A $(2+\varepsilon)$-approximation for scheduling parallel jobs in platforms

Pierre-Francois Dutot, Klaus Jansen, Christina Robenek, Denis Trystram

Bericht Nr. 1217
February 2013
ISSN 2192-6247

Institut für Informatik der
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40
D – 24098 Kiel

A $(2 + \varepsilon)$-approximation for scheduling parallel jobs in platforms

Pierre-Francois Dutot, Klaus Jansen, Christina Robenek, Denis Trystram

Bericht Nr. 1217
February 2013
ISSN 2192-6247

e-mail: $\{kj, cot\}$@informatik.uni-kiel.de,
$\{dutot, trystram\}$@imag.fr

Research supported by German Research Foundation (DFG) project JA612/12-2.
Dieser Technische Bericht enthält Teile der Dissertation von Christina Robenek

A $(2+\varepsilon)$-approximation for scheduling parallel jobs in platforms

Pierre-Francois Dutot Klaus Jansen Christina Robenek
Denis Trystram

February 6, 2013

Abstract

We consider the problem of SCHEDULING PARALLEL JOBS IN HETEROGENEOUS PLATFORMS: We are given a set $\mathcal{J} = \{1, \ldots, n\}$ of n jobs, where a job $j \in \mathcal{J}$ is described by a pair (p_j, q_j) of a processing time $p_j \in \mathbb{Q}_{>0}$ and the number of processors $q_j \in \mathbb{N}$ that are required to execute j. We are also given a set $\mathcal{B}$ of N heterogeneous platforms $P_1, \ldots, P_N$, where each P_i contains m_i processors for $i \in \{1, \ldots, N\}$. The objective is to find a schedule for the jobs in the platforms minimizing the makespan, i.e. the latest finishing time of a job. Unless $\mathcal{P} = \mathcal{NP}$ there is no approximation algorithm with absolute ratio strictly better than 2 for the problem. We give a $(2+\varepsilon)$-approximation for the problem improving the previously best known absolute approximation ratio 3.

1 Introduction

This paper considers the problem of SCHEDULING PARALLEL JOBS IN HETEROGENEOUS PLATFORMS (SPP): We are given a set $\mathcal{J} = \{1, \ldots, n\}$ of n jobs, where a job $j \in \mathcal{J}$ is described by a pair (p_j, q_j) of a processing time $p_j \in \mathbb{Q}_{>0}$ and the number of processors $q_j \in \mathbb{N}$ that are required to execute j. We are also given a set $\mathcal{B}$ of N platforms $P_1, \ldots, P_N$, where each P_i contains a set M_i of $|M_i| = m_i$ processors for $i \in [N] := \{1, \ldots, N\}$. In general we assume that the numbers m_i may be different, that are *heterogeneous platforms.* If all values m_i are equal we have *identical platforms*. For heterogeneous platforms we may assume $m_1 \geq \ldots \geq m_N$. A schedule is an assignment $a : \mathcal{J} \to \bigcup_{i=1}^{N} 2^{M_i} \times \mathbb{Q}_{\geq 0}$, that assigns every job j to a starting time t_j and to a subset $A_j \subset M_i$ of the processors of a platform P_i with $|A_j| = q_j$. Obviously, a job j can only be scheduled in platform P_i if $m_i \geq q_j$. A schedule is feasible if every processor in every platform executes at most one job at any time. The objective is to find a feasible schedule with minimum makespan $\max_{i\in[N]} C^{(i)}_{\max}$, where $C^{(i)}_{\max} = \max_{\{j|A_j\subset M_i\}} t_j + p_j$ denotes the local makespan for platform P_i. We denote with $\mathrm{OPT}_{\mathrm{SPP}}(\mathcal{J}, \mathcal{B})$ the optimum value for the makespan of a schedule for the jobs in $\mathcal{J}$ into the platforms in $\mathcal{B}$.

By reduction from 3-Partition it follows that SPP is strongly $\mathcal{NP}$-hard even for identical platforms. Moreover, there exists no approximation algorithm with absolute ratio strictly better than 2, unless $\mathcal{P} = \mathcal{NP}$.

For $N = 1$ the problem is equal to SCHEDULING PARALLEL JOBS, in the relevant literature denoted with $P|size_j|C_{\max}$. This problem is strongly $\mathcal{NP}$-hard even for a constant number of processors $m \geq 5$ [8]. By reduction from PARTITION it can be shown that there is no approximation algorithm for $P|size_j|C_{\max}$ with ratio strictly less than 1.5, unless $\mathcal{P} = \mathcal{NP}$. If we constrain the co-domain of the assignment a further and assume identical platforms the problem is equivalent to STRIP PACKING (for $N = 1$) and MULTIPLE STRIP PACKING($N \geq 2$): In addition to $A_j \in \bigcup_{i=1}^{N} 2^{M_\ell}$ we postulate that A_j is equal to a set of consecutively numbered processors for every job $j \in \mathcal{J}$. Every job then corresponds to a rectangle of width q_j and height p_j. Keep in mind here, that in general because of this contiguity constraint, algorithms for SPP cannot be directly applied to MULTIPLE STRIP PACKING, since rectangles may be cut. But the optimal value for MULTIPLE STRIP PACKING is an upper bound for the optimal value for the corresponding SPP problem with identical platforms. Interestingly, fractional versions of both problems coincide and therefore a solution of FRACTIONAL (MULTIPLE) STRIP PACKING gives a fractional solution for SPP with identical platforms.

1.1 Related Work

There are several approximation algorithms for SCHEDULING PARALLEL JOBS. To name a few, the best known is List Schedule by Garey and Graham [10]. It was shown by Feldmann et al. that List Schedule has absolute approximation ratio $(2 - 1/m)$ [9] using a dynamic and slightly different model. If the number of processors is bounded by a constant the problem admits a PTAS [1, 15]. In case that the number of machines is polynomially bounded in the number of jobs a $(1.5 + \varepsilon)$-approximation for the contiguous case (where a job has to be executed on processors with consecutive adresses) and a $(1 + \varepsilon)$-approximation for the non-contiguous problem were given in [17]. Recently, for an arbitrary number of processors Jansen gave a tight approximation algorithm with absolute ratio $1.5 + \varepsilon$ in [14]. Also for an arbitrary number of processors the contiguous case of $P|size_j|C_{\max}$ is closely related to STRIP PACKING as described above. A vast number of approximation algorithms for STRIP PACKING have been developed during the last decades. [7, 27, 24, 12, 16] One of the most powerful results for STRIP PACKING is an asymptotic fully polynomial time approximation scheme given by Kenyon and Rémila based on a linear program relaxation for BIN PACKING [18]. For any accuracy $\varepsilon > 0$ their algorithm produces a $(1 + \varepsilon)$-approximative packing plus an additive height of $\mathcal{O}(1/\varepsilon^2)h_{\max}$, where $h_{\max}$ denotes the height of the tallest rectangle. Recently, we showed that the additive term can be reduced to $\mathcal{O}(1/\varepsilon \log(1/\varepsilon))h_{\max}$ using a more sensitive rounding technique [5]. We will use the algorithm in [18] as a subroutine and refer to it as the KR algorithm.

A similar problem is SCHEDULING MALLEABLE JOBS. Here the processing time of a job j depends on the number of allotted machines and can be described by a function $p_j : [m_N] \longrightarrow \mathcal{Q}^+ \cup \infty$, where $p_j(k)$ is the length of job j running on k parallel processors of a platform. The PTAS in [15] does also apply for malleable jobs if the number of processors is constant. Interestingly, in [17] it can be derived from the paper that the algorithms can also be applied to malleable jobs without using the assumption $m \leq poly(n)$. In [22] Mounié et al. presented a $(1.5 + \varepsilon)$-approximation for scheduling malleable jobs with processing times given by monotone functions where the jobs are assigned to processors of consecutive addresses. The running time of this algorithm depends on the number of processors. An AFPTAS for scheduling malleable jobs on an arbitrary number of processors is given in [13].

For SCHEDULING PARALLEL JOBS IN PLATFORMS (SPP) Tchernykh et al. presented in [28] an algorithm with absolute ratio 10. Earlier Remy claimed in [23] that the approximation ratio 2 of List Schedule is preserved when applied to SPP WITH IDENTICAL PLATFORMS while in [28] and again later in [25] it is shown that List Schedule cannot even guarantee a constant approximation ratio for this problem. Schwiegelshohn et al. [25] achieved absolute approximation ratio 3 for SPP, and ratio 5 for the SPP WITH RELEASE TIMES.

For SPP WITH IDENTICAL PLATFORMS, we proposed a low cost approximation algorithm with absolute ratio $5/2$ in [3]. Recently, we presented a low-cost tight 2-approximation for this problem in case that no job does require more than half of the processors [6]. We were also able to extend our result in [3] to a fast $5/2$-approximation to SPP for HETEROGENEOUS PLATFORMS under the additional constraint that every job can be scheduled in each platform [4].

1.2 New Result

As the platforms may have different numbers of processors the 2-approximation given in [5] for MULTIPLE STRIP PACKING and the $(2 + \varepsilon)$-approximation given by

Ye et al. in [29] cannot be applied to heterogeneous platforms they work only for identical platforms. For the same reason the algorithm in [6] does not work. Note that the 2-approximation for MSP given in [5] cannot even be applied to schedule parallel jobs: In case of a constant number of platforms, a subroutine for rectangle packing with area maximization is used that cannot be applied to select jobs. So currently, the best known absolute ratio for an approximation algorithm for SPP without any additional constraints is 3 given in [25]. This ratio is achieved by a clever combination of several list procedures. In this article we present a polynomial time algorithm with absolute ratio $(2+\varepsilon)$ for SPP improving the previously best known absolute ratio 3 for an approximation algorithm for SPP. Moreover, we nearly close the gap between the inapproximability bound of 2 and the currently best absolute ratio.

Theorem 1.1. *For any accuracy $\varepsilon > 0$ there is an algorithm that for a set $\mathcal{J}$ of n parallel jobs and a set $\mathcal{B}$ of heterogeneous platform generates a schedule for $\mathcal{J}$ into the platforms in $\mathcal{B}$ with makespan at most $(2+\varepsilon)\mathrm{OPT}_{SPP}(\mathcal{J},\mathcal{B})$. The algorithm has running time $g(1/\varepsilon) \cdot n^{\mathcal{O}(f(1/\varepsilon))}$ for some functions g, f with $g(1/\varepsilon), f(1/\varepsilon) = 2^{\mathcal{O}(1/\varepsilon \log(1/\varepsilon))}$.*

1.3 Methods and Overview

To obtain a simpler structure for the set of platforms $\mathcal{B}$ we use a new technique to group and round the platforms by the number of processors: Initially, we partition the platforms into a set $\mathcal{B}_0$ containing a constant number of the largest platforms, and a set $\mathcal{B}_1$ containing the remaining smaller platforms with less processors. For a certain number L the platforms in $\mathcal{B}_1$ are grouped and rounded obtaining a set $\tilde{\mathcal{B}}_1$ that contains L groups $\widetilde{B}_1, \ldots, \widetilde{B}_L$ of equal constant cardinality, so that that the platforms in each group $\widetilde{B}_\ell$ are identical, see Section 2.1. Later we convert a solution for the rounded platforms $\mathcal{B}_0 \cup \tilde{\mathcal{B}}_1$ into one for the original ones in $\mathcal{B} = \mathcal{B}_0 \cup \mathcal{B}_1$, see Figure 6.

Using gap creation [16] we simplify the structure of an optimum solution in $\mathcal{B}_0$, see Section 2.2 and Figure 3. Then we allocate a subset of jobs with large processing time jobs in $\mathcal{B}_0$. The main difficulty here is to place the correct subset of large narrow jobs, that have large processing time and require only few processors, since we cannot enumerate an assignment for them in polynomial time. Instead we guess an approximate gap structure for them.

With a skillful linear program relaxation (refer to Section 2.3) we fractionally assign a subset of large narrow jobs to the guessed gaps in $\mathcal{B}_0$, subsets of jobs with small and medium processing time to $\mathcal{B}_0$, and the remaining jobs to $\tilde{\mathcal{B}}_1$. In this new approach we have both, horizontal and vertical fractions of large narrow jobs, which are related by a nice covering constraint. Interestingly, we can apply a result for scheduling unrelated machines [21] to round those fractions to integral jobs producing only a small error even though there are different kinds of fractions. The LP in 2.3 also produces a fractional schedule for $\tilde{\mathcal{B}}_1$. Here, the crucial part is to round the fractional schedule to an integral one without loosing too much. Therefore, the jobs involved in that fractional schedule have harmonically rounded processing times, see Section 2.3.1. That is, relatively large processing times are rounded up to the next value of the form $1/q$, $q \in \mathbb{N}$. We use the harmonically rounded processing times for rounding the fractional schedule in $\tilde{\mathcal{B}}_1$ to an integral one using an idea by Bansal et al. [2] based on the fact that any integer can be represented as an integral multiple of $1/q$, see Lemma 2.5. Again the large narrow jobs are difficult as for one large narrow job we may produce fractions referring to different processing times in $\mathcal{B}_0$ and $\tilde{\mathcal{B}}_1$. This problem is also cleverly modelled and solved in our LP-relaxation.

1.4 Principles and Notations

First we define some notations and recall some well-known packing and scheduling principles. For $j \in \mathcal{J}$ we define the *size* of a jobs as $q_j p_j$ and $SIZE(\mathcal{J}) := \sum_{j\in\mathcal{J}} q_j p_j$ for a set of jobs. With $p_{\max} := \max_{j\in\mathcal{J}} p_j$ we denote the largest processing time of a job. A rectangle is a pair $r = (w_r, h_r)$ of a width $w_r \in \mathbb{Q}_{>0}$ and height $h_r \in \mathbb{Q}_{>0}$. The *size* of r is defined as $w_r h_r$. The size of a set of rectangles $\mathcal{R}$ is $SIZE(\mathcal{R}) := \sum_{r\in\mathcal{R}} w_r h_r$. A *two-dimensional bin* of width x and height y will be denoted with $b(x, y)$. In this context a *strip* is a bin of width 1 and infinite height $b(1, \infty)$. We also use the notation $b(x, \infty)$ for a strip of width x. If $x \in \mathbb{N}$ a strip $b(x, \infty)$ corresponds to a platform with x processors.

Geometric rounding: For a set $\mathcal{R}_{wide}$ of rectangles $r = (w_r, h_r)$ we obtain the geometrically rounded set $\mathcal{R}_{sup}$ with only M different width in the following way: Order the rectangles by non-increasing width and stack them left aligned on top of each other, starting with the widest rectangles. Let H denote the height of the stack. Then draw horizontal lines at heights $(iH)/M$ for $i = 0, 1, \ldots, M$ through the stack. For $i = 0, 1, \ldots, M-1$ group together those rectangles that lie completely with their interior between the ith and $(i+1)th$ line or intersect with their interior the $(i+1)th$ line. In every group round up the width of every rectangle to the width of the widest rectangles contained in this group.

Fractional strip packing: For a set of rectangles $\mathcal{R}$ with $w_r \in (0, w]$ for $r \in \mathcal{R}$ a fractional strip packing of height $h > 0$ into a strip $b(w, \infty)$ corresponds to a feasible solution of a linear program of the form

$$\min\left\{\sum_i x_i | \sum_{i:C_i(r)=1} x_i \geq h_r \;\; r \in \mathcal{R}, x_i \geq 0\right\} \tag{1}$$

with cost at most h. The variable x_i denotes the height (or length) of a configuration $C_i : \mathcal{R} \to \{0,1\}$, that is a function that represents a subset of rectangles that can be placed next to each into the strip $b(w,\infty)$, i.e. $\sum_{\{r\in\mathcal{R}|C_i(r)=1\}} w_r \le w$. If $x_i > 0$, for every rectangle with $C_i(r) = 1$ a fraction of height x_i and width w_r is placed into the strip. If for $\mathcal{R}$ there exists a fractional strip packing of height h, we say $\mathcal{R}$ *fits fractionally into* $b(w,h)$. The content of the following Lemma is given in [18].

Lemma 1.2. *Let $\mathcal{R}$ be a set of rectangles $r = (w_r, h_r)$ with width $w_r \in (0,w]$ and heights $h_r \in (0,1]$. Let $\varepsilon' > 0$ and $M := 1/\varepsilon'^2$ and let $\mathcal{R}_{wide} := \{r \in \mathcal{R} | w_r > \varepsilon' w\}$ and $\mathcal{R}_{narrow} := \mathcal{R} \setminus \mathcal{R}_{wide}$. If $\mathcal{R}_{wide}$ fits fractionally into a bin $b(w,h)$, then $\mathcal{R}_{sup}$ fits fractionally into bin $b(w, h(1+\varepsilon'))$. Moreover, $\mathcal{R}$ can be packed integrally into a strip $b(w,\infty)$ with height at most $\frac{(1+\varepsilon')h}{1-\varepsilon'} + (4M+1)\max_{r\in\mathcal{R}} h_r$.*

The above result uses the fact that for $\mathcal{R}_{sup}$ the rank of the constraint matrix of (1) is bounded by the number of different width M. Because of this, there exists a solution x of (1) with input $\mathcal{R}_{sup}$ with at most M non-zero entries. According to the solution x we can construct a fractional solution with at most $2M$ different configurations. An integral packing for $\mathcal{R}_{sup}$ is achieved by adding additional space of height $\max_{r\in\mathcal{R}_{sup}} h_r$ to each configuration and filling the rectangles integrally into them. The rectangles in $\mathcal{R}_{narrow}$ are placed next to the configurations and on top of the solution using NFDH policy.

Our algorithm considers two main scenarios for the shape of the platforms given by the input. For $\varepsilon > 0$ with $3/18 \ge \varepsilon$ and $\gamma = \frac{8}{3}N_1$, where $N_1 = \mathcal{O}(1/\varepsilon^4)$ (specified in Lemma 2.6) we distinguish:

1. For all $i \in [N]$ we have $\frac{m_1}{m_i} \le \gamma$.
2. There is a number $K \in [N]$ with $\frac{m_1}{m_i} \le \gamma$ for all $i \le K$ and $\frac{m_1}{m_i} > \gamma$ for all $i > K$.

2 Case 1: Similar Platforms

2.1 Platform Rounding

For $N_0 = 2(2N_1 + 1)$ we partition the set of platforms $\mathcal{B}$ into $L+1$ groups $B_0, B_1, \ldots, B_L$ by $L-$times collecting the N_1 smallest platforms where $L := \max\left\{0, \lfloor \frac{N-N_0}{N_1} \rfloor\right\}$. Let

$$\mathcal{B}_0 = B_0 := \{P_1, \ldots, P_{N-LN_1}\}$$

and for $\ell \in [L]$ define

$$B_\ell := \{P_{N-(L-(\ell-1))N_1+1}, \ldots, P_{N-(L-\ell)N_1}\}$$

and $\mathcal{B}_1 = \bigcup_{\ell=1}^{L} B_\ell$. Therefore, group $\mathcal{B}_1$ is further partitioned into several groups B_ℓ of equal constant cardinality. Each group $B_\ell \subseteq \mathcal{B}_1$ contains exactly N_1 platforms.

Group $\mathcal{B}_0$ contains a constant number of platforms, moreover we have $5N_1 + 2 = N_0 + N_1 > |\mathcal{B}_0| \geq N_0$.

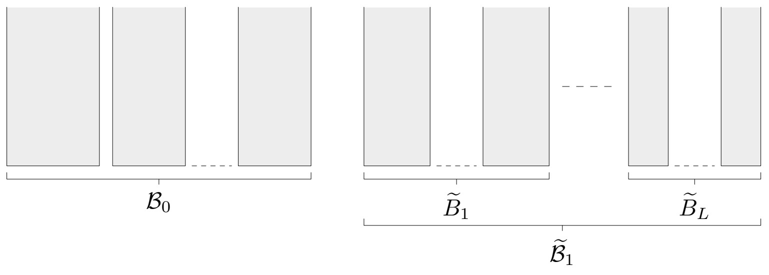

Figure 1: Grouping and rounding platforms in case 1.

In each group B_ℓ, $\ell \in [L]$ we round the number of processors of each platform up to the number of processors $\widetilde{m}_\ell := m_{N-(L-(\ell-1))N_1+1}$ of the largest platform $P_{N-(L-(\ell-1))N_1+1}$ contained in this group and denote with $\widetilde{B}_\ell$ the group of rounded platforms, see also Figure 1. We will compute a schedule for $\mathcal{B}_0 \cup \widetilde{\mathcal{B}}_1$, where $\widetilde{\mathcal{B}}_1 = \bigcup_\ell \widetilde{B}_\ell$, and convert this solution into a solution for $\mathcal{B}_0 \cup \mathcal{B}_1$ using a shifting argument, see Figure 6.

It might be possible that the number of platforms is bounded by a constant $N < N_0 + N_1$. In this case we have only one group $\mathcal{B}_0$ and do not round the platforms. The algorithm simplifies at some points as it only performs the steps concerning $\mathcal{B}_0$.

2.2 Simplifying the Structure of an Optimum Solution in $\mathcal{B}_0$

Consider an optimum solution with makespan equal to 1 and denote with $\mathcal{J}^\star(\mathcal{B}_0)$ the set of jobs that are scheduled in $\mathcal{B}_0$ by the optimum solution. Using the gap creation technique [16] we find a subset of jobs with medium processing time

$$\mathcal{J}^\star_{medium}(\mathcal{B}_0) \subseteq \mathcal{J}^\star(\mathcal{B}_0)$$

and small total load. We can remove these medium jobs from the instance and schedule them later on top of the solution constructed for the reduced instance only slightly increasing the makespan.

Define $\sigma_0 = \frac{\varepsilon}{9}$ and $\sigma_{k+1} = \sigma_k^5$ and sets $\mathcal{J}_k = \{j \in \mathcal{J} | p_j \in (\sigma_k, \sigma_{k-1}]\}$ for $k \geq 1$. Let $\mathcal{J}_k^\star(\mathcal{B}_0)$ and $\mathcal{J}_k^\star(\mathcal{B}_1)$ denote those jobs in $\mathcal{J}_k$ that are scheduled by an optimum

Algorithm 1 $(2+\varepsilon)$-Algorithm

```
Input: $\mathcal{J}$ , $\varepsilon > 0$
Output: A schedule of length $(2+\mathcal{O}(\varepsilon))\mathrm{OPT}_{\mathrm{SPP}}(\mathcal{J})$
1: For a certain constant $N_1 = \mathcal{O}(1/\varepsilon^3 \log(1/\varepsilon))$ partition the set of platforms into
   $L+1$ groups $\mathcal{B}_0, B_1 \ldots, B_L$ and let $\mathcal{B}_1 := \bigcup_{\ell=1}^{L} B_\ell$.
2: Round the number of processors of the platforms in each group $B_\ell$ and obtain $\widetilde{\mathcal{B}}_1$
   containing groups $\widetilde{B}_\ell$ of $N_1$ similar platforms
3: for a candidate value for the makespan $T \in \left[\frac{SIZE(\mathcal{J})}{\sum_{i=1}^{N} m_i}, np_{\max}\right]$ do
4:     for $k \in \{1, \ldots, \frac{|\mathcal{B}_0|}{\varepsilon}\}$ do
5:         Let $\delta := \sigma_{k-1}$ where $\sigma_0 = \varepsilon/20$, $\sigma_{k+1} = \sigma_k^5$ for $k \geq 1$.
6:         For $\delta$ distinguish small, medium, and large jobs
7:         Round the processing times and possible starting times of large jobs to inte-
   gral multiples $\delta^2$.
8:         For $\alpha = \delta^4/16$ distinguish wide and narrow large jobs.
9:         Enumerate an assignment vector $V$ of large wide jobs to $\mathcal{B}_0$ and let
   $\mathcal{J}_{la-wi}(\mathcal{B}_0)$ denote the selected jobs.
10:        for an assignment vector $V$ of large wide jobs do
11:            Approximately guess the total load $\Pi$ of large narrow jobs for each start-
   ing time and height in every platform of $\mathcal{B}_0$ and block gaps corresponding to $\Pi$.
12:            for a guess $\Pi$ do
13:                Compute free layers of height $\delta^2$ in $\mathcal{B}_0$.
14:                Round the processing times $p_j$ of the jobs $\mathcal{J}' = \mathcal{J} \setminus \mathcal{J}_{la-wi}(\mathcal{B}_0)$ har-
   monically.
15:                Compute a solution of LP (2)
16:                if There is no feasible solution for (2) then
17:                    Discard the guess $\Pi$ and take another one and go back to Step 13.
   If all guesses have failed discard $V$, take another and go back to Step 11. If all pairs
   $(V, \Pi)$ have failed, increase $k$ and go to Step 5.
18:                end if
19:                Round the solution of (2) using a result of Lenstra et al. [21] and ob-
   tain an almost integral assignment of
     • a subset of the small jobs to the free layers in $\mathcal{B}_0$
     • a subset of the large narrow jobs to the gaps $\Pi$ in $\mathcal{B}_0$
     • the remaining jobs to the groups $\widetilde{B}_\ell$ in $\widetilde{\mathcal{B}}_1$.
20:                Pack small jobs with STRIP PACKING subroutine into the layers.
21:                for $\ell = 1, \ldots, L$ do
22:                    Pack the jobs assigned to $\widetilde{B}_\ell$ with 2D-BIN PACKING subroutine
   into at most $2N_1$ bins of size $[0, \widetilde{m}_\ell] \times [0, 1]$.
23:                end for
24:            end for
25:        end for
26:    end for
27: end for
28: Convert the schedule for $\mathcal{B}_0 \cup \widetilde{\mathcal{B}}_1$ into a schedule for $\mathcal{B}_0 \cup \mathcal{B}_1$
29: Schedule medium jobs in $\mathcal{J}_\tau(\mathcal{B}_0)$ in $P_1$
```

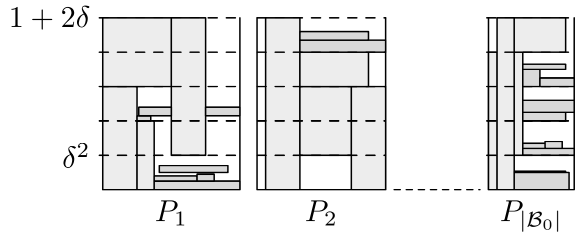

Figure 2: Simplified structure of an optimum solution in $\mathcal{B}_0$.

algorithm in $\mathcal{B}_0$ and $\mathcal{B}_1$, respectively. Using $T = 1$ we have

$$\sum_{k\geq 1}\sum_{j\in\mathcal{J}_k(\mathcal{B}_0)} p_j q_j \leq \sum_{i=1}^{|\mathcal{B}_0|} m_i \leq |\mathcal{B}_0| m_1.$$

Using the pigeonhole principle we proof the existence of a set $\mathcal{J}^\star_\tau(\mathcal{B}_0)$ with $\tau \in \{1,\dots,\frac{|\mathcal{B}_0|}{\varepsilon}\}$ so that $\sum_{j\in\mathcal{J}_\tau(\mathcal{B}_0)} p_j q_j \leq \varepsilon m_1$: If not, we have

$$\sum_{k\geq 1}^{|\mathcal{B}_0|/\varepsilon}\sum_{j\in\mathcal{J}_k(\mathcal{B}_0)} p_j q_j > |\mathcal{B}_0| m_1$$

which is a contradiction. Then we choose $\delta = \sigma_{\tau-1}$ and may assume that ε^{-1} is integral and thus

$$\delta^{-1} = \left(\frac{\varepsilon}{20}\right)^{-(5^{\tau-1})} = \left(\frac{20}{\varepsilon}\right)^{5^{\tau-1}}$$

is integral (if not we choose the next smaller value for ε). Furthermore, note that in the worst case

$$\delta^{-1} = \left(\frac{20}{\varepsilon}\right)^{\frac{|\mathcal{B}_0|}{\varepsilon}-1}.$$

We partition the jobs into

- small jobs $\mathcal{J}_{small} := \{j \in \mathcal{J} | p_j \leq \delta^5\}$,
- medium jobs $\mathcal{J}_{medium} := \mathcal{J}_\tau = \{j \in \mathcal{J} | p_j \in (\delta^5, \delta]\}$,
- large jobs with $\mathcal{J}_{large} := \{j \in \mathcal{J} | p_j \in (\delta, 1]\}$.

Scheduling the medium jobs in $\mathcal{J}^\star_\tau(\mathcal{B}_0)$ in the end on top of the largest platform P_1 using List Schedule [10] increases the makespan by at most

$$2\max\left\{(1/m_1)\sum_{j\in\mathcal{J}_\tau(\mathcal{B}_0)} p_j q_j, \max_{j\in\mathcal{J}_\tau} p_j\right\}$$
$$= 2\max\left\{\frac{\varepsilon m_1}{m_1},\delta\right\} \leq 2\max\{\varepsilon,\delta\} \leq 2\varepsilon.$$

For $\mathcal{B}_0$ we can now simplify the structure of the starting times and different processing times of large jobs. We round up the processing time of each job with processing time $p_j > \delta$ to $\bar{p}_j = h\delta^2$, the next integral multiple of δ^2 with $(h-1)\delta^2 < p_j \leq h\delta^2 = \bar{p}_j$, for $h \in \{\frac{1}{\delta}+1, \ldots, \frac{1}{\delta^2}\}$. Since there can be at most $1/\delta$ jobs with height $> \delta$ on each processor within each platform this increases the makespan in $\mathcal{B}_0$ by only $\delta^2/\delta = \delta$. The number of different large jobs sizes H is bounded by

$$H = \frac{1}{\delta^2} - (\frac{1}{\delta} + 1) + 1 \leq \frac{1}{\delta^2}.$$

In a similar way we round the starting time of each large job in $\mathcal{B}_0$ to $a\delta^2$, the next integral multiple of δ^2. This increases the makespan again by at most δ to $1 + 2\delta$. Therefore the large jobs have starting times $a\delta^2$ with $a \in \{0, 1, \ldots, \frac{1+2\delta}{\delta^2} - 1\}$ and the number of different starting times is $A = \frac{1+2\delta}{\delta^2}$.

An optimum schedule for $\mathcal{J}^\star(\mathcal{B}_0) \setminus \mathcal{J}^\star_\tau(\mathcal{B}_0)$ in $\mathcal{B}_0$ with rounded processing times $\bar{p}_j$ and rounded starting times for the large jobs has length at most $1 + 2\delta$. The schedule with simplified structure in $\mathcal{B}_0$ is illustrated in Figure 2.

Let $\tau \in \{1, \ldots, \frac{|\mathcal{B}_0|}{\varepsilon}\}$ be the current iteration step for finding $\mathcal{J}_\tau$ with the desired properties and $\delta = \sigma_{\tau-1}$. We enumerate the set of large wide jobs and approximately guess the structure of large narrow jobs in $\mathcal{B}_0$ that correspond to a good solution for the jobs with rounded processing times $\bar{p}_j$. We distinguish between wide and narrow large jobs and as follows. We may assume that $m_N \geq 32/\delta^4$. Otherwise the number of processors in P_1 and (since $m_1 \geq \ldots \geq m_N$) in every platforms is bounded by a constant

$$m_1 \leq m_N \gamma \leq \frac{32\gamma}{\delta^4}.$$

Thus, in every platform also the number of jobs that fit next to each other is bounded by $\frac{32\gamma}{\delta^4}$, moreover, the total number of large jobs in each platform is bounded by $\frac{32\gamma}{\delta^4} \cdot A$. Then we do not distinguish between large and narrow jobs and can enumerate the entire subset of large jobs that has to be scheduled in $\mathcal{B}_0$ as it is done for the large wide jobs in Section 2.2.1.

We choose $\alpha = \delta^4/16$. Then α satisfies $\alpha m_N \geq 2$, implying $\lfloor \alpha m_N \rfloor \geq \alpha m_N - 1 \geq \alpha m_N/2$. A job $j \in \mathcal{J}$ is called *wide* if $q_j \geq \lfloor \alpha m_N \rfloor$ and *narrow* otherwise. Furthermore distinguish

- large narrow jobs $\mathcal{J}_{la-na} := \{j \in \mathcal{J}_{large} | q_j \leq \lfloor \alpha m_N \rfloor\}$,
- large wide jobs $\mathcal{J}_{la-wi} := \{j \in \mathcal{J}_{large} | q_j > \lfloor \alpha m_N \rfloor\}$.

2.2.1 Assignment of Large Wide Jobs in $\mathcal{B}_0$

The number of large wide jobs that fit next to each other within one platform is bounded by $\frac{m_1}{\lfloor \alpha m_N \rfloor} \leq \frac{m_1}{\alpha m_N - 1} \leq \frac{m_1}{(\alpha m_N)/2} \leq (2\gamma)/\alpha$. Since large jobs have processing times $> \delta$, at most $\frac{1+2\delta}{\delta}$ rounded large jobs can be finished on one processor before time $1 + 2\delta$. Therefore, the number of large wide jobs that have to be placed in one platform in $\mathcal{B}_0$ is bounded by $\frac{2\gamma}{\alpha} \cdot \frac{1+2\delta}{\delta}$. Furthermore, in every platform a large jobs can have A different starting times. Each possible assignment of large wide jobs to platform and starting time can be represented by a tuple of vectors $V = (v_1, \ldots, v_{|\mathcal{B}_0|}) \in \left(([n] \cup \{0\})^{A \cdot \frac{2\gamma}{\alpha} \cdot \frac{1+2\delta}{\delta}}\right)^{|\mathcal{B}_0|}$. The running time of a dynamic program to compute such an assignment is equal to the number of possible vectors which is bounded by $(n+1)^{|\mathcal{B}_0| \cdot A \cdot \frac{2\gamma}{\alpha} \cdot \frac{1+2\delta}{\delta}}$. Let $\mathcal{J}_{la-wi}(\mathcal{B}_0)$ denote the set of large wide jobs selected and let $\mathcal{J}' := \mathcal{J} \setminus \mathcal{J}_{la-wi}(\mathcal{B}_0)$.

2.2.2 Gaps for Large Narrow Jobs in $\mathcal{B}_0$

In every platform $P_i \in \mathcal{B}_0$ we approximately guess the total load $\Pi^\star_{i,a,h}$ of jobs with height $h\delta^2$ starting at time $a\delta^2$. Note that we only need to consider those triples (i, a, h) with $h\delta^2 + a\delta^2 \leq (1 + 2\delta)$. Therefore we compute a vector $\Pi = (\Pi_{i,a,h})$ with $\Pi_{i,a,h} = b \cdot \lfloor \alpha m_N \rfloor$, $b \in \{0, 1, \ldots, \frac{2\gamma}{\alpha}\}$ and $\Pi_{i,a,h} \leq \Pi^\star_{i,a,h} \leq \Pi_{i,a,h} + \lfloor \alpha m_N \rfloor$. Here the condition $\alpha m_N - 1 \geq \alpha m_N/2$ guarantees that $\frac{2\gamma}{\alpha} \cdot \lfloor \alpha m_N \rfloor \geq \frac{2\gamma}{\alpha} \cdot (\alpha m_N - 1) \geq m_1$. There is only a constant number $(1 + \frac{2\gamma}{\alpha})^{|\mathcal{B}_0| \cdot A \cdot H}$ of different vectors Π. For every triple (i, a, h) we block a gap of $\Pi_{i,a,h} + \lfloor \alpha m_N \rfloor$ (not necessary contiguous) processors for large narrow jobs with $\bar{p}_j = h\delta^2$. Later we will place large narrow jobs with $\bar{p}_j = h\delta^2$ total width $\geq \Pi^\star_{i,a,h}$ into them. This will be done using linear programming and the subsequent rounding. Let G denote the total number of gaps, clearly $G \leq |\mathcal{B}_0| \cdot A \cdot H$.

Since $\gamma, |\mathcal{B}_0| = \mathcal{O}(N_1) = \mathcal{O}(1/\varepsilon^4)$ we have

$$\delta^{-1} \leq \mathcal{O}(\frac{1}{\varepsilon}^{|\mathcal{B}_0|/\varepsilon}) = 2^{\mathcal{O}(\varepsilon^{-5} \log(1/\varepsilon))}.$$

With $\alpha = \Theta(\delta^4)$ and $A = \mathcal{O}(\delta^{-2})$ we have

$$|\mathcal{B}_0| \cdot A \cdot \frac{2\gamma}{\alpha} \cdot \frac{1+2\delta}{\delta} = \mathcal{O}(\delta^{-7}\varepsilon^{-8}) = 2^{\mathcal{O}(\varepsilon^{-5} \log(1/\varepsilon))}.$$

Therefore, the steps described above take time $g(1/\varepsilon) \cdot n^{\mathcal{O}(f(1/\varepsilon))}$ for some function g and $f(1/\varepsilon) = 2^{\mathcal{O}(\varepsilon^{-5} \log(1/\varepsilon))}$.

2.2.3 Layers for Small Jobs

We can now compute the free layers of height δ^2 between the large wide jobs allocated by the dynamic program and the gaps designated for the large narrow jobs. Let $L_1, \dots, L_F$ denote the free layers, each having m_f processors for $f \in [F]$. In Figure 3 an allocation of the enumerated large wide jobs and a guess Π for the gaps reserved for the large narrow jobs in $\mathcal{B}_0$ is illustrated. The empty spaces of height δ^2 between and next to the large narrow and large wide jobs represent the layers for small jobs.

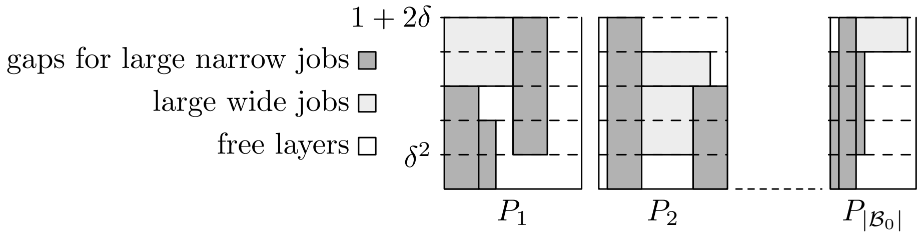

Figure 3: Simplified structure of large jobs in $\mathcal{B}_0$.

2.3 Linear Program for the Remaining Jobs $\mathcal{J}'$

In this section we give a linear programming relaxation for the following problem:

- place a set of small jobs $\mathcal{J}_{small}(\mathcal{B}_0) \subset \mathcal{J}_{small}$ into the layers $L_1, \dots, L_F$
- select large narrow jobs $\mathcal{J}_{la-na}(\mathcal{B}_0) \subset \mathcal{J}_{la-na}$ to be placed into the gaps Π,
- fractionally place the remaining jobs into $\widetilde{\mathcal{B}}_1$.

We then round the solution of the LP and obtain an approximate and almost integral solution for the problem. For integrally scheduling the large jobs in the platforms of $\widetilde{\mathcal{B}}_1$ we round their processing times in advance harmonically as described in the next section.

2.3.1 Harmonic Rounding

The harmonic transformation was first introduced by Lee and Lee [20]. For $k \in \mathbb{N}$ it is described by a function $f_k : [0,1] \longrightarrow [0,1]$ with $f_k(x) = 1/q$ for $x \in (1/(q+1), 1/q]$ for $q = 1, \dots, k-1$ and $f_k(x) = kx/(k-1)$ if $x \in (0, 1/k]$. The harmonic transformation has the following property:

Lemma 2.1. *[20] For a sequence $x_1, \dots, x_n$ with $x_i \in (0,1]$ for $i \in [n]$ and $\sum_{i=1}^{n} x_i \leq 1$ we have $\sum_{i=1}^{n} f_k(x_i) \leq T_k$, where $T_k \leq T_\infty + 1/(k-1)$ and $T_\infty \approx 1.691$ is the Harmonic constant.*

We use a slightly modified function $h_k : [0,1] \longrightarrow [0,1]$, $h_k(x) = f_k(x)$ for $x > 1/k$ and $h_k(x) = x$ for $x \leq 1/k$. According to [2] for a sequence of numbers $x_1, \ldots, x_n$ with values in $(0,1]$ and $\sum_{i=1}^{n} x_i \leq 1$ we have $\sum_{i=1}^{n} h_k(x_i) \leq T_\infty$. For $k = \lfloor \frac{20}{\varepsilon} \rfloor$ we round the processing times of the jobs in $\mathcal{J}'$ via h_k. Since $\varepsilon \leq 3/18$, we have $k = \frac{20}{\varepsilon} \geq 120$.

For each job $j \in \mathcal{J}'$ let $\widetilde{p}_j := h_k(p_j) \in (0,1]$ denote its harmonically rounded processing time. In fact, we only modify the processing times of large jobs in $\mathcal{J}'$, because the small and medium jobs have processing times $p_j \leq \delta \leq \varepsilon/20 = 1/k$. Consequently, for all small and medium jobs we have $\widetilde{p}_j = p_j$. It might also be possible that there are large jobs with processing time $1/k \geq p_j > \delta$ for which we have $p_j = \widetilde{p}_j$.

Lemma 2.2. *Assume that the choice of $\mathcal{J}_{la-wi}(\mathcal{B}_0)$ was correct and let $\mathcal{J}^\star(\mathcal{B}_1) \subset \mathcal{J}'$ be the subset of jobs scheduled in $\mathcal{B}_1$ in an optimum solution. If the processing times of the jobs in $\mathcal{J}^\star(\mathcal{B}_1)$ are rounded harmonically, an optimum schedule of the rounded jobs into $\mathcal{B}_1$ (and therefore in $\widetilde{B}_1$) has makespan at most T_∞.*

Proof. The Lemma follows from the fact that for a sequence of numbers $x_1, \ldots, x_n$ with values in $(0,1]$ and $\sum_{i=1}^{n} x_i \leq 1$ we have $\sum_{i=1}^{n} h_k(x_i) \leq T_\infty$ [2, 20] and using a result of Seiden and van Stee [26]. □

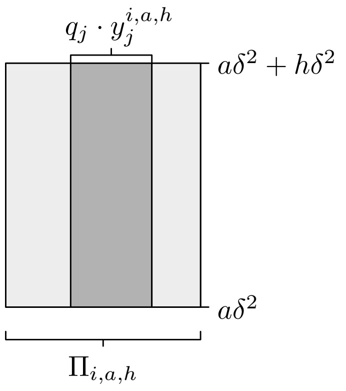

Figure 4: Slice of job j with $\bar{p}_j = h\delta^2$ in gap $\Pi_{i,a,h}$ in $P_i \in \mathcal{B}_0$.

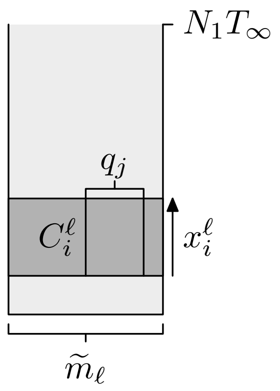

Figure 5: Job j in configuration C_i^ℓ in $\widetilde{B}_\ell \subseteq \widetilde{\mathcal{B}}_1$.

2.3.2 LP-Formulation.

For every group $\widetilde{B}_\ell$ we introduce a set $\mathcal{C}^\ell$ of feasible configurations $C^\ell : \mathcal{J}' \rightarrow \{0,1\}$ representing a subset of jobs in $\mathcal{J}'$ that fit next to each other in a platform with $\widetilde{m}_\ell$ processors, i.e. $\sum_{\{j \in \mathcal{J}' | C^\ell(j)=1\}} q_j \leq \widetilde{m}_\ell$. Let $q(\ell)$ denote the number of different configurations for $\widetilde{B}_\ell$. In the LP below the variable x_i^ℓ indicates the length of configuration C_i^ℓ for $i \in [q(\ell)]$ (compare to Figure 5). That means each job in

$\{j \in \mathcal{J}' | C^\ell(j) = 1\}$ is executed in $\widetilde{B}_\ell$ during time x_i^ℓ. In a similar way, for each layer L_f in $\mathcal{B}_0$ we introduce a set $\mathcal{C}^f$ of feasible configurations $C^f : \mathcal{J}_{small} \to \{0,1\}$ of small jobs that fit next to each other on m_f processors and denote with $q(f)$ the number of different configurations for L_f. The variable x_i^f indicates the length of configuration C_i^f for $i \in [q(f)]$. Furthermore, for every job $j \in \mathcal{J}_{la-na}$ we introduce variables $y_j^{i,a,h} \in [0,1]$, each $y_j^{i,a,h}$ indicates the vertical slice of job j (with $\bar{p}_j = h\delta^2$) that is assigned to a gap $\Pi_{i,a,h}$ in $\mathcal{B}_0$ (compare to Figure 4) .

$$
\begin{aligned}
\sum_{i=1}^{q(\ell)} x_i^\ell &\le N_1 T_\infty \quad \ell \in [L] \\
\sum_{i=1}^{q(f)} x_i^f &\le \delta^2 \quad f \in [F] \\
\sum_{\{j \in \mathcal{J}_{la-na} | \bar{p}_j = h\delta^2\}} y_j^{i,a,h} \cdot q_j &\le \Pi_{i,a,h} + \lfloor \alpha m_N \rfloor \quad i \in [|\mathcal{B}_0|], a \in [A], h \in [H] \\
\sum_{f=1}^{F} \sum_{\{i \in [q(f)] | C_i^f(j) = 1\}} x_i^f + \sum_{\ell=1}^{L} \sum_{\{i \in [q(\ell)] | C_i^\ell(j) = 1\}} x_i^\ell &\ge \widetilde{p}_j = p_j \quad j \in \mathcal{J}_{small} \\
\sum_{\ell=1}^{L} \sum_{\{i \in [q(\ell)] | C_i^\ell(j) = 1\}} x_i^\ell &\ge \widetilde{p}_j \quad j \in \mathcal{J}_{la-wi} \setminus \mathcal{J}_{la-wi}(\mathcal{B}_0) \\
\sum_{\ell=1}^{L} \sum_{\{i \in [q(\ell)] | C_i^\ell(j) = 1\}} x_i^\ell \cdot q_j + \tilde{p}_j \cdot \sum_{\substack{i,a,h \\ \bar{p}_j = h\delta^2}} y_j^{i,a,h} \cdot q_j &\ge \tilde{p}_j \cdot q_j \quad j \in \mathcal{J}_{la-na} \\
\sum_{j \in \mathcal{J}_\tau} p_j q_j - \sum_{j \in \mathcal{J}_\tau} \sum_{\ell=1}^{L} \sum_{\{i \in [q(\ell)] | C_i^\ell(j) = 1\}} x_i^\ell q_j &\le \varepsilon m_1 \\
x_i^f, x_i^\ell &\ge 0 \\
y_j^{i,a,h} &\in [0,1]
\end{aligned}
\tag{2}
$$

In the LP above we have one constraint for every group $\tilde{B}_\ell$ that guarantees that the makespan of the fractional schedule corresponding to a feasible LP-solution does not exceed length T_∞ for any $P_i \in \tilde{B}_\ell$. In a similar way we have one constraint for every layer L_f. For each gap $\Pi_{i,a,h}$ a constraint guarantees that the total load of large narrow jobs (fractionally) assigned to the gap does not exceed $\Pi_{i,a,h} + \lfloor \alpha m_N \rfloor$. For every small job we have a covering constraint combined from heights of configurations in L_f and in B_ℓ. Furthermore, we have a covering constraint for each large wide job that is not placed in $\mathcal{B}_0$, i.e. $j \in \mathcal{J}_{la-wi} \setminus \mathcal{J}_{la-wi}(\mathcal{B}_0)$. Every large narrow job $j \in \mathcal{J}_{la-na}$ is covered by an area constraint: The total width of job j assigned to $\mathcal{B}_0$ multiplied with its height $\tilde{p}_j$ in $\widetilde{\mathcal{B}}_1$ plus the area covered by configurations in $\widetilde{\mathcal{B}}_1$ should be at least $\tilde{p}_j q_j$. For the medium jobs $\mathcal{J}_\tau$ the last constraint ensures that the total area of uncovered medium jobs is small, i.e. less than εm_1. If the LP has no feasible solution either the enumerated set of large wide jobs was not correct, the choice of Π does not fit or the choice of δ, moreover the choice of τ, was not correct.

2.3.3 Solving the LP

We can compute an approximate solution of the linear program above by solving approximately a MAX-MIN RESOURCE SHARING problem.

For $n' := |\mathcal{J}'|$ and $n_\tau = |\mathcal{J}_\tau|$ the linear program (2) is a feasibility problem with an exponential number of variables and $L + F + G + n' - n_\tau + 1 \leq L + F + G + n + 1$ constraints (not counting non-negativity constraints). We can formulate the problem as a fractional covering problem with convex set K and at most $n' - n_\tau + 1$ covering constraints. The system of inequalities (defined by concave functions) is given as

$$\begin{aligned}
\frac{1}{\widetilde{p}_j}\left(\sum_{f=1}^{F}\sum_{\{i\in[q(f)]|C_i^f(j)=1\}} x_i^f + \sum_{\ell=1}^{L}\sum_{\{i\in[q(\ell)]|C_i^\ell(j)=1\}} x_i^\ell\right) &\geq 1 \;\; j\in\mathcal{J}_{small}\\
\frac{1}{\widetilde{p}_j}\sum_{\ell=1}^{L}\sum_{\{i\in[q(\ell)]|C_i^\ell(j)=1\}} x_i^\ell &\geq 1 \;\; j\in\mathcal{J}_{la-wi}\setminus\mathcal{J}_{la-wi}(\mathcal{B}_0)\\
\frac{1}{\widetilde{p}_j}\left(\sum_{\ell=1}^{L}\sum_{\{i\in[q(\ell)]|C_i^\ell(j)=1\}} x_i^\ell + \sum_{\substack{i,a,h\\ \tilde{p}_j=h\delta^2}} \tilde{p}_j y_j^{i,a,h}\right) &\geq 1 \;\; j\in\mathcal{J}_{la-na}\\
\frac{1}{\sum_{j\in\mathcal{J}_\tau}\tilde{p}_j q_j - \varepsilon m_1}\left(\sum_{j\in\mathcal{J}_\tau}\sum_{\ell=1}^{L}\sum_{\{i\in[q(\ell)]|C_i^\ell(j)=1\}} x_i^\ell q_j\right) &\geq 1
\end{aligned} \tag{3}$$

for a column vector $(x, y) \in K$ where

$$K := \left(\bigtimes_{\ell=1}^{L} K^\ell \times \bigtimes_{f=1}^{F} K^f\right) \times \bigtimes_{i,a,h} K^{i,a,h}$$

is given by the cartesian product of $L + F$ simplices

$$K^\ell := \left\{x^\ell = (x_i^\ell)_{i\in[q(\ell)]} | \sum_{i=1}^{q(\ell)} x_i^\ell = N_1 T_\infty,\; x_i^\ell \geq 0\right\},$$

$$K_f := \left\{x^f = (x_i^f)_{i\in[q(f)]} | \sum_{i=1}^{q(f)} x_i^f = \delta^2,\; x_i^f \geq 0\right\}$$

and for $\mathcal{J}^h_{la-na} := \{j \in \mathcal{J}_{la-na} | \bar{p}_j = h\delta^2\}$ we have F sets of the form

$$K^{i,a,h} := \left\{ y^{i,a,h} = (y^{i,a,h}_j)_{j \in \mathcal{J}^h_{la-na}} \mid \sum_{j=1}^{|\mathcal{J}^h_{la-na}|} y^{i,a,h}_j q_j = \Pi_{i,a,h} + \lfloor \alpha m_N \rfloor,\ y^{i,a,h}_j \in [0,1] \right\}.$$

We represent the system of inequalities (3) as $A(x,y) \geq e$ where e is the vector of all ones and A consists of the row vectors $a_1, \ldots, a_{n'-n_\tau+1}$. We can get a feasible (approximate) solution of the covering problem computing an approximate solution of the following MAX-MIN RESOURCE SHARING problem

$$\lambda^\star = \max\{\lambda | A(x,y) \geq \lambda e, (x,y) \in K\}. \tag{4}$$

According to [11] we can compute a $(1-\rho)$-approximate solution for (4) in $\mathcal{O}(n(\rho^{-2} + \log n))$ iterations: We start the iteration with an initial solution $(\bar{x}, \bar{y}) \in K$ and compute a price vector $b = (b(\bar{x}, \bar{y})) \in \mathbb{Q}^{n'-n_\tau+1}$ depending on $(\bar{x}, \bar{y})$. Then we compute a $(1-\rho')$-approximate solution $(\hat{x}, \hat{y}) \in K$ for the block problem $\max\{b^\top A(x,y) | (x,y) \in K\}$ for $\rho' = \rho/6$ and reset $(\bar{x}, \bar{y}) := (1-\tau)(\bar{x}, \bar{y}) + \tau(\hat{x}, \hat{y})$ for a certain step width $\tau \in (0,1)$. After $\mathcal{O}(n(\rho^{-2} + \log n))$ the algorithm stops at a vector $(\bar{x}, \bar{y}) \in K$ so that $A(\bar{x}, \bar{y}) \geq (1-\rho)\lambda^\star e$. For $(x,y) \in K$ we define $\lambda(x,y) := \min\{a_j \cdot (x,y) | 1 \leq j \leq n' - n_\tau + 1\}$ and proceed by case distinction.

If $\lambda(\bar{x}, \bar{y}) < 1 - \rho$ we conclude that there is no solution $(x,y) \in K$ satisfying $A(x,y) \geq e$ and so for (2). We discard the vector Π and the assignment V of large wide jobs and compute another pair (V, Π). If all possible pairs (V, Π) fail we increase the value for τ.

In case of $\lambda(\bar{x}, \bar{y}) \geq (1-\rho)$ we have $\lambda^\star \geq 1$ and can therefore compute a fractional schedule from $(\bar{x}, \bar{y})$ as follows.

We slightly extend the length of each configuration setting $(x,y) := (1 + \frac{\rho}{1-\rho})(\bar{x}, \bar{y})$ to achieve

$$a_j(x,y) = (1 + \frac{\rho}{1-\rho}) a_j(\bar{x}, \bar{y}) \geq (1 + \frac{\rho}{1-\rho})(1-\rho) \geq 1$$

for $j \in [n' - n_\tau + 1]$. Consequently, for every $\ell \in [L]$ we have $\sum_{i=1}^{q(\ell)} x^\ell_i = (1 + \frac{\rho}{1-\rho}) \sum_{i=1}^{q(\ell)} \bar{x}^\ell_i = (1 + \frac{\rho}{1-\rho}) N_1 T_\infty$. The fractional schedule in each Layer L_f and therefore in each of the platforms in group $\mathcal{B}_0$ is also increased by factor $(1 + \frac{\rho}{1-\rho})$. The same holds for the total width of large narrow jobs assigned to a gap in $\mathcal{B}_0$. So the width of a gap is also increased by the same factor . For ρ sufficient small, namely $\frac{\rho}{1-\rho} \leq \frac{\alpha}{2\gamma} \leq 1/2$, using $\Pi_{i,a,h} \leq m_1$ and $\frac{m_1 \alpha}{2\gamma} \leq \lfloor \alpha m_N \rfloor$ the increased width can be bounded by

$$(1+\frac{\alpha}{2\gamma})(\Pi_{i,a,h}+\lfloor \alpha m_N \rfloor) \leq \Pi_{i,a,h}+3\lfloor \alpha m_N \rfloor. \tag{5}$$

For $\frac{\rho}{1-\rho} = \frac{\alpha}{2\gamma}$ we have

$$\frac{\rho}{1-\rho} = \frac{\alpha}{2\gamma} \overset{\alpha=\delta^4/16, \delta \leq \varepsilon, \gamma \geq 1}{\leq} \delta^4/16 \leq \varepsilon^4.$$

and $\rho = \frac{\alpha}{2\gamma+\alpha} = \Theta(\alpha/\gamma) = \Theta(\delta^4\varepsilon^3 \log^{-1}(1/\varepsilon))$ if $N_1 = \Theta(1/\varepsilon^3 \log(1/\varepsilon))$ since $\gamma = \frac{8}{3}N_1$.

Solving the Block Problem. The block problem $\max\{b^\top A(x,y) | (x,y) \in K\}$ can be decomposed into $L+F+G$ independent smaller block problems. We introduce a weighting $w \in \mathbb{Q}^{n'}$ with

$$w_j := \begin{cases} \frac{b_j}{\tilde{p}_j} \text{ if } j \in \mathcal{J}' \setminus \mathcal{J}_\tau \\ \frac{q_j b_{n'-n_\tau+1}}{\sum_{k\in\mathcal{J}_\tau} p_k q_k - \varepsilon m_1} \text{ if } j \in \mathcal{J}_\tau. \end{cases}$$

Then for $(x,y) \in K$ we obtain after some rearrangement

$$b^\top A(x,y) =$$
$$\sum_{\ell=1}^{L} \sum_{j\in\mathcal{J}'} \sum_{\{i\in[q(\ell)] | C_i^\ell(j)=1\}} w_j x_i^\ell + \sum_{f=1}^{F} \sum_{j\in\mathcal{J}_{small}} \sum_{\{i\in[q(f)] | C_i^f(j)=1\}} w_j x_i^f + \sum_{i,a,h} \sum_{j\in\mathcal{J}_{la-na}^h} w_j \tilde{p}_j y_j^{i,a,h}.$$

So the block problem can be rewritten as

$$\begin{aligned} \max \sum_{\ell=1}^{L} \sum_{j\in\mathcal{J}'} \sum_{\{i\in[q(\ell)] | C_i^\ell(j)=1\}} & w_j x_i^\ell \\ + \sum_{f=1}^{F} \sum_{j\in\mathcal{J}_{small}} \sum_{\{i\in[q(f)] | C_i^f(j)=1\}} & w_j x_i^f \\ + \sum_{i,a,h} \sum_{j\in\mathcal{J}_{la-na}^h} & b_j y_j^{i,a,h} \\ x^\ell \in K^\ell \quad & \text{for } \ell \in [L] \\ x^f \in K^f \quad & \text{for} f \in [F] \\ y^{i,a,h} \in K^{i,a,h} \quad & \text{for all } (i,a,h). \end{aligned} \tag{6}$$

For solving (6) it is sufficient to compute a solution for L problems of the kind

$$\begin{aligned} \max \sum_{j\in\mathcal{J}'} \sum_{\{i\in[q(\ell)] \mid C_i^\ell(j)=1\}} w_j x_i^\ell \\ x^\ell \in K^\ell \end{aligned} \tag{7}$$

and F problems

$$\begin{aligned} \max \sum_{j\in\mathcal{J}_{small}} \sum_{\{i\in[q(f)] \mid C_i^f(j)=1\}} w_j x_i^f \\ x^f \in K^f \end{aligned} \tag{8}$$

and G problems of the form

$$\begin{aligned} \max \sum_{j\in\mathcal{J}^h_{la-na}} b_j y_j^{i,a,h} \\ y^{i,a,h} \in K^{i,a,h} \end{aligned} \tag{9}$$

Each of the last G problems corresponds to a fractional 0-1 knapsack problem of the following form

$$\begin{aligned} &\max \sum_j b_j x_j \\ &\text{s.t. } \sum_j q_j x_j \leq \Pi_{i,a,h} + \lfloor \alpha m_N \rfloor \\ &x_j \in [0,1] \; j \in \mathcal{J}^h_{la-na}. \end{aligned} \tag{10}$$

Which can be solved in time $\mathcal{O}(n \log n)$. As K^ℓ and K^f are simplices we find the optimum of (7) and (8) at a vertex $\tilde{x}^\ell \in K^\ell$ and $\bar{x}^f \in K^f$, respectively. For K^ℓ such a vertex corresponds to a configuration $C^\ell_{\bar{i}}$ with $\tilde{x}^\ell_{\bar{i}} = N_1 T_\infty$ and $\tilde{x}^\ell_i = 0$ for $C^\ell_i \neq C^\ell_{\bar{i}}$. That means for solving (7) we have to find a configuration C^ℓ_i with maximum weight $\sum_{\{j\in\mathcal{J}' \mid C_i^\ell(j)=1\}} w_j$. This can be formulated as a knapsack problem:

$$\begin{aligned} &\max \sum_{j\in\mathcal{J}'} w_j x_j \\ &\text{s.t. } \sum_{j\in\mathcal{J}'} q_j x_j \leq \tilde{m}_\ell \\ &x_j \in \{0,1\} \end{aligned} \tag{11}$$

Finding an optimum solution of (8) corresponds to

$$\begin{aligned} \max \sum_{j\in\mathcal{J}_{small}} & w_j x_j \\ \text{s.t.} \sum_{j\in\mathcal{J}_{small}} & q_j x_j \le m_f \\ & x_j \in \{0,1\}, \end{aligned} \tag{12}$$

where the value m_f denotes the number of processors of the free layer L^f of height δ^2 in $\mathcal{B}_0$. Using the algorithm by Lawler [19] we compute a $(1-\rho')$ approximate solution for each of the knapsack problems in time $\mathcal{O}(n\log(1/\rho') + 1/\rho'^4)$. Summing up all near optimal solutions for (7) and (8) gives a $(1-\rho')$-approximate solution for (6).
This implies that the MAX-MIN RESOURCE SHARING problem can be solved approximately with accuracy $\rho \le \frac{\alpha}{2\gamma+\alpha}$ in time $poly(n, 1/\rho)$ where $\rho = \mathcal{O}(\alpha\gamma^{-1}) = \mathcal{O}(\delta^4\gamma^{-1})$.

2.4 Rounding the LP-solution

A solution $((x^f), (x^\ell), (y_j^{i,a,h}))$ of (2) can be transformed into a fractional solution of a general assignment problem. The fractional assignment can be rounded using a result of Lenstra et al. [21] for scheduling unrelated machines similar as in Section 2.3.1. The main difficulty and difference here is to handle the large narrow narrow jobs as they are placed fractionally in $\mathcal{B}_0$ and $\widetilde{B}_0$.

For all jobs $j \in \mathcal{J}'$ we introduce new variables $x^\ell(j), x^f(j) \ge 0$ that indicate the length of the fraction of job j that is scheduled in $\widetilde{B}_\ell$, $\ell \ge 1$, and in L_f, respectively. Formally this is

$$x^\ell(j) = \sum_{\{i\in[q(\ell)] \mid C_i^\ell(j)=1\}} x_i^\ell$$

the sum of the length of all configurations in $\widetilde{B}_\ell$ in which job j appears, and

$$x^f(j) = \sum_{\{i\in[q(f)] \mid C_i^f(j)=1\}} x_i^f.$$

Additionally, for medium jobs $j \in \mathcal{J}_\tau$ (here again $\tilde{p}_j = p_j$) we define

$$x^0(j) := p_j - \sum_{\ell=1}^{L} x^\ell(j) \in [0, \tilde{p}_j],$$

the fraction of the job that should be placed in $\mathcal{B}_0 = B_0$. For the medium jobs assigned to $\mathcal{B}_0$ we therefore have the inequality

$$\varepsilon m_1 \geq \sum_{j \in \mathcal{J}_\tau} x^0(j) q_j.$$

Then the following covering constraints hold for the jobs:

$$\begin{aligned}
\sum_{\ell=1}^{L} x^\ell(j) + \sum_{f=1}^{F} x^f(j) \geq \tilde{p}_j = p_j,\ j \in \mathcal{J}_{small} \\
x^0(j) + \sum_{\ell=1}^{L} x^\ell(j) = \tilde{p}_j = p_j\ j \in \mathcal{J}_\tau \\
\sum_{\ell=1}^{L} x^\ell(j) \geq \tilde{p}_j,\ j \in \mathcal{J}_{la-wi} \setminus \mathcal{J}_{la-wi}(\mathcal{B}_0) \\
\sum_{\ell=1}^{L} x^\ell(j) q_j + \tilde{p}_j \cdot \sum_{\substack{i,a,h \\ \tilde{p}_j = h\delta^2}} y_j^{i,a,h} \cdot q_j \geq \tilde{p}_j \cdot q_j\ j \in \mathcal{J}_{la-na}
\end{aligned} \tag{13}$$

By deleting job j from appropriate configurations or replace a configuration by two "shorter" configurations (one with job j and one without, their total length is the same as the one of the original configuration) we may assume equality in each of the inequalities above. For the same reason we may also assume that $x^f(j), x^\ell(j) \in [0, \tilde{p}_j]$ now.

For all fractions $x^\ell(j)$,$x^f(j)$,$x^0(j)$ we build rectangles of width q_j and height $x^\ell(j)$, $x^f(j)$ and $x^0(j)$, respectively.

The rectangles belonging to fractions of medium jobs for B_0 we simply collect in a set

$$\mathcal{R}^0 := \{(x^0(j), q_j) | j \in \mathcal{J}_\tau\}.$$

For $\varepsilon' := \varepsilon/9$ we partition the rectangles of every group $\widetilde{B}_\ell$, $\ell \geq 1$, and Layer L_f into wide rectangles

$$\begin{aligned}
\mathcal{R}^\ell_{wide} &:= \{(x^\ell(j), q_j) | q_j > (\varepsilon'/2)\widetilde{m}_\ell\} \\
\mathcal{R}^f_{wide} &:= \{(x^f(j), q_j) | q_j > (\varepsilon'/2) m_f\}
\end{aligned}$$

and narrow rectangles

$$\begin{aligned}
\mathcal{R}^\ell_{narrow} &:= \{(x^\ell(j), q_j) | q_j \leq (\varepsilon'/2)\widetilde{m}_\ell\} \\
\mathcal{R}^f_{narrow} &:= \{(x^f(j), q_j) | q_j \leq (\varepsilon'/2) m_f\}.
\end{aligned}$$

The wide rectangles in $\mathcal{R}^\ell_{wide}$, $\mathcal{R}^f_{wide}$ are partitioned further (by width) into $M = \mathcal{O}(1/\varepsilon'^2)$ groups G^ℓ_k using geometric rounding. For each set $\mathcal{R}^\ell_{wide}$ and group $k = 1, \ldots, M$ of $\mathcal{R}^\ell_{wide}$ we introduce a variable $x^\ell_k(j) \in [0, \tilde{p}_j]$ that indicates the fraction of Job $j = 1, \ldots, n$ that is contained in this group. In a similar way we introduce variables $x_k^f(j)$. For the fraction of job j in $\mathcal{R}_{narrow}^{\ell}$ and $\mathcal{R}_{narrow}^{f}$ we introduce variables $x_0^{\ell}(j)$, $x_0^f(j) \in [0,1]$, respectively. Note that $\sum_{k=0}^{M} x_k^f(j) = x^f(j)$ and $\sum_{k=0}^{M} x_k^{\ell}(j) = x^{\ell}(j)$ holds by construction. We scale all variables $x_k^f(j)$, $x_k^{\ell}(j)$, $x^0(j)$ by $1/\tilde{p}_j$ and introduce the corresponding variables $z_k^f(j), z_k^{\ell}(j), z^0(j) \in [0,1]$. Then we can rewrite (13) (dividing every equation by its right hand side):

$$
\begin{aligned}
\sum_{\ell=1}^{L}\sum_{k=0}^{M} z_k^{\ell}(j) + \sum_{f=1}^{F}\sum_{k=0}^{M} z_k^{f}(j) &= 1,\ j \in \mathcal{J}_{small} \\
x^0(j) + \sum_{\ell=1}^{L}\sum_{k=0}^{M} z_k^{\ell}(j) &= 1\ j \in \mathcal{J}_{\tau} \\
\sum_{\ell=1}^{L}\sum_{k=0}^{M} z_k^{\ell}(j) &= 1,\ j \in \mathcal{J}_{la-wi} \setminus \mathcal{J}_{la-wi}(\mathcal{B}_0) \\
\sum_{\ell=1}^{L}\sum_{k=0}^{M} z_k^{\ell}(j) + \sum_{\substack{i,a,h \\ \tilde{p}_j = h\delta^2}} y_j^{i,a,h} &= 1\ j \in \mathcal{J}_{la-na}
\end{aligned}
\tag{14}
$$

We compute $SIZE(\mathcal{R}^0)(\leq \varepsilon m_1)$, $SIZE(\mathcal{R}_{narrow}^{\ell})$ and $SIZE(\mathcal{R}_{narrow}^{f})$ and observe that the following capacity constraints hold:

$$
\begin{aligned}
\sum_{j \in \mathcal{J}_{small}} z_0^f(j) \cdot \tilde{p}_j \cdot q_j &\leq SIZE(\mathcal{R}_{narrow}^{f})\ \ f \in [F] \\
\sum_{j \in \mathcal{J}_{small}} z_k^f(j) \cdot \tilde{p}_j &\leq H(G_k^f)\ \ k \in [M], f \in [F] \\
\sum_{j \in \mathcal{J}_{\tau}} z^0(j) \cdot \tilde{p}_j \cdot q_j &\leq SIZE(\mathcal{R}^0) \\
\sum_{j=1}^{n} z_0^{\ell}(j) \cdot \tilde{p}_j \cdot q_j &\leq SIZE(\mathcal{R}_{narrow}^{\ell})\ \ \ell \in [L] \\
\sum_{j=1}^{n} z_k^{\ell}(j) \cdot \tilde{p}_j &\leq H(G_k^{\ell})\ \ k \in [M], \ell \in [L] \\
\sum_{\{j \in \mathcal{J}_{la-na} | \tilde{p}_j = h\delta^2\}} y_j^{i,a,h} \cdot q_j &\leq \Pi_{i,a,h} + \lfloor \alpha m_N \rfloor\ \ i \in [|\mathcal{B}_0|], a \in [A], h \in [H]
\end{aligned}
\tag{15}
$$

Now we observe that $(z_k^f(j),\ z_k^{\ell}(j),\ z^0(j), y_j^{i,a,h})$ is a fractional solution of a general assignment problem formulated by (14) and (15). This assignment problem corresponds to scheduling n jobs on $|\mathcal{B}_0| \cdot A \cdot H + (F + L)(M + 1) + 1$ unrelated machines. Using a result by Lenstra et al. [21] a fractional solution of this problem can be rounded to an almost integral one with only one fractionally assigned job per machine.

Let $(\tilde{z}_k^f(j), \tilde{z}_k^\ell(j), \tilde{z}^0(j), \tilde{y}_j^{i,a,h})$ be such a rounded solution. Define

$$\begin{aligned}
\widetilde{\mathcal{R}}_{wide}^\ell &:= \{(\tilde{z}_k^\ell(j)\tilde{p}_j, q_j) | \tilde{z}_k^\ell(j) = 1, k > 0\}, \\
\widetilde{\mathcal{R}}_{narrow}^\ell &:= \{(\tilde{z}_k^\ell(j)\tilde{p}_j, q_j) | \tilde{z}_k^\ell(j) = 1, k = 0\}, \\
Frac^\ell &:= \{(\tilde{z}_k^\ell(j)\tilde{p}_j, q_j) | \tilde{z}_k^\ell(j) < 1\}, \\
\widetilde{\mathcal{R}}_{wide}^f &:= \{(\tilde{z}_k^f(j)\tilde{p}_j, q_j) | \tilde{z}_k^f(j) = 1, k > 0\}, \\
\widetilde{\mathcal{R}}_{narrow}^f &:= \{(\tilde{z}_k^\ell(j)\tilde{p}_j, q_j) | \tilde{z}_k^f(j) = 1, k = 0\}, \\
Frac^f &:= \{(\tilde{z}_k^f(j)\tilde{p}_j, q_j) | \tilde{z}_k^f(j) < 1\}, \\
\widetilde{\mathcal{R}}^0 &:= \{(z^0(j)\tilde{p}_j, q_j) | j \in \mathcal{J}_\tau, z^0(j) = 1\}, \\
frac^0 &:= (z^0(j)\tilde{p}_j, q_j) \text{ with } j \in \mathcal{J}_\tau \text{ and } z^0(j) < 1.
\end{aligned}$$

For every group $\widetilde{B}_\ell$ we obtain a set of integrally assigned wide $\widetilde{\mathcal{R}}_{wide}^\ell$ and narrow rectangles $\widetilde{\mathcal{R}}_{narrow}^\ell$ and M fractionally assigned wide rectangles plus one fractional narrow job which we collect in $Frac^\ell$. For every layer L_f we obtain in a similar way sets of integral rectangles corresponding to small jobs $\widetilde{\mathcal{R}}_{wide}^f$, $\widetilde{\mathcal{R}}_{narrow}^f$ and $M + 1$ fractional small jobs (M wide and one narrow rectangles) collected in $Frac^f$.

In addition, we have a set of integral rectangles $\widetilde{\mathcal{R}}^0$ corresponding to a set of medium jobs to be scheduled in $\mathcal{B}_0$ with total load $SIZE(\widetilde{\mathcal{R}}^0) \le SIZE(\mathcal{R}^0) \le \varepsilon m_1$ plus one fractional medium job $frac^0$ with processing time $< \delta$. We schedule the jobs in $\widetilde{\mathcal{R}}^0 \cup \{frac^0\}(= \mathcal{J}_\tau(\mathcal{B}_0))$ using list schedule on top of the largest platform P_1 in the end. This will increase the length of the schedule in P_1 by at most $2\varepsilon + \delta \le 3\varepsilon$.

2.5 Packing into the Gaps

For every gap with width $\Pi_{i,a,h}$ we have rounded variables $\tilde{y}_j^{i,a,h}$. Except for one value with $\tilde{y}_j^{i,a,h} < 1$ all of them are integral $\tilde{y}_j^{i,a,h} = 1$ and the widths of the corresponding jobs (completely including the fractional one) sum up to at most $\Pi_{i,a,h} + 2\lfloor \alpha m_N \rfloor$. Since $\Pi_{i,a,h} \le \Pi_{i,a,h}^\star \le \Pi_{i,a,h} + \lfloor \alpha m_N \rfloor$ we need to remove large narrow jobs of total width at most $3\lfloor \alpha m_N \rfloor$ for every gap that cannot be finished before $1 + 2\delta$ and schedule them on top of the solution. For all gaps their total width sums up to

$$|\mathcal{B}_0| \cdot 3\lfloor \alpha m_N \rfloor \cdot A \cdot H \le |\mathcal{B}_0| \frac{3(1+2\delta)\lfloor \alpha m_N \rfloor}{\delta^4} \le |\mathcal{B}_0| \cdot \frac{4\lfloor \alpha m_N \rfloor}{\delta^4}.$$

As those additional large narrow jobs have small width, placing some of them next to each other in a platform $P_i \in \mathcal{B}_0$, we use at least $m_i - \lfloor \alpha m_N \rfloor \ge m_N - \alpha m_N =$

$(1-\alpha)m_N$ processors of this platform. Since $\frac{|\mathcal{B}_0|\cdot 4\lfloor \alpha m_N\rfloor}{\delta^4(1-\alpha)m_N} \overset{\alpha\le 1/2}{\le} \frac{|\mathcal{B}_0|\cdot 8\alpha}{\delta^4}$ we need at most $\left\lceil \frac{|\mathcal{B}_0|\cdot 8\alpha}{\delta^4} \right\rceil$ platforms to schedule those jobs. In those platforms the schedule is increased by 1.

2.6 Packing into the Layers

The rectangles in $\widetilde{\mathcal{R}}^f_{wide}$, $\widetilde{\mathcal{R}}^f_{narrow}$ can be packed integrally into a layer L^f with the KR algorithm increasing the total height of the layer only slightly. We can show that if we then add the rectangles in $Frac^f$ greedily in the end we increase the total length of the schedule in every platform only by 3ε: According to Lemma 1.2 for Layer L^f we get an integral packing of height $\frac{(1+\varepsilon')^2}{1-\varepsilon'}\delta^2 + (4M+1)\delta^5$ where the additional $(1+\varepsilon')$-factor is due to the rounding via the general assignment problem (for details see [5]). Since $\varepsilon' = \varepsilon/9$ this is less than $(1+\varepsilon)\delta^2 + (4M+1)\delta^5$ Adding the rectangles in $Frac^f$ greedily in the end increases the length to $(1+\varepsilon)\delta^2 + (5M+2)\delta^5$. Since there can be at most $(1+2\delta)/\delta^2$ layers on top of each other the makespan in every platform is increased by at most

$$(1+2\delta)/\delta^2(\varepsilon\delta^2 + (5/\varepsilon'^2+2)\delta^5) \overset{\delta\le\varepsilon/5\le\varepsilon'}{\le} (1+2\delta)(\varepsilon+5\delta+2\delta^3) \overset{\delta\le\varepsilon/9}{\le} (1+2\delta)2\varepsilon \overset{\delta\le 1/4}{\le} 3\varepsilon.$$

So far we have scheduled a subset of the jobs into $\mathcal{B}_0$ so that almost all jobs can be finished before $1+2\delta+6\varepsilon \le 1+7\varepsilon$ except for some large narrow jobs that can be distributed within $\left\lceil \frac{|\mathcal{B}_0|\cdot 8\alpha}{\delta^4} \right\rceil$ extra platforms in $\mathcal{B}_0$. In those platforms the length of the schedule is at most $2+7\varepsilon$.

2.7 2D-Bin Packing Subroutine for $\widetilde{\mathcal{B}}_1$

We introduce here a subroutine for 2D-Bin Packing that packs the rectangles in $\widetilde{\mathcal{R}}^\ell_{wide} \cup \widetilde{\mathcal{R}}^\ell_{narrow} \cup Frac^\ell$ into $\widetilde{B}_\ell$, $\ell \in [L]$. Remember that we assume an optimum makepsan equal to 1, $p_{\max} \le 1$ and $|\widetilde{B}_\ell| = N_1$. We define similar as in [2] the following property for integral packings.

Definition 2.3. A packing of a set of rectangles with heights $\in [0,1]$ and widths at most w into a strip $b(w,\infty)$ has the *tall not sliced property for ε*, if, when drawing horizontal lines through the packing at heights $i = 1,2,3,\ldots$, no rectangle with height $> \varepsilon$ intersects with its interior such a horizontal line.

For packings having the tall not sliced property one can provide "good cutting properties":

Lemma 2.4. *Let $\mathcal{R}$ be a set of rectangles with heights bounded by 1 and widths at most w that can be integrally packed into a strip $b(w,\infty)$ with height at most h. If the packing has* *the tall not sliced property for some ε, it can be converted into a 2-dimensional bin packing using at most $(h+1)(1+\varepsilon)$ bins $b(w,1)$.*

Proof. By drawing horizontal lines at height $i = 1, \ldots, \lceil h \rceil$ through the packing we cut the packing into $\lceil h \rceil$ slices. The rectangles that lie completely with their interior between two consecutive horizontal lines can be packed into a bin $b(w,1)$. This gives at most $h+1$ bins. For every horizontal line we take out the rectangles intersecting it and get a slice of height at most ε of cut rectangles (as the packing has the tall not sliced property for ε). We can pack $\frac{1}{\varepsilon}$ of such slices together into a bin $b(w,1)$. Since we have at most $\lfloor h \rfloor$ ε-slices we get another $\varepsilon(h+1)$ bins. □

The following lemma is derived using a rounding technique in [2].

Lemma 2.5. *The rectangles in $\widetilde{\mathcal{R}}^{\ell}_{wide} \cup \widetilde{\mathcal{R}}^{\ell}_{narrow}$ can be packed integrally into a strip $b(\tilde{m}_{\ell}, \infty)$ obtaining a packing of height at most*

$$(1+\varepsilon')^2 T_{\infty} N_1 + (4M + (M+1)k)$$

having the tall not sliced property for $1/k$.

Proof. First we observe that according to the solution of the solution of (2), by construction a fractional strip packing for $\mathcal{R}^{\ell}_{wide} \cup \mathcal{R}^{\ell}_{narrow}$ into $b(\tilde{m}_{\ell}, \infty)$ has height at most $N_1 T_{\infty}$ (compare to the first L constraints of (2)). Thus, we also know that the height of an optimal fractional packing for $\mathcal{R}^{\ell}_{wide}$, denoted with $FSP(\mathcal{R}^{\ell}_{wide})$, is bounded by $N_1 T_{\infty}$.

Now we consider the rounded rectangles $\widetilde{\mathcal{R}}^{\ell}_{wide}$ and round them geometrically obtaining $\widetilde{\mathcal{R}}^{\ell}_{sup}$ with only $M = \mathcal{O}(1/\varepsilon'^2)$ (remember $\varepsilon' = \varepsilon/9$) different width. We compute an optimum solution of (1) for $\widetilde{\mathcal{R}}^{\ell}_{round}$ (modulo scaling of rectangle width by $1/\tilde{m}_{\ell}$) and construct the corresponding fractional packing with at most $2M$ non-zero configurations. As proved earlyer in [5] (using Lemma 1.2 twice) for the height of an optimal fractional packing for $\widetilde{\mathcal{R}}^{\ell}_{round}$, denoted with $FSP(\widetilde{\mathcal{R}}^{\ell}_{round})$, we have the following bound:

$$FSP(\widetilde{\mathcal{R}}^{\ell}_{round}) \overset{[5]}{\leq} (1+\varepsilon')^2 FSP(\mathcal{R}^{\ell}_{wide}) \leq (1+\varepsilon')^2 N_1 T_{\infty}$$

Thus, we obtain a fractional packing of $\widetilde{\mathcal{R}}^{\ell}_{round}$ into $b(\tilde{m}_{\ell}, \infty)$ with at most $2M$ non-zero configurations and height at most $(1+\varepsilon')^2 N_1 T_{\infty}$. Using this fractional solution for $\widetilde{\mathcal{R}}^{\ell}_{round}$ we produce an integral packing of $\widetilde{\mathcal{R}}^{\ell}_{round} \cup \widetilde{\mathcal{R}}^{\ell}_{narrow}$ in the remainder of the proof:
The rectangles in $\widetilde{\mathcal{R}}^{\ell}_{round} \cup \widetilde{\mathcal{R}}^{\ell}_{narrow} \cup Frac^{\ell}$ correspond to jobs that have harmonically rounded processing times. Thus, rectangles that correspond to jobs with processing times $> 1/k$ have heights in $\left\{ \frac{1}{q} \middle| \, q = 1, \ldots, k-1 \right\}$. Using a rounding tech-

nique by Bansal et al. [2] the fractional packing of the wide rectangles $\widetilde{\mathcal{R}}^{\ell}_{round}$ can be converted into an integral packing with height $(1+\varepsilon')^2 T_\infty N_1 + (2M + Mk)p_{\max}$ having the tall not sliced property for $1/k$: We first add an extra height of $p_{\max}(\leq 1)$ to each configuration. For each non-zero configuration we generate columns of different width according to the configuration, i.e. reserved space for the rectangles of the corresponding width. The height of each column is equal to the height of the corresponding configuration. This increases the height of the packing to $(1+\varepsilon')^2 T_\infty N_1 + 2Mp_{\max}$. For each width w_i in $\widetilde{\mathcal{R}}^{\ell}_{round}$ we order the rectangles of width w_i and height $> 1/k$ by height and fill the columns of width w_i greedily starting at height 0. Whenever the height changes we shift the (vertical) starting position of the upcoming rectangle to the next integral. So it is guaranteed that rectangles of height $1/q$ always start at integral multiples of $1/q$. Since there are M different width and k different heights for the rectangles, this may happen at most Mk times. We do not care about the starting position of rectangles with height $\leq 1/k$. In total this increases the height of the packing again by Mk.

We use NFDH to place the narrow rectangles in $\widetilde{\mathcal{R}}^{\ell}_{narrow}$ in the empty space next to the configurations in a similar way. If the height changes we open a new level with baseline at the next integral. This increases the total height again by at most $(2M + k)$ since the configuration changes at most $2M$ times and the height changes k times. The final packing has height less than

$$(1+\varepsilon')^2 T_\infty N_1 + (4M + (M+1)k).$$

□

Adding now the $M+1$ fractional rectangles in $Frac^{\ell}$ preserving the tall not sliced property for $1/k$ we get a strip packing for $\widetilde{\mathcal{R}}^{\ell}_{round} \cup \widetilde{\mathcal{R}}^{\ell}_{narrow} \cup Frac^{\ell}$ into $b(\tilde{m}_\ell, \infty)$ with height at most

$$\begin{aligned}(1+\varepsilon')^2 N_1 T_\infty + (M+1)k + 5M + 2 &\overset{k\geq 6, \varepsilon' \leq \varepsilon/9, M \geq 2}{\leq} (1+\varepsilon)N_1 T_\infty + 2Mk + Mk \\ &= (1+\varepsilon)N_1 T_\infty + 3Mk.\end{aligned}$$

For the appropriate choice of N_1 we can now convert the strip packing into a 2-dimensional bin packing using at most $2N_1$ bins $b(\tilde{m}_\ell, 1)$. Stacking any two bins on top of each other gives a packing into N_1 strips $b(\tilde{m}_\ell, \infty)$ of height 2 that corresponds to a schedule of length 2 for the platforms in $\widetilde{B}_\ell$.

Lemma 2.6. *For $N_1 = \frac{(3M(k+1)+2)k}{2k-(k+1)(1+\varepsilon)T_\infty} = \mathcal{O}(Mk^2)$ we can convert the strip packing for $\widetilde{\mathcal{R}}^{\ell}_{round} \cup \widetilde{\mathcal{R}}^{\ell}_{narrow} \cup Frac^{\ell}$ into $b(\tilde{m}_\ell, \infty)$ into a 2-dimensional bin packing using at most $2N_1$ bins $b(\tilde{m}_\ell, 1)$.*

Proof. Using the above Lemma 2.4 for $h := (1+\varepsilon)N_1 T_\infty + 3Mk$ and $1/k$ we can convert the strip packing into $b(\tilde{m}_\ell, \infty)$ into a 2-dimensional bin packing using at most $(h+1)(1+1/k)$ bins $b(\tilde{m}_\ell, 1)$. To proof our claim we show that $(h+1)(1+1/k) \leq 2N_1$.

First we show that $k > \frac{(1+\varepsilon)T_\infty}{2-(1+\varepsilon)T_\infty}$: Since $\varepsilon \leq \frac{3}{18}$ we have

$$2-(1+\varepsilon)T_\infty \geq 2-(1+\varepsilon)1.7 \overset{\varepsilon \leq 3/18}{\geq} 1/60 > 0$$

and therefore $\frac{(1+\varepsilon)T_\infty}{2-(1+\varepsilon)T_\infty} \leq \frac{1.7(1+\frac{3}{18})}{2-1.7(1+\frac{3}{18})} = 119 \leq k$ since $k = \frac{20}{\varepsilon} \geq 120$. Furthermore we have

$$\begin{aligned} N_1 &= \frac{(3M(k+1)+2)k}{2k-(k+1)(1+\varepsilon)T_\infty} \\ &= \frac{C \cdot k^2 M}{k(2-(1+\varepsilon)T_\infty) - (1+\varepsilon)T_\infty} \text{ for a constant } C > 0 \\ &\leq \frac{C \cdot k^2 M}{k/60 - (1+3/18)1.7} \\ &\overset{k \geq 120}{\leq} C \cdot 60 \cdot k^2 M. \end{aligned}$$

And finally we can prove

$$\begin{aligned} (h+1)(1+1/k) &= \frac{(k+1)[(1+\varepsilon)N_1 T_\infty + 3Mk]}{k} + \frac{k+1}{k} \\ &\leq \frac{(k+1)(1+\varepsilon)N_1 T_\infty}{k} + \frac{k(3M(k+1)+2)}{k} \\ &\leq \frac{(k+1)(1+\varepsilon)N_1 T_\infty}{k} + \frac{N_1(2k-(k+1)(1+\varepsilon)T_\infty)}{k} \\ &\leq 2N_1. \end{aligned}$$

□

2.8 Converting Process

So far we constructed a schedule for the rounded platforms $\mathcal{B}_0 \cup \widetilde{\mathcal{B}}_1$. It remains to convert the schedule into one for $\mathcal{B}_0 \cup \mathcal{B}_1$.

Lemma 2.7. *The schedule can be converted into a schedule of length* $2 + \mathcal{O}(\varepsilon)$ *for* $\mathcal{B}_1 \cup \mathcal{B}_0$.

Proof. Remember from Section 2.6 that for $\mathcal{B}_0$ the schedule produced so far has length $\leq 1 + 7\varepsilon$ except for $\left\lceil \frac{|\mathcal{B}_0| \cdot 8\alpha}{\delta^4} \right\rceil$ platforms in which the schedule has length $\leq 2 + 7\varepsilon$. If it is possible to distribute the jobs scheduled in group $\widetilde{B}_1 \subseteq \widetilde{\mathcal{B}}_1$ among the platforms in $\mathcal{B}_0$ we can apply a shifting argument (see Figure 6) and obtain a schedule for $\mathcal{B}_0 \cup \mathcal{B}_1$. Recall that the schedule in every platform in $\widetilde{B}_1$ is composed

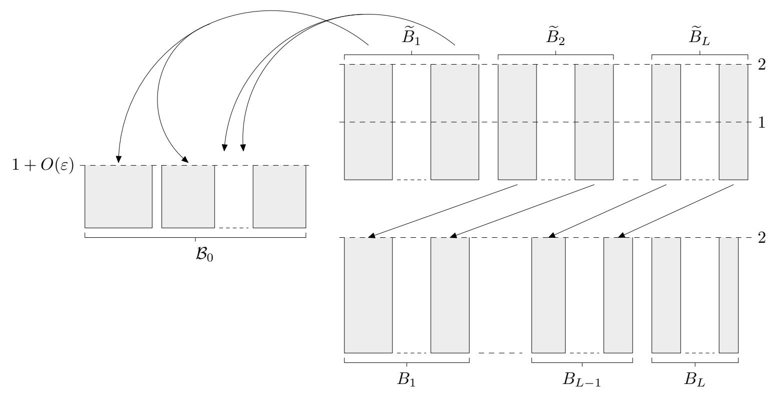

Figure 6: Shifting technique.

by stacking at most two 2-dimensional bins $b(\widetilde{m}_1, 1)$ (where $\widetilde{m}_1 \leq m_{|\mathcal{B}_0|}$) on top of each other. Thus, we need to distribute $2N_1$ bins $b(\widetilde{m}_1, 1)$ among the platforms in $\mathcal{B}_0$.

In total, if $|\mathcal{B}_0|$ satisfies the inequality

$$|\mathcal{B}_0| \geq 2N_1 + \left\lceil \frac{|\mathcal{B}_0| \cdot 8\alpha}{\delta^4} \right\rceil \tag{16}$$

we can convert the schedule. Since $|\mathcal{B}_0| \geq N_0$ equation (16) holds, if

$$|\mathcal{B}_0|(1 - \frac{8\alpha}{\delta^4}) \geq 2N_1 + 1 \tag{17}$$

Since $\alpha = \delta^4/16$ this is fulfilled for $|\mathcal{B}_0| \geq N_0 = 2(2N_1 + 1)$.

□

3 Case 2: Using the Gap γ

We may now assume that there is a number $K \in [N]$, so that $m_1/m_K \leq \gamma$ and $m_1/m_{K+1} > \gamma$, where $\gamma = \frac{8}{3}N_1$.

If $K \geq N_0 = \mathcal{O}(1/\varepsilon^4)$ a variant of the algorithm for the first scenario can be applied achieving a $(2 + \mathcal{O}(\varepsilon))$-approximation. In this variant we partition the platforms in the same way as in Case 1 and consider jobs as wide if they satisfy $q_j \geq \lfloor \alpha m_{|\mathcal{B}_0|} \rfloor$ where $\alpha = \delta^4/16$ as before. The rest of the algorithm can be directly applied. Thus, throughout this section we assume $K < N_0$.

3.1 Structural Simplifications

We define $\mathcal{B}_0 := \{P_1, \dots, P_K\}$ and $\mathcal{B}_1 := \{P_{K+1}, \dots, P_N\}$. For $N_1 = \mathcal{O}(1/\varepsilon^4)$ as in Lemma 2.6 we partition $\mathcal{B}_1$ into $L := \left\lceil \frac{N-K}{N_1} \right\rceil$ groups

$$B_\ell = \{P_{K+(\ell-1)N_1+1}, \dots, P_{K+\ell N_1}\} \text{ for } \ell \in [L-1]$$

containing exactly N_1 platforms and $B_L := \{P_{K+(L-1)N_1+1,\dots,P_N}\}$ containing maybe less that N_1 platforms. In each group B_ℓ, $\ell \in [L]$ we round the number of processors of each platform up to the number of processors $\widetilde{m}_\ell := m_{K+(\ell-1)N_1+1}$ of the largest platform $P_{K+(\ell-1)N_1+1}$ contained in this group. In group B_L we add $N_1 - |B_L|$ dummy platforms with $\widetilde{m}_L$ processors, so that every modified group, denoted with $\widetilde{B}_\ell$, $\ell \in [L]$, contains exactly N_1 platforms of the same kind.

We first compute a schedule for $\mathcal{B}_0 \cup \widetilde{\mathcal{B}}_1$, where $\widetilde{\mathcal{B}}_1 = \bigcup_\ell \widetilde{B}_\ell$, and convert this solution into a solution for $\mathcal{B}_0 \cup \mathcal{B}_1$. With a similar argument as we assumed $m_N \geq 32/\delta^4$ in the first case we may assume here that $m_K \geq 32/\delta^4$. (Otherwise the number of processors in platform P_1 and therefore in every platform is bounded by a constant. This implies that also the number of jobs that fit next to each other is bounded by a constant and we do not distinguish between wide and narrow jobs to enumerate them.) We choose $\alpha = \delta^4/16$. Then we have $\alpha m_K \geq 2$ implying $\alpha m_K - 1 \geq \alpha m_K/2$ We call a job (p_j, q_j) *wide* if $q_j \geq \lfloor \alpha m_K \rfloor$ and *narrow* otherwise.

3.2 Algorithm for Case 2

As in the first case we enumerate and assignment of large wide for $\mathcal{B}_0$ and approximately guess the vector $\Pi^\star = (\Pi^\star_{i,a,h})$ of loads of the large narrow jobs in $\mathcal{B}_0$. We compute the free layers of height δ^2 in $\mathcal{B}_0$ and use the techniques as described in Sections 2.3-2.7.

It remains now to convert the schedule for $\mathcal{B}_0 \cup \widetilde{\mathcal{B}}_1$ into a schedule for $\mathcal{B}_0 \cup \mathcal{B}_1$. As in Case 1 we need to distribute the jobs scheduled in $\widetilde{B}_1$ and some extra large narrow jobs among the platforms in $\mathcal{B}_0$.

3.3 Converting Process and Choice of γ

In the worst case $K = 1$ and we have to place the additional $2N_1$ 2D-bins $b(\tilde{m}_1, 1)$ of width at most $m_{K+1} \geq \tilde{m}_1$ plus additional large narrow jobs of total width

$$K \cdot \frac{4\lfloor \alpha m_K \rfloor}{\delta^4} = \frac{4\lfloor \alpha m_1 \rfloor}{\delta^4}$$

next to each other on P_1. Thus, in the worst case the number of processors at least needed in P_1 can be bounded by

$$2N_1 \cdot m_{k+1} + \frac{4\lfloor \alpha m_1 \rfloor}{\delta^4} \overset{\alpha=\delta^4/16}{\leq} 2N_1 \cdot m_{K+1} + \frac{m_1}{4} \tag{18}$$

If we choose $\gamma = \frac{8}{3}N_1$ we have $m_1 > 2N_1 \cdot m_{K+1} + \frac{m_1}{4}$.

Finally, we obtained a $(2 + \mathcal{O}(\varepsilon))$-approximation in both cases and proved Theorem 1.1.

4 Conclusion

We have obtained an Algorithm that constructs a schedule of a set $\mathcal{J}$ of n parallel jobs into a set $\mathcal{B}$ of N heterogeneous platforms with makespan at most $(2 + \varepsilon)\mathrm{OPT}(\mathcal{J}, \mathcal{B})$. We assume that it is also possible to find an algorithm that packs a set of n rectangles into N strips of different widths. Many of the techniques used also apply to rectangles. The main difficulties will be the selection and packing process of the large narrow rectangles for $\mathcal{B}_0$ as the gaps provided by our algorithm might contain non-contiguous processors.

Furthermore, we can decrease the running time of the algorithm using a different rounding technique: Instead of geometric rounding we can apply the same rounding technique as in our improved version of the KR algorithm in [5]. Using this we obtain $M = \mathcal{O}(1/\varepsilon \log(1/\varepsilon))$ and therefore $N_1, N_0 = \mathcal{O}(1/\varepsilon^3 \log(1/\varepsilon))$. This improves the running time since in this case we have $f(1/\varepsilon) = 2^{\mathcal{O}(\varepsilon^{-4} \log^2(1/\varepsilon))}$.

References

[1] A. K. Amoura, E. Bampis, C. Kenyon, and Y. Manoussakis. Scheduling independent multiprocessor tasks. *Algorithmica*, 32(2):247–261, 2002.

[2] N. Bansal, X. Han, K. Iwama, M. Sviridenko, and G. Zhang. Harmonic algorithm for 3-dimensional strip packing problem. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms (SODA 2007)*, pages 1197–1206, 2007.

[3] M. Bougeret, P. F. Dutot, K. Jansen, C. Otte, and D. Trystram. Approximating the non-contiguous multiple organization packing problem. In *Proceedings of the 6th IFIP International Conference on Theoretical Computer Science (TCS 2010)*, pages 316–327, 2010.

[4] M. Bougeret, P.-F. Dutot, K. Jansen, C. Otte, and D. Trystram. A fast 5/2-approximation algorithm for hierarchical scheduling. In *Proceedings of the*

16th International Euro-Par Conference- Parallel Processing Part I (Euro-Par 2010), LNCS 6272, pages 157–167, 2010.

[5] M. Bougeret, P.-F. Dutot, K. Jansen, C. Robenek, and D. Trystram. Approximation algorithms for multiple strip packing and scheduling parallel jobs in platforms. *Discrete Mathematics, Algorithms and Applications*, 3(4):553–586, 2011.

[6] M. Bougeret, P.-F. Dutot, K. Jansen, C. Robenek, and D. Trystram. Tight approximation for scheduling parallel jobs on identical clusters. In *26th IEEE International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPS Workshops 2012)*, pages 878–885, 2012.

[7] E. G. Coffman Jr., M. R. Garey, D. S. Johnson, and R. E. Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9(4):808–826, 1980.

[8] J. Du and J. Y.-T. Leung. Complexity of scheduling parallel task systems. *SIAM Journal of Discrete Mathematics*, 2(4):473–487, 1989.

[9] A. Feldmann, J. Sgall, and S.-H. Teng. Dynamic scheduling on parallel machines. *Theoretical Computer Science*, 130(1):49–72, 1994.

[10] M. R. Garey and R. L. Graham. Bounds for multiprocessor scheduling with resource constraints. *SIAM Journal on Computing*, 4(2):187–200, 1975.

[11] M. D. Grigoriadis, L. G. Khachiyan, L. Porkolab, and J. Villavicencio. Approximate max-min resource sharing for structured concave optimization. *SIAM Journal on Optimization*, 11(4):1081–1091, 2001.

[12] R. Harren, K. Jansen, L. Prädel, and R. van Stee. A $(5/3+\epsilon)$-approximation for strip packing. In *Proceedings of the 12th International Symposium on Algorithms and Data Structures (WADS 2011)*, pages 475–487, 2011.

[13] K. Jansen. Scheduling malleable parallel tasks: An asymptotic fully polynomial time approximation scheme. *Algorithmica*, 39(1):59–81, 2004.

[14] K. Jansen. A $(3/2+\epsilon)$-approximation algorithm for scheduling moldable and non-moldable parallel tasks. In *24th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2012)*, pages 224–235, 2012.

[15] K. Jansen and L. Porkolab. Linear-time approximation schemes for scheduling malleable parallel tasks. *Algorithmica*, 32(3):507–520, 2002.

[16] K. Jansen and R. Solis-Oba. Rectangle packing with one-dimensional resource augmentation. *Discrete Optimization*, 6(3):310–323, 2009.

[17] K. Jansen and R. Thöle. Approximation algorithms for scheduling parallel jobs. *SIAM Journal on Computing*, 39(8):3571–3615, 2010.

[18] C. Kenyon and E. Rémila. A near-optimal solution to a two-dimensional cutting stock problem. *Mathematics of Operations Research*, 25(4):645–656, 2000.

[19] E. L. Lawler. Fast approximation algorithms for knapsack problems. *Mathematics of Operations Research*, 4(4):339–356, 1979.

[20] C. C. Lee and D. T. Lee. A simple on-line bin-packing algorithm. *Journal of the ACM*, 32(3):562–572, 1985.

[21] J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990.

[22] G. Mounie, C. Rapine, and D. Trystram. A 3/2-approximation algorithm for scheduling independent monotonic malleable tasks. *SIAM Journal on Computing*, 37(2):401–412, 2007.

[23] J. Remy. Resource constrained scheduling on multiple machines. *Information Processing Letters*, 91(4):177–182, 2004.

[24] I. Schiermeyer. Reverse-fit: A 2-optimal algorithm for packing rectangles. In *Proceedings of the 2nd European Symposium on Algorithms (ESA 1994), LNCS 855*, pages 290–299, 1994.

[25] U. Schwiegelshohn, A. Tchernykh, and R. Yahyapour. Online scheduling in grids. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS 2008)*, pages 1–10, 2008.

[26] S. S. Seiden and R. van Stee. New bounds for multidimensional packing. *Algorithmica*, 36(3):261–293, 2003.

[27] A. Steinberg. A strip-packing algorithm with absolute performance bound 2. *SIAM Journal on Computing*, 26(2):401–409, 1997.

[28] A. Tchernykh, J. Ramírez, A. Avetisyan, N. Kuzjurin, D. Grushin, and S. Zhuk. Two level job-scheduling strategies for a computational grid. In *Proceedings of the 6th International Conference on Parallel Processing and Applied Mathematics (PPAM 2005), LNCS 3911*, pages 774–781, 2005.

[29] D. Ye, X. Han, and G. Zhang. Online multiple-strip packing. *Theoretical Computer Science*, 412(3):233–239, 2011.