dup}$: allowed duplication of jobs, **(3) objective:** $C_{\mathsf{max}}$: minimize makespan; $\sum w_j C_j$: minimize weighted sum of completion times.

due to a result by Lenstra and Rinnooy Kan [LRK78], that it is **NP**-hard to obtain better than 4/3-approximation, and in 2010 Svensson [Sve10] showed that, assuming a variant of the Unique Games Conjecture [BK10], it is **NP**-hard to get a $(2-\varepsilon)$-approximation for any $\varepsilon > 0$.

The above precedence-constrained scheduling problem models the task of distributing workloads onto multiple processors or servers, which is ubiquitous in computing. This basic setting takes the dependencies between work units into account, but not the data transfer costs between machines, which is critical in applications. A precedence constraint $j \prec j'$ typically implies that the input to $j'$ depends on the output of $j$. In many real-world scenarios, especially in the context of scheduling in data centers, if $j$ and $j'$ are executed on different machines, then the *communication delay* due to transferring this output to the other machine cannot be ignored. This is an active area of research in applied data center scheduling literature, where several new abstractions have been proposed to deal with communication delays [CZM+11, GFC+12, HCG12, SZA+18, ZZC+12, ZCB+15, LYZ+16]. Another timely example is found in the parallelization of Deep Neural Network training (the machines being accelerator devices such as GPUs, TPUs, or FPGAs). There, when training the network on one sample/minibatch per device in parallel, the communication costs incurred by synchronizing the weight updates in fact dominate the overall running time [NHP+19]. Taking these costs into account, it turns out that it is better to split the network onto multiple devices, forming a "model-parallel" computation pipeline [HCB+19]. In the resulting *device placement* problem, the optimal split crucially depends on the communication costs between dependent layers/operators.

A classic model that captures the effect of data transfer latency on scheduling decisions is the problem of *scheduling jobs with precedence and communication delay constraints*, introduced by Rayward-Smith [RS87] and Papadimitriou and Yannakakis [PY90]. The setting, denoted by $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$, is similar to the makespan minimization problem described earlier, except for one crucial difference. Here we are given a *communication delay parame-*

*ter* $c \in \mathbb{Z}_{\geq 0}$, and the output schedule must satisfy the property that 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. In a closely related problem, denoted by $\mathsf{P}\infty \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$, a schedule can use as many machines as desired. The goal is to schedule jobs *non-preemptively* so as to minimize the makespan. In a non-preemptive schedule, each job $j$ needs to be assigned to a single machine and executed during $p_j$ consecutive timeslots. The problems $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$ and $\mathsf{P}\infty \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$ are the focus of this chapter.

Despite its theoretical significance and practical relevance, very little is known about the communication delay setting. A direct application of Graham's [Gra66b] list scheduling algorithm yields a $(c+2)$-approximation, and no better algorithm is known for the problem. Over the years, the problem has attracted significant attention, but all known results, which we discuss below in Section 4.1.4, concern special settings, small communication delays, or hardness of approximation. To put this in perspective, we note that the current best algorithm for general $c$ [GKMP08], which achieves an approximation factor of $2/3 \cdot (c+1)$, only marginally improves on Graham's algorithm while requiring the additional assumptions that the number of machines is unbounded and $p_j = 1$. This is in sharp contrast to the basic problem $\mathsf{P} \mid \mathsf{prec} \mid C_{\mathsf{max}}$ (which would correspond to the case $c = 0$), where the approximability of the problem is completely settled under a variant of the Unique Games Conjecture. This situation hints that incorporating communication delays in scheduling decisions requires fundamentally new algorithmic ideas compared to the no-delay setting. Schuurman and Woeginger [SW99] placed the quest for getting better algorithms to the problem in their influential list of top-10 open problems in scheduling theory. In a recent MAPSP 2017 survey talk, Bansal [Ban17] highlighted the lack of progress on this model, describing it as "not understood at all; almost completely open", and suggested that this is due to the lack of promising LP/SDP relaxations.

### *4.1.1 Our Contributions*

The main result of our paper [DKR$^+$20] is the following:

**Theorem 27.** *There is a randomized $O(\log c \cdot \log m)$-approximation algorithm for $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$ with expected polynomial running time, where $c, p_j \in \mathbb{N}$.*

In any non-preemptive schedule the number $m$ of machines is at most the number $n$ of jobs, so for the easier $\mathsf{P}\infty$ version of the problem, the above theorem implies the following:

**Corollary 28.** *There is a randomized $O(\log c \cdot \log n)$-approximation algorithm for $\mathsf{P}\infty \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$ with expected polynomial running time, where $c, p_j \in \mathbb{N}$.*

For both problems one can replace either $c$ or $m$ by $n$, yielding an $O(\log^2 n)$-approximation algorithm. Our results make substantial progress towards resolving one of the questions in "Open Problem 3" in the survey of Schuurman and Woeginger [SW99], which asks whether a constant-factor approximation algorithm exists for $\mathsf{P}\infty \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$.

Our approach is based on a Sherali-Adams lift of a time-indexed linear programming relaxation for the problem, followed by a randomized clustering of the semimetric space induced by this lift. It is possible to write a more compact LP for the problem for which our algorithm and analysis still work. However, as we will show in Section 4.5, this compact LP has integrality gap $\Omega(\sqrt{\log n})$. To our knowledge, this is the first instance of a multiple-machine scheduling problem being viewed via the lens of metric space clustering. We believe that our framework is fairly general and should extend to other problems involving scheduling with communication delays. To demonstrate the broader applicability of our approach, we also consider the objective of minimizing the weighted sum of completion times. Here each job $j$ has a weight $w_j$, and the goal is to minimize $\sum_j w_j C_j$, where $C_j$ is the completion time of $j$.

**Theorem 29.** *There is a randomized $O(\log c \cdot \log n)$-approximation algorithm for $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c \mid \sum_j w_j C_j$ with expected polynomial running time, where $c \in \mathbb{N}$.*

No non-trivial approximation ratio was known for this problem prior to our work.

### *4.1.2 Independent work of Maiti* et al

In a parallel and independent work, Maiti et al. [MRS+20] developed an $O(\log^2 n \log^2 m \log c / \log\log c)$-approximation algorithm for the makespan objective function on *related machines* ($Q \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$). Interestingly, they obtained the results using completely different techniques compared to ours. While our results are 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. Maiti et al. also prove an integrality gap for the LP they consider, which is different than ours. However, after seeing their result, we found that their integrality gap instance works for a compact form of our LP as well, which we prove in Section 4.5.

### *4.1.3 Our Techniques*

As we alluded earlier, there is a lack of combinatorial lower bounds for scheduling with communication delays. For example, consider Graham's list scheduling algorithm, which greedily processes jobs on $m$ machines as soon as they become available. One can revisit the analysis of Graham [Gra66b] and show that there exists a chain $Q$ of dependent jobs such that the makespan achieved by list scheduling is bounded by

$$\frac{1}{m}\sum_{j\in J} p_j + \sum_{j\in Q} p_j + c\cdot(|Q|-1).$$

The first two terms are each lower bounds on the optimum — the 3rd term is not. In particular, it is unclear how to certify that the optimal makespan is high because of the

communication delays. However, this argument suffices for a $(c+2)$-approximation, since $p_j \geq 1$ for all $j \in J$.

As pointed out by Bansal [Ban17], there is no known promising LP relaxation. To understand the issue let us consider the special case $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c \mid C_{\mathsf{max}}$. Extending, for example, the LP of Munier and König [MK97], one might choose variables $C_j$ as completion times, as well as decision variables $x_{j_1,j_2}$ denoting whether $j_2$ is executed in the time window $[C_{j_1}, C_{j_1} + c)$ on the same machine as $j_1$. Then we can try to enforce communication delays by requiring that $C_{j_2} \geq C_{j_1} + 1 + (c-1) \cdot (1 - x_{j_1,j_2})$ for $j_1 \prec j_2$. Further, we enforce load constraints $\sum_{j_1 \in J} x_{j_1,j_2} \leq c$ for $j_2 \in J$ and $\sum_{j_2 \in J} x_{j_1,j_2} \leq c$ for $j_1 \in J$. To see why this LP fails, note that in any instance where the maximum dependence degree is bounded by $c$, one could simply set $x_{j_1,j_2} = 1$ and completely avoid paying any communication delay. Moreover, this problem seems to persist when moving to more complicated LPs that incorporate indices for time and machines.

A convenient observation is that, in exchange for a constant-factor loss in the approximation guarantee, it suffices to find an assignment of jobs to length-$c$ intervals such that dependent jobs scheduled in the same length-$c$ interval must be assigned to the same machine. (The latter condition will be enough to satisfy the communication delay constraints as, intuitively, between every two length-$c$ intervals we will insert an empty one.) In order to obtain a stronger LP relaxation, we consider an $O(1)$-*round Sherali-Adams lift* of an inital LP with indices for time and machines. From the lifted LP, we extract a *distance function* $d : J \times J \to [0,1]$ which satisfies the following properties:

(i) The function $d$ is a *semimetric*.

(ii) $C_{j_1} + d(j_1, j_2) \leq C_{j_2}$ for $j_1 \prec j_2$.

(iii) Any set $U \subseteq J$ with a diameter of at most $\frac{1}{2}$ w.r.t. $d$, satisfies $|U| \leq 2c$.

Here we have changed the interpretation of $C_j$ to the *index* of the length-$c$ interval in which $j$ will be processed. Intuitively, $d(j_1, j_2)$ can be understood as the probability that jobs $j_1, j_2$

are *not* being scheduled within the same length-$c$ interval on the same machine. To see why a constant number of Sherali-Adams rounds are helpful, observe that the triangle inequality behind $(i)$ is really a property depending only on *triples* $\{j_1, j_2, j_3\}$ of jobs and an $O(1)$-round Sherali-Adams lift would be locally consistent for every triple of variables.

We will now outline how to round such an LP solution. For jobs whose LP completion times are sufficiently different, say $C_{j_1} + \Theta(\frac{1}{\log(n)}) \leq C_{j_2}$, we can afford to deterministically schedule $j_1$ and $j_2$ at least $c$ time units apart while only paying an $O(\log n)$-factor more than the LP. Hence the critical case is to sequence a set of jobs $J^* = \{j \in J \mid C^* \leq C_j \leq C^* + \Theta(\frac{1}{\log(n)})\}$ whose LP completion times are very close to each other. Note that by property $(ii)$, we know that any dependent jobs $j_1, j_2 \in J^*$ must have $d(j_1, j_2) \leq \Theta(\frac{1}{\log(n)})$. As $d$ is a semimetric, we can make use of the rich toolset from the theory of metric spaces. In particular, we use an algorithm by Calinescu, Karloff and Rabani [CKR04]: For a parameter $\Delta > 0$, one can partition a semimetric space into *random clusters* so that the diameter of every cluster is bounded by $\Delta$ and each $\delta$-neighborhood around a node is separated, meaning contains jobs assigned to different clusters, with probability at most $O(\log(n)) \cdot \frac{\delta}{\Delta}$. Setting $\delta := \Theta(\frac{1}{\log(n)})$ and $\Delta := \Theta(1)$ one can then show that a fixed job $j \in J^*$ will be in the same cluster as *all* its ancestors in $J^*$ with probability at least $\frac{1}{2}$, while all clusters have diameter at most $\frac{1}{2}$. By $(iii)$, each cluster will contain at most $2c$ many (unit-length) jobs, and consequently we can schedule all the clusters in parallel, where we drop any job that got separated from any ancestor. Repeating the sampling $O(\log n)$ times then schedules all jobs in $J^*$. This reasoning results in an $O(\log^2 n)$-approximation for this problem, which we call $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c\text{—}\mathsf{intervals} \mid C_{\mathsf{max}}$. With a bit of care the approximation factor can be improved to $O(\log c \cdot \log m)$.

Finally, the promised $O(\log c \cdot \log m)$-approximation for the more general problem $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$ follows from a reduction to the described special case $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c\text{—}\mathsf{intervals} \mid C_{\mathsf{max}}$.

### *4.1.4 History of the Problem*

Precedence-constrained scheduling problems of minimizing the makespan and sum of completion times objectives have been extensively studied for many decades in various settings. We refer the reader to [Pin18, LLKS93, PST04, AMMS08, Sve09] for more details. Below, we only discuss results directly related to the communication delay problem in the offline setting.

**Approximation algorithms.** As mentioned earlier, Graham's [Gra66b] list scheduling algorithm yields a $(c+2)$-approximation for $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$, and a $(c+1)$-approximation for the $\mathsf{P}\infty$ variant. For unit-size jobs and $c \geq 2$, Giroudeau, König, Moulai and Palaysi [GKMP08] improved the latter ($\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c \geq 2 \mid C_{\mathsf{max}}$) to a $\frac{2}{3}(c+1)$-approximation. For unit-size jobs and $c = 1$, Munier and König [MK97] obtained a 4/3-approximation via LP rounding ($\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c = 1 \mid C_{\mathsf{max}}$); for the $\mathsf{P}$ variant, Hanen and Munier [MH01] gave an easy reduction from the $\mathsf{P}\infty$ variant that loses an additive term of 1 in the approximation ratio, thus yielding a 7/3-approximation. Thurimella and Yesha [Thu92] gave a reduction that, given an $\alpha$-approximation algorithm for $\mathsf{P}\infty \mid \mathsf{prec}, c, p_j = 1 \mid C_{\mathsf{max}}$, would yield a $(1+2\alpha)$-approximation algorithm for $\mathsf{P} \mid \mathsf{prec}, c, p_j = 1 \mid C_{\mathsf{max}}$.

For a constant number of machines, a hierarchy-based approach of Levey and Rothvoss [LR16] for the no-delay setting ($\mathsf{P}m \mid \mathsf{prec}, p_j = 1 \mid C_{\mathsf{max}}$) was generalized by Kulkarni, Li, Tarnawski and Ye [KLTY20] to allow for communication delays that are also bounded by a constant. For any $\varepsilon > 0$ and $\hat{c} \in \mathbb{Z}_{\geq 0}$, they give a nearly quasi-polynomial-time $(1+\varepsilon)$-approximation algorithm for $\mathsf{P}m \mid \mathsf{prec}, p_j = 1, c_{jk} \leq \hat{c} \mid C_{\mathsf{max}}$. The result also applies to arbitrary job sizes, under the assumption that preemption of jobs is allowed, but migration is not.

**Hardness.** Hoogeveen, Lenstra and Veltman [HLV94] showed that even the special case $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c = 1 \mid C_{\mathsf{max}}$ is **NP**-hard to approximate to a factor better than 7/6. For the case with bounded number of machines (the $\mathsf{P}$ variant) they show 5/4-hardness. These two results can be generalized for $c \geq 2$ to $(1 + 1/(c+4))$-hardness [GKMP08] and

$(1 + 1/(c + 3))$-hardness [BGK96], respectively.[2]

**Duplication model.** The communication delay problem has also been studied (to a lesser extent) in a setting where jobs can be duplicated (replicated), i.e., executed on more than one machine, in order to avoid communication delays. This assumption seems to significantly simplify the problem, especially when we are also given an unbounded number of machines: already in 1990, Papadimitriou and Yannakakis [PY90] gave a rather simple 2-approximation algorithm for $\mathsf{P}\infty \mid \mathsf{prec}, p_j, c_{jk}, \mathsf{dup} \mid C_{\mathsf{max}}$. Observe that this result holds even when communication delays are unrelated (they depend on the pair of jobs). The only non-trivial approximation algorithm for arbitrary $c$ and a bounded number of machines is due to Lepere and Rapine [LR02], who gave an asymptotic $O(\log c/ \log \log c)$-approximation for $\mathsf{P} \mid \mathsf{prec}, p_j = 1, c, \mathsf{dup} \mid C_{\mathsf{max}}$. On the hardness side, Papadimitriou and Yannakakis [PY90] showed **NP**-hardness of $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c, \mathsf{dup} \mid C_{\mathsf{max}}$ (using a large delay $c = \Theta(n^{2/3})$).

Many further references can be found in [VLL90, GKMP08, Dro09, GK07, CC91, JKS93, MH97].

## *4.2 Preliminaries*

### *4.2.1 The Sherali-Adams Hierarchy for LPs with Assignment Constraints*

In this section, we review the *Sherali-Adams hierarchy* which provides an automatic strengthening of linear relaxations for 0/1 optimization problems. The authorative reference is certainly Laurent [Lau03], and we adapt the notation from Friggstad et al. [FKK+14]. Consider a set of variable indices $[n] = \{1, \dots, n\}$ and let $U_1, \dots, U_N \subseteq [n]$ be subsets of variable indices. We consider a polytope

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

[2] Papadimitriou and Yannakakis [PY90] claim a 2-hardness for $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c \mid C_{\mathsf{max}}$, but give no proof. Schuurman and Woeginger [SW99] remark that "it would be nice to have a proof for this claim".

which we also write in a more compact form as $K = \{x \in \mathbb{R}^n \mid Ax \geq b\}$ with $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. We note that we included explicitly the "box constraints" $0 \leq x_i \leq 1$ for all variables $i$. Moreover, the constraint matrix contains *assignment constraints* of the form $\sum_{i \in U_k} x_i = 1$. This is the aspect that is non-standard in our presentation.

The general goal is to obtain a strong relaxation for the integer hull $\text{conv}(K \cap \{0,1\}^n)$. Observe that any point $x \in \text{conv}(K \cap \{0,1\}^n)$ can be interpreted as a *probability distribution* $X$ over points $K \cap \{0,1\}^n$. We know that any distribution can be described by the $2^n$ many values $y_I = \Pr[\bigwedge_{i \in I}(X_i = 1)]$ for $I \subseteq [n]$ — in fact, the probability of any other event can be reconstructed using the *inclusion-exclusion formula*, for example $\Pr[X_1 = 1 \text{ and } X_2 = 0] = y_{\{1\}} - y_{\{1,2\}}$. While this is an exact approach, it is also an inefficient one. In order to obtain a polynomial-size LP, we only work with variables $y_I$ where $|I| \leq O(1)$. Hence, for $r \geq 0$, we denote $\mathcal{P}_r([n]) := \{S \subseteq [n] \mid |S| \leq r\}$ as all the index sets of size at most $r$.

**Definition 4.** 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 $|I| + |J| \leq r$.

The parameter $r$ in the definition is usually called the *rank* or *number of rounds* of the Sherali-Adams lift. It might be helpful for the reader to verify that for $I = J = \emptyset$, the constraint simplifies to $\sum_{i=1}^{n} A_{\ell,i} y_{\{i\}} \geq b_\ell y_\emptyset = b_\ell$, which implies that $(y_{\{1\}}, \ldots, y_{\{n\}}) \in K$. Moreover it is instructive to verify that for any feasible integral solution $x \in K \cap \{0,1\}^n$ one can set $y_I := \prod_{i \in I} x_i$ to obtain a vector $y \in SA_r(K)$.

**Theorem 30** (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* $k \in [N]$ *one has* $y_I = \sum_{i\in U_k} y_{I\cup\{i\}}$.

(f) *Take* $H \subseteq [N]$ *with* $|H| \leq r$ *and set* $J := \bigcup_{k\in H} U_k$. *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* $I \in \mathcal{P}_{r+1-|H|}([n])$.

*Proof.* For (a)-(d), we refer to the extensive coverage in Laurent [Lau03]. We prove (e) and (f) which are non-standard and custom-tailored to LPs with assignment constraints:

(e) Fix $I \subseteq [n]$ with $|I| \leq r$. We apply (d) to obtain a distribution over $\tilde{y}$ with $\tilde{y} \in SA_{r-|I|}(K)$ so that $\tilde{y}_i \in \{0,1\}$ for $i \in I$. Then

$$\sum_{i\in U_k} y_{I\cup\{i\}} \overset{\text{linearity}}{=} \mathbb{E}\Big[\sum_{i\in U_k} \tilde{y}_{I\cup\{i\}}\Big] \overset{(c)}{=} \mathbb{E}\Big[\tilde{y}_I \cdot \underbrace{\sum_{i\in U_k} \tilde{y}_i}_{=1}\Big] = \mathbb{E}[\tilde{y}_I] = y_I.$$

Here we apply (*c*) for index sets $I \cup \{i\}$ where variables in $J := I$ have been made integral. Note that indeed $|I \cup (I \cup \{i\})| \leq r+1$ as required.

(f) By an inductive argument it suffices to consider the case of $|H| = 1$. Let $H = \{k\}$ and set $U := U_k$, i.e. the constraints for polytope $P$ contain the assignment constraint $\sum_{i\in U} x_i = 1$ and we want to make all variables in $U$ integral while only losing a *single* round in the hierarchy. Abbreviate $U^+ := \{i \in U \mid y_{\{i\}} > 0\}$. For $i \in U^+$, define $y^{(i)} \in \mathbb{R}^{\mathcal{P}_r([n])}$ to be the vector with $y^{(i)}_I := \frac{y_{I\cup\{i\}}}{y_i}$. By (a) we know that $y^{(i)} \in SA_{r-1}(K)$. Moreover $y^{(i)}_{\{i\}} = \frac{y_{\{i\}}}{y_{\{i\}}} = 1$. Then the assignment constraint of the LP forces that $y^{(i)}_{\{i'\}} = 0$ for $i' \in U \setminus \{i\}$. Now we define a probability distribution over vectors $\tilde{y}$ as follows: for

$i \in U^+$, with probability $y_i$ we set $\tilde{y} := y^{(i)}$. Then (i) and (ii) hold for $\tilde{y}$ as discussed. Property (iii) follows from

$$\mathbb{E}[\tilde{y}_I] = \sum_{i \in U^+} y_i y_I^{(i)} = \sum_{i \in U^+} y_i \frac{y_{I \cup \{i\}}}{y_i} = \sum_{i \in U^+} y_{I \cup \{i\}} \overset{(b)}{=} \sum_{i \in U} y_{I \cup \{i\}} \overset{(e)}{=} y_I$$

□

It is known that Theorem 30.(f) holds in a stronger form for the SDP-based *Lasserre hierarchy*. Karlin, Mathieu and Nguyen [KMN11] proved a result that can be paraphrased as follows: *if one has any set $J \subseteq [n]$ of variables with the property that there is no LP solution with more than $k$ ones in $J$, then one can make all variables of $J$ integral while losing only $k$ rounds*. Interestingly, Karlin, Mathieu and Nguyen prove that this is completely false for Sherali-Adams. In particular, for a Knapsack instance with unit size items and capacity $2-\varepsilon$, the integrality gap is still $2 - 2\varepsilon$ after $\Theta_\varepsilon(n)$ rounds of Sherali-Adams. In a different setting, Friggstad et al. [FKK$^+$14] realized that given a "tree constraint", a Sherali-Adams lift can provide the same guarantees that Rothvoss [Rot11] derived from Lasserre. While Friggstad et al. did not state their insight in the generality that we need here, our Lemma 30.(e)+(f) are inspired by their work.

### *4.2.2 Semimetric Spaces*

A *semimetric 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 *semimetric*, i.e.

- $d(u, u) = 0$ for all $u \in U$.
- Symmetry: $d(u, v) = d(v, u)$ for all $u, v \in V$.
- Triangle inequality: $d(u, v) + d(v, w) \geq d(u, w)$ for all $u, v, w \in V$.

Recall that the more common notion is that of a *metric*, which additionally requires 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 partition $V = V_1 \dot\cup \ldots \dot\cup V_k$ such that the diameter of every cluster $V_i$ is bounded by some parameter $\Delta$. We denote $d(w, U) := \min\{d(w, u) : u \in U\}$ as the distance to the set $U$. Moreover, for $r \geq 0$ and $U \subseteq V$, let $N(U, r) := \{v \in V \mid d(v, U) \leq r\}$ be the *distance $r$-neighborhood* of $U$.

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

| CKR Clustering algorithm |
|---|
| **Input:** Semimetric space $(V, d)$ with $V = \{v_1, \ldots, v_n\}$, parameter $\Delta > 0$ |
| **Output:** Clustering $V = V_1 \dot\cup \ldots \dot\cup V_k$ for some $k$. |
| (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 $c_1, \ldots, c_k \in V$ and return clusters $V_i := \sigma^{-1}(c_i)$ for $i = 1, \ldots, k$ |

Note that the algorithm has two sources of randomness: it picks a random parameter $\beta$, and independently it picks a random ordering $\pi$. Here the ordering is to be understood such that element $v_\ell$ 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 a short 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 $0 \leq t < \frac{\Delta}{8}$. 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). Instead of the algorithm by Calinescu, Karloff and Rabani [CKR04], one could also cluster using the techniques of Leighton and Rao [LR99] or those of Garg, Vazirani and Yannakakis [GVY93].

We state the formal claim in a form that will be convenient for us. The proof will be

available in the next subsection.

**Theorem 31** (Analysis of CKR)**.** *Let $V = V_1 \dot\cup \ldots \dot\cup V_k$ be the random partition of the CKR algorithm. The following holds:*

*(a) The blocks have $diam(V_i) \leq \Delta$ for $i = 1, \ldots, k$.*

*(b) Let $U \subseteq V$ be a subset of points. Then*

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

In the above, *separated* means that there is more than one index $i$ with $V_i \cap U \neq \emptyset$.

### *4.2.3 The analysis of the CKR clustering*

The claim from Theorem 31.(a) is easy to show as

$$\operatorname{diam}(V_i) = \max_{u,v \in V_i} d(u,v) \leq 2 \max_{u \in V_i} \underbrace{d(u, c_i)}_{\leq \beta\Delta} \leq \Delta$$

The tricky part is to show Theorem 31.(b). The following definition and lemma are needed.

**Definition 5.** Let us say that a node $w$ is a *separator* for $U$, if

(A) $\sigma(u) = w$ for at least one $u \in U$

(B) $\sigma(u) \neq w$ for at least one $u \in U$

Moreover, if the set of separators of $U$ is non-empty, then we call the separator that comes first in the order $\pi$ the *first separator*.

Next, we show that nodes that are closer to the set $U$ are the most likely to be the first separator:

**Lemma 32.** *Let $w_1, \ldots, w_n$ be the nodes sorted so that $d(w_1, U) \leq \ldots \leq d(w_n, U)$. Then* $\Pr[w_s \text{ is the first separator for } U] \leq \frac{4}{s} \cdot \frac{diam(U)}{\Delta}$.

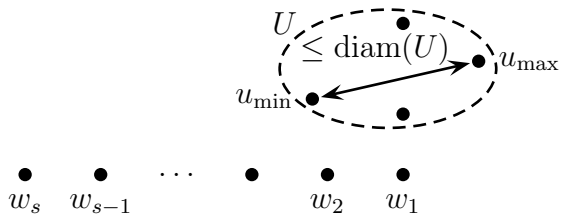



Figure 4.1: Visualization of CKR analysis

*Proof.* Let $u_{\min} := \operatorname{argmin}\{d(u, w_s) : u \in U\}$ and $u_{\max} := \operatorname{argmax}\{d(u, w_s) : u \in U\}$ be the closest and furthest point from $w_s$. We claim that in order for $w_s$ to be the first separator, both of the following conditions must hold:

(i) $d(w_s, u_{\min}) \leq \beta \cdot \Delta < d(w_s, u_{\max})$

(ii) The order selects $w_s$ as the first node among $w_1, \ldots, w_s$.

We assume that $w_s$ is the first separator, and suppose for the sake of contradiction that either (i) or (ii) (or both) are not satisfied. We verify the cases:

- *Case:* $\beta\Delta < d(w_s, u_{\min})$. Then no point will be assigned to $w_s$ and $w_s$ is not a separator at all.

- *Case:* $\beta\Delta \geq d(w_s, u_{\max})$. As $w_s$ is a separator, there are nodes $u_1, u_2 \in U$ with $\sigma(u_1) = w_s$ and $\sigma(u_2) \neq w_s$. Then $\sigma(u_2)$ has to come earlier in the order $\pi$ as $d(w_s, u_2) \leq \beta\Delta$. Hence $w_s$ is not the first separator.

- *Case:* $w_s$ *is not first among* $w_1, \ldots, w_s$ *with respect to* $\pi$. By assumption there is an index $1 \leq s_2 < s$ such that $\pi(w_{s_2}) < \pi(w_s)$. As $w_s$ is a separator, there is a $u_1 \in U$ with $\sigma(u_1) = w_s$. Let $u_2 := \operatorname{argmin}\{d(u, w_{s_2}) : u \in U\}$ be the point in the set $U$ that is closest to $w_{s_2}$. Then $d(u_2, w_{s_2}) = d(w_{s_2}, U) \leq d(w_s, U) \leq d(w_s, u_1) \leq \beta\Delta$. Hence $u_2$ would be assigned to a point of order at most $\pi(w_{s'}) < \pi(w_s)$, and therefore $w_s$ is not the first separator.

Now we estimate the probability that $w_s$ is the first separator. The parameter $\beta$ and the permutation are chosen independently, so $(i)$ and $(ii)$ are independent events. Clearly $\Pr[(ii)] = \frac{1}{s}$. Moreover

$$\Pr[(i)] = \frac{|[d(w_s, u_{\min}), d(w_s, u_{\max})] \cap [\frac{\Delta}{4}, \frac{\Delta}{2}]|}{\Delta/4} \leq \frac{4d(u_{\min}, u_{\max})}{\Delta} \leq \frac{4\mathrm{diam}(U)}{\Delta},$$

where we have used the triangle inequality and the notation $|[a, b]| = b - a$ for the length of an interval. □

Now we can finish the proof of Theorem 31:

*Proof of Theorem 31.* As in Lemma 32, let $w_1, \ldots, w_n$ be an order of nodes such that $d(w_1, U) \leq \ldots \leq d(w_n, U)$. Note that $L := |N(U, \frac{\Delta}{2})| \leq n$ is the maximal index with $d(w_L, U) \leq \frac{\Delta}{2}$. If $U$ is separated, then there has to be a first separator. Therefore, the following holds:

$$\begin{aligned} \Pr[U \text{ is separated}] &\leq \sum_{s=1}^{L} \Pr[w_s \text{ is first separator for } U] \\ &\overset{\text{Lem 32}}{\leq} \underbrace{\sum_{s=1}^{L} \frac{1}{s}}_{\leq \ln(2L)} \cdot \frac{4\mathrm{diam}(U)}{\Delta} \leq \ln(2L) \cdot \frac{4\mathrm{diam}(U)}{\Delta} \end{aligned}$$

Here we use that $\Pr[w_s \text{ is first separator for } U] = 0$ for $s > L$ since a node $w_s$ that has a distance bigger than $\frac{\Delta}{2}$ to $U$ will never be a separator. □

### *4.3 An Approximation for* $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c\text{−}\mathsf{intervals} \mid C_{\max}$

In this section, we provide an approximation algorithm for scheduling $n$ unit-length jobs $J$ with communication delay $c \in \mathbb{N}$ on an unbounded number of machines so that precedence constraints given by a partial order $\prec$ are satisfied. Instead of working with $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c \mid C_{\max}$ directly, it will be more convenient to consider a slight variant that we call $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c\text{−}\mathsf{intervals} \mid C_{\max}$. This problem variant has the same input but the time horizon is partitioned into time *intervals* of length $c$, say $I_s = [sc, (s+1)c)$ for $s \in \mathbb{Z}_{\geq 0}$. The goal is to assign jobs to intervals and machines. We require that if $j_1 \prec j_2$ then either $j_1$ is

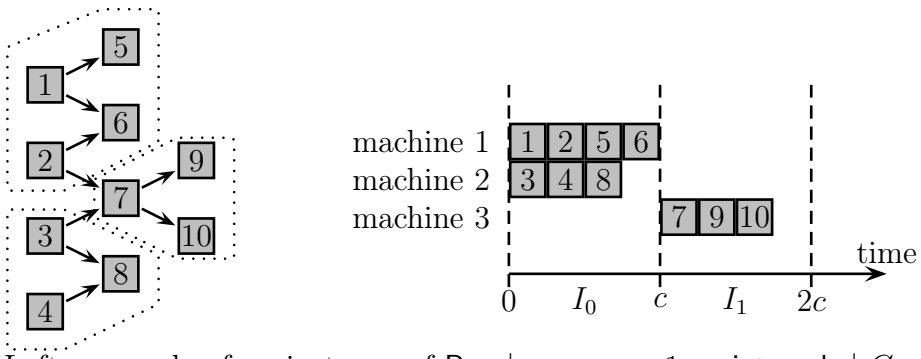



Figure 4.2: Left: example of an instance of $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c\text{–}\mathsf{intervals} \mid C_{\max}$ with $c = 4$ (where the partial order $\prec$ is the transitive closure of the depicted digraph). Right: a valid schedule in 2 intervals.

scheduled in an earlier interval than $j_2$ or $j_1$ and $j_2$ are scheduled in the same interval on the same machine. Other than that, there are no further communication delays. The objective function is to minimize the number of intervals used to process the jobs. In fact we do not need to decide the order of jobs within intervals as any topological order will work. In a more mathematical notation, the problem asks to find a partition $J = \dot{\bigcup}_{s\in\{0,\dots,S-1\},i\in\mathbb{N}} J_{s,i}$ with $|J_{s,i}| \leq c$ such that $S$ is minimized and for every $j_1 \prec j_2$ with $j_1 \in J_{s_1,i_1}$ and $j_2 \in J_{s_2,i_2}$ one has either $s_1 < s_2$ or $(s_1, i_1) = (s_2, i_2)$. See Figure 4.2 for an illustration.

It is rather straightforward to give reductions between $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c \mid C_{\max}$ and $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c\text{–}\mathsf{intervals} \mid C_{\max}$ that only lose a small constant factor in both directions. The only subtle point to consider here is that when the optimum makespan for $\mathsf{P}\infty \mid \mathsf{prec}, c \mid C_{\max}$ is less than $c$, the problem admits a PTAS; we refer to Section 4.4 for details.

### *4.3.1 The Linear Program*

Let $m \in \mathbb{N}$ be a parameter defining the number of machines that we admit for the LP. Moreover, let $S \in \mathbb{N}$ be the number of intervals that we allow for the time horizon. To obtain an approximation for $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c\text{–}\mathsf{intervals} \mid C_{\max}$ one can set $m := n$ and perform a binary search to find the minimal $S$ for which the LP is feasible. But we prefer to keep the approach general.

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

$$x_{j,i,s} = \begin{cases} 1 & \text{if } j \text{ is scheduled on machine } i \text{ in interval } I_s \\ 0 & \text{otherwise} \end{cases} \qquad \forall j \in J, i \in [m], s \in \{0, \ldots, S-1\}$$

Let $K$ be the set of fractional solutions to the following linear system

$$\begin{aligned} \sum_{i \in [m]} \sum_{s \geq 0} x_{j,i,s} &= 1 \quad \forall j \in J \\ \sum_{j \in J} x_{j,i,s} &\leq c \quad \forall i \in [m] \ \ \forall s \in \{0, \ldots, S-1\} \\ 0 \leq x_{j,i,s} &\leq 1 \quad \forall j \in J, i \in [m], s \in \{0, \ldots, S-1\} \end{aligned}$$

So far, $K$ simply assigns jobs (fractionally) to intervals and machines without taking any precedence constraints into account. Next, we will use a lift $x \in \mathrm{SA}_r(K)$ containing variables $x_{(j_1,i_1,s_1),(j_2,i_2,s_2)}$, which provide the probability for the event that $j_1$ is scheduled in interval $s_1$ on machine $i_1$ and $j_2$ is scheduled in interval $s_2$ on machine $i_2$. We introduce two more types of decision variables:

$$\begin{aligned} y_{j_1,j_2} &= \begin{cases} 1 & j_1 \text{ and } j_2 \text{ are scheduled on the same machine in the same interval} \\ 0 & \text{otherwise} \end{cases} \\ C_j &= \text{index of interval where } j \text{ is processed} \end{aligned}$$

Let $Q(r)$ be the set of vectors $(x, y, C)$ that satisfy

$$\begin{aligned} y_{j_1,j_2} &= \sum_{s \in \{0,\ldots,S-1\}} \sum_{i \in [m]} x_{(j_1,i,s),(j_2,i,s)} \\ C_{j_2} &\geq C_{j_1} + (1 - y_{j_1,j_2}) \quad \forall j_1 \prec j_2 \\ C_j &\geq 0 \quad \forall j \in J \\ x &\in SA_r(K) \end{aligned}$$

The analysis of our algorithm will work for all $r \geq 5$ while solving the LP takes time $n^{O(r)}$. Here we make no attempt at optimizing the constant $r$. The main technical contribution of this section is the following rounding result:

**Theorem 33.** *Consider an instance with unit-length jobs $J$, a partial order $\prec$, and parameters $c, S, m \in \mathbb{N}$ such that $Q(r)$ is feasible for $r := 5$. Then there is a randomized algorithm with expected polynomial running time that finds a schedule for $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c\text{−}\mathsf{intervals} \mid C_{\max}$ using at most $O(\log m \cdot \log c) \cdot S$ intervals.*

We would like to emphasize that we require $\prec$ to be a partial order, which implies that it is transitive. While replacing any acyclic digraph with its transitive closure does not change the set of feasible integral schedules and hence can be done in a preprocessing step, it corresponds to adding constraints to the LP that we rely on in the algorithm and in its analysis.

We will now discuss some properties that are implied by the Sherali-Adams lift:

**Lemma 34.** *Let $(x, y, C) \in Q(r)$ with $r \geq 2$. Then for any set $\tilde{J} \subseteq J$ of $|\tilde{J}| \leq r - 2$ jobs, there exists a distribution $\mathcal{D}(\tilde{J})$ over pairs $(\tilde{x}, \tilde{y})$ such that*

*(A) $\tilde{x}_{j,i,s} \in \{0, 1\}$ for all $j \in \tilde{J}$, all $i \in [m]$ and $s \geq 0$.*
*(B) $\tilde{y}_{j_1,j_2} = \sum_{s\geq 0} \sum_{i\in[m]} \tilde{x}_{j_1,i,s} \cdot \tilde{x}_{j_2,i,s}$ if $|\{j_1, j_2\} \cap \tilde{J}| \geq 1$.*
*(C) $\tilde{x} \in K$, $\tilde{y}_{j_1,j_2} = \sum_{s\in\{0,\dots,S-1\}} \sum_{i\in[m]} \tilde{x}_{(j_1,i,s),(j_2,i,s)}$ for all $j_1, j_2 \in J$.*
*(D) $\mathbb{E}[\tilde{x}_{j,i,s}] = x_{j,i,s}$ and $\mathbb{E}[\tilde{y}_{j_1,j_2}] = y_{j_1,j_2}$ for all $j, j_1, j_2, i, s$.*

*Proof.* By Theorem 30.(f), there is a distribution over $\tilde{x} \in SA_2(K)$ which satisfies $(A)$ and has $\tilde{x} \in K$, $\mathbb{E}[\tilde{x}_{j,i,s}] = x_{j,i,s}$ and $\mathbb{E}[\tilde{x}_{(j_1,i_1,s_1),(j_2,i_2,s_2)}] = x_{(j_1,i_1,s_1),(j_2,i_2,s_2)}$, and additionally is integral on variables involving only jobs from $\tilde{J}$, where $|\tilde{J}| \leq r - 2$. Here, we crucially use that every job $j \in \tilde{J}$ is part of an assignment constraint $\sum_{i\in[m]} \sum_{s\geq 0} x_{j,i,s} = 1$, hence making these variables integral results in the loss of only one round per job. Then, the $y$-variables are just linear functions depending on the $x$-variables, so we can define

$$\tilde{y}_{j_1,j_2} := \sum_{s\in\{0,\dots,S-1\}} \sum_{i\in[m]} \tilde{x}_{(j_1,i,s),(j_2,i,s)}$$

and the claim follows. ∎

From the LP solution, we define a semimetric $d$. Here the intuitive interpretation is that a small distance $d(j_1, j_2)$ means that the LP schedules $j_1$ and $j_2$ mostly on the same machine and in the same interval.

**Lemma 35.** *Let $(x, y, C) \in Q(r)$ be a solution to the 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 (see Section 4.2.2) are clearly satisfied. We verify the triangle inequality. Consider three jobs $j_1, j_2, j_3 \in J$. We apply Lemma 34 with $\tilde{J} := \{j_1, j_2, j_3\}$ 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

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

□

**Lemma 36.** *For every $j_1 \in J$ one has $\sum_{j_2 \in J} y_{j_1,j_2} \leq c$.*

*Proof.* Consider the distribution $(\tilde{x}, \tilde{y}) \sim \mathcal{D}(\{j_1\})$. From Lemma 34.(B) we know that $\mathbb{E}[\tilde{y}_{j_1,j_2}] = y_{j_1,j_2}$ and $\tilde{y}_{j_1,j_2} = \sum_{s \in \{0,\dots,S-1\}} \sum_{i \in [m]} \tilde{x}_{j_1,i,s} \cdot \tilde{x}_{j_2,i,s}$. By linearity it suffices to prove that $\sum_{j_2 \in J} \tilde{y}_{j_1,j_2} \leq c$ always. Fix a pair $(\tilde{x}, \tilde{y})$. There is a unique pair of indices $(i_1, s_1)$ with $\tilde{x}_{j_1,i_1,s_1} = 1$. Then

$$\sum_{j_2 \in J} \tilde{y}_{j_1,j_2} = \sum_{s \in \{0,\dots,S-1\}} \sum_{j_2 \in J} \sum_{i \in [m]} \underbrace{\tilde{x}_{j_1,i,s}}_{0 \text{ if } i \neq i_1 \text{ or } s \neq s_1} \cdot \tilde{x}_{j_2,i,s} = \sum_{j_2 \in J} \tilde{x}_{j_2,i_1,s_1} \leq c.$$

□

A crucial insight is that for any job $j^*$, few jobs are very close to $j^*$ with respect to $d$.

**Lemma 37.** *Fix $j^* \in J$ and abbreviate $U := \{j \in J \mid d(j, j^*) \leq \beta\}$ for $0 < \beta < 1$. Then $|U| \leq \frac{c}{1-\beta}$.*

*Proof.* For each $j \in U$ we have $y_{j,j^*} = 1 - d(j, j^*) \geq 1 - \beta$. Combining with the last lemma we have $(1 - \beta)|U| \leq \sum_{j \in J} y_{j,j^*} \leq c$. □

### *4.3.2 Scheduling a Single Batch of Jobs*

We now come to the main building block of our algorithm. We consider a subset $J^*$ of jobs whose LP completion times $C_j$ are very close (within a $\Theta(\frac{1}{\log(c)})$ term of each other) and show we can schedule half of these jobs in a single length-$2c$ interval. The following lemma is the main technical contribution of the paper.

**Lemma 38.** *Let $(x, y, C) \in Q(r)$ with $r \geq 5$ and let $0 < \delta \leq \frac{1}{64 \log(4c)}$ be a parameter. Let $C^* \geq 0$ and set $J^* \subseteq \{j \in J \mid C^* \leq C_j \leq C^* + \delta\}$. Then there is a randomized rounding procedure that finds a schedule for a subset $J^{**} \subseteq J^*$ in a single interval of length at most $2c$ such that every job $j \in J^*$ is scheduled with probability at least $1 - 32 \log(4c) \cdot \delta \geq \frac{1}{2}$.*

We denote $\Gamma^-(j)$ as the predecessors of $j$ and $\Gamma^+(j)$ as the successors, and similarly $\Gamma^{-/+}(J') = \{j \in J : \exists j' \in J' \text{ s.t. } j \in \Gamma^{-/+}(j')\}$. Again, recall that we assume $\prec$ to be transitive. The rounding algorithm is the following:

| Scheduling a Single Batch |
|---|
| (1) Run a CKR clustering on the semimetric space $(J^*, d)$ with parameter $\Delta := \frac{1}{4}$ and let $V_1, \ldots, V_k$ be the clusters. |
| (2) Let $V'_\ell := \{j \in V_\ell \mid \Gamma^-(j) \cap J^* \subseteq V_\ell\}$ for $\ell = 1, \ldots, k$. |
| (3) Schedule $V'_\ell$ on one machine for all $\ell = 1, \ldots, k$. |

We now discuss the analysis. First we show that no cluster is more than a constant factor too large.

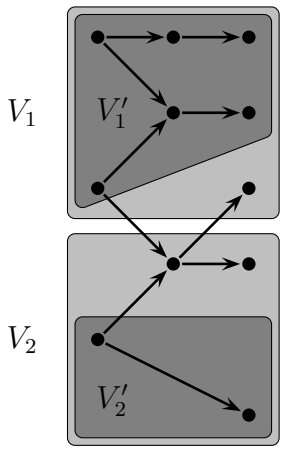



Figure 4.3: Visualization of the partition $V = V_1 \dot\cup \ldots \dot\cup V_k$ and the induced sets $V_\ell' \subseteq V_\ell$. Here $\prec$ is the transitive closure of the depicted digraph.

**Lemma 39.** *One has $|V_\ell'| \leq 2c$ for all $\ell = 1, \ldots, k$.*

*Proof.* We know by Theorem 31 that $\operatorname{diam}(V_\ell') \leq \operatorname{diam}(V_\ell) \leq \Delta < \frac{1}{2}$ where the diameter is with respect to $d$. Fix a job $j^* \in V_\ell'$. Then we know by Lemma 37 that there are at most $2c$ jobs $j$ with $d(j, j^*) \leq \frac{1}{2}$ and the claim follows. □

Next, we see that the clusters respect the precedence constraints.

**Lemma 40.** *The solution $V_1', \ldots, V_k'$ 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 $V_\ell'$. □

A crucial property that makes the algorithm work is that predecessors of some job $j \in J^*$ must be very close in $d$ distance.

**Lemma 41.** *For every $j_1, j_2 \in J^*$ with $j_1 \prec j_2$ one has $d(j_1, j_2) \leq \delta$.*

*Proof.* We know that

$$C^* \overset{j_1 \in J^*}{\leq} C_{j_1} \leq C_{j_1} + \underbrace{(1 - y_{j_1,j_2})}_{=d(j_1,j_2)} \overset{LP}{\leq} C_{j_2} \overset{j_2 \in J^*}{\leq} C^* + \delta$$

and so $d(j_1, j_2) \leq \delta$. □

We will use the three statements above together with Theorem 31 to prove Lemma 38.

*Proof of Lemma 38.* We have already proven that the scheduled blocks have size $|V'_\ell| \leq 2c$ and that there are no dependent jobs in different sets of $V'_1, \ldots, V'_k$. To finish the analysis, we need 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 $J^*$.

Since the diameter of $U$ is at most $2\delta$ by Lemma 41, we can use Lemma 37 to see that $|N(U, \Delta/2)| \leq \frac{c}{1-2\delta-\Delta}$. For our choice of $\Delta = 1/4$ and $\delta \leq \frac{1}{64 \log(4c)}$, $|N(U, 1/8)| \leq 2c$. From Theorem 31, the cluster is separated with probability at most $\log(4c) \cdot \frac{8\delta}{\Delta} \leq \frac{1}{2}$. □

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

**Lemma 42.** *Let $(x, y, C) \in Q(r)$ with $r \geq 5$. Let $C^* \geq 0$ and set $J^* \subseteq \{j \in J \mid C^* \leq C_j < C^* + \delta\}$. Assume that all jobs in $\Gamma^-(J^*) \setminus J^*$ have been scheduled respecting precedence constraints. Then there is an algorithm with expected polynomial running time that schedules all jobs in $J^*$ using at most $O(\log m) + \frac{|J^*|}{mc}$ many intervals.*

*Proof.* Our algorithm in Lemma 38 schedules each $j \in J^*$ in an interval of length $2c$ with probability at least $1/2$. We run the algorithm for $2 \log m$ iterations, where input to iteration $k + 1$ is the subset of jobs that are not scheduled in the first $k$ iterations. For $k \in \{1, 2, \ldots, 2 \log m\}$, let $J^{**}_k$ denote the subset of jobs that are scheduled in the $k^{th}$ iteration, and let $J^*_{k+1} := J^* \setminus \{\bigcup_{k'=1}^{k} J^{**}_{k'}\}$. In this notation, $J^*_1 := J^*$. Let $\mathcal{S}(J^{**}_k)$ denote the schedule of jobs $J^{**}_k$ given by Lemma 38. We schedule $\mathcal{S}(J^{**}_1)$ first, then for $k = 2, \ldots, 2 \log m$, we append the schedule $\mathcal{S}(J^{**}_k)$ after $\mathcal{S}(J^{**}_{k-1})$. Let $J' := J^*_{2 \log m + 1}$ denote the set of jobs that

were not scheduled in the $2\log m$ iterations. We schedule all jobs in $J'$ consecutively on a single machine 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 $O(\log m)+\lceil\frac{|J'|}{c}\rceil$ many intervals. For $k \in \{1, 2, \ldots, 2\log m\}$, Lemma 38 guarantees that each job $j \in J^*_k$ gets scheduled in the $k^{th}$ iteration with probability at least $1/2$. Therefore, the probability that $j \in J'$, i.e., it does not get scheduled in the first $2\log m$ iterations, is at most $\frac{1}{2m}$. This implies that $\mathbb{E}[|J'|] \leq \frac{|J^*|}{2m}$. By Markov's inequality $\Pr[|J'| > \frac{|J^*|}{m}] \leq \Pr[|J'| > 2\cdot\mathbb{E}[|J'|]] \leq 1/2$. Hence we can repeat the described procedure until indeed we have a successful run with $|J'| \leq \frac{|J^*|}{m}$ which results in the claimed expected polynomial running time.

Let us now argue that the schedule of $J^*$ is feasible. For $k \in \{1, 2, \ldots, 2\log m\}$ and any two jobs $j, j' \in \mathcal{S}(J^{**}_k)$, Lemma 38 guarantees that precedence and communication constraints are satisfied. Furthermore, Lemma 38 also ensures that there cannot be jobs $j$, $j'$ such that $j \in \mathcal{S}(J^{**}_k)$, $j' \in \mathcal{S}(J^{**}_{k'})$ and $j' \prec j$ and $k' > k$. Finally note that every length-$2c$ interval can be split into 2 length-$c$ intervals. The claim follows. □

### *4.3.3 The Complete Algorithm for* $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c{-}\mathsf{intervals} \mid C_{\max}$

Now we have all the pieces to put the rounding algorithm together and prove its correctness. We partition the jobs into batches, where each batch consists of subset of jobs that have $C_j$ very close to each other in the LP solution. The complete algorithm is given below.

**The Complete Algorithm**

(1) Solve the LP and let $(x, y, C) \in Q(r)$ with $r \geq 5$.

(2) For $\delta = \frac{1}{64\log(4c)}$ and $k \in \{0, 1, 2 \ldots \frac{S-1}{\delta}\}$, define
$J_k = \{j \in J : k\cdot\delta \leq C_j < (k+1)\cdot\delta\}$

(3) FOR $k = 0$ TO $\frac{S-1}{\delta}$ DO

(4) Schedule the jobs in $J_k$ using the algorithm in Subsection 4.3.2.

Now we finish the analysis of the rounding algorithm.

*Proof of Theorem 33.* Let us quickly verify that the schedule constructed by our algorithm is feasible. For jobs $j_1 \prec j_2$ with $j_1 \in J_{k_1}$ and $j_2 \in J_{k_2}$, the LP implies that $C_{j_1} \leq C_{j_2}$ and so $k_1 \leq k_2$. If $k_1 < k_2$, then $j_1$ will be scheduled in an earlier interval than $j_2$. If $k_1 = k_2 = k$, then Lemma 42 guarantees that precedence constraints are satisfied.

It remains to bound the makespan of our algorithm. Lemma 42 guarantees that for $k \in \{0, 1, 2 \dots \frac{S-1}{\delta}\}$, the jobs in $J_k$ are scheduled using at most $O(\log m) + \frac{|J_k|}{cm}$ many intervals. Then the total number of intervals required by the algorithm is bounded by

$$\frac{S}{\delta} \cdot O(\log m) + \sum_{k=0}^{\frac{S-1}{\delta}} \frac{|J_k|}{cm} = O(\log m \cdot \log c) \cdot S + \frac{|J|}{cm} \leq O(\log m \cdot \log c) \cdot S.$$

Here we use that $|J| \leq S \cdot cm$ is implied by the constraints defining $K$. □

*Remark* 1. We note that it is possible to reverse-engineer our solution and write a more compact LP for the problem, enforcing only the necessary constraints such as those given by Lemmas 35 and 36. We present this compact LP and prove an integrality gap of $\Omega(\sqrt{\log n})$ for it in Section 4.5. While the compact LP is simpler and could be solved more efficiently, we feel that the Sherali-Adams hierarchy gives a more principled and intuitive way to tackle the problem and explain how the LP arises, and hence we choose to present it that way.

### *4.4 Reductions*

We now justify our earlier claim: the special case $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c\text{—}\mathsf{intervals} \mid C_{\max}$ indeed captures the full computational difficulty of the more general problem $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\max}$. The main result for this section will be the following reduction:

**Theorem 43.** *Suppose there is a polynomial time algorithm that takes a solution for the LP $Q(r)$ with parameters $m, c, S \in \mathbb{N}$ and $r \geq 5$ and transforms it into a schedule for $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c\text{—}\mathsf{intervals} \mid C_{\max}$ using at most $\alpha \cdot S$ intervals. Then there is a polynomial time $O(\alpha)$-approximation for $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\max}$.*

For the reduction we will make use of the very well known list scheduling algorithm by

Graham [Gra66b] that can be easily extended to the setting with communication delays. Here the notation $\sigma(j) = ([t, t+p_j), i)$ means that the job $j$ is processed in the time interval $[t, t+p_j)$ on machine $i \in [m]$.

GRAHAM'S LIST SCHEDULING

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

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

  (3) Select any job $j \in J$ with $\sigma(j) = \emptyset$ where every $j' \prec j$ satisfies the following:

    - If $j'$ is scheduled on machine $i$ then $j'$ is finished at time $\leq t$
    - If $j'$ is schedule on machine $i' \neq i$ then $j'$ finished at time $\leq t-c$

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

For example, for the problem $\mathsf{P} \mid \mathsf{prec} \mid C_{\max}$, Graham's algorithm gives a 2-approximation. The analysis works by proving that there is a chain of jobs covering all time units where not all machines are busy. Graham's algorithm does *not* give a constant factor approximation for our problem with communication delays, but it will still be useful for our reduction.

Recall that a set of jobs $\{j_1, \ldots, j_\ell\} \subseteq J$ with $j_\ell \prec j_{\ell-1} \prec \ldots \prec j_1$ is called a *chain*. We denote $\mathcal{Q}(J)$ as the set of all chains in $J$ w.r.t. precedence order $\prec$.

**Lemma 44.** *Graham's list scheduling on an instance of* $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\max}$ *results in a schedule with makespan at most* $\frac{1}{m}\sum_{j\in J} p_j + \max_{Q\in\mathcal{Q}(J)}\{\sum_{j\in Q} p_j + c \cdot (|Q| - 1)\}$.

*Proof.* We will show how to construct the chain $Q$ that makes the inequality hold. Let $j_1$ be the job which finishes last in the schedule produced by Graham's algorithm and let $t_{j_1}$ be its start time. Let $j_2$ be the predecessor of $j_1$ that finishes last. More generally in step $i$, we denote $j_{i+1}$ as the predecessor of $j_i$ that finishes last. The construction finishes with a job $j_\ell$ without predecessors. Now let $Q$ be the chain of jobs $j_\ell \prec j_{\ell-1} \prec \ldots \prec j_1$. The crucial observation is that for any $i \in \{1, \ldots, \ell-1\}$, either all machines are busy in the time interval $[t_{j_{i+1}} + p_{j_{i+1}} + c, t_{j_i})$ or this interval is empty. The reason is that Graham's algorithm

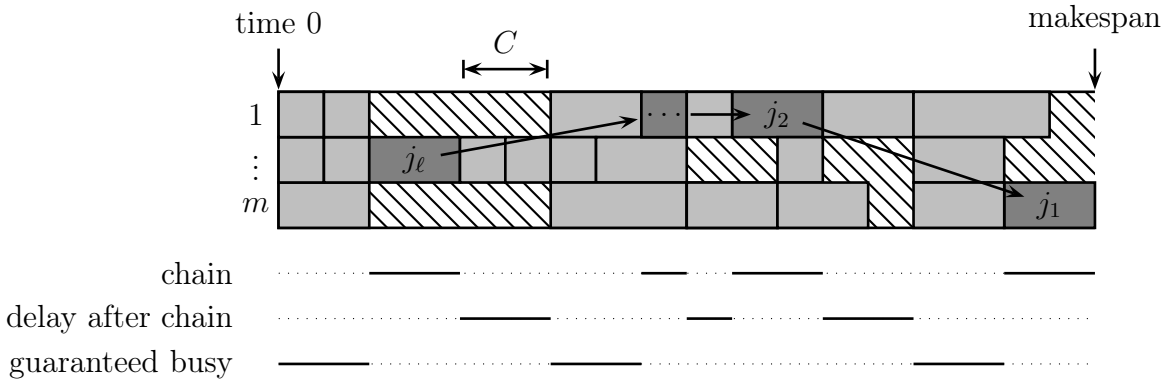



Figure 4.4: Analysis of Graham's algorithm with communication delay $c$.

does not leave unnecessary idle time and would have otherwise processed $j_i$ earlier. It is also true that all $m$ machines are busy in the time interval $[0, t_{j_\ell})$. The total amount of work processed in these busy time intervals is

$$L := m \cdot \left( t_{j_\ell} + \sum_{i=1}^{\ell-1} \max\{t_{j_i} - (c + p_{j_{i+1}} + t_{j_{i+1}}), 0\} \right) \leq \sum_{j \in J} p_j - \sum_{i=1}^{\ell} p_{j_i}.$$

Then any time between 0 and the makespan falls into at least one of the following categories: (a) the busy time periods described above, (b) the times that a job of the chain $Q$ is processed, (c) the interval of length $c$ following a job in the chain $Q$. Thus, we see that the makespan from Graham's list scheduling is at most

$$t_{j_1} + p_{t_{j_1}} \leq \frac{L}{m} + \sum_{j \in Q} p_j + c \cdot (|Q| - 1) \leq \frac{1}{m} \sum_{j \in J} p_j + \left(1 - \frac{1}{m}\right) \sum_{j \in Q} p_j + c \cdot (|Q| - 1).$$

□

It will also be helpful to note that the case of very small optimum makespan can be well approximated:

**Lemma 45.** *Any instance for* $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\max}$ *with optimum objective function value at most* $c$ *admits a PTAS.*

*Proof.* Let $J$ be the jobs in the instance and let $\mathrm{OPT}_m \leq c$ be the optimum value. Consider the undirected graph $G = (J, E)$ with $\{j_1, j_2\} \in E \Leftrightarrow ((j_1 \prec j_2)$ or $(j_2 \prec j_1))$. Let

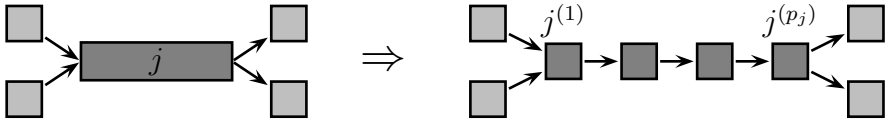
$J = J_1 \dot\cup \ldots \dot\cup J_N$ be the partition of jobs into connected components w.r.t. graph $G$. We abbreviate $p(J') := \sum_{j \in J'} p_j$. The assumption guarantees that the optimum solution cannot afford to pay the communication delay and hence there is a length-$c$ schedule that assigns all jobs of the same connected component to the same machine. If we think of a connected component $J_\ell$ as an "item" of size $p(J_\ell)$, then for any fixed $\varepsilon > 0$ we can use a PTAS for $\mathsf{P} \,||\, C_{\max}$ (i.e. makespan minimization without precedence constraints) to find a partition of "items" as $[N] = I_1 \dot\cup \ldots \dot\cup I_m$ with $\sum_{\ell \in I_i} p(J_\ell) \leq (1+\varepsilon) \cdot \mathrm{OPT}_m$ in polynomial time [HS87b]. Arranging the jobs $\bigcup_{\ell \in I_i} J_\ell$ on machine $i$ in any topological order finishes the argument. □

Additionally, it is a standard argument to convert an instance with arbitrary $p_j$ to an instance where all $p_j \leq n/\varepsilon$, while only losing a factor of $(1+2\varepsilon)$ in the approximation. For $p_{\max} := \max_j p_j$, we simply scale the job lengths and communication delay down by a factor of $\frac{n}{\varepsilon p_{\max}}$ then round them to the nearest larger integer. This results in at most a $2\varepsilon$ fraction of the optimal makespan being rounded up and all job sizes are integral and at most $n/\varepsilon$.

Now we can show the main reduction:

*Proof of Theorem 43.* Consider an instance of $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\max}$ with $p_j, c \in \mathbb{N}$. Let $J$ denote its job set with precedence constraints, and $\mathrm{OPT}_m(J)$ denote its optimal value where $m$ is the number of available machines. By the previous argument, we may assume that $p_j \leq 2n$ for all $j \in J$. Moreover, by Lemma 45 we only need to focus on the case where $\mathrm{OPT}_m(J) > c$. We may guess the optimum value of $\mathrm{OPT}_m(J)$ as $\mathrm{OPT}_m(J) \in \{1, \ldots, 2n^2\}$.

Let $J'$ denote the job set obtained by splitting each job $j \in J$ into a chain of $p_j$ unit sub-jobs $j^{(1)} \prec \cdots \prec j^{(p_j)}$. Moreover, precedence constraints in $J$ are preserved in $J'$ as we set all predecessors of $j$ to be predecessors of $j^{(1)}$ and all successors of $j$ to be successors of $j^{(p_j)}$, see Figure 4.5. We note that $\mathrm{OPT}_m(J') \leq \mathrm{OPT}_m(J)$ as splitting does not increase the value of the optimum. Let $\mathcal{S}_m(J')$ be a schedule achieving the value of $\mathrm{OPT}_m(J')$. Next, observe that $\mathcal{S}_m(J')$ implies an integral solution for $Q(r)$ with parameters $m, c, S$ where $S := \lceil \mathrm{OPT}_m(J)/c \rceil$ and $r := 5$. In particular here we use that if jobs $j_1 \prec j_2$ are scheduled on different machines by $\mathcal{S}_m(J')$, then their starting times differ by at least $c+1$ and hence




instance $J$ instance $J'$

Figure 4.5: Splitting jobs into chains of unit-length jobs.

they are assigned to different length-$c$ intervals.

Now we execute the assumed $\alpha$-approximate rounding algorithm and obtain a schedule $\mathcal{S}_{\infty,\text{int}}(J')$ that uses $T \leq \alpha S$ many intervals. We will use this solution $\mathcal{S}_{\infty,\text{int}}(J')$ to construct a schedule $\mathcal{S}_\infty(J)$ for $\mathsf{P}\infty \mid \mathsf{prec}, c \mid C_{\max}$ with job set $J$ by running split sub-jobs consecutively on the same processor. This will use $4T$ time intervals in total. Recall that $I_s$ denotes the time interval $[sc, (s+1)c)$. The rescheduling process is as follows:

For a fixed job $j \in J$, let $I_{s_1}$ be the time interval where $j^{(1)}$ is scheduled in $\mathcal{S}_{\infty,\text{int}}(J')$. Then all other sub-jobs of $j$ should be either scheduled in $I_{s_1}$ or the time intervals after $I_{s_1}$.

- **Case 1: Some sub-job of $j$ is not scheduled in $I_{s_1}$.**
  Schedule job $j$ at the beginning of time interval $I_{4s_1+1}$ on a new machine. If $j$ is a short job, then it will finish running by the end of the interval. Otherwise $j$ is a long job. Let $I_{s_2}$ be the last time interval where a sub-job of $j$ is scheduled in $\mathcal{S}_{\infty,\text{int}}(J')$. Then, the length satisfies $p_j \leq c \cdot (s_2 - s_1 + 1)$, which implies that the job finishes by time $c \cdot (4s_1 + 1) + p_j \leq c \cdot (4s_2 - 1)$.

- **Case 2: All sub-jobs of $j$ are scheduled in $I_{s_1}$.**
  Simply schedule job $j$ during time interval $I_{4s_1}$ on the same machine as in $\mathcal{S}_{\infty,\text{int}}(J')$.

See Figure 4.6 for a visualization. Then $\mathcal{S}_\infty(J)$ is a valid schedule for $\mathsf{P}\infty \mid \mathsf{prec}, c \mid C_{\max}$, with makespan $\leq 4c \cdot T$. Moreover, $\mathcal{S}_\infty(J)$ satisfies the following:

(a) A short job is fully contained in some interval $I_s$.

(b) A long job's start time is at the beginning of some interval $I_s$.

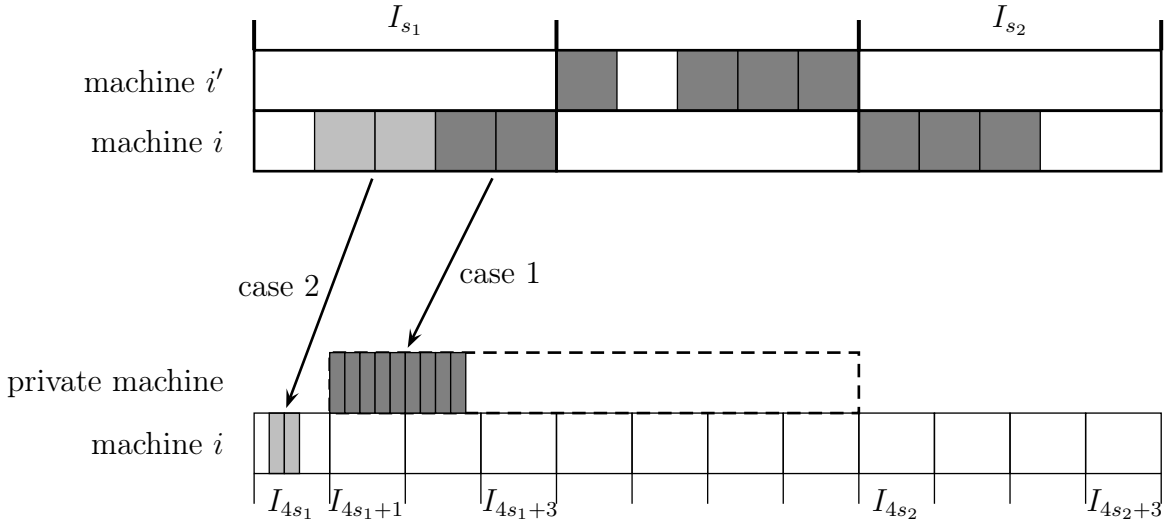



Figure 4.6: Transformation of the schedule $\mathcal{S}_{\infty,\text{int}}(J')$ (top) to $\mathcal{S}_\infty(J)$ (bottom), where $\mathcal{S}_\infty(J)$ is compressed by a factor of 4. Here a "private" machine for a job $j$ means the machine never processes any job other than $j$.

For $\mathcal{S}_\infty(J)$, define a new job set $H$. Every long job $j$ becomes an element of $H$ with its original running time $p_j$. Meanwhile, every set of short jobs that are assigned to the same machine in one time interval becomes an element of $H$, with running time equal to the sum of running times of the short jobs merged. To summarize, a new job $h \in H$ corresponds to a set $h \subseteq J$ and $p_h = \sum_{j \in h} p_j$.

We define the partial order $\tilde{\prec}$ on $H$ with $h_1 \tilde{\prec} h_2$ if and only if there are $j_1 \in h_1$ and $j_2 \in h_2$ with $j_1 \prec j_2$. One can check that this partial order is well defined. Moreover, by the fact that jobs assigned to the same interval but different machines do not have precedence constraints, the length of the longest chain in $(H, \tilde{\prec})$ in terms of the number of elements is bounded by the number of intervals that are used, which is at most $4T$.

Now run Graham's list scheduling on the new job set $H$ with order $\tilde{\prec}$ and $m$ machines. By Lemma 44, the makespan of the list scheduling is bounded by $\frac{1}{m}\sum_{h \in H} p_h + \max_{Q \in \mathcal{Q}(H)}\{\sum_{h \in Q} p_h + c \cdot |Q|\}$. As the total sum of the processing times does not change from $J$ to $H$, we see that $\frac{1}{m}\sum_{h \in H} p_h \leq \text{OPT}_m(J)$. Moreover, for any chain $Q \in \mathcal{Q}(H)$,

$\sum_{h \in Q} p_h$ is no greater than the makespan of $\mathcal{S}_\infty(J)$, which is $4cT$. Finally, as argued earlier, the chain has $|Q| \leq 4T$ elements. Above all,

$$\begin{aligned}
\frac{1}{m}\sum_{h \in H} p_h + \max_{Q \in \mathcal{Q}(H)} \Big\{ \sum_{h \in Q} p_h + c \cdot |Q| \Big\} &\leq \mathrm{OPT}_m(J) + 4cT + 4cT \\
&\leq \mathrm{OPT}_m(J) + 8\alpha \cdot cS \\
&\leq \mathrm{OPT}_m(J) + 16\alpha \cdot \mathrm{OPT}_m(J) \\
&= O(\alpha) \cdot \mathrm{OPT}_m(J).
\end{aligned}$$

□

### *4.5 Integrality Gap for the Compact LP*

We consider the following layered expander graph from Maiti et al. [MRS+20]. Let $V_1, \ldots, V_L$ be sets of $n$ vertices and for each $j \in V_k$, for $k \in [L-1]$, let $j$ sample $d$ successors from $V_{k+1}$ uniformly at random without replacement. We choose $n = cm/L$ and $c = d^L$, and set $L = \epsilon_1\sqrt{\log n}$, $d = 2^{\epsilon_2\sqrt{\log n}}$, and $m = c$. We will fix $\epsilon_1$ and $\epsilon_2$ as in Maiti et al.; the constants have several specifications, and we will not enumerate all of them, except that $\epsilon_2 \geq 5\epsilon_1$. Call this instance $\mathcal{I}$. Note that we have purposefully defined $\mathcal{I}$ so that its partial order is not closed under transitivity, as it will be notationally convenient later to consider the successors and predecessors of a node $j \in V_k$ as the nodes $j'$ in $\Gamma^+(j) \cap V_{k+1}$ and $\Gamma^-(j) \cap V_{k-1}$, respectively.

Our compact LP is the following:

$$
\begin{aligned}
y_{j_1,j_1} &= 1 \qquad \forall j_1 \in J \\
y_{j_1,j_2} &= y_{j_2,j_1} \qquad \forall j_1, j_2 \in J \\
1 - y_{j_1,j_2} &\leq 1 - y_{j_1,j_3} + 1 - y_{j_2,j_3} \qquad \forall j_1, j_2, j_3 \in J \\
\sum_{j \in J} y_{j_1,j} &\leq c \qquad \forall j_1 \in J \\
C_{j_2} &\geq C_{j_1} + (1 - y_{j_1,j_2}) \qquad \forall j_1 \prec j_2 \\
C_j &\geq 0 \quad \forall j \in J \\
0 \leq y_{j,j'} &\leq 1 \qquad \forall j, j' \in J
\end{aligned}
$$

Note the above completely does away with the $x$ variables from LP. We will prove the following when parameters for the layered expander graph are specified as above:

**Lemma 46.** *The compact LP has value at most* 1 *(c-interval) on instance* $\mathcal{I}$*.*

Maiti et al. prove that the number of $c$-intervals required for an integral solution to schedule $\mathcal{I}$ is $\Omega(L)$. Combining this with Lemma 46, we have the following theorem:

**Theorem 47.** *The compact LP has integrality gap at least* $\Omega(\sqrt{\log n})$*.*

Before we begin the proof of Lemma 46 we will define some new terms. First let $\phi(j) : J \to [L]$ be the map sending each job to its corresponding layer; so $j \in V_k$ if and only if $\phi(j) = k$. Next we let $r(j, j') = \min\{L, s(j, j')\}$, where $s(j, j')$ is the distance between $j$ and $j'$ in the undirected graph, i.e. the number of edges in the shortest path between $j$ and $j'$, when edges of the DAG are made undirected. Note that for $j \in V_k$ with $j' \in V_k \cap \Gamma^+(j)$ and $j'' \in V_{k+2} \cap \Gamma^+(j')$, we have that $r(j, j') = r(j', j'') = 1 - 1/L$ and $r(j, j'') = 1 - 2/L$. The function $r$ is clearly symmetric. We will show that $r$ satisfies the triangle inequality, since normally it is not the case that a minimum of two metrics is still a metric.

**Lemma 48.** *Let* $s(j, j')$ *be the distance between* $j$ *and* $j'$ *in the undirected graph. Then*

$r(j, j') = \min\{L, s(j, j')\}$ *satisfies the triangle inequality.*

*Proof.* Fix jobs $j, j', j'' \in J$. Suppose first that $r(j, j') = s(j, j')$. Since $s(j, j') \leq s(j, j'') + s(j'', j')$ and $s(j, j') \leq L$, it follows that $r(j, j') \leq r(j, j'') + r(j'', j')$. On the other hand if $r(j, j') = L$, the triangle inequality would only be violated if $j''$ was such that $L > s(j, j'') + s(j'', j')$. However, this inequality implies there is a path from $j$ to $j''$ through $j'$ that has $s(j, j'') + s(j'', j') < L$ edges, which contradicts the definition of $r(j, j')$. □

*Proof of Lemma 46.* The set of jobs in instance $\mathcal{I}$ is partitioned into layers $V_1, \ldots, V_L$. For every job $j$, let $j$'s completion time be $C_j = \phi(j)/L$. For jobs $j, j'$, let $y_{j,j'} = 1 - r(j, j')/L$.

Now we check the inequalities of the LP. We begin by seeing that these $y$ values form a semimetric $d(j, j') = 1 - y_{j,j'}$. For $j \in J$, $y_{j,j} = 1 - 0/L = 1$ so that $d(j, j) = 0$, and $d$ is symmetric since $r(j, j')$ is too. To see that the triangle inequality holds, fix $j, j', j''$ and observe that by Lemma 48.

$$1 - y_{j,j'} = r(j, j')/L \leq r(j, j'')/L + r(j'', j')/L = 1 - y_{j,j''} + 1 - y_{j'',j'}.$$

Further, we see that for $j, j'$ with $j \prec j'$, $C_j = \phi(j)/L$, $C_{j'} = \phi(j')/L$, and $y_{j,j'} = 1 - (\phi(j') - \phi(j))/L$, so that $C_{j'} - C_j \geq 1 - y_{j,j'}$.

The last inequality to check is that for all $j \in J$, $\sum_{j'} y_{j,j'} \leq c$. Fix a job $j$, and recall a job $j'$ has $y_{j,j'} = 1 - \ell/L$ exactly when $r(j, j') = \ell$. Every job, except those in $V_L$, has $d$ successors, and with probability at least $1 - n^{-2}$, every job, except those in $V_1$, has at most $d + 10\sqrt{d \log n}$ predecessors. Therefore, at most $2^\ell(d + 10\sqrt{d \log n})^\ell$ jobs $j'$ have $r(j, j') = \ell$. It follows that

$$\begin{aligned}
\sum_{j'} y_{j,j'} &\leq \sum_{\ell=1}^{L-1} 2^\ell (d + 10\sqrt{d \log n})^\ell (1 - \ell/L) \\
&= \frac{2(d + 10\sqrt{d \log n})((2(d + 10\sqrt{d \log n}))^L - 2(d + 10\sqrt{d \log n})L + L - 1)}{(2(d + 10\sqrt{d \log n}) - 1)^2 L} \\
&\leq \frac{2^{L+1}(d + 10\sqrt{d \log n})^{L+1}}{(d + 10\sqrt{d \log n})^2} \leq 2^{L+1}(d + 10\sqrt{d \log n})^{L-1} \leq 2^{L+1}(2d)^{L-1} \leq d^L = c,
\end{aligned}$$

where the second to last inequality follows from the fact that $d \geq 10\sqrt{\log n}$, and the last inequality follows from the choice of $d$ and $L$ and the fact that $\epsilon_2 \geq 2\epsilon_1$.

□

## *4.6 Minimizing Weighted Sum of Completion Times*

To illustrate the generality of our framework we show that it can be extended to handle different objective functions, in particular we can minimize the *weighted sum of completion times* of the jobs. Here we restrict to the simplest case where jobs have unit length and an unbounded number of machines are available. In the 3-field notation, this problem is denoted by $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c \mid \sum_j w_j C_j$. The input for this problem is the same as for the makespan minimization problem except that each job $j$ now has a weight $w_j \geq 0$.

The goal is to minimize the objective function $\sum_j w_j C_j$, where $C_j$ is the completion time of $j$, which is defined as the time slot in which job $j$ is scheduled.

Note that the LP $Q(r)$ has variables $C_j$ that denote the index of the length-$c$ interval where $j$ is being scheduled. A natural approach would be to interpret $c \cdot C_j$ as the completion time of job $j$ and minimize $\sum_{j \in J} w_j \cdot c \cdot C_j$ over $Q(r)$. Then the rounding algorithm from Section 4.3 will indeed schedule each job $j$ so that the completion time is at most $(O(\log c \cdot \log n) \cdot C_j + \Theta(\log n)) \cdot c$. We can observe that if an $O(\log c \cdot \log n)$ approximation is the goal, then this argument suffices for all jobs $j$ where the LP solution has $C_j \geq \Omega(\frac{1}{\log c})$ — but it fails for jobs with $0 \leq C_j \ll 1$.

### *4.6.1 The linear program*

In order to address this case, we first start with a more general LP relaxation compared to the makespan result which tracks the actual time slot where the jobs are processed, rather than just the interval. Again, we use the parameter $m \in \mathbb{N}$ to denote the number of machines that we allow the LP to use (one can set $m := n$) and the parameter $S \in \mathbb{N}$ to denote the number of intervals that we allow for the time horizon. We abbreviate $T := S \cdot c$ as the number of time slots. Note that $T \leq nc$ always suffices for any non-idling schedule.

We index time slots as $[T] := \{1, \ldots, T\}$ and consider an interval as a discrete set of slots $I_s := \{cs + 1, \ldots, c(s + 1)\}$ where $s \in \{0, \ldots, S - 1\}$.

Recall that in the makespan result $x_{j,i,s}$ variables indicated if job $j$ got scheduled on machine $i$ in the interval $s$. Here, we introduce additional variables of the form $z_{j,i,t}$ which indicate if job $j$ is scheduled on machine $i$ at time $t \in [T]$. The variables $x_{j,i,s}$ are fully determined by summing over appropriate variables $z_{j,i,t}$, but we retain them for notational convenience. Further, similar to our makespan result, we impose an interval structure on the optimal solution and lose an $O(1)$ factor in the approximation ratio.

Let $\tilde{K}$ be the set of fractional solutions to the following LP.

$$\begin{aligned}
\sum_{i\in[m]}\sum_{t\in[T]} z_{j,i,t} &= 1 \quad \forall j \in J \\
\sum_{j\in J} z_{j,i,t} &\leq 1 \quad \forall i \in [m] \;\; \forall t \in [T] \\
\sum_{t'<t}\sum_{i\in[m]} z_{j_1,i,t'} &\geq \sum_{t'\leq t}\sum_{i\in[m]} z_{j_2,i,t'} \quad \forall j_1 \prec j_2 \;\; \forall t \in [T] \\
\sum_{t\in I_s} z_{j,i,t} &= x_{j,i,s} \quad \forall j \in J \;\; \forall s \in \{0, \ldots, S-1\} \\
0 \leq z_{j,i,t} &\leq 1 \quad \forall j \in J, i \in [m], t \in [T]
\end{aligned}$$

Similar to the makespan LP, let $\tilde{Q}(r)$ be the set of feasible solutions $(x, y, z, C)$ to the following LP:

$$\begin{aligned}
\text{Minimize} \quad & \sum_{j\in J} w_j \cdot C_j \\
y_{j_1,j_2} &= \sum_{s\in\{0,\ldots,S-1\}}\sum_{i\in[m]} x_{(j_1,i,s),(j_2,i,s)} \\
C_{j_2} &\geq C_{j_1} + (1 - y_{j_1,j_2}) \cdot c \quad \forall j_1 \prec j_2 \\
C_j &= \sum_{i\in[m]}\sum_{t\in[T]} z_{j,i,t} \cdot t \quad \forall j \in J \\
(x, z) &\in SA_r(\tilde{K})
\end{aligned}$$

Note that the $C_j$ variables in this LP relaxation denote the actual completion time of

$j$ unlike their role in the makespan result, where they were used to indicate the interval in which $j$ was scheduled. The main technical result for this section is the following:

**Theorem 49.** *Consider an instance for* $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c \mid \sum_j w_j C_j$ *and a solution* $(x, y, z, C) \in \tilde{Q}(r)$ *with* $r \geq 5$. *Then there is a randomized algorithm with expected polynomial running time that finds a feasible schedule so that (i)* $\mathbb{E}[C_j^A] \leq O(\log c \cdot \log n) \cdot C_j$ *and (ii)* $C_j^A \leq O(\log c \cdot \log n) \cdot C_j + O(\log n) \cdot c$ *for all* $j \in J$, *where* $C_j^A$ *is the completion time of job* $j$.

We briefly describe how Theorem 49 implies the approximation algorithm promised in Theorem 29:

*Proof of Theorem 29.* Note that strictly speaking $\tilde{Q}(r)$ is not actually a relaxation of $\mathsf{P}\infty \mid \mathsf{prec}, p_j = 1, c \mid \sum_j w_j C_j$. However one can take an optimum integral schedule and insert $c$ idle time slots every $c$ time units and obtain a feasible solution for $\tilde{Q}(r)$. This increases the completion time of any job by at most a factor of 2. Then we set $r := 5$ and $m := n$ and solve the LP $\tilde{Q}(r)$ in time polynomial in $n$. Now consider the randomized schedule from Theorem 49 with completion times $C_j^A$. Then the expected objective function is $\mathbb{E}[\sum_{j \in J} w_j \cdot C_j^A] \leq O(\log n \cdot \log c) \cdot \left(\sum_{j \in J} w_j \cdot C_j\right)$. Markov's inequality guarantees that we can find in expected polynomial time a schedule that satisfies this inequality if we increase the right hand side by a constant factor. This completes the proof. □

### *4.6.2 The Rounding Algorithm*

Let $(x, y, z, C)$ be an optimal solution to the LP relaxation $\tilde{Q}(r)$ with $r \geq 5$. It remains to show Theorem 49. We partition the jobs based on their fractional completion times. For $\delta = \frac{c}{64 \log(4c)}$ and $k \geq 0$, let $J_k := \{j \in J : k \cdot \delta \leq C_j < k \cdot \delta\}$.

We give a separate algorithm for scheduling jobs in $J_0$ within an interval of length at most $O(\log n) \cdot c$. Now consider the remaining jobs. For $k = 1, 2, ...$, we schedule jobs in the set $J_k$ using the algorithm from Section 4.3.3,

inserting $c$ empty time slots between the schedule of jobs in the set $J_k$ and $J_{k+1}$. Let $C_j^A$ denote the completion time of job $j$ in our algorithm.

**Lemma 50.** *For $k \geq 1$, consider a job $j \in J_k$. Then deterministically $C_j^A \leq O(\log n \cdot \log c) \cdot C_j$.*

*Proof.* The claim follows from repeating the arguments in Lemma 42, so we only give a sketch here. Fix $k$ and consider scheduling the jobs in the set $J_k$ using the procedure described in Lemma 42, where we repeat the CKR clustering algorithm for $k = \{1, 2, \ldots 2\log n\}$ iterations. Then the expected number of jobs that did not get scheduled in the first $2\log n$ iterations is at most $\frac{|J_k|}{n^2} < 1$. Therefore, in expected polynomial time we can find a schedule such that $C_j^A \in [2\log n \cdot O(c) \cdot k, 2\log n \cdot O(c) \cdot (k+1)]$. From the definition of set $J_k$, the fractional completion time $C_j$ of every job $j$ in $J_k$ is at least $k \cdot \frac{c}{64\log(4c)}$ in the LP solution. This completes the proof. □

The only new ingredient for the completion time result is scheduling the jobs in the set $J_0$. For $j \in J_0$, let $t_j^*$ denote the earliest time instant $t$ at which the job is scheduled to a fraction of at least $1 - \varepsilon$ in the LP solution. Here $0 < \varepsilon < 1$ is a small constant that we determine later. In scheduling theory this time is also called *$\alpha$-point* with $\alpha = 1 - \varepsilon$. Formally

$$t_j^* := \min \left\{ t' \in [T] : \sum_{i=1}^{m} \sum_{t=1}^{t'} z_{j,i,t} \geq 1 - \varepsilon \right\} \tag{4.1}$$

We use the same semimetric $d(j_1, j_2) := 1 - y_{j_1,j_2}$ as in Section 4.3 and schedule jobs in $J_0$ using the following procedure.

SCHEDULE FOR $J_0$

(1) Run a CKR clustering on the semimetric space $(J_0, d)$ with parameter $\Delta := \frac{1}{12}$ and let $V_1, \ldots, V_k$ be the clusters.

(2) Let $V'_\ell := \{j \in V_\ell \mid (\Gamma^-(j) \cap J_0) \subseteq V_\ell\}$ for $\ell = 1, \ldots, k$.

(3) For all $\ell = 1, \ldots, k$ assign jobs in $V'_\ell$ on a single machine and schedule them in the increasing of order of $t^*_j$ values breaking ties in an arbitrary manner.

(4) Insert a gap of $c$ time slots.

(5) Let $J'_0 \subseteq J_0$ be the set of jobs that did not get scheduled in steps (1) - (3). Use Lemma 42 to schedule $J'_0$.

**Lemma 51.** *For a job $j_1 \in J_0$, the probability that $j_1$ gets scheduled in step (5) of the algorithm, i.e., $j_1 \in J'_0$, is at most $O(\log c) \cdot \frac{C_{j_1}}{c}$.*

*Proof.* The arguments are a slight refinement of Lemma 38. 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 $0 \leq C_{j_0} + c \cdot d(j_0, j_1) \leq C_{j_1}$ by the LP constraints and so $d(j_0, j_1) \leq \frac{C_{j_1}}{c}$. Then the diameter of $U$ with respect to semimetric $d$ is bounded by $2C_{j_1}$ and hence by Theorem 31.(b) the probability that $U$ is separated is bounded by $\ln(2|N(U, \Delta/2)|) \cdot \frac{4\text{diam}(U)}{\Delta} \leq O(\log c) \cdot \frac{C_{j_1}}{c}$. □

The next lemma follows from repeating the arguments in Lemma 50.

**Lemma 52.** *For a job $j \in J_0$, condition on the event that $j \in J'_0$. Then, $C^A_j|_{(j \in J'_0)} \leq O(\log n) \cdot c$.*

We can now prove that every cluster $V'_\ell$ can be scheduled on one machine so that the completion time of any job is at most twice the LP completion time.

**Lemma 53.** *For a small enough constant $\varepsilon > 0$ ($\varepsilon = \frac{1}{12}$ suffices) the following holds: Let $U \subseteq J$ be a set of jobs with diam$(U) \leq \varepsilon$ w.r.t. distance $d$. Define $t^*_j$ as in Eq (4.1) and schedule the jobs in $U$ in increasing order of $t^*_j$ on one machine and denote the completion*

*time of $j$ by $C_j^A$. Then, in expectation $C_j^A \leq 2t_j^*$ for every $j \in U$.*

*Proof.* Let us index the jobs in $U = \{j_1, \ldots, j_{|U|}\}$ so that $t^*_{j_1} \leq \ldots \leq t^*_{j_{|U|}}$. Suppose for the sake of contradiction that there is some job $j_N$ with $C^A_{j_N} > 2t^*_{j_N}$. Abbreviate $U^* := \{j_1, \ldots, j_N\}$ and $\theta^* := t^*_{j_N}$ so that $1 \leq t^*_j \leq \theta^*$ for $j \in U^*$. We observe that $|U^*| = \sum_{j \in U^*} p_j > 2\theta^*$. Then we have

$$(A) \sum_{j \in U^*} \sum_{i \in [m]} \sum_{t=1}^{\theta^*} z_{j,i,t} \geq (1-\varepsilon)|U^*|, \quad (B) \sum_{j \in U^*} y_{j,j_N} \geq (1-\varepsilon)|U^*|, \quad (C) \sum_{i \in [m]} \sum_{t=1}^{\theta^*} z_{j_N,i,t} \geq 1-\varepsilon$$

where $(A)$ and $(C)$ are by definition of $t^*_j$ and $(B)$ follows from $\mathrm{diam}(U^*) \leq \mathrm{diam}(U) \leq \varepsilon$. Intuitively, this means that we have $|U^*| > 2\theta^*$ many jobs that the LP schedules almost fully on slots $\{1, \ldots, \theta^*\}$ while (B) means that the jobs are almost fully scheduled on the same machine.

As before, we will use the properties of the Sherali-Adams hierarchy to formally derive a contradiction. We know by Lemma 34[3] that there is a distribution $(\tilde{x}, \tilde{z}, \tilde{y}) \sim \mathcal{D}(j_N)$ so that $\mathbb{E}[\tilde{x}_{j,i,s}] = x_{j,i,s}$, $\mathbb{E}[\tilde{z}_{j,i,t}] = z_{j,i,t}$ and $\mathbb{E}[\tilde{y}_{j_1,j_2}] = y_{j_1,j_2}$ while the variables involving job $j_N$ are integral, i.e. $\tilde{x}_{j_N,i,s}, \tilde{z}_{j_N,i,t} \in \{0,1\}$. Consider the three events

$$(A') \sum_{j \in U^*} \sum_{i \in [m]} \sum_{t=1}^{\theta^*} \tilde{z}_{j,i,t} \geq (1-3\varepsilon)|U^*|, \quad (B') \sum_{j \in U^*} \tilde{y}_{j,j_N} \geq (1-3\varepsilon)|U^*|, \quad (C') \sum_{i \in [m]} \sum_{t=1}^{\theta^*} \tilde{z}_{j_N,i,t} = 1.$$

Then by Markov inequality $\Pr[A'] \geq \frac{2}{3}$, $\Pr[B'] \geq \frac{2}{3}$ and $\Pr[C'] \geq 1 - \varepsilon$, and so by the union bound $\Pr[A' \wedge B' \wedge C'] > 0$, assuming $\varepsilon < \frac{1}{3}$. Fix an outcome for $(\tilde{x}, \tilde{y}, \tilde{z})$ where the events $A', B', C'$ happen and let $i_N \in [m], t_N \in \{1, \ldots, \theta^*\}$ be the indices with $\tilde{z}_{j_N,i_N,t_N} = 1$. Then the interval index with $t_N \in I_{s_N}$ satisfies $\tilde{x}_{j_N,i_N,s_N} = 1$. Hence

$$\begin{aligned}(1-3\varepsilon)|U^*| &\overset{(B')}{\leq} \sum_{j \in U^*} \tilde{y}_{j,j_N} \overset{LP}{=} \sum_{j \in U^*} \sum_{i \in [m]} \sum_{s \in \{0,\ldots,S-1\}} \tilde{x}_{(j,i,s),(j_N,i,s)} \overset{\tilde{x}_{j_N,i_N,s_N}=1}{=} \sum_{j \in U^*} \tilde{x}_{j,i_N,s_N} \\ &\overset{LP}{\leq} \underbrace{\sum_{j \in U^*} \sum_{t=1}^{\theta^*} \tilde{z}_{j,i_N,t}}_{\leq \theta^* \text{ by LP}} + \underbrace{\sum_{j \in U^*} \sum_{t > \theta^*} \tilde{z}_{j,i_N,t}}_{\leq 3\varepsilon|U^*| \text{ by } (A')} \leq \theta^* + 3\varepsilon|U^*|\end{aligned}$$

[3]Strictly speaking, Lemma 34 describes the SA properties for LP $Q(r)$, but an absolutely analogous statement holds for $\tilde{Q}(r)$.

Rearranging gives $|U^*| \leq \frac{1}{1-6\varepsilon}\theta^*$, which is a contradiction for $\varepsilon \leq \frac{1}{12}$. □

**Lemma 54.** *For a job $j \in J_0$, condition on the event that it got scheduled in the step (3) of the algorithm. Then, $C_j^A|_{(j \notin J_0')} \leq O(C_j)$.*

*Proof.* Consider a job $j \in J_0 \setminus J_0'$. Then $j \in V_\ell'$ and by construction, the set $V_\ell'$ has diameter at most $\Delta = \frac{1}{12}$ w.r.t. $d$. Then Lemma 53 guarantees that the completion time is $C_j^A \leq 2t_j^*$ where we set $\varepsilon = \frac{1}{12}$. Finally note that an $\varepsilon$-fraction of $j$ was finished at time $t_j^*$ or later and hence $C_j = \sum_{i\in[m]} \sum_{t\in[T]} z_{j,i,t} \cdot t \geq \varepsilon \cdot t_j^*$. Putting everything together we obtain $C_j^A \leq \frac{2}{\varepsilon} C_j$. □

We have everything to finish the proof of the completion time result.

*Proof of Theorem 49.* From Lemma 50, for $k \geq 1$ and $j \in J_k$, we have deterministically $C_j^A \leq O(\log n \cdot \log c) \cdot C_j$. Now consider a job $j \in J_0$. Then,

$$\begin{aligned}
\mathbb{E}[C_j^A] &= \mathbb{E}[C_j^A | (j \notin J_0')] \cdot \Pr[(j \notin J_0')] + \mathbb{E}[C_j^A | (j \in J_0')] \cdot \Pr[(j \in J_0')] \\
&\leq O(C_j) + O(\log c) \cdot \frac{C_j}{c} \cdot O(\log n) \cdot c \quad \text{(from Lemmas 51, 52, 54)} \\
&\leq O(\log n \cdot \log c) \cdot O(C_j)
\end{aligned}$$

Finally note that the completion time of a job $j \in J_0$ is always bounded by $C_j^A \leq O(\log n) \cdot c$. The claim follows. □

### *Discussion and Open Problems*

We gave a new framework for scheduling jobs with precedence constraints and communication delays based on metric space clustering. Our results take the first step towards resolving several important problems in this area. One immediate open question is to understand whether our approach can yield a constant-factor approximation for $\mathsf{P} \mid \mathsf{prec}, c \mid C_{\mathsf{max}}$. A more challenging problem is to handle *non-uniform* communication delays in the problem $\mathsf{P} \mid \mathsf{prec}, c_{jk} \mid C_{\mathsf{max}}$, where $c_{jk}$ is the communication delay between jobs $j \prec k$.

# Chapter 5

# Scheduling with Communication delays on Related Machines

## *5.1 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 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*

[1]We again adopt the convention of [GLLK79, VLL90], where the respective fields denote: **(1) machine environment:** $Q$ for related machines, $P$ for identical machines, **(2) job properties:** $\mathsf{prec}$ for precedence constraints; $c$ for communication delays of length $c$; $p_j = 1$ for unit length case, **(3) objective:** $C_{\mathsf{max}}$ for minimizing makespan, $\sum w_j C_j$ for minimizing weighted sum of completion times.

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, MH01, Thu92, 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] (included in Chapter 4) 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.

Maiti et al. [MRS+20] obtain a polylogarithmic approximation algorithm for scheduling in the presence of precedence and communication delay constraints in the *job duplication* model. Here, 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. then show 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. To create a strict contrast between our techniques and that of Maiti et al., we do not consider the duplication setting or any reduction from it. Additionally, the LP of Maiti et al. does not give rise to a *semimetric* on the set of jobs, which is a crucial

part of our scheduling algorithm.

The framework of Maiti et al. extends to the weighted sum of completion times objective, but it obtains an extra additive term in the approximation that may dominate the factor; we discuss this further in Section 5.1.2. Moreover, our previous results [DKR+20] (see also Chapter 4) 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 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 $O(\log m/\log\log m)\cdot \mathrm{OPT} + C$, 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 $c = 0$.

### *5.1.1 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 our paper [DKR+21] is the following:

**Theorem 55.** *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 56.** *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].

### *5.1.2 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 $\alpha$-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 $O\left((\log^5 n/\log\log n)\cdot \mathrm{OPT} + c\cdot\sum_j w_j\right)$, which only implies an approximation factor of $\max\{c, \log^5 n/\log\log n\}$.

### *5.1.3 Our Techniques and Algorithm Overview*

We obtain our results by generalizing the LP hierarchy framework introduced by [DKR+20][2] 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

[2] see also Chapter 4

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 earlier 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. Similar to [DKR$^+$20] and Chapter 4, 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 $\Delta$ such that any set $U \subseteq J$ has a probability of at most $O(\log(n) \cdot \frac{\operatorname{diam}(U)}{\Delta})$ 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] and Chapter 4. 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 $V'_\ell$ 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 $V'_\ell$ 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 $V'_\ell$. But we can prove a novel structural lemma that for each $V'_\ell$ 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 $V'_\ell$ 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 $\alpha$-*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 $\theta$, the number of jobs in $J'_0(i, k)$ with $\alpha$-point at most $\theta$ 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 $V'_\ell$ there is a speed class $k$ with $|V'_\ell| \leq O(s_k c)$ such that the LP solution schedules jobs in $V'_\ell$ 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 $V'_\ell$, 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 $\mathsf{Q} \mid \mathsf{prec}, c \mid \sum w_j C_j$.

### *5.1.4 Outline*

The rest of this chapter is organized as follows. In Section 5.2 we give preliminaries on hierarchies, semimetric spaces, and a useful version of the Chernoff bound. In Section 5.3 we discuss our linear program and its properties. In Sections 5.4 and 5.5 we give our algorithm for the weighted sum of completion times objective in the special case of unit-size jobs. Then, in Section 5.6, we show how to reduce the general processing time case to the unit-size case. Together, Sections 5.4–5.6 constitute the proof of Theorem 55. Finally, in Section 5.7 we solve the makespan minimization version while saving one $O(\log n)$ factor, thus proving Theorem 56.

## *5.2 Preliminaries*

Most contents of both subsections are contained in Section 4.2. For conveniece of readers, we still list them in this section for the sake of self-containing of the whole chapter.

### 5.2.1 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 6.** 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 57** (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] and Chapter 4. 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 57.(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 $\mathcal{E} =$ "$\{\exists i \in J : x_i = 1\}$" 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 58.** *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 57.(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 57.(e). □

Suppose that $\bar{y} \in SA_{r-1}(K)$ is the vector obtained by conditioning on the event $\mathcal{E} =$ "$\{\exists i \in J : x_i = 1\}$" according to Lemma 58 with $J \subseteq U_h$. Then we will also abbreviate the distribution $\mathcal{D}_{\bar{y}}(h')$ more conveniently as the conditional distribution $\mathcal{D}_y(h' \mid \mathcal{E})$.

### *5.2.2 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 chapter 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 $\Delta$. 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 $\beta$. 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 $\beta$, and independently selects a random ordering $\pi$. Here the ordering is to be understood so that element $v_\ell$ 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 Section 4.2.3.

**Theorem 59** (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*

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

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].

### *5.2.3 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 60.** *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* $\Pr[X > C\frac{\log N}{\log\log N} \cdot \alpha] \leq \frac{1}{N}$.

## *5.3 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 $s_k$.

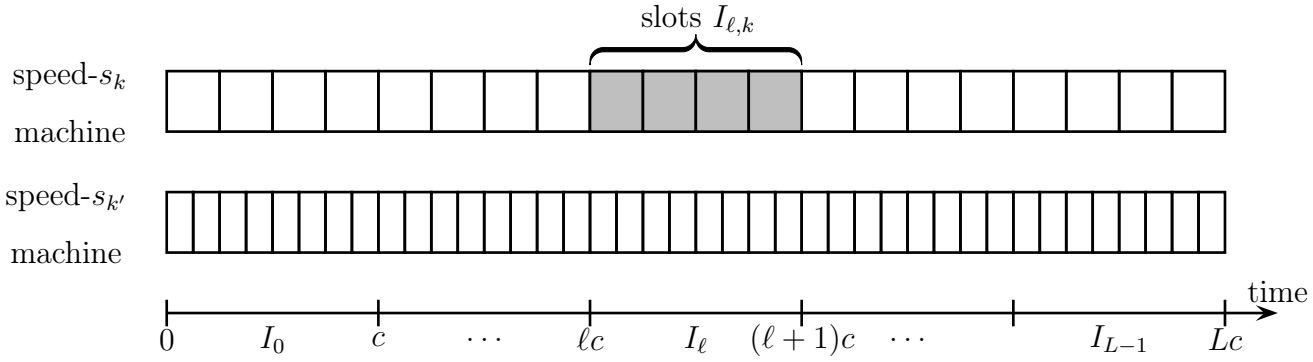



Figure 5.1: Slots for different machine types.

### *5.3.1 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 $I_\ell$ as $I_{\ell,k} := \{\ell \cdot c + \frac{t}{s_k} : t = 1, \ldots, s_k c\}$. 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:

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

We note that the assignment constraints are the set of equations $\sum_{k\in[M], i\in[m_k]} \sum_{t\in I_{*,k}} x_{j,i,t} = 1$ for every $j \in J$.

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:

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

$$y_{j_1,j_2} = \begin{cases} 1 & j_1 \text{ and } j_2 \text{ are scheduled on the same machine in the same interval,} \\ 0 & \text{otherwise,} \end{cases}$$

$$C_j = \text{completion time of job } j.$$

The LP relaxation is then as follows:

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

Following the discussion in Section 5.2.1, 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 and $(\tilde{x}, \tilde{y})$ satisfies the first two constraints in ($\mathsf{LP}$). 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 5.2.1 for details.

### *5.3.2 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 the paper.

**Lemma 61.** *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

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

□

This property was proven in [DKR+20] and Section 4.3.1 for a special case of $s_k = 1$ and is not hard to extend to our more general LP. From now on, the symbol $d$ as well as the quantity $\mathrm{diam}(\cdot)$ 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] and Section 4.3.1 that any set $U \subseteq J$ with $\mathrm{diam}(U) \leq \frac{1}{2}$ has size $|U| \leq 2c$. While this is obviously false for arbitrary speeds $s_k$, we can prove a similar claim:

**Lemma 62.** *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 $\sum_{i\in[m_k]}\sum_t x_{j_1,i,t} \le 1$, 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 $\ell_1$. Then indeed

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

□

Lemmas 63 and 64 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 63.** *Let $U \subseteq J$ be a non-empty subset of jobs with $diam(U) \le \frac{1}{4}$. Then there exists a $k^* \in [M]$ such that*

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

*Proof.* Let us sort the speed classes $[M]$ so that $s_1 \ge \ldots \ge s_M$ 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 $\sum_{k\le k_{\text{median}}} z_{j^*,k} \ge \frac{1}{2}$ and $\sum_{k\ge k_{\text{median}}} z_{j^*,k} \ge \frac{1}{2}$. We split the remaining proof into two separate claims.

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

**Proof of Claim I.** First we observe that for every $j \in U$ we have $\sum_{k\ge k_{\text{median}}} y_{j^*,j,k} \ge$

$\frac{1}{2} - d(j^*, j) \geq \frac{1}{4}$. We use this to estimate

$$\begin{aligned}\frac{|U|}{4} &\leq \sum_{k \geq k_{\text{median}}} \sum_{j \in U} y_{j^*,j,k} \overset{\text{Lem 62}}{\leq} \sum_{k \geq k_{\text{median}}} s_k c \sum_{i \in [m_k]} \sum_t x_{j^*,i,t} \\ &\leq s_{k_{\text{median}}} c \underbrace{\sum_{k \geq k_{\text{median}}} \sum_{i \in [m_k]} \sum_t x_{j^*,i,t}}_{\leq 1} \leq s_{k_{\text{median}}} c\end{aligned}$$

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 $\Omega(1/M)$ fraction of *every one of these jobs* on the same machine type. Recall that Lemma 63 only satisfied this condition on average.

**Lemma 64.** *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)* $|U| \leq 2 \cdot s_k \cdot c$ *for all* $k \in \pi(U)$.

*Proof.* We prove the points in order.

(i) Actually we claim the stronger property of $|z_{j_1,k} - z_{j_2,k}| \leq d(j_1, j_2)$. 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

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

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

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

(iii) We have

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

(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 $\Pr_{(\tilde{x},\tilde{y}) \sim \mathcal{D}(j^*)}[\mathcal{E}] = z_{j^*,k} \geq \frac{1}{4M}$. Now, let $(\bar{x}, \bar{y})$ be the Sherali-Adams lift (see Lemma 58) 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 $z_{j^*,k} \geq \frac{1}{4M}$, the chance of other events cannot increase by more than a factor of $4M$ (see again Lemma 58). In particular for $j \in U$ we have $1 - \bar{y}_{j^*,j} \leq 4M \cdot (1 - y_{j^*,j}) = 4M \cdot d(j^*, j) \leq \frac{1}{2}$. As $\bar{y}_{j^*,j,k'} = 0$ for all $k' \neq k$ and $j \in U$ we can conclude that $\bar{y}_{j^*,j,k} \geq \frac{1}{2}$ for all $j \in U$. Finally by Lemma 62 we know that $\sum_{j \in J} \bar{y}_{j^*,j,k} \leq s_k c$ which then gives $|U| \leq 2 s_k c$.

□

As it is always the case when using the Sherali-Adams hierarchy in the context of approximation algorithms, it is a valid question how much of the power of the hierarchy is really needed. Some properties such as the triangle inequality from Lemma 61 or the constraints from Lemma 62 could be enforced by simply adding them to the original LP. However at various places such as Lemmas 64, 67 and 68 our analysis makes heavily use of the local consistency provided by the hierarchy. It remains open whether there is a more efficient linear program, say only with the original variables $x_{i,j,t}, y_{j_1,j_2}, C_j$ that suffices.

### *5.4 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 65. 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 65.** *There is a polynomial-time randomized algorithm that, given a solution $(x, y, C)$ to (LP) for $r \geq 5$, produces a schedule with completion times $\{C_j^A\}_{j \in J}$ such that*

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

Before we describe our rounding algorithm, we set up some notation. Let $(x, y, C)$ be an optimal solution to the LP with $r \geq 5$. 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

$$J_0 := \{j : C_j < \delta \cdot c\} \tag{5.1}$$

and for $\gamma \geq 1$ define

$$J_\gamma := \left\{j : 2^{\gamma-1} \cdot \delta \cdot c \leq C_j < 2^\gamma \cdot \delta \cdot c\right\} \tag{5.2}$$

Our rounding algorithm has the following steps:

- Schedule $J_0$ via the first batch scheduling algorithm given in Section 5.4.1

- For $\gamma = 1, 2, \ldots$, schedule $J_\gamma$ via the intermediate scheduling algorithm given in Section 5.4.2.

- 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 65. We need the following scheduling subroutine for the rounding steps above.

**Theorem 66.** *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 5.5.

### *5.4.1 First Batch Scheduling*

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

$$t_j^* := \min \left\{ t' \in [T] : \sum_{i=1}^{m} \sum_{t \leq t'} x_{j,i,t} \geq \alpha \right\}. \tag{5.3}$$

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 64. 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 $V'_\ell$ 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 $\alpha$-points for $\alpha = (1 - \frac{1}{100M})$ as defined in Eq. (5.3).

(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 66 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 67.** *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 (5.3). For $\theta > 0$, consider the set of jobs $U^* := \{j \in U : t^*_j < \theta\}$ with $\alpha$-point less than $\theta$. 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 $\theta$ 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

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

$$(C)\ \sum_{i\in[m_{k^*}]}\sum_{t\leq\theta} x_{j^*,i,t} \geq \frac{1}{5M}.$$

where $(A)$ follows from the definition of $\alpha$-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 58 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 $\theta$.

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 58). 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, $\sum_{j\in U^*}\sum_{i\in[m]}\sum_{t>\theta}\bar{x}_{j,i,t} \leq 5M\cdot(\sum_{j\in U^*}\sum_{i\in[m]}\sum_{t>\theta} x_{j,i,t}) \leq 5M\cdot\frac{1}{100M}\cdot|U^*| \leq \frac{|U^*|}{10}$. 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 $i^* \in [m_{k^*}], t^* \in \{1,\ldots,\theta\}$ be the indices with $\tilde{x}_{j^*,i^*,t^*} = 1$. Let

$\ell^*$ 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 68.** *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 $X_1, \dots, X_q$ are independent and $X = |\{j \in J_0'(i^*,k) : t_j^* \leq \theta^*\}|$. From Lemma 67 we know that $X_\ell \leq 2 s_k \cdot \theta^*$ for all $\ell$.

Next, we estimate $\mathbb{E}[X]$. Recall that the algorithm uses the indices $\pi(V_\ell') \subseteq [M]$ from Lemma 64 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 $\ell$ 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 $\ell$ 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 $\alpha$-points and the assumption $t^*_j \leq \theta^*$. Then

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

Now we can bound the expectation of our random variable as

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

Here we also have used that $\sum_{k'\in[M]} \mu_{\ell,k'} \geq \frac{1}{2}$. As $X$ is sum of independent random variables $X_\ell$ each of which is bounded by $O(s_k \cdot \theta^*)$, we apply the Chernoff bound from Lemma 60 and obtain that $\Pr[X > C' \cdot \frac{\log n}{\log\log n} \cdot s_k\theta^*] \leq 1/n^{1000}$ for some constant $C' > 0$. To complete the lemma, we simply do a union bound over all possible values of $\theta^*$ and $i$. The number

of machines is $m \leq n$ and the number of relevant $\theta^*$'s in the time horizon is bounded by $\sum_{k\in[M]} |I_{*,k}| \leq M \cdot 2n$. □

**Lemma 69.** *For every job $j \in J'_0$, with high probability, the completion time of $j$ in our schedule $C_j^A \leq O(M \cdot \frac{\log n}{\log\log n} \cdot C_j)$.*

*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 68, there are at most $O(\frac{\log n}{\log\log n} \cdot s_k \cdot t^*_{j^*})$ jobs whose $t^*_j \leq t^*_{j^*}$. Therefore, the completion time is $C^A_{j^*} \leq O(\frac{\log n}{\log\log n} \cdot t^*_{j^*})$. Note that in the LP solution at least an $\frac{1}{100M}$-fraction of $j^*$ was scheduled at or after $t^*_{j^*}$. Hence, $C_j \geq \frac{1}{100M} \cdot t^*_{j^*}$. Putting things together we conclude $C^A_j \leq O(\frac{\log n}{\log\log n} \cdot t^*_{j^*}) \leq O(M \cdot \frac{\log n}{\log\log n} \cdot C_j)$. □

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

**Lemma 70.** *For every job $j \in J''_0$, the completion time of $j$ in our schedule $C_j^A \leq O(\log n \cdot c)$.*

*Proof.* From the definition of set $J_0$, for all $j \in J''_0$ the completion time $C_j \leq c \cdot \delta$ 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 66 to construct a schedule of jobs $J''_0$. Theorem 66 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 $\delta$. 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 69 every job in $J'_0$ has completion time at most $O(\log n \cdot c)$. Thus the lemma follows. □

**Lemma 71.** *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 $0 \leq C_{j_0} + c \cdot d(j_0, j_1) \leq C_{j_1}$ by the LP constraints and so $d(j_0, j_1) \leq \frac{C_{j_1}}{c}$. Then the diameter of $U$ with respect to semimetric $d$ is bounded by $2C_{j_1}/c$ and hence by Theorem 59.(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 72.** *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 69, 70, 71. □

We also note the following simple consequence of previous lemmas:

**Lemma 73.** *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 $O(\log n \cdot c)$.*

### *5.4.2 Intermediate Batch Scheduling*

Now we describe our algorithm to schedule jobs in the set $J_\gamma$ for $\gamma > 0$, which is similar to that of scheduling jobs in $J_0''$. Recall that from the definition of set $J_\gamma$ in Eq. (5.2), $2^{\gamma-1}\delta \cdot c < C_j \leq 2^\gamma \delta \cdot c$ in the LP solution. We take the LP solution restricted to $J_\gamma$ in the interval $[0, 2^\gamma \delta \cdot c]$, and invoke Theorem 66 to construct a schedule of jobs $J_\gamma$. Theorem 66 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 74.** *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 $J_\gamma$. 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 $T_\gamma$ denote the makespan of $\mathcal{S}(J_\gamma)$. From Lemma 73 and

Theorem 66, $T_\gamma \le O(2^\gamma \cdot \delta \cdot c \cdot \log m \cdot \log n) + O(\log n \cdot c)$. Fix the job $j^* \in J_{\gamma^*}$ with the highest completion time in $\mathcal{S}$. We can bound the completion time of $j^*$ as

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

which completes the proof. □

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

*Proof of Theorem 65.* For every job $j \in J_0$, from Lemma 72, we have $\mathbb{E}[C^A_j] \le O(M \cdot \log^2 n) \cdot C_j$. From the previous Lemma 74, for $\gamma > 0$ and $j \in J_\gamma$ we have, $C^A_j \le O(\gamma \cdot \log n \cdot c) + O(2^\gamma \cdot \delta \cdot c \cdot \log m \cdot \log n)$. On the other hand, for every $j \in J_\gamma$, by definition, $C_j > 2^\gamma \cdot \delta \cdot c$. Thus, $C^A_j \le O(M \cdot \log^2 n) \cdot C_j$. □

## 5.5 Proof of Theorem 66

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

### 5.5.1 Main Scheduling Subroutine

The following lemma is the main scheduling subroutine towards proving Theorem 66.

**Lemma 75.** *Let $C^* \ge 0$ and $\delta = 1/64 \log(2n)$. Consider the set $J^* \subseteq \{j \in J' \mid C^* \le C_j \le C^* + \delta \cdot c\}$. 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 $V'_\ell$ to type $k^*$ satisfying the conditions in Lemma 63. |
| (4) For all $\ell = 1, \ldots, q$, if $V'_\ell$ is assigned to type $k^*$, assign all jobs in $V'_\ell$ 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 76.** *For all $\ell = 1, \ldots, q$, one has $|V'_\ell| \leq O(s_{k^*}c)$. Here, $k^*$ is the machine type that $V'_\ell$ is assigned in our algorithm. Thus, it takes $O(c)$ time to process all jobs in $V'_\ell$.*

*Proof.* We know by Theorem 59 that $\operatorname{diam}(V'_\ell) \leq \operatorname{diam}(V_\ell) \leq \Delta \leq \frac{1}{4}$. The lemma follows by Lemma 63 and by our choice of $k^*$. □

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

**Lemma 77.** *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 $V'_\ell$. □

We will use the two statements above together with Theorem 59 to prove Lemma 75.

*Proof of Lemma 75.* Call the schedule produced by the above algorithm $\mathcal{S}(J^*)$. By Lemma 76, for all $\ell = 1, \ldots, q$, the processing time of $V'_\ell$ is at most $O(c)$. By Lemma 77, 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 $V'_\ell$, 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 $2\delta$. Now we appeal to Lemma 63. For our choice of $\Delta = 1/4$ and $\delta \leq \frac{1}{64 \log(2n)}$, we apply Theorem 59 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 78.** *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 75. 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 75 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. □

### *5.5.2 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 5.5.1.

(5) Insert $c$ new empty idle slots.

*Proof of Theorem 66.* 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 78, and $J^{(k)}=\cup_{h=0}^{\frac{T-1}{c\cdot\delta}}J_h^{(k)}$. From Lemma 63 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 78 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.

### 5.6 *A Reduction from* $Q\mid \text{prec}, c\mid\sum w_jC_j$ *to* $Q\mid \text{prec}, p_j=1, c\mid\sum w_jC_j$

An extension of the classic *list scheduling* algorithm of Graham [Gra66b] 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 $\infty$ 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 [Gra66b].

**Lemma 79.** *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*

$$\sum_{k=1}^{M} D_k + \max\left\{\Big(\sum_{j\in C}\frac{p_j}{s_{\pi(j)}}\Big) + (|C|-1)\cdot c \mid C \subseteq J \text{ is a chain}\right\} \tag{5.4}$$

*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 $j_\ell$ is scheduled and let $[t_{j_\ell}, t_{j_\ell} + \frac{p_{j_\ell}}{s_{\pi(j_\ell)}})$ be the time interval in which $j_\ell$ is scheduled. We can make the observation that by the time $t_{j_{\ell+1}} + c$, all predecessors of $j_\ell$ have been completed and moreover a time of $c$ has passed. Hence if the period between $t_{j_{\ell+1}} + c$ and $t_{j_\ell}$ 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 (5.4). The claim is then proven. □

This can be combined with an assignment argument:

**Lemma 80.** *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. $\pi$ 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 $\pi$. □

We use this for the main reduction:

**Theorem 81.** *Suppose there is an $\alpha$-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 $\alpha$-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 $\sigma$ 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 $\sum_{j \in J} w_j C_j \leq \alpha \mathrm{OPT}$ 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 $\sigma^*$ 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 80 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 $C \subseteq \tilde{J}$. □

### *5.7 Makespan Minimization on Related Machines*

We now prove our result on makespan minimization on related machines, Theorem 56. The proof essentially follows from first using Theorem 66 to obtain a schedule of cost $O(\log^2 n)$ for the unit length case and then appealing to the reduction in Theorem 81, 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 66 and Theorem 81 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 66 and Theorem 81 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](see also Chapter 4), 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 66 to obtain a schedule $\mathcal{S}$ with cost at most $O(\log m \cdot \log n) \cdot 2T(\mathcal{I})$. Finally we use Theorem 81 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 81, 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.

□

# Bibliography


[AFS08] Arash Asadpour, Uriel Feige, and Amin Saberi. Santa claus meets hypergraph matchings. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008, Boston, MA, USA, August 25-27, 2008. Proceedings*, pages 10–20, 2008.

[AFS12] Arash Asadpour, Uriel Feige, and Amin Saberi. Santa claus meets hypergraph matchings. *ACM Trans. Algorithms*, 8(3):24:1–24:9, July 2012.

[AKS15] Chidambaram Annamalai, Christos Kalaitzis, and Ola Svensson. Combinatorial algorithm for restricted max-min fair allocation. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1357–1372, 2015.

[AMMS08] Christoph Ambühl, Monaldo Mastrolilli, Nikolaus Mutsanas, and Ola Svensson. Precedence constraint scheduling and connections to dimension theory of partial orders. In *Bulletin of the European Association for Theoretical Computer Science (EATCS*. Citeseer, 2008.

[Ann16] Chidambaram Annamalai. Finding perfect matchings in bipartite hypergraphs. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1814–1823, 2016.

[AS10] Arash Asadpour and Amin Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. *SIAM Journal on Computing*, 39(7):2970–2989, 2010.

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

[BD05] Ivona Bezáková and Varsha Dani. Allocating indivisible goods. *ACM SIGecom Exchanges*, 5(3):11–18, 2005.

[BGK96] Evripidis Bampis, Aristotelis Giannakos, and Jean-Claude König. On the complexity of scheduling with large communication delays. *European Journal of Operational Research*, 94(2):252 – 260, 1996.

[BK10] Nikhil Bansal and Subhash Khot. Inapproximability of hypergraph vertex cover and applications to scheduling problems. In *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part I*, volume 6198 of *Lecture Notes in Computer Science*, pages 250–261. Springer, 2010.

[BS06] Nikhil Bansal and Maxim Sviridenko. The santa claus problem. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, STOC '06, pages 31–40, New York, NY, USA, 2006. ACM.

[CC91] Jean-Yves Colin and Philippe Chrétienne. C.p.m. scheduling with small communication delays and task duplication. *Operations Research*, 39(4):680–684, 1991.

[CCK09] Deeparnab Chakrabarty, Julia Chuzhoy, and Sanjeev Khanna. On allocating goods to maximize fairness. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 107–116, 2009.

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

[CM18] Siu-Wing Cheng and Yuchen Mao. Restricted max-min fair allocation. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 37:1–37:13, 2018.

[CM19] Siu-Wing Cheng and Yuchen Mao. Restricted max-min allocation: Approximation and integrality gap. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece.*, pages 38:1–38:13, 2019.

[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] Mosharaf Chowdhury, Matei Zaharia, Justin Ma, Michael I. Jordan, and Ion Stoica. Managing data transfers in computer clusters with orchestra. In *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM ?11, page 98?109. 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.

[DKR+21] Sami Davies, Janardhan Kulkarni, Thomas Rothvoss, Jakub Tarnawski, and Yihao Zhang. Scheduling with communication delays via lp hierarchies and clustering ii: weighted completion times on related machines. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2958–2977. SIAM, 2021.

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

[Dro09] Maciej Drozdowski. *Scheduling with Communication Delays*, pages 209–299. Springer London, London, 2009.

[DRZ20] Sami Davies, Thomas Rothvoss, and Yihao Zhang. A tale of santa claus, hypergraphs and matroids. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2748–2757. SIAM, 2020.

[EKS08] Tomás Ebenlendr, Marek Krcál, and Jirí Sgall. Graph balancing: a special case of scheduling unrelated parallel machines. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 483–490, 2008.

[ES99] Leah Epstein and Jiří Sgall. Approximation schemes for scheduling on uniformly related and identical parallel machines. In *European Symposium on Algorithms*, pages 151–162. Springer, 1999.

[Fei08] Uriel Feige. On allocations that maximize fairness. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 287–293, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[FKK+14] Zachary Friggstad, Jochen Könemann, Young Kun-Ko, Anand Louis, Mohammad Shadravan, and Madhur 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] Jittat Fakcharoenphol, Satish Rao, and Kunal 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.

[GK07] Rodolphe Giroudeau and Jean-Claude König. Scheduling with communication delays. In Eugene Levner, editor, *Multiprocessor Scheduling*, chapter 4. IntechOpen, Rijeka, 2007.

[GKMP08] Rodolphe Giroudeau, Jean-Claude Konig, Farida Kamila Moulai, and Jérôme Palaysi. Complexity and approximation for precedence constrained scheduling problems with large communication delays. *Theoretical Computer Science*, 401(1):107 – 119, 2008.

[GLLK79] Ronald Lewis Graham, Eugene Leighton Lawler, Jan Karel Lenstra, and AHG Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Ann. Discrete Math.*, 4:287–326, 1979.

[Gra66a] Ronald L Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45(9):1563–1581, 1966.

[Gra66b] Ronald 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.

[Hax95] Penny E. Haxell. A condition for matchability in hypergraphs. *Graphs and Combinatorics*, 11(3):245–248, 1995.

[HCB+19] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *Advances in Neural Information Processing Systems*, 2019.

[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] JA Hoogeveen, Jan Karel Lenstra, and Bart Veltman. Three, four, five, six, or the complexity of scheduling with communication delays. *Operations Research Letters*, 16(3):129–137, 1994.

[HS87a] Dorit S Hochbaum and David B Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *Journal of the ACM (JACM)*, 34(1):144–162, 1987.

[HS87b] Dorit S. Hochbaum and David B. Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *J. ACM*, 34(1):144–162, 1987.

[HSS11] Bernhard Haeupler, Barna Saha, and Aravind Srinivasan. New constructive aspects of the lovász local lemma. *Journal of the ACM (JACM)*, 58(6):1–28, 2011.

[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.

[JKS93] Hermann Jung, Lefteris M Kirousis, and Paul Spirakis. Lower bounds and efficient algorithms for multiprocessor scheduling of directed acyclic graphs with communication delays. *Information and Computation*, 105(1):94 – 104, 1993.

[JR17] Klaus Jansen and Lars Rohwedder. On the configuration-lp of the restricted assignment problem. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM*

*Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2670–2678, 2017.

[JR18a] Klaus Jansen and Lars Rohwedder. Compact LP relaxations for allocation problems. In *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New Orleans, LA, USA*, pages 11:1–11:19, 2018.

[JR18b] Klaus Jansen and Lars Rohwedder. Local search breaks 1.75 for graph balancing. *CoRR*, abs/1811.00955, 2018.

[JR18c] Klaus Jansen and Lars Rohwedder. A note on the integrality gap of the configuration LP for restricted santa claus. *CoRR*, abs/1807.03626, 2018.

[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.

[KLTY20] Janardhan Kulkarni, Shi Li, Jakub Tarnawski, and Minwei Ye. *Hierarchy-Based Algorithms for Minimizing Makespan under Precedence and Communication Constraints*, pages 2770–2789. 2020.

[KMN11] Anna R. Karlin, Claire Mathieu, and C. Thach Nguyen. Integrality gaps of linear and semi-definite programming relaxations for knapsack. In *Integer Programming and Combinatoral Optimization - 15th International Conference, IPCO 2011, New York, NY, USA, June 15-17, 2011. Proceedings*, pages 301–314, 2011.

[Lau03] Monique 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.

[LLKS93] Eugene L Lawler, Jan Karel Lenstra, Alexander HG Rinnooy Kan, and David B Shmoys. Sequencing and scheduling: Algorithms and complexity. *Handbooks in operations research and management science*, 4:445–522, 1993.

[LR99] Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, November 1999.

[LR02] Renaud Lepere and Christophe Rapine. An asymptotic o(ln $\rho$/ ln ln $\rho$)-approximation algorithm for the scheduling problem with duplication on large communication delay graphs. In Helmut Alt and Afonso Ferreira, editors, *STACS 2002*, pages 154–165, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[LR16] Elaine Levey and Thomas Rothvoss. A (1+epsilon)-approximation for makespan scheduling with precedence constraints using LP hierarchies. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 168–177, 2016.

[LRK78] Jan Karel Lenstra and AHG Rinnooy Kan. Complexity of scheduling under precedence constraints. *Oper. Res.*, 26(1):22–35, February 1978.

[LST87] Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. In *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*, pages 217–224, 1987.

[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.

[MH97] Alix Munier and Claire Hanen. Using duplication for scheduling unitary tasks on m processors with unit communication delays. *Theoretical Computer Science*, 178(1):119 – 127, 1997.

[MH01] Alix Munier and Claire Hanen. An approximation algorithm for scheduling dependent tasks on m processors with small communication delays. *Discrete Applied Mathematics*, 108(3):239 – 257, 2001.

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

[MN06] Manor Mendel and Assaf 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] Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil Devanur, Greg Ganger, Phil Gibbons, and Matei Zaharia. Pipedream: Generalized pipeline parallelism for dnn training. In *Proc. 27th ACM Symposium on Operating Systems Principles (SOSP)*, Huntsville, ON, Canada, October 2019.

[Pin18] Michael L. Pinedo. *Scheduling: theory, algorithms, and systems*. Springer, 2018.

[PST04] Kirk Pruhs, Jirí Sgall, and Eric Torng. Online scheduling. In Joseph Y.-T. Leung, editor, *Handbook of Scheduling - Algorithms, Models, and Performance Analysis*. Chapman and Hall/CRC, 2004.

[PY90] Christos H. Papadimitriou and Mihalis Yannakakis. Towards an architecture-independent analysis of parallel algorithms. *SIAM J. Comput.*, 19(2):322–328, 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.

[Rot11] Thomas Rothvoß. Directed steiner tree and the lasserre hierarchy. *CoRR*, abs/1111.5473, 2011.

[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.

[Sve09] Ola Svensson. *Approximability of some classical graph and scheduling problems*. PhD thesis, Università della Svizzera italiana, 2009.

[Sve10] Ola Svensson. Conditional hardness of precedence constrained scheduling on identical machines. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC '10, pages 745–754, New York, NY, USA, 2010. Association for Computing Machinery.

[Sve11] Ola Svensson. Santa claus schedules jobs on unrelated machines. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 617–626, 2011.

[SW99] Petra Schuurman and Gerhard 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.

[Thu92] Ramakrishna Thurimella. A scheduling principle for precedence graphs with communication delay. *International Conference on Parallel Processing*, 3:229–236, 1992.

[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.

[VLL90] Bart Veltman, BJ Lageweg, and Jan Karel Lenstra. Multiprocessor scheduling with communication delays. *Parallel Computing*, 16(2):173 – 182, 1990.

[VW11] José Verschae and Andreas Wiese. On the configuration-lp for scheduling on unrelated machines. In *Algorithms - ESA 2011 - 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings*, pages 530–542, 2011.

[Woe97] Gerhard J Woeginger. A polynomial-time approximation scheme for maximizing the minimum machine completion time. *Operations Research Letters*, 20(4):149–154, 1997.

[Woe00] Gerhard J Woeginger. When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (fptas)? *INFORMS Journal on Computing*, 12(1):57–74, 2000.

[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.