Scheduling jobs on heterogeneous platforms

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

Scheduling jobs on heterogeneous platforms*

Marin Bougeret[1], Pierre Francois Dutot[1], Klaus Jansen[2], Christina Robenek[2], and Denis Trystram[1]

[1] LIG, Grenoble University, France
{bougeret,dutot,trystram}@imag.fr
[2] Department of Computer Science
Christian-Albrechts-University Kiel
Christian-Albrechts-Platz 4, 24098 Kiel, Germany.
{kj,cot}@informatik.uni-kiel.de

Abstract. In the context of grid scheduling we consider a scheduling scenario, where parallel jobs have to be scheduled non-preemptively on heterogeneous computational platforms of processors. The speed of the processors may differ among the platforms and the jobs are submitted simultaneously or over the time and cannot run across multiple platforms. We focus on the target of minimizing the total makespan, i.e. the global latest finishing time of a job. In this paper we present an AFPTAS for the problem without release times and show how to generalize our result to malleable jobs and jobs with release times.

1 Introduction

For solving problems that include large-scale computation grid computing gets more and more important. The efficient coordination of those computations appearing as atomic jobs on distributed resources is a difficult task. To get theoretical insights one first need to think of an adequate model that realizes the main principles of grid computing. In this paper we focus on a scheduling scenario, where parallel jobs have to be scheduled non-preemptively on heterogeneous computational platforms of processors. In order to complete extensive computations as fast as possible we are interested in minimizing the total makespan, i.e. the global latest finishing time of a job. The jobs are submitted simultaneously or over the time and cannot run across multiple platforms. In the following we describe our basic model where we consider heterogeneous platforms with different speeds and parallel jobs without release times (*SPP*). Later we fit this model to take malleable jobs and release times into account.

Model. In our setting we have n *jobs* $\{J_1, \ldots, J_n\}$ that have to be scheduled on N *platforms* P_ℓ, $\ell \in \{1, \ldots, N\}$. A platform P_ℓ contains a set M_ℓ of m_ℓ identical processors. We assume the platforms to be sorted by non-decreasing order of their number of processors (or machines), i.e. $m_1 \leq m_2 \leq \ldots \leq m_N$. To each platform is assigned a *speed value* $s_\ell \in \mathbb{R}_{>}0$. Every job J_j is described

* Research supported by German Research Foundation (DFG) project JA612/12-1, "Design and analysis of approximation algorithms for two- and three- dimensional packing problems", and DGA-CNRS

by a pair (p_j, q_j) of the *length of a job* p_j (*number of operations*) and a *number of parallel processors* q_j (*degree of parallelism*), that J_j requires when executed. We assume $q_j \leq m_N = \max_\ell m_\ell$ for all jobs, if not there is no feasible schedule. Since sometimes we will identify jobs with rectangles we call q_j the width of job J_j. Consequently, the *area* (or *work*) of a job is $p_j q_j$ and for a list of jobs or rectangles L we denote with $A(L)$ the total area of the jobs (or rectangles) in L. A job J_j is only allowed to be scheduled within one platform, its processing time in platform P_ℓ is $t_j^\ell := \frac{p_j}{s_\ell}$ if $q_j \leq m_\ell$ else $t_j^\ell = \infty$. We assume furthermore (by scaling) $\min_\ell s_\ell = 1$ and define $t_{\max} := \max_{j,\ell}\{t_j^\ell | t_j^\ell < \infty\}$, which is less than $p_{\max} := \max_j p_j$ (as $\min_\ell s_\ell = 1$). Our objective is to find a non-preemptive schedule of all jobs into the platforms minimizing $C_{\max} := \max_\ell C_{\max}(\ell)$, where $C_{\max}(\ell)$ denotes the completion time of a feasible schedule in P_ℓ. For an instance J of *SPP* let $\mathrm{OPT}(J)$ denote the optimum value for $C_{\max}$.
For a minimization problem as SPP we say that an algorithm B has *absolute ratio* α, if $\sup_J {}^{B(J)}/_{OPT(J)} \leq \alpha$, and *asymptotic ratio* α, if
$\alpha \geq \limsup_{OPT(J)\to\infty} {}^{B(J)}/_{OPT(J)}$, respectively. A minimization problem admits an *(asymptotic) polynomial-time approximation scheme* ((A)PTAS), if there exists a family of polynomial-time approximation algorithms $\{B_\varepsilon | \varepsilon > 0\}$ of (asymptotic) $(1+\varepsilon)$-approximations. We call an approximation scheme *fully polynomial* ((A)FPTAS), if the running time of every algorithm B_ε is bounded by a polynomial in the size of the input $|J|$ and $\frac{1}{\varepsilon}$.
Related work. For $N = 1$ the problem is equivalent to scheduling n parallel jobs on m identical machines. The well-known List Algorithm of Garey and Graham [13] achieves absolute ratio 2 for this problem. For the case that the number of machines is polynomially bounded in the number of jobs a $(1.5+\varepsilon)$-approximation for the contiguous case and a $(1+\varepsilon)$-approximation for the non-contiguous case were given in [11]. For malleable job scheduling there are several results, as e.g. in [4], [9], [7] and [18].

If $N = 1$ and the jobs are assigned to processors of consecutive addresses, the problem corresponds directly to strip packing. For strip packing classical shelf-based algorithms are given in [12]. Further results are given in [20], [22] and [6]. An important result is an AFPTAS for strip packing with additive constant $\mathcal{O}({}^1/_{\varepsilon^2} h_{\max})$ given by Kenyon and Rémila in [14], where $h_{\max}$ denotes the height of the tallest rectangle (i.e. the length of the longest job). This constant was improved by Jansen and Solis-Oba, who presented in [10] an APTAS with additive constant $h_{\max}$.

The problem *SPP* is also closely related to the generalized multiple strip packing (*MSP*) where rectangles have to be packed into strips of infinite height and different widths. Here one wants to find a packing that minimizes the maximum of the heights used in every strip. It corresponds to the case that all platforms are identical and the jobs need to be scheduled on contiguous processors. For *MSP* Zhuk [25] showed that there is no polynomial time approximation algorithm with absolute ratio better than 2 (unless $P = NP$). Later, Ye *et al.* [24] obtained an algorithm for *MSP* with ratio $2+\varepsilon$. In [1] we presented a tight 2-approximation and an AFPTAS for *MSP*. Keep in mind that because of the

contiguity constraint algorithms for *SPP* cannot be directly applied to the generalized *MSP*, but vice versa. However, in general approximation ratios are not preserved, but the optimal value for generalized *MSP* is an upper bound of the optimal value for *SPP*.

Schwiegelshohn *et al.* [21] achieved ratio 3 scheduling parallel jobs on heterogeneous platforms with identical speeds without release times, and ratio 5 with release times. Tchernykh *et al.* presented in [23] an algorithm with absolute ratio 10 without release times. For scheduling parallel jobs on identical platforms, we proposed recently a low cost approximation algorithm with absolute ratio $5/2$ in [2]. We were able to improve our result to a fast $5/2$-approximation for heterogeneous platforms with identical speeds and under the additional constraint that every job can be scheduled in each platform [3].

Our results. In this paper we present an AFPTAS for *SPP* with additive factor $\mathcal{O}(1/\varepsilon^2 p_{\max})$.

Theorem 1. *For every accuracy ε there exists an approximation algorithm with running time polynomial in the size of the input $|J|$ and $1/\varepsilon$ that produces for every instance J of* SPP *a schedule of length at most* $(1+\varepsilon)OPT(J)+\mathcal{O}(1/\varepsilon^2 p_{\max})$.

In practical applications the jobs are usually small compared to the optimum so that an algorithm with a good asymptotic ratio is more applicable than one with a worse absolute ratio. If $p_{\max} \leq \varepsilon^3 \mathrm{OPT}(J)$ for an instance J, the makespan of the schedule constructed by our algorithm is very close to the optimum ($\leq (1+\varepsilon)OPT(J)+O(1/\varepsilon^2 p_{\max}) \leq (1+c\varepsilon)OPT(J)$) for a constant $c \in \mathbb{R}_{\geq 0}$, while an absolute 2 or 3-approximation may be quite far away. To our best knowledge this is also the first result for platform scheduling that takes different speed values among the platforms into account. Since the platforms may have different numbers of processors the AFPTAS in [1] does not apply for *SPP*, because it is based on cutting a solution for a single strip and distributing it well-balanced. Additionally, we do not assume that every job fits in every platform. Thus, the algorithm in [3] does also not apply.

The algorithm and its analysis are given in Section 2. Since we assign each job to processors of consecutive addresses our algorithm also applies to the generalized *MSP*. Moreover, in Section 4 we show how our model and the algorithm can be slightly modified to achieve an AFPTAS for scheduling malleable jobs in heterogeneous platforms with different speeds. In Section 5 we give an AFPTAS for *SPP* with release times. Due to space reduction missing proofs can be found in the appendix.

2 An AFPTAS for *SPP*

Our algorithm is based on an *LP*-relaxation where migration and preemption are allowed. That is a job is allowed to be split into fractions that are executed in different platforms (if they fit). Emanating from the solution of the *LP* we compute a unique assignment of almost all jobs to the platforms. This is done by skillful rounding the fractions of jobs using a result of Lenstra et al. [17]; i.e. the number of remaining fractional jobs per platform will be bounded by $\mathcal{O}(1/\varepsilon^2)$. Remarkably, the rounding technique needs except an (approximate) solution

of the LP no extra information about the speed values. For each platform we reschedule the obtained integral jobs with an approximation algorithm for strip packing [14] and schedule the fractional jobs behind them. An overview of the algorithm is given in Figure 1.

Algorithm 1

1: Solve a linear program relaxation of the problem (1) and get a fractional schedule where preemption and migration are allowed.
2: Group the fractional jobs corresponding to the LP-solution as described in steps 1-4 in Section 2.2 according their widths and for every platform P_ℓ obtain sets L^ℓ_{wide} and L^ℓ_{narrow} of wide and narrow fractional rectangles, respectively.
3: Via a general assignment problem (2) round the fractional rectangles and obtain sets of rounded rectangles $\tilde{L}^\ell_{wide}$, $\tilde{L}^\ell_{narrow}$ and fractional rectangles F^ℓ for $\ell \in \{1, \ldots, N\}$.
4: **for all** $\ell \in \{1, \ldots, N\}$ **do**
5: Pack $\tilde{L}^\ell_{wide} \cup \tilde{L}^\ell_{narrow}$ with the approximation algorithm for strip packing in [14] into platform P_ℓ.
6: Schedule the fractional jobs in F^ℓ greedily on top of the schedule corresponding to the packing obtained before.
7: **end for**

2.1 Relaxed Schedule

Let J be an instance of SPP and let T be the makespan of an optimum schedule for J. To simplify the structure of the schedule instead of handling the specific processing times t^ℓ_j we consider each platform as a two-dimensional bin of width m_ℓ and height Ts_ℓ and schedule the jobs concerning their lengths p_j within this bin. Furthermore, we abandon the constraint that a job has to be scheduled non-preemptively and within only one platform. We represent the schedule of a job $J_j = (p_j, q_j)$ as a (finite) sequence of pairs $(I_i, Q_i)_{i \in I(j)}$, $I(j) \subset \mathbb{N}$, where every $I_i \subset [0, T]$ is a time interval and every Q_i is a set of processors so that there is a uniquely defined platform $P_{\ell_i} \in \{1, \ldots, N\}$ with $Q_i \subset M_{\ell_i}$ and $|Q_i| = q_j$. Additionally, we assume that the following conditions hold:

(i) the time intervals for job J_j within the same platform do not overlap except maybe at the endpoints, i.e. for all $\ell \in \{1, \ldots, N\}$
$$\bigcup_{\substack{i,i' \in I(j), i \neq i' \\ \ell_i = \ell = \ell_{i'}}} \left(\mathring{I}_i \cap \mathring{I}_{i'} \right) = \emptyset, \text{ where } \mathring{A} \text{ denotes the interior of a set } A.$$
(ii) $\sum_{\ell=1}^{N} s_\ell \sum_{\{i \in I(j) | Q_i \subset M_\ell\}} |I_i| \geq p_j$ (covering constraint).
(iii) at any time for every processor there is at most one job running on it.

Keep in mind that under this constraints a job is allowed to be split among the platforms and may be executed in two different platforms at the same time, but never in parallel with itself within the same platform (except for a discrete time, when one piece starts and another ends). It can be executed on two different (not necessary disjoint) subsets of processors within the same platform during different time intervals, where only the endpoints of the time intervals may overlap. An example how such a relaxed schedule can look like is given in Figure 1:

Assume that $T = 10/s_{\ell_1}$ and job J_j needs to be scheduled on $q_j = 3$ processors for $p_j = 7.5$ operations. So in P_{ℓ_1} it is scheduled on processors $\{7, 8, 9\}$ during time $[0, 1/s_{\ell_1}]$ and on processors $\{2, 3, 4\}$ during time $[5/s_{\ell_1}, 7/s_{\ell_1}]$. In P_{ℓ_2} it is scheduled on processors $\{1, 2, 3\}$ during time $[0, 3/s_{\ell_2}]$ and in P_{ℓ_3} it is scheduled on processors $\{3, 4, 5\}$ during time $[3.5/s_{\ell_3}, 5/s_{\ell_3}]$. This gives $1 + 2 = 3$ operations in P_{ℓ_1}, 3 operations in P_{ℓ_2} and 1.5 operations in P_{ℓ_3} (this fulfills the covering constraint). The relaxed schedule can be formulated via the linear

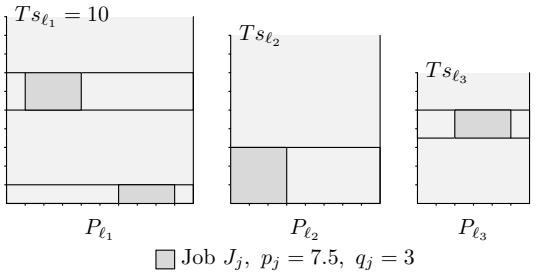

Fig. 1. Relaxed schedule

program below: For each platform in P_ℓ, $1 \leq \ell \leq N$ we introduce configurations C^ℓ. A configuration C^ℓ is a function $C^\ell : \{1, \ldots, n\} \longrightarrow \{0, 1\}$, so that $\sum_{\{j \in \{1,\ldots,n\} | C^\ell(j)=1\}} q_j \leq m_\ell$. It tells us which jobs can be scheduled in parallel in platform P_ℓ. By definition, the number $q(\ell)$ of different configurations for P_ℓ is bounded by 2^n. Let $\mathcal{C}^\ell = \{C^\ell_1, \ldots, C^\ell_{q(\ell)}\}$ denote the set of all configurations for a platform P_ℓ. In the LP below the variable $x_{C^\ell_k}$ indicates the length of configuration C^ℓ_k. That means that the jobs in $\{j \in \{1, \ldots, n\} | C^\ell_k(j) = 1\}$ are executed in platform P_ℓ during $x_{C^\ell_k}$ operation steps.

$$\begin{aligned} \sum_{k=1}^{q(\ell)} x_{C^\ell_k} &= s_\ell T \quad \ell \in \{1, \ldots, N\} \\ \sum_{\ell=1}^{N} \sum_{\{k \in \{1,\ldots,q(\ell)\} | C^\ell_k(j)=1\}} x_{C^\ell_k} &\geq p_j \;\; j \in \{1, \ldots, n\} \\ x_{C^\ell_k} &\geq 0 \quad k \in \{1, \ldots, q(\ell)\}, \;\; \ell \in \{1, \ldots, N\} \end{aligned} \tag{1}$$

The first N constraints ensure that the makespan $C_{\max}(\ell)$ in each platform P_ℓ does not exceed T. The next n constraints are covering constraints for the n jobs. They make sure that every job is executed sufficiently long. We describe how to solve the LP efficiently in the full version of this article.

Lemma 1. *If T is the makespan of an optimum schedule for* SPP(J)*, the linear program above (1) is a relaxation of* SPP(J)*.*

2.2 Rounding the Fractional Solution.

In this section we round the jobs in order to get a unique assignment of every job to a subset of processors of a platform. Consider an approximate solution

($x_{C_k^\ell}$) of the LP-relaxation. We introduce a new variable $x_j^\ell \in [0, p_j]$ that indicates the length of the fraction of job J_j that is scheduled on P_ℓ. Formally this is $x_j^\ell = \sum_{\{k \in \{1,\dots,q(\ell)\} | C_k^\ell(j)=1\}} x_{C_k^\ell}$, the sum of the length of all configurations in P_ℓ in which J_j appears. We can assume for all jobs J_j the equality $\sum_{\ell=1}^{N} x_j^\ell = p_j$, if not we simply delete job J_j from appropriate configurations or replace a configuration by two "shorter" configurations (one with job J_j and one without, their total length is the same as the one of the original configuration). For all fractions x_j^ℓ of a platform P_ℓ we build rectangles (x_j^ℓ, q_j) of height x_j^ℓ and width q_j. In the following steps the rectangles of every platform P_ℓ are grouped geometrically.

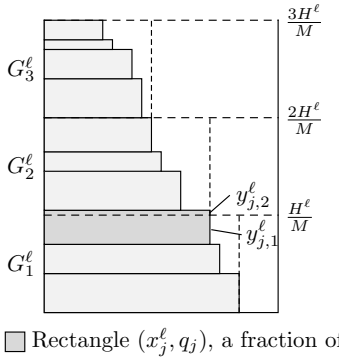

Fig. 2. Constructing L'^ℓ_{wide}

1. Choose $\varepsilon' := \varepsilon/3$ and partition the rectangles into wide and narrow rectangles, $L^\ell_{wide} := \{(x_j^\ell, q_j) | q_j > \varepsilon' m_\ell\}$ and $L^\ell_{narrow} := \{(x_j^\ell, q_j) | q_j \leq \varepsilon' m_\ell\}$.
2. Build a stack of the rectangles in L^ℓ_{wide} ordered by non-increasing width. The total height of the stack is denoted with H^ℓ.
3. Set $M := (1/\varepsilon'^2)$. Divide the stack into M groups G_i^ℓ by drawing $M-1$ horizontal lines at height iH^ℓ/M for $i \in \{1, \dots, M-1\}$ through it. If the interior of a rectangle intersects a horizontal line, cut the rectangle along this line and introduce two new rectangles, so that every rectangle can be assigned to exactly one group. Let L'^ℓ_{wide} denote the modified list of rectangles (see Figure 2). With $y_{j,i}^\ell \in [0, p_j]$ we denote the fraction of job j that is assigned to G_i^ℓ. Let $z_{j,i}^\ell = y_{j,i}^\ell/p_j \in [0,1]$ denote the scaled fraction.
4. Compute $A(L^\ell_{narrow}) = \sum_{(x_j^\ell, q_j) \in L^\ell_{narrow}} x_j^\ell q_j$ and locate the corresponding rectangles on top of the stack as group G_0^ℓ. Let $y_{j,0}^\ell \in [0, p_j]$ denote the fraction of a narrow job J_j that is assigned to G_0^ℓ and and let $z_{j,0}^\ell = y_{j,0}^\ell/p_j \in [0,1]$.

If we were able to round the variables $z_{j,i}^\ell$ to integer values $\{0,1\}$ (without losing too much), this would imply a unique assignment of every rectangle to exactly one group of a platform. Re-identifying the rectangles with jobs, where we identify the height of a rectangle with the length of a job, this would also

imply a unique assignment of every job to a platform. We achieve such a rounding of the variables $z^\ell_{j,i}$ via the following general assignment problem, so that there remain at most $M+1$ fractionally assigned rectangles per platform.

$$\begin{aligned} \sum_{j=1}^{n} z^\ell_{j,0} p_j q_j \le A(L^\ell_{narrow}) \;\; \ell \in \{1,\dots,N\} \\ \sum_{j=1}^{n} z^\ell_{j,i} p_j \le \frac{H^\ell}{M} \;\; i \in \{1,\dots,M\}, \ell \in \{1,\dots,N\} \\ \sum_{\ell=1}^{N}\sum_{i=0}^{M} z^\ell_{j,i} \ge 1 \;\; j \in \{1,\dots,n\} \\ z^\ell_{i,j} \in [0,1] \end{aligned} \tag{2}$$

The above formulation is related to the problem of scheduling jobs on parallel unrelated machines with $(M+1)N$ machines. Each group G^ℓ_i corresponds to a machine. Lenstra et al. showed in [17] that a feasible solution $(z^\ell_{i,j})$ of this problem can be rounded to a feasible solution $(\tilde{z}^\ell_{i,j})$ of the corresponding integer program formulation in polynomial time, so that there remains at most one fractional job $\tilde{z}^\ell_{i,j} < 1$ per machine. Hence, we get a unique assignment of almost all rectangles to the platforms P_ℓ except at most $M+1$ fractionally assigned rectangles per platform. Let F^ℓ denote the set of rectangles with fractional variables $\tilde{z}^\ell_{j,i}$ after the rounding. We will execute the corresponding jobs at the end of the schedule; their total processing time is bounded by $(M+1)t_{\max}$. From now on we consider for each platform P_ℓ an instance of strip packing containing a set of wide rectangles $\tilde{L}^\ell_{wide} := \{(\tilde{z}^\ell_{j,i}p_j, q_j) | \tilde{z}^\ell_{j,i} = 1, i > 0\}$ and a set of narrow rectangles $\tilde{L}^\ell_{narrow} := \{(\tilde{z}^\ell_{j,0}p_j, q_j) | \tilde{z}^\ell_{j,0} = 1\}$. In every platform we repack the pre-assigned rectangles in $\tilde{L}^\ell_{wide} \cup \tilde{L}^\ell_{narrow}$ using the following strip packing subroutine by Kenyon and Rémila [14].

2.3 Strip Packing Subroutine.

For wide rectangles in $\tilde{L}^\ell_{wide}$ we generate a list of rounded rectangles $\tilde{L}^\ell_{sup}$ with only a constant number M of different widths $w_1,\dots,w_M$ for the rectangles: We partition the stack of $\tilde{L}^\ell_{wide}$ into M groups by drawing M horizontal lines at height $\frac{i\tilde{H}^\ell}{M}$, $i \in \{0,1\dots,M-1\}$. Thus, we obtain at most M threshold rectangles, i.e. rectangles that intersect either with their lower bounds or with their interiors such a horizontal line. The widths of rectangles between the ith and the $(i+1)th$ line are rounded up to the width of the ith threshold rectangle, $i \in \{0,1,\dots,M-1\}$. For rectangles above the Mth line we take the width of the Mth threshold rectangle.

The main part of the algorithm is to produce a fractional packing for the rectangles in $\tilde{L}^\ell_{sup}$ using a linear program. In doing so we build configurations $\tilde{C}^\ell_j := \{\tilde{\alpha}^\ell_{ij} : w_i | i \in \{1,\dots,M\}\}$, i.e. multisets of widths where $\tilde{\alpha}^\ell_{ij}$ denotes the number of rectangles of width w_i in $\tilde{C}^\ell_j$ and $\sum_{i=1}^{M} \tilde{\alpha}^\ell_{ij} w_i \le m_\ell$. Then the

following LP is solved.

$$\begin{aligned} \min \sum_{j=1}^{q^\ell} \tilde{x}_j^\ell \\ \text{s.t.} \sum_{j=1}^{q^\ell} \tilde{\alpha}_{ij}^\ell \tilde{x}_j^\ell \geq \beta_i^\ell \text{ for all } i \in \{1,\dots,M\} \\ \tilde{x}_j^\ell \geq 0 \text{ for all } j \in \{1,\dots,q^\ell\}. \end{aligned} \tag{3}$$

The variable $\tilde{x}_j^\ell$ indicates the height of configuration $\tilde{C}_j^\ell$, β_i^ℓ is the total height of rectangles of width w_i in $\tilde{L}_{sup}^\ell$ and q^ℓ denotes the number of possible configurations. A feasible solution of the LP corresponds to a fractional strip packing. The fractional packing can be converted into an integral one. Then the narrow rectangles in $\tilde{L}_{narrow}^\ell$ are added in the remaining space next to the integral packing and on top of it with Next Fit Decreasing Height heuristic.

3 Analysis

In the end we re-identify the rectangles with jobs, i.e. their widths with q_j and their heights with p_j. Note that a packing of the rectangles of total height h^ℓ in platform P_ℓ corresponds to a schedule with makespan h^ℓ/s_ℓ. Then the fractional jobs in F^ℓ are scheduled on top. To directly apply strip packing results we scale the widths of all rectangles in $\tilde{L}_{sup}^\ell \cup \tilde{L}_{narrow}^\ell$ by $1/m_\ell$. Furthermore we consider platform P_ℓ as a strip of width 1 and infinite height. As we consider each platform and the allocated jobs independently, this has no impact on the solution.

3.1 Analyzing the Output

Let $(x_{C_k^\ell})$ be an approximate solution of (1) and let $\tilde{L}_{wide}^\ell \cup \tilde{L}_{narrow}^\ell$ contain the rectangles that have to be repacked in Step 5 of Algorithm 1 with the strip packing subroutine above. For a list of rectangles L let $Lin_{SP}(L)$ denote the height of an optimal fractional strip packing for the rectangles in L. By construction we have that the height of an optimal fractional strip packing for the wide and narrow rectangles in $L_{wide}^\ell \cup L_{narrow}^\ell$ into platform P_ℓ, is less than the length of the schedule corresponding to the approximate solution of (1) constructed in step 1, that is $Lin_{SP}(L_{wide}^\ell \cup L_{narrow}^\ell) \leq s_\ell(1+3\varepsilon)Lin(J)$. Let "$\leq$" denote a partial order on lists of rectangles. For a list of rectangles L let S denote the shape of a stack built as described above. We say $L \leq L'$ for two lists of rectangles, if shape S' covers S. It is clear that $Lin_{SP}(\tilde{L}_{wide}^\ell) \leq Lin_{SP}(\tilde{L}_{sup}^\ell)$, since $\tilde{L}_{wide}^\ell \leq \tilde{L}_{sup}^\ell$. With Lemma 3 in [14] we conclude $Lin_{SP}(\tilde{L}_{sup}^\ell) \leq \left(1+\frac{1}{M\varepsilon'}\right) Lin_{SP}(\tilde{L}_{wide}^\ell)$ and $A(\tilde{L}_{sup}^\ell) \leq \left(1+\frac{1}{M\varepsilon'}\right) A(\tilde{L}_{wide}^\ell)$.

We go on with step 2 and consider the stack built from $L_{wide}'^\ell$ in the 3rd step of the grouping procedure in Section 2.2. We introduce a new list of rectangles $L_{sup}'^\ell$ that is developed when in each group G_i^ℓ of the stack, $i \in \{1,\dots,M\}$, the width of each rectangle is rounded up to the widest width of a rectangle that is contained in this group. Remember that every rectangle in $L_{wide}'^\ell$ is uniquely assigned to one of the groups G_i^ℓ since we introduced two new rectangles for border rectangles before. Notice that during building $\tilde{L}_{wide}^\ell$ in step 3 of Algorithm

1 we do not increase the total height of any group G_i^ℓ and we do not exceed the largest width of a rectangle that appears in it. Thus, we obtain $\tilde{L}^\ell_{wide} \leq L'^\ell_{sup}$. Since $Lin_{SP}(L'^\ell_{wide}) = Lin_{SP}(L^\ell_{wide})$ and $A(L'^\ell_{wide}) = A(L^\ell_{wide})$ this gives:

Lemma 2. *For all $\ell \in \{1, \ldots, N\}$ we have*

a) $Lin_{SP}(\tilde{L}^\ell_{sup}) \leq \left(1 + \frac{1}{M\varepsilon'}\right)^2 Lin_{SP}(L^\ell_{wide})$
b) $A(\tilde{L}^\ell_{sup}) \leq \left(1 + \frac{1}{M\varepsilon'}\right)^2 A(L^\ell_{wide})$.

Let h^ℓ_{sup} denote the height of the packing produced by converting the fractional solution of (3) into an integral one. This is done by adding for each configuration appearing with height > 0 in the fractional solution the maximum height of a rectangle. Each basic solution of (3) has at most M non-zero entries and one can show that there are effectively at most $2M$ different configurations in platform P_ℓ [14]. So we conclude $h^\ell_{sup} \leq Lin_{SP}(\tilde{L}^\ell_{sup}) + (1 + 2M) \max\{p_j | (p_j, q_j) \in \tilde{L}^\ell_{sup}\}$. Note that we only packed the rounded rectangles in $\tilde{L}^\ell_{sup}$ so far. Let h^ℓ denote the height after adding the narrow rectangles in $\tilde{L}^\ell_{narrow}$ to platform P_ℓ, $\ell \in \{1, \ldots, N\}$. We can now bound h^ℓ:

Lemma 3. *For all $\ell \in \{1, \ldots, N\}$ we have*
$h^\ell \leq (1 + 7\varepsilon) Lin(J) s_\ell + \mathcal{O}(1/\varepsilon^2) \max\{p_j | (p_j, q_j) \in L^\ell_{wide} \cup L^\ell_{narrow}\}$.

The packing in each platform P_ℓ corresponds to a schedule with length (referring to p_j) at most $(1 + 7\varepsilon) Lin(J) s_\ell + (\frac{36}{\varepsilon^2} + 1) \max\{p_j | (p_j, q_j) \in L^\ell_{wide} \cup L^\ell_{narrow}\}$, thus we conclude that its completion time (referring to t^ℓ_j) is bounded by $(1 + 7\varepsilon) Lin(J) + \mathcal{O}(\frac{1}{\varepsilon^2} t_{\max})$. The remaining jobs in F^ℓ have total processing time bounded by $(M + 1) t_{\max} \in \mathcal{O}(\frac{1}{\varepsilon^2} t_{\max}) \leq \mathcal{O}(\frac{1}{\varepsilon^2} p_{\max})$, since $t_{\max} \leq p_{\max}$ as $\min s_\ell = 1$. Adding now the remaining jobs in F^ℓ to the schedule does not change the magnitude of the additive factor. With rescaling ε and since $Lin(J) \leq \mathrm{OPT}(J)$ we obtain that the makespan of the produced schedule in each platform P_ℓ is less than $C_{\max}(\ell) \leq (1 + \varepsilon)\mathrm{OPT}(J) + \mathcal{O}(\frac{1}{\varepsilon^2} p_{\max})$ and conclude our main Theorem 1. Since during the repacking process we considered jobs as rectangles, we assigned every job to a set of processors with consecutive addresses. Thus we also obtain an AFPTAS for multiple strip packing for strips with different widths (in this case we have $s_\ell = 1$ for all $\ell \in \{1, \ldots, N\}$).

3.2 Running Time of the Algorithm

The time needed for solving (1) approximately via max-min resource sharing (details in the full version)in step 1 is $O(Nn^2\varepsilon^{-6} \log^2(n) \log^2(1/\varepsilon) \log(N \max s_\ell))$. The number of non-zero configurations in the final solution is bounded by $O(n(\varepsilon^{-2} + \ln n))$ [15]. So step 2 takes time $O(Nn^2(\varepsilon^{-2} + \log n) \log(n^2(\varepsilon^{-2} + \log n))$
$= O(Nn^2\varepsilon^{-2} \log^2(n) \log(1/\varepsilon))$, since there are at most $n^2(\varepsilon^{-2} + \log n)$ rectangles in each platform that have to be sorted. We represent the assignment problem in step 3 as a weighted bipartite graph $G = (V_1, V_2, E)$, where V_1 corresponds to the $N(M + 1)$ machines (parts of the stacks), V_2 to the jobs. There is an edge between the node representing part i of the stack for P_ℓ and

the node representing job J_j if $z^\ell_{j,i} > 0$. This assignment problem can be converted in time $O(|E||V_1|) = O(|V_1|^2|V_2|) = O(\varepsilon^{-2}N^2n)$ into another assignment problem, whose corresponding graph is a forest [19]. Applying the rounding technique in [17] to the new assignment takes time in $O(|V_1| + |V_2|) = O(\varepsilon^{-2}N + n)$. So step 3 takes time in $O(\varepsilon^{-2}N^2n)$. In step 5 it is sufficient to solve the corresponding linear program (3) approximatively with accuracy ε also via a max-min resource sharing problem. This can be done in time $O(M(\varepsilon^{-2}+\ln M)\ln(\varepsilon^{-1})\max\{M+\varepsilon^{-3}, M\ln\ln(M\varepsilon^{-1})\})$ for every platform [8]. Since $M \in O(\varepsilon^{-2})$ this gives for step 5 a total running time in $O(N\varepsilon^{-7})$. The overall running time sums up to $O(\varepsilon^{-7}N^2n^2\log^2(n)\log^2(1/\varepsilon)\log(N\max s_\ell))$.

4 Malleable Jobs

One can also obtain an AFPTAS for scheduling malleable jobs non-preemptively by only adding a few modifications to the algorithm. To get a better overview we do not consider the platform speeds here. But remember that one can easily add speeds here by considering bins of height $s_\ell T$ instead of T, where T denotes an optimum value for the makespan for scheduling malleable jobs in platforms. In the following we give a short instruction how to adjust our algorithm:
In malleable scheduling a job J_j is described by a function $p_j : \{1,\dots,m_N\} \longrightarrow \mathcal{Q}^+ \cup \infty$, where $p_j(k)$ is the length of job j running on k parallel processors of a platform. We introduce a configuration as a map $f_\ell : \{1,\dots,m_\ell\} \longrightarrow \{0\} \cup \{1,\dots,n\}$ that assigns a processor to a job (0 for idle time). Instead of solving (1) we can solve in a similar way the following linear program:

$$\begin{aligned} \sum_{f_\ell \in \mathcal{F}^\ell} x_{f_\ell} &= T \;\; \ell \in \{1,\dots,N\} \\ \sum_{\ell=1}^{N}\sum_{k=1}^{m_\ell} \frac{1}{p_j(k)} \sum_{f_\ell \in \mathcal{F}^\ell, |f^{-1}(j)=k|} x_{f_\ell} &\geq 1 \;\; j \in \{1,\dots,n\} \\ x_{f_\ell} &\geq 0. \end{aligned} \tag{4}$$

Consider step 2 of the algorithm. Let a^ℓ_i, b^ℓ_i be the smallest and the largest width of a rectangle in group G^ℓ_i and let $W^\ell_{i,j}$ be the set of widths job J_j adopts in G^ℓ_i. To guarantee that we have chosen the right number of processors for a job we add the following steps before rounding the jobs via the general assignment problem:

- For $i \in \{1,\dots,M\}$ and $w \in W^\ell_{i,j}$ let $y^\ell_{j,i}(w)$ denote the fraction of job j of width w that is assigned to G^ℓ_i. Let $z^\ell_{j,i} = \sum_{w\in W_{i,j}} y^\ell_{j,i}(w)$ be the complete fraction of job j in G^ℓ_i.
- For each part $i \in \{1,\dots,M\}$ and job j with $|W^\ell_{j,i}| \geq 2$ compute $k^\ell_{j,i} := \arg\min_{k\in[a^\ell_i,b^\ell_i]} p^\ell_j(k)$ and replace the rectangles corresponding to job j in G^ℓ_i by $(z^\ell_{j,i}p_j(k^\ell_{j,i})k^\ell_{j,i})$. Note that $p_j(k^\ell_{j,i})$ is the smallest processing time among all processor numbers $k \in [a^\ell_i, b^\ell_i]$.
- For each job j with $|W^\ell_{j,0}| \geq 2$ compute $k^\ell_{j,0} := \arg\min_{k\in[0,\varepsilon' m_\ell]} p^\ell_j(k)k$ and replace all rectangles corresponding to job j in G^ℓ_0 by $(z^\ell_{j,0}p_j(k^\ell_{j,0}), k^\ell_{j,0})$.

Including different speed values we define the processing time of job J_j in platform P_ℓ as $t_j^\ell(k) = \frac{p_j(k)}{s_\ell}$. Note that $t_j^\ell(k) = \infty$ is possible. We define $p_{\max} := \max_{j,k}\{p_j(k)|p_j(k) < \infty\}$ and $t_{\max} := \max_{j,k,\ell}\{t_j^\ell(k)|t_j^\ell(k) < \infty\}$. To include speed values in the linear program we change the first N constraints of LP (4) into $... = s_\ell T$, since different speeds can be considered as providing length $s_\ell T$ instead of T for the schedule. During the repacking process the algorithm remains the same and finally we achieve the following theorem

Theorem 2. *There is an AFPTAS for scheduling non-preemptive malleable jobs in heterogeneous platforms with different speeds with additive factor* $\mathcal{O}(1/\varepsilon^2 p_{\max})$.

5 Release Times

Theorem 3. *There is an AFPTAS for scheduling parallel jobs in heterogeneous platforms with different speeds and release times with additive factor* $O(1/\varepsilon^3 p_{\max})$.

For a better overview we describe here the idea for the proof when all platforms run with the same speed, i.e. $s_\ell = 1$ for all $\ell \in \{1, \ldots, N\}$. The general case can be derived from it. Let r_j denote the release time of job J_j and $\Phi := \max_j r_j$. We assume that $\Phi > \varepsilon T$, otherwise it is easy. As in [5] we round down the release times to the next multiples of $i\varepsilon T$ $i \in \{0, 1, \ldots, 1/\varepsilon\}$ and obtain new release times $\tilde{r}_1 \ldots, \tilde{r}_n$ with at most $R = O(1/\varepsilon)$ different values $\rho_1, \ldots, \rho_R$. To recover the loss we made by rounding down we shift the final schedule by εT in the end. For every platform P_ℓ we consider R new platforms $\tilde{P}_{\ell,i}$, $i \in \{1, \ldots, R\}$, with m_ℓ processors and create a new instance $\tilde{J}_R$ of *SPP* (without release times) with RN platforms and n jobs. A job J_j can now be scheduled in platform $\tilde{P}_{\ell,i}$ if it fits and if it is already released, i.e. $q_j \leq m_\ell$ and $\tilde{r}_j \leq \rho_i$. For each of the new platforms $\tilde{P}_{\ell,i}$ the value of an optimal fractional schedule is at most εT.

References

1. M. Bougeret, P. F. Dutot, K. Jansen, C. Otte, and D. Trystram. Approximation algorithms for multiple strip packing. In *Proceedings of the 7th Workshop on Approximation and Online Algorithms (WAOA 2009), Lecture Notes in Computer Science, Springer,* 2009.
2. M. Bougeret, P. F. Dutot, K. Jansen, C. Otte, and D. Trystram. A low cost 5/2 approximation for scheduling rigid jobs on multiple organizations. In *6th IFIP International Conference on Theoretical Computer Science (TCS 2010)*, 2009.
3. M. Bougeret, P. F. Dutot, K. Jansen, C. Otte, and D. Trystram. A fast 5/2-approximation algorithm for hierarchical scheduling. In *Euro-Par 2010, Lecture Notes in Computer Science, Springer.*, 2010.
4. J. Du and Joseph Y.-T. Leung. Complexity of scheduling parallel task systems. *SIAM J. Discrete Math.*, 2(4):473–487, 1989.
5. L. A. Hall and D. B. Shmoys. Approximation Schemes for Constrained Scheduling Problems. In *30th Annual Symposium on Foundations of Computer Science (FOCS 1989)*, pages 134–139, 1989.
6. R. Harren, K. Jansen, L. Prädel, and R. van Stee. A $5/3+\varepsilon$ approximation for strip packing. *To appear in: The 12th Symposium on Algorithms and Data Structures, (WADS 2011).*

7. K. Jansen. Scheduling malleable parallel tasks: An asymptotic fully polynomial time approximation scheme. *Algorithmica*, 39(1):59–81, 2004.
8. K. Jansen. Approximation algorithms for min-max and max-min resource sharing problems and applications. *Efficient Approximation and Online Algorithms*, volume 3483 of *Lecture Notes in Computer Science*, pages 156–202, Springer, 2006.
9. K. Jansen and L. Porkolab. Linear-time approximation schemes for scheduling malleable parallel tasks. *Algorithmica*, 32(3):507–520, 2002.
10. K. Jansen and R. Solis-Oba. New approximability results for 2-dimensional packing problems. In *Mathematical Foundations of Computer Science*, volume 4708 of *Lecture Notes in Computer Science*, pages 103–114, Springer, 2007.
11. K. Jansen and R. Thöle. Approximation algorithms for scheduling parallel jobs: Breaking the approximation ratio of 2. In *International Colloquium on Automata, Languages and Programming*, pages 234–245, 2008.
12. 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 of Computing*, 9(4):808–826, 1980.
13. M. R. Garey and R. L. Graham. Bounds for Multiprocessor Scheduling with Resource Constraints. *SIAM Journal of Computing*, 4(2):187–200, 1975.
14. C. Kenyon and E. Rémila. A near optimal solution to a two-dimensional cutting stock problem. *Mathematics of Operations Research*, 25:645–656, 2000.
15. L.G. Khachiyan, M. D. Grigoriadis, L. Porkolab, and J. Villavicencio. Approximate max-min resource sharing for structured concave optimization. *SIAM Journal on Optimization*, 11:1081–1091, 2001.
16. E. L. Lawler. Fast approximation algorithms for knapsack problems. *Mathematics of Operation Research*, 4(4):339–356, 1979.
17. J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Program.*, 46:259–271, 1990.
18. G. Mounié, C. Rapine, and D. Trystram. Efficient approximation algorithms for scheduling malleable tasks. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 23–32, 1999.
19. S. A. Plotkin, D. B. Shmoys, É. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20:257–301, 1995.
20. I. Schiermeyer. Reverse-fit: A 2-optimal algorithm for packing rectangles. In *Proceedings of the Second European Symposium on Algorithms*, pages 290–299. Springer-Verlag, 1994.
21. U. Schwiegelshohn, A. Tchernykh, and R. Yahyapour. Online scheduling in grids. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1–10, 2008.
22. A. Steinberg. A strip-packing algorithm with absolute performance bound 2. *SIAM Journal of Computing*, 26(2):401–409, 1997.
23. A. Tchernykh, J. Ramírez, A. Avetisyan, N. Kuzjurin, D. Grushin, and S. Zhuk. Two level job-scheduling strategies for a computational grid. In *6th International Conference on Parallel Processing and Applied Mathematics (PPAM 2005)*, pages 774–781, 2005.
24. D. Ye, X. Han, and G. Zhang. On-line multiple-strip packing. In *The 3rd Annual International Conference on Combinatorial Optimization and Applications(COCOA 2009)*, 2009.
25. S.N. Zhuk. Approximate algorithms to pack rectangles into several strips. *Discrete Mathematics and Applications*, 16(1):73–85, 2006.