A $(2+\varepsilon)$-Approximation for Scheduling Parallel Jobs in Platforms*

Pierre-François Dutot[1], Klaus Jansen[2], Christina Robenek[2], and Denis Trystram[1]

[1] LIG, Grenoble Institute of Technology INPG
51 rue Jean Kuntzmann F-38330 Montbonnot St. Martin, France
{pfdutot,trystram}@imag.fr
[2] Department of Computer Science, University of Kiel
Christian-Albrechts-Platz 4, 24098 Kiel, Germany
{kj,cot}@informatik.uni-kiel.de

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 required $q_j \in \mathbb{N}$. 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. 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 approximation ratio.

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 $\max_{i \in [N]} C_{\max}^{(i)}$, where $C_{\max}^{(i)} = \max_{\{j | A_j \subset M_i\}} t_j + p_j$ denotes the local makespan for platform P_i. We denote with $\mathrm{OPT}_{\mathrm{SPP}}(\mathcal{J}, \mathcal{B})$

* Research supported by German Research Foundation (DFG) project JA612/12-2.

F. Wolf, B. Mohr, and D. an Mey (Eds.): Euro-Par 2013, LNCS 8097, pp. 78–89, 2013.

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 $m \geq 5$ [7]. 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 $A_j \in \bigcup_{\ell=1}^{N} 2^{M_\ell}$ 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. 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.

If the number of processors is bounded by a constant, the problem admits a PTAS [1]. In case that the number of machines is polynomially bounded in the number of jobs, a $(1.5+\varepsilon)$-approximation for the contiguous problem (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 [12]. Recently, for an arbitrary number of processors a tight approximation algorithm with absolute ratio $1.5+\varepsilon$ was achieved [10].

Also for an arbitrary number of processors the contiguous case of $P|size_j|C_{\max}$ is closely related to Strip Packing. A vast number of approximation algorithms for Strip Packing have been developed during the last decades.

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 [13]. 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 [13] as a subroutine and refer to it as the KR algorithm.

Multiple Strip Packing was first considered by Zhuk [23], who also showed that there is no approximation algorithm with absolute approximation ratio

better than 2. Meanwhile, several approximation algorithms for Multiple Strip Packing and Scheduling parallel Jobs in Platforms have been developed. Some of them can be applied to both problems achieving the same approximation ratio. However, due to different underlying techniques used for designing those algorithms, some of them are only suitable for one of the problems or require even more constraints on the jobs (rectangles) and platforms. In Table 1 we give an overview about the different kinds of algorithms and their approximation ratios.

Table 1. Known approximation algorithms and their ratios

		Platforms	Jobs	Rect.	Ratio	Constraints
Tchernykh et al.	[20] 2005	het.	✓	✓	10	none
Schwiegelshohn et al	[18] 2008	het.	✓	no	3	non-clairvoyant
Ye et al.	[22] 2009	ident.	✓	✓	2ρ	ρ makespan of $P\|\|C_{\max}$
Pascual et al.	[16] 2009	ident.	✓	no	4	none
Tchernykh et al.	[21] 2010	het.	✓	✓	$2e+1$	release dates
Quezada-Pina et al.	[17] 2012	het.	✓	no	3	$q_j \leq \min_i m_i$
Bougeret et al.	[5] 2009	ident.	no	✓	2	none
	[3] 2010	ident.	✓	no	2.5	none
	[4] 2010	het.	✓	no	2.5	$q_j \leq \min_i m_i$
	[5] 2011	het.	✓	✓	AFPTAS	none
	[6] 2012	ident.	✓	no	2	$\max_j q_j \leq m/2$

1.2 New Result

Currently, the best known absolute ratio for an approximation algorithm directly applicable to SPP is 3 given in [18]. Remark that in [18], processing times are not available to the scheduler. In this article we present a polynomial time algorithm with absolute ratio $(2+\varepsilon)$. Moreover, we nearly close the gap between the inapproximability bound of 2 and the currently best absolute ratio.

Theorem 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 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 1.

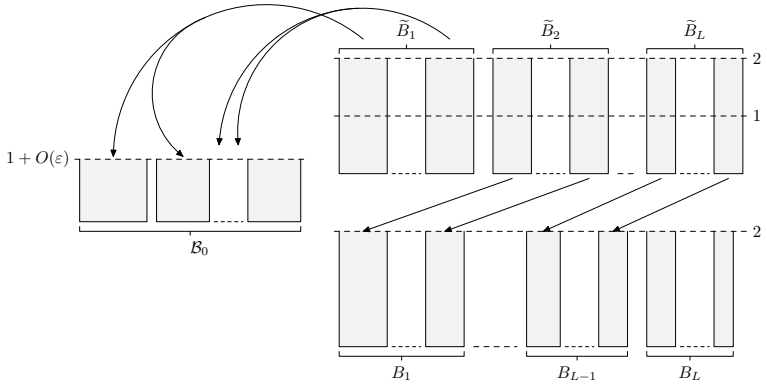

Fig. 1. Converting the schedule

Using gap creation [11] we simplify the structure of an optimum solution in $\mathcal{B}_0$, see Section 2.2 and Figure 2. 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 2.6) 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 [15] to round those fractions to integral jobs producing only a small error even though there are different kinds of fractions. The Linear Program in 2.6 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 2.5. 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 a multiple of $1/q$, see [8]. 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. An overview about the algorithm is given in Algorithm 1.

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 job 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 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 widths 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 rectangle 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 $\min\{\sum_i x_i | \sum_{i:C_i(r)=1} x_i \geq h_r \quad r \in \mathcal{R}, x_i \geq 0\}$ 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. $\sum_{\{r\in\mathcal{R}|C_i(r)=1\}} w_r \leq w$. 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 [13].

Lemma 1. *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 $\mathcal{R}_{narrow} := \mathcal{R} \backslash \mathcal{R}_{wide}$. 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 $\frac{(1+\varepsilon')h}{1-\varepsilon'} + (4M + 1) \max_{r\in\mathcal{R}} h_r$.*

2 Algorithm

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

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

In this section we give a detailed description of the algorithm for the first scenario from which the algorithm for the second scenario is derived. Details for the second scenario can be found in our technical report [8].

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 $5N_1+2 = N_0+N_1 > |\mathcal{B}_0| \geq N_0$. 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. We compute a schedule for $\mathcal{B}_0 \cup \widetilde{\mathcal{B}}_1$, where $\widetilde{\mathcal{B}}_1 = \bigcup_\ell \widetilde{B}_\ell$, and later convert this solution into a solution for $\mathcal{B}_0 \cup \mathcal{B}_1$ applying a shifting argument, see Figure 1.

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

Via binary search in the interval $\left[SIZE(\mathcal{J})/(\sum_{i=1}^{N} m_i), np_{\max}\right]$ we find a candidate T for the makespan. By scaling we may assume $T = 1$. Now consider an optimum solution with makespan 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. In the following we use the gap creation technique [11] to 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 only slightly increasing the makespan. Define $\sigma_0 = \frac{\varepsilon}{20}$ 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 $\mathcal{J}^\star_k(\mathcal{B}_0)$ and $\mathcal{J}^\star_k(\mathcal{B}_1)$ denote those jobs in $\mathcal{J}_k$ that are scheduled by an optimum algorithm in $\mathcal{B}_0$ and $\mathcal{B}_1$, respectively. 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 $\tau \in \{1, \ldots, \frac{|\mathcal{B}_0|}{\varepsilon}\}$ 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 an integer and thus $\delta^{-1} = \left(\frac{\varepsilon}{20}\right)^{-\left(5^{\tau-1}\right)} = \left(\frac{20}{\varepsilon}\right)^{5^{\tau-1}}$ is an integer (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 $\mathcal{J}_{medium} := \mathcal{J}_\tau = \{j \in \mathcal{J} | p_j \in (\delta^5, \delta]\}$ and large jobs with $\mathcal{J}_{large} := \{j \in \mathcal{J} | p_j \in (\delta, 1]\}$.

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

Input: $\mathcal{J}$, $\varepsilon > 0$

Output: A schedule of length $(2+\varepsilon)\mathrm{OPT}_{\mathrm{SPP}}(\mathcal{J})$

1: For a certain constant $N_1 = \mathcal{O}(1/\varepsilon^4)$ 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_ℓ 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 δ distinguish small, medium, and large jobs
7: Round the processing times and possible starting times of large jobs to integral multiples δ^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 Π of large narrow jobs for each starting time and height in every platform of $\mathcal{B}_0$ and block corresponding gaps.
12: **for** a guess Π **do**
13: Compute free layers of height δ^2 in $\mathcal{B}_0$.
14: Round the processing times p_j of the jobs $\mathcal{J}' = \mathcal{J} \setminus \mathcal{J}_{la-wi}(\mathcal{B}_0)$ harmonically.
15: Compute a solution of the LP in 2.6
16: **if** There is no feasible solution **then**
17: Discard the guess Π 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, Π) have failed, increase k and go to Step 5.
18: **end if**
19: Round the solution of the LP using a result of Lenstra et al. [15] and obtain 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 Π 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: Schedule medium jobs in $\mathcal{J}_\tau(\mathcal{B}_0)$ in P_1.
22: **for** $\ell = 1, \ldots, L$ **do**
23: Pack the jobs assigned to $\widetilde{B}_\ell$ into at most $2N_1$ bins $b(\tilde{m}_\ell, 1)$
24: **end for**
25: **end for**
26: **end for**
27: **end for**
28: **end for**
29: Convert the schedule for $\mathcal{B}_0 \cup \widetilde{\mathcal{B}}_1$ into a schedule for $\mathcal{B}_0 \cup \mathcal{B}_1$

Scheduling the medium jobs in $\mathcal{J}_\tau^\star(\mathcal{B}_0)$ in the end on top of the largest platform P_1 using List Schedule [9] 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 integer 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 $\frac{1}{\delta^2} - (\frac{1}{\delta}+1) + 1 \leq \frac{1}{\delta^2}$. In a similar way we round the starting time of each large job in $\mathcal{B}_0$ to $a\delta^2$. 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 $\mathcal{J}^\star(\mathcal{B}_0)\setminus\mathcal{J}_\tau^\star(\mathcal{B}_0)$ in $\mathcal{B}_0$ with rounded processing times $\bar{p}_j$ and rounded starting times for the large jobs has length at most $1+2\delta$.

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 as follows. Assume that $m_N \geq 32/\delta^4$, otherwise the number of different platform sizes is a constant depending on γ and $1/\delta$. 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 $\mathcal{J}_{la-na} := \{j \in \mathcal{J}_{large} | q_j \leq \lfloor \alpha m_N \rfloor\}$ and large wide jobs $\mathcal{J}_{la-wi} := \{j \in \mathcal{J}_{large} | q_j > \lfloor \alpha m_N \rfloor\}$.

2.3 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 every $P_i \in \mathcal{B}_0$ is bounded by $\frac{2\gamma}{\alpha} \cdot \frac{1+2\delta}{\delta}$. Furthermore, in every platform 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 $\mathcal{J}' := \mathcal{J}\setminus\mathcal{J}_{la-wi}(\mathcal{B}_0)$.

2.4 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 subsequent rounding. Let G denote the total number of gaps, clearly $G \leq |\mathcal{B}_0| \cdot A \cdot H$. Since $\gamma, |\mathcal{B}_0| = \mathcal{O}(N_1) = \mathcal{O}(1/\varepsilon^3 \log(1/\varepsilon))$ and $\delta^{-1} = 2^{\mathcal{O}(1/\varepsilon \log(1/\varepsilon))}$, the steps described above take time $g(1/\varepsilon) \cdot n^{\mathcal{O}(f(1/\varepsilon))}$ for some function g and $f(1/\varepsilon) = 2^{\mathcal{O}(1/\varepsilon \log(1/\varepsilon))}$.

In Figure 2 an allocation of the enumerated large wide jobs and a guess Π for the gaps reserved for the large narrow jobs in $\mathcal{B}_0$ are illustrated. We compute the free layers of height δ^2 that correspond to the empty space between and next to the gaps and the large wide jobs. Let $L_1, \ldots, L_F$ denote the free layers, each having m_f processors for $f \in [F]$.

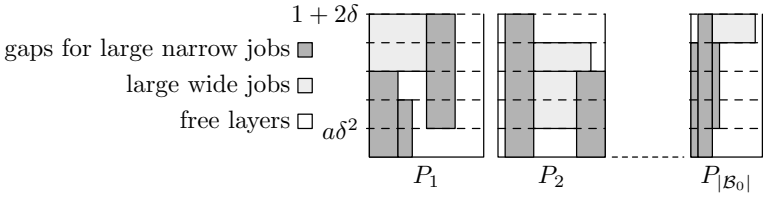

Fig. 2. Simplified structure of large jobs in $\mathcal{B}_0$

2.5 Rounding Jobs in $\widetilde{\mathcal{B}}_1$

Let $\mathcal{J}^{\star}(\mathcal{B}_1) \subset \mathcal{J}'$ be the subset of jobs scheduled in $\mathcal{B}_1$ in an optimum solution. Let $k := \frac{20}{\varepsilon}$. We assign to every job $\mathcal{J}^{\star}(\mathcal{B}_1)$ its harmonically rounded processing time $\tilde{p}_j := h_k(p_j) \in [0, 1]$, where $h_k : [0, 1] \longrightarrow [0, 1]$ is defined as in [2] via $h_k(x) = 1/q$ for $x \in (1/(q+1), 1/q], q = 1, \ldots, k-1$ and $h_k(x) = x$ for $x \leq 1/k$. Since $\varepsilon \leq 1/6$, we have $k = \frac{20}{\varepsilon} \geq 120$. In fact, we only modify the processing times of large jobs in $\mathcal{J}^{\star}(\mathcal{B}_1)$, 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 $p_j = \widetilde{p}_j$. The following Lemma can be derived 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 $\sum_{i=1}^{n} h_k(x_i) \leq T_\infty$ [2,14] together with a result of Seiden and van Stee [19] (where T_∞ is a constant $\simeq 1.691$, see [2]).

Lemma 2. *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_∞.*

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

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 $L_1, \ldots, L_F$
- 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 think of every group $\widetilde{B}_\ell$ as a single strip $b(\widetilde{m}_\ell, \infty)$ and introduce a set $\mathcal{C}^\ell$ of feasible configurations $C^\ell : \mathcal{J}' \to \{0,1\}$. 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 $i \in [q(\ell)]$. In a similar way, we think of each layer L_f in $\mathcal{B}_0$ as a strip $b(\widetilde{m}_f, \infty)$ and introduce a set $\mathcal{C}^f$ of feasible configurations $C^f : \mathcal{J}_{small} \to \{0,1\}$ of small jobs 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 $i \in [q(f)]$. For every job $j \in \mathcal{J}_{la-na}$ we introduce variables $y_j^{i,a,h} \in [0,1]$, that indicate the vertical fraction of job j (with $\bar{p}_j = h\delta^2$) that is assigned to a gap $\Pi_{i,a,h}$ in $\mathcal{B}_0$. For every group $\tilde{B}_\ell$ we need a constraint that guarantees that the length of the fractional schedule in $b(\widetilde{m}_\ell, \infty)$ corresponding to a feasible LP-solution does not exceed length NT_∞. Since we have N platforms in $\tilde{B}_\ell$ this gives a fractional schedule of length T_∞ in every platform. 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 $\Pi_{i,a,h} + \lfloor \alpha m_N \rfloor$. To guarantee that all jobs are scheduled we have covering constraints.

For every small job we have a covering constraint combined from heights of configurations in L_f and in $\widetilde{B}_\ell$. 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 a clever area constraint: The total fractional width of job j assigned to $\mathcal{B}_0$ multiplied with its height $\tilde{p}_j$ in $\widetilde{\mathcal{B}}_1$ plus the fraction of the area of this job 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. Finally we add $x_i^f, x_i^\ell \geq 0$ and $y_j^{i,a,h} \in [0,1]$.

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

If the LP has no feasible solution either the enumerated set $\mathcal{J}_{la-wi}(\mathcal{B}_0)$ was not correct, the choice of Π does not fit or the choice of δ, moreover the choice of τ, was not correct. We can compute an approximate solution of the linear program above by solving approximately a Max-Min Resource Sharing problem.

A solution $((x^f),(x^\ell),(y_j^{i,a,h}))$ of the LP can be transformed into a fractional solution of a general assignment problem. This assignment problem corresponds to scheduling n jobs on $|\mathcal{B}_0| \cdot A \cdot H + (F+L)(M+1)+1$ unrelated machines, for $M = 1/\varepsilon'^2$, $\varepsilon' = \varepsilon/(4+\varepsilon)$. Using a result by Lenstra et al. [15] a fractional solution of this problem can be rounded to an almost integral one with only one fractionally assigned job per machine.

Different job manipulations are then described in [8] to assign those fractionally assigned jobs integrally to parts of $\mathcal{B}_0$ or in some gaps, without increasing the makespan. Then using a rounding technique for strip packing with harmonically rounded rectangles presented in [2], we show in [8] using $N_1 = \frac{(3M(k+1)+2)k}{2k-(k+1)(1+\varepsilon)T_\infty}$, how to produce a schedule with makespan $2+\mathcal{O}(\varepsilon)$ of all jobs in $\mathcal{J}$ in the platforms of $\mathcal{B}_0 \cup \widetilde{\mathcal{B}_1}$. The schedule is finally converted into one for $\mathcal{B}_0 \cup \mathcal{B}_1$ with a shifting procedure illustrated in Figure 1.

3 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.

References

1. Amoura, A.K., Bampis, E., Kenyon, C., Manoussakis, Y.: Scheduling independent multiprocessor tasks. Algorithmica 32(2), 247–261 (2002)
2. Bansal, N., Han, X., Iwama, K., Sviridenko, M., Zhang, G.: Harmonic algorithm for 3-dimensional strip packing problem. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2007), pp. 1197–1206 (2007)
3. Bougeret, M., Dutot, P.F., Jansen, K., Otte, C., Trystram, D.: Approximating the non-contiguous multiple organization packing problem. In: Calude, C.S., Sassone, V. (eds.) TCS 2010. IFIP AICT, vol. 323, pp. 316–327. Springer, Heidelberg (2010)
4. Bougeret, M., Dutot, P.-F., Jansen, K., Otte, C., Trystram, D.: A fast 5/2-approximation algorithm for hierarchical scheduling. In: D'Ambra, P., Guarracino, M., Talia, D. (eds.) Euro-Par 2010, Part I. LNCS, vol. 6271, pp. 157–167. Springer, Heidelberg (2010)
5. Bougeret, M., Dutot, P.-F., Jansen, K., Robenek, C., Trystram, D.: Approximation algorithms for multiple strip packing and scheduling parallel jobs in platforms. Discrete Mathematics, Algorithms and Applications 3(4), 553–586 (2011)

6. Bougeret, M., Dutot, P.-F., Jansen, K., Robenek, C., Trystram, D.: Tight approximation for scheduling parallel jobs on identical clusters. In: IEEE International Parallel and Distributed Processing Symposium Workshops, pp. 878–885 (2012)
7. Du, J., Leung, J.Y.-T.: Complexity of scheduling parallel task systems. SIAM Journal of Discrete Mathematics 2(4), 473–487 (1989)
8. Dutot, P.-F., Jansen, K., Robenek, C., Trystram, D.: A $(2+\varepsilon)$-approximation for scheduling parallel jobs in platforms. Department of Computer Science, University Kiel, Technical Report No. 1217 (2013)
9. Garey, M.R., Graham, R.L.: Bounds for multiprocessor scheduling with resource constraints. SIAM Journal on Computing 4(2), 187–200 (1975)
10. Jansen, K.: A $(3/2+\varepsilon)$ approximation algorithm for scheduling moldable and non-moldable parallel tasks. In: 24th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2012), pp. 224–235 (2012)
11. Jansen, K., Solis-Oba, R.: Rectangle packing with one-dimensional resource augmentation. Discrete Optimization 6(3), 310–323 (2009)
12. Jansen, K., Thöle, R.: Approximation algorithms for scheduling parallel jobs. SIAM Journal on Computing 39(8), 3571–3615 (2010)
13. Kenyon, C., Rémila, E.: A near-optimal solution to a two-dimensional cutting stock problem. Mathematics of Operations Research 25(4), 645–656 (2000)
14. Lee, C.C., Lee, D.T.: A simple on-line bin-packing algorithm. Journal of the ACM 32(3), 562–572 (1985)
15. Lenstra, J.K., Shmoys, D.B., Tardos, É.: Approximation algorithms for scheduling unrelated parallel machines. Mathematical Programming 46, 259–271 (1990)
16. Pascual, F., Rzadca, K., Trystram, D.: Cooperation in multi-organization scheduling. Journal of Concurrency and Computation: Practice and Experience 21, 905–921 (2009)
17. Quezada-Pina, A., Tchernykh, A., Gonzlez-Garca, J., Hirales-Carbajal, A., Miranda-Lpez, V., Ramrez-Alcaraz, J., Schwiegelshohn, U., Yahyapour, R.: Adaptive job scheduling on hierarchical Grids. Future Generation Computer Systems 28(7), 965–976 (2012)
18. Schwiegelshohn, U., Tchernykh, A., Yahyapour, R.: Online scheduling in grids. In: IEEE International Parallel and Distributed Processing Symposium (IPDPS 2008), pp. 1–10 (2008)
19. Seiden, S.S., van Stee, R.: New bounds for multidimensional packing. Algorithmica 36(3), 261–293 (2003)
20. Tchernykh, A., Ramírez, J., Avetisyan, A., Kuzjurin, N., Grushin, D., Zhuk, S.: Two level job-scheduling strategies for a computational grid. In: Wyrzykowski, R., Dongarra, J., Meyer, N., Waśniewski, J. (eds.) PPAM 2005. LNCS, vol. 3911, pp. 774–781. Springer, Heidelberg (2006)
21. Tchernykh, A., Schwiegelsohn, U., Yahyapour, R., Kuzjurin, N.: On-line Hierarchical Job Scheduling on Grids with Admissible Allocation. Journal of Scheduling 13(5), 545–552 (2010)
22. Ye, D., Han, X., Zhang, G.: On-line multiple-strip packing. In: Du, D.-Z., Hu, X., Pardalos, P.M. (eds.) COCOA 2009. LNCS, vol. 5573, pp. 155–165. Springer, Heidelberg (2009)
23. Zhuk, S.N.: Approximate algorithms to pack rectangles into several strips. Discrete Mathematics and Applications 16(1), 73–85 (2006)