Compact LP Relaxations for Allocation Problems*

Klaus Jansen[1] and Lars Rohwedder[2]

1 **Christian-Albrechts-Universität, Kiel, Germany**
`kj@informatik.uni-kiel.de`
2 **Christian-Albrechts-Universität, Kiel, Germany**
`lro@informatik.uni-kiel.de`

Abstract

We consider the restricted versions of Scheduling on Unrelated Machines and the Santa Claus problem. In these problems we are given a set of jobs and a set of machines. Every job j has a size p_j and a set of allowed machines $\Gamma(j)$, i.e., it can only be assigned to those machines. In the first problem, the objective is to minimize the maximum load among all machines; in the latter problem it is to maximize the minimum load. For these problems, the strongest LP relaxation known is the configuration LP. The configuration LP has an exponential number of variables and it cannot be solved exactly unless P = NP.

Our main result is a new LP relaxation for these problems. This LP has only $O(n^3)$ variables and constraints. It is a further relaxation of the configuration LP, but it obeys the best bounds known for its integrality gap (11/6 and 4).

For the configuration LP these bounds were obtained using two local search algorithm. These algorithms, however, differ significantly in presentation. In this paper, we give a meta algorithm based on the local search ideas. With an instantiation for each objective function, we prove the bounds for the new compact LP relaxation (in particular, for the configuration LP). This way, we bring out many analogies between the two proofs, which were not apparent before.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Linear programming, unrelated machines, makespan, max-min, restricted assignment, santa claus

Digital Object Identifier 10.4230/OASIcs.SOSA.2018.11

1 Introduction

We consider the problem of allocating jobs $\mathcal{J}$ to machines $\mathcal{M}$. A popular variation is the restricted case, where $j \in \mathcal{J}$ has a size p_j and can only be assigned to $\Gamma(j) \subseteq \mathcal{M}$. Two natural objective functions are to minimize the maximum load or to maximize the minimum load among all machines, where the load of a machine is defined as the sum of the sizes over the jobs assigned to it. The first objective will be referred to as Makespan and the latter as Max-min. These problems are special cases of Scheduling on Unrelated Machines and the Santa Claus problem.

Recent breakthroughs in both problems can be attributed to the study of the exponential size configuration LP which started with [4]. It was shown for the Max-Min problem that the LP has an integrality gap of at most 4 [3], which was the first constant factor guarantee there. Later, Svensson transferred these ideas to the Makespan problem and proved an upper bound of 33/17 for the integrality gap in this case [11], thereby giving the first improvement over

* Research was supported by German Research Foundation (DFG) project JA 612/15-2

the classical 2-approximation (see [9]). This has since been improved to 11/6 by us [7]. The known lower bounds for the approximation ratio assuming P $\neq$ NP are are 3/2 (Makespan) [9] and 2 (Max-min) [5]. This matches the known instances with the highest integrality gap for the configuration LP. Such instances can be derived by easy modification of the lower bounds given in this paper later on. Note that all of the upper bounds mentioned above are non-constructive, i.e., they do not give an efficient algorithm to compute the integral solution of the respective quality. A significant amount of research has gone into making these proofs constructive [10, 2, 8, 1], but this is not the focus of this paper.

In this paper, we show that these upper bounds can already be achieved by a weaker LP relaxation, which has polynomial size. First, we show a necessary condition for the existence of a fractional solution of a certain value. Then, we use this condition together with a local search algorithm similar to those used to prove the bounds for the configuration LP. This local search algorithm might not terminate in polynomial time; hence the result is non-constructive. We present the local search algorithms and their analysis in a unified way for both problems. In previous literature, many similarities between the problems were hidden by the different presentations of the algorithms. The algorithm in this paper is given in a natural way like in some physical system and uses much less technical definitions than in the previous papers. Starting with an arbitrary allocation, jobs are repelled or attracted by certain machines. This depends for example on whether a machine has too much or too little load. Based on these rules, they are moved away from machines by which they are repelled or towards machines by which they are attracted. The allocation eventually converges to one that has the desired properties. This is also significant change in the technical aspects, in particular compared to the previous algorithm for Max-min. There, jobs were always considered in large sets and such a set can only be exchanged altogether for a disjoint set of jobs. Our approach is more fine-grained by arguing over the jobs individually, which is also how it was traditionally done in the Makespan case.

One advantage of the small linear program is that it is much simpler to solve. An optimal solution can be computed directly and efficiently by an LP solver. The configuration LP on the other hand cannot be solved exactly in polynomial time, unless P=NP (see Appendix A), and even approximating it requires non-trivial techniques. A detailed description can be found in [4].

Furthermore, this paper improves the understanding of the configuration LP by pointing out which properties are necessary and which are not to obtain the currently known bounds for the integrality gap. This points to aspects of the configuration LP that should be investigated in order to reduce the bound on the integrality gap further. The compact linear program we give in this paper works by enforcing some properties of the configuration LP only on jobs greater than a certain threshold. It is intuitive that the integrality gap approaches that of the configuration LP as this threshold tends to 0, but it is surprising that we get the exact same bounds and this even with a rather large threshold.

It is also a direction of further research to investigate if efficient rounding procedures for the weaker linear program exist, since now it is clear that it has the potential for them.

Notation

Throughout the paper we will encounter numerous occasions, where one inequality (e.g., $a \leq b$) holds for Makespan objective and the opposite holds for Min-max (e.g., $a \geq b$). To save space, we will write $a \leq (\geq)\ b$ in that case. The first symbol always refers to the Makespan objective and the latter one to Max-min.

For a set of jobs $A \subseteq \mathcal{J}$, we write $p(A)$ in place of $\sum_{j \in A} p_j$. For other variables indexed by jobs, we may do the same. An allocation is a function $\sigma : \mathcal{J} \to \mathcal{M}$, where $\sigma(j) \in \Gamma(j)$ for all $j \in \mathcal{J}$. We write $\sigma^{-1}(i)$ for the set of all jobs j which have $\sigma(j) = i$.

1.1 Linear programming relaxations

All of the LPs presented below do not have an objective function. Instead, they are parameterized by a value $T \in [0, n \cdot \max_{j \in \mathcal{J}} p_j]$ and the optimum is the lowest T (Makespan) or highest T (Max-min) for which the LP is feasible. If the LP can be solved in polynomial time, such a T can be found in polynomial time using a binary search.

First, we define the allocation polytope, which captures every legal (fractional) allocation of jobs.

▶ **Definition 1** (Allocation polytope)**.**

$$\sum_{i \in \Gamma(j)} x_{i,j} \geq (\leq)\ 1 \qquad \forall j \in \mathcal{J} \tag{1}$$

$$\sum_{i \notin \Gamma(j)} x_{i,j} = 0 \qquad \forall j \in \mathcal{J} \tag{2}$$

$$x_{i,j} \in [0,1] \qquad \forall j \in \mathcal{J}, i \in \mathcal{M}$$

Here the variable $x_{i,j}$ specifies if job j is assigned to machine i. Note that it would perhaps be more intuitive to enforce equality in (1), but this would make certain upcoming arguments more lengthy than necessary. In general, a solution that does not satisfy equality can be converted without loss to one that does.

It remains to add constraints that guarantee every machine has a load of at most T (Makespan) or at least T (Max-min). We give these constraints in an indirect form. This is to improve comparability between these relaxations. In Section 2 we will give an explicit version of LP_r, which is the polynomial linear program this paper focuses on.

▶ **Definition 2** (Assignment LP)**.** The straight forward method is to ensure for all $i \in \mathcal{M}$ that $\sum_{j \in \mathcal{J}} p_j x_{i,j} \leq (\geq)\ T$ holds. This can also be written as

$$(x_{i,j})_{j \in \mathcal{J}} \in \{\chi \in [0,1]^{\mathcal{J}} : p^T \chi \leq (\geq)\ T\} \qquad \forall i \in \mathcal{M}. \tag{3}$$

This basic relaxation goes back to [9]. For Makespan it has an integrality gap of exactly 2; for Max-min the integrality gap is unbounded.

▶ **Definition 3** (Configuration LP)**.** For the configuration LP it is required that the assignment of jobs to a particular machine is a convex combination of so-called configurations (sets of jobs that do not exceed T in size or have size of at least T), i.e.,

$$(x_{i,j})_{j \in \mathcal{J}} \in \mathrm{conv}\{\chi \in \{0,1\}^{\mathcal{J}} : p^T \chi \leq (\geq)\ T\} \qquad \forall i \in \mathcal{M}. \tag{4}$$

Note that it is not necessary to require $\chi_j = 0$ for all $j \in \mathcal{J}$ with $i \notin \Gamma(j)$ (which is typical for the definition of the configurations), since this is already implied by the allocation polytope. The common definition of the configuration LP uses an exponential number of variables. Wiese and Verschae observed that the definition above is equivalent [12]. Clearly these constraints imply those from the assignment LP. Hence, the configuration LP is the stronger of the two.

▶ **Definition 4** (LP_r)**.** As a natural intermediate between assignment LP and configuration LP, for a constant $r \in \mathbb{N}_0$ we propose the following constraint.

$$(x_{i,j})_{j\in\mathcal{J}} \in \mathrm{conv}\{\chi \in [0,1]^{\mathcal{J}} : p^T\chi \leq (\geq)\ T, \chi_j \in \{0,1\} \text{ if } p_j \cdot r > T\} \qquad \forall i \in \mathcal{M}. \tag{5}$$

To our best knowledge, the cases where $0 < r < \infty$ have not been considered in literature.

1.2 Other related work

The Graph Balancing problem is the special case of Makespan minimization where $|\Gamma(j)| = 2$ for all $j \in \mathcal{J}$. For this problem a strong polynomial size LP relaxation is already known. This is the assignment LP with the additional constraint that $\sum_{j\in\mathcal{J}:p_j>T/2} x_{i,j} \leq 1$ for all $i \in \mathcal{M}$. It was shown to have an integrality gap of exactly 1.75 for Graph Balancing, but for arbitrary restrictions it only gives 2 [6]. This LP can be written as the points in the allocation polytope that satisfy

$$(x_{i,j})_{j\in\mathcal{J}} \in \mathrm{conv}\{\chi \in [0,\infty)^{\mathcal{J}} : p^T\chi \leq T \text{ and } \chi_j \in \{0,1\} \text{ if } p_j \cdot 2 > T\} \qquad \forall i \in \mathcal{M}.$$

In this form we see clearly the similarities to LP_2. Interestingly, LP_2 remains strong even for arbitrary restrictions.

1.3 Our contribution

▶ **Theorem 5.** *For $r \geq 2$ there is a linear program with $O(n^{r+1})$ variables and constraints (Makespan) or $O(n^{r+2})$ (Max-min) that is equivalent to LP_r, where $n = |\mathcal{J}| + |\mathcal{M}|$.*

▶ **Theorem 6.** *LP_2 for Makespan has an integrality gap between $10/6$ and $11/6$.*

▶ **Theorem 7.** *LP_4 for Max-min has an integrality gap between 2.5 and 4.*

With some optimization in the Max-min case, we can further reduce the size of LP_4 (see Appendix B).

▶ **Corollary 8.** *For Makespan (Max-min) objective there is a linear programming relaxation with $O(n^3)$ variables and constraints that approximates the problem with a ratio of $11/6$ (respectively, 4).*

Notable is that the lower bounds are higher than those known for the configuration LP. There, the worst instances known give an integrality gap of 2 (Max-min) and 9/6 (Makespan). This means for LP_2 / LP_4 we are closer to a full understanding. It also shows that, assuming that the integrality gap of the configuration LP is indeed the respective lower bound, proving it will require utilizing constraints that are not already implied by LP_2 (Makespan) or LP_4 (Max-min).

▶ **Corollary 9.** *There exists an $11/6$-estimation (4-estimation) algorithm for the Makespan objective (respectively, the Max-min objective).*

The estimation algorithm is based on computing the optimum of the relaxation. This improves on the $11/6 + \epsilon$ $(4 + \epsilon)$ rate previously known. The error of ϵ in the previous result comes from the fact that the configuration LP can only be solved approximately.

2 Compact linear program

In this section, we present a polynomial size linear program, which is feasible if and only if LP_r is feasible. Note that in order to meet the claimed size, we need to eliminate unnecessary variables, which is discussed in 2.1.

For simplicity of notation, define big jobs $\mathcal{J}_B := \{j \in \mathcal{J} : r \cdot p_j > T\}$ and small jobs $\mathcal{J}_S := \mathcal{J} \setminus \mathcal{J}_B$. In the first part, we write an LP for the big jobs and only then deal with the small ones. We write the set of big configurations as $C_B(T) = \{\chi \in \{0,1\}^{\mathcal{J}_B}\}$.

The convexity constraint for LP_r implies that $(x_{i,j})_{j\in\mathcal{J}_B} \in \mathrm{conv}(C_B(T))$ for every $i \in \mathcal{M}$. In other words, there exist $a_{i,\chi} \geq 0$ $(i \in \mathcal{M}, \chi \in C_B(T))$ such that $\sum_{\chi\in C_B(T)} a_{i,\chi} = 1$ $(*)$ for every $i \in \mathcal{M}$ and $x_{i,j} = \sum_{\chi\in C_B(T)} \chi_j a_{i,\chi}$ for every $i \in \mathcal{M}, j \in \mathcal{J}$. With this idea in mind, we construct an LP by using variables $a_{i,\chi}$, the constraint $(*)$, and the allocation LP where we substitute every occurrence of $x_{i,j}$ for $\sum_{\chi\in C_B(T)} \chi_j a_{i,\chi}$.

$$\sum_{\chi\in C_B(T)} a_{i,\chi} = 1 \qquad \forall i \in \mathcal{M} \tag{6}$$

$$\sum_{i\in\Gamma(j)} \sum_{\chi\in C_B(T)} \chi_j a_{i,\chi} \geq (\leq)\ 1 \qquad \forall j \in \mathcal{J}_B \tag{7}$$

$$\sum_{i\notin\Gamma(j)} \sum_{\chi\in C_B(T)} \chi_j a_{i,\chi} = 0 \qquad \forall j \in \mathcal{J}_B \tag{8}$$

$$a_{i,\chi} \geq 0$$

In the following, we will show how to cope with small jobs. For every $j \in \mathcal{J}_S$, $i \in \mathcal{M}$, and $\chi \in C_B(T)$ we use a variable $b_{j,i,\chi}$ that describes how much of j is used on machine i together with χ. Here $b_{j,i,\chi} = a_{i,\chi}$ means it is fully used and $b_{j,i,\chi} = 0$ means it is not used at all.

$$\sum_{i\in\Gamma(j)} \sum_{\chi\in C_B(T)} b_{j,i,\chi} \geq (\leq)\ 1 \qquad \forall j \in \mathcal{J}_S \tag{9}$$

$$\sum_{i\notin\Gamma(j)} \sum_{\chi\in C_B(T)} b_{j,i,\chi} = 0 \qquad \forall j \in \mathcal{J}_S \tag{10}$$

$$\sum_{j\in\mathcal{J}_S} p_j b_{j,i,\chi} \leq (\geq)\ (T - \sum_{j\in\mathcal{J}_B} p_j\chi_j) a_{i,\chi} \qquad \forall i \in \mathcal{M}, \chi \in C_B(T) \tag{11}$$

$$0 \leq b_{j,i,\chi} \leq a_{i,\chi}$$

2.1 Restricting the variables

We denote by $\mathrm{supp}(\chi)$ the non-zero components of $\chi \in C_B(T)$. Observe that in the makespan case, a configuration $\chi \in C_B(T)$ with $|\mathrm{supp}(\chi)| \geq r$ cannot be used, i.e., $a_{i,\chi} = 0$ must hold for a feasible solution. Otherwise, the right hand side of (11) is negative. Hence, we can throw away such variables and the number of remaining configurations is at most $\sum_{k=0}^{r-1} \binom{n}{k} = O(n^{r-1})$. It is easy to see that $O(n^{r+1})$ variables and constraints are left.

For Max-min, we notice that when a configuration $\chi \in C_B(T)$ has $|\mathrm{supp}(\chi)| \geq r$, then (11) is trivially satisfied. If a configuration $\chi \in C_B(T)$ with $|\mathrm{supp}(\chi)| > r$ is used, then we can shift its value to any $\chi' \leq \chi$ (component-wise) where $|\mathrm{supp}(\chi)| = r$. Hence, if there is a feasible solution, then there is also one that uses only configurations $\chi \in C_B(T)$ with $|\mathrm{supp}(\chi)| \leq r$. This gives a total of $O(n^r)$ relevant configurations and thus $O(n^{r+2})$ variables and constraints.

2.2 Equivalence to LP_r

▶ **Lemma 10.** *If the compact linear program is feasible, then* LP_r *is feasible.*

Proof. Consider a feasible solution a, b. Recall, we have that each machine is assigned a combination of configurations in $C_B(T)$. We will extend these configurations by adding small jobs to them. For this purpose, define for every $i \in \mathcal{M}$ and $\chi \in C_B(T)$ a vector $f(i,\chi) \in [0,1]^{\mathcal{J}}$ by

$$f(i,\chi)_j := \begin{cases} \chi_j & \text{if } j \in \mathcal{J}_B, \\ b_{j,i,\chi}/a_{i,\chi} & \text{if } j \in \mathcal{J}_S \text{ and } a_{i,\chi} > 0, \\ 0 & \text{otherwise.} \end{cases}$$

Note that since $b_{j,i,\chi} \leq a_{i,\chi}$, we have $f(i,\chi)_j \in [0,1]$. We define the solution x for LP_r on every machine as a convex combination of these vectors, more formally

$$x_{i,j} := \sum_{\chi \in C_r(T)} a_{i,\chi} \cdot f(i,\chi)_j.$$

It follow directly from the constraints in the compact linear program that x is in the allocation polytope. Let us verify that x satisfies the convexity constraint for i. Since $\sum_{\chi \in C_r(T)} a_{i,\chi} = 1$, it is sufficient to show that for every $i \in \mathcal{M}, \chi \in C_B(T)$ with $a_{i,\chi} > 0$,

$$f(i,\chi) \in \{\chi \in [0,1]^{\mathcal{J}} : p^T\chi \leq (\geq)\ T \text{ and } \chi_j \in \{0,1\} \text{ if } p_j \cdot r > T\}.$$

By definition we have $f(i,\chi)_j \in \{0,1\}$ if $p_j \cdot r > T$ (i.e., $j \in \mathcal{J}_B$) and using constraint (11) we find,

$$p^T f(i,\chi) = \sum_{j \in \mathcal{J}_B} p_j\chi_j + \sum_{j \in \mathcal{J}_S} p_j b_{j,i,\chi}/a_{i,\chi} \leq (\geq) \sum_{j \in \mathcal{J}_B} p_j\chi_j + (T - \sum_{j \in \mathcal{J}_B} p_j\chi_j)a_{i,\chi}/a_{i,\chi} = T. \blacktriangleleft$$

▶ **Lemma 11.** *If* LP_r *is feasible, then the compact linear program is feasible.*

Proof. Let x be a solution for LP_r and define

$$K = \{\chi \in [0,1]^{\mathcal{J}} : p^T\chi \leq (\geq)\ T, \chi_j \in \{0,1\} \text{ if } p_j \cdot r > T\}.$$

Let $i \in \mathcal{M}$. Then by the convexity constraint, there exist $(\lambda_\chi)_{\chi \in K}$ non-negative with $\sum_{\chi \in K} \lambda_\chi = 1$ and $x_{i,j} = \sum_{\chi \in K} \lambda_\chi \chi_j$ for all $j \in \mathcal{J}$.

Let $\psi \in C_B(T)$ and let $K(\psi) \subseteq K$ denote those $\chi \in K$ which have $\chi_j = \psi_j$ for all $j \in \mathcal{J}_B$. We define the variables for i and ψ as $a_{i,\psi} = \sum_{\chi \in K(\psi)} \lambda_\chi$ and $b_{i,j,\psi} = \sum_{\chi \in K(\psi)} \lambda_\chi \chi_j$ for all $j \in \mathcal{J}_S$. Note that

$$\sum_{\psi \in C_B(T)} b_{i,j,\psi} = \sum_{\psi \in C_B(T)} \sum_{\chi \in K(\psi)} \lambda_\chi \chi_j = \sum_{\chi \in K} \lambda_\chi \chi_j = x_{i,j}.$$

With that in mind, the constraints except for (11) are straight-forward. Moreover, we show said constraint as follows.

$$\begin{aligned} \sum_{j \in \mathcal{J}_B} p_j\psi_j + \frac{1}{a_{i,\psi}} \sum_{j \in \mathcal{J}_S} p_j b_{j,i,\psi} &= \sum_{j \in \mathcal{J}_B} p_j\psi_j + \frac{1}{a_{i,\psi}} \sum_{j \in \mathcal{J}_S} [p_j \sum_{\chi \in K(\psi)} \lambda_\chi \chi_j] \\ &= \underbrace{\sum_{\chi \in K(\psi)} [\frac{\lambda_\chi}{a_{i,\psi}}}_{=1} \sum_{j \in \mathcal{J}_B} p_j \underbrace{\chi_j}_{=\psi_j}] + \sum_{\chi \in K(\psi)} [\frac{\lambda_\chi}{a_{i,\psi}} \sum_{j \in \mathcal{J}_S} p_j\chi_j] \\ &= \sum_{\chi \in K(\psi)} \frac{\lambda_\chi}{a_{i,\psi}} p^T\chi \leq (\geq)\ T. \end{aligned}$$

◀

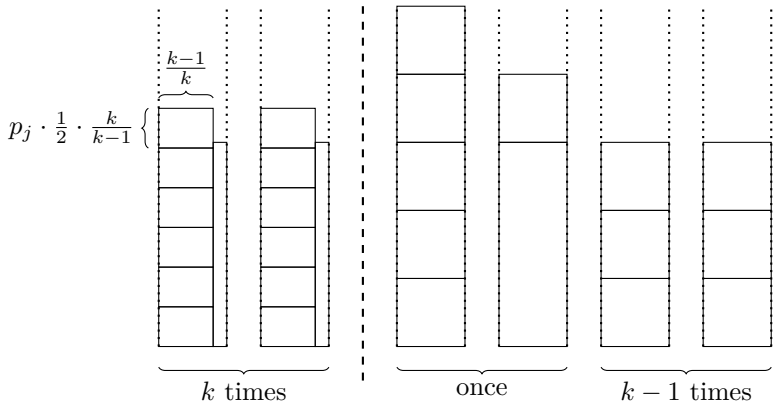

Figure 1 Fractional and integral solution for lower bound (Makespan)

3 Lower bound (Makespan)

Here, we give a lower bound of 5/3 for LP_3 (in particular, for the weaker LP_2) in the Makespan case. A similar construction for the Max-min case is given in the appendix.

Let k be an even number and consider an instance with k machines, i.e., $k/2$ pairs of machines, and $3k+1$ jobs. For each of the $k/2$ pairs (i_1, i_2) let there be 6 jobs j with $p_j = 1/3$ and $\Gamma(j) = \{i_1, i_2\}$. Furthermore let there be one job j_B with $p_{j_B} = 1$ and $\Gamma(j_B) = \mathcal{M}$, i.e., it can be assigned anywhere.

Assume toward contradiction that there is a schedule with makespan strictly less than 5/3. j_B has to be assigned somewhere and we denote this machine by i. There can be at most one job of size 1/3 that is assigned to i as well. Hence, 5 jobs of size 1/3 must be assigned to the other machine in this pair; thus the load on that machine is at least 5/3. A contradiction.

Next, we show that LP_2 is feasible for $T = k/(k-1)$. For every $i \in \mathcal{M}$ let $x_{i,j_B} = 1/k$, i.e., the big job is distributed evenly across all machines. For every other job j, split it across the two machines it is allowed on. More formally, let $x_{i_1,j} = x_{i_2,j} = 1/2$ with $\{i_1, i_2\} = \Gamma(j)$. Clearly x is in the allocation polytope. Let $i \in \mathcal{M}$. We need to verify that

$$(x_{i,j})_{j\in\mathcal{J}} \in \mathrm{conv}\{\chi \in [0,1]^{\mathcal{J}} : p^T\chi \le k/(k-1) \text{ and } \chi_j \in \{0,1\} \text{ if } p_j \cdot 3 > k/(k-1)\}.$$

We define one vector for the big job and one for each machine:

$$\chi_j^{(j_B)} := \begin{cases} 1 & \text{if } j = j_B, \\ 0 & \text{otherwise,} \end{cases} \quad \text{and } \chi_j^{(i)} := \begin{cases} \frac{1}{2} \cdot \frac{k}{k-1} & \text{if } p_j = 1/3 \text{ and } i \in \Gamma(j), \\ 0 & \text{otherwise.} \end{cases}$$

Note that we have $p^T\chi^{(j_B)} = p_{j_B} = 1 \le \frac{k}{k-1} = T$ as well as $p^T\chi^{(i)} = 6 \cdot \frac{1}{3} \cdot \frac{1}{2} \cdot \frac{k}{k-1} = \frac{k}{k-1} = T$. The integrality constraint is satisfied for j_B and since $1/3 \cdot 3 = 1 \le k/(k-1)$, it is not necessary for the small jobs. Also, it holds that $(x_{i,j})_{j\in\mathcal{J}} = 1/k \cdot \chi^{(j_B)} + (k-1)/k \cdot \chi^{(i)}$, as shown below.

$$x_{i,j} = \begin{cases} 1/k \cdot 1 + (1-1/k) \cdot 0 = 1/k \cdot \chi_j^{(j_B)} + (k-1)/k \cdot \chi_j^{(i)} & \text{if } j = j_B, \\ 1/k \cdot 0 + \frac{k-1}{k} \cdot \frac{1}{2} \cdot \frac{k}{k-1} = 1/k \cdot \chi_j^{(j_B)} + (k-1)/k \cdot \chi_j^{(i)} & \text{if } p_j = 1/3 \text{ and } i \in \Gamma(j), \\ 1/k \cdot 0 + (1-1/k) \cdot 0 = 1/k \cdot \chi_j^{(j_B)} + (k-1)/k \cdot \chi_j^{(i)} & \text{otherwise.} \end{cases}$$

For this instance, we get a gap that approaches 5/3 as k tends to infinity. This construction does not work for LP_4 and higher, since in those cases integrality constraints are enforced

for the jobs of size 1/3 as well. However, similar constructions work when using 10 jobs of size 1/5, 14 jobs of size 1/7, etc. The lower bound becomes weaker the smaller the size is chosen, but it always stays strictly above 3/2, the best lower bound for the configuration LP.

4 Proof of integrality gap

The proof for the integrality gap is by using a potentially exponential time approximation algorithm. The algorithm takes as an input T and returns a solution of value αT, where α is the bound on the integrality gap, or proves that LP_2 (Makespan) or LP_4 (Max-min) is infeasible w.r.t. T.

In 4.1, we prove a criterion for the infeasibility of LP_r. In previous literature, for the configuration LP a similar criterion was derived from its dual. In fact, if r tends to infinity (i.e., $p_j \cdot r > T$ for all $j \in \mathcal{J}$), our criterion is equivalent to that one.

In the proofs for the configuration LP, our criterion can replace the previous one in a straight-forward way and give the integrality gap for LP_2 (Makespan) or LP_4 (Max-min).

4.1 Criterion for infeasibility

The criterion is derived from the LP_r in another equivalent representation and by using the duality theorem (e.g., unbounded dual implies infeasible primal). For this purpose, we construct a representation of LP_r, where we do not care about its size, but the goal is to obtain a rather simple dual.

▶ **Lemma 12.** *LP_r is infeasible w.r.t. T if there are $y \in \mathbb{R}^{\mathcal{M}}_{\geq 0}$ and $z \in \mathbb{R}^{\mathcal{J}}_{\geq 0}$ with $\sum_{j\in\mathcal{J}} z_j > (<) \sum_{i\in\mathcal{M}} y_i$ such that for every $i \in \mathcal{M}$, $\chi \in [0,1]^{\mathcal{J}}$ with*
1. *$p^T\chi \leq (\geq)\ T$,*
2. *$\chi_j \in \{0,1\}$ for every $j \in \mathcal{J}$ with $p_j \cdot r > T$, and*
3. *$\chi_j = 0$ for every $j \in \mathcal{J}$ with $i \notin \Gamma(j)$,*

it holds that $z^T\chi \leq (\geq)\ y_i$.

For this lemma even equivalence holds. To conserve space we will only show this direction.

Proof of Lemma 12. Recall for the compact linear program, the big jobs were assigned to machines in configurations. We want to include the (fractional) allocation of small jobs in those configurations as well. The natural approach is to define

$$C(T) := \{\chi \in [0,1]^{\mathcal{J}} : p^T\chi \leq (\geq)\ T, \chi_j \in \{0,1\} \text{ if } p_j \cdot r > T\}.$$

Then a representation of LP_r is the following.

$$\min(\max)\ 0 \tag{12}$$

$$\sum_{\chi\in C(T)} a_{\chi,i} = 1 \qquad \forall i \in \mathcal{M} \tag{13}$$

$$\sum_{i\in\Gamma(j)} \sum_{\chi\in C(T)} \chi_j \cdot a_{\chi,i} \geq (\leq)\ 1 \qquad \forall j \in \mathcal{J} \tag{14}$$

$$\sum_{i\notin\Gamma(j)} \sum_{\chi\in C(T)} \chi_j \cdot a_{\chi,i} = 0 \qquad \forall j \in \mathcal{J} \tag{15}$$

$$a_{\chi,i} \geq 0$$

There is, however, an issue with this definition. Since $C(T)$ can have infinitely many elements, the dimension of the LP is potentially infinite. This means, we cannot simply apply results from LP duality. Thus, we will first show that a finite number of variables is suffices.

Recall that the constraint for LP_r is $(x_{i,j})_{j\in\mathcal{J}} \in \text{conv}(C(T))$. We will show that $\text{conv}(C(T)) = \text{conv}(V(T))$ for some finite $V(T) \subseteq C(T)$. This means, we can substitute $C(T)$ for $V(T)$.

Observe that $C(T)$ is a union of polytopes, where each polytope corresponds to one integral allocation of the big jobs. More formally, let $\chi^B \in \{0,1\}^{\mathcal{J}_B}$. Then the set of vectors in $C(T)$ where the values for big jobs equal χ^B are exactly

$$\{\chi \in [0,1]^{\mathcal{J}} : p^T\chi \leq (\geq)\ T \text{ and } (\chi_j)_{j\in\mathcal{J}_B} = \chi^B\}.$$

For each of these χ^B, this is clearly a polytope, which can be written as the convex hull of finitely many basic solutions. Let $V(T)$ be the set of all basic solutions for all allocations χ_B of big jobs. Then $(x_{i,j})_{j\in\mathcal{J}} \in \text{conv}(C(T))$ is equivalent to $(x_{i,j})_{j\in\mathcal{J}} \in \text{conv}(V(T))$

We substitute $C(T)$ for $V(T)$ in the LP above and, for an easier dual, we multiply (13) by -1. Then the dual is the following:

$$\max(\min) \sum_{j\in\mathcal{J}} z_j - \sum_{i\in\mathcal{M}} y_i$$

$$\sum_{j\in\mathcal{J}:i\in\Gamma(j)} \chi_j \cdot z_j + \sum_{j\in\mathcal{J}:i\notin\Gamma(j)} \chi_j \cdot \overline{z}_j \leq (\geq)\ y_i \quad \forall i \in \mathcal{M}, \chi \in V(T)$$

$$z_j \geq 0$$

$$\overline{z}_j, y_i \in \mathbb{R}$$

Now consider the values z, y from Lemma 12 and set $\overline{z}_j$ to a negative (Makespan) or positive (Max-min) number of very large magnitude for all $j \in \mathcal{J}$. Then this is a feasible solution for the dual: Let $i \in \mathcal{M}$ and $\chi \in V(T)$. If $\chi_j = 0$ for all $j \in \mathcal{J}$ with $i \notin \Gamma(j)$, then the constraint is satisfied by definition of z and y. Otherwise, the constraint holds when $\overline{z}_j$ is chosen sufficiently small (large).

The solution has a positive (negative) objective value and we can scale it by any constant and construct a new feasible solution. This way we can obtain any positive (negative) objective value, i.e., the dual is unbounded. By duality this implies the primal is infeasible. ◀

4.2 Local search algorithm

▶ **Definition 13** (Good and bad machines)**.** Given an allocation $\sigma : \mathcal{J} \to \mathcal{M}$, we call a machine i bad, if $\sum_{j\in\sigma^{-1}(i)} p_j > 11/6 \cdot T$ (Makespan) or $\sum_{j\in\sigma^{-1}(i)} p_j < 1/4 \cdot T$ (Max-min). A machine is good, if it is not bad.

The local search algorithm starts with an arbitrary allocation and moves jobs until all machines are good, or it can prove that LP_2 (Makespan) or LP_4 (Max-min) is infeasible w.r.t. T. During this process, a machine that is already good will never be made bad.

The central data structure of the algorithm is an ordered list of moves $L = (L_1, L_2, \ldots, L_\ell)$. Here, every component $L_k = (j, i)$, $j \in \mathcal{J}$ and $i \in \Gamma(j)$, stands for a move the algorithm wants to perform. It will not perform the move, if this would create a bad machine, i.e., $p(\sigma^{-1}(i)) + p_j > 11/6 \cdot T$ (Makespan) or $p(\sigma^{-1}(\sigma(j))) - p_j < 1/4 \cdot T$ (Max-min). If it does not create a bad machine, we say that the move (j, i) is valid. For every $0 \leq k \leq \ell$ define $L_{\leq k} := (L_1, \ldots, L_k)$, the first k elements of L (with $L_{\leq 0}$ being the empty list).

Algorithm 1 Local search meta-algorithm

1. Let σ be an arbitrary allocation;
2. $\ell \leftarrow 0$; // length of the list of moves L
3. while there is a bad machine do
   a. if there exists a valid move $(j, i) \in L$ then
      i. Let $0 \leq k \leq \ell$ be minimal such that
         (Makespan) j is repelled by $\sigma(j)$ w.r.t. $L_{\leq k}$;
         (Max-min) j is attracted by i w.r.t. $L_{\leq k}$;
      ii. $\sigma(j) \leftarrow i$;
      iii. $L \leftarrow L_{\leq k}$; $\ell \leftarrow k$; // Forget moves $L_{k+1}, \ldots, L_\ell$
   b. else
      i. Choose a move $(j, i) \notin L$, $j \in \mathcal{J}$ and $i \in \Gamma(j)$, with p_j minimal and
         (Makespan) j is repelled by $\sigma(j)$ and not repelled by i w.r.t. L;
         (Max-min) j is not attracted by $\sigma(j)$ and attracted by i w.r.t. L;
      ii. $L_{\ell+1} \leftarrow (j, i)$; $\ell = \ell + 1$; // Append (j, i) to L

Depending on the current schedule σ and list of moves L, we define for every machine i which jobs are repelled or not repelled (Makespan). In the Max-min case we use the term attracted instead of not repelled; not attracted instead of repelled. This is only to make the definitions easier to read. The definition of repelled/attracted jobs differs in the two algorithms and is given in Section 4.2.1. The algorithm will only add a new move (j, i) to the current list L, if j is repelled (not attracted) by its current machine and not repelled (attracted) by the target i w.r.t. L.

4.2.1 Repelled and attracted jobs

Here, we will start with the Max-min case, since it is the simpler one.

Max-min

Depending on the current schedule σ and the current set of moves $L = L_{\leq \ell}$, we define which jobs are attracted by which machines. We call a job $j \in \mathcal{J}$ big, if $p_j > 1/4 \cdot T$ and small otherwise. The first two rules are that bad machines attract all jobs and that rules are propagated from prefixes of the list.
(initialization) If $\ell = 0$, every bad machine i attracts every job j.
(monotonicity) If $\ell > 0$ and i attracts j w.r.t. $L_{\leq \ell-1}$, then i attracts j w.r.t. $L_{\leq \ell}$.
For the remaining rules, assume $\ell > 0$, and let $(j_\ell, i_\ell) := L_\ell$ be the last move added. We will define which new rules this move adds to the existing ones.

We can assume that all moves in $L_{\leq \ell-1}$ are not valid, since otherwise the algorithm would execute them instead of adding a new one. Since i_ℓ tries to steal j_ℓ from $\sigma(j_\ell)$, but the move is not valid, the machine $\sigma(j_\ell)$ should attract jobs in order to make (j_ℓ, i_ℓ) valid. More precisely,
(small-all) if j_ℓ is small, $\sigma(j_\ell)$ attracts all jobs and
(big-big) if j_ℓ is big, $\sigma(j_\ell)$ attracts all big jobs.

This misses one important case. Suppose that j_ℓ is big. Intuitively, $\sigma(j_\ell)$, should also collect small jobs to make the move (j_ℓ, i_ℓ) valid. However, the straight-forward way ($\sigma(j)$ attracts all small jobs as well) does not work out in the analysis. Hence, a more sophisticated strategy is required.

For $i \in \mathcal{M}$ define $S_i(L)$ to be all small jobs j which have either $\sigma(j) = i$ or which have $i \in \Gamma(j)$ and are not attracted by their current machine, $\sigma(j)$, w.r.t. the rules above. The intuition behind $S_i(L)$ is the following. In the best case, i could collect all jobs from $S_i(L)$. If $p(S_i(L)) < 1/4 \cdot T$, then we cannot expect i to satisfy its demand only using small jobs. On the other hand, if it gets a big job, then this job alone satisfies its demand. Hence, i should not attract small jobs. More formally,

(big-all) if j_ℓ is big and $p(S_{\sigma(j_\ell)}(L)) \geq 1/4 \cdot T$, then $\sigma(j_\ell)$ attracts all jobs.

Makespan

Here, we define big jobs to be those $j \in \mathcal{J}$ that have $p_j > 1/2 \cdot T$ and small jobs all others.

(initialization) If $\ell = 0$, every bad machine i repels every job j.

(monotonicity) If $\ell > 0$ and i repels j w.r.t. $L_{\leq \ell-1}$, then i repels j w.r.t. $L_{\leq \ell}$.

Again, the remaining rules regard $\ell > 0$ and we define $(j_\ell, i_\ell) := L_\ell$, i.e., the last move added. In order to make space for j_ℓ, the machine i_ℓ should repel jobs.

(small-all) If j_ℓ is small, i_ℓ repels all jobs.

This still leaves one case to resolve. If j_ℓ is big, which jobs does i_ℓ repel? It helps to imagine that the algorithm is a lazy one: It repels jobs only if it is really necessary.

For $i \in \mathcal{M}$ define $S_i(L)$ to be those small jobs j which have $\sigma(j) = i$ and which are repelled by all other potential machines, i.e., $\Gamma(j) \setminus \{i\}$, w.r.t. the rules above. The intuition behind $S_i(L)$ is that we do not expect that i can get rid of any of the jobs in $S_i(L)$.

Next, define a threshold $t(L, (j_\ell, i_\ell))$ as the minimum $t \geq 0$ such that S_i and all big jobs below this threshold are already too large to add j_ℓ:

$$p(\{j \in \sigma^{-1}(i_\ell) : j \in S_{i_\ell}(L) \text{ or } 1/2 < p_j \leq t\}) + p_{j_\ell} > 11/6 \cdot T,$$

or $t(L, (j_\ell, i_\ell)) = \infty$ if no such t exists. In order to make (j_ℓ, i_ℓ) valid, it is necessary (although not always sufficient) to remove one of the big jobs with size at most $t(L, (j_\ell, i_\ell))$. Hence, we define,

(big-all) if j_ℓ is big and $t(L, (j_\ell, i_\ell)) = \infty$, then i_ℓ repels all jobs and

(big-big) if j_ℓ is big and $t(L, (j_\ell, i_\ell)) < \infty$, then i_ℓ repels $S_{i_\ell}(L)$ and all jobs j with $1/2 < p_j \leq t(L, (j_\ell, i_\ell))$.

Note that repelling $S_{i_\ell}(L)$ seems unnecessary, since those jobs do not have any machine to go to. However, this definition simplifies the analysis. It is also notable that the special case where $t(L, (j_\ell, i_\ell)) = 0$ is equivalent to $p(S_{i_\ell}(L)) + p_{j_\ell} > 11/6 \cdot T$ and here the algorithm gives up making (j_ℓ, i_ℓ) valid. Finally, we want to highlight the following counter-intuitive (but intentional) aspect of the algorithm. It might happen that some job of size greater than $t(L, (j_\ell, i_\ell))$ is moved to i_ℓ, only to be removed again later on, when $t(L, (j_\ell, i_\ell))$ has increased.

4.3 Analysis (Max-min)

For completeness, we give the analysis for the Makespan case in the appendix. It is not included here so as to avoid repetitive arguments.

To verify the correctness of the algorithm, it has to be shown that (1) it terminates and (2) in each iteration of the main loop, there is either a valid move in L or some move that can be added to L.

▶ **Theorem 14.** *The algorithm terminates after finitely many iterations of the main loop.*

Proof. As before, let ℓ denote the current length of L. We define a potential function

$$\Phi(L) = (b, s_0, s_1, s_2, \ldots, s_\ell, \infty),$$

where b is the number of bad machines and s_k is the number of pairs $(i, j) \in \mathcal{M} \times \mathcal{J}$ where i attracts j w.r.t. $L_{\leq k}$ and $\sigma(j) \neq i$. Intuitively, the algorithm makes progress when this number decreases.

Note that the length of $\Phi(L)$ is bounded by $|\mathcal{M}| \cdot |\mathcal{J}| + 2$ and every component can only have $|\mathcal{M}| \cdot |\mathcal{J}| + 1$ many different values. Thus, the range of the potential function is finite.

We will show that the vector decreases lexicographically after every iteration of the main loop; hence the running time is bounded by the number of such vectors times the maximum number of operations in one iteration and, in particular, is finite. For the lexicographic decrease consider two cases. Either a new move is added, which decreases the potential function by replacing the last component with some finite value, or a move is performed. If a move turns a bad machine good, b decreases and so does the lexicographic value of $\Phi(L)$. Otherwise, let $\ell' \leq \ell$ be the length of the list after a move (j, i) is performed. Recall the algorithm prevents jobs attracted by their current machine from being moved, i.e., j is not attracted by its previous machine w.r.t. $L_{\leq \ell'}$. Moreover, observe that the attracted jobs w.r.t. $L_{\leq 0}, \ldots, L_{\leq \ell'}$ do not change. This can be seen from the definition of attracted jobs. Therefore $s_0, \ldots, s_{\ell'}$ do not increase. Finally, since j is attracted by i w.r.t. $L_{\leq \ell'}$ and after the move $\sigma(j) = i$ holds, the value of $s_{\ell'}$ has decreased. ◀

▶ **Theorem 15.** *If* LP_4 *is feasible w.r.t.* T*, the algorithm always has an operation it can perform.*

Proof. As in the algorithm, call a job j small, if $p_j < 1/4 \cdot T = 1/r \cdot T$ and big otherwise. Suppose toward contradiction, there are bad machines, no move in L is valid, and no move can be added to L. We will construct values $(z_j)_{j \in \mathcal{J}}$, $(y_i)_{i \in \mathcal{M}}$ with the properties as in Lemma 12 and thereby show that LP_4 is infeasible w.r.t. T.

Define $y_i = 3/4$ for every $i \in \mathcal{M}$ where i is bad or $(j', i') \in L$ for some $j' \in \mathcal{J}$ with $\sigma(j') = i$. For all other machines i define $y_i = 0$. Furthermore, define $z_j = 3/4$ if j is big and attracted by $\sigma(j)$; define $z_j = p_j/T$ if j is small and attracted by $\sigma(j)$; and $z_j = 0$ if j is not attracted by $\sigma(j)$. We proceed to show that these values indeed satisfy the properties as in Lemma 12.

▶ **Fact 16.** *Let* j *be a job attracted by some machine* $i \in \Gamma(j)$ *(not necessarily* $\sigma(j)$*). Then* $z_j = 3/4$ *if* j *is big and* $z_j = p_j/T$*, otherwise.*

We need to show that j is attracted by $\sigma(j)$. If $\sigma(j) = i$, then this holds trivially. If $\sigma(j) \neq i$, then either j is attracted by $\sigma(j)$ or $(j, i) \in L$, since no moves can be added. In the latter case $\sigma(j)$ attracts at least all big jobs if j is big and all jobs if j is small. In either case $\sigma(j)$ attracts j and therefore Fact 16 holds.

Let $i \in \mathcal{M}$ and $\chi \in [0, 1]^{\mathcal{J}}$ with $p^T \chi \geq T$, $\chi_j \in \{0, 1\}$ for every big job, and $\chi_j = 0$ if $i \notin \Gamma(j)$. We must show that $z^T \chi \geq y_i$.

If $y_i = 0$ then $z^T \chi \geq y_i$, since $z^T \chi$ is non-negative. Hence, assume w.l.o.g. that $y_i = 3/4$. By definition of attracted jobs, this means i attracts at least all big jobs. Moreover, the inequality holds trivially if $\chi_j = 1$ for some big job j, since $z_j = 3/4$ (Fact 16). Because for big jobs j we have $\chi_j \in \{0, 1\}$ and the case $\chi_j = 1$ is trivial, the only interesting case is where $\chi_j = 0$ for all big jobs j. We note that this is the only argument in which we use the integrality of some component in χ.

Define $S_i(L_{\leq k})$ for all $0 \leq k \leq \ell$ as in the algorithm. Because $y_i = 3/4$, we know that there is a $(j_k, i_k) = L_k$ $(k \leq \ell)$ such that $\sigma(j_k) = i$. Then there are two cases.

1. **Case: i attracts all jobs.** Since $\chi_j = 0$ is assumed for all big jobs j and i attracts all jobs, by Fact 16 we get $z^T\chi = p^T\chi/T \geq 1 > y_i$.
2. **Case: i attracts only big jobs.** Then j_k must be big and $p(S_i(L_{\leq k})) < 1/4 \cdot T$ (rule big-big). Note that $z_j = p_j/T$ for every small job $j \notin S_i(L_{\leq \ell}))$ with $i \in \Gamma(j)$, since by definition j is attracted by $\sigma(j)$ w.r.t. $L_{\leq k}$ (in particular, w.r.t. $L_{\leq \ell}$). Hence,

$$z^T\chi \geq \sum_{j \in \mathcal{J} \setminus S_i(L)} z_j\chi_j = \sum_{j \in \mathcal{J} \setminus S_i(L)} p_j\chi_j/T \geq (p^T\chi - p(S_i(L)))/T > (T - 1/4 \cdot T)/T = y_i.$$

It remains to show that $\sum_{j \in \mathcal{J}} z_j < \sum_{i \in \mathcal{M}} y_i$. We show that, with amortization, good machines satisfy $z(\sigma^{-1}(i)) \leq y_i$ and for bad machines strict inequality holds.

Let i be a bad machine. A bad machine cannot have a big job assigned to it. Moreover, all jobs (in particular those in $\sigma^{-1}(i)$) are attracted by i. Hence,

$$z(\sigma^{-1}(i)) = p(\sigma^{-1}(i))/T < 1/4 < y_i.$$

Let i be a good machine. If $y_i = 0$, then i attracts no jobs and therefore $z(\sigma^{-1}(i)) = 0$. For the remaining part, assume that $y_i = 3/4$, that is to say, there exists a move $(j_k, i_k) = L_k$ $(k \leq \ell)$ such that $\sigma(j_k) = i$.

Case (big-big): i attracts only big jobs. Then j_k must be big and since (j_k, i_k) is not valid, it must be the only big job on i. Thus,

$$z(\sigma^{-1}(i)) = z_{j_B} = 3/4 = y_i.$$

Case (big-all): i attracts all jobs and j_k is big. Since (j_k, i_k) is not valid, we have $p(\sigma^{-1}(i) \setminus \{j_k\}) < 1/4 \cdot T$. In particular, $\sigma^{-1}(i) \setminus \{j_k\}$ does not contain another big job. Thus,

$$z(\sigma^{-1}(i)) = z_{j_k} + z(\sigma^{-1}(i) \setminus \{j_k\}) = z_{j_B} + p(\sigma^{-1}(i) \setminus \{j_k\})/T < 3/4 + 1/4 = y_i + 1/4.$$

Case (small-all): i attracts all jobs and j_k is small. Again, (j_k, i_k) is not valid. In particular, $\sigma^{-1}(i)$ cannot contain a big job. Hence,

$$z(\sigma^{-1}(i)) = p(\sigma^{-1}(i))/T = (p(\sigma^{-1}(i)) - p_{j_k})/T + p_{j_k}/T < 1/4 + 1/4 = y_i - 1/4.$$

It is easy to see that cases (big-all) and (small-all) are disjoint: Once a machine attracts all jobs, no new move will be added for a job assigned to it.

▶ **Fact 17.** *There are at least as many machines of case (small-all) as there are of case (big-all).*

The proof of Fact 17 is postponed to the end. With Fact 17, we can amortize those two cases and get

$$\sum_{j \in \mathcal{J}} z_j = \sum_{i \in \mathcal{M}} z(\sigma^{-1}(i)) < \sum_{i \in \mathcal{M}} y_i.$$ ◀

Proof of Fact 17. Let $L_{b_1}, L_{b_2}, \ldots, L_{b_h}$ be the moves that correspond to case (big-all) machines, i.e., for each $(j_k, i_k) = L_{b_k}$ $(k \leq h)$, $\sigma(j_k)$ attracts all jobs and j_k is big.

We argue that there are $L_{s_1}, L_{s_2}, \ldots, L_{s_h}$ such that $b_1 < s_1 < b_2 < s_2 < \ldots b_h < s_h$, where for each $(j_k, i_k) = L_{s_k}$ $(k \leq h)$, j_k is small and therefore $\sigma(j_k)$ is a case (small-all) machine.

Let $k \leq h$. A critical argument is that the algorithm prefers moves of small jobs over those of big jobs. In particular, when $L_{b_{k+1}}$ is added, there does not exist a small job move that it can add instead. However, we have that $p(S_{\sigma(j_k)}(L_{\leq b_k})) \geq 1/4 \cdot T$ and since (j_k, i_k) is and was not valid, the load of small jobs on $\sigma(j_k)$ is strictly less than $1/4 \cdot T$. Hence, there exists a small job j in $S_{\sigma(j_k)}(L_{\leq b_k})$ which is not assigned to $\sigma(j_k)$. If no small move was added after L_{b_k} and before $L_{b_{k+1}}$, then $(j, \sigma(j_k))$ would have been preferred over $L_{b_{k+1}}$. ◀

References

1 Chidambaram Annamalai. Lazy local search meets machine scheduling. *CoRR*, abs/1611.07371, 2016. arXiv:1611.07371.

2 Chidambaram Annamalai, Christos Kalaitzis, and Ola Svensson. Combinatorial algorithm for restricted max-min fair allocation. In Piotr Indyk, editor, *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. SIAM, 2015. doi:10.1137/1.9781611973730.90.

3 Arash Asadpour, Uriel Feige, and Amin Saberi. Santa claus meets hypergraph matchings. *ACM Trans. Algorithms*, 8(3):24:1–24:9, 2012. doi:10.1145/2229163.2229168.

4 Nikhil Bansal and Maxim Sviridenko. The santa claus problem. In Jon M. Kleinberg, editor, *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 31–40. ACM, 2006. doi:10.1145/1132516.1132522.

5 Ivona Bezáková and Varsha Dani. Allocating indivisible goods. *SIGecom Exchanges*, 5(3):11–18, 2005. doi:10.1145/1120680.1120683.

6 Tomás Ebenlendr, Marek Krcál, and Jirí Sgall. Graph balancing: A special case of scheduling unrelated parallel machines. *Algorithmica*, 68(1):62–80, 2014. doi:10.1007/s00453-012-9668-9.

7 Klaus Jansen and Lars Rohwedder. On the configuration-lp of the restricted assignment problem. In Philip N. Klein, editor, *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. SIAM, 2017. doi:10.1137/1.9781611974782.176.

8 Klaus Jansen and Lars Rohwedder. A quasi-polynomial approximation for the restricted assignment problem. In Friedrich Eisenbrand and Jochen Könemann, editors, *Integer Programming and Combinatorial Optimization - 19th International Conference, IPCO 2017, Waterloo, ON, Canada, June 26-28, 2017, Proceedings*, volume 10328 of *Lecture Notes in Computer Science*, pages 305–316. Springer, 2017. doi:10.1007/978-3-319-59250-3_25.

9 Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Program.*, 46:259–271, 1990. doi:10.1007/BF01585745.

10 Lukás Polácek and Ola Svensson. Quasi-polynomial local search for restricted max-min fair allocation. *ACM Trans. Algorithms*, 12(2):13:1–13:13, 2016. doi:10.1145/2818695.

11 Ola Svensson. Santa claus schedules jobs on unrelated machines. *SIAM J. Comput.*, 41(5):1318–1341, 2012. doi:10.1137/110851201.

12 José Verschae and Andreas Wiese. On the configuration-lp for scheduling on unrelated machines. *J. Scheduling*, 17(4):371–383, 2014. doi:10.1007/s10951-013-0359-4.

A NP-Hardness of the configuration-LP

The following reduction uses a typical idea for these kind of problems and we doubt that it is novel. We include it here only for the sake of completeness.

▶ **Theorem 18.** *Solving the configuration LP is NP-hard.*

Proof. We give the proof w.r.t. the Makespan objective. For Max-min the proof is analogous. Consider the NP-hard Partition problem: Given a set of natural numbers $a_1, \dots, a_n$, decide if there exists $A \subseteq \{a_1, \dots, a_n\}$ with $\sum_{j \in A} a_j = 1/2 \cdot \sum_{j=1}^{n} a_j$.

We construct the following instance for Makespan minimization and show that solving the configuration LP it is equivalent to the problem above.

Let there be two machines i_1, i_2 and for each a_j a job j with $\Gamma(j) = \{i_1, i_2\}$ and $p_j = a_j$. There exists such a subset if and only if the optimum of the configuration LP is $1/2 \cdot \sum_{j=1}^{n} a_j$. Recall that the configuration LP for makespan T assigns to each machine a convex combination of configurations, i.e., sets of jobs that do not exceed T in size. If there exists a solution for the Partition problem, then the optimum of the configuration LP is $1/2 \cdot \sum_{j=1}^{n} a_j$. We will assign one configuration with the solution of the Partition problem to one machine and one configuration with all remaining jobs to the other. If the optimum of the configuration LP is $1/2 \cdot \sum_{j=1}^{n} a_j$, then take the biggest configuration that is used. It must have a size of $1/2 \cdot \sum_{j=1}^{n} a_j$, or else the optimum would be lower. This configuration is a solution for the Partition problem. ◀

B Reducing the size of LP_4 (Max-min)

We have shown that LP_4 is strong for the Max-min case. In the Makespan case, already LP_2 gives the best bounds, which is much smaller than LP_4. In order to achieve a similar size, we will show that after a simple preprocessing step of the Max-min instance, the linear program can be reduced to $O(n^3)$ variables and constraints.

Let I be an instance of Max-min with job sizes p_j. Construct a new instance I^T by changing the size of each job $j \in \mathcal{J}$ to

$$p_j^T := \begin{cases} T & \text{if } p_j > T/4, \\ p_j & \text{otherwise.} \end{cases}$$

Recall that in the LP, the configurations for big jobs $\mathcal{J}_B$ (that have size $> T/4$) are defined as $C_B(T) := \{\chi \in \{0,1\}^{\mathcal{J}_B}\}$.

The argument for restricting the support of these configurations was that we do not need a configuration χ if there is a $\chi' \leq \chi$ (component-wise) with $|\mathrm{supp}(\chi')| < |\mathrm{supp}(\chi)|$ and $p^T \chi' \geq T$. It is easy to see that after rounding the sizes, the relevant configuration have at most one non-zero component. Hence, there are only $O(n)$ many, which gives a total size of $O(n^3)$ for the compact LP. Now we need to show that after rounding the sizes, we still get a ratio of 4. In other words,

$$\frac{1}{4}\mathrm{OPT}(\mathrm{LP}_4(I^T)) \leq \mathrm{OPT}(I) \leq \mathrm{OPT}(\mathrm{LP}_4(I^T)).$$

Since the sizes have only increased, the relaxation can only have gotten weaker, i.e., $\mathrm{OPT}(\mathrm{LP}_4(I^T)) \geq \mathrm{OPT}(\mathrm{LP}_4(I)) \geq \mathrm{OPT}(I)$.

Let $T > 4 \cdot \mathrm{OPT}(I)$. We need to show that $\mathrm{LP}_4(I^T)$ is infeasible w.r.t. T. Notice how $\mathrm{OPT}(I^T) = \mathrm{OPT}(I)$: Given the optimal allocation for I^T, the machine that has the minimum load cannot have any jobs j with $p_j \geq T/4 > \mathrm{OPT}(I)$. Otherwise this allocation would yield a higher value for I than $\mathrm{OPT}(I)$. Hence, on the machine with minimum load the jobs have the same size in I and I^T. This means the solution has the same value for I. Since LP_4 has an integrality gap of at most 4, we get

$$\frac{1}{4}\mathrm{OPT}(\mathrm{LP}_4(I^T)) \leq \mathrm{OPT}(I^T) = \mathrm{OPT}(I).$$

In other words, the highest value for which $\mathrm{LP}_4(I^T)$ is feasible is $4 \cdot \mathrm{OPT}(I^T) < T$.

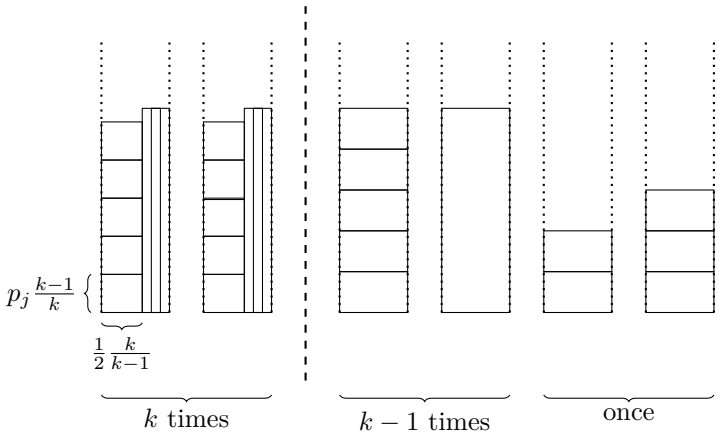

Figure 2 Fractional and integral solution for lower bound (Max-min)

C Lower bound (Max-min)

Here, we give a lower bound of 2.5 for LP_4 (in particular, LP_3, LP_2) for Max-min.

Let k be an even number and consider an instance with k machines, i.e., $k/2$ pairs of machines, and $6k-1$ jobs. For each of the $k/2$ pairs (i_1, i_2) let there be 5 jobs j with $p_j = 1/5$ and $\Gamma(j) = \{i_1, i_2\}$. Furthermore let there be $k/2 - 1$ jobs j with $p_j = 1$ and $\Gamma(j) = \mathcal{M}$, i.e., they can be assigned anywhere.

Assume toward contradiction that there is a schedule with makespan strictly more than $2/5$. There must be at least one pair of machines (i_1, i_2) that do not have a job of size 1 assigned to them. Since there are only 5 jobs of size $1/5$ that are allowed on i_1 and i_2, one of the two machines has at most two. A contradiction.

Next, we show that LP_4 is feasible for $T = (k-1)/k$. For every $i \in \mathcal{M}$ and every job j_B of size 1, let $x_{i,j_B} = 1/k$, i.e., the big job is distributed evenly across all machines. For every other job j, split it across the two machines it is allowed on. More formally, let $x_{i_1,j} = x_{i_2,j} = 1/2$ with $\{i_1, i_2\} = \Gamma(j)$. Clearly x is in the allocation polytope. Let $i \in \mathcal{M}$. We need to verify that

$$(x_{i,j})_{j\in\mathcal{J}} \in \mathrm{conv}\{\chi \in [0,1]^{\mathcal{J}} : p^T\chi \geq (k-1)/k \text{ and } \chi_j \in \{0,1\} \text{ if } p_j \cdot 4 > (k-1)/k\}.$$

We define one vector for every $j_B \in \mathcal{J}$ with $p_{j_B} = 1$ and one vector for the small jobs, which depends on the machine it is used for.

$$\chi_j^{(j_B)} := \begin{cases} 1 & \text{if } j = j_B, \\ 0 & \text{otherwise,} \end{cases} \quad \text{and } \chi_j^{(i)} := \begin{cases} \frac{k-1}{k} & \text{if } p_j = 1/5 \text{ and } i \in \Gamma(j), \\ 0 & \text{otherwise.} \end{cases}$$

Note that we have $p^T\chi^{(j_B)} = p_{j_B} = 1 \geq \frac{k-1}{k} = T$ as well as $p^T\chi^{(i)} = 5 \cdot \frac{1}{5} \cdot \frac{k-1}{k} = \frac{k-1}{k} = T$. The integrality constraint is satisfied for all jobs of size 1 and since $1/5 \cdot 4 \leq (k-1)/k$, for sufficiently large k, it is not necessary for the small jobs. Also, it holds that $(x_{i,j})_{j\in\mathcal{J}} = \sum_{j_B\in\mathcal{J}:p_{j_B}=1} 1/k \cdot \chi^{(j_B)} + 1/2 \cdot k/(k-1) \cdot \chi^{(i)}$, as shown below.

Let j_B be a job of size 1. Then all vectors but $\chi^{(j_B)}$ have 0 for j_B, hence $x_{i,j_B} = 1/k \cdot \chi_{j_B}^{(j_B)}$. Next, let j be a job of size $1/5$. Then $x_{i,j} = 1/2 = 1/2 \cdot k/(k-1) \cdot \chi_j^{(i)}$ if $i \in \Gamma(j)$ and $x_{i,j} = 0 = 1/2 \cdot k/(k-1) \cdot \chi_j^{(i)}$, otherwise.

For this instance, we get a gap that approaches $5/2$ as k tends to infinity. Similar constructions work when using 14 jobs of size $1/7$, 18 jobs of size $1/9$, etc. The lower bound

becomes weaker the smaller the size is chosen, but it always stays strictly above 4, the best lower bound known for the configuration LP.

D Analysis (Makespan)

▶ **Theorem 19.** *The algorithm terminates after finitely many iterations of the main loop.*

Proof. Let ℓ be the length of L. We define a potential function

$$\Phi(L) = (g, s_0, s_1, s_2, \ldots, s_\ell, -1),$$

where g is the number of good machines and s_k is the number of pairs $(i,j) \in \mathcal{M} \times \mathcal{J}$ where i repels j w.r.t. $L_{\leq k}$ and $\sigma(j) \neq i$. Note that the length of $\Phi(L)$ is bounded by $|\mathcal{M}| \cdot |\mathcal{J}| + 2$ and every component is an integer between -1 and $|\mathcal{J}| \cdot |\mathcal{M}|$. Thus, the range of the potential function is finite.

We will show that the vector increases lexicographically after every iteration of the main loop; hence the running time is bounded by the number of such vectors times the maximum number of operations in an iteration and is in particular finite. For the lexicographic increase, consider two cases. Either a new move is added, which increases the potential function by replacing the last component with some non-negative value, or a move is performed.

If the move turns a bad machine good, g increases and thereby $\Phi(L)$ increases lexicographically as well. Otherwise, let $\ell' \leq \ell$ be the length of the list after a move (j,i) is performed. Recall the algorithm prevents jobs repelled by a machine from being moved there, i.e., j is not repelled by i machine w.r.t. $L_{\leq \ell'}$. Moreover, the set of repelled jobs by some machine w.r.t. $L_{\leq 0}, \ldots, L_{\leq \ell'}$ can only grow. This can be observed from the definition of repelled jobs. It follows that $s_0, \ldots, s_{\ell'}$ do not decrease. Finally, since j is repelled by its previous machine i' w.r.t. $L_{\leq \ell'}$ and after the move $\sigma(j) \neq i'$ holds, the value of $s_{\ell'}$ has increased. ◀

▶ **Theorem 20.** *If* LP_2 *is feasible, the algorithm always has an operation to perform.*

Proof. As in the algorithm, call a job j small if $p_j < 1/2 \cdot T = 1/r \cdot T$ and big otherwise. Suppose toward contradiction, there are bad machines, no move in L is valid, and no move can be added to L. We will construct values $(z_j)_{j \in \mathcal{J}}$, $(y_i)_{i \in \mathcal{M}}$ with the properties as in Lemma 12 and thereby show that LP_2 is infeasible.

For every $j \in \mathcal{J}$ let $z_j = \min\{p_j/T, 5/6\}$ if j is repelled by $\sigma(j)$ and $z_j = 0$ otherwise. Let $y_i := 1$ if $i \in \mathcal{M}$ repels all jobs and $y_i = z(\sigma^{-1}(i))$ otherwise.

▶ **Fact 21.** *Let j be a small job not repelled by $i \in \Gamma(j)$. Then $z_j = 0$.*

If $i = \sigma(j)$, this is by definition. In the other case, assume toward contradiction $z_j \neq 0$, i.e. j is repelled by $\sigma(j)$. (j,i) cannot be in L or else i would repel small jobs. Since (j,i) also cannot be added to L, i must repel j. A contradiction.

Let $i \in \mathcal{M}$ and $\chi \in [0,1]^{\mathcal{J}}$ with $p^T\chi \leq T$, $\chi_j \in \{0,1\}$ for every big job, and $\chi_j = 0$ if $i \notin \Gamma(j)$. We must show that $z^T\chi \leq y_i$.

If $y_i = 1$ it holds because of $z^T\chi \leq p^T\chi/T \leq 1$. We assume w.l.o.g. that i does not repel all jobs and thus $y_i = z(\sigma^{-1}(i))$. In particular, i does not repel small jobs that are on other machines. If $\chi_j = 0$ or $z_j = 0$ for all big jobs j, then with Fact 21 we get $z^T\chi \leq z(\sigma^{-1}(i))$. Also, there can be at most one big job j_B with $\chi_{j_B} = 1$, since it has $p_{j_B} > 1/2 \cdot T$ and thus $p^T\chi$ would be greater than T, otherwise.

We recap: The only interesting case is when $y_i = z(\sigma^{-1}(i))$, there is one big job j_B with $\chi_{j_B} z_{j_B} = \min\{p_{j_B}/T, 5/6\}$, and all other big jobs j'_B have $\chi_{j'_B} z_{j'_B} = 0$.

1. **Case: j_B is repelled by i.** This must be because there is some move $(j_k, i_k) = L_k$ such that $t(L_{\leq k}, (j_k, i_k)) \geq p_{j_B}$ (big-big). Recall that $t(L_{\leq k}, (j_k, i_k))$ is the minimum t with

   $$p(\{j \in \sigma^{-1}(i) : j \in S_i(L_{\leq k}) \text{ or } 1/2 < p_j \leq t\}) + p_{j_k} > 11/6.$$

   In particular, there must be a big job $j'_B \in \sigma^{-1}(i)$ with $t(L_{\leq k}, (j_k, i_k)) = p_{j'_B} \geq p_{j_B}$ and j'_B is also repelled by i (big-big). Using Fact 21 we get

   $$z^T\chi = \sum_{j \in \mathcal{J}} z_j\chi_j \leq z(S_i(L)) + z_{j_B} \leq z(S_i(L)) + z_{j'_B} \leq z(\sigma^{-1}(i)) = y_i.$$

2. **Case: j_B is not repelled by i.** Then (j_B, i) must already be in L, i.e., $L_k = (j_B, i)$ for some $k \leq \ell$, and since i does not repel all jobs, rule (big-big) must apply for this move. Let $R = \{j \in \sigma^{-1}(i) : j \in S_i(L_{\leq k}) \text{ or } 1/2 < p_j \leq t(L_{\leq k}, (j_B, i))\}$. Then

   $$p(R) + p_{j_B} > 11/6 \cdot T \geq p^T\chi + 5/6 \cdot T.$$

   Furthermore, all jobs in R are also repelled by i. If $z_{j'_B} = 5/6$ for some $j'_B \in R$, like in the previous case we get $z^T\chi \leq z(S_i(L)) + z_{j_B} \leq z(S_i(L)) + z_{j'_B} \leq z(\sigma^{-1}(i))$. Otherwise, we have $z_{j'} = p_{j'}/T$ for all $j' \in R$. Thus,

   $$\begin{aligned} z^T\chi = z_{j_B} + \sum_{j \in \mathcal{J} \setminus \{j_B\}} z_j\chi_j &\leq z_{j_B} + \sum_{j \in \mathcal{J} \setminus \{j_B\}} p_j\chi_j/T = z_{j_B} + (p^T\chi - p_{j_B})/T \\ &< z_{j_B} + (p(R) - 5/6 \cdot T)/T \leq p(R)/T \leq z(\sigma^{-1}(i)). \end{aligned}$$

It remains to show that $\sum_{j \in \mathcal{J}} z_j > \sum_{i \in \mathcal{M}} y_i$. We prove that, with amortization, good machines satisfy $z(\sigma^{-1}(i)) \geq y_i$ and on bad machines strict inequality holds.

Let i be a bad machine. Then i repels all jobs (in particular those in $\sigma^{-1}(i)$). Hence,

$$z(\sigma^{-1}(i)) \geq 5/6 \cdot p(\sigma^{-1}(i))/T > 55/36 > y_i.$$

For good machines that do not repel all jobs, equality holds by definition. We will partition those good machines that do repel all jobs into those $i \in \mathcal{M}$ which have $(j, i) \in L$ for a small job j (case small-all) and those that do not (case big-all).

▶ **Fact 22.** *There are at least as many machines of case (small-all) as there are of case (big-all).*

The proof of Fact 22 is postponed until after the main proof. Let i be a machine of case (big-all). Then there is a big job j_B with $(j_B, i) \in L$ and this move is not valid. Either there is a job $j \in \sigma^{-1}(i)$ with $z_j = 5/6$ or $z_j = p_j/T$ for all $j \in \sigma^{-1}(i)$. Thus,

$$\begin{aligned} z(\sigma^{-1}(i)) &\geq \min\{5/6,\ p(\sigma^{-1}(i))/T\} \\ &\geq \min\{5/6,\ 11/6 - p_{j_B}/T\} \\ &\geq 5/6 = y_i - 1/6. \end{aligned}$$

Next, let i be a machine of case (small-all). Then there is a move $(j_S, i) \in L$ with j_S small. Of course, this move is not valid. In the following, we distinguish between the cases where $\sigma^{-1}(i)$ has no job j with $z_j = 5/6$, one such job, or at least two. Note that these jobs have $p_j \leq T$.

$$\begin{aligned} z(\sigma^{-1}(i)) &\geq \min\{p(\sigma^{-1}(i))/T,\ (p(\sigma^{-1}(i)) - T)/T + 5/6,\ 10/6\} \\ &\geq \min\{11/6 - p_{j_S}/T - 1/6,\ 10/6\} \\ &\geq 7/6 = y_i + 1/6. \end{aligned}$$

Because of Fact 22, we can amortize case (small-all) and case (big-all) and get

$$\sum_{j\in\mathcal{J}} z_j = \sum_{i\in\mathcal{M}} z(\sigma^{-1}(i)) > \sum_{i\in\mathcal{M}} y_i. \qquad \blacktriangleleft$$

Proof of Fact 22. Let $L_{b_1}, L_{b_2}, \ldots, L_{b_h}$ be the moves that correspond to case (big-all) machines, i.e., for each $(j_k, i_k) = L_{b_k}$ $(k \leq h)$, i_k repels all jobs and j_k is big.

We argue that there are $L_{s_1}, L_{s_2}, \ldots, L_{s_h}$ such that $b_1 < s_1 < b_2 < s_2 < \ldots b_h < s_h$, where for each $(j_k, i_k) = L_{s_k}$ $(k \leq h)$, j_k is small and therefore i_k is a case (small-all) machine.

Let $k \leq h$. A critical argument is that the algorithm prefers moves of small jobs over those of big jobs. In particular, when $L_{b_{k+1}}$ is added, there does not exist a small job move that it can add instead. Either L_{b_k} has already been subject to rule (big-all) when it was added or it turned to this after a repelled job was removed. Either way, at this time it was the last move in L and we had that

$$p(\{j \in \sigma^{-1}(i_k) : p_j > 1/2\}) + p(S_{i_k}(L_{\leq b_k})) + p_{j_k} \leq 11/6 \cdot T < p(\sigma^{-1}(i_k)) + p_{j_k},$$

where the first inequality comes from rule (big-all) and the second one from the fact that (j_k, i_k) is and was not valid. Hence, there must be a small job j in $\sigma^{-1}(i)$ which is not in $S_{i_k}(L_{\leq b_k})$. By definition of $S_{i_k}(L_{\leq b_k})$, j has a machine $i \in \Gamma(j)$ by which it was not repelled. Therefore there has been a small job move that could be added. This means $L_{b_{k+1}}$ was only added after a small job move has been. $\blacktriangleleft$