Approximation schemes for machine scheduling with resource (in-)dependent processing times*

Klaus Jansen, Marten Maack, Malin Rau
Institute of Computer Science, University of Kiel, 24118 Kiel, Germany
{kj,mmaa,mra}@informatik.uni-kiel.de

Abstract

We consider two related scheduling problems: resource constrained scheduling on identical parallel machines and a generalization with resource dependent processing times. In both problems, jobs require a certain amount of an additional resource and have to be scheduled on machines minimizing the makespan, while at every point in time a given resource capacity is not exceeded. In the first variant of the problem the processing times and resource amounts are fixed, while in the second the former depends on the latter. We present asymptotic fully polynomial approximation schemes (AFPTAS) for the problems: For any $\varepsilon > 0$ a schedule of length at most $(1+\varepsilon)$ times the optimum plus an additive term of $\mathcal{O}(1/\varepsilon^2)$ is provided, and the running time is polynomially bounded in $1/\varepsilon$ and the input length. Up to now only approximation algorithms with constant approximation ratios were known.

1 Introduction

In this paper we study the two following scheduling problems. First, the *(single) resource constrained scheduling problem*, where n jobs have to be scheduled on m identical parallel machines, while each job has a certain requirement of one additional limited renewable resource; and second *scheduling with resource dependent processing times*, where the resource requirement is not fixed and the processing times depend on the resource amount allocated to a job. For both problems the goal is to minimize the makespan, i.e. the length of the schedule.

These problems arise naturally in different contexts, e.g. in highly parallelized computing where simultaneously active jobs share common memory, or in production logistics where additional personnel may speed up certain tasks. From a theoretical perspective on the other hand, the problems are sensible generalizations of classical problems like scheduling on identical parallel machines or bin packing.

We study approximation algorithms: A *α-approximation algorithm* for a minimization problem computes a solution of value $A(I) \leq \alpha \mathrm{OPT}(I)$, where $\mathrm{OPT}(I)$ is the optimal value for a given instance I. For some problems the *asymptotic* approximation ratio $\lim_{x\to\infty} \sup\{A(I)/x | \forall I : \mathrm{OPT}(I) = x\}$ is of interest. Moreover, a family of algorithms consisting of $(1+\varepsilon)$-approximation algorithms for each $\varepsilon > 0$ with running time polynomial in the input length (and $1/\varepsilon$) is called *(fully) polynomial time approximation scheme* (F)PTAS; and if the asymptotic ratio is concerned it is called *asymptotic* (F)PTAS or A(F)PTAS.

Related Work The problems studied here are generalizations of the widely studied problem of scheduling on identical parallel machines. As early as 1966 Graham presented his famous list scheduling for this problem, which provides a schedule of length at most $(2-\frac{1}{m})\mathrm{OPT}(I)$. Later a PTAS was found by Hochbaum and Shmoys [9] and for the special case that the number of machines is constant an FPTAS is possible [10]. The first result for scheduling jobs on identical machines with additional resources was presented in 1975 by Garey and

*Research was in part supported by German Research Foundation (DFG) project JA 612 /14-2

Graham [4]. Given s distinct resources they have shown that the greedy list algorithm delivers a schedule of length at most $(s+2-(2s+1)/m)\mathrm{OPT}$. In the same year Garey and Johnson [5] showed that this problem is NP-complete even if just one resource is given. By a simple reduction to the partition problem, one can see that there is no algorithm with a better approximation guarantee than 3/2 for this case, unless P = NP. Therefore a PTAS is not possible for this problem (and also for the resource dependent variant), while an A(F)PTAS still can be achieved. Lately, Niemeier and Wiese [17] presented a $(2+\varepsilon)$-approximation for resource constrained scheduling, and this is the best ratio known so far. In the case that the jobs have unit processing times, we have the problem of bin packing with capacity constraints, for which an AFPTAS was recently presented by Epstein and Levin [3]. If the resource has to be allocated continuously, the problem of strip packing with cardinality constraints is given. Moreover, if we have less jobs than machines, both of the here considered problems are reducible to the problem of scheduling parallel or malleable parallel tasks, for which AFPTAS were presented by Kenyon and Rémila [14] and Jansen [11] respectively.

For scheduling with resource dependent processing times the first result was achieved by Grigoriev et al. [7], who studied the unrelated machines variant in which the processing times depend on the machine as well as on the resource assignment. They achieved a 3.75-approximation algorithm. This was improved to $3.5+\varepsilon$ by Kellerer [13] for the identical machine version. Grigoriev et al. [8] presented a $3+\varepsilon$-approximation for a version of the problem where the jobs are preassigned to machines, that also works when the processing time functions are encoded more succinctly.

Results and methodology We present AFPTAS for both problems at hand which, given an accuracy $\varepsilon > 0$ and an instance I, produce schedules with makespan at most $(1+\varepsilon)\mathrm{OPT}(I) + O(1/\varepsilon^2)p_{\max}$ and have a running time that is polynomial in the input length and $1/\varepsilon$.

We carefully alter and recombine a variety of linear programming and rounding techniques. To solve certain configuration LPs, we apply highly efficient algorithms [6] for the so called max-min resource sharing problem, as was done in [11]. We partition the jobs into wide and narrow jobs based on their resource requirements and apply linear grouping for the wide jobs. To handle the narrow jobs we adapt the notion of windows that was introduced by Epstein and Levin [3]. However, and this is crucial for both our algorithms, we manage to bound the number of windows to be in $\mathcal{O}(1/\varepsilon^2)$, via a second elaborate rounding step. This makes the small additive factor possible, and is essential for the generalization to the resource dependent variant. Our results also yield a significantly improved AFPTAS for bin packing with capacity constraints. Both of our algorithms even work for the case when the resource needs to be allocated contiguously.

2 Scheduling with one additional resource

An instance $I = (\mathcal{J}, m, R)$ of the resource constrained scheduling problem is given by a number of machines m, a set of n jobs $\mathcal{J}$ and a resource limit $R \in \mathbb{Q}$. Each job in $\mathcal{J}$ is given by a triple $(j,p,r) \in \mathbb{N}\times\mathbb{Q}^2$, where j is its identifier, p its processing time and r its resource amount. In the following we write $j \in \mathcal{J}$ and mean its identifier and write p_j and r_j to refer to its processing time and resource amount. The goal is to find mappings $\mu : \mathcal{J} \to [m]$ from jobs to machines and $\tau : \mathcal{J} \to \mathbb{Q}$ from jobs to starting times minimizing the makespan $\max_{j\in\mathcal{J}}(\tau(j)+p_j)$ of this schedule. μ and τ must ensure that at every point in time each machine processes at most one job and the total resource amount of jobs scheduled at this point in time is at most R. For each mapping $\tau : \mathcal{J} \to \mathbb{Q}$, with

$$\forall t \in \mathbb{Q} : \sum_{j : t \in [\tau(j), \tau(j)+p_j)} r_j \le R, \tag{2.1}$$

$$\forall t \in \mathbb{Q} : \sum_{j : t \in [\tau(j), \tau(j)+p_j)} 1 \le m, \tag{2.2}$$

we can generate in polynomial time a mapping $\mu : \mathcal{J} \to [m]$, which ensures that each machine

processes at most one job at each point in time. For each set of jobs $\mathcal{J}'$ we define $P(\mathcal{J}') = \sum_{j\in\mathcal{J}'} p_j$.

THEOREM 2.1. *Let $I = (\mathcal{J}, m, R)$ be an instance of the resource constrained scheduling problem. For each $\varepsilon \in (0,1)$ there is an algorithm which computes a schedule with makespan*

$$T \leq (1+\varepsilon)\mathrm{OPT}(I) + \mathcal{O}(1/\varepsilon^2)p_{\max},$$

where $p_{\max}$ is the maximal processing time in $\mathcal{J}$. The algorithm needs

$$\mathcal{O}(n(\ln(n)+\varepsilon^{-2})(n^{1.5356}+m\varepsilon^{-4}))$$

operations.

In the following we will call a jobs resource amount its width and its processing time its height. We differ two cases: first $1/\varepsilon < m$ and second $1/\varepsilon \geq m$. In the first case the jobs are partitioned in wide and narrow jobs, where wide jobs have a larger resource amount than narrow jobs. The resource amount of the wide jobs is rounded by linear grouping, such that we have do deal with just $\mathcal{O}(\varepsilon^{-2})$ different sizes.

For this rounded instance the algorithm computes a preemptive schedule from a configuration LP. Broadly speaking, a configuration is a selection of jobs, that can run at the same time. After that each configuration is partitioned in the wide job part and the narrow job part. The space (resource amount and used machines) not used by the wide jobs will be called window, following the notation in [3]. In simplified terms a window can be seen as the residual space (resource and machine number) that is left by a configuration.

By constructing a preemptive schedule first, instead of solving the LP by Levin and Epstein [3] directly, we manage to choose each windows width more adjusted to the given instance. This improves the number of operations and the makespan of or solution. In particular the number of operations needed does no longer depend on the variable R or the minimal resource amount besides zero. Nevertheless the crucial step is to reduce the number of different window sizes such that it just depends on ε by simultaneously adding (not too much) processing time to the schedule. This is done by a further grouping step. After a solution with reduced number of windows is found an integral schedule can be computed by adding some processing time to the schedule.

In the second case we do not need to partition the jobs by resource amount. This makes things much easier, such that we can generate a schedule directly after getting the preemptive solution.

Algorithm Given an instance I and $\varepsilon > 1/m$ the algorithm can be summarized as follows:

(i) Define $\varepsilon' := \varepsilon/5$, compute $p_{\max} = \max\{p_j | j \in \mathcal{J}(I)\}$ and construct a rounded instance $I_{\sup}$ with at most $1/\varepsilon'^2$ wide jobs using linear grouping and excluding the widest group.

(ii) Find a preemptive schedule x_{pre} for $I_{\sup}$ with makespan at most $(1+\varepsilon')\mathrm{OPT}_{\mathrm{pre}}$.

(iii) Transform the obtained configurations into generalized configurations, which are composed of a configuration part for wide and a window part for narrow jobs.

(iv) Reduce the number of different windows using a grouping step.

(v) Generate a solution $(\bar{x}, \bar{y})$ for $I_{\sup}$ and a generalized configuration linear program LP_W, wich has at most $4+3/\varepsilon'^2+|\mathcal{J}_N|$ nonzero components, and which is lengthened at most by a factor of $(1+\varepsilon')$.

(vi) Given $(\bar{x}, \bar{y})$, generate an integral schedule for I without the group of widest jobs lengthening the schedule by at most $(4+3/\varepsilon'^2)p_{\max}$ and ε' times the previous schedule length.

(vii) Add the widest jobs greedily at the end of the integral schedule obtaining an overall makespan of at most $((1+\varepsilon')^3 + \varepsilon')\mathrm{OPT}_{\mathrm{pre}}(I_{\sup}) + (5+1/\varepsilon'+3/\varepsilon'^2)p_{\max}$.

2.1 First Case: $1/\varepsilon < m$.

Let $\varepsilon > 0$, with $1/\varepsilon < m$, and an instance $I = (\mathcal{J}, m, R)$ be given. W.l.o.g. we assume that $1/\varepsilon \in \mathbb{Z}$ and $m < n$, since else we have the problem of scheduling parallel tasks, for which an AFPTAS is already known [11]. We define $\varepsilon' = \varepsilon/5$. First we partition the given set of jobs $\mathcal{J}$ into a set of wide jobs

$\mathcal{J}_W := \{j \in \mathcal{J} | r_j \geq \varepsilon' R\}$ with resource amount at least $\varepsilon' R$ and a set of narrow jobs $\mathcal{J}_N := \mathcal{J} \setminus \mathcal{J}_W$. Now we apply linear grouping, a method introduced by Fernandez de la Vega and Lueker [2] for bin packing and extended by Kenyon and Rémila [14] to strip packing, to round the resource amount of the wide jobs. We interpret each job j as a rectangle with width r_j and height p_j. We place this rectangles on a vertical stack by order of increasing resource amount. Let $P_W := P(\mathcal{J}_W)$ be the height of the stack. Now consider the horizontal lines at height $i\varepsilon'^2 P_W$. Each job intersecting with one of these lines is divided at this line and split into two new jobs. Define by $\mathcal{J}_{W,i}$ the set of jobs, which lie between the lines $(i-1)\varepsilon'^2 P_W$ and $i\varepsilon'^2 P_W$. We get $G := 1/\varepsilon'^2$ many sets $\mathcal{J}_{W,i}$, called groups, where $\mathcal{J}_{W,G}$ is the group of jobs with the largest resource amount. Define by $R_i := \max\{r_j | j \in \mathcal{J}_{W,i}\}$ the largest resource amount in group i. Note that $R_i \leq R_{i+1}$ for all $i < G$.

We now generate an instance where the resource amount of the wide jobs is rounded up. Let $\mathcal{J}_{\text{sup}}$ be the set containing all jobs from $\mathcal{J}_N$ and one additional job for each $i \in \{1, \dots G-1\}$ with processing time $P(\mathcal{J}_{W,i})$ and resource amount R_i and let $I_{\text{sup}} := (\mathcal{J}_{\text{sup}}, m, R)$. The next step is to generate a preemptive schedule for this rounded up instance. Note that the jobs in $\mathcal{J}_{W,G}$, the jobs with largest resource amount, are scheduled in the last step. We denote by $\mathcal{J}_{\text{sup},W}$ the wide jobs in I_{sup}.

Preemptive Schedule In a preemptive schedule each job can be interrupted and restarted at no cost. We even allow wide jobs to be scheduled on more than one machine at the same point in time. This is necessary, since wide jobs with the same rounded resource amount are allowed to be scheduled at the same point in time. We denote its optimum by $\text{OPT}_{\text{pre}}(I)$.

A *configuration* $C \in \mathbb{N}^{\mathcal{J}}$ is a multiset of jobs, where $C(j)$ says how often the job j is contained in in the configuration. We allow $C(j) \in \{0,1\}$, if $j \in \mathcal{J}_N$ and $C(j) \in \{0, \dots, 1/\varepsilon'\}$, if $j \in \mathcal{J}_W$. A configuration is *valid* for a given instance I, if $m(C) := \sum_{j \in \mathcal{J}} C(j) \leq m$ and $R(C) := \sum_{j \in \mathcal{J}} C(j) r_j \leq R$. Denote by $\mathcal{C}_I$ the set of all valid configurations of I. An optimal solution of the following linear program $\text{LP}_{\text{pre}}(I)$ delivers an optimal preemptive schedule for an instance I. The variable x_C denotes the processing time of configuration $C \in \mathcal{C}_I$.

$$\min \sum_{C \in \mathcal{C}_I} x_C \tag{2.3}$$

$$\sum_{C \in \mathcal{C}_I} C(j) x_C \geq p_j \quad \forall j \in \mathcal{J} \tag{2.4}$$

$$x_C \geq 0 \quad \forall C \in \mathcal{C}_I \tag{2.5}$$

LEMMA 2.1. *For any Instance I and any $\varepsilon > 0$ we can find a solution x_{pre} to $\text{LP}_{\text{pre}}(I)$ with*

$$\sum_{C \in \mathcal{C}_I} (x_{pre})_C \leq (1+\varepsilon')\text{OPT}_{\text{pre}}(I)$$

in $\mathcal{O}(n(\ln(n) + \varepsilon^{-2})(|\mathcal{J}_W|/\varepsilon' + n + m/\varepsilon^4))$ operations.

Furthermore a solution with the same objective value and at most $|\mathcal{J}|+1$ non zero components can be found in $\mathcal{O}(n^{2.5356}(\ln(n) + \varepsilon^{-2}))$ operations.

Proof. This can be done solving a max-min-resource-sharing problem with the algorithm by Grigoriadis et al. [6].

In the max-min-resource-sharing problem a nonempty convex compact set B and a set of M nonnegative continuous concave functions $f : B \to \mathbb{R}^M$ is given. We are interested in finding the value

$$\lambda^* := \max\{\lambda | \exists x \in B \forall j \leq M : f_j(x) \geq \lambda\}$$

and a vector $x \in B$, for which $f(x) \geq \lambda^* e$ holds, where $e \in \mathbb{R}^M$ is the vector which is 1 at each position.

When we interpret LP_{pre} as an max-min-resource-sharing problem, we define

$$B := \{x \in \mathbb{R}_{\geq 0}^{|\mathcal{C}_I|} | \sum_{C \in \mathcal{C}_I} x_C = 1\}$$

the set of selections of valid configurations with overall makespan 1, where 1 is the normalisation to $\text{OPT}_{\text{pre}}(I)$. For each $j \in \mathcal{J}$ we define

$$f_j : B \to \mathbb{R}_{\geq 0}, x \mapsto \sum_{C \in \mathcal{C}_I} C(j) \frac{x_C}{p_j}$$

the fraction of job j that is scheduled in a fractional schedule derived from x.

The algorithm by Grigoriadis et al. [6] computes $x \in B$ that satisfies $f_j(x) \geq (1-\rho)\lambda^*$ for each $j \in \mathcal{J}$ and a given ρ. We will choose $\rho \in \mathcal{O}(\varepsilon)$. The algorithm iterates the following steps: Given an initial solution $\check{x}$ the algorithm computes a price vector $q = q(\check{x}) \in \mathbb{R}^{|\mathcal{J}|}$ for the current value $\check{x}$. After that an $(1-\delta)$-approximative solution $\hat{x}$ of the problem

$$\max\{q^T f(x) | x \in B\}$$

has to be computed, where δ depends linear on ρ. This problem is called block-problem and a solver has to be provided. In the last step of an iteration the new value of the vector $\check{x}$ is set to $(1-\tau)\check{x} + \tau\hat{x} \in B$, where $\tau \in (0,1)$ is an appropriate step length. The algorithm needs $\mathcal{O}(|\mathcal{J}|(\ln(|\mathcal{J}|) + \rho^{-2}))$ of this iterations.

We now describe how to solve the block-problem $\max\{q^T f(x) | x \in B\}$. Note that

$$q^T f(x) = \sum_{C \in \mathcal{C}_I} x_C \sum_{j \in \mathcal{J}} \frac{q_j}{p_j} C(j).$$

Hence to solve the problem it suffices to compute a configuration C^* that maximizes $\sum_{j \in \mathcal{J}} \frac{q_j}{p_j} C(j)$ and set $x_{C^*} := 1$ and $x_C := 0$ for all $C \neq C^*$. So in each coordination step at most one new configuration is added to the solution. To find such a configuration C^* we have to solve the following integer linear program $\mathrm{ILP}_{kKP}(q, \mathcal{J}_N, \mathcal{J}_W, m, R)$:

$$\begin{aligned} \max \sum_{j \in \mathcal{J}} \frac{q_j}{p_j} a_j & \\ \sum_{j \in \mathcal{J}} a_j \leq m & \\ \sum_{j \in \mathcal{J}'} r_j a_j \leq R & \\ a_j \in \{0,1\} \quad \forall j \in \mathcal{J}_N & \\ a_j \in \{0, \ldots, 1/\varepsilon'\} \quad \forall j \in \mathcal{J}_W & \end{aligned}$$

This ILP is similar to the ILP formulation of the knapsack problem with cardinality constraint (kKP). The difference is that some items can be picked several times. To use the FPTAS described by Mastrolilli and Hutter [16] we have to duplicate each wide job $1/\varepsilon'$ times. The computation of a $(1+\delta)$-approximate solution takes $\mathcal{O}(|\mathcal{J}_W|/\varepsilon' + |\mathcal{J}_N| + m\delta^{-4})$ operations. Let a^* be the obtained solution to the ILP. To get a configuration C^* we define $C^*(j) := a^*_j$ for each $j \in \mathcal{J}$.

The next step is to scale x such that $f(x) \geq e$. By this scaling we have $\sum_{C \in \mathcal{C}_I} C(j) x_C \geq p_j$ for each $j \in \mathcal{J}$, which ensures that each job is scheduled. Further it is shown how to choose ρ such that $\sum_{C \in \mathcal{C}_I} x_C \leq (1+\varepsilon')\mathrm{OPT}_{\mathrm{pre}}(I)$ holds.

REMARK 2.1. *Let $x \in B$ with $f(x) \geq (1-\rho)\lambda^*$ and $\hat{\lambda} := \min\{f_j(x) | j \in \mathcal{J}\}$. It holds that*

$$\frac{1}{(1-\rho)} \mathrm{OPT}_{\mathrm{pre}}(I) \geq \sum_{C \in \mathcal{C}_I} \frac{1}{\hat{\lambda}} x_C$$

and we have $f(x/\hat{\lambda}) \geq e$.

To proof this remark consider a more general definition of λ^* and B. Let $t \in \mathbb{R}_{\geq 0}$ be a target makespan and:

$$\begin{aligned} B_t &:= \{x \in \mathbb{R}_{\geq 0}^{|\mathcal{C}_I|} | \sum_{C \in \mathcal{C}_I} x_C = t\} \\ \lambda^*_t &:= \max\{\lambda | \exists x \in B_t \forall j \leq M : f_j(x) \geq \lambda\} \end{aligned}$$

Now, let $t, t' \in \mathbb{R}_{\geq 0}$. By making use of the linearity of f it is easy to see that $B_t = t/t' B_{t'}$ and $\lambda^*_t = t/t' \lambda^*_{t'}$. This yields:

$$\frac{1}{\hat{\lambda}} \sum_{C \in \mathcal{C}_I} x_C \leq \frac{1}{1-\rho} \frac{1}{\lambda^*} \quad \leq \frac{1}{1-\rho} \mathrm{OPT}_{\mathrm{pre}}(I)$$

This concludes the proof of the remark.

So if we scale x by $1/\hat{\lambda}$ and set $\rho := \frac{\varepsilon'}{1+\varepsilon'} \in \mathcal{O}(\varepsilon)$, we get the desired properties for x. Since $f(x) \geq e$ holds, x fulfils equality 2.4 for each $j \in \mathcal{J}$. Hence x is a feasible solution to LP_{pre}.

Since $\rho \in \mathcal{O}(\varepsilon)$ we need

$$\mathcal{O}(n(\ln(n) + \varepsilon^{-2})(|\mathcal{J}_W|/\varepsilon + n + m/\varepsilon^4))$$

operations total, where $n = |\mathcal{J}|$.

The algorithm performed $\mathcal{O}(n(\ln(n) + \varepsilon^{-2}))$ coordination steps adding at most one configuration in every step. Therefore x_{pre} has at most this amount of non zero components. We reduce this number some more by computing

a basic solution to $\mathrm{LP}_{\mathrm{pre}}$ by using x_{pre} as a start vector. Beling and Megiddo [1] described how to compute a basic solution for the problem $Ax = b$, $x \geq 0$, given a start solution $\bar{x}$, where $A \in \mathbb{Q}^{m \times n}$. Later Ke et al. [12] presented a faster way of rectangular matrix multiplication. Combined it is possible to find a basic solution in $\mathcal{O}(m^{1.5356}n)$ time. Since our linear program is not in standard form, we have to add $|\mathcal{J}|$ variables and the equation $\sum_{C \in \mathcal{C}_I} x_C = \sum_{C \in \mathcal{C}_I} (x_{\mathrm{pre}})_C$. Further we use just the configurations, which have a non zero component in x_{pre}. Therefore we can compute a basic solution in $\mathcal{O}(n^{1.5356}(n(\ln(n)+\varepsilon^{-2}))) \leq \mathcal{O}(n^{2.5356}(\ln(n)+\varepsilon^{-2}))$.

In total we need

$$\mathcal{O}(n(\ln(n)+\varepsilon^{-2})(|\mathcal{J}_W|/\varepsilon + n^{1.5356} + m/\varepsilon^4))$$

operations to get a basic solution to the linear program.

□

We now look at the relation between optimal solutions to I and I_{sup}: Let a solution to $\mathrm{LP}_{\mathrm{pre}}(I)$ and $\mathrm{LP}_{\mathrm{pre}}(I_{\mathrm{sup}})$ each be given. Since each group $\mathcal{J}_{W,i}$ has the same summed up processing time, we can split that large job in I_{sup} with processing time $P(\mathcal{J}_{W,i})$ and resource amount R_i and schedule it instead of the jobs in $\mathcal{J}_{W,i+1}$ in the solution to $\mathrm{LP}_{\mathrm{pre}}(I)$. So for each solution to $\mathrm{LP}_{\mathrm{pre}}(I)$, we can generate a solution to $\mathrm{LP}_{\mathrm{pre}}(I_{\mathrm{sup}})$ with the same makespan, hence it holds that

$$\mathrm{OPT}_{\mathrm{pre}}(I_{\mathrm{sup}}) \leq \mathrm{OPT}_{\mathrm{pre}}(I).$$

Let x_{pre} be the $(1+\varepsilon')$-approximate feasible solution to $\mathrm{LP}_{\mathrm{pre}}(I_{\mathrm{sup}})$ with at most $|\mathcal{J}_{\mathrm{sup}}|+1$ non zero components. Since we have $|\mathcal{J}_{\mathrm{sup},W}| \leq \varepsilon'^{-2}$ we can generate this solution in $\mathcal{O}(n(\ln(n)+\varepsilon^{-2})(n^{1.5356}+m/\varepsilon^4))$ operations.

For any solution x_{pre} to $\mathrm{LP}_{\mathrm{pre}}(I')$ denote by $\mathcal{C}_{\mathrm{pre}} := \{C \in \mathcal{C}_{I'} | (x_{\mathrm{pre}})_C > 0\}$ the set of configurations with a non-zero component in x_{pre}, where $(x_{\mathrm{pre}})_C$ denotes the processing time of configuration C in x_{pre}. We define by $P_{\mathrm{pre}} := \sum_{C \in \mathcal{C}_{\mathrm{pre}}} (x_{\mathrm{pre}})_C$ the processing time of the preemptive schedule. Since the number of configurations in x_{pre} still depends on the input length, we have to give additional structure to this solution. For this purpose we use a special type of containers for the narrow jobs, called windows, which where first introduced by Epstein and Levin [3].

Handling the narrow jobs A *window* $w = (w_r, w_m)$ is a pair consisting of a resource amount $R(w) = w_r$ and a number of machines $m(w) = w_m$. In each window w there can be processed $m(w)$ jobs with summed up resource amount $R(w)$ at each point in time. For windows w_1, w_2 we write $w_1 \leq w_2$ iff $R(w_1) \leq R(w_2)$ and $m(w_1) \leq m(w_2)$.

We split each configuration into the part with wide jobs and the part with narrow jobs. For a given configuration $C \in \mathcal{C}_I$ we denote by $C|_{\mathcal{J}_W}$ the configuration consisting of all wide jobs in C. For a given set of configurations $\mathcal{C} \subseteq \mathcal{C}_I$ we define $\mathcal{C}_W := \{C|_{\mathcal{J}_W} | C \in \mathcal{C}\}$. Note that each configuration in $\mathcal{C}_W$ contains at most $1/\varepsilon'$ items, since each of the wide jobs needs at least $\varepsilon' R$ resource.

A *generalized configuration* (C, w) is a pair consisting of a configuration $C \in \mathcal{C}_W$ and a window w. (C, w) is valid for an instance I, if $m(w) \leq m - m(C)$ and $R(w) \leq R - R(C)$. For a configuration $C \in \mathcal{C}_W$ with $R(C) < R$ we define by $w(C) := (R - R(C), m - m(C))$ the *main window* for C. If $m(C) = m$ the main window is given by $w(C) = (0, 0)$.

Let $\mathcal{W}$ be any set of windows and $\mathcal{C}_W$ any set of configurations consisting just of wide jobs. In the following a linear program LP_W is presented. It describes the relation between generalized configurations and jobs assigned to those. This linear program was first introduced by Epstein and Levin [3] in a similar way.

$\mathrm{LP}_W(I, \mathcal{C}_W, \mathcal{W})$:

$$\sum_{C \in \mathcal{C}_W} \sum_{\substack{w \in \mathcal{W} \\ w \leq w(C)}} C(j) x_{(C,w)} \geq p_j, \quad \forall j \in \mathcal{J}_W \tag{2.6}$$

$$\sum_{w \in \mathcal{W}} y_{j,w} \geq p_j, \quad \forall j \in \mathcal{J}_N \tag{2.7}$$

$$m(w)\sum_{\substack{C\in\mathcal{C}_W\\ w(C)\geq w}} x_{(C,w)} \geq \sum_{j\in\mathcal{J}_N} y_{j,w}, \forall w \in \mathcal{W} \tag{2.8}$$

$$R(w)\sum_{\substack{C\in\mathcal{C}_W\\ w(C)\geq w}} x_{(C,w)} \geq \sum_{j\in\mathcal{J}_N} r_j y_{j,w}, \forall w \in \mathcal{W} \tag{2.9}$$

$$x_{(C,w)} \geq 0, \quad \forall C \in \mathcal{C}_W, \forall w \in \mathcal{W} \tag{2.10}$$

$$y_{j,w} \geq 0, \quad \forall w \in \mathcal{W}, \forall j \in \mathcal{J}_N \tag{2.11}$$

Again the variable $x_{(C,w)}$ denotes the processing time of the generalized configuration (C, w). The value $y_{j,w}$ indicates which amount of job j is processed in window w. Inequalities (2.6) and (2.7) ensure that for each job there is enough processing time reserved, while equalities (2.8) and (2.9) ensure that in each window there is enough space to schedule the contained jobs. Given a solution x to LP_W we define

$$P(x) := \sum_{C\in\mathcal{C}_W}\sum_{\substack{w\in\mathcal{W}\\ w\leq w(C)}} x_{(C,w)},$$

which is the makespan of x, and

$$P(w,x) := \sum_{\substack{C\in\mathcal{C}_W\\ C(w)\geq w}} x_{(C,w)}$$

which is the summed up processing time of a window $w \in \mathcal{W}$ in x.

LEMMA 2.2. *Let x_{pre} be the solution for $\mathrm{LP}_{\mathrm{pre}}(I)$ form lemma 2.1. Let $\mathcal{W}_{\mathrm{pre}} := \{w(C)|C \in \mathcal{C}_{\mathrm{pre},W}\}$. We can generate a solution $(\tilde{x}, \tilde{y})$ to $\mathrm{LP}_W(I, \mathcal{C}_{\mathrm{pre},W}, \mathcal{W}_{\mathrm{pre}})$, which fulfils*

$$P(\tilde{x}) = P_{\mathrm{pre}} \tag{2.12}$$

Proof. To generate this solution we simply look at each configuration $C \in \mathcal{C}_W$ and sum up the processing time of each configuration $C' \in \mathcal{C}_{\mathrm{pre}}$, which is reduced to C, meaning $C'|_{\mathcal{J}_W} = C$. To build a generalized configuration we have to combine C with a window from $\mathcal{W}_{\mathrm{pre}}$. We simply choose the main window. Hence we define

$$\tilde{x}_{(C,w(C))} := \sum_{\substack{C'\in\mathcal{C}_{\mathrm{pre}}\\ C'|_{\mathcal{J}_W}=C}} (x_{\mathrm{pre}})_{C'}.$$

Equality 2.12 holds for this choice for $\tilde{x}$, since there is no configuration $C' \in \mathcal{C}_{\mathrm{pre}}$ which processing time is added to more than one generalized configurations processing time and each configuration $C' \in \mathcal{C}_{\mathrm{pre}}$ has at least one generalized configuration, where its processing time is added to. With a similar argument one can see, that inequality 2.6 holds, since x_{pre} fulfils inequality 2.4.

Now we have to ensure that inequalities 2.7 to 2.9 hold. For this purpose we look at each configuration $C \in \mathcal{C}_{\mathrm{pre}}$ and consider the reduced configuration $C|_{\mathcal{J}_W}$ and its main window $w := w(C|_{\mathcal{J}_W})$. For each job $j \in \mathcal{J}_N$ we add its processing time in C, which is $C(j)(x_{\mathrm{pre}})_C$, to the window w. In total we get for each window $w \in W$ and each job $j \in \mathcal{J}_N$

$$\tilde{y}_{j,w} := \sum_{\substack{C\in\mathcal{C}_{\mathrm{pre}}\\ w(C|_{\mathcal{J}_W})=w}} C(j)(x_{\mathrm{pre}})_C.$$

Since the configuration C was valid and equalities 2.4 hold for x_{pre} the equalities 2.7 to 2.9 hold for $(\tilde{x}, \tilde{y})$. □

We define $P_{\mathrm{pre}}(C) := \tilde{x}_{C,w(C)}$ for each $C \in \mathcal{C}_W$ and $P_{\mathrm{pre}}(K) := \sum_{C\in K} P_{\mathrm{pre}}(C)$ for each $K \subseteq \mathcal{C}_W$.

LEMMA 2.3. *Given a solution $(\tilde{x}, \tilde{y})$ to $\mathrm{LP}_W(I, \mathcal{C}_W, \mathcal{W}_{\mathrm{pre}})$, we can find a set $\mathcal{W}' \subseteq \mathcal{W}_{\mathrm{pre}}$ with $|\mathcal{W}'| \leq \varepsilon'^{-2} + 2$ and a solution $(\bar{x}, \bar{y})$ to $\mathrm{LP}_W(I, \mathcal{C}_W, \mathcal{W}')$, which fulfils*

$$P(\bar{x}) \leq (1 + \varepsilon')P_{\mathrm{pre}} \tag{2.13}$$

and has at most $|\mathcal{J}_N| + |\mathcal{J}_W| + 2|\mathcal{W}| + 1$ non zero components in $\mathcal{O}((n + \varepsilon^{-2})^{1.5356} n\varepsilon^{-2})$ operations.

Proof. The steps to find $\mathcal{W}'$ and $(\bar{x}, \bar{y})$ are described in the following. The sets, configurations and sizes defined for these steps are marked in the figures 1 and 2.

To find the set $\mathcal{W}'$ we reduce the number of different resource amounts by a further grouping step. Since we just want to reduce the number of different resource amounts, we partition the set of generalised configurations by number of machines in the window. For each $i \in \{1, \ldots, m\}$ we define $K_i := \{C \in \mathcal{C}_W|m(C) = i\}$ to denote the set with all configurations using

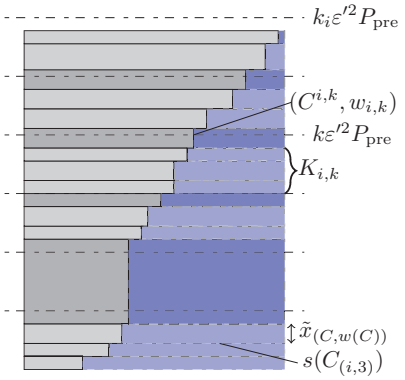

Figure 1: A stack of the generalized configurations in K_i. The grey rectangles represent the configurations and the blue rectangles represent the windows. The dashed lines are multiplies of $\varepsilon'^2 P_{\text{pre}}$.

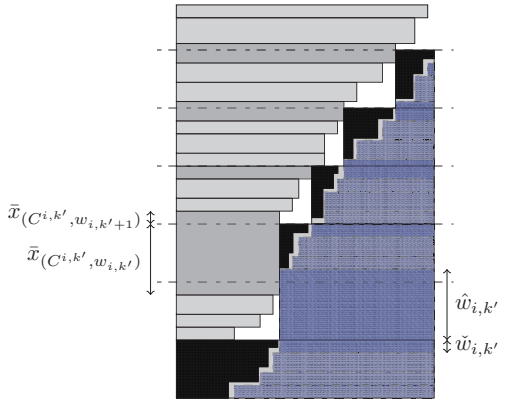

Figure 2: The same stack of the generalized configurations as in figure 1. But this time the windows are shifted $\varepsilon'^2 P_{\text{pre}}$ downwards. One can see that their area fits into the new windows (hatched rectangles).

exactly i machines. Since at most $1/\varepsilon'$ wide jobs can be part of one configuration, just the sets K_1 to $K_{1/\varepsilon'}$ are not empty.

For each of this sets we apply linear grouping: We number the configurations in K_i such that $R(C_{i,1}) \leq R(C_{i,2}) \leq R(C_{i,3}) \dots$ holds. Then we stack the configurations defining start positions $s(C_{i,1}) := 0$ and $s(C_{i,j}) := s(C_{i,j-1}) + P_{\text{pre}}(C_{i,j-1})$ for each $j > 1$. Furthermore we define a configurations end position $e(C_{i,j}) := s(C_{i,j+1})$.

The summed up height of all stacks is P_{pre}. We want to get about ε'^{-2} windows, so we have to split our stacks in ε'^{-2} pieces total. Therefore we consider in each stack the multiples of $\varepsilon'^2 P_{\text{pre}}$ to group the windows.

For each $0 < i \leq 1/\varepsilon'$ we define $k_i := \lceil P_{\text{pre}}(K_i)/(\varepsilon'^2 P_{\text{pre}}) \rceil$ which is the first multiple of $\varepsilon'^2 P_{\text{pre}}$ with no generalized configuration intersected with. For each $k \in \{1, \dots, k_i - 1\}$ there is a configuration C which intersects with $k\varepsilon'^2 P_{\text{pre}}$ meaning $s(C) < k\varepsilon'^2 P_{\text{pre}} \leq s(C) + P_{\text{pre}}(C)$. We denote this configuration by $C^{i,k}$ and define $w_{i,k} := w(C^{i,k})$ and $w_{i,k_i} := (0,0)$. Further we define the set $K_{i,k}$ as the set of configurations lying between the configurations $C^{i,k-1}$ and $C^{i,k}$. We select all windows intersected by a multiple of $\varepsilon'^2 P_{\text{pre}}$ for our set $\mathcal{W}'$. We define

$$\mathcal{W}_{K_i} := \{w(C^{i,k}) | k \in \{1, \dots, k_i - 1\}\}$$

the set of chosen windows from K_i. It holds that $|\mathcal{W}_{K_i}| \leq \lfloor P_{\text{pre}}(K_i)/(\varepsilon'^2 P_{\text{pre}}) \rfloor$. We define

$$\mathcal{W}' := \bigcup_{i=1}^{1/\varepsilon'} \mathcal{W}_{K_i} \cup \{(0,0), (R,m)\}$$

and have

$$|\mathcal{W}'| = 2 + \sum_{i=1}^{1/\varepsilon'} \left\lfloor \frac{P_{\text{pre}}(K_i)}{\varepsilon'^2 P_{\text{pre}}} \right\rfloor \leq 2 + \frac{1}{\varepsilon'^2}$$

After choosing the set of windows $\mathcal{W}'$ the next step is to generate a solution to the LP_W, which uses just windows in $\mathcal{W}'$ and needs not much further processing time. The crucial step is to shift the windows in each stack exactly $\varepsilon'^2 P_{\text{pre}}$ downwards.

Let us consider the configurations $C^{i,k}$ and $C^{i,k-1}$. The resource amount of $C^{i,k-1}$ is less or equal to the resource amount of $C^{i,k}$. So for each configuration $C \in K_{i,k}$ we have $w(C) \geq w_{i,k}$. If we shift the windows $\varepsilon'^2 P_{\text{pre}}$ downwards, the window $w_{i,k}$ is now processed alongside $C^{i,k-1}$. The windows above $w_{i,k}$ have a lesser resource amount. We now round up the resource amount

of the windows alongside the configurations in $C \in K_{i,k}$ such that they have the same resource amount as $w_{i,k}$. We therefore define for each $1 \leq 1/\varepsilon'$, for all $k \leq k_i$ and for each $C \in K_{i,k}$:

$$\bar{x}_{(C,w_{i,k})} := \tilde{x}_{(C,w(C))}$$

With configuration $C^{i,k}$ we have to be more careful. The lower part of this configuration should be alongside window $w_{i,k}$, the other alongside $w_{i,k+1}$. We have to find the biggest multiple of $\varepsilon'^2 P_{\text{pre}}$ the configuration $C^{i,k}$ was intersected by. In case $P(C^{i,k}) > \varepsilon'^2 P_{\text{pre}}$ the configuration $C^{i,k}$ will be intersected by more than one multiple of $\varepsilon'^2 P_{\text{pre}}$. The biggest multiple is defined by $\varepsilon'^2 P_{\text{pre}} \lfloor \frac{e(C^{i,k})}{\varepsilon'^2 P_{\text{pre}}} \rfloor$. This multiple defines the height at which the configuration is split. We define for each $i \leq 1/\varepsilon'$ and for each $k \leq k_i$:

$$\bar{x}_{(C^{i,k},w_{i,k})} := \varepsilon'^2 P_{\text{pre}} \left\lfloor \frac{e(C^{i,k})}{\varepsilon'^2 P_{\text{pre}}} \right\rfloor - s(C^{i,k})$$

$$\bar{x}_{(C^{i,k},w_{i,k+1})} := \tilde{x}_{(C^{i,k},w_{i,k})} - \bar{x}_{(C^{i,k},w_{i,k})}$$

Finally we need some extra space for the windows, which where shifted below the lowest configuration. In each stack this concerns window parts of total height $\varepsilon'^2 P_{\text{pre}}$. Since we have $1/\varepsilon'$ stacks we need $\varepsilon' P_{\text{pre}}$ extra space. The windows we need space for, are the widest windows in each stack. Therefore we round them up to the window (R, m). So the configuration $(\emptyset, (R, m))$ is the configuration, which needs extra space. So we define

$$\bar{x}_{(\emptyset,(R,m))} := \tilde{x}_{(\emptyset,(R.m))} + \varepsilon' P_{\text{pre}}.$$

In the following is described how to assign the narrow jobs to the round up windows. Lets consider a configuration $C \in K_{i,k+1}$. The window $w(C)$ was shifted down, such that it can be rounded up to $w_{i,k}$. To round this window up, we have to know which amount of it was processed alongside C. This amount is given by $\varphi_C := P_{\text{pre}}(C)/P_{\text{pre}}(w(C))$. So for each configuration $C \in K_{i,k+1}$ and each job $j \in \mathcal{J}_N$ we add $\varphi_C \tilde{y}_{j,w(C)}$ processing time to $\bar{y}_{j,w_{i,k}}$.

Next we consider a window $w_{i,k}$. In general this has to be split: as one can see in figure 2 the upper part of window $w_{i,k}$ stays in this window while the lower part is put into window $w_{i,k-1}$. This time we have to split the window at the smallest multiple of $\varepsilon'^2 P_{\text{pre}}$. We get this number by $\varepsilon'^2 P_{\text{pre}} \lceil s(C^{i,k})/\varepsilon'^2 P_{\text{pre}} \rceil$. Again we have to know which amount of window $w_{i,k}$ has to be processed where. Let $\check{w}_{i,k} = \varepsilon'^2 P_{\text{pre}} \lceil s(C^{i,k})/\varepsilon'^2 P_{\text{pre}} \rceil - s(C^{i,k})$ be the processing time of window $w_{i,k}$ which hast to be scheduled in the window $w_{i,k-1}$ and $\hat{w}_{i,k} = \tilde{x}_{(C^{i,k},w_{i,k})} - \check{w}_{i,k}$ the processing time of window $w_{i,k}$, which can stay in $w_{i,k}$. Furthermore we have to know which fraction of the total processing time of $w_{i,k}$ is presented by these two values. We therefore define $\hat{\varphi}_{i,k} := \hat{w}_{i,k}/P_{\text{pre}}(w_{i,k})$ and $\check{\varphi}_{i,k} := \check{w}_{i,k}/P_{\text{pre}}(w_{i,k})$. We now know that we have to add $\hat{\varphi}_{i,k}\tilde{y}_{j,w_{i,k}}$ to $\bar{y}_{j,w_{i,k}}$ and $\check{\varphi}_{i,k}\tilde{y}_{j,w_{i,k}}$ to $\bar{y}_{j,w_{i,k-1}}$. In total we have

$$\bar{y}_{j,w_{i,k}} := \sum_{C \in K_{i,(k+1)}} \varphi_C \tilde{y}_{j,w(C)} + \hat{\varphi}_{i,k}\tilde{y}_{j,w_{i,k}} + \check{\varphi}_{i,k+1}\tilde{y}_{j,w_{i,k+1}}$$

for each $i \leq 1/\varepsilon'$, $k \leq k_i$ and $j \in \mathcal{J}_N$.

We consider the window (R, m) separately. First we have to round the main windows from all configurations in $K_{i,1}$ up to (R, m). Meaning for each $i \leq 1/\varepsilon'$, $C \in K_{i,1}$ and $j \in \mathcal{J}_N$ we add $\varphi_C \tilde{y}_{j,w(C)}$ to $\bar{y}_{j,(R,m)}$. Furthermore we have to round up the lower part of window $w_{i,1}$ to (R, m). Additional jobs which where processed in window (R, m) before stay there. So in total we have

$$\bar{y}_{j,(R,m)} := \tilde{y}_{j,(R,m)} + \sum_{i=1}^{1/\varepsilon'} (\check{\varphi}_{i,1}\tilde{y}_{j,w_{i,1}} + \sum_{C \in K_{i,1}} \varphi_C \tilde{y}_{j,w(C)})$$

for all $j \in \mathcal{J}_N$.

$(\bar{x}, \bar{y})$ is a solution to $\text{LP}_W(\mathcal{W}')$. Using suitable data structures $(\tilde{x}, \tilde{y})$ as well as $(\bar{x}, \bar{y})$ can be computed in $\mathcal{O}((m + \log(n)/\varepsilon)n)$ operations.

This linear program has $|\mathcal{J}_N| + |\mathcal{J}_{\sup W}| + 2|\mathcal{W}|$ constraints and at most $|\mathcal{C}_W||\mathcal{W}| + |\mathcal{J}_N||\mathcal{W}| \in \mathcal{O}(|\mathcal{J}_{\sup}|/\varepsilon^2)$ variables. So we can compute a solution with at most $|\mathcal{J}_N| + |\mathcal{J}_{\sup W}| + 2|\mathcal{W}| + 1$ non zero components in $\mathcal{O}((n + \varepsilon^{-2})^{1.5356} n\varepsilon^{-2}) \leq \mathcal{O}(n^{2.5356}\varepsilon^{-8})$ operations. □

Let $(\bar{x}, \bar{y})$ be the solution form lemma 2.3 to $\mathrm{LP}_W(I_{\sup}, \mathcal{C}_W, \mathcal{W}')$. It can be generated in no more than $\mathcal{O}(n(\ln(n)+\varepsilon^{-2})(n^{1.5356}+m/\varepsilon^4))$ operations total and has at most $|\mathcal{J}_N|+3/\varepsilon'^2+4$ non zero components.

Integral Solution We now generate an integral schedule of the jobs in $\mathcal{J} \setminus \mathcal{J}_{W,G}$. The used technique to schedule the wide jobs is similar to the technique used by Kenyon and Rémila [14] to place the wide rectangles into their fractional packing of rectangles. To place the narrow jobs we use a similar argument as was used by Levin and Epstein in [3].

LEMMA 2.4. *Let $(\bar{x}, \bar{y})$ be the solution from lemma 2.3 to $\mathrm{LP}_W(I_{\sup}, \mathcal{C}_W, \mathcal{W}')$. There is an integral solution of jobs in $\mathcal{J} \setminus \mathcal{J}_{W,G}$ which has a makespan of at most*

$$(1+\varepsilon')(P(\bar{x}) + (5 + \frac{1}{\varepsilon'} + \frac{3}{\varepsilon'^2})p_{\max}).$$

First we modify the solution $(\bar{x}, \bar{y})$ to have enough space to schedule the jobs: We define $\hat{x}_{(C,w)} := \bar{x}_{(C,w)} + p_{\max}$ if $\bar{x}_{(C,w)} > 0$, and $\hat{x}_{(C,w)} = 0$ otherwise. We denote by $\mathcal{J}_{frac} \subseteq \mathcal{J}_N$ the set of fractional scheduled narrow jobs in $(\bar{x}, \bar{y})$. For each $j \in \mathcal{J}_{frac}$ and each window $w \in \mathcal{W}$ we set $\hat{y}_{j,w} := 0$ and $\hat{y}_{j,(R,m)} := p_j$. Further we add $P(\mathcal{J}_{frac}) \leq p_{\max}|\mathcal{J}_{frac}|$ to $\hat{x}_{(\emptyset,(R,m))}$. For every other $j \in \mathcal{J}_N \setminus \mathcal{J}_{frac}$ we set $\hat{y}_{j,w} = \bar{y}_{j,w}$. Since each job $j \in \mathcal{J}_N$ needs a non-zero component, we have at most $3/\varepsilon'^2 + 4$ configurations or fractional jobs. For each fractional job and for each configuration we add at most $p_{\max}$ processing time. Hence we have

$$P(\hat{x}) \leq P(\bar{x}) + (4 + 3/\varepsilon'^2)p_{\max}.$$

Next we sort the generalized configurations, such that configurations with the same window are scheduled consecutively and number them in ascending order. We iterate over each $i < G$ and all generalized configurations and fill the spaces for the job $i \in \mathcal{J}_{W,\sup}$, which has a resource amount of R_i, with jobs from $\mathcal{J}_{W,i} \cap \mathcal{J}_W$, till the height of the configuration, $\bar{x}_{(C,w)}$, is reached. Each last job in a configuration is allowed to overlap the border $\bar{x}_{(C,w)}$ since we have added $p_{\max}$ extra space. All jobs from $\mathcal{J}_{W,i} \cap \mathcal{J}_W$ can be placed in the configurations. Additional there is one generalized configuration (C, w) where the border $\bar{x}_{(C,w)}$ is not overlapped by a job from $\mathcal{J}_{W,i}$, since $\sum_{(C,w)\in\tilde{\mathcal{C}}} C(i)\bar{x}_{(C,w)} \geq P(\mathcal{J}_{W,i})$. In this particular configuration we place the job which was intersected by the multiple of $\varepsilon'^2 P_W$ in the first linear grouping and which fractions were put in $\mathcal{J}_{W,i}$ as well as in $\mathcal{J}_{W,i+1}$. Since one of its fractions was in $\mathcal{J}_{W,i}$ it fits into the reserved space.

To place the small jobs we consider each window $w \in \mathcal{W}$. Let $\mathcal{J}(w)$ be the set of jobs contained in w. We order the jobs in $\mathcal{J}(w)$ by decreasing resource amount. Since the generalized configurations containing window w are scheduled consecutively we can build stacks of jobs form $\mathcal{J}(w)$ with total processing time between $P(w, \bar{x})$ and $P(w, \bar{x}) + p_{\max}$ and schedule them in window w. Since $m(w)P(w, \bar{x}) \geq \sum_{j\in\mathcal{J}_N} \hat{y}_{j,w}$ we have to build at most $m(w)$ of this stacks. Because we have $m(w)$ free machines in each window there is a free machine for each stack.

Consider the jobs $\mathcal{J}_{top}$ intersecting the bound $P(w, \bar{x})$. These are the jobs with the lowest resource amount of their stack. Hence at each point in time the small jobs in the stacks have at least a total resource amount of $R(\mathcal{J}_{top})$ and therefore $P(w, \bar{x})R(\mathcal{J}_{top}) \leq \sum_{j\in J(w)} r_j p_j \leq P(w, \bar{x})R(w)$. So the jobs in $\mathcal{J}_{top}$ fit into the window w.

We remove the stack the with largest resource amount. Then the summed up resource amount of the jobs at the bottom of the stacks is less than $R(\mathcal{J}_{top})$ since each of this jobs was added right after a job from $\mathcal{J}_{top}$. Therefore at each point in time in window w it holds that the resource amount of small jobs is at most $R(\mathcal{J}_{top})$.

The stack we removed has a total processing time of at most $P(w, \bar{x}) + p_{\max}$. Since we remove this stack in each window, we have to place jobs with summed up processing time at most $\sum_{w\in\mathcal{W}\setminus\{(0,0),(R,m)\}}(P(w, \bar{x}) + p_{\max}) \leq P(\bar{x}) + \varepsilon'^{-2}p_{\max}$ at the end of the schedule. Since each of this jobs has a resource amount of at most $\varepsilon' R$, we can schedule $1/\varepsilon'$ of them at the same time. Hence we need an additional

processing time of $\varepsilon' P(\bar{x}) + (1+\varepsilon'^{-1})p_{\max}$. So far our schedule has a makespan of

$$\begin{aligned} &P(\hat{x}) + \varepsilon' P(\bar{x}) + (1+\varepsilon'^{-1})p_{\max} \\ &\leq (1+\varepsilon')P(\bar{x}) + (5 + \frac{1}{\varepsilon'} + \frac{3}{\varepsilon'^2})p_{\max} \\ &\leq (1+\varepsilon')^3 \mathrm{OPT}_{\mathrm{pre}}(I) + \mathcal{O}(1/\varepsilon^2)p_{\max}. \end{aligned}$$

The last step is to add the jobs from $\mathcal{J}_{W,G}$ to the schedule. We have $P(\mathcal{J}_{W,G}) \leq \varepsilon' OPT_{\mathrm{pre}}(I)$, since $P(\mathcal{J}_{W,G}) \leq \varepsilon'^2 P(\mathcal{J}_W)$ and $\varepsilon' P(\mathcal{J}_W) \leq OPT_{\mathrm{pre}}(I)$, because at most $1/\varepsilon'$ jobs from $\mathcal{J}_W$ can be scheduled at the same point in time. So if we schedule these jobs one after another at the end of the schedule, we get an integral schedule for I with makespan at most

$$\begin{aligned} &((1+\varepsilon')^3 + \varepsilon')\mathrm{OPT}_{\mathrm{pre}}(I) + \mathcal{O}(1/\varepsilon^2)p_{\max} \\ \leq &(1+\varepsilon)\mathrm{OPT}(I) + \mathcal{O}(1/\varepsilon^2)p_{\max} \end{aligned}$$

The number of operations to build the integral schedule is dominated by the number of operations to build the fractional schedule, which is $\mathcal{O}(n(\ln(n) + \varepsilon^{-2})(n^{1.5356} + m\varepsilon^{-4}))$. Since each step we used to manipulate x_{pre} was dominated by this number, this is our total processing time.

Note that we stack jobs in container, which resources can be allocated contiguously. So this schedule is feasible for instances, where contiguous resource is required, too.

2.2 Second Case: $m \leq 1/\varepsilon$.

Let $\varepsilon > 0$ and an Instance $I = (\mathcal{J}, m, R)$, with $1/\varepsilon \geq m$ be given. In this case we do not partition the set of jobs into wide and narrow jobs. We apply the linear grouping to all jobs. Again we get $G \leq 1/\varepsilon'^2$ groups $\mathcal{J}_i$. We denote by $\mathcal{J}_{\mathrm{sup}}$ the set of jobs, which contains for each $i < G$ one job with processing time $P(\mathcal{J}_i)$ and resource amount $\max\{r_j | j \in \mathcal{J}_i\}$. We denote by $\mathcal{C}_{\mathrm{sup}}$ the set of valid configurations of jobs in $\mathcal{J}_{\mathrm{sup}}$. We denote by $I_{\mathrm{sup}} := (\mathcal{J}_{\mathrm{sup}}, m, R)$ the round up instance. To get a preemptive schedule for I_{sup} we have to solve $\mathrm{LP}_{\mathrm{pre}}(I_{\mathrm{sup}})$, while interpreting each job in $\mathcal{J}_{\mathrm{sup}}$ as a wide job. By lemma 2.1 we get a solution x_{pre} with $\mathcal{O}(|\mathcal{J}_{\mathrm{sup}}|(\ln(|\mathcal{J}_{\mathrm{sup}}|)+\varepsilon'^{-2})) = \mathcal{O}(\varepsilon'^{-4})$ non-zero components and $\sum_{C \in \mathcal{C}_{\mathrm{sup}}} (x_{\mathrm{pre}})_C \leq (1+\varepsilon')\mathrm{OPT}_{\mathrm{pre}}$ in $\mathcal{O}(|\mathcal{J}_{\mathrm{sup}}|(\ln(|\mathcal{J}_{\mathrm{sup}}|)+\varepsilon^{-2})(|\mathcal{J}_{\mathrm{sup}}| + m/\varepsilon^4))$ operations .

Since we have now a polynomial number of variables, we can construct a solution $\tilde{x}$, which has at most $1/\varepsilon'^2$ non zero components and for which it holds that $\sum_{C \in \mathcal{C}_{\mathrm{sup}}} (x_{\mathrm{pre}})_C = \sum_{C \in \mathcal{C}_{\mathrm{sup}}} \tilde{x}_C$, in $\mathcal{O}(|\mathcal{J}_{\mathrm{sup}}|^{1.5356}\varepsilon'^{-4})$ operations.

To each of this components we add $p_{\max}$ processing time. Now we place the jobs from $\mathcal{J}_i$ for each $i < G$ in the configurations as we placed the wide jobs in the first case. We get a schedule of length at most $(1+\varepsilon')\mathrm{OPT}_{\mathrm{pre}} + \varepsilon'^{-2}p_{\max}$. The jobs in $\mathcal{J}_G$ are added in the end of the schedule. These jobs have a total makespan of at most $\varepsilon'\mathrm{OPT}_{\mathrm{pre}}(I)$, since $P(\mathcal{J}_G) \leq \varepsilon'^2 P(\mathcal{J})$ and $\mathrm{OPT}_{\mathrm{pre}}(I) \geq P(\mathcal{J})/m \geq \varepsilon' P(\mathcal{J})$. So in the end we have a schedule with a total makespan of at most

$$\begin{aligned} &(1+\varepsilon')\mathrm{OPT}_{\mathrm{pre}}(I) + \varepsilon'\mathrm{OPT}_{\mathrm{pre}}(I) + 1/\varepsilon'^2 p_{\max} \\ &\leq (1+\varepsilon)\mathrm{OPT}(I) + \mathcal{O}(1/\varepsilon^2)p_{\max}. \end{aligned}$$

So in both cases we have an algorithm which has the required properties to proof theorem 2.1.

3 Scheduling with resource dependent processing times

An instance $\tilde{I}$ for scheduling with resource dependent processing times is given by a set of n jobs $\tilde{\mathcal{J}}$, a number of machines m, a resource bound $R \in \mathbb{Z}_{\geq 0}$, and a set $D \subseteq [R]$. Furthermore for each job $j \in \tilde{\mathcal{J}}$ there is a processing time function $\pi_j : D \to \mathbb{Q}_{>0} \cup \{\infty\}$. The goal is to find for each job j a resource assignment $\rho_j \in D$, and a starting time $t_j \in \mathbb{N}$, such that

$$\begin{aligned} \forall t \in \mathbb{N} : \sum_{j : t \in [t_j, t_j + \pi_j(\rho_j))} \rho_j \leq R, \\ \forall t \in \mathbb{N} : \sum_{j : t \in [t_j, t_j + \pi_j(\rho_j))} 1 \leq m, \end{aligned}$$

minimizing the makespan $\max_{j \in \mathcal{J}}(t_j + \pi_j(\rho_j))$. We set $\pi_{\max} := \max\{\pi_j(\rho) | \forall j \in \tilde{\mathcal{J}}, \rho \in D : \pi_j(\rho) < \infty\}$ and denote the makespan of an optimal schedule by $\mathrm{OPT}(\tilde{I})$. Our main result in this section is an AFPTAS for the resource dependent scheduling problem:

THEOREM 3.1. *There is an algorithm which, given an instance $\tilde{I}$ for scheduling with resource dependent processing times and a positive number $\varepsilon > 0$, produces a schedule with makespan at most:*

$$(1+\varepsilon)\mathrm{OPT}(\tilde{I}) + O(1/\varepsilon^2)\pi_{\max}$$

The running time of the algorithm is polynomially bounded in n, m, $|D|$ and $1/\varepsilon$.

W.l.o.g. we may assume that $m \leq n$, because otherwise we have the problem of scheduling malleable parallel tasks, for which an appropriate AFPTAS is already known [11].

The basic idea of the algorithm is to find a resource allotment for the jobs and to use the AFPTAS for the fixed-resource variant. Finding a resource allotment that allows analysis is the main difficulty here. W.l.o.g. we assume that $1/\varepsilon \in \mathbb{Z}_{>0}$ and set $\varepsilon' := \varepsilon/8$. Like in the fixed-resource variant, the AFPTAS works differently in the two cases $1/\varepsilon < m$ and $1/\varepsilon \geq m$, where the second case is much simpler. We give a brief overview of the algorithm for the first case, followed by a detailed description and analysis for both cases, and lastly some concluding remarks about the scope of the algorithm.

Algorithm Given an instance $\tilde{I}$ and $\varepsilon > 1/m$ the algorithm can be summarized as follows:

(i) Compute via max-min resource sharing a preemptive schedule with length at most $(1+\varepsilon')\mathrm{OPT}_{\mathrm{pre}}(\tilde{I})$.
(ii) Using the preemptive schedule, define an instance I for the fixed-resource variant, for which $\mathrm{OPT}_{\mathrm{pre}}(I) \leq (1+\varepsilon')\mathrm{OPT}_{\mathrm{pre}}(\tilde{I})$ holds.
(iii) Apply the steps (i) to (v) of the fixed-resource AFPTAS, yielding a solution $(\bar{x}, \bar{y})$ for the window LP for the rounded instance I', that uses at most $1/\varepsilon'^2 + 2$ windows.
(iv) Compute a unique resource allotment for all but at most $3/\varepsilon'^2 + 4$ of the original jobs in $\tilde{\mathcal{J}}$, which is in some sense compatible with both the linear grouping for I and the windows.
(v) Use the unique resource allotment to define a new instance for the fixed-resource variant and a new solution $(\breve{x}, \breve{y})$ of the window LP that has the same length as $(\bar{x}, \bar{y})$.
(vi) Apply the steps (vi) to (vii) of the fixed-resource AFPTAS, yielding a schedule for almost all of the jobs.
(vii) Put the jobs that did not get a unique resource allotment on top of the schedule using at most $(3/\varepsilon'^2 + 4)\pi_{\max}$ extra height.

Preemptive schedule Unlike before, here we mean by *preemptive*, a schedule where the jobs can be interrupted at any time at no cost and restarted *later*, possibly on another processor with a different resource-allotment. We denote the length of an optimal preemptive schedule for $\tilde{I}$ by $\mathrm{OPT}_{\mathrm{pre}}(\tilde{I})$. Like before for a given instance I a configuration $C := \{j : \rho_{C,j} | j \in \tilde{\mathcal{J}}\}$ is a multiset of jobs from $\tilde{\mathcal{J}}$. However, in this context we will interpret the multiplicity of a job as the number of resource-units that are assigned to it. A configuration C will be called *valid*, if $\sum_{j:\rho_{C,j}>0} 1 \leq m$, $\sum_{j\in\tilde{\mathcal{J}}} \rho_{C,j} \leq R$, and $\rho_{C,j} \in D \cup \{0\}$ for each $j \in \tilde{\mathcal{J}}$. Note that $\rho_{C,j} = 0$ corresponds to the case that the job j is not a part of the configuration. The set of valid configurations is denoted by $\tilde{\mathcal{C}}$.

The computation of the schedule is done via the following LP:

$$\min \sum_{C\in\tilde{\mathcal{C}}} x_C$$

$$\sum_{\rho\in D} \frac{1}{\pi_j(\rho)} \sum_{C:\rho_{C,j}=\rho} x_C \geq 1 \qquad \forall j \in \tilde{\mathcal{J}}$$

$$x_C \geq 0 \qquad \forall C \in \tilde{\mathcal{C}}$$

In the following this LP is denoted by $\mathrm{LP}_{\mathrm{pre}}(\tilde{I})$. It is easy to see, that a solution of $\mathrm{LP}_{\mathrm{pre}}(\tilde{I})$ corresponds to a preemptive schedule and vice-versa. Note that $\mathrm{LP}_{\mathrm{pre}}(\tilde{I})$ is closely related to the LP used to find a preemptive solution for the fixed-resource variant, and indeed we will follow the same approach to find an approximate solution as was done in lemma 2.1.

LEMMA 3.1. *A solution x to $\mathrm{LP}_{\mathrm{pre}}(\tilde{I})$ that satisfies $\sum_{C\in\tilde{\mathcal{C}}} x_C \leq (1+\varepsilon')\mathrm{OPT}_{\mathrm{pre}}(\tilde{I})$ can be found in time $\mathcal{O}(n^3m^2|D|\varepsilon^{-1}(\varepsilon^{-2} + \ln n))$.*

Furthermore a solution with the same objective value and at most $n+1$ non zero components can be found using $\mathcal{O}(n^{2.5356}(\varepsilon^{-2} + \ln n)$ operations.

Proof. We present the adaptations that have to be made in the proof of lemma 2.1. First we define a (non-empty convex compact) set:

$$B := \Big\{(x_C)_{C\in\tilde{\mathcal{C}}}\Big| \sum_{C\in\tilde{\mathcal{C}}} x_C = 1,\, x_C \geq 0,\, C \in \tilde{\mathcal{C}}\Big\}$$

For each $j \in \tilde{\mathcal{J}}$ we define a non-negative linear function:

$$f_j : \mathbb{R}^{|\tilde{\mathcal{C}}|} \to \mathbb{R}_{>0}, x \mapsto \sum_{\rho\in D} \frac{1}{\pi_j(\rho)} \sum_{C:\rho_{C,j}=\rho} x_C$$

Moreover we set:

$$\lambda^* = \max\{\lambda | f_j(x) \geq \lambda,\, j \in \tilde{\mathcal{J}},\, x \in B\}$$

The algorithm by Grigoriades et al. [6] can be applied to find an $x \in B$ that satisfies $f_j(x) \geq (1-\gamma)\lambda^*$ for each $j \in \tilde{\mathcal{J}}$. In each iteration of the algorithm a price vector $q \in \mathbb{R}^n$ is obtained and the following block problem has to be solved approximately with an accuracy γ' that depends linear on γ:

$$\max\Big\{\sum_{j\in\tilde{\mathcal{J}}} q_j f_j(x)\Big| x \in B\Big\}$$

It can be easily seen that an optimum to this problem is obtained at a point $x \in B$ with $x_{C^*} = 1$ for a single configuration C^*, and $x_C = 0$ for $C \neq C^*$. The problem to find such a configuration can be described by the following ILP:

$$\text{(3.14)}\quad \max \sum_{j\in\tilde{\mathcal{J}}}\sum_{\rho\in D} \frac{q_j}{\pi_j(\rho)} x_{j,\rho}$$

$$\text{(3.15)}\quad \sum_{j\in\tilde{\mathcal{J}}}\sum_{\rho\in D} \rho x_{j,\rho} \leq R$$

$$\text{(3.16)}\quad \sum_{j\in\tilde{\mathcal{J}}}\sum_{\rho\in D} x_{j,\rho} \leq m$$

$$\text{(3.17)}\quad \sum_{\rho\in D} x_{j,\rho} \leq 1 \qquad \forall j \in \tilde{\mathcal{J}}$$

$$\text{(3.18)}\quad x_{j,\rho} \in \{0,1\} \qquad \forall \rho \in D,\, j \in \tilde{\mathcal{J}}$$

The variables $x_{j,\rho}$ express, whether ρ units of the resource are assigned to a job j in the configuration. The constraint (3.16) and (3.15) guarantee that there are at most m jobs in the configuration, with summed up resource allocation at most R; and due to (3.17) every job is scheduled at most once. This problem can be seen as a Multiple-Choice Knapsack Problem with an additional capacity constraint (kMCKP): A variant of the knapsack problem where the items are partitioned into equivalence classes, and only one item from every class may be packed. Moreover, the capacity constraint bounds the *number* of items that can be packed. A naive application of basic techniques by Lawler [15] provides an FPTAS for this problem with running time $\mathcal{O}(tk^3\delta^{-1})$, where t is the number of items to be packed, k the maximum number of items that may be packed, and δ the accuracy. Applied to this case we get a running time of $\mathcal{O}(n|D|m^3\gamma'^{-1})$.

Scaling x and setting $\gamma = \varepsilon'/(1+\varepsilon')$ we get a solution with the required makespan after $\mathcal{O}(n(\varepsilon^{-2} + \ln n))$ iterations. In each iteration there is a numerical overhead of $\mathcal{O}(n \ln\ln(n\gamma^{-1}))$ and the knapsack problem has to be solved. The running time of the FPTAS can be bounded by $\mathcal{O}(n^2|D|m^2\gamma'^{-1})$, because $m \leq n$, and therefore the overhead can be ignored. Since $\varepsilon'/2 \leq \gamma \leq \varepsilon'$ and $\varepsilon' = \varepsilon/8$, we get the asserted running time. We turn this solution into a basic feasible one in time $\mathcal{O}(n^{2.5356}(\varepsilon^{-2} + \ln n)$. □

Fixed-resource instance Based on the preemptive schedule we now define an instance I of the problem with fixed resources. Let $x_{(j,\rho)} := 1/\pi_j(\rho)\sum_{C:\rho_{C,j}=\rho} x_C$ be the fraction of job $j \in \tilde{\mathcal{J}}$ that is scheduled with a resource amount of $\rho \in D$ according to the preemptive solution. W.l.o.g. we may assume that:

$$\text{(3.19)}\qquad \forall j \in \tilde{\mathcal{J}} : \sum_{\rho\in D} x_{(j,\rho)} = 1$$

Since, if the left side should be larger than one, we can just scale down the $x_{(j,\rho)}$ values. The new set of jobs is defined as $\mathcal{J} = \{(j,\rho) | j \in \tilde{\mathcal{J}}, \rho \in D, x_{j,\rho} > 0\}$ with processing time $p_{(j,\rho)} := x_{(j,\rho)}\pi_j(\rho)$ and resource requirement $r_{(j,\rho)} = \rho$. The machine set and the resource bound stay the same.

This instance I has at most $(n+1)m$ jobs, because in the preemptive solution at most $n+1$

configurations are used, each containing at most m jobs. Furthermore note that:

$$\begin{aligned}\mathrm{OPT}_{\mathrm{pre}}(I) &\leq (1+\varepsilon')\mathrm{OPT}_{\mathrm{pre}}(\tilde{I}) \\ &\leq (1+\varepsilon')\mathrm{OPT}(\tilde{I})\end{aligned} \tag{3.20}$$

The first inequality holds, because (x_C) yields a solution for the preemptive version of the fixed-resource problem, and the second is obvious.

From here on we have to differentiate the two cases $1/\varepsilon < m$ and $1/\varepsilon \geq m$. In both cases we will make use of results for the fixed-resource problem. However, the second case is much simpler since a unique resource allotment can be found prior to the application of the AFPTAS for the fixed-resource variant. Therefore, the whole algorithm can be applied, while in the first case some steps are conducted before finding a unique resource allotment for all the jobs, and some afterwards.

First case: $1/\varepsilon < m$ For instance I and ε' we now use the steps (i) to (v) of the AFPTAS for the fixed-resource problem. By application of linear grouping the set of jobs is divided into wide $\mathcal{J}_W$ and narrow jobs $\mathcal{J}_N$. The wide jobs are split up into $G = 1/\varepsilon'^2$ groups $\mathcal{J}_{W,i}$. It may happen that jobs that are part of multiple groups are split up correspondingly. We formally handle this by introducing a factor $\delta_{(j,\rho),i}$ that denotes the fraction of a wide job $(j,\rho) \in \mathcal{J}_W$ that lies in the i'th group in the linear grouping stack. In the next step a modified instance $I_{\mathrm{sup}} = (\mathcal{J}_{\mathrm{sup},W} \cup \mathcal{J}_N, m, R)$ with replaced wide jobs is formed, for which lemma 2.1, and subsequently lemma 2.3 can be applied. Summarizing we get:

LEMMA 3.2. *We can obtain a set of windows $\mathcal{W}'$, a set of configurations $\mathcal{C}_W$ and a basic feasible solution $(\bar{x},\bar{y})$ to $\mathrm{LP}_W(I_{\mathrm{sup}},\mathcal{C}_W,\mathcal{W}')$ with the following properties:*

$$|\mathcal{W}'| \leq 1/\varepsilon'^2 + 2 \tag{3.21}$$

$$P(\bar{x}) \leq (1+\varepsilon')^2\mathrm{OPT}_{\mathrm{pre}}(I_{\mathrm{sup}}) \tag{3.22}$$

The running time can be bounded by $\mathcal{O}(nm(\ln(nm)+\varepsilon^{-2})((nm)^{1.5356}+m\varepsilon^{-4}))$.

Proof. The running time is due to lemma 2.1 and $|\mathcal{J}| \leq (n+1)m$. All the remaining claims follow from lemma 2.3. □

Unique resource allotment The goal in this section is to find a unique resource allocation r_j for almost all the jobs $j \in \tilde{\mathcal{J}}$ of the original instance.

First case: $1/\varepsilon < m$ Using a linear program, we will find a new instance for the fixed-resource variant, in which the wide jobs still fit into the linear grouping stack given by I, and the narrow jobs fit into the windows given by $(\bar{x},\bar{y})$. We set $u_{(j,\rho),i} := \delta_{(j,\rho),i}x_{(j,\rho)}$ for each $i \in [G]$ and $(j,\rho) \in \mathcal{J}_{W,i}$; and $v_{(j,\rho),w} := (\pi_j(\rho))^{-1}\bar{y}_{(j,\rho),w}$ for each $(j,\rho) \in \mathcal{J}_N$ and $w \in \mathcal{W}'$. This yields a solution to the following LP:

$$\begin{aligned}&\sum_{(j,\rho)\in\mathcal{J}_{W,i}} \pi_j(\rho)u_{(j,\rho),i} \\ &\quad\leq\ P(\mathcal{J}_{W,i}) \qquad\qquad \forall i \in [G]\end{aligned} \tag{3.23}$$

$$\begin{aligned}&\sum_{(j,\rho)\in\mathcal{J}_N} \pi_j(\rho)v_{(j,\rho),w} \\ &\quad\leq\ m(w)\sum_{C\in C(w)} \bar{x}_{C,w} \qquad\qquad \forall w \in \mathcal{W}'\end{aligned} \tag{3.24}$$

$$\begin{aligned}&\sum_{(j,\rho)\in\mathcal{J}_N} r_{(j,\rho)}\pi_j(\rho)v_{(j,\rho),w} \\ &\quad\leq\ R(w)\sum_{C\in C(w)} \bar{x}_{C,w} \qquad\qquad \forall w \in \mathcal{W}'\end{aligned} \tag{3.25}$$

$$\begin{aligned}&\sum_{i\in[G]}\sum_{(j,\rho)\in\mathcal{J}_{W,i}} u_{(j,\rho),i} \\ &\quad+\sum_{w\in\mathcal{W}'}\sum_{(j,\rho)\in\mathcal{J}_N} v_{(j,\rho),w} = 1 \qquad\qquad \forall j \in \tilde{\mathcal{J}}\end{aligned} \tag{3.26}$$

$$u_{(j,\rho),i} \geq 0 \qquad \forall i \in [G], (j,\rho) \in \mathcal{J}_{W,i} \tag{3.27}$$

$$v_{(j,\rho),w} \geq 0 \qquad \forall w \in \mathcal{W}', \forall (j,\rho) \in \mathcal{J}_N \tag{3.28}$$

The inequality (3.23) holds due to the definition of the linear grouping, since $\pi_j(\rho)u_{(j,\rho),i} = \delta_{(j,\rho)}p_{(j,\rho)}$ is exactly the height that job (j,ρ) contributes to the height $P(\mathcal{J}_{W,i})$ of group i. The next two constraints (3.24) and (3.25) are satisfied, because $\pi_j(\rho)v_{(j,\rho),w} = \bar{y}_{(j,\rho),w}$, and $(\bar{x},\bar{y})$ satisfies the constraints (2.8) and (2.9) of LP_W. Furthermore it holds that:

$$\sum_{i\in[G]}\sum_{(j,\rho)\in\mathcal{J}_{W,i}} u_{(j,\rho),i} + \sum_{w\in\mathcal{W}'}\sum_{(j,\rho)\in\mathcal{J}_N} v_{(j,\rho),w}$$

$$
\begin{aligned}
&= \sum_{(j,\rho)\in\mathcal{J}_W} x_{j,\rho} \sum_{i\in[G]} \delta_{(j,\rho),i} \\
&\qquad + \sum_{(j,\rho)\in\mathcal{J}_N} (\pi_j(\rho))^{-1} \sum_{w\in\mathcal{W}'} \bar{y}_{(j,\rho),w} \\
&= \sum_{(j,\rho)\in\mathcal{J}_W} x_{j,\rho} + \sum_{(j,\rho)\in\mathcal{J}_N} (\pi_j(\rho))^{-1} p_{(j,\rho)} \\
&= \sum_{(j,\rho)\in\mathcal{J}_W} x_{j,\rho} + \sum_{(j,\rho)\in\mathcal{J}_N} (\pi_j(\rho))^{-1} x_{j,\rho}\pi_j(\rho) \\
&= \sum_{(j,\rho)\in\mathcal{J}_W} x_{j,\rho} + \sum_{(j,\rho)\in\mathcal{J}_N} x_{j,\rho} \\
&= \sum_{(j,\rho)\in\mathcal{J}} x_{j,\rho} \underset{(3.19)}{=} 1
\end{aligned}
$$

This yields (3.26).

Next we convert (u, v) into a basic feasible solution for the LP. Now (u, v) has at most $G+2|\mathcal{W}'|+|\tilde{\mathcal{J}}|$ non-zero variables. Furthermore, due to (3.26) and a simple counting argument there are at most $G + 2|\mathcal{W}'| \leq 3/\varepsilon'^2 + 4$ jobs $j \in \tilde{\mathcal{J}}$ with more then one non-zero variable from the set $\{u_{(j,\rho),i}, v_{(j,\rho),w} | \rho \in D, i \in [G], w \in \mathcal{W}'\}$ of variables related to j. We will schedule the jobs with more then one non-zero variable separately in the end. The rest of the jobs $\bar{\mathcal{J}}$ yield a new instance $\bar{I}$ for the fixed-resource variant. For $j \in \bar{\mathcal{J}}$ exactly one of the following holds:

$$
\begin{aligned}
\exists \rho \in D,\, i \in [G]: &\qquad u_{(j,\rho),i} = 1 \\
\exists \rho \in D,\, w \in \mathcal{W}': &\qquad v_{(j,\rho),w} = 1
\end{aligned}
$$

In the first case j is a wide job $j \in \bar{\mathcal{J}}_W$ and in the second it is narrow $j \in \bar{\mathcal{J}}_N$, while in both cases the processing time and resource requirement are given by $p_j := \pi_j(\rho)$ and $r_j := \rho$. Moreover, the wide jobs are uniquely assigned to groups $\bar{\mathcal{J}}_{W,i}$.

We define a modified rounded instance $\bar{I}_{\text{sup}} := (\mathcal{J}_{\text{sup},W} \cup \bar{\mathcal{J}}_N, m, R)$ for $\bar{I}$ using the old rounded wide jobs and the new narrow jobs. Furthermore, let $\breve{x}_{C,w} := \bar{x}_{C,w}$ for each $C \in \mathcal{C}_W$, and for each $j \in \bar{\mathcal{J}}_N$ let:

$$
\breve{y}_{j,w} := \begin{cases} p_j & \text{, if } v_{(j,\rho),w} = 1 \\ 0 & \text{, otherwise} \end{cases}
$$

LEMMA 3.3. *It holds that $(\breve{x}, \breve{y})$ is a solution to $\text{LP}_W(\bar{I}_{\text{sup}}, \mathcal{C}_W, \mathcal{W}')$ that has the the same makespan as $(\bar{x}, \bar{y})$, and to which lemma 2.4 can be applied. It can be computed using $\mathcal{O}(nm\varepsilon^{-2}(nm + \varepsilon^{-2})^{1.5356})$ operations.*

Proof. The generalized configurations keep the same height, hence the makespan stays the same, and, since also the same rounded wide jobs are used, the constraint (2.6) of LP_W holds. The constraints (2.8) and (2.9) are satisfied, because v satisfies (3.24) and (3.25), while constraint (2.7) holds by the definition of $\breve{y}$.

Because of (3.23) it holds that $P(\bar{\mathcal{J}}_{W,i}) \leq P(\mathcal{J}_{W,i})$ for each $i \in [G]$, i.e. the new wide jobs still fit into the linear grouping stack and can therefore be scheduled by the fixed-resource AFPTAS. Also the narrow jobs can be handled, because (2.8) and (2.9) are satisfied.

The running time is dominated by the computation of the basic feasible solution. Since the LP has at most $\mathcal{O}((nm + \varepsilon^{-2}))$ constraints and $\mathcal{O}(nm\varepsilon^{-2})$ variables, this can be done in time $\mathcal{O}(nm\varepsilon^{-2}(nm + \varepsilon^{-2})^{1.5356})$. □

Second case: $1/\varepsilon \geq m$ In this case it is possible to treat all the jobs as wide jobs. We apply linear grouping and define u as above. This yields a solution for the LP that is given by (3.23), (3.27) and:

$$
\sum_{i\in[G]} \sum_{(j,\rho)\in\mathcal{J}_{W,i}} u_{(j,\rho),i} = 1 \qquad \forall j \in \tilde{\mathcal{J}} \tag{3.29}
$$

We transform u into a basic feasible solution and due to the same counting argument there are at most $G = 1/\varepsilon'^2$ fractional variables. Next, the fractional jobs are removed and the wide jobs defined as above. Due to (3.23) they fit into the linear grouping stack.

The integral schedule We can now apply the last steps of the fixed-resource AFPTAS.

First case: $1/\varepsilon < m$ We can apply (the proof of) lemma 2.4 to $(\breve{x}, \breve{y})$ yielding an integral schedule for almost all the jobs with makespan at most:

$$
((1+\varepsilon')^3 + \varepsilon')\text{OPT}_{\text{pre}}(I) + (5 + \frac{1}{\varepsilon'} + \frac{3}{\varepsilon'^2})p_{\max}
$$

Note that $p_{\max} \leq \pi_{\max}$ holds. We schedule the remaining jobs $\tilde{\mathcal{J}} \setminus \bar{\mathcal{J}}$ successively at the end

of the schedule, each with an optimal resource assignment. In doing so we add at most

$$|\tilde{\mathcal{J}} \setminus \bar{\mathcal{J}}| \max_{j \in \tilde{\mathcal{J}}} \min_{\rho \in D} \pi_j(\rho) \leq (4 + 3/\varepsilon'^2)\pi_{\max}$$

length to the schedule. Summing up and applying (3.20) we get a makespan of at most:

$$\begin{aligned}&((1+\varepsilon')^3 + \varepsilon')(1+\varepsilon')\mathrm{OPT}_{\mathrm{pre}}(\tilde{I}) \\ &\quad + (9 + \frac{1}{\varepsilon'} + \frac{6}{\varepsilon'^2})\pi_{\max} \\ \leq &(1+\varepsilon)\mathrm{OPT}(\tilde{I}) + \mathcal{O}(1/\varepsilon^2)\pi_{max}\end{aligned}$$

The overall running time is polynomial in n, $|D|$ and ε^{-1}. Due to the previous observations it can be bounded by:

$$\mathcal{O}\Big(n^2 \frac{m}{\varepsilon}|D|(\ln(nm) + \frac{1}{\varepsilon^2})((nm)^{1.5356} + \frac{m}{\varepsilon^4})\Big)$$

This analysis however, is not tight.

Second case: $1/\varepsilon \geq m$ In this case we can apply all the steps of the fixed-resource AFPTAS that follow the linear grouping, yielding a schedule for all but at most $G = 1/\varepsilon'^2$ jobs. The remaining jobs again are scheduled in the end. The corresponding schedule has a makespan of at most:

$$\begin{aligned}&(1+2\varepsilon')\mathrm{OPT}_{\mathrm{pre}}(I) + 2/\varepsilon'^2 p_{max} \\ \leq &(1+2\varepsilon')(1+\varepsilon')\mathrm{OPT}(\tilde{I}) + \mathcal{O}(1/\varepsilon^2)p_{max} \\ \leq &(1+\varepsilon)\mathrm{OPT}(\tilde{I}) + \mathcal{O}(1/\varepsilon^2)p_{max}\end{aligned}$$

The overall running time is obviously smaller than in the first case.

Concluding remarks One might note that there are many instances for which $\pi_{\max}$ is very big. Therefore we want to point out that the above algorithm can be modified such that the $\pi_{\max}$-factor can at least be bounded by $\mathrm{OPT}_{\mathrm{pre}}(I)$. This can be done by changing the computation of the preemptive solution: We guess $\mathrm{OPT}_{\mathrm{pre}}(I)$ via binary search and only use configurations in which every job has a processing time of at most $\mathrm{OPT}_{\mathrm{pre}}(I)$. This increases the overall running time only slightly.

Furthermore the algorithm will also work for variants of the problem with more succinctly encoded processing time functions, as long as an approximate preemptive schedule can be efficiently computed.

References

[1] Peter A. Beling and Nimrod Megiddo. Using fast matrix multiplication to find basic solutions. *Theoretical Computer Science*, 205(1):307–316, 1998.

[2] Wenceslas Fernandez de la Vega and George S. Lueker. Bin packing can be solved within 1+epsilon in linear time. *Combinatorica*, 1(4):349–355, 1981.

[3] Leah Epstein and Asaf Levin. AFPTAS results for common variants of bin packing: A new method for handling the small items. *SIAM Journal on Optimization*, 20(6):3121–3145, 2010.

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

[5] Michael R. Garey and David S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing*, 4(4):397–411, 1975.

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

[7] Alexander Grigoriev, Maxim Sviridenko, and Marc Uetz. Machine scheduling with resource dependent processing times. *Mathematical programming*, 110(1):209–228, 2007.

[8] Alexander Grigoriev and Marc Uetz. Scheduling jobs with time-resource tradeoff via nonlinear programming. *Discrete optimization*, 6(4):414–419, 2009.

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

[10] Ellis Horowitz and Sartaj Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *J. ACM*, 23(2):317–327, April 1976.

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

[12] ShanXue Ke, BenSheng Zeng, WenBao Han, and Victor Y Pan. Fast rectangular matrix multiplication and some applications. *Science in China Series A: Mathematics*, 51(3):389–406, 2008.

[13] Hans Kellerer. An approximation algorithm for identical parallel machine scheduling with resource dependent processing times. *Operations Research Letters*, 36(2):157–159, 2008.
[14] Claire Kenyon and Eric Rémila. A near-optimal solution to a two-dimensional cutting stock problem. *Mathematics of Operations Research*, 25(4):645–656, 2000.
[15] Eugene L. Lawler. Fast approximation algorithms for knapsack problems. *Mathematics of Operations Research*, 4(4):339–356, 1979.
[16] Monaldo Mastrolilli and Marcus Hutter. Hybrid rounding techniques for knapsack problems. *Discrete Applied Mathematics*, 154(4):640 – 649, 2006. Efficient Algorithms Efficient Algorithms.
[17] Martin Niemeier and Andreas Wiese. Scheduling with an orthogonal resource constraint. In *Approximation and Online Algorithms*, pages 242–256. Springer, 2013.