Approximation Algorithms for Scheduling with Reservations

Florian Diedrich[1,⋆,⋆⋆], Klaus Jansen[1,⋆⋆], Fanny Pascual[2], and Denis Trystram[2,⋆⋆⋆]

[1] Institut für Informatik, Christian-Albrechts-Universität zu Kiel, Olshausenstr. 40, 24098 Kiel, Germany
{fdi,kj}@informatik.uni-kiel.de

[2] LIG – Grenoble University, 51 avenue Jean Kuntzmann, 38330 Montbonnot Saint-Martin, France
{fanny.pascual,denis.trystram}@imag.fr

Abstract. We study the problem of scheduling n independent jobs on a system of m identical parallel machines in the presence of reservations. This constraint is practically important; for various reasons, some machines are not available during specified time intervals. The objective is to minimize the makespan. This problem is inapproximable in the general case unless $\mathsf{P} = \mathsf{NP}$ which motivates the study of suitable restrictions. We use an approach based on algorithms for multiple subset sum problems; our technique yields a polynomial time approximation scheme (PTAS) which is best possible in the sense that the problem does not admit an FPTAS unless $\mathsf{P} = \mathsf{NP}$. The PTAS presented here is the first one for the problem under consideration; so far, not even for special cases approximation schemes have been proposed. We also derive a low cost algorithm with a constant approximation ratio and discuss additional FPTASes for special cases and complexity results.

1 Introduction

In parallel machine scheduling, an important issue is a scenario where time intervals of machine unavailability must be taken into account. This phenomenon occurs due to periods of regular maintenance or because high-priority jobs are present. Here we obtain deterministic models capturing realistic industrial settings and scheduling problems in parallel

⋆ Research supported in part by a grant "DAAD Doktorandenstipendium" of the German Academic Exchange Service. Part of this work was done while visiting the LIG, Grenoble University.

⋆⋆ Supported in part by EU research project AEOLUS, "Algorithmic Principles for Building Efficient Overlay Computers", EU contract number 015964, and in part by DFG priority program 1126, "Algorithmics of Large and Complex Networks".

⋆⋆⋆ Part of this work was supported by the "CoreGRID" Network of Excellence.

computing. We study non-preemptive scheduling of sequential jobs on a system of identical parallel machines; however, these may be unavailable for certain periods of time which are known beforehand. This setting is also called the *non-resumable* case [17,19,20]. The objective is to minimize the makespan $C_{\max}$, which is the maximum of the completion times of all jobs. As discussed below, quite restricted special cases of the model considered here have been studied. On the algorithmic side, only list scheduling algorithms (or similar approaches) and exact exponential algorithms have been analyzed and experimentally evaluated, respectively.

Contributions. We take a novel approach by using algorithms for multiple subset sum problems to govern the scheduling of jobs on identical parallel machines with reservations. We obtain a dual approximation algorithm [8], more precisely a PTAS, for the case of an arbitrary number m of machines. We show that our problem does not admit an FPTAS unless $\mathsf{P} = \mathsf{NP}$ and present additional complexity results. For $m \in \{1, 2\}$ with one reservation we obtain FPTASes; we also discuss how fast greedy algorithms can be obtained from our approach.

This article is organized as follows. Sect. 2 defines the problem and discusses the inapproximability of the general case. In Sect. 3 we present a PTAS for a suitably restricted problem as well as FPTASes for $m \in \{1, 2\}$ with one reservation. Finally we sketch how to obtain a fast approximation algorithm for our general problem. In Sect. 3.3 our approximation algorithms are complemented by suitable hardness results. Finally we summarize the results and conclude in Sect. 4.

Related problems and previous results. Lee [16] and Lee et al. [18] studied identical parallel machines which may have different starting times; here, the LPT policy (where tasks are greedily scheduled from the largest task to the smallest task) was analyzed. Lee [17] studied the case where at most one reservation per machine is permitted while one machine is continuously available and obtained suitable approximation ratios for low-complexity list scheduling algorithms. Liao et al. [20] presented an experimental study of an exact algorithm for $m = 2$ within the same scenario. Hwang et al. [9] studied the LPT policy for the case where at most one interval of unavailability per machine is permitted. They proved a tight bound of $1 + \lceil m/(m - \lambda) \rceil / 2$ where at most $\lambda \in [m - 1]$ machines are permitted to be unavailable simultaneously. The reader can find in [19], Chapt. 22, problem definitions and a survey about previous results. In [23], Scharbrodt et al. present approximation schemes and inapproximability results for a setting where the reservations also contribute to the makespan. So far, the model under consideration has not been approached with approximation schemes, not even for the special cases which have already been studied [9,17,20].

The approach taken in our work is based on multiple subset sum problems. These are special cases of knapsack problems, which belong to the oldest problems in combinatorial optimization and theoretical computer science. Hence, we benefit from the fact that they are relatively well understood. For the classical problem (KP) with one knapsack, besides the result by Ibarra & Kim [10], Lawler presented a sophisticated FPTAS [15] which was later improved by Kellerer & Pferschy [13]; see also the textbooks by Martello & Toth [21] and Kellerer et al. [14] for surveys. The case where the item profits equal their weights is called the subset sum problem and denoted as SSP. The problem with *multiple* knapsacks (MKP) is a natural generalization of KP; the case with multiple knapsacks where the item profits equal their weights is called the *multiple* subset sum problem (MSSP). Various special cases and extensions of these problems have been studied [1,2,3,4,5,11,12], finally yielding PTASes and FPTASes for the cases upon which our approach is based [2,4,12].

2 Problem Definition and Preliminaries

Now we formally define our problem. Let $m \in \mathbb{N}^*$ denote the number of machines. An instance I consists of n jobs characterized by processing times $p_1, \dots, p_n$, and r reservations $R_1, \dots, R_r$. For each $k \in [r]$, $R_k = (i_k, s_k, t_k)$ indicates unavailability of machine i_k in the time interval $[s_k, t_k)$, where $s_k, t_k \in \mathbb{N}, i_k \in [m]$ and $s_k < t_k$. We suppose that for reservations on the same machine there is no overlap; for two reservations $R_k, R_{k'}$ such that $i_k = i_{k'}$ holds, we have $[s_k, t_k) \cap [s_{k'}, t_{k'}) = \emptyset$. For each $i \in [m]$ let $R'_i := \{R_k \in I | i_k = i\}$ denote the set of reservations for machine i. Finally, for each $i \in [m]$ suppose that R'_i is sorted increasingly with respect to the starting times of the reservations; more precisely, $R'_i = \{(i, s_{i1}, t_{i1}), \dots, (i, s_{ir_i}, t_{ir_i})\}$ such that $s_{i1} < \cdots < s_{ir_i}$ where we set $r_i := |R'_i|$. These assumptions are established algorithmically in $O(r \log r)$ time by sorting $\{R_1, \dots, R_r\}$ lexicographically with respect to the first two components of its elements and partitioning it into $R'_1, \dots, R'_m$ and finally merging adjacent reservations in R'_i for each $i \in [m] \setminus \{1\}$. In the sequel we use $P(I) := \sum_{j=1}^{n} p_j$ to denote the total processing time of an instance I and for each $S \subseteq [n]$ we write $P(S) := \sum_{j \in S} p_j$ for the total processing time of S. A schedule is a function $\sigma : [n] \to [m] \times [0, \infty)$ that maps each job to its executing machine and starting time; if σ is clear from the context it may be dropped from notation. Our goal is to compute a non-preemptive schedule of the tasks such that no task is scheduled on a machine that is unavailable, and on each machine at most one task runs at a given time; the objective is to minimize the makespan $C_{\max}$. Using the 3-field notation, we denote our problem by $\mathrm{P}m|nr\text{-}a|C_{\max}$ and show its inapproximability for $m \geq 2$.

Lemma 1. *No polynomial time algorithm for* $\mathrm{P}m|nr\text{-}a|C_{\max}$ *with* $m \geq 2$ *has a constant approximation ratio unless* $\mathsf{P} = \mathsf{NP}$*.*

Proof. Let $c \in \mathbb{N}^*$; for an instance I of Partition, which is NP-complete [7], given by $I = \{a_1, \ldots, a_n\}$ such that $\sum_{i \in I} a_i = 2A$, we define an instance I' of $\mathrm{P}m|nr\text{-}a|C_{\max}$ by setting $p_i := a_i$ for each $i \in [n]$, $R_1 := (1, A, A+c)$, $R_2 := (2, A, A+c)$ and $R_k := (k, 0, A+c)$ for $k \in [m] \setminus \{1, 2\}$. Then I is a yes-instance of Partition if and only if I' has an optimal makespan of $C^*_{\max} = A$. However, any suboptimal schedule of I' for a yes-instance I of Partition has a makespan $C_{\max} > c$; by choosing c large, any suboptimal solution can be arbitrarily bad. □

The inapproximability of the general case is due to the permission of intervals in which no machine is available. Hence it is reasonable to suppose that at each time step there is an available machine. This is not sufficient since we can prove in this case the same inapproximability result by considering, for example, the following instance: there is, for a given period p, a set of reservations which alternate on two machines in a such a way that there are no two reservations at the same time and the period between two consecutive reservations is smaller than the length of any task of the instance. In this case, no task can be put during time period p and we get the same inapproximability result as if we had on each of these machines a big reservation of length p. Hence we suppose that at least one machine is always available. If we consider that reservations are jobs with high priority which are already scheduled, then, since the machines are identical, the reservations can be put on the machines in such a way that w.l.o.g. the *first* machine is always available, hence $i_k \neq 1$ for each reservation R_k. This can be done by distributing the reservations one by one and always putting a reservation on the machine with maximum index $i \in [m]$ among the available machines.

We use $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ to denote this restricted problem; $1up$ means that at least one machine is always available. This problem is NP-hard even for $m = 2$, which can be seen by following the lines of the proof of Lemma 1 using one reservation $R_1 := (2, A, A+1)$ and arguing that I' has an optimal makespan $C^*_{\max} = A$ if and only if I is a yes-instance of Partition.

3 Approximation Algorithms and Complexity Results

We present approximation algorithms and complexity results. In Subsect. 3.1 we obtain approximation schemes; in Subsect. 3.2 we discuss fast greedy algorithms that are based on the same idea. We close the section with complexity results in Subsect. 3.3.

3.1 Polynomial Time Approximation Schemes

We explain the MSSP approach for $m \geq 2$ to obtain a PTAS for our problem. Later we discuss the cases $m \in \{1, 2\}$, which admit FPTASes for the case where one reservation is permitted. Our idea is based on obtaining a complementary representation for the periods of availability in order to reduce the problem to MSSP which admits a PTAS [2,4]; we derive a dual approximation algorithm [8] by using binary search on the makespan where a PTAS for MSSP serves as a relaxed decision procedure, as illustrated with a Gantt chart in Fig. 1. In Sect. 2 we argued how to obtain sorted sets R'_i of reservations for each $i \in [m] \setminus \{1\}$. We use the algorithm in Fig. 2 to obtain sets of inclusionwise maximal availability intervals A_i for each $i \in [m]$, each containing elements (i, s, t) indicating that machine i is available in $[s, t)$ where $s \in \mathbb{N}$ and $t \in \mathbb{N} \cup \{\infty\}$. Due to space restrictions a detailed discussion of Fig. 2 is omitted.

The running time of the algorithm in Fig. 2 is linear in m, r and independent from n; at most $2r$ intervals of availability are generated. For a fixed $i \in [m]$, we use the initial sorting of R'_i to obtain that the intervals of availability for machine i are sorted with respect to their starting times. More important is the following subroutine that uses $A_1, \ldots, A_m$

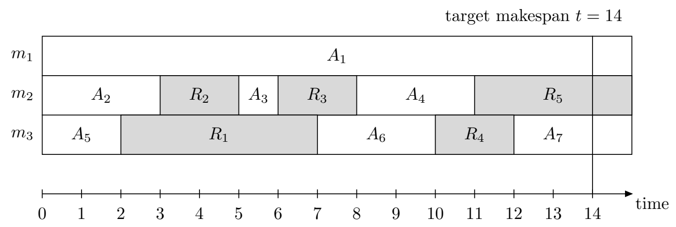


Fig. 1: Sketch illustrating the approach of the algorithm in Fig. 4. The grey zones $R_1, \ldots, R_5$ are the reservations. If the target makespan is 14, we try to fill all the jobs in knapsacks of sizes corresponding to $A_1, \ldots, A_7$; zones A_1 and A_7 end at time 14

```
1. Set $A_1 := \{(1, 0, \infty)\}$ and for each $i \in [m] \setminus \{1\}$ set $A_i := \emptyset$.
2. For each $i \in [m] \setminus \{1\}$ execute Steps 2.1–2.3.
   2.1. If $r_i = 0$, set $A_i := A_i \cup \{(i, 0, \infty)\}$ and proceed with the next iteration of the
        loop in started in Step 2.
   2.2. Set $t := 0$.
   2.3. For each $r \in [r_i]$ execute Steps 2.3.1–2.3.2.
        2.3.1 If $s_{ir} = 0$ then proceed with the next iteration of the loop started in
              Step 2.1, otherwise set $A_i := A_i \cup \{(i, t, s_{ir})\}$ and $t := t_{ir}$.
        2.3.2 If $r = r_i$ then set $A_i := A_i \cup \{(i, t, \infty)\}$.
```

Fig. 2: Algorithm GenAvail

to generate the finite intervals of *availability* for a *fixed finite* planning horizon $[0, t)$ where $t \in \mathbb{N}$.

```
1. For each $i \in [m]$ execute Steps 1.1–1.2.
   1.1. Set $A'_i(t) = \{(i, s', t') \in A_i | s' < t)\}$ and $a_i := |A'_i(t)|$.
   1.2. If $a_i > 0$ set $t_{ia_i} := \min\{t_{ia_i}, t\}$.
```

Fig. 3: Algorithm GenAvailFinite

Step 1.1 in Fig. 3 removes all intervals of availability that begin outside of $[0, t)$ while Step 1.2, if necessary, truncates the last interval on a machine to fit exactly into the planning horizon. The running time of the algorithm in Fig. 3 is independent from n and linear in m, r. We denote $A(t) := \cup_{i=1}^{m} A'_i(t)$ and will use the at most $2r$ intervals stored in $A(t)$ as knapsacks in which we like to pack the jobs in $[n]$. To this end, we use a PTAS for MSSP and for each job $j \in [n]$ define an item j with weight p_j to obtain an instance of MSSP. The algorithm is described in Fig. 4, where MSSPPTAS is a PTAS for MSSP where the capacities of the knapsacks may be different [2,4]. We suppose that MSSPPTAS does not only select a desired $S \subseteq [n]$ but also stores the feasible assignment to the knapsacks as a byproduct.

Theorem 1. *The algorithm in Fig. 4 is a PTAS for* $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$.

Proof. Since the first machine is available at each time step $t \in [0, \infty)$, the sum of processing times $P(I)$ is an upper bound for the optimal makespan $C^*_{\max}$; hence in Step 2, the lower bound LB and the upper bound UB are initialized to have the following properties.

```
1. Use the algorithm in Fig. 2 to generate $A_i$ for each $i \in [m]$.
2. Set $LB := 0$ and $UB := P(I)$.
3. While $UB - LB > 1$ repeat Steps 3.1–3.3.
   3.1 Set $t := \lfloor (UB - LB)/2 \rfloor$. Use the algorithm in Fig. 3 to generate $A(t)$, the set
       of availability intervals for fixed planning horizon $[0, t)$.
   3.2 Use MSSPPTAS with accuracy $\epsilon/m$ to select a set of jobs $S \subseteq [n]$ such that
         $P(S) \geq (1 - \epsilon/m) \max\{P(S') | S' \subseteq [n],$
                 $S'$ permits a feasible packing into the intervals in $A(t)\}$.
   3.3 If $P(S) < (1 - \epsilon/m)P(I)$ then set $LB := t$ else store $S$ and set $UB := t$.
4. Schedule the jobs in the last stored set $S$ into the interval $[0, UB)$ as indicated by
   the solution generated by MSSPPTAS when $S$ was returned; schedule the jobs in
   $[n] \setminus S$ in the interval $[UB, \infty)$ on the first machine without unnecessary idle time.
```

Fig. 4: Algorithm MultiSubsetSumScheduler

1. $LB < C^*_{\max}$.
2. There is a set $S \subseteq [n]$ such that the jobs in S permit a feasible schedule into the time horizon $[0, UB)$ and $P(S) \geq (1 - \epsilon/m)P(I)$.

The second property follows from the fact that, since $C^*_{\max} \leq UB$, all jobs can be scheduled in $[0, UB)$ and thus it is impossible that the algorithm MSSPPTAS returns a set $S \subseteq [n]$ such that $P(S) < (1-\epsilon/m)P(I)$ holds; both properties are invariant under the update of LB and UB in Step 3.3. The number of iterations of the binary search in Step 3 is bounded by $\log P(I) \leq \log(np_{\max}) = \log n + \log p_{\max}$ which is polynomially bounded in the encoding length of I. On termination of the binary search in Step 3, $LB + 1 = UB$ holds, hence $UB \leq C^*_{\max}$ since $LB < C^*_{\max}$ is satisfied. This means that the set S selected in Step 4 can be scheduled in $[0, UB)$ and satisfies $P(S) \geq (1 - \epsilon/m)P(I)$; hence $P([n] \setminus S) \leq \epsilon P(I)/m$ holds. Furthermore the jobs in $[n] \setminus S$ can be scheduled on the first machine in $[UB, \infty)$ since the first machine is available. We have $P(I)/m \leq C^*_{\max}$; in total, the makespan of the schedule generated by the algorithm in Fig. 4 is bounded by $UB + \epsilon P(I)/m \leq C^*_{\max} + \epsilon C^*_{\max} = (1 + \epsilon)C^*_{\max}$ and we obtain the desired approximation ratio. □

However, since the running time of MSSPPTAS may grow exponentially in $1/\epsilon$, the running time of the algorithm in Fig. 4 may also grow exponentially in m. Furthermore, it is known that MSSP does not admit an FPTAS even for the special case of two knapsacks of equal capacity, unless $\mathsf{P} = \mathsf{NP}$ holds, as discussed in [14], Subsect. 10.4. Hence it is impossible for the approach used above to yield an FPTAS for our scheduling problem by replacing MSSPPTAS with a better algorithm, which is not surprising in the light of Corollary 1 in Subsect. 3.3.

For $m = 1$ the situation is different. Lee [17] remarked that $1|nr\text{-}a|C_{\max}$ is strongly NP-hard via reduction from 3-Partition, and the inapproximability can be seen by generalizing a suitable construction. If there is only one reservation, an FPTAS can be obtained since in [12,14] an FPTAS for SSP is available. This case corresponds to a simple knapsack problem; if all the tasks can be scheduled before the reservation then we get an optimal solution; otherwise we use the FPTAS for SSP to schedule as much as possible load before the reservation.

Now we sketch $m = 2$ with one reservation $R_1 = (2, s, t)$; for this case, an FPTAS can be obtained. Due to space constraints we omit the precise algorithmic details. The FPTAS is based on dynamic programming which yields an optimal algorithm with a pseudopolynomial runtime bound. This dynamic programming algorithm can be used to build an FPTAS by a suitable discretization of the state space; we obtain the following result.

Theorem 2. *The problem* $\mathrm{P}2, 1up|nr\text{-}a|C_{\max}$ *with one reservation admits an FPTAS.*

3.2 Greedy Algorithms

In [5], a greedy 2-approximation algorithm for MSSP with running time $O(n^2)$ is briefly mentioned; the subject is also discussed in [14], Subsect. 10.4.1, with a slightly different approach yielding the same runtime bound. By using this algorithm instead of MSSPPTAS and changing the bound $1 - \epsilon/m$ to $1/2$ in Step 3 of the algorithm in Fig. 4 we obtain an approximation algorithm with ratio $1 + m/2$ for $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ by following the lines of the proof of Theorem 1. On the other hand, scheduling all jobs on the first machine here yields an m-approximation algorithm; hence the algorithm sketched above yields a better bound than this approach only if $m > 2$ holds.

In [17], Lee studied the case where at most one reservation per machine is permitted and one machine is always available; a tight approximation ratio of $(m + 1)/2$ for LPT is proved. For our generalization $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ we obtain the same asymptotic behaviour in m with our greedy approach. Comparing our result here with the tight bound $1 + \lceil m/(m - \lambda)\rceil/2$ for LPT [9] where $\lambda \in [m - 1]$ is the maximum number of machines which are permitted to be unavailable at the same time, we basically get the same ratio for our case $\lambda = m - 1$. In total, we obtain similar approximation ratios for more general problems, which comes at the cost of increased computational effort, however.

3.3 Complexity Results

We present an inapproximability result which shows that the PTAS for $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ is close to best possible; hence $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ is substantially harder than $\mathrm{P}m||C_{\max}$ which permits an FPTAS [22].

Theorem 3. *The problem* $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ *is strongly* NP*-hard for* $m \geq 2$.

Proof. We reduce from the strongly NP-complete problem 3-Partition [7]; see Fig. 5 for a sketch of the construction.

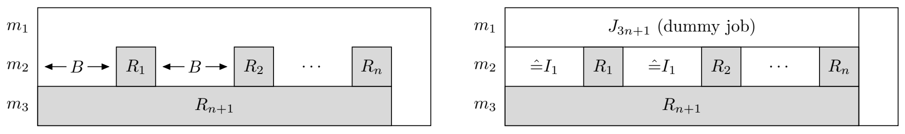
- *Given:* Index set $S = [3n]$, $a_i \in \mathbb{N}^*$ for each $i \in S$ and $B \in \mathbb{N}^*$ such that $B/4 < a_i < B/2$ for each $i \in S$ and $\sum_{i=1}^{3n} = nB$ holds.
- *Question:* Is there a partition of the set S into $S_1, \ldots, S_n$ such that $\sum_{i \in S_j} a_i = B$ holds for each $j \in [n]$?

Given an instance I of 3-Partition we define an instance I' of the problem $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ for $m \geq 2$. We set $p_i := a_i$ for each $i \in [3n]$ (*small* jobs), $p_{3n+1} := n(B+1)$ (*dummy* job) and define suitable reservations $R_i := (2, i(B+1)-1, i(B+1))$, $i \in [n]$, $R_{n+i} := (2+i, 0, n(B+1))$ for each $i \in [m-2]$. I' can be generated from I in time polynomial in the length of I and has an optimal makespan of $C^*_{\max} = n(B+1)$ if and only if I is a yes-instance of 3-Partition by putting the small jobs according to the existing partition $S_1, \ldots, S_n$ in the intervals $[0, B), \ldots, [(n-1)(B+1), n(B+1)-1)$ on machine 2 and putting the dummy job on machine 1; conversely in a schedule with makespan $n(B+1)$ the dummy job must be put on machine 1 and hence the small jobs run on machine 2 which indicates the partition of S into $S_1, \ldots, S_n$ since no more than 3 small jobs can fit into an interval of length B. In total, $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ is strongly NP-hard. □

Since the objective values of feasible schedules for $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ are integral and $C^*_{\max} \leq P(I)$, the next result immediately follows from [6].

Corollary 1. $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ *does not admit an FPTAS for* $m \geq 2$ *unless* P = NP.

It is a natural question whether the problem becomes easier if the number of reservations per machine is restricted to one. Surprisingly, this is not the case, which can be shown by adaptation of a construction from [1]. The following result implies that $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ with at most one reservation per machine for $m \geq 3$ is strongly NP-hard.



(a) Structure of reservations (b) Optimal solution of I'

Fig. 5: Sketch illustrating the proof of Theorem 3

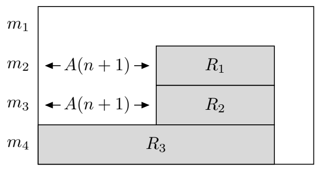
Theorem 4. $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ *does not admit an FPTAS, even if there is at most one reservation per machine, for* $m \geq 3$ *unless* $\mathsf{P} = \mathsf{NP}$.

Proof. We use a reduction from the following problem, Equal Cardinality Partition or ECP for short, which is NP-complete [7]; see Fig. 6 for a sketch of the construction.

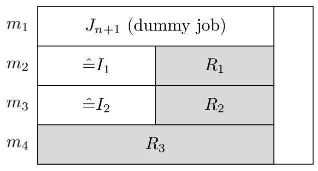
– *Given:* Finite list $I = (a_1, \ldots, a_n)$ of even cardinality with $a_i \in \mathbb{N}^*$ for each $i \in [n]$, $A \in \mathbb{N}^*$ such that $\sum_{i=1}^{n} a_i = 2A$ holds.
– *Question:* Is there a partition of the list I into lists I_1 and I_2 such that $|I_1| = n/2 = |I_2|$ and $\sum_{i \in I_1} a_i = A = \sum_{i \in I_2} a_i$ holds?

Given an instance I of ECP we define an instance I' of $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$ for $m \geq 3$ as follows. We set $p_i := 2A + a_i$ for each $i \in [n]$ (*small* jobs), $p_{n+1} := 2A(n+1)$ (*dummy* job) and $R_k := (k, A(n+1), 2A(n+1))$ for $k \in \{2, 3\}$ and $R_k := (k, 0, 2A(n+1))$ for each $k \in [m] \setminus \{1, 2, 3\}$. I' is generated from I in running time polynomial in the length of I. Furthermore I' has an optimal makespan of $C^*_{\max} = 2A(n+1)$ if and only if I is a yes-instance by executing the small jobs according to the partition I_1 and I_2 on machines 2 and 3 and putting the dummy job on machine 1; conversely in a schedule with makespan $2A(n+1)$ the dummy job is put on machine 1 and hence the small jobs run on machines 2 and 3 which indicates the partition of I into I_1 and I_2 since no more than $n/2$ jobs fit into an availability interval of length $A(n+1)$. Let I be a yes-instance of ECP and consider a suboptimal schedule of I'; the makespan of a suboptimal schedule of I' must be at least $2A(n+1) + A$ since every job in I' has a processing time larger than A and is scheduled either on machine $i \in [m] \setminus \{1\}$ or on machine 1 together with the dummy job, unless the dummy job is scheduled on a machine other than the first one. Given an FPTAS for $\mathrm{P}m, 1up|nr\text{-}a|C_{\max}$, choose $\epsilon \in (0, 1)$ such that

$$1 + \epsilon < \frac{2A(n+1) + A}{2A(n+1)} = \frac{2n+3}{2n+2}$$



(a) Structure of reservations



(b) Optimal solution of I'

Fig. 6: Sketch illustrating the proof of Theorem 4

holds, which is equivalent to $\epsilon < 1/(2n+2)$; consequently ϵ can be chosen in such a way that $1/\epsilon$ is polynomially bounded in n and hence polynomially bounded in the encoding length of I. Then, the FPTAS generates a schedule with makespan $C_{\max}$ such that

$$C_{\max} \leq (1+\epsilon)C^*_{\max} < \frac{2A(n+1)+A}{2A(n+1)} 2A(n+1) = 2A(n+1)+A$$

holds. Hence I' is solved to optimality in polynomial time and I is identified as a yes-instance of ECP, which is impossible unless $\mathsf{P} = \mathsf{NP}$. □

4 Conclusion

We studied scheduling on a constant number of identical parallel machines with reservations and have shown that a sensible restriction to Pm, 1up|nr-a|$C_{\max}$ is necessary to obtain a bounded approximation ratio. On the algorithmic side we have taken an approach that is based on using approximation algorithms for SSP and MSSP. We obtained FPTASes for 1|nr-a|$C_{\max}$ and P2, 1up|nr-a|$C_{\max}$ with one reservation, respectively. For the case of arbitrary constant m our approach yields a PTAS and we have shown that no FPTAS exists unless $\mathsf{P} = \mathsf{NP}$ holds, even if the number of reservations per machine is restricted to one.

Acknowledgements. The authors thank Erik Saule and Ulrich M. Schwarz for many fruitful discussions.

References

1. A. Caprara, H. Kellerer, and U. Pferschy. The multiple subset sum problem. Technical report, Technische Universität Graz, 1998.

2. A. Caprara, H. Kellerer, and U. Pferschy. A PTAS for the multiple subset sum problem with different knapsack capacities. *Inf. Process. Lett.*, 73(3-4):111–118, 2000.
3. A. Caprara, H. Kellerer, and U. Pferschy. A 3/4-approximation algorithm for multiple subset sum. *J. Heuristics*, 9(2):99–111, 2003.
4. C. Chekuri and S. Khanna. A polynomial time approximation scheme for the multiple knapsack problem. *SIAM J. Comput.*, 35(3):713–728, 2005.
5. M. Dawande, J. Kalagnanam, P. Keskinocak, F. S. Salman, and R. Ravi. Approximation algorithms for the multiple knapsack problem with assignment restrictions. *J. Comb. Optim.*, 4(2):171–186, 2000.
6. M. R. Garey and D. S. Johnson. "strong" NP-completeness results: Motivation, examples, and implications. *J. ACM*, 25(3):499–508, 1978.
7. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.
8. D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: theoretical and practical results. *J. ACM*, 34(1):144–162, 1987.
9. H.-C. Hwang, K. Lee, and S. Y. Chang. The effect of machine availability on the worst-case performance of LPT. *Disc. App. Math.*, 148(1):49–61, 2005.
10. O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *J. ACM*, 22(4):463–468, 1975.
11. H. Kellerer. A polynomial time approximation scheme for the multiple knapsack problem. In D. S. Hochbaum, K. Jansen, J. D. P. Rolim, and A. Sinclair, editors, *RANDOM-APPROX*, volume 1671 of *Lecture Notes in Computer Science*, pages 51–62. Springer, 1999.
12. H. Kellerer, R. Mansini, U. Pferschy, and M. G. Speranza. An efficient fully polynomial approximation scheme for the subset-sum problem. *J. Comput. Syst. Sci.*, 66(2):349–370, 2003.
13. H. Kellerer and U. Pferschy. A new fully polynomial time approximation scheme for the knapsack problem. *J. Comb. Optim.*, 3(1):59–71, 1999.
14. H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems.* Springer, 2004.
15. E. L. Lawler. Fast approximation algorithms for knapsack problems. *Math. Oper. Res.*, 4(4):339–356, 1979.
16. C.-Y. Lee. Parallel machines scheduling with non-simultaneous machine available time. *Disc. App. Math.*, 30:53–61, 1991.
17. C.-Y. Lee. Machine scheduling with an availability constraint. *J. Global Optimization, Special Issue on Optimization of Scheduling Applications*, 9:363–384, 1996.
18. C.-Y. Lee, Y. He, and G. Tang. A note on "parallel machine scheduling with non-simultaneous machine available time". *Disc. App. Math.*, 100(1-2):133–135, 2000.
19. J. Y.-T. Leung, editor. *Handbook of Scheduling.* Chapman & Hall, 2004.
20. C.-J. Liao, D.-L. Shyur, and C.-H. Lin. Makespan minimization for two parallel machines with an availability constraint. *European J. of Operational Research*, 160:445–456, 2003.
21. S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations.* Wiley, 1990.
22. S. Sahni. Algorithms for scheduling independent tasks. *J. ACM*, 23(1):116–127, 1976.
23. M. Scharbrodt, A. Steger, and H. Weisser. Approximability of scheduling with fixed jobs. *J. Scheduling*, 2:267–284, 1999.