The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)

A Multivariate Complexity Analysis of the Material Consumption Scheduling Problem

Matthias Bentert,[1] **Robert Bredereck,**[1, 2] **Péter Györgyi,**[3]
Andrzej Kaczmarczyk,[1] and **Rolf Niedermeier**[1]
[1] Technische Universität Berlin, Faculty IV, Algorithmics and Computational Complexity, Berlin, Germany
[2] Humboldt-Universität zu Berlin, Institut für Informatik, Algorithm Engineering, Berlin, Germany
[3] Institute for Computer Science and Control, Budapest, Hungary
{matthias.bentert, a.kaczmarczyk, rolf.niedermeier}@tu-berlin.de, robert.bredereck@hu-berlin.de, gyorgyi.peter@sztaki.hu

Abstract

The NP-hard MATERIAL CONSUMPTION SCHEDULING PROBLEM and related problems have been thoroughly studied since the 1980's. Roughly speaking, the problem deals with minimizing the makespan when scheduling jobs that consume non-renewable resources. We focus on the single-machine case without preemption: from time to time, the resources of the machine are (partially) replenished, thus allowing for meeting a necessary pre-condition for processing further jobs, each of which having individual resource demands. We initiate a systematic exploration of the parameterized computational complexity landscape of the problem, providing parameterized tractability as well as intractability results. Doing so, we mainly investigate how parameters related to the resource supplies influence the computational complexity. Thereby, we get a deepened understanding of this fundamental scheduling problem.

Introduction

Consider the following motivating example. Every day, an agent works for a number of clients, all of equal importance. The clients, one-to-one corresponding to jobs, each time request a service having individual processing time and individual consumption of a non-renewable resource; examples for such resources include raw material, energy, and money. The goal is to finish all jobs as early as possible, known as minimizing the makespan in the scheduling literature. Unfortunately, the agent only has a limited initial supply of the resource which is to be renewed (with potentially different amounts) at known points of time during the day. Since the job characteristics (resource consumption, job length) and the resource delivery characteristics (delivery amount, point of time) are known in advance, the objective thus is to find a feasible job schedule minimizing the makespan. Notably, jobs cannot be preempted and only one at a time can be executed. Figure 1 provides a concrete numerical example with six jobs with varying job lengths and resource requirements.

The described problem setting is known as minimizing the makespan on a single machine with non-renewable resources. Notably, in our example we considered the special but perhaps most prominent case of just one type of resource. More specifically, we study the single-machine variant of the NP-hard MATERIAL CONSUMPTION SCHEDULING PROBLEM. Formally, we have a set $\mathcal{R}$ of resources and a set $\mathcal{J} = \{J_1, \dots, J_n\}$ of jobs to be scheduled on a single machine without preemption. The machine can process at most one job at a time. Each job has a processing time $p_j \in \mathbb{Z}_+$ and a resource requirement $a_{ij} \in \mathbb{Z}_+$ from resource $i \in \mathcal{R}$. We have resource supplies at q different points of time $0 = u_1 < u_2 < \cdots < u_q$, where the vector $\tilde{b}_\ell = (\tilde{b}_{i,\ell})_{i \in \mathcal{R}} \in \mathbb{Z}_+^{|\mathcal{R}|}$ represents the quantities supplied at time u_ℓ. The starting time S_j for each job J_j is specified by a *schedule* σ, which is *feasible* if (i) the jobs do not overlap in time, and (ii) at any point of time t the total supply from each resource is at least the total request of the jobs starting until t, that is,

$$\sum_{\ell: u_\ell \le t} \tilde{b}_{i,\ell} \ge \sum_{j: S_j \le t} a_{ij}, \ \forall i \in \mathcal{R}.$$

Note that in case of just one resource type (as in our starting example in Figure 1), we simply drop the indices corresponding to the single resource. The objective is to minimize the maximum job completion time (makespan) $C_{\max} := \max_{j \in \mathcal{J}} C_j$, where C_j is the completion time of job J_j. In the remainder of the paper, we make the following simplifying assumptions (which can be easily achieved) guaranteeing sanity of the instances and filtering out trivial cases.

Assumption 1. *Without loss of generality, we assume that*

1. *there are enough resources supplied to process all jobs:* $\sum_{\ell=1}^{q} \tilde{b}_\ell \ge \sum_{j \in \mathcal{J}} a_j$;
2. *each job has at least one non-zero resource requirement:* $\forall_{j \in \mathcal{J}} \sum_{i \in \mathcal{R}} a_{i,j} > 0$; *and*
3. *at least one resource unit is supplied at time* 0: $\sum_{i \in \mathcal{R}} \tilde{b}_{i,0} > 0$.

The MATERIAL CONSUMPTION SCHEDULING PROBLEM is NP-hard even in case of one machine, only two supply dates ($q = 2$), and if the processing time of each job is the same as its resource requirement, that is, $p_j = a_j$, $\forall j \in \mathcal{J}$ (Carlier 1984). While many variants of the MATERIAL CONSUMPTION SCHEDULING PROBLEM

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

p_j	1	1	1	2	2	3
a_j	3	1	2	3	2	6

u_ℓ	0	3	5	9
$\tilde{b}_\ell$	3	6	2	6

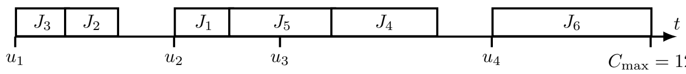

Figure 1: An example (left) with one resource type and a solution (right) with makespan 12. The processing times and the resource requirements are in the first table, while the supply dates and the supplied quantities are in the second. Note that J_3 and J_2 consume all of the resources supplied at $u_1 = 0$, thus we have to wait for the next supply to schedule further jobs.

have been studied in the literature in terms of heuristics, polynomial-time approximation algorithms, or the detection of polynomial-time solvable special cases, we are not aware of any previous systematic studies concerning a multivariate complexity analysis. In other words, we study, seemingly for the first time, several natural problem-specific parameters and investigate how they influence the computational complexity of the problem. Doing so, we prove both parameterized hardness as well as encouraging fixed-parameter tractability results for this NP-hard problem.

Related Work. Over the years, performing multivariate, parameterized complexity studies for fundamental scheduling problems became more and more popular (Bentert, van Bevern, and Niedermeier 2019; van Bevern et al. 2015; van Bevern, Niedermeier, and Suchý 2017; Bodlaender and Fellows 1995; Bodlaender and van der Wegen 2020; Ganian, Hamm, and Mescoff 2020; Fellows and McCartin 2003; Heeger et al. 2021; Hermelin et al. 2019a,b, 2020; Hermelin, Shabtay, and Talmon 2019; Knop and Koutecký 2018; Mnich and van Bevern 2018; Mnich and Wiese 2015). We contribute to this field by a seemingly first-time exploration of the MATERIAL CONSUMPTION SCHEDULING PROBLEM, focusing on one machine and the minimization of the makespan.

The problem was introduced in the 1980's (Carlier 1984; Slowinski 1984). Indeed, even a bit earlier a problem where the jobs required non-renewable resources, but without any machine environment, was studied (Carlier and Rinnooy Kan 1982). The problem appears in several real-world applications, for instance, in the continuous casting stage of steel production (Herr and Goel 2016), in managing deliveries by large-scale distributors (Belkaid et al. 2012), or in shoe production (Carrera, Ramdane-Cherif, and Portmann 2010).

Carlier (1984) proved several complexity results for different variants in the single-machine case, while Slowinski (1984) studied the parallel machine variant of the problem with preemptive jobs. Previous theoretical results mainly concentrate on the computational complexity and polynomial-time approximability of different variants; in this literature review we mainly focus on the most important results for the single-machine case and minimizing makespan as the objective. We remark that there are several recent results for variants with other objective functions (Bérczi, Király, and Omlor 2020; Györgyi and Kis 2019, 2020), with a more complex machine environment (Györgyi and Kis 2017), and with slightly different resource constraints (Davari et al. 2020).

Toker, Kondakci, and Erkip (1991) proved that the variant where the jobs require one non-renewable resource reduces to the 2-MACHINE FLOW SHOP PROBLEM provided that the single non-renewable resource has a unit supply in every time period. Later, Xie (1997) generalized this result to multiple resources. Grigoriev, Holthuijsen, and van de Klundert (2005) showed that the variant with unit processing times and two resources is NP-hard, and they also provided several polynomial-time 2-approximation algorithms for the general problem. There is also a polynomial-time approximation scheme (PTAS) for the variant with one resource and a constant number of supply dates and a fully polynomial-time approximation scheme (FPTAS) for the case with $q = 2$ supply dates and one non-renewable resource (Györgyi and Kis 2014). Györgyi and Kis (2015b) presented approximation-preserving reductions between problem variants in case of $q = 2$ and variants of the MULTIDIMENSIONAL KNAPSACK PROBLEM. These reductions have several consequences, for example, it was shown that the problem is NP-hard if there are two resources, two supply dates, and each job has a unit processing time, or that there is no FPTAS for the problem with two non-renewable resources and $q = 2$ supply dates, unless P = NP. Finally, there are three further results (Györgyi and Kis 2015a): (i) a PTAS for the variant where the number of resources and the number of supply dates are constants; (ii) a PTAS for the variant with only one resource and an arbitrary number of supply dates if the resource requirements are proportional to job processing times; and (iii) APX-hardness when the number of resources is part of the input.

Preliminaries and Notation. We use the standard three field $\alpha|\beta|\gamma$-notation (Graham et al. 1979), where α denotes the machine environment, β the further constraints like additional resources, and γ the objective function. We always consider a single machine, that is, there is a 1 in the α field. The non-renewable resources are described by nr in the β field and $\mathrm{nr} = r$ means that there are r different resource types. In our work, the only considered objective is the makespan $C_{\max}$. For example, the MATERIAL CONSUMPTION SCHEDULING PROBLEM variant with a single machine, single resource type, and with the makespan as the objective is expressed as $1|\,\mathrm{nr} = 1|C_{\max}$. We also sometimes consider the so-called *non-idling scheduling* (introduced by Chrétienne (2008)), indicated by NI in the α field, in which a machine can only process all jobs continuously, without intermediate idling. As we make the simplifying assumption that the machine has to start processing jobs at time 0, we drop the optimization goal $C_{\max}$ whenever considering non-idling scheduling. When there is just one resource ($\mathrm{nr} = 1$), then we write a_j instead of $a_{1,j}$ and $\tilde{b}_j$

	q	$b_{\max}$	$u_{\max}$	$a_{\max}$	$a_{\max}+q$
$1\|\,\text{nr}=1,\ p_j=1\|C_{\max}$			P‡		
$1\|\,\text{nr}=1,\ p_j=ca_j\|C_{\max}$	W[1]-h◇, XP♣	p-NP-h■	FPT♢	XP▲	FPT†
$1\|\,\text{nr}=1,\ \text{unary}\|C_{\max}$	W[1]-h◇, XP♣	p-NP-h■	FPT♢	XP▲	FPT†
$1\|\,\text{nr}=2,\ p_j=1,\ \text{unary}\|C_{\max}$	W[1]-h♡, XP♣	p-NP-h■	XP♣	open	FPT♠
$1\|\,\text{nr}=\text{const},\ \text{unary}\|C_{\max}$	W[1]-h♡, XP♣	p-NP-h■	XP♣	open	FPT♠
$1\|\,\text{nr},\ p_j=1\|C_{\max}$	p-NP-h▼	p-NP-h▼	W[1]-h▼	p-NP-h▼	W[1]-h▼

Table 1: Our results for a single resource type (top) and multiple resource types (bottom). The results correspond to Theorem 3 (‡), Theorem 4 (♢), Theorem 2 (◇), Theorem 1 (■), Györgyi and Kis (2014) (♣), Proposition 1 (♡), Proposition 2 (♠), Theorem 5 (†), Theorem 6 (▲), and Theorem 7 (▼). W[1]-h and p-NP-h stand for, respectively, W[1]-hard and para-NP-hard.

n	number of jobs
q	number of supply dates
j	job index
ℓ	index of a supply
p_j	processing time of job j
$a_{i,j}$	resource requirement of job j from resource i
u_ℓ	the ℓ^{th} supply date
$\tilde{b}_{i,\ell}$	quantity supplied from resource i at u_ℓ
$b_{i,\ell}$	total resource supply from resource i over the first ℓ supplies, that is, $\sum_{k=1}^{\ell}\tilde{b}_{i,k}$

Table 2: Parameter overview. To simplify matters, we introduce the shorthands $p_{\max}$, $a_{\max}$, and $b_{\max}$ for $\max_{j\in\mathcal{J}} p_j$, $\max_{j\in\mathcal{J},i\in\mathcal{R}} a_{ij}$, and $\max_{\ell\in\{1,\ldots,q\},i\in\mathcal{R}} \tilde{b}_{i,\ell}$, respectively.

instead of $\tilde{b}_{1,j}$, etc. We also write $p_j=1$ or $p_j=ca_j$ whenever, respectively, jobs have solely unit processing times or the resource requirements are proportional to the job processing times. Finally, we use "unary" to indicate that all numbers in an instance are encoded in unary. Thus, for example, $1,\text{NI}\,|p_j=1,\text{unary}|-$ denotes a single non-idling machine, unit-processing-time jobs and the unary encoding of all numbers. We summarize the notation of the parameters that we consider in Table 2.

Primer on Multivariate Complexity. To analyze the parameterized complexity (Cygan et al. 2015; Downey and Fellows 2013; Flum and Grohe 2006; Niedermeier 2006) of the MATERIAL CONSUMPTION SCHEDULING PROBLEM, we declare some part of the input the *parameter* (e.g., the number of supply dates). A parameterized problem is *fixed-parameter tractable* if it is in the class FPT of problems solvable in $f(\rho)\cdot|I|^{O(1)}$ time, where $|I|$ is the size of a given instance encoding, ρ is the value of the parameter, and f is an arbitrary computable (usually super-polynomial) function. Parameterized hardness (and completeness) is defined through parameterized reductions similar to classical polynomial-time many-one reductions. For our work, it suffices to additionally ensure that the value of the parameter in the problem we reduce to depends only on the value of the parameter of the problem we reduce from. To obtain parameterized intractability, we use parameterized reductions from problems of the class W[1] which is widely believed to be a proper superclass of FPT.

The class XP contains all problems that can be solved in $|I|^{f(\rho)}$ time for a function f solely depending on the parameter ρ. While XP ensures polynomial-time solvability when ρ is a constant, FPT additionally ensures that the degree of the polynomial is independent of ρ. Unless P = NP, membership in XP can be excluded by showing that the problem is NP-hard for a constant parameter value (for short, we say the problem is para-NP-hard).

Our Contribution. Most of our results are summarized in Table 1. We focus on the parameterized computational complexity of the MATERIAL CONSUMPTION SCHEDULING PROBLEM with respect to several parameters describing resource supplies. We show that the case of a single resource and jobs with unit processing time is polynomial-time solvable. However, if each job has a processing time proportional to its resource requirement, then the MATERIAL CONSUMPTION SCHEDULING PROBLEM becomes NP-hard even for a single resource and when each supply provides one unit of the resource. Complementing an algorithm solving the MATERIAL CONSUMPTION SCHEDULING PROBLEM in polynomial time for a constant number q of supply dates, we show, by proving W[1]-hardness, that the parameterization by q presumably does not yield fixed-parameter tractability. We circumvent the W[1]-hardness by combining the parameter number q of supply dates with the maximum resource requirement $a_{\max}$ of a job, thereby obtaining fixed-parameter tractability for the combined parameter $q+a_{\max}$. Moreover, we show fixed-parameter tractability for the parameter $u_{\max}$ that denotes the last resource supply time. Finally, we provide an outlook on cases with multiple resources and show that fixed-parameter tractability for $q+a_{\max}$ extends when we additionally add the number of resources r to the combined parameter, that is, we show fixed-parameter tractability for $q+a_{\max}+r$. For the MATERIAL CONSUMPTION SCHEDULING PROBLEM with an unbounded number of resources, we show intractability even for the case where all other previously discussed parameters are combined.

Missing details are due to space constraints deferred to the full version (Bentert et al. 2021).

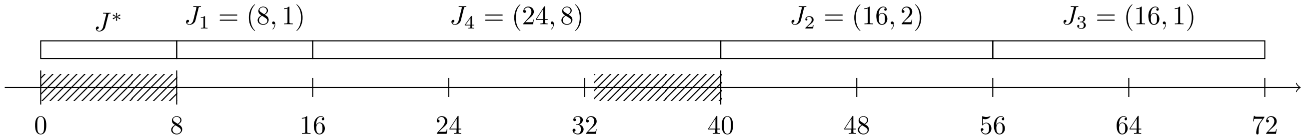

Figure 2: An example of the construction in Theorem 1 for an instance of UNARY BIN PACKING consisting of $k = 2$ bins each of size $B = 4$ and four objects o_1 to o_4 of sizes $s_1 = 1$, $s_2 = s_3 = 2$, and $s_4 = 3$. In the resulting instance of $1|\,\mathrm{nr} = 1, p_j = ca_j|C_{\max}$, there are five jobs ($J^*$ and one job corresponding to each input object) and in each (whole) point in time in the hatched periods there is a supply of one resource. An optimal schedule that first schedules J^* is depicted. Notice that the time periods between the (right-hand) ends of hatched periods correspond to a multiple of the bin size and a schedule is gapless if and only if the objects corresponding to jobs scheduled between the ends of two consecutive shaded areas exactly fill a bin.

Computational Complexity Limits

We start our investigation on the computational complexity of the MATERIAL CONSUMPTION SCHEDULING PROBLEM with outlining the limits of efficient computability. Setting up clear borders of tractability, we identify potential scenarios suitable for seeking efficient solutions. This approach seems especially justified because the MATERIAL CONSUMPTION SCHEDULING PROBLEM is already NP-hard for the quite constrained scenario of unit processing times and two resources (Grigoriev, Holthuijsen, and van de Klundert 2005).

Both hardness results in this section use reductions from UNARY BIN PACKING. Given a number k of bins, a bin size B, and a set $O = \{o_1, o_2, \dots o_n\}$ of n objects of sizes $s_1, s_2, \dots s_n$ (encoded in unary), UNARY BIN PACKING asks to distribute the objects to the bins such that no bin exceeds its capacity. UNARY BIN PACKING is NP-hard and W[1]-hard parameterized by the number k of bins even if $\sum_{i=1}^{n} s_i = kB$ (Jansen et al. 2013).

We first focus on the case of a single resource, for which we find a strong intractability result. In the following theorem, we show that even if each supply comes with a single unit of a resource, then the problem is already NP-hard.

Theorem 1. $1|\,\mathrm{nr} = 1, p_j = ca_j|C_{\max}$ *is para-*NP*-hard with respect to the maximum number* $b_{\max}$ *of resources supplied at once even if all numbers are encoded in unary.*

Proof. Given an instance I of UNARY BIN PACKING with $\sum_{i=1}^{n} s_i = kB$, we construct an instance I' of $1|\,\mathrm{nr} = 1|C_{\max}$ with $b_{\max} = 1$ as described below.

We define n jobs $J_1 = (p_1, a_1), J_2 = (p_2, a_2), \dots, J_n = (p_n, a_n)$ such that $p_i = 2Bs_i$ and $a_i = 2s_i$. We also introduce a special job $J^* = (p^*, a^*)$, with $p^* = 2B$ and $a^* = 1$. Then, we set $2kB$ supply dates as follows. For each $i \in \{0, 1, \dots, k-1\}$ and $x \in \{0, 1, \dots, 2B-1\}$, we create a supply date $q_i^x = (u_i^x, \tilde{b}_i^x) := ((2B + i2B^2) - x, 1)$. We add a special supply date $q^* := (0, 1)$. Next, we show that I is a yes-instance if and only if there is a gapless schedule for I', that is, $C_{\max} = 2(B^2 + B)$. An example of this construction is depicted in Figure 2.

We first show that each solution to I can be efficiently transformed to a schedule with $C_{\max} = 2(B^2 + B)$. A yes-instance for I is a partition of the objects into k bins such that each bin is (exactly) full. Formally, there are k sets $S_1, S_2, \dots S_k$ such that $\bigcup_i S_i = O$, $S_i \cap S_j = \emptyset$ for all $i \neq j$, and $\sum_{o_i \in S_j} s_i = B$ for all j. We form a schedule for I' as follows. First, we schedule job j^* and then, continuously, all jobs corresponding to elements of set S_1, S_2, and so on. The special supply q^* guarantees that the resource requirement of job j^* is met at time 0. The remaining jobs, corresponding to elements of the partitions, are scheduled earliest at time $2B$, when j^* is processed. The jobs representing each partition, by definition, require in total $2B$ resources and take, in total, $2B^2$ time. Thus, it is enough to ensure that in each point $2B + i2B^2$, for $i \in \{0, 1, \dots, k-1\}$, there are at least $2B$ resources available. This is true because, for all $i \in \{0, 1, \dots, k-1\}$, every time $2B + iB^2$ when there is a supply of a single resource is preceded with $2B - 1$ supplies of one resource. Furthermore, none of the preceding jobs can use the freshly supplied resources as the schedule must be gapless and all processing times are multiples of $2B$. As a result, the schedule is feasible.

Now we show that a gapless schedule for I' implies that I is a yes-instance. Let S be a gapless schedule for I'. Observe that all processing times are multiples of $2B$; thus each job has to start at a time that is a multiple of $2B$. For each $i \in \{0, 1, \dots, k-1\}$, we show that there is no job that starts before $2B + i2B^2$ and ends after this time. We show this by induction on i. Since at time 0 there is only one resource available, job J^* (with processing time $2B$) must be scheduled first. Hence the statement holds for $i = 0$. Assuming that the statement holds for all $i < i'$ for some i', we show that it also holds for i'. Assume towards a contradiction that there is a job J that starts before $t := 2B + i'2B^2$ and ends after this time. Let S be the set of all jobs that were scheduled to start between $t_0 := 2B + (i'-1)2B^2$ and t. Recall that for each job $J_{j'} \in S$, we have that $p_{j'} = a_{j'}B$. Hence, since J ends after t, the number of resources used by S is larger than ${}^{(t-t_0)}/_{B} = 2B$. Since only $2B$ resources are available at time t, job J cannot be scheduled before time t or there is a gap in the schedule (a gap would allow to use some of the $2B$ resources supplied in the $2B$ time units just before time t), a contradiction. Hence, there is no job that starts before t and ends after it. Thus, the jobs can be partitioned into "phases," that is, there are $k+1$ sets $T_0, T_1, \dots, T_k$ such that $T_0 = \{J^*\}$, $\bigcup_{h>0} T_h = \mathcal{J} \setminus \{J^*\}$, $T_h \cap T_j = \emptyset$ for all $h \neq j$, and $\sum_{J_j \in T_g} p_j = 2B^2$ for all g. This corresponds to a bin packing where o_g belongs to bin $h > 0$ if and only if $J_g \in T_h$. □

Note that Theorem 1 excludes pseudopolynomial algorithms for the case under consideration since the theorem statement is true also when all numbers are encoded in unary.

Theorem 1 motivates to study further problem-specific parameters. Observe that in the reduction presented in the proof of Theorem 1, we used an unbounded number of supply dates. Györgyi and Kis (2014) have shown a pseudopolynomial algorithm for $1|\,\mathrm{nr}=1|C_{\max}$ for the case that the number q of supplies is a constant. Thus, the question is whether we can even obtain fixed-parameter tractability for our problem by taking the number of supply dates as a parameter. Devising a reduction from UNARY BIN PACKING, we answer this question negatively in the following theorem.

Theorem 2. $1|\,\mathrm{nr}=1,\ p_j=a_j|C_{\max}$ *parameterized by the number of supply dates is* $\mathrm{W}[1]$*-hard even if all numbers are encoded in unary.*

The theorems presented in this section show that our problem is (presumably) not fixed-parameter tractable either with respect to the number of supply dates or with respect to the maximum number of resources per supply. However, as we show in the following section, combining these two parameters allows for fixed-parameter tractability. Furthermore, we present other algorithms that, partially, allow us to successfully evade the hardness presented above.

(Parameterized) Tractability

Our search for efficient algorithms for our variant of the MATERIAL CONSUMPTION SCHEDULING PROBLEM starts with an introductory part presenting two lemmata exploiting structural properties of problem solutions. Afterwards, we employ the lemmata and provide several tractability results, including polynomial-time solvability for one specific case.

Identifying Structured Solutions

A solution to the MATERIAL CONSUMPTION SCHEDULING PROBLEM is an ordered list of jobs to be executed on the machine(s). Additionally, the jobs need to be associated with their starting times. The starting times have to be chosen in such a way that no job starts when the machine is still processing another scheduled job and that each job requirement is met at the moment of starting the job. We show that, in fact, given an order of jobs, one can always compute the times of starting the jobs minimizing the makespan in polynomial time. Formally, we present in Lemma 1 a polynomial-time Turing reduction from $1|\,\mathrm{nr}=r|C_{\max}$ to $1,\mathrm{NI}\,|\,\mathrm{nr}=r|-$. The crux of this lemma is to observe that there always exists an optimal solution to $1|\,\mathrm{nr}=r|C_{\max}$ that is decomposable into two parts. First, when the machine is idling, and second, when the machine is continuously busy until all jobs are processed.

Lemma 1. *There is a polynomial-time Turing reduction from* $1|\,\mathrm{nr}=r|C_{\max}$ *to* $1,\mathrm{NI}\,|nr=r|-$.

Let us further explain the crucial observation backing Lemma 1 since we will extend it in the subsequent Lemma 2. Assume that, for some instance of the MATERIAL CONSUMPTION SCHEDULING PROBLEM, there is some optimal schedule where some job J starts being processed at some time t (in particular, the resource requirements of J are met at t). If, directly after the job the machine idles for some time, then we can postpone processing J to the latest moment which still guarantees that J is ended before the next job is processed. Naturally, in any case, at the new starting time of J we can only have more resources than at the old starting time. Applying this observation exhaustively produces a solution that is clearly separated into idling time and busy time.

We will now further exploit the observation of the previous paragraph beyond only "moving" jobs without changing their mutual order. We first define a *domination relation* over jobs; intuitively, a job dominates another job if it is not shorter and at the same time it requires not more resources.

Definition 1. A job J_j *dominates* a job $J_{j'}$ (written $J_j \leq_D J_{j'}$) if $p_j \geq p_{j'}$ and, for all $i \in \mathcal{R}$, $a_{i,j} \leq a_{i,j'}$.

When we deal with non-idling schedules, for a pair of jobs J_j and $J_{j'}$ where J_j dominates $J_{j'}$, it is better (or at least not worse) to schedule J_j before $J_{j'}$. Indeed, since among these two, J_j's requirements are not greater and its processing time is not smaller, surely after the machine stops processing J_j there will be at least as many resources available as if the machine had processed $J_{j'}$. We formalize this observation in the following lemma.

Lemma 2. *For an instance of* $1,\mathrm{NI}\,|\,\mathrm{nr}\,|-$ *let* $<_D$ *be an asymmetric subrelation of* $\leq_D$*. There always exists a feasible schedule where for every pair* J_j *and* $J_{j'}$ *of jobs it holds that if* $J_j <_D J_{j'}$*, then* J_j *is processed before* $J_{j'}$*.*

Note that in the case of two jobs J_j and $J_{j'}$ dominating each other (i.e., $J_j \leq_D J_{j'}$ and $J_{j'} \leq_D J_j$), Lemma 2 allows for either of them to be processed before the other one.

Applying Structured Solutions

We start with a polynomial-time algorithm that applies both Lemma 1 and Lemma 2 to solve a specific case of the MATERIAL CONSUMPTION SCHEDULING PROBLEM where each two jobs can be compared according to the domination relation (Definition 1). Recall that if this is the case, then Lemma 2 almost exactly specifies the order in which the jobs should be scheduled.

Theorem 3. $1,\mathrm{NI}\,|\,\mathrm{nr}\,|-$ *and* $1|\,\mathrm{nr}\,|C_{\max}$ *are solvable in, respectively, cubic and quadratic time if the domination relation is a weak order on a set of jobs. In particular, for the time* $u_{\max}$ *of the last supply,* $1|\,\mathrm{nr}=1,\ p_j=1|C_{\max}$ *and* $1|\,\mathrm{nr}=1,\ a_j=1|C_{\max}$ *are solvable in* $O(n\log n\log u_{\max})$ *time and* $1,\mathrm{NI}\,|\,\mathrm{nr}=1,\ p_j=1|-$ *and* $1,\mathrm{NI}\,|\,\mathrm{nr}=1,\ a_j=1|-$ *are solvable in* $O(n\log n)$ *time.*

Importantly, it is simple (requiring at most $O(n^2)$ comparisons) to identify the cases for which the above algorithm can be applied successfully.

If the given jobs cannot be weakly ordered by domination, then the problem becomes NP-hard as shown in Theorem 1. This is to be expected since when jobs appear which are incomparable with respect to domination, then one cannot efficiently decide which job, out of two, to schedule first:

the one which requires fewer resource units but has a shorter processing time, or the one that requires more resource units but has a longer processing time. Indeed, it could be the case that sometimes one may want to schedule a shorter job with smaller resource consumption to save resources for later, or sometimes it is better to run a long job consuming, for example, all resources knowing that soon there will be another supply with sufficient resource units. Since NP-hardness presumably excludes polynomial-time solvability, we turn to a parameterized complexity analysis to get around the intractability.

The time $u_{\max}$ of the last supply seems a promising parameter. We show that it yields fixed-parameter tractability. Intuitively, we demonstrate that the problem is tractable when the time until all resources are available is short.

Theorem 4. $1, \mathrm{NI} \,|\, \mathrm{nr} = 1 | C_{\max}$ *parameterized by the time* $u_{\max}$ *of the last supply is fixed-parameter tractable and can be solved in* $O(2^{u_{\max}} \cdot n + n \log n)$ *time.*

Proof. We first sort all jobs by their processing time in $O(n)$ time using bucket sort. We then sort all jobs with the same processing time by their resource requirement in overall $O(n \log n)$ time. We then iterate over all subsets R of $\{1, 2, \ldots, u_{\max}\}$. We will refer to the elements in R by $r_1, r_2, \ldots, r_k$, where $k = |R|$ and $r_i < r_j$ for all $i < j$. For simplicity, we will use $r_0 = 0$. For each r_i in ascending order, we check whether there is a job with a processing time $r_i - r_{i-1}$ that was not scheduled before and if so, then we schedule the respective job that is first in each bucket (the job with the lowest resource requirement). Next, we check whether there is a job left that can be scheduled at r_k and which has a processing time at least $u_{\max} - r_k$. Finally, we schedule all remaining jobs in an arbitrary order and check whether the total number of resources suffices to run all jobs.

We will now prove that there is a valid gapless schedule if and only if all of these checks are met. Notice that if all checks are met, then our algorithm provides a valid gapless schedule. Now assume that there is a valid gapless schedule. We will show that our algorithm finds a (possibly different) valid gapless schedule. Let, without loss of generality, $J_{j_1}, J_{j_2}, \ldots, J_{j_n}$ be a valid gapless schedule and let j_k be the index of the last job that is scheduled latest at time $u_{\max}$. We now focus on the iteration where $R = \{0, p_{j_1}, p_{j_1} + p_{j_2}, \ldots, \sum_{i=1}^{k} p_{j_i}\}$. If the algorithm schedules the jobs $J_{j_1}, J_{j_2}, \ldots, J_{j_k}$, then it computes a valid gapless schedule and all checks are met. Otherwise it schedules some jobs differently but, by construction, it always schedules a job with processing time p_{j_i} at position $i \leq k$. Due to Lemma 2 the schedule computed by the algorithm is also valid. Thus the algorithm computes a valid gapless schedule and all checks are met.

It remains to analyze the running time. The sorting steps in the beginning take $O(n \log n)$ time. There are $2^{u_{\max}}$ iterations for R, each taking $O(n)$ time. Indeed, we can check in constant time for each r_i which job to schedule and this check is done at most n times (as afterwards there is no job left to schedule). Searching for the job that is scheduled at time r_k also takes $O(n)$ time as we can iterate over all remaining jobs and check in constant time whether it fulfills both requirements. □

Another possibility for fixed-parameter tractability via parameters measuring the resource supply structure comes from combining the parameters q and $b_{\max}$. Although both parameters alone yield intractability, combining them gives fixed-parameter tractability in an almost trivial way: By Assumption 1, every job requires at least one resource so $b_{\max} \cdot q$ is an upper bound for the number of jobs. Hence, with this parameter combination, we can try out all possible schedules without idling (which by Lemma 1 extends to solving to $1, \mathrm{NI} \,|\, \mathrm{nr} = 1 | C_{\max}$).

Motivated by this, we replace the parameter $b_{\max}$ by the presumably much smaller (and hence practically more useful) parameter $a_{\max}$. We consider scenarios with only few resource supplies and jobs that require only small units of resources as practically relevant.

Next, Theorem 5 employs the technique of Mixed Integer Linear Programming (MILP) (Bredereck et al. 2020) to positively answer the question of fixed-parameter tractability for the combined parameter $q + a_{\max}$.

Theorem 5. $1, \mathrm{NI} \,|\, \mathrm{nr} = 1 | C_{\max}$ *is fixed-parameter tractable for the combined parameter* $q + a_{\max}$*, where* q *is the number of supplies and* $a_{\max}$ *is the maximum resource requirement per job.*

Proof. Applying the famous theorem of Lenstra (1983), we describe an integer linear program that uses only $f(q, a_{\max})$ integer variables. Lenstra (1983) showed that an (mixed) integer linear program is fixed-parameter tractable when parameterized by the number of integer variables (see also Frank and Tardos (1987) and Kannan (1987) for later improvements). To significantly simplify the description of the integer program, we use an extension to integer linear programs that allows concave transformations on variables (Bredereck et al. 2020).

Our approach is based on two main observations. First, by Lemma 2 we can assume that there is always an optimal schedule that is consistent with the domination order. Second, within a phase (between two resource supplies), every job can be arbitrarily reordered. Roughly speaking, a solution can be fully characterized by the number of jobs that have been started for each phase and each resource requirement.

We use the following non-negative integer variables:

1. $x_{w,s}$ denoting the number of jobs requiring s resources started in phase w,
2. $x^{\Sigma}_{w,s}$ denoting the number of jobs requiring s resources started in all phases between 1 and w (inclusive),
3. α_w denoting the number of resources available in the beginning of phase w,
4. d_w denoting the endpoint of phase w, that is, the time when the last job started in phase w ends.

Naturally, the objective is to minimize d_q.

First, we ensure that $x^{\Sigma}_{w,s}$ are correctly computed from $x_{w,s}$ by adding: $x^{\Sigma}_{w,s} = \sum_{w'=1}^{w} x_{w',s}$ Second, we ensure that all jobs are scheduled sometime. To this end, using $\#_s$

to denote the number of jobs J_j with resource requirement $a_j = s$ we add: $\forall s \in [a_{\max}] : \sum_{w \in [q]} x_{w,s} = \#_s$. Third, we ensure that the α_w variables are set correctly, by setting $\alpha_1 = \tilde{b}_1$, and $\forall 2 \leq w \leq q : \alpha_w = \alpha_{w-1} + \tilde{b}_w - \sum_{s \in [a_{\max}]} x_{w-1,s} \cdot s$. Fourth, we ensure that we always have enough resources: $\forall 2 \leq w \leq q : \alpha_w \geq \tilde{b}_w$. Next, we compute the endpoints d_w of each phase, assuming a schedule respecting the domination order. To this end, let $p_1^s, p_2^s, \ldots, p_{\#_s}^s$ denote the processing times of jobs with resource requirement exactly s in non-increasing order. Further, let $\tau_s(y)$ denote the processing time spent to schedule the y longest jobs with resource requirement exactly s, that is, we have $\tau_s(y) = \sum_{i=1}^{y} p_i^s$. Clearly, $\tau_s(x)$ is a concave function that can be precomputed for each $s \in [a_{\max}]$. To compute the endpoints, we add:

$$\forall w \in [q] : d_w = \sum_{s \in [a_{\max}]} \tau_s(x_{w,s}^{\Sigma}). \qquad (1)$$

Since we assume gapless schedules, we ensure that there is no gap: $\forall 1 \leq w \leq q-1 : d_w \geq u_{w+1} - 1$. This completes the construction of the mixed ILP using concave transformations. The number of integer variables used in the ILP is $2q \cdot a_{\max}$ (for $x_{w,s}^{(\Sigma)}$ variables) plus $2q$ (q for α_w and d_w variables, respectively). Moreover, the only concave transformations used in Constraint Set (1) are piecewise linear with only a polynomial number of pieces (in fact, the number of pieces is at most the number of jobs), as required to obtain fixed-parameter tractability of this extended class of ILPs (Bredereck et al. 2020). ☐

Motivated by Theorem 5, we are interested in the computational complexity of the MATERIAL CONSUMPTION SCHEDULING PROBLEM for cases where only $a_{\max}$ is small. When $a_{\max} = 1$, then we have polynomial-time solvability via Theorem 3. The next theorem shows that this extends to every constant value of $a_{\max}$. To obtain this results, we develop a dynamic-progamming-based algorithm for $1, \mathrm{NI}\,|nr = 1|-$ and apply Lemma 1.

Theorem 6. $1|\,\mathrm{nr} = 1|C_{\max}$ *can be solved in* $O(q \cdot a_{\max} \cdot u_{\max} \cdot \log u_{\max} \cdot n^{2a_{\max}})$ *time.*

The question whether $1|\,\mathrm{nr} = 1|C_{\max}$ is in FPTor W[1]-hard with respect to $a_{\max}$ remains open.

A Glimpse on Multiple Resources

So far we focused on scenarios with only one non-renewable resource. In this section, we give a brief outlook on scenarios with multiple resources (still considering only one machine). Naturally, all hardness results transfer. For the tractability results, we identify several cases where tractability extends in some form, while other cases become significantly harder.

We start with showing that already with two resources and unit processing times of the jobs, the MATERIAL CONSUMPTION SCHEDULING PROBLEM becomes computationally intractable, even when parameterized by the number of supply dates. Note that NP-hardness for $1|\,\mathrm{nr} = 2, p_j = 1|C_{\max}$ can also be transferred from Grigoriev, Holthuijsen, and van de Klundert (2005)[Theorem 4] (the statement is for a different optimization goal but the proof works).

Proposition 1. $1|\,\mathrm{nr} = 2, p_j = 1|C_{\max}$ *is* W[1]*-hard when parameterized by the number of supply dates even if all numbers are encoded in unary.*

Proposition 1 limits the hope for obtaining positive results for the general case with multiple resources. Still, when adding the number of different resources to the combined parameter, we can extend our fixed-parameter tractability result from Theorem 5. Since we expect the number of different resources to be rather small in real-world applications, we consider this result to be of practical interest.

Proposition 2. $1, \mathrm{NI}\,|\,\mathrm{nr} = r|C_{\max}$ *is fixed-parameter tractable for the combined parameter* $q+a_{\max}+r$*, where* q *is the number of supplies and* $a_{\max}$ *is the maximum resource requirement of a job.*

Finally, by a reduction from INDEPENDENT SET we show that the MATERIAL CONSUMPTION SCHEDULING PROBLEM is intractable for an unbounded number of resources even when combining all considered parameters.

Theorem 7. $1|\,\mathrm{nr}, p_j = 1|C_{\max}$ *is* NP*-hard and* W[1]*-hard parameterized by* $u_{\max}$ *even if* $p_{\max} = a_{\max} = b_{\max} = 1$ *and* $q = 2$*.*

Conclusion

We provided a seemingly first thorough multivariate complexity analysis of the MATERIAL CONSUMPTION SCHEDULING PROBLEM on a single machine. Our main concern was the case of one resource type ($\mathrm{nr} = 1$).

Open questions here refer to the parameterized complexity with respect to the single parameters $a_{\max}$ and $p_{\max}$, their combination, and the closely related parameter number of job types. Notably, this might be challenging to answer because these questions are closely related to longstanding open questions for BIN PACKING and $P||C_{\max}$ (Mnich and van Bevern 2018; Knop and Koutecký 2018; Knop, Koutecký, and Mnich 2020). Indeed, parameter combinations may be unavoidable to identify practically relevant tractable cases. Note that it is not hard to derive from our statements (particularly Assumption 1 and Lemma 1) fixed-parameter tractability for $b_{\max} + q$ while for the single parameters $b_{\max}$ and q it is both times computationally hard.

Another challenge is to study the case of multiple machines, which is obviously computationally at least as hard as the case of a single machine but possibly very relevant in practice. It is, however, far from obvious to generalize our algorithms to the multiple-machines case.

We have also seen that cases where the jobs can be ordered with respect to the domination ordering (Definition 1) are polynomial-time solvable. It seems promising to consider structural parameters measuring the distance from this tractable case in the spirit of distance from triviality parameterization (Guo, Hüffner, and Niedermeier 2004; Niedermeier 2006).

Our results for multiple resources certainly mean only first steps. They invite to further investigations.

Acknowledgments

Main work done while RB was with TU Berlin. AK was supported by the DFG project AFFA (BR 5207/1 and NI 369/15). PG was supported by the National Research, Development and Innovation Office – NKFIH (ED_18-2-2018-0006), and by the J. Bolyai Research Scholarship. Project started while RB, PG, and RN were attending the Lorentz center workshop "Scheduling Meets Fixed-Parameter Tractability" February 4–8, 2019, Leiden, the Netherlands, organized by Nicole Megow, Matthias Mnich, and Gerhard J. Woeginger.

References

Belkaid, F.; Maliki, F.; Boudahri, F.; and Sari, Z. 2012. A Branch and Bound Algorithm to Minimize Makespan on Identical Parallel Machines with Consumable Resources. In *Advances in Mechanical and Electronic Engineering*, 217–221. Springer.

Bentert, M.; van Bevern, R.; and Niedermeier, R. 2019. Inductive k-independent graphs and c-colorable subgraphs in scheduling: a review. *Journal of Scheduling* 22(1): 3–20.

Bentert, M.; Bredereck, R.; Györgyi, P.; Kaczmarczyk, A.; and Niedermeier, R. 2021. A Multivariate Complexity Analysis of the Material Consumption Scheduling Problem. *arXiv preprint arXiv:2102.13642 [cs.GT]* .

Bérczi, K.; Király, T.; and Omlor, S. 2020. Scheduling with Non-renewable Resources: Minimizing the Sum of Completion Times. In *Proceedings of the 6th International Symposium on Combinatorial Optimization*, 167–178. Springer.

van Bevern, R.; Mnich, M.; Niedermeier, R.; and Weller, M. 2015. Interval scheduling and colorful independent sets. *Journal of Scheduling* 18(5): 449–469.

van Bevern, R.; Niedermeier, R.; and Suchý, O. 2017. A parameterized complexity view on non-preemptively scheduling interval-constrained jobs: few machines, small looseness, and small slack. *Journal of Scheduling* 20(3): 255–265.

Bodlaender, H. L.; and Fellows, M. R. 1995. W[2]-hardness of precedence constrained K-processor scheduling. *Operations Research Letters* 18(2): 93–97.

Bodlaender, H. L.; and van der Wegen, M. 2020. Parameterized Complexity of Scheduling Chains of Jobs with Delays. In *Proceedings of the 15th International Symposium on Parameterized and Exact Computation (IPEC '20)*, 4:1–4:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Bredereck, R.; Faliszewski, P.; Niedermeier, R.; Skowron, P.; and Talmon, N. 2020. Mixed integer programming with convex/concave constraints: Fixed-parameter tractability and applications to multicovering and voting. *Theoretical Computer Science* 814: 86–105.

Carlier, J. 1984. *Problèmes d'ordonnancements à contraintes de ressources: algorithmes et complexité. Thèse d'état*. Université Paris 6.

Carlier, J.; and Rinnooy Kan, A. H. G. 1982. Scheduling subject to nonrenewable resource constraints. *Operations Research Letters* 1: 52–55.

Carrera, S.; Ramdane-Cherif, W.; and Portmann, M.-C. 2010. Scheduling supply chain node with fixed component arrivals and two partially flexible deliveries. In *Proceedings of the 5th International Conference on Management and Control of Production and Logistics (MCPL '10)*, 6. IFAC Publisher.

Chrétienne, P. 2008. On single-machine scheduling without intermediate delays. *Discrete Applied Mathematics* 156(13): 2543–2550.

Cygan, M.; Fomin, F. V.; Kowalik, L.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2015. *Parameterized Algorithms*. Springer.

Davari, M.; Ranjbar, M.; De Causmaecker, P.; and Leus, R. 2020. Minimizing makespan on a single machine with release dates and inventory constraints. *European Journal of Operational Research* 286(1): 115–128.

Downey, R. G.; and Fellows, M. R. 2013. *Fundamentals of Parameterized Complexity*. Springer.

Fellows, M. R.; and McCartin, C. 2003. On the parametric complexity of schedules to minimize tardy tasks. *Theoretical Computer Science* 298(2): 317–324.

Flum, J.; and Grohe, M. 2006. *Parameterized Complexity Theory*. Springer.

Frank, A.; and Tardos, É. 1987. An application of simultaneous Diophantine approximation in combinatorial optimization. *Combinatorica* 7(1): 49–65.

Ganian, R.; Hamm, T.; and Mescoff, G. 2020. The Complexity Landscape of Resource-Constrained Scheduling. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI '20)*, 1741–1747. ijcai.org.

Graham, R. L.; Lawler, E. L.; Lenstra, J. K.; and Kan, A. R. 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics* 5: 287–326.

Grigoriev, A.; Holthuijsen, M.; and van de Klundert, J. 2005. Basic scheduling problems with raw material constraints. *Naval Research of Logistics* 52: 527–553.

Guo, J.; Hüffner, F.; and Niedermeier, R. 2004. A Structural View on Parameterizing Problems: Distance from Triviality. In *Proceedings of the First International Workshop on Parameterized and Exact Computation*, 162–173. Springer.

Györgyi, P.; and Kis, T. 2014. Approximation schemes for single machine scheduling with non-renewable resource constraints. *Journal of Scheduling* 17: 135–144.

Györgyi, P.; and Kis, T. 2015a. Approximability of scheduling problems with resource consuming jobs. *Annals of Operations Research* 235(1): 319–336.

Györgyi, P.; and Kis, T. 2015b. Reductions between scheduling problems with non-renewable resources and knapsack problems. *Theoretical Computer Science* 565: 63–76.

Györgyi, P.; and Kis, T. 2017. Approximation schemes for parallel machine scheduling with non-renewable resources. *European Journal of Operational Research* 258(1): 113–123.

Györgyi, P.; and Kis, T. 2019. Minimizing total weighted completion time on a single machine subject to non-renewable resource constraints. *Journal of Scheduling* 22(6): 623–634.

Györgyi, P.; and Kis, T. 2020. New complexity and approximability results for minimizing the total weighted completion time on a single machine subject to non-renewable resource constraints. *arXiv preprint arXiv:2004.00972* .

Heeger, K.; Hermelin, D.; Mertzios, G. B.; Molter, H.; Niedermeier, R.; and Shabtay, D. 2021. Equitable Scheduling on a Single Machine. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI '21)*. AAAI Press.

Hermelin, D.; Kubitza, J.-M.; Shabtay, D.; Talmon, N.; and Woeginger, G. J. 2019a. Scheduling two agents on a single machine: A parameterized analysis of NP-hard problems. *Omega* 83: 275–286.

Hermelin, D.; Manoussakis, G.; Pinedo, M.; Shabtay, D.; and Yedidsion, L. 2020. Parameterized Multi-Scenario Single-Machine Scheduling Problems. *Algorithmica* 82(9): 2644–2667.

Hermelin, D.; Pinedo, M.; Shabtay, D.; and Talmon, N. 2019b. On the parameterized tractability of single machine scheduling with rejection. *European Journal of Operational Research* 273(1): 67–73.

Hermelin, D.; Shabtay, D.; and Talmon, N. 2019. On the parameterized tractability of the just-in-time flow-shop scheduling problem. *Journal of Scheduling* 22(6): 663–676.

Herr, O.; and Goel, A. 2016. Minimising total tardiness for a single machine scheduling problem with family setups and resource constraints. *European Journal of Operational Research* 248(1): 123–135.

Jansen, K.; Kratsch, S.; Marx, D.; and Schlotter, I. 2013. Bin packing with fixed number of bins revisited. *Journal of Computer and System Sciences* 79(1): 39–49.

Kannan, R. 1987. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research* 12(3): 415–440.

Knop, D.; and Koutecký, M. 2018. Scheduling meets n-fold integer programming. *Journal of Scheduling* 21(5): 493–503.

Knop, D.; Koutecký, M.; and Mnich, M. 2020. Combinatorial n-fold integer programming and applications. *Mathematical Programming* 184(1): 1–34.

Lenstra, Jr, H. W. 1983. Integer Programming with a Fixed Number of Variables. *Mathematics of Operations Research* 8(4): 538–548.

Mnich, M.; and van Bevern, R. 2018. Parameterized complexity of machine scheduling: 15 open problems. *Computers & Operations Research* 100: 254–261.

Mnich, M.; and Wiese, A. 2015. Scheduling and fixed-parameter tractability. *Mathematical Programming* 154(1-2): 533–562.

Niedermeier, R. 2006. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press.

Slowinski, R. 1984. Preemptive scheduling of independent jobs on parallel machines subject to financial constraints. *European Journal of Operational Research* 15: 366–373.

Toker, A.; Kondakci, S.; and Erkip, N. 1991. Scheduling under a non-renewable resource constraint. *Journal of the Operational Research Society* 42: 811–814.

Xie, J. 1997. Polynomial algorithms for single machine scheduling problems with financial constraints. *Operations Research Letters* 21(1): 39–42.