Math. Program., Ser. B (2015) 154:533–562
DOI 10.1007/s10107-014-0830-9

FULL LENGTH PAPER

Scheduling and fixed-parameter tractability

Matthias Mnich · Andreas Wiese

Received: 24 April 2014 / Accepted: 10 October 2014 / Published online: 4 November 2014

Abstract Fixed-parameter tractability analysis and scheduling are two core domains of combinatorial optimization which led to deep understanding of many important algorithmic questions. However, even though fixed-parameter algorithms are appealing for many reasons, no such algorithms are known for many fundamental scheduling problems. In this paper we present the first fixed-parameter algorithms for classical scheduling problems such as makespan minimization, scheduling with job-dependent cost functions—one important example being weighted flow time—and scheduling with rejection. To this end, we identify crucial parameters that determine the problems' complexity. In particular, we manage to cope with the problem complexity stemming from numeric input values, such as job processing times, which is usually a core bottleneck in the design of fixed-parameter algorithms. We complement our algorithms with W[1]-hardness results showing that for smaller sets of parameters the respective problems do not allow fixed-parameter algorithms. In particular, our positive and negative results for scheduling with rejection explore a research direction proposed by Dániel Marx.

Keywords Scheduling · Fixed-parameter tractability · Integer linear programming

Mathematics Subject Classification 68W05 · 90B35

A preliminary version of this article appeared in the proceedings of IPCO 2014 [34].

M. Mnich
Cluster of Excellence MMCI, Campus E1-7, 66123 Saarbrücken, Germany
e-mail: m.mnich@mmci.uni-saarland.de

A. Wiese (✉)
Max-Planck-Institute for Computer Science, Campus E1-4, 66123 Saarbrücken, Germany
e-mail: awiese@mpi-inf.mpg.de

1 Introduction

Scheduling and fixed-parameter tractability are two very well-studied research areas. In scheduling, the usual setting is that one is given a set of machines and a set of jobs with individual characteristics. The jobs need to be scheduled on the machines according to some problem-specific constraints, such as release dates, precedence constraints, or rules regarding preemption and migration. Typical objectives are minimizing the global makespan, the weighted sum of completion times of the jobs, or the total flow time. During the last decades of research on scheduling, many important algorithmic questions have been settled. For instance, for minimizing the makespan and the weighted sum of completion time on identical machines, $(1+\epsilon)$-approximation algorithms (PTASs) are known for almost all NP-hard settings [1,25].

However, the running time of these approximation schemes usually has a bad dependence on ϵ, and in practice exact algorithms are often desired. These and other considerations motivate to study which scheduling problems are *fixed-parameter tractable* (FPT), which amounts to identifying instance-dependent parameters k that allow for algorithms that find optimal solutions in time $f(k) \cdot n^{O(1)}$ for instances of size n and some function f depending only on k. Separating the dependence of k and n is often much more desirable than a running time of, e.g., $O(n^k)$, which becomes infeasible even for small k and large n. The parameter k measures the complexity of a given instance and thus, problem classification according to parameters yields an instance-depending measure of problem hardness.

Despite the fundamental nature of scheduling problems, and the clear advantages of fixed-parameter algorithms, to the best of our knowledge no such algorithms are known for the classical scheduling problems we study here. One obstacle towards obtaining positive results appears to be that—in contrast to most problems known to be fixed-parameter tractable—scheduling problems involve many numerical input data (e.g., job processing times, release dates, job weights), which alone render many problems NP-hard, thus ruling out fixed-parameter algorithms. One contribution of this paper is that—for the fundamental problems studied here—choosing the number of distinct numeric values or an upper bound on them as the parameter suffices to overcome this impediment. Note that this condition is much weaker than assuming the parameter to be bounded by a constant (that can appear in the exponent of the run time).

We hope that our work gives rise to a novel perspective on scheduling as well as on FPT, yielding further interesting research on fixed-parameter tractability of the many well-established and important scheduling problems.

1.1 Our contributions

In this paper we present the first fixed-parameter algorithms for several fundamental scheduling problems. In Sect. 2 we study one of the most classical scheduling problems, which is minimizing the makespan on an arbitrary number of machines without preemption, i.e. the problem $P||C_{\max}$. Assuming integral input data, our parameter $p_{\max}$ defines an upper bound on the job processing times appearing in an instance with n jobs. We first prove that for any number of machines, we can restrict ourselves to (optimal) solutions where jobs of the same length are almost equally distributed among the machines, up to an additive error term of $\pm f(p_{\max})$ jobs. This insight can be used as an independent preprocessing routine which optimally assigns the majority of the jobs of an instance (given that $n \gg p_{\max}$). After this preparation, we show that the remaining problem can be formulated as an integer program in fixed dimension, yielding an overall running time bounded by $f(p_{\max}) \cdot n^{O(1)}$. We note that without a fixed parameter, the problem is strongly NP-hard. For the much more general machine model of unrelated machines, we show that $R||C_{\max}$ is fixed-parameter tractable when choosing the number of machines and the number of distinct processing times as parameters. We reduce this problem again to integer programming in fixed dimension.

We remark that these problems are sufficiently complex so that we do not see a way of using the "number of numbers" result by Fellows et al. [17]. Note that if the number of machines or the number of processing times are constant, the problem is still NP-hard [31], and thus no fixed-parameter algorithms can exist for those cases, unless $\mathsf{P} = \mathsf{NP}$.

Then, in Sect. 3, we study scheduling with rejection. Each job j is specified by a processing time p_j, a weight w_j, and a rejection cost e_j (all jobs are released at time zero). We want to reject a set J' of at most k jobs, and schedule all other jobs on one machine to minimize $\sum_{j \notin J'} w_j C_j + \sum_{j \in J'} e_j$. We identify three key parameters: the number of distinct processing times, the number of distinct weights, and the maximum number k of jobs to be rejected. We show that if any two of the three values are taken as parameters, the problem becomes fixed-parameter tractable. If k and either of the other two are parameters, then we show that an optimal solution is characterized by one of sufficiently few possible patterns of jobs to be rejected. Once we guessed the correct pattern, an actual solution can be found by a dynamic program efficiently. If the number of distinct processing times and lengths are parameters (but not k), we provide a careful modeling of the problem as an integer program with convex objective function in fixed dimension. Here, we need to take particular care to incorporate the rejection costs as bounded-degree polynomials, to be able to use the deep theory of solving convex programs in fixed dimension efficiently. To the best of our knowledge, this is the first time that convex programming is used in fixed-parameter algorithms. We complement this result by showing that if only the number of rejected jobs k is the fixed parameter, then the problem becomes W[1]-hard, which prohibits the existence of a fixed-parameter algorithm, unless FPT = W[1] (which would imply subexponential time algorithms for many canonical NP-complete problems such as 3-SAT, which would in particular refute the Exponential Time Hypothesis). Our results respond to a question by Marx [32] for investigating the fixed-parameter tractability of scheduling with rejection.

Finally, in Sect. 4 we turn our attention to the parametrized dual of the latter problem: scheduling with rejection of at least $n - s$ jobs (s being the parameter). We reduce this to a much more general problem which can be cast as the profit maximization version of the *general scheduling problem* (*GSP*) [5]. We need to select a subset J' of at most s jobs to schedule from a given set J, and each scheduled job j yields a profit $f_j(C_j)$, depending on its completion time C_j. Note that this function can be different for each job and might stem from a difficult scheduling objective such as

Table 1 Summary of our results

Problem	Parameters	Result
$P\|\|C_{\max}$	maximum p_j	FPT
$R\|\|C_{\max}$	#distinct p_j and #machines	FPT
$1\|\| \sum_{\leq k} e_j + \sum w_j C_j$	#rejected jobs k and #distinct p_j	FPT
$1\|\| \sum_{\leq k} e_j + \sum w_j C_j$	#rejected jobs k and #distinct w_j	FPT
$1\|\| \sum_{\leq k} e_j + \sum w_j C_j$	#distinct p_j and #distinct w_j	FPT
$1\|\| \sum_{\leq k} e_j + \sum w_j C_j$	#rejected jobs k	W[1]-hard
$1\|r_j, (pmtn)\| \max \sum_{\leq s} f_j(C_j)$	#selected jobs s and #distinct p_j	FPT
$1\|r_j, (pmtn)\| \max \sum_{\leq s} f_j(C_j)$	#selected jobs s	W[1]-hard
$1\|r_j, (pmtn)\| \max \sum_{\leq s} f_j(C_j)$	#distinct p_j (in fact $\forall\, p_j \in \{1, 3\}$)	para-NP-hard

Here, for job j by w_j denotes its weight, by e_j its rejection cost, by C_j its completion time and by f_j its cost function

weighted flow time. Additionally, each job j has a release date r_j and a processing time p_j. The goal is to schedule these jobs on one machine to maximize $\sum_{j\in J'} f_j(C_j)$. We study the preemptive as well as the non-preemptive version of this problem. In its full generality, GSP is not well understood. Despite that, we are able to give a fixed-parameter algorithm if the number of distinct processing times is bounded by a parameter, as well as the maximum cardinality of J'. We complement our findings by showing that for fewer parameters the problem is W[1]-hard or para-NP-hard (so NP-hard for constant parameter value), respectively. Our contributions are summarized in Table 1.

1.2 Related work

1.2.1 Scheduling

One very classical scheduling problem studied in this paper is to schedule a set of jobs non-preemptively on a set of m identical machines, i.e., $P||C_{\max}$. Research for it dates back to the 1960s when Graham showed that the greedy list scheduling algorithm yields a $(2 - \frac{1}{m})$-approximation and a 4/3-approximation when the jobs are ordered non-decreasingly by length [23]. After a series of improvements [14,20,29,35], Hochbaum and Shmoys present a polynomial time approximation scheme (PTAS), even if the number of machines is part of the input [25]. On unrelated machines, the problem is NP-hard to approximate with a better factor than 3/2 [15,31] and there is a 2-approximation algorithm [31] that extends to the generalized assignment problem [36]. For the restricted assignment case, i.e., each job has a fixed processing time and a set of machines where one can assign it to, Svensson [38] gives a polynomial time algorithm that estimates the optimal makespan up to a factor of $33/17+\epsilon \approx 1.9412+\epsilon$.

For scheduling jobs with release dates preemptively on one machine, a vast class of important objective functions is captured by the GSP. In its full generality, Bansal and Pruhs [5,6] give a $O(\log\log P)$-approximation, where P is the maximum ratio of processing times. One particularly important special case is the weighted flow time objective where prior to this result the best known approximation factors where $O(\log^2 P)$, $O(\log W)$, and $O(\log nP)$ [4,11]; where W is the maximum ratio of job weights. Also, a quasi-PTAS with running time $n^{O(\log P \log W)}$ is known [10]. Since the best known complexity result is NP-hardness much remains unclear. A much better understood classical objective in the GSP framework is minimizing $\sum_j w_j C_j$, i.e., the weighted sum of completion time; here, even on an arbitrary number of identical machines an EPTAS is known [1].

A generalization of classical scheduling problems is *scheduling with rejection*. There, each job j additionally has a *rejection cost* e_j. The scheduler has the freedom to reject job j and to pay a penalty of e_j, in addition to some (ordinary) objective function for the scheduled jobs. For one machine and the objective being to minimize the sum of weighted completion times, Engels et al. [16] give an optimal pseudopolynomial dynamic program for the case that all jobs are released at time zero and show that the problem is weakly NP-hard. Sviridenko and Wiese [39] give a PTAS for arbitrary release dates. For the objective to minimize the makespan plus the rejection cost on multiple machines, Hoogeveen et al. [26] give FPTASs for almost all machine settings, and a 1.58-approximation for the APX-hard case of an arbitrary number of unrelated machines (when allowing preemption).

In high-multiplicity scheduling, one considers the setting where there are only few different job types, with jobs of the same type appearing in large bulks; one might consider the number of job types as a fixed parameter. We refer to the survey by Brauner et al. [9] for more information on high-multiplicity scheduling. Also, recently BIN PACKING with a constant number of item sizes was shown to be solvable in polynomial time [22], implying the same for $P||C_{\max}$ with constantly many processing times. However, this result does not give a fixed-parameter algorithm.

1.2.2 Fixed-parameter tractability

Until now, to the best of our knowledge, no fixed-parameter algorithms for the classical scheduling problems studied in this paper have been devised. In contrast, classical scheduling problems investigated in the framework of parameterized complexity appear to be intractable; for example, k-processor scheduling with precedence constraints is W[2]-hard [8] and scheduling unit-length tasks with deadlines and precedence constraints and k tardy tasks is W[1]-hard [19], for parameter k. Also, Jansen et al.'s [27] W[1]-hardness result for *unary* bin packing parameterized by the number k of bins implies that makespan minimization on k machines with jobs of polynomially bounded processing times is W[1]-hard. Mere exemptions seem to be an algorithm by Marx and Schlotter [33] for makespan minimization where k jobs have processing time $p \in \mathbb{N}$ and all other jobs have processing time 1, for combined parameter (k, p), as well as work of Alon et al. [2] who show that makespan minimization on m identical machines is fixed-parameter tractable parameterized by the optimal makespan. We also mention that Chu et al. [12] consider the parameterized complexity of checking *feasibility* of a schedule (rather than optimization). We remark that some scheduling-type problems can be addressed by choosing as parameter the "number of numbers", as done by Fellows et al. [17]. Finally, Bessy and Giroudeau [7] show fixed-parameter tractability for the parameter k of tardy jobs in a model where each task consists of two sub-tasks all of the same length and delayed by the same constant amount of idle time.

A potential reason for this lack of positive results (fixed-parameter algorithms) might be that the knowledge of fixed-parameter algorithms for weighted problems is still in a nascent stage, and scheduling problems are inherently weighted, having job processing times, job weights, etc.

2 Minimizing the makespan

2.1 Identical machines

We first consider the problem $P||C_{\max}$, where a given set J of n jobs (with individual processing time p_j and released at time zero) must be scheduled non-preemptively on a set of m identical machines, as to minimize the makespan of the schedule. We develop a fixed-parameter algorithm for solving this problem in time $f(p_{\max}) \cdot n^{O(1)}$, where $p_{\max}$ is the maximum processing time over all jobs.

In the sequel, we say that some job j is of *type t* if $p_j = t$; let $J_t := \{j \in J \mid p_j = t\}$. First, we prove that there is always an optimal solution in which each machine has almost the same number of jobs of each type, up to an additive error of $\pm f(p_{\max})$ for suitable function f. This allows us to fix some jobs on the machines. For the remaining jobs, we show that each machine receives at most $2f(p_{\max})$ jobs of each type; hence there are only $(2f(p_{\max}))^{p_{\max}}$ possible configurations for each machine. We solve the remaining problem with an integer linear program in fixed dimension.

As a first step, for each type t, we assign $\left\lfloor \frac{|J_t|}{m} \right\rfloor - f(p_{\max})$ jobs of type t to each machine; let $J_0 \subseteq J$ be this set of jobs. This is justified by the next lemma, in whose proof we start with an arbitrary optimal schedule and exchange jobs carefully between the machines until the claimed property holds.

Lemma 1 *There is a function $f : \mathbb{N} \to \mathbb{N}$ with $f(p_{\max}) \leq 2^{O(p_{\max} \cdot \log p_{\max})}$ for all $p_{\max} \in \mathbb{N}$ such that every instance of $P||C_{\max}$ admits an optimal solution in which for each type t, each of the m machines schedules at least $\lfloor |J_t|/m \rfloor - f(p_{\max})$ and at most $\lfloor |J_t|/m \rfloor + f(p_{\max})$ jobs of type t.*

Proof For each machine i denote by $J_t^{(i)}$ the set of jobs of type t scheduled on i in some (optimal) schedule. We prove the following (more technical) claim: there always exists an optimal solution in which for every pair of machines i and i', and for each $\ell \in \{1, \ldots, p_{\max}\}$, we have that $||J_\ell^{(i)}| - |J_\ell^{(i')}|| \leq h(\ell) \cdot g(p_{\max})$, where $g(p_{\max}) := (p_{\max})^3 + p_{\max}$ and $h(\ell)$ is inductively defined by setting $h(p_{\max}) := 1$ and $h(\ell) := 1 + \sum_{j=\ell+1}^{p_{\max}} j \cdot h(j)$.

Suppose that there are two machines i, i' such that there is a value ℓ for which $||J_\ell^{(i)}| - |J_\ell^{(i')}|| > h(\ell) \cdot g(p_{\max})$. Assume, without loss of generality, that $|J_\ell^{(i)}| - |J_\ell^{(i')}| > h(\ell) \cdot g(p_{\max})$ and $||J_{\ell'}^{(\bar{i})}| - |J_{\ell'}^{(\bar{i}')}|| \leq h(\ell') \cdot g(p_{\max})$ for all $\ell' > \ell$ and all machines $\bar{i}, \bar{i}'$. We show that i' has at least a certain volume of jobs which are shorter than ℓ. We calculate that

$$\begin{aligned}
\sum_{j=1}^{\ell-1} j \cdot |J_j^{(i')}| &= \left(\sum_{j=1}^{p_{\max}} j \cdot |J_j^{(i')}|\right) - \ell \cdot |J_\ell^{(i')}| - \left(\sum_{j=\ell+1}^{p_{\max}} j \cdot |J_j^{(i')}|\right)\\
&\geq \left(-p_{\max} + \sum_{j=1}^{p_{\max}} j \cdot |J_j^{(i)}|\right) - \ell \cdot |J_\ell^{(i')}| - \left(\sum_{j=\ell+1}^{p_{\max}} j \cdot |J_j^{(i')}|\right)\\
&\geq \left(-p_{\max} + \sum_{j=1}^{p_{\max}} j \cdot |J_j^{(i)}|\right) - \ell \cdot |J_\ell^{(i')}|\\
&\quad - \left(\sum_{j=\ell+1}^{p_{\max}} j \cdot (|J_j^{(i)}| + h(j) \cdot g(p_{\max}))\right)\\
&> -p_{\max} + \sum_{j=1}^{\ell} j \cdot |J_j^{(i)}| + \ell \cdot (h(\ell) \cdot g(p_{\max}) - |J_\ell^{(i)}|)\\
&\quad - \sum_{j=\ell+1}^{p_{\max}} j \cdot h(j) \cdot g(p_{\max})\\
&= -p_{\max} + \sum_{j=1}^{\ell-1} j \cdot |J_j^{(i)}| + \ell \cdot h(\ell) \cdot g(p_{\max})\\
&\quad - \sum_{j=\ell+1}^{p_{\max}} j \cdot h(j) \cdot g(p_{\max})\\
&\geq (p_{\max})^3,
\end{aligned}$$

where the first inequality stems from the fact that the considered schedule is optimal and therefore the loads of any two machines can differ by at most $p_{\max}$. Hence, there must be at least one type $\ell' < \ell$ such that $\ell' \cdot |J_\ell^{(i')}| \geq (p_{\max})^2$, and thus $|J_\ell^{(i')}| \geq p_{\max}$. Since the least common multiple of any two values in $\{1, \ldots, p_{\max}\}$ is at most $(p_{\max})^2$, we can swap $r \in \{1, \ldots, p_{\max}\}$ jobs of type ℓ from machine M_i with $r' \in \{1, \ldots, p_{\max}\}$ jobs of type ℓ' from machine $M_{i'}$, without changing the total load of any of the two machines. By continuing inductively we obtain a schedule satisfying $||J_{\ell'}^{(i)}| - |J_{\ell'}^{(i')}|| \leq h(\ell') \cdot g(p_{\max})$ for any two machines i, i' and any type $\ell' \geq \ell$. Inductively, we obtain an optimal schedule satisfying the claim for all types ℓ. Thus, the claim of the lemma holds for the function $f(p_{\max}) := h(1) \cdot g(p_{\max})$.

We now prove the claimed upper bound on f. To this end, consider first the function h as recursively defined above. Notice that h is parametrized by ℓ, but also depends on $p_{\max}$. We prove, by induction on ℓ, that $h(\ell) = \frac{(p_{\max}+1)!}{(\ell+1)!}$. The base case is for $\ell = p_{\max}$, and it follows from the definition that $h(p_{\max}) = 1 = \frac{(p_{\max}+1)!}{(p_{\max}+1)!}$. For the inductive step, let $\ell < p_{\max}$ and take as inductive hypothesis that $h(\ell') = \frac{(p_{\max}+1)!}{(\ell'+1)!}$ for all $\ell' \in \{\ell + 1, \ldots, p_{\max}\}$. We then have

$$\begin{aligned}
h(\ell) &= 1 + \sum_{j=\ell+1}^{p_{\max}} j \cdot h(j) \\
&= 1 + (\ell+1)h(\ell+1) + \sum_{j=\ell+2}^{p_{\max}} j \cdot h(j) \\
&= 1 + (\ell+1)h(\ell+1) + (h(\ell+1) - 1) \\
&= (\ell+2)h(\ell+1) \\
&= (\ell+2)\frac{(p_{\max}+1)!}{(\ell+2)!} \\
&= \frac{(p_{\max}+1)!}{(\ell+1)!},
\end{aligned}$$

as claimed.

Thus, the claim of the lemma holds for $f(p_{\max}) := h(1) \cdot g(p_{\max})$, and therefore we have $f(p_{\max}) \leq ((p_{\max}+1)!)((p_{\max})^3 + p_{\max}) = 2^{O(p_{\max} \log p_{\max})}$. □

Denote by $J' = J \setminus J_0$ the set of yet unscheduled jobs. We ignore all other jobs from now on. By Lemma 1, there is an optimal solution in which each machine receives at most $2 \cdot f(p_{\max}) + 1$ jobs from each type. Hence, there are at most $(2 \cdot f(p_{\max}) + 2)^{p_{\max}}$ ways how the schedule for each machine can look like (up to permuting jobs of the same length). Therefore, the remaining problem can be solved with the following integer program. Define a set $\mathcal{C} = \{0, \ldots, 2 \cdot f(p_{\max}) + 1\}^{p_{\max}}$ of at most $(2 \cdot f(p_{\max}) + 2)^{p_{\max}}$ "configurations", where each *configuration* is a vector $C \in \mathcal{C}$ encoding the number of jobs from J' of each type assigned to a machine.

In any optimal solution for J', the makespan is in the range $\{\lceil p(J')/m \rceil, \ldots, \lceil p(J')/m \rceil + p_{\max}\}$, where $p(J') = \sum_{j \in J'} p_j$, as $p_j \leq p_{\max}$ for each j. For each value T in this range we try whether $\mathsf{opt} \leq T$, where opt denotes the minimum makespan of the instance. So fix a value T. We allow only configurations $C = (c_1, \ldots, c_{p_{\max}})$ which satisfy $\sum_{i=1}^{p_{\max}} c_i \cdot i \leq T$; let $\mathcal{C}(T)$ be the set of these configurations. For each $C \in \mathcal{C}(T)$, introduce a variable y_C for the number of machines with configuration C in the solution. (As the machines are identical, only the number of machines following each configuration is important.)

$$\sum_{C \in \mathcal{C}(T)} y_C \leq m \tag{1}$$

$$\sum_{C=(c_1,\ldots,c_{p_{\max}}) \in \mathcal{C}(T)} y_C \cdot c_p \geq |J' \cap J_p|, \quad p = 0, \ldots, p_{\max} \tag{2}$$

$$y_C \in \{0, \ldots, m\}, \; C \in \mathcal{C}(T) \tag{3}$$

Inequality (1) ensures that at most m machines are used, inequalities (2) ensure that all jobs from each job type are scheduled. The whole integer program (1)–(3) has at most $(2 \cdot f(p_{\max}) + 2)^{p_{\max}}$ dimensions.

To determine feasibility of (1)–(3), we employ results about integer programming in fixed dimension. As we will need it later, we cite here an algorithm due to Heinz [24,28] that even allows (quasi-)convex polynomials as cost functions, rather than only linear functions. While the functions we will use are all convex, note that a function is called *quasi-convex* if every lower level set $L_\lambda = \{x \in \mathbb{R}^n : g(x) \le \lambda\}$ is a convex set.

Theorem 1 ([24,28]) *Let $\bar{f}, \bar{g}_1, \ldots, \bar{g}_m \in \mathbb{Z}[x_1, \ldots, x_t]$ be quasi-convex polynomials of degree at most $d \ge 2$, whose coefficients have binary encoding length at most ℓ. There is an algorithm that in time $m \cdot \ell^{O(1)} \cdot d^{O(t)} \cdot 2^{O(t^3)}$ computes a minimizer $\mathbf{x}^\star \in \mathbb{Z}^t$ with binary encoding size $\ell \cdot d^{O(t)}$ of the following problem*

$$\min \bar{f}(x_1, \ldots, x_t), \quad \textit{subject to } \bar{g}_i(x_1, \ldots, x_t) \le 0, \quad i = 1, \ldots, m \quad \mathbf{x} \in \mathbb{Z}^t, \tag{4}$$

or reports that no minimizer exists.

The smallest value T for which (1)–(3) is feasible gives the optimal makespan and together with the preprocessing routine of Lemma 1 yields an optimal schedule. Thus, we have proven the following.

Theorem 2 *There is a function $\tilde{f}$ such that instances of $P||C_{\max}$ with n jobs and m machines can be solved in time $\tilde{f}(p_{\max}) \cdot (n+m)^{O(1)}$.*

Recall that without choosing a parameter, problem $P||C_{\max}$ is strongly NP-hard (as it contains 3- Partition).

A natural extension to consider is the problem $P||C_{\max}$ parameterized by the number $\overline{p}$ of distinct processing times. Unfortunately, for this extension Lemma 1 is no longer true as the following example shows. Let q_1, q_2 be two different (large) prime numbers. Consider an instance with two identical machines M_1, M_2, and q_1 many jobs with processing time q_2, and similarly q_2 many jobs with processing time q_1. The optimal makespan is $T := q_1 \cdot q_2$ which is achieved by assigning all jobs with processing time q_1 on M_1 and all other jobs on M_2, giving both machines a makespan of exactly T. However, apart from swapping the machines this is the only optimal schedule since the equation $q_1 \cdot q_2 = x_1 \cdot q_1 + x_2 \cdot q_2$ only allows integral solutions (x_1, x_2) such that x_1 is a multiple of q_2 and x_2 is a multiple of q_1.

On the other hand, for constantly many processing times the problem was recently shown to be polynomial time solvable for any constant $\overline{p}$ [22].

2.2 Bounded number of unrelated machines

We study the problem $Rm||C_{\max}$ where now the machines are unrelated, meaning that a job can have different processing times on different machines. In particular, it might be that a job cannot be processed on some machine at all, i.e., has infinite processing time on that machine.

We choose as parameters the number $\overline{p}$ of distinct (finite) processing times, and the number m of machines of the instance. This choice of parameters is motivated as follows. If only the number of machines is a fixed parameter, then already $P||C_{\max}$ is W[1]-hard, even if all processing times are polynomially bounded [27]. On the other hand, $R||C_{\max}$ is NP-hard if only processing times $\{1, 2, \infty\}$ are allowed [15,31]. This justifies to take both m and $\bar{p}$ as a parameters in the unrelated machine case.

We model the problem as an integer program in fixed dimension. We say that two jobs j, j' are of the same *type* if $p_{i,j} = p_{i,j'}$ for each machine i. Note that there are only $(\bar{p}+1)^m$ different types of jobs. Denote by Z the set of all job types, and for each $z \in Z$ denote by n_z the number of jobs of type z. Moreover, for each type $z \in Z$ and each machine i denote by $q_{i,z}$ the processing time of the jobs of type z on machine i. For each combination of a type z and a machine i we introduce an integral variable $y_{i,z} \in \{0, \ldots, n\}$ which models how many jobs of type z are assigned to machine i. The total number of these variables is then bounded by $m \cdot (\bar{p}+1)^m$. This allows us to formulate our problem by the following integer program:

$$\min\ T \tag{5}$$

$$\text{s.t.}\ \sum_{z \in Z} y_{i,z} \cdot q_{i,z} \leq T \quad i = 1, \ldots, m \tag{6}$$

$$\sum_{i=1}^{m} y_{i,z} = n_z \quad \forall z \in Z \tag{7}$$

$$y_{i,z} \in \{0, \ldots, n\} \quad i = 1, \ldots, m,\ z \in Z \tag{8}$$

$$T \geq 0 \tag{9}$$

The above integer program (5)–(9) has only $m \cdot (\bar{p}+1)^m + 1$ dimensions and $m \cdot (\bar{p}+1)^m + m$ constraints. We solve it using Theorem 1. The inequalities (6) ensure that the makespan on each machine is bounded by T. Inequalities (7) ensure that for each type $z \in Z$ all its jobs are assigned. This yields the following theorem.

Theorem 3 *Instances of $R||C_{\max}$ with m machines and n jobs with $\overline{p}$ distinct finite processing times can be solved in time $f(\overline{p}, m) \cdot (n + \max_i \max_j \log p_{i,j})^{O(1)}$ for a suitable function f.*

Proof Any feasible schedule implies a solution (y, T) of the above integer program such that T equals the makespan of the schedule and each variable $y_{i,z}$ equals the (integral) number of jobs of type z that are assigned on machine i (for each machine i and each type z). Conversely, given a solution (y, T) of the integer program we can obtain a solution with makespan T by assigning the jobs of each type $z \in Z$ arbitrarily on the machines, subject to the constraint that each machine i gets at most $y_{i,z}$ jobs of type z. Due to inequalities (7), this procedure assigns all jobs and due to inequalities (6) the resulting makespan is indeed at most T. □

3 Scheduling with rejection

In this section we study scheduling with rejection to optimize the weighted sum of completion time plus the total rejection cost, i.e, $1||\sum_{\leq k} e_j + \sum w_j C_j$. Formally, we are given an integer k and a set J of n jobs, all released at time zero. Each job $j \in J$ is characterized by a processing time $p_j \in \mathbb{N}$, a weight $w_j \in \mathbb{N}$ and rejection cost $e_j \in \mathbb{N}$. The goal is to reject a set $J' \subseteq J$ of at most k jobs and to schedule all other jobs non-preemptively on a single machine, as to minimize $\sum_{j \in J \setminus J'} w_j C_j + \sum_{j \in J'} e_j$. Note that if rejection is not allowed ($k = 0$), the problem is solved optimally by scheduling jobs according to non-decreasing Smith ratios w_j/p_j, breaking ties arbitrarily [37].

3.1 Number of rejected jobs and processing times or weights

Denote by $\overline{p} \in \mathbb{N}$ the number of distinct processing times in a given instance. First, we assume that $\overline{p}$ and the maximum number k of rejected jobs are parameters. Thereafter, using a standard reduction, we will derive an algorithm for the case that k and the number $\overline{w}$ of distinct weights are parameters.

Denote by $q_1, \ldots, q_{\overline{p}}$ the distinct processing times in a given instance. For each $i \in \{1, \ldots, \overline{p}\}$, we guess the number of jobs with processing time q_i which are rejected in an optimal solution. Each possible guess is characterized by a vector $\mathbf{v} = \{v_1, \ldots, v_{\bar{p}}\}$ whose entries v_i contain integers between 0 and k, and whose total sum is at most k. There are at most $(k+1)^{\overline{p}}$ such vectors $\mathbf{v}$, each one prescribing that at most v_i jobs of processing time p_i can be rejected. We enumerate them all. One of these vectors must correspond to the optimal solution, so the reader may assume that we know this vector $\mathbf{v}$.

In the following, we will search for the optimal schedule that *respects* $\mathbf{v}$, meaning that for each $i \in \{1, \ldots, \overline{p}\}$ at most v_i jobs of processing time q_i are rejected. To find an optimal schedule respecting $\mathbf{v}$, we use a dynamic program. Suppose the jobs in J are labeled by $1, \ldots, n$ by non-increasing *Smith ratios* w_j/p_j. Each dynamic programming cell is characterized by a value $n' \in \{0, \ldots, n\}$, and a vector $\mathbf{v}'$ with $\overline{p}$ entries which is *dominated* by $\mathbf{v}$, meaning that $v'_i \leq v_i$ for each $i \in \{1, \ldots, \overline{p}\}$. For each pair $(n', \mathbf{v}')$ we have a cell $C(n', \mathbf{v}')$ modeling the following subproblem. Assume that for jobs in $J' := \{1, \ldots, n'\}$ we have already decided whether we want to schedule them or not. For each processing time q_i denote by n'_i the number of jobs in J' with processing time q_i. Assume that for each type i, we have decided to reject $v_i - v'_i$ jobs from J'. Note that then the total processing time of the scheduled jobs sums up to $t := \sum_i q_i \cdot (n'_i - (v_i - v'_i))$. It remains to define a solution for the jobs in $J'' := \{n'+1, \ldots, n\}$ during time interval $[t, \infty)$, such that for each type i we can reject up to v'_i jobs. The problem described by each cell $C(n', \mathbf{v}')$ can be solved in polynomial time, given one has already computed the values for each cell $C(n'', \mathbf{v}'')$ with $n'' > n'$:

Lemma 2 *Let $C(n', \mathbf{v}')$ be a cell and let $\mathsf{opt}(n', \mathbf{v}')$ be the optimal solution value to its subproblem. Let $i \in \{1, \ldots, \overline{p}\}$ be such that $p_{n'+1} = q_i$, and let $t := \sum_i q_i \cdot (n'_i - (v_i - v'_i))$. If $v'_i = 0$ then*

$$\mathsf{opt}(n', (v'_1, \ldots, v'_i, \ldots, v'_{\overline{p}})) = \mathsf{opt}(n'+1, (v'_1, \ldots, v'_i, \ldots, v'_{\overline{p}})) + (t + p_{n'+1}) \cdot w_{n'},$$

$$\begin{aligned} &\textit{otherwise } \mathsf{opt}(n', (v'_1, \ldots, v'_i, \ldots, v'_{\overline{p}})) \\ &= \min\Big\{\mathsf{opt}(n'+1, (v'_1, \ldots, v'_i, \ldots, v'_{\overline{p}})) + (t + p_{n'+1}) \cdot w_{n'}, \\ &\quad \mathsf{opt}(n'+1, (v'_1, \ldots, v'_i - 1, \ldots, v'_{\overline{p}})) + e_{n'+1}\Big\}. \end{aligned}$$

Proof First, suppose that $\mathsf{opt}(n', \mathbf{v}')$ schedules job n'. Due to our sorting of jobs, n' has the largest Smith ratio of all jobs in $\{n', \dots, n\}$, and an optimal schedule with value $\mathsf{opt}(n', \mathbf{v}')$ schedules n' at time t at cost $(t + p_{n'}) \cdot w_{n'}$, and the optimal solution value to the remaining subproblem is given by $\mathsf{opt}(n' + 1, (v'_1, \dots, v'_i, \dots, v'_\ell))$. If $\mathsf{opt}(n', \mathbf{v}')$ rejects n' (which can only happen if $v'_i > 0$) then $\mathsf{opt}(n', \mathbf{v}') = \mathsf{opt}(n' + 1, (v'_1, \dots, v'_i - 1, \dots, v'_\ell)) + e_{n'}$.

On the other hand, there is a solution for cell $C(n', \mathbf{v}')$ given by scheduling n' at time t and all other non-rejected jobs in the interval $[t + p_{n'}, \infty)$ at cost $\mathsf{opt}(n' + 1, (v'_1, \dots, v'_i, \dots, v'_\ell)) + (t + p_{n'}) \cdot w_{n'}$. If $v'_i > 0$ there is also a feasible solution with cost $\mathsf{opt}(n'+1, (v'_1, \dots, v'_i - 1, \dots, v'_\ell)) + e_{n'}$ which rejects job n' and schedules all other non-rejected jobs during the interval $[t, \infty)$. □

The size of the dynamic programming table is bounded by $n \cdot (k + 1)^{\overline{p}}$. Since for $1||\sum w_j C_j$ one can interchange weights and processing times and get an equivalent instance [13, Theorem 3.1], we obtain the same result when there are only $\bar{p}$ distinct weights.

Theorem 4 *For sets J of n jobs with $\overline{p}$ distinct processing times or weights, the problem $1||\sum_{\leq k} e_j + \sum w_j C_j$ is solvable in time $O(n \cdot (k + 1)^{\overline{p}} + n \cdot \log n)$.*

We show next that when only the number k of rejected jobs is taken as parameter, the problem becomes $\mathsf{W}[1]$-hard. (This requires that the numbers in the input can be super-polynomially large. Note that for polynomially bounded processing times the problem admits a polynomial time algorithm for arbitrary k [16].) This justifies to define additionally the number of weights or processing times as parameter. We remark that when jobs have non-trivial release dates, then even for $k = 0$ the problem is NP-hard [30].

Theorem 5 *Problem $1||\sum_{\leq k} e_j + \sum w_j C_j$ is $\mathsf{W}[1]$-hard for parameter the number k of rejected jobs.*

Proof We reduce from k- Subset Sum, for which the input consists of integers $s_1, \dots, s_n \in \mathbb{N}$ and two values $k, q \in \mathbb{N}$. The goal is to select a subset of k of the given integers s_i that sum up to q. Parameterized by k, this problem is known to be $\mathsf{W}[1]$-hard [18]. Our reduction mimics closely a reduction from (ordinary) Subset Sum in [16] that shows that $1||\sum e_j + \sum w_j C_j$ is weakly NP-hard.

Suppose we are given an instance of k- Subset Sum. Let $S = \sum_{i=1}^{n} s_i$. We construct an instance of $1||\sum_{\leq k} e_j + \sum w_j C_j$ with n jobs, where each job j has processing time $p_j := s_j$, weight $w_j := s_j$, rejection cost $e_j := (S - q) \cdot s_j + \frac{1}{2} s_j^2$. Since $p_j = w_j$ for all jobs j, the ordering of the scheduled jobs $J \setminus J'$ does not affect the value of $\sum_{j \in J \setminus J'} w_j C_j$. Using this fact and substituting for the rejection penalty, we can rewrite the objective function as follows:

$$\begin{aligned}\sum_{j\in J\setminus J'} w_jC_j + \sum_{j\in J'} e_j &= \sum_{j\in J\setminus J'} s_j \sum_{i\le j, i\in J\setminus J'} s_i + \sum_{j\in J'} e_j\\ &= \sum_{j\in J\setminus J'} s_j^2 + \sum_{j<i, i,j\in J\setminus J'} s_js_i + \sum_{j\in J'}\left((S-q)\cdot s_j + \frac{1}{2}s_j^2\right)\\ &= \frac{1}{2}\left[\left(\sum_{j\in J\setminus J'} s_j\right)^2 + \sum_{j\in J\setminus J'} s_j^2\right] + (S-q)\sum_{j\in J'} s_j + \frac{1}{2}\sum_{j\in J'} s_j^2\\ &= \frac{1}{2}\left(\sum_{j\in J\setminus J'} s_j\right)^2 + (S-q)\sum_{j\in J'} s_j + \frac{1}{2}\sum_{j=1}^{n} s_j^2 .\end{aligned}$$

Since $\sum_{j=1}^{n} s_j^2$ does not depend on the choice of J', this is equivalent to minimizing the following function $h(x)$, with $x = \sum_{j\in J\setminus J'} s_j$:

$$h(x) := \frac{1}{2}x^2 + (S-q)(S-x) = \frac{1}{2}\left(\sum_{j\in J\setminus J'} s_j\right)^2 + (S-q)\left(S - \sum_{j\in J\setminus J'} s_j\right).$$

The unique minimum of $h(x)$ is $\frac{1}{2}S^2 - \frac{1}{2}q^2$ at $x = S-q$, i.e., when $\sum_{j\in J\setminus J'} s_j = S-q$. Hence, if such a set J' with $|J'| \le k$ exists, the resulting value is optimal for the scheduling problem. Therefore, if the solution to the created instance of $1||\sum_{\le k} e_j + \sum w_jC_j$ has value less than or equal to $\frac{1}{2}S^2 + \frac{1}{2}q^2 + \frac{1}{2}\sum_{j=1}^{n} s_j^2$, then the instance of k- Subset Sum is a "yes"-instance. Conversely, if the instance of Subset Sum is "yes", then there is a schedule of value $\frac{1}{2}S^2 + \frac{1}{2}q^2 + \frac{1}{2}\sum_{j=1}^{n} s_j^2$. □

3.2 Number of distinct processing times and weights

We consider the number of distinct processing times and weights as parameters. To this end, we say that two jobs j, j' are of the same *type* if $p_j = p_{j'}$ and $w_j = w_{j'}$; let τ be the number of types in an instance. Note, however, that jobs with the same type might have different rejection costs, so we cannot bound the "number of input numbers" like Fellows et al. [17]. Instead, we resort to convex integer programming, which to the best of our knowledge is used here for the first time in fixed-parameter algorithms. The running time of our algorithm will depend only *polynomially* on k, the upper bound on the number of jobs we are allowed to reject. For each type i, let $w^{(i)}$ be the weight and $p^{(i)}$ be the processing time of jobs of type i. Assume that job types are numbered $1, \dots, \tau$ such $w^{(i)}/p^{(i)} \ge w^{(i+1)}/p^{(i+1)}$ for each $i \in \{1, \dots, \tau-1\}$. Clearly, an optimal solution schedules jobs ordered non-increasingly by Smith's ratio without preemption.

The basis for our algorithm is a problem formulation as a convex integer minimization problem with dimension at most 2τ. In an instance, for each i, we let n_i be the number of jobs of type i and introduce an integer variable $x_i \in \mathbb{N}_0$ modeling how many jobs of type i we decide to schedule. We introduce the linear constraint $\sum_{i=1}^{\tau}(n_i - x_i) \leq k$, to ensure that at most k jobs are rejected.

The objective function is more involved. For each type i, scheduling the jobs of type i costs

$$\begin{aligned}\sum_{\ell=1}^{x_i} w^{(i)} \cdot (\ell \cdot p^{(i)} + \sum_{i'<i} x_{i'} \cdot p^{(i')}) &= w^{(i)} \cdot x_i \cdot \sum_{i'<i} x_{i'} \cdot p^{(i')} + w^{(i)} \cdot p^{(i)} \sum_{\ell=1}^{x_i} \ell \\ &= w^{(i)} \cdot x_i \cdot \sum_{i'<i} x_{i'} \cdot p^{(i')} + w^{(i)} \cdot p^{(i)} \cdot \frac{x_i \cdot (x_i+1)}{2} \\ &=: s_i(x).\end{aligned}$$

As we want to formulate the overall problem as a convex program, ideally we would like that each function $s_i(x)$ is convex. Unfortunately, this is not necessarily the case. However, the *sum* $\sum_i s_i(x)$ is convex, which is sufficient for our purposes.

Lemma 3 *The function $\sum_{i=1}^{\tau} s_i(x)$ is convex.*

Proof We define $y_i = p^{(i)} \cdot x_i$ for $i = 1, \ldots, \tau$. It then follows that

$$s_i(x) = \frac{w^{(i)}}{p^{(i)}} y_i \sum_{i'=1}^{i-1} y_{i'} + \frac{1}{2} \frac{w^{(i)}}{p^{(i)}} y_i^2 + \frac{1}{2} w^{(i)} y_i. \tag{10}$$

Then the sum of the $s_i(x)$ can be written as

$$\sum_{i=1}^{\tau} s_i(x) = \frac{1}{2} \sum_{i=1}^{\tau} \frac{w^{(i)}}{p^{(i)}} y_i^2 + \sum_{i=1}^{\tau} \frac{w^{(i)}}{p^{(i)}} y_i \sum_{i'=1}^{\tau-1} y_{i'} + \frac{1}{2} \sum_{i=1}^{\tau} w^{(i)} y_i \tag{11}$$

We claim that (the right-hand side of) (11) can be rewritten as

$$\begin{aligned}&\frac{1}{2}\left(\frac{w^{(1)}}{p^{(1)}} - \frac{w^{(2)}}{p^{(2)}}\right) y_1^2 + \frac{1}{2}\left(\frac{w^{(2)}}{p^{(2)}} - \frac{w^{(3)}}{p^{(3)}}\right)(y_1 + y_2)^2 \\ &+ \cdots + \frac{1}{2}\left(\frac{w^{(\tau-1)}}{p^{(\tau-1)}} - \frac{w^{(\tau)}}{p^{(\tau)}}\right)(y_1 + \cdots + y_{\tau-1})^2 \\ &+ \frac{1}{2}\frac{w^{(\tau)}}{p^{(\tau)}}(y_1 + \cdots + y_{\tau-1} + y_\tau)^2 + \frac{1}{2}\sum_{i=1}^{\tau} w^{(i)} y_i.\end{aligned}$$

This will suffice to prove the convexity of $\sum_i^\tau s_i(x)$: since we sorted the job types by decreasing Smith ratios $\frac{w^{(i)}}{p^{(i)}}$, the convexity of $\sum_i^\tau s_i(x)$ follows from the facts that we can express it as a sum of squares with non-negative coefficients, and that the sum of convex functions is convex.

To prove the claim, we can proceed by induction on τ. For $\tau = 1$, the first term on the right-hand side of (10) vanishes, and thus the claim holds. Suppose now that $\tau \geq 2$, and that we have proven the claim for all instances with strictly fewer than τ types. Then

$$\begin{aligned}
\sum_{i=1}^{\tau} s_i(x) &= \frac{1}{2}\sum_{i=1}^{\tau}\frac{w^{(i)}}{p^{(i)}}y_i^2 + \sum_{i=1}^{\tau}\frac{w^{(i)}}{p^{(i)}}y_i\sum_{i'<i}y_{i'} + \frac{1}{2}\sum_{i=1}^{\tau}w^{(i)}y_i \\
&= \left[\frac{1}{2}\sum_{i=1}^{\tau-1}\frac{w^{(i)}}{p^{(i)}}y_i^2 + \sum_{i=1}^{\tau-1}\frac{w^{(i)}}{p^{(i)}}y_i\sum_{i'<i}y_{i'} + \frac{1}{2}\sum_{i=1}^{\tau-1}w^{(i)}y_i\right] \\
&\quad + \left[\frac{1}{2}\frac{w^{(\tau)}}{p^{(\tau)}}y_\tau^2 + \frac{w^{(\tau)}}{p^{(\tau)}}y_\tau\sum_{i'=1}^{\tau-1}y_{i'} + \frac{1}{2}w^{(\tau)}y_\tau\right] \\
&= \left[\frac{1}{2}\left(\frac{w^{(1)}}{p^{(1)}} - \frac{w^{(2)}}{p^{(2)}}\right)y_1^2 + \frac{1}{2}\left(\frac{w^{(2)}}{p^{(2)}} - \frac{w^{(3)}}{p^{(3)}}\right)(y_1+y_2)^2 + \cdots\right. \\
&\quad + \frac{1}{2}\left(\frac{w^{(\tau-2)}}{p^{(\tau-2)}} - \frac{w^{(\tau-1)}}{p^{(\tau-1)}}\right)(y_1+\cdots+y_{\tau-2})^2 \\
&\quad \left. + \frac{1}{2}\frac{w^{(\tau-1)}}{p^{(\tau-1)}}(y_1+\cdots+y_{\tau-2}+y_{\tau-1})^2 + \frac{1}{2}\sum_{i=1}^{\tau-1}w^{(i)}y_i\right] \\
&\quad + \left[\frac{1}{2}\frac{w^{(\tau)}}{p^{(\tau)}}y_\tau^2 + \frac{w^{(\tau)}}{p^{(\tau)}}y_\tau\sum_{i'=1}^{\tau-1}y_{i'} + \frac{1}{2}w^{(\tau)}y_\tau\right] \\
&= \frac{1}{2}\left(\frac{w^{(1)}}{p^{(1)}} - \frac{w^{(2)}}{p^{(2)}}\right)y_1^2 + \frac{1}{2}\left(\frac{w^{(2)}}{p^{(2)}} - \frac{w^{(3)}}{p^{(3)}}\right)(y_1+y_2)^2 + \cdots \\
&\quad + \frac{1}{2}\left(\frac{w^{(\tau-1)}}{p^{(\tau-1)}} - \frac{w^{(\tau)}}{p^{(\tau)}}\right)(y_1+\cdots+y_{\tau-1})^2 \\
&\quad \frac{1}{2}\frac{w^{(\tau)}}{p^{(\tau)}}(y_1+\cdots+y_{\tau-1}+y_\tau)^2 + \frac{1}{2}\sum_{i=1}^{\tau}w^{(i)}y_i,
\end{aligned}$$

as desired. □

Observe that when scheduling x_i jobs of type i, it is optimal to reject the $n_i - x_i$ jobs with lowest rejection cost among all jobs of type i. Assume the jobs of each type i are labeled $j_1^{(i)}, \ldots, j_{n_i}^{(i)}$ by non-decreasing rejection cost. For each $s \in \mathbb{N}$ let $f_i(s) := \sum_{\ell=1}^{n_i-s} e_{j_\ell^{(i)}}$. In particular, to schedule x_i jobs of type i we can select them such that we need to pay $f_i(x_i)$ for rejecting the non-scheduled jobs (and this is an optimal decision). The difficulty is that the function $f_i(s)$ is in general not expressible by a polynomial whose degree is globally bounded (i.e., for each possible instance), which prevents a direct application of Theorem 1.

However, in Lemma 4 we show that $f_i(s)$ is the maximum of n_i linear polynomials, allowing us to formulate a convex program and solve it by Theorem 1.

Lemma 4 *For each type i there is a set of n_i polynomials $p_i^{(1)}, \dots, p_i^{(n_i)}$ of degree one such that $f_i(s) = \max_\ell p_i^{(\ell)}(s)$ for each $s \in \{0, \dots, n_i\}$.*

Proof For each $\ell \in \{0, \dots, n_i - 1\}$ we define $p_i^{(\ell)}(s)$ to be the unique polynomial of degree one such that $p_i^{(\ell)}(\ell) = f_i(\ell)$ and $p_i^{(\ell)}(\ell+1) = f_i(\ell+1)$. We observe that $f_i(s)$ is convex, since $f_i(\ell) - f_i(\ell+1) = e_{j^{(i)}_{n_i-\ell}} \geq e_{j^{(i)}_{n_i-\ell-1}} = f_i(\ell+1) - f_i(\ell+2)$ for each $\ell \in \{0, \dots, n_i - 2\}$. Therefore, the definition of the polynomials implies that $f_i(s) \geq p_i^{(\ell)}(s)$ for each $\ell \in \{0, \dots, n_i - 1\}$ and each $s \in \{0, \dots, n_i\}$. Since for each $s \in \{0, \dots, n_i - 1\}$ we have that $p_i^{(s)}(s) = f_i(s)$ and $p_i^{(n_i-1)}(n_i) = f_i(n_i)$, we conclude that $f_i(s) = \max_\ell p_i^{(\ell)}(s)$ for each $s \in \{0, \dots, n_i\}$. □

Lemmas 3 and 4 allow modeling the entire problem with the following convex program, where for each type i the variable g_i models the rejection cost for jobs of type i.

$$\begin{aligned}
\min \quad & \sum_{i=1}^{\tau} g_i + s_i(x) \\
\text{s.t.} \quad & \sum_{i=1}^{\tau} (n_i - x_i) \leq k \\
& g_i \geq p_i^{(\ell)}(x_i) \ \forall\, i \in \{1, \dots, \tau\}\ \forall\, \ell \in \{1, \dots, n_i\} \\
& \mathbf{g}, \mathbf{x} \in \mathbb{Z}^{\tau}_{\geq 0}
\end{aligned}$$

Observe that the above integer convex program admits an optimal solution with $g_i = \max_\ell p_i^{(\ell)}(x_i) = f_i(x_i)$ for each i. Thus, solving it yields an optimal solution to the overall instance.

Theorem 6 *Any instance of the problem $1||\sum_{\leq k} e_j + \sum w_j C_j$ on n jobs of τ types can be solved in time $(2\tau n + \log(\max_j \max\{e_j, p_j, w_j\}))^{O(1)} \cdot 2^{O(\tau^3)}$.*

Proof The number of integer variables is 2τ. All constraints and the objective function can be expressed by convex polynomials of degree at most 2. Hence, using Theorem 1 we obtain an overall running time of $(2\tau + n \cdot \tau) \log(\max_j \max\{e_j, p_j, w_j\}))^{O(1)} 2^{O(\tau^3)}$. □

4 Profit maximization for general scheduling problems

In this section, we consider the parameterized dual problem to scheduling jobs with rejection: the problem to reject at least $n - s$ jobs (s being the parameter) to minimize the total cost given by the rejection penalties plus the cost of the schedule. As we will show, this is equivalent to selecting at most s jobs and to schedule them in order to maximize the profit of the scheduled jobs. In contrast to the previous section, here we allow jobs to have release dates and that the profit of a job depends on the time when it finishes in the schedule, described by a function that might be different for each job (similarly as for job dependent cost functions [5,6]).

Formally, we are given a set J of n jobs, where each job j is characterized by a release date r_j, a processing time p_j, and a non-increasing profit function $f_j(t)$. We assume that the functions f are given as oracles and that we can evaluate them in time $O(1)$ per query. Let $\overline{p}$ denote the number of distinct processing times p_j in the instance. We want to schedule a set $\bar{J} \subseteq J$ of at most s jobs from J on a single machine. Our objective is to maximize $\sum_{j \in \bar{J}} f_j(C_j)$, where C_j denotes the completion time of j in the computed schedule. We call this problem the *s-bounded General Profit Scheduling Problem*, or *s-GPSP* for short. Observe that this generic problem definition allows to model profit functions that stem from scheduling objectives such as weighted flow time and weighted tardiness.

We like to note that in a first version of this paper we presented an algorithm using three, rather than two parameters. An anonymous referee gave us advice how to remove one parameter and we present the improved version here. First, we show how to find the best solution with a non-preemptive schedule, and subsequently we show how to extend the algorithm to the preemptive setting.

4.1 Non-preemptive schedules

Denote by $q_1, \ldots, q_{\overline{p}}$ the set of arising processing times in the instance. We start with a color-coding step, where we color each job uniformly at random with one color from $\{1, \ldots, s\}$. When coloring the jobs randomly, with probability $s!/s^s$ all jobs scheduled in an optimal solution (the set $\bar{J}$) will receive distinct colors. We call such a coloring of $\bar{J}$ a *good coloring*. When trying enough random colorings, with high probability we will find at least one good coloring. We will discuss later how to derandomize this algorithm. For now, assume that we know a good coloring $c : J \to \{1, \ldots, s\}$.

We describe now a dynamic program that finds the best non-preemptive schedule of at most s jobs with different colors. The dynamic program has one cell $C(S, t)$ for each combination of a set $S \subseteq \{1, \ldots, s\}$ and a time $t \in T$, where $T := \{r_j + \sum_{i=1}^{\overline{p}} s_i \cdot q_i \mid j \in J, s_1, \ldots, s_{\overline{p}} \in \{0, \ldots, s\}\}$. Observe that the set T contains all possible completion times of a job in an optimal non-preemptive schedule (we will show later that this is even true for preemptive schedules). Also, $|T| \leq n \cdot (s+1)^{\overline{p}}$. The cell $C(S, t)$ models the subproblem of scheduling a set of at most $|S|$ jobs such that their colors are pairwise different and contained in S, and so that the last job finishes by time t the latest. We will denote by $\mathsf{opt}(S, t)$ the value of the optimal solution for this subproblem with respect to the fixed coloring c.

Consider an optimal solution with value $\mathsf{opt}(S, t)$. If no job finishes at time t, then $\mathsf{opt}(S, t) = \mathsf{opt}(S, t')$ for some $t' < t$. Else, there is some job j with $c(j) \in S$ that finishes at time t; then $\mathsf{opt}(S, t) = f_j(t) + \mathsf{opt}(S \setminus \{c(j)\}, t')$, where t' is the largest value in T with $t' \leq t - p_j$. We now prove this formally.

Lemma 5 *Let $c : J \to \{1, \dots, s\}$ be a coloring, let $S \subseteq \{1, \dots, s\}$ and let $t \in T$. The optimal value* $\mathsf{opt}(S, t)$ *for the dynamic programming cell $C(S, t)$ equals*

$$\max \left\{ \max_{t' \in T:\, t' < t} \mathsf{opt}(S, t'), \max_{j \in J:\, (r_j \leq t - p_j) \wedge c(j) \in S} f_j(t) \right. \\ \left. + \mathsf{opt}(S \setminus \{c(j)\}, \max\{t' \in T \mid t' \leq t - p_j\}), 0 \right\}. \tag{12}$$

Proof For cells $C(S, t_{\min})$ with $t_{\min} = \min_{t \in T} t$ the claim is true since $t_{\min} = \min_j r_j$ and therefore $\mathsf{opt}(S, t_{\min}) = 0$. Now consider a cell $C(S, \bar{t})$ with $\bar{t} > t_{\min}$. Let $\mathsf{opt}'(S, \bar{t})$ denote the right-hand side of (12). We show that $\mathsf{opt}(S, \bar{t}) \geq \mathsf{opt}'(S, \bar{t})$ and $\mathsf{opt}(S, \bar{t}) \leq \mathsf{opt}'(S, \bar{t})$.

For the first claim, we show that there is a schedule with value $\mathsf{opt}'(S, \bar{t})$. If $\mathsf{opt}'(S, \bar{t}) = 0$ or $\mathsf{opt}'(S, \bar{t}) = \max_{t' \in T : t' < \bar{t}} \mathsf{opt}(S, t')$ then the claim is immediate. Now suppose that there is a job j with $r_j \leq \bar{t} - p_j$ and $c(j) \in S$ such that $\mathsf{opt}(S, \bar{t}) = f_j(\bar{t}) + \mathsf{opt}(S \setminus \{c(j)\}, \max\{t' \in T | t' \leq \bar{t} - p_j\})$. Then there is a feasible schedule for the problem encoded in cell $C(S, \bar{t})$ that is given by the jobs in the optimal solution for the cell $C(S \setminus \{c(j)\}, \max\{t' \in T \mid t' \leq \bar{t} - p_j\})$, and additionally schedules job j during $[\bar{t} - p_j, \bar{t})$. The profit of this solution is $f_j(\bar{t}) + \mathsf{opt}(S \setminus \{c(j)\}, \max\{t' \in T \mid t' \leq \bar{t} - p_j\})$.

For showing that $\mathsf{opt}(S, \bar{t}) \leq \mathsf{opt}'(S, \bar{t})$ consider the schedule for $\mathsf{opt}(S, \bar{t})$ and assume w.l.o.g. that the set T contains all start and end times of jobs in that schedule. If no job finishes at time $\bar{t}$ then $\mathsf{opt}(S, \bar{t}) = \max_{t' \in T:\, t' < t} \mathsf{opt}(S, t')$, using that T contains all finishing times of a job in the optimal schedule. Note that $\{t' \in T | t' < t\} \neq \emptyset$ since we assumed that $\bar{t} > t_{\min}$. If a job j finishes at time $\bar{t}$ then $\mathsf{opt}(S, \bar{t}) = f_j(\bar{t}) + \mathsf{opt}(S \setminus \{c(j)\}, \max\{t' \in T \mid t' \leq \bar{t} - p_j\})$. This implies that $\mathsf{opt}(S, \bar{t}) \leq \mathsf{opt}'(S, \bar{t})$. □

To compute the value $\mathsf{opt}(S, t)$, our dynamic program evaluates expression (12). Note that this transition implies $\mathsf{opt}(S, t_{\min}) = 0$ for $t_{\min}$ the minimum value in T. The number of dynamic programming cells is at most $2^s \cdot |T| = 2^s n \cdot (s+1)^{\overline{p}}$. Evaluating expression (12) takes time $O(|T| + n) = O((s+1)^{\overline{p}} + n)$, given that optimal solutions to all subproblems are known. Together with Lemma 5 this yields the following lemma.

Lemma 6 *There is an algorithm that, given an instance of s-GPSP with a set J of n jobs of at most $\overline{p}$ distinct processing times together with a good coloring $c : J \to \{1, \dots, s\}$, in time $O(2^s \cdot n^2 \cdot (s+1)^{2\overline{p}})$ computes an optimal non-preemptive schedule for the jobs in J.*

Instead of coloring the jobs randomly, we can use a family of hash functions, as described by Alon et al. [3]. Using our notation, they show that there is a family $\mathcal{F}$ of $2^{O(s)} \log n$ hash functions $J \to \{1, \dots, s\}$ such that for any set $J' \subseteq J$ (in particular for the set of jobs J' that is scheduled in an optimal solution) there is a hash function $F \in \mathcal{F}$ such that F is bijective on J'. The value of each of these functions on each specific element of J can be evaluated in $O(1)$ time. This yields the following.

Theorem 7 *There is a deterministic algorithm that, given an instance of s- GPSP with n jobs and $\overline{p}$ processing times, computes an optimal non-preemptive schedule in time $2^{O(s)} \cdot (s+1)^{2\overline{p}} \cdot n^2 \log n$.*

4.2 Preemptive schedules

We now extend our algorithm from the previous section to the setting when preemption of jobs is allowed. Now the dynamic program becomes more complicated. As before, we color all jobs in J uniformly at random with colors from $\{1, \ldots, s\}$. Fix a good coloring $c : J \rightarrow \{1, \ldots, s\}$. In the dynamic programming table we have one cell $C(S, t_1, t_2, P)$ for each combination of a set $S \subseteq \{1, \ldots, s\}$, values $t_1, t_2 \in T$, where as above $T = \left\{r_j + \sum_{i=1}^{\overline{p}} s_i \cdot q_i \mid j \in J, s_1, \ldots, s_{\overline{p}} \in \{0, \ldots, s\}\right\}$, and a value $P \in \mathcal{P} := \left\{\sum_{i=1}^{\overline{p}} s_i \cdot q_i \mid s_1, \ldots, s_{\overline{p}} \in \{0, \ldots, s\}\right\}$ (note that like for T we have $|\mathcal{P}| \leq (s+1)^{\overline{p}}$). This cell encodes the problem of selecting at most $|S|$ jobs with total processing time at most P such that (i) their colors are pairwise different and contained in S, (ii) they are released during $[t_1, t_2)$; we want to schedule them during $[t_1, t_2)$, possibly with preemption, the objective being to maximize the total profit.

To this end, we first prove that the set T contains all possible start- and completion times of jobs of an optimal solution. The former denotes the first time in a schedule where each job is being processed.

Lemma 7 *There is an optimal solution such that for each scheduled job j, its start time and completion time are both contained in T.*

Proof Consider an optimal schedule $\mathcal{S}$ for the set of at most k jobs that we want to schedule. If we fix the completion times of the jobs in the schedule as deadlines and run the policy of earliest-deadline-first (EDF) with these deadlines, then we obtain a schedule $\mathcal{S}'$ where each job finishes no later than in $\mathcal{S}$. This is true since given one machine and jobs with release dates and deadlines, there is a feasible preemptive schedule (i.e., a schedule where each job finishes by its deadline) if and only if EDF finds such a schedule. In particular, since the profit-functions of the jobs are non-increasing, the profit given by $\mathcal{S}'$ is at most the profit given by $\mathcal{S}$.

We prove the claim by induction over the deadlines of the jobs in $\mathcal{S}'$. In fact, we prove the following slightly stronger claim: for the job with s'-th deadline, its start time and completion time are in the set $T_{s'} = \left\{r_j + \sum_{i=1}^{\overline{p}} s_i \cdot q_i \mid j \in J, s_1, \ldots, s_{\overline{p}} \in \{0, \ldots, s'\}\right\}$.

For the job j_1 with smallest deadline, we have that j_1 starts at time $r_{j_1} \in T_1$ and finishes at time $r_{j_1} + p_{j_1} \in T_1$, since j_1 has highest priority. Suppose by induction that the claim is true for the jobs with the $s'-1$ smallest deadlines, and note that the deadlines are distinct. Consider the job $j_{s'}$ with the s'-smallest deadline. Let $R_{s'}$ denote its start time. Let J_1 denote the jobs finishing before $R_{s'}$. Similarly, let J_2 denote the jobs finishing during $(R_{s'} C_{s'}]$. Note that $j_{s'} \in J_2$ but J_2 does not contain a job with deadline later than $j_{s'}$. Since we run EDF, $R_{s'} = r_{j_{s'}}$ or $R_{s'} = C_{j_{s''}}$ for some job $j_{s''}$. In both cases $R_{s'} \in T_{|J_1|}$, and thus $R_{s'} = r_j + \sum_{i=1}^{\overline{p}} s_i^{(1)} \cdot q_i$ for some job $j \in J$

and values $s_1^{(1)}, \ldots, s_{\overline{p}}^{(1)} \in \{0, \ldots, |J_1|\}$. During $(R_{s'} C_{s'}]$, at most $|J_2|$ jobs finish, and their total length is $\sum_{i=1}^{\overline{p}} s_i^{(2)} \cdot q_i$ for values $s_1^{(2)}, \ldots, s_{\overline{p}}^{(2)} \in \{0, \ldots, |J_2|\}$. Hence,

$$C_{s'} = r_j + \sum_{i=1}^{\overline{p}} s_i^{(1)} \cdot q_i + \sum_{i=1}^{\overline{p}} s_i^{(2)} \cdot q_i = r_j + \sum_{i=1}^{\overline{p}} s_i \cdot q_i$$

for some job $j \in J$ and values $s_1, \ldots, s_{\overline{p}} \in \{0, \ldots, |J_1| + |J_2|\}$. By $|J_1| + |J_2| = s'$, we have that $C_{j_{s'}} \in T_{s'}$. $\square$

Suppose we want to compute an optimal solution of value $\mathsf{opt}(S, t_1, t_2, P)$ for a cell $C(S, t_1, t_2, P)$. In any optimal solution, we can assume that the last finishing job has smallest priority, i.e., we work on this job only if no other job is pending. Let j be this (last) job and let R_j denote its start time. Then, during $[R_j, t_2)$ the only scheduled jobs are j and some jobs that are released during $[R_j, t_2)$. Also, all jobs scheduled during $[t_1, R_j)$ are also released during $[t_1, R_j)$ and they are finished within $[t_1, R_j)$.

This observation gives rise to the dynamic programming transition. Given a cell $C(S, t_1, t_2, P)$, we enumerate all possibilities for the last job j, its start time R_j, the total processing time of jobs scheduled during $[t_1, R_j)$ and $[R_j, t_2)$, and their colors. One combination corresponds to an optimal solution, and we reduce the overall problem to the respective two subproblems for $[t_1, R_j)$, $[R_j, t_2)$. (Observe that the color coding step prevents to select the same job in both subproblems.) We justify this transition in the following lemma. For ease of notation, for a cell $C(S, t_1, t_2, P)$ denote by $\mathsf{en}(S, t_1, t_2, P)$ the set of tuples (j, S', R_j, P', P'') that we need to enumerate for splitting the cell, i.e., all such tuples with $c(j) \in S$, $t_1 \leq r_j \leq t_2 - p_j$, $S' \subseteq S \backslash \{c(j)\}$, $t_2 - R_j \geq P'' + p_j$, $P', P'' \in \mathcal{P}$, and $P' + P'' + p_j \leq P$.

Lemma 8 *For a coloring* $c : J \to \{1, \ldots, s\}$, *each dynamic programming cell* $C(S, t_1, t_2, P)$ *satisfies*

$$\begin{aligned} \mathsf{opt}(S, t_1, t_2, P) = \max \Big\{ & \max_{t' \in T:\, t_1 \leq t' < t_2} \mathsf{opt}(S, t_1, t', P), \\ & \max_{(j, S', R_j, P', P'') \in \mathsf{en}(S, t_1, t_2, P)} f_j(t_2) + \mathsf{opt}(S', t_1, R_j, P') \\ & + \mathsf{opt}(S \backslash (S' \cup \{c(j)\}, R_j, t_2, P''), 0 \Big\}. \end{aligned} \tag{13}$$

Proof Similarly as in the proof of Lemma 5, let $\mathsf{opt}'(S, t_1, t_2, P)$ denote the right-hand side of (13) for the cell $C(S, t_1, t_2, p)$.

Consider a cell $C(S, t_1, t_2, P)$. First, we show that $\mathsf{opt}(S, t_1, t_2, P) \geq \mathsf{opt}'(S, t_1, t_2, P)$. To this end, it suffices to show that there is a feasible solution with value at least $\mathsf{opt}'(S, t_1, t_2, P)$. If $\mathsf{opt}'(S, t_1, t_2, P) = 0$ then the claim is immediate (just take the empty solution). Otherwise, first assume that $\mathsf{opt}'(S, t_1, t_2, P) = \mathsf{opt}(S, t_1, t', P)$ for some $t' \in T$ with $t' < t_2$. Any solution for the cell $C(S, t_1, t', P)$ is also feasible for $C(S, t_1, t_2, P)$, and thus $\mathsf{opt}(S, t_1, t_2, P) \geq \mathsf{opt}(S, t_1, t', P)$. Hence, $\mathsf{opt}(S, t_1, t_2, P) \geq \mathsf{opt}'(S, t_1, t_2, P)$.

Finally, assume that there is a tuple (j, S', R_j, P', P'') such that $\mathsf{opt}'(S, t_1, t_2, P) = f_j(t_2) + \mathsf{opt}(S', t_1, R_j, P') + \mathsf{opt}(S \backslash (S' \cup \{c(j)\}, R_j, t_2, P'')$. We build a solution with this value as follows. During $[t_1, R_j)$, we schedule an optimal solution for the cell $C(S', t_1, R_j, P')$. During $[R_j, t_2)$, we schedule an optimal solution for the cell $C(S \backslash (S' \cup \{c(j)\}, R_j, t_2, P'')$. By assumption, $t_2 - R_j \geq P'' + p_j$. Hence, the schedule for the latter cell leaves total idle time during $[R_j, t_2)$ of at least p_j. During these idle times we schedule job j and, thus, j finishes at time t_2 the latest. As $S' \subseteq S \backslash \{c(j)\}$ and $S' \cap (S \backslash (S' \cup \{c(j)\}) = \emptyset$, no job is scheduled in both subproblems. Also, $P' + P'' + p_j \leq P$. Hence, the solution is feasible for the cell $C(S, t_1, t_2, P)$ and its value is at least $f_j(t_2) + \mathsf{opt}(S', t_1, R_j, P') + \mathsf{opt}(S \backslash (S' \cup \{c(j)\}, R_j, t_2, P'')$. Thus, $\mathsf{opt}(S, t_1, t_2, P) \geq \mathsf{opt}'(S, t_1, t_2, P)$.

Conversely, we want to show that $\mathsf{opt}(S, t_1, t_2, P) \leq \mathsf{opt}'(S, t_1, t_2, P)$. Consider an optimal solution for the cell $C(S, t_1, t_2, P)$. If $\mathsf{opt}(S, t_1, t_2, P) = 0$, then the claim is immediate. Otherwise, we can assume that in the schedule for the solution, the jobs are ordered according to EDF by their respective completion time (see also the proof of Lemma 7). Let j be the job with largest deadline (and hence smallest priority); then $c(j) \in S$. If j ends at a time $t' < t_2$ then $\mathsf{opt}(S, t_1, t_2, P) = \mathsf{opt}(S, t_1, t', P) \leq \mathsf{opt}'(S, t_1, t_2, P)$.

Now suppose that j finishes at time t_2, and let R_j denote its start time in an optimal schedule of value $\mathsf{opt}(S, t_1, t_2, P)$. Observe that $t_1 \leq r_j \leq t_2 - p_j$. Let $S' \subseteq S \backslash \{c(j)\}$ denote the set of colors of jobs scheduled during $[t_1, R_j)$ in an optimal solution with value $\mathsf{opt}(S, t_1, t_2, P)$ (observe that $c(j) \notin S'$), and let P' denote the total processing time of jobs scheduled during $[t_1, R_j)$. Since all jobs scheduled during $[t_1, R_j)$ start and finish during that interval, we have that $P' \in \mathcal{P}$. Also, let P'' denote the total processing time of jobs other than j that are scheduled during $[R_j, t_2)$ in a solution of value $\mathsf{opt}(S, t_1, t_2, P)$. In particular, $t_2 - R_j \geq P'' + p_j$. As all those jobs start and finish during $[R_j, t_2)$, this implies that $P'' \in \mathcal{P}$. Hence, $(j, S', R_j, P', P'') \in \mathsf{en}(S, t_1, t_2, P)$, and thus

$$\begin{aligned}\mathsf{opt}(S, t_1, t_2, P) &= f_j(t_2) + \mathsf{opt}(S', t_1, R_j, P') \\ &\quad + \mathsf{opt}(S \backslash (S' \cup \{c(j)\}, R_j, t_2, P'') \leq \mathsf{opt}'(S, t_1, t_2, P).\end{aligned}$$

□

Our dynamic program evaluates Eq. (13) for each cell. Observe that cells $C(S, t_1, t_2, p)$ with $S = \emptyset$ or $t_1 = t_2$ obey $\mathsf{en}(S, t_1, t_2, P) = \emptyset$, and so $\mathsf{opt}(S, t_1, t_2, p) = 0$. This yields the following.

Lemma 9 *There is an algorithm that, given a set of n jobs with $\overline{p}$ distinct processing times and a good coloring $c : J \to \{1, \ldots, s\}$, in time $2^{O(s)} s^{O(\overline{p})} n^4$ computes an optimal preemptive schedule for s-GSP.*

Using the same derandomization technique as above, we obtain the following theorem.

Theorem 8 *There is a deterministic algorithm that, given a set of n jobs with $\overline{p}$ distinct processing times, in time $2^{O(s)} s^{O(\overline{p})} n^4 \log n$ computes an optimal preemptive schedule for s-GSP.*

Observe that $(\log n)^{O(\overline{p})} = O((\overline{p} \log \overline{p})^{O(\overline{p})} + n)$; this yields the following.

Corollary 1 *There is a deterministic algorithm that, given a set of n jobs with $\overline{p}$ distinct processing times, computes an optimal solution for $O(\log n)$-GSP in time $(\overline{p}\log\overline{p})^{O(\overline{p})}\cdot n^{O(1)}$.*

Applying Theorem 8, we obtain the following corollary when at least $n-s$ jobs have to be rejected.

Corollary 2 *There is a deterministic algorithm that, given a set of n jobs with at most $\overline{p}$ distinct processing times, each job having a private cost function w_j, solves the problem $1|r_j,(pmtn)|\sum_S w_j(C_j)+\sum_{\bar{S}} e_j$ in time $2^{O(s)}s^{O(\overline{p})}n^4\log n$, when at least $n-s$ jobs have to be rejected.*

Proof Any instance of the problem to reject $n-s$ jobs and schedule the remaining s jobs to minimize $1|r_j,(pmtn)|\sum_S w_j(C_j)+\sum_{\bar{S}} e_j$ is equivalent to an instance of s-GPSP (with or without preemption, respectively) where for each job j we define a profit function $f_j(t):=e_j-w_j(t)$ and keep the release dates and processing times unchanged. Then the corollary follows from Theorems 7 and 8. □

Observe that the objective function $\sum_S f_j(C_j)+\sum_{\bar{S}} e_j$ is in particular a generalization of

– weighted flow time with rejection, $\sum_S w_j(C_j-r_j)+\sum_{\bar{S}} e_j$, where each job j has a weight w_j associated with it
– weighted tardiness with rejection, $\sum_S w_j\max\{0,t-d_j\}+\sum_{\bar{S}} e_j$, where each job j has additionally a deadline d_j, and
– weighted sum of completion times with rejection, $\sum_S w_jC_j+\sum_{\bar{S}} e_j$.

So our algorithm works for these objective functions as well.

When only the number s of scheduled jobs is chosen as parameter the problem becomes W[1]-hard, as pointed out to us by an anonymous reviewer.

Theorem 9 *Scheduling a set $\bar{J}$ of s jobs from n given jobs (non-)preemptively on a single machine to maximize $\sum_{j\in\bar{J}} b_j-w_j(C_j)$ is W[1]-hard for parameter s, even if all n jobs have the same function w_j and $b_j=p_j$ for each job j. Thus, (non-) preemptive s-GPSP is W[1]-hard for parameter s.*

Proof Consider an instance of k-Subset Sum, specified by integers s_i and a target value q. In our reduction, for each s_j, we create a job j with processing time $p_j:=s_j$, profit $b_j:=s_j$ and cost function w_j defined as

$$w_j(t)=\begin{cases}0, & \text{if } t\le q,\\ +\infty, & \text{if } t>q,\end{cases}$$

for all times t. Then there is a set of s integers s_i whose sum is exactly q, if and only if we can schedule the corresponding set $\bar{J}$ of s jobs preemptively or non-preemptively on a single machine such that $\sum_{j\in\bar{J}} b_j-w_j(C_j)=q$. Notice that there is only a single function $w_j(t)$ that is used for all jobs $j\in J$ in this reduction. When defining as profit function $f_j(t):=b_j-w_j(t)$ for each job j this yields an instance of k-GPSP which is thus W[1]-hard. □

On the other hand, we prove that choosing only the number of distinct processing times $\bar{p}$ as a parameter is not enough, as we show the problem to be NP-hard even if $p_j \in \{1, 3\}$ for all jobs j. The same holds for the GSP [5] where—without the profit maximization aspect—we are given a set of jobs j, each of them having a release date r_j and an individual cost function $w_j(t)$, and we want to schedule all of them on one machine in order to minimize $\sum_j w_j(C_j)$ where C_j denotes the completion time of job j in the computed (preemptive) schedule. We prove this theorem in the next section.

Theorem 10 *The general profit scheduling problem (GPSP) and the GSP are* NP-*hard, even if the processing time of each job is either exactly 1 or 3. This holds in both the preemptive and non-preemptive setting.*

5 Hardness of GPSP with processing times one and three

In this section we show that GPSP is NP-hard, even if $p_j \in \{1, 3\}$ for each job j. We reduce from 3- Dimensional Matching, where one is given three sets $A = \{a_1, \dots, a_n\}$, $B = \{b_1, \dots, b_n\}$, and $C = \{c_1, \dots, c_n\}$ for some $n \in \mathbb{N}$, and a set $T \subseteq A \times B \times C$. The goal is to find a subset $T' \subseteq T$ with $|T'| = n$ such that for any two different triples $(a_i, b_j, c_k), (a_i', b_j', c_k') \in T'$ it holds that $a_i \neq a_i'$, $b_j \neq b_j'$, and $c_k \neq c_k'$. Note that this implies that each element in A, B, and C appears in exactly one triple in T'. We say that the given instance is a "yes"-instance if there exists such a set T'. The 3- Dimensional Matching problem is NP-hard [21].

Given an instance of 3- Dimensional Matching, we construct an instance of GPSP. With each triple $(a_i, b_j, c_k) \in T$ we associate an interval of length 3 on the time axis. For each such triple, we define a value $t(a_i, b_j, c_k)$ such that for any two different triples, the corresponding intervals $[t(a_i, b_j, c_k), t(a_i, b_j, c_k) + 3)$ are disjoint and $\bigcup_{(a_i,b_j,c_k)\in T} [t(a_i, b_j, c_k), t(a_i, b_j, c_k)+3) = [0, 3|T|)$. We achieve this by assigning to each triple $(a_i, b_j, c_k) \in T$ a unique identifier $z_{(a_i,b_j,c_k)} \in \{0, |T| - 1\}$ and define $t(a_i, b_j, c_k) := 3 \cdot z_{(a_i,b_j,c_k)}$. For each triple $(a_i, b_j, c_k) \in T$, we introduce four jobs, all released at time $t(a_i, b_j, c_k)$: three jobs of unit length called $A(a_i, b_j, c_k)$, $B(a_i, b_j, c_k)$, $C(a_i, b_j, c_k)$ that represent the occurrence of the respective element from A, B, C in the triple (a_i, b_j, c_k) (so the occurrence of the elements a_i, b_j, and c_k in (a_i, b_j, c_k), respectively); and additionally one job of length three that we denote by $L(a_i, b_j, c_k)$.

For each element in $A \cup B \cup C$ we specify a unique point in time. We define $t(a_i) := 3|T| + i$, $t(b_j) := 3|T| + n + j$, and $t(c_k) := 3|T| + 2n + k$ for each respective element $a_i \in A$, $b_j \in B$, $c_k \in C$; see Fig. 1 for a sketch of the subdivision of the time axis and the profit functions (that we define later).

Before specifying the profit function for each job, we give some intuition. The idea is that if the given instance is a "yes"-instance, then there is an optimal schedule where

– for each triple (a_i, b_j, c_k) in the optimal solution we schedule the job $L(a_i, b_j, c_k)$ during the interval $[t(a_i, b_j, c_k), t(a_i, b_j, c_k) + 3)$ and the jobs $A(a_i, b_j, c_k)$, $B(a_i, b_j, c_k)$, and $C(a_i, b_j, c_k)$ during $[t(a_i) - 1, t(a_i))$, $[t(b_j) - 1, t(b_j))$, and

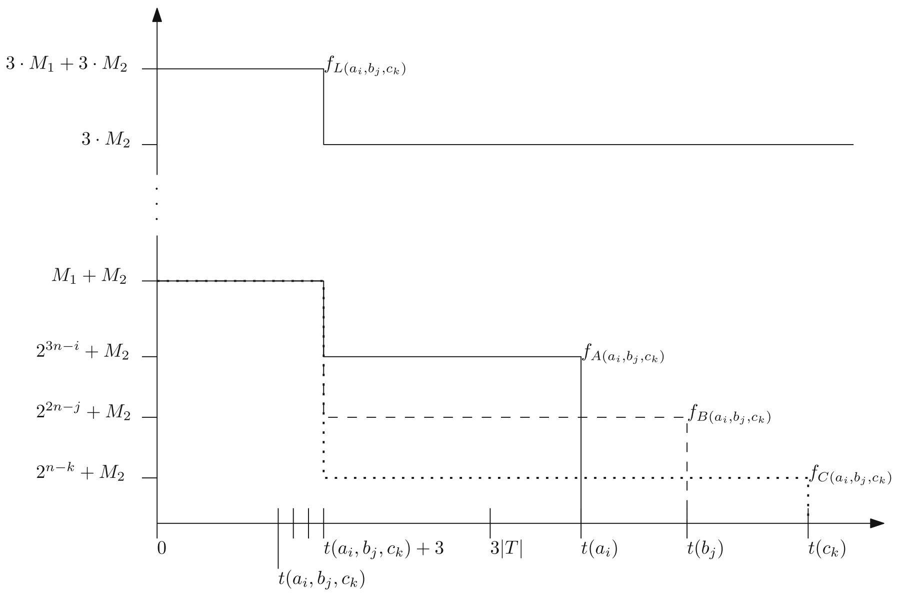

Fig. 1 Sketch for the reduction in Theorem 11

$[t(c_k)-1, t(c_k))$, respectively. Note that this implies that during $[t(a_i), t(c_k)+1)$ the machine is busy.

- for each triple (a_i, b_j, c_k) that is *not* in the optimal solution we schedule jobs $A(a_i, b_j, c_k)$, $B(a_i, b_j, c_k)$, $C(a_i, b_j, c_k)$ during $[t(a_i, b_j, c_k), t(a_i, b_j, c_k)+3)$ (in an arbitrary order) and the job $L(a_i, b_j, c_k)$ somewhere after time $3|T|+3n$.

We define the profit functions so that there is always an optimal solution which is in standard from. We say that a solution is in *standard form* if

1. each job $A(a_i, b_j, c_k)$ (job $B(a_i, b_j, c_k)$, job $C(a_i, b_j, c_k)$) is scheduled either during the interval $[t(a_i, b_j, c_k), t(a_i, b_j, c_k)+3)$ or during $[3|T|, t(a_i))$ (during $[3|T|, t(b_j))$, during $[3|T|, t(c_k))$); and
2. each job $L(a_i, b_j, c_k)$ is scheduled non-preemptively either completely during the interval $[t(a_i, b_j, c_k), t(a_i, b_j, c_k)+3)$ or completely during $[3|T|+3n, \infty)$.

Observe in particular that solutions in standard form schedule *all* jobs.

Formally, we define the profit functions as follows. Let $M_1 := 1 + 2^{3n}$ and $M_2 := M_1 \cdot 3|T| + 2^{3n}$. For the jobs $A(a_i, b_j, c_k)$, $B(a_i, b_j, c_k)$, and $C(a_i, b_j, c_k)$ for each triple $(a_i, b_j, c_k) \in T$ we define their profit function as follows:

$$f_{A(a_i,b_j,c_k)}(t) := \begin{cases} M_1 + M_2, & \text{for } t \le t(a_i, b_j, c_k) + 3, \\ 2^{3n-i} + M_2, & \text{for } t(a_i, b_j, c_k) + 3 < t \le t(a_i), \\ 0, & \text{for } t > t(a_i); \end{cases}$$

$$f_{B(a_i,b_j,c_k)}(t) := \begin{cases} M_1 + M_2, & \text{for } t \le t(a_i, b_j, c_k) + 3, \\ 2^{2n-j} + M_2, & \text{for } t(a_i, b_j, c_k) + 3 < t \le t(b_j), \\ 0, & \text{for } t > t(b_j); \end{cases}$$

$$f_{C(a_i,b_j,c_k)}(t) := \begin{cases} M_1 + M_2, & \text{for } t \leq t(a_i,b_j,c_k) + 3, \\ 2^{n-k} + M_2, & \text{for } t(a_i,b_j,c_k) + 3 < t \leq t(c_k), \\ 0, & \text{for } t > t(c_k). \end{cases}$$

For the long job $L(a_i, b_j, c_k)$ for each triple $(a_i, b_j, c_k) \in T$ we define its profit function as

$$f_{L(a_i,b_j,c_k)}(t) := \begin{cases} 3 \cdot M_1 + 3 \cdot M_2, & \text{for } t \leq t(a_i,b_j,c_k) + 3, \\ 3 \cdot M_2, & \text{for } t > t(a_i,b_j,c_k) + 3. \end{cases}$$

Lemma 10 *There is always an optimal solution of the above instance of GPSP that is in standard form.*

Proof Given an optimal solution to the defined GPSP instance, suppose that it does not satisfy the first property in the definition of the standard form. First, suppose for contradiction that there is a job $A(a_i, b_j, c_k)$ that finishes after time $t(a_i)$, thus contributing zero towards the objective. Then the objective value can be at most $3|T| \cdot (M_1 + M_2) + |T| \cdot (3 \cdot M_1 + 3 \cdot M_2) - M_1 - M_2$ (assuming that all other jobs give the maximal possible profit). However, then there is a strictly better solution S' which is given by

– scheduling each job $A(a_i, b_j, c_k)$, $B(a_i, b_j, c_k)$, and $C(a_i, b_j, c_k)$ for each triple $(a_i, b_j, c_k) \in T$ during $[t(a_i, b_j, c_k), t(a_i, b_j, c_k) + 3)$, respectively, and
– scheduling each job $L(a_i, b_j, c_k)$ for each triple $(a_i, b_j, c_k) \in T$ somewhere during $[3|T| + 3n, \infty)$.

The solution S' yields a profit of at least $3|T| \cdot (M_1 + M_2) + |T| \cdot (3 \cdot M_2) > 3|T| \cdot (M_1 + M_2) + |T| \cdot (3 \cdot M_1 + 3 \cdot M_2) - M_2$, using that $M_2 > 3|T| \cdot M_1$. Hence, the first considered schedule was not optimal. For jobs $B(a_i, b_j, c_k)$ and $C(a_i, b_j, c_k)$, the claim can be shown similarly.

Secondly, for contradiction suppose that there is a job $A(a_i, b_j, c_k)$ that is scheduled during interval $[t(a_i, b_j, c_k)+3, 3 \cdot |T|)$. Then the computed schedule yields a profit of at most $3|T| \cdot (M_1 + M_2) - M_1 + \sum_{\ell=0}^{3n-1} 2^\ell + |T| \cdot 3M_2 < 3|T| \cdot (M_1 + M_2) + |T| \cdot 3M_2$, using that $M_1 > \sum_{\ell=0}^{3n-1} 2^\ell = 2^{3n} - 1$. Hence, solution S' (as defined above) is better than the considered optimal solution, which yields a contradiction. Hence, any optimal solution satisfies the first property.

Suppose now that we are given an optimal solution that violates the second property. Then there is a job $L(a_i, b_j, c_k)$ that is not completely scheduled during $[t(a_i, b_j, c_k), t(a_i, b_j, c_k) + 3)$. Then its contribution for the global profit is $3 \cdot M_2$ and this does not depend on where exactly after $t(a_i, b_j, c_k) + 3$ it finishes. Thus, we can move $L(a_i, b_j, c_k)$ so that it is completely scheduled non-preemptively during $[3|T| + 3n, \infty)$, without changing its contribution to the profit. □

To show the correctness of our reduction, we show that there is a solution with an overall profit of $|T| \cdot (3 \cdot M_1 + 3 \cdot M_2) + |T| \cdot 3 \cdot M_2 + 2^{3n} - 1$ if and only if the instance of 3- DIMENSIONAL MATCHING is a "yes"-instance. In the proof of the next lemma we simply calculate the profit we can obtain, assuming that we reduced from a "yes"-instance. The schedule yielding this profit is the one defined above where we gave some initial intuition.

Lemma 11 *If the given instance of* 3- DIMENSIONAL MATCHING *is a "yes"-instance, then there is a solution with profit* $|T| \cdot (3 \cdot M_1 + 3 \cdot M_2) + |T| \cdot 3 \cdot M_2 + 2^{3n} - 1$.

Proof Let $T' \subseteq T$ be a solution of the instance of 3- DIMENSIONAL MATCHING (note that $|T'| = n$). For each tuple $(a_i, b_j, c_k) \in T'$ we schedule the job $L(a_i, b_j, c_k)$ during $[t(a_i, b_j, c_k), t(a_i, b_j, c_k) + 3)$, and jobs $A(a_i, b_j, c_k)$, $B(a_i, b_j, c_k)$, $C(a_i, b_j, c_k)$ during $[t(a_i) - 1, t(a_i))$, $[t(b_j) - 1, t(b_j))$, and $[t(c_k) - 1, t(c_k))$, respectively. Note that since T' is a feasible solution, at most one job is scheduled in each interval $[t(a_i) - 1, t(a_i))$, $[t(b_j) - 1, t(b_j))$, and $[t(c_k) - 1, t(c_k))$. For all tuples $(a_i, b_j, c_k) \notin T'$ we schedule the job $L(a_i, b_j, c_k)$ in some arbitrary interval during $[3|T| + 3n, \infty)$ and the jobs $A(a_i, b_j, c_k)$, $B(a_i, b_j, c_k)$, and $C(a_i, b_j, c_k)$ during $[t(a_i, b_j, c_k), t(a_i, b_j, c_k) + 3)$. This solution yields an overall profit of $3|T| \cdot (M_1 + M_2) + \sum_{\ell=0}^{3n-1} (M_2 + 2^\ell) + (|T| - n) \cdot 3 \cdot M_2 = |T| \cdot (3 \cdot M_1 + 3 \cdot M_2) + |T| \cdot 3 \cdot M_2 + 2^{3n} - 1$. □

The following lemma shows the converse statement of the previous lemma. In a nutshell, in its proof we argue that

- for each triple (a_i, b_j, c_k) the machine is busy during the interval $[t(a_i, b_j, c_k), t(a_i, b_j, c_k) + 3)$, either with the job $L(a_i, b_j, c_k)$, and, if this is not the case, then the three jobs $A(a_i, b_j, c_k)$, $B(a_i, b_j, c_k)$, and $C(a_i, b_j, c_k)$ are scheduled there (the latter jobs give the most profit when scheduled there),
- some jobs of the types $A(a_i, b_j, c_k)$, $B(a_i, b_j, c_k)$, and $C(a_i, b_j, c_k)$ are scheduled during $[3|T|, t(c_n))$. Their respective non-zero profits during this interval are $2^{3n-i} + M_2$, $2^{2n-j} + M_2$, and $2^{n-k} + M_2$, respectively. Thus, apart from the additive term M_2 they are geometrically spaced. This ensures that in order to get a large enough total profit, one has to schedule exactly one job $A(a_i, b_j, c_k)$ for each $i \in \{1, \ldots, n\}$ during $[3|T|, t(a_n))$, and similarly for each $j, k \in \{1, \ldots, n\}$ one has to schedule exactly one job $B(a_i, b_j, c_k)$ and exactly one job $C(a_i, b_j, c_k)$ during $[3|T|, t(b_n))$ and during $[3|T|, t(c_n))$, respectively. This will then imply that we reduced from a "yes"-instance.

Lemma 12 *If there is an optimal solution of the GPSP instance yielding a profit of at least* $|T| \cdot (3 \cdot M_1 + 3 \cdot M_2) + |T| \cdot 3 \cdot M_2 + 2^{3n} - 1$*, then the instance of* 3- DIMENSIONAL MATCHING *is a "yes"-instance.*

Proof Given a solution to our defined GPSP instance, by Lemma 10 we can assume it is in standard form. Let $T' \subseteq T$ be the set of triples $(a_i, b_j, c_k) \in T$ corresponding to the jobs $L(a_i, b_j, c_k)$ that are scheduled during their respective intervals $[t(a_i, b_j, c_k), t(a_i, b_j, c_k) + 3)$. We claim that they form a feasible solution and that $|T'| = n$. The profit obtained by the jobs $L(a_i, b_j, c_k)$ corresponding to the triples in T' equals $|T'| \cdot (3 \cdot M_1 + 3 \cdot M_2)$. The total profit given by all other jobs $L(a_i, b_j, c_k)$ equals $(|T| - |T'|) \cdot 3 \cdot M_2$. For all jobs $A(a_i, b_j, c_k)$, $B(a_i, b_j, c_k)$, and $C(a_i, b_j, c_k)$ such that $L(a_i, b_j, c_k)$ is *not* scheduled during $[t(a_i, b_j, c_k), t(a_i, b_j, c_k) + 3)$ we can assume that the former are all scheduled during $[t(a_i, b_j, c_k), t(a_i, b_j, c_k) + 3)$ (since there they yield the maximum profit and no other job is scheduled there). The profit of those jobs equals then $(|T| - |T'|) \cdot 3(M_1 + M_2)$. Denote by J' the set of all other jobs $A(a_i, b_j, c_k)$, $B(a_i, b_j, c_k)$, and $C(a_i, b_j, c_k)$, i.e., such that $L(a_i, b_j, c_k)$ is scheduled during $[t(a_i, b_j, c_k), t(a_i, b_j, c_k) + 3))$. Their total profit equals $3|T'| \cdot M_2 + \sum_{A(a_i,b_j,c_k)\in J'} 2^{3n-i} + \sum_{B(a_i,b_j,c_k)\in J'} 2^{2n-j} + \sum_{C(a_i,b_j,c_k)\in J'} 2^{n-k}$. Thus, we can write the total profit of the solution as

$$\begin{aligned}
&|T'| \cdot (3 \cdot M_1 + 3 \cdot M_2) + (|T| - |T'|) \cdot 3 \cdot M_2 \\
&\quad +(|T| - |T'|) \cdot 3(M_1 + M_2) + 3|T'| \cdot M_2 \\
&\quad + \sum_{A(a_i,b_j,c_k)\in J'} 2^{3n-i} + \sum_{B(a_i,b_j,c_k)\in J'} 2^{2n-j} + \sum_{C(a_i,b_j,c_k)\in J'} 2^{n-k} \\
&\quad = |T| \cdot (3 \cdot M_1 + 3 \cdot M_2) + |T| \cdot 3 \cdot M_2 + \sum_{A(a_i,b_j,c_k)\in J'} 2^{3n-i} \\
&\quad + \sum_{B(a_i,b_j,c_k)\in J'} 2^{2n-j} + \sum_{C(a_i,b_j,c_k)\in J'} 2^{n-k}
\end{aligned}$$

By assumption, the total profit is at least $|T| \cdot (3 \cdot M_1 + 3 \cdot M_2) + |T| \cdot 3 \cdot M_2 + 2^{3n} - 1$. This implies that $\sum_{A(a_i,b_j,c_k)\in J'} 2^{3n-i} + \sum_{B(a_i,b_j,c_k)\in J'} 2^{2n-j} + \sum_{C(a_i,b_j,c_k)\in J'} 2^{n-k} \geq 2^{3n} - 1$.

Claim For each $i \in \{1, \ldots, n\}$ there is exactly one job $A(a_i, b_j, c_k) \in J'$, for each $j \in \{1, \ldots, n\}$ there is exactly one job $B(a_i, b_j, c_k) \in J'$, and for each $k \in \{1, \ldots, n\}$ there is exactly one job $C(a_i, b_j, c_k) \in J'$.

Proof of the claim As we assumed the solution to be in standard form, all jobs in J' are scheduled during $[3|T|, t(c_n))$.

First, we show that there must be a job $A(a_i, b_j, c_k) \in J'$ with $i = 1$. The maximum profit one can obtain during the interval $[3|T|, t(c_n))$ from jobs $A(a_i, b_j, c_k)$ with $i \geq 2$, and from jobs $B(a_i, b_j, c_k)$, $C(a_i, b_j, c_k)$ with arbitrary i is obtained by scheduling two jobs $A(a_2, b_j, c_k)$, $A(a_2, b_{j'}, c_{k'})$ during $[t(a_1) - 1, t(a_1) + 1) = [3|T|, 3|T| + 2)$, one job $A(a_i, b_j, c_k)$ during $[t(a_i) - 1, t(a_i))$ for each $i \geq 3$, one job $B(a_i, b_j, c_k)$ during $[t(b_j) - 1, t(b_j))$ for each j, and one job $C(a_i, b_j, c_k)$ during $[t(c_k) - 1, t(c_k))$ for each k. This yields a profit of $2^{3n-2} + \sum_{i=2}^{n} 2^{3n-i} + \sum_{j=1}^{n} 2^{2n-j} + \sum_{k=1}^{n} 2^{n-k} = 2^{3n-2} + 2^{3n-1} - 1 < 2^{3n} - 1$. Therefore, since $\sum_{A(a_i,b_j,c_k)\in J'} 2^{3n-i} + \sum_{B(a_i,b_j,c_k)\in J'} 2^{2n-j} + \sum_{C(a_i,b_j,c_k)\in J'} 2^{n-k} \geq 2^{3n} - 1$ this implies that there must be one job $A(a_i, b_j, c_k) \in J'$ with $i = 1$. Let $A(a_1, b_{j_1}, c_{k_1}) \in J'$ be this job.

As our solution is in standard form, we know that $A(a_1, b_{j_1}, c_{k_1})$ must be scheduled in the interval $[3|T|, 3|T| + 1)$. Thus, all jobs in $J' \setminus \{A(a_1, b_{j_1}, c_{k_1})\}$ are scheduled during $[3|T| + 1, 3|T| + n)$ and they yield a total profit of at least $2^{3n} - 1 - 2^{3n-1} = 2^{3n-1} - 1$. With a similar argumentation as above we can show that for each $i \in \{2, \ldots, n\}$ there is exactly one job $A(a_i, b_j, c_k) \in J'$, and similarly for each $j \in \{1, \ldots, n\}$ and each $k \in \{1, \ldots, n\}$ there is exactly one job $B(a_i, b_j, c_k) \in J'$ and one job $C(a_i, b_j, c_k) \in J'$. This completes the proof of the claim. □

Now observe that the jobs in J' correspond exactly to the elements a_i, b_j, c_k of the triples in T'. The above claim implies that each element in $A \cup B \cup C$ appears exactly once in them. Thus, the triples T' form a feasible solution to the instance of 3-DIMENSIONAL MATCHING that we reduced from which is therefore a “yes”-instance. □

Lemmas 11 and 12 together show the correctness of the above reduction. Since by Lemma 10 we can restrict ourselves to solutions in standard form and those are non-preemptive, our reduction works for the preemptive as well as for the non-preemptive setting.

Theorem 11 *The General Profit Scheduling Problem (GPSP) is* NP*-hard, even if the processing time of each job is either exactly 1 or 3. This holds in the preemptive as well as in the non-preemptive setting.*

Using the fact that solutions in standard form schedule all jobs, we can modify the above transformation to show that also the GSP is NP-hard if the processing time of each job is either exactly 1 or 3. All we need to do is to define a cost function w_j for each job by setting $w_j(t) := 3 \cdot M_1 + 3 \cdot M_2 - f_j(t)$, which in particular ensures that for no completion time any job has negative costs. Then the goal is to find a feasible schedule on one machine to minimize $\sum_j w_j(C_j)$.

Corollary 3 *The GSP is* NP*-hard, even if the processing time of each job is either exactly 1 or 3. This holds in the preemptive setting as well as in the non-preemptive setting.*

6 Discussion and future directions

In this paper, we obtained the first fixed-parameter algorithms for several fundamental problems in scheduling. We identified key problem parameters that determine the complexity of an instance, so that when these parameters are fixed the problem becomes polynomial time solvable.

There are several interesting open questions in the connection of scheduling and FPT. One important scheduling objective is weighted flow time (which in the approximative sense is not completely understood yet, even on one machine). It would be interesting to investigate whether this problem is FPT when parametrized by, for instance, an upper bound on the maximum processing time and the maximum weight of a job (assuming integral input data). For scheduling jobs non-preemptively on identical machines to minimize the makespan, in Sect. 2 we proved the first FPT result for this problem. It would be interesting to extend this to the setting with release dates or to consider as parameter the number of distinct processing times (rather than an upper bound on those). Recently the problem was shown to be polynomial time solvable for any constant number of distinct processing times [22], but the given algorithm is not a fixed-parameter algorithm. When the jobs have precedence constraints, a canonical parameter would be the width of the partial order given by these constraints, and it would be interesting to study whether $P|prec|C_{\max}$ admits fixed-parameter algorithms when parameterized by the partial order width and additionally by the maximum processing time. For scheduling with rejection, we further suggest the number of distinct rejection costs as a parameter to investigate. Finally, it would be desirable to understand the kernelizability of all scheduling problems we considered here.

Acknowledgments We thank the reviewers of IPCO 2014 and *Mathematical Programming* for their remarks on improving the presentation of the results. We further thank an anonymous reviewer of an earlier version for suggestions how to improve the algorithms in Sect. 4 and to prove Theorem 9, as well as Akiyoshi Shioura for helpful discussions about convexity. Finally, we would like to thank the anonymous reviewers for pointing out the IP formulation in Sect. 2.2 which is simpler than our original formulation.

References

1. Afrati, F., Bampis, E., Chekuri, C., Karger, D., Kenyon, C., Khanna, S., Milis, I., Queyranne, M., Skutella, M., Stein, C., Sviridenko, M.: Approximation schemes for minimizing average weighted completion time with release dates. In: Proc. FOCS, pp. 32–43 (1999)
2. Alon, N., Azar, Y., Woeginger, G.J., Yadid, T.: Approximation schemes for scheduling on parallel machines. J. Sched. **1**(1), 55–66 (1998)
3. Alon, N., Yuster, R., Zwick, U.: Color-coding. J. ACM **42**(4), 844–856 (1995)
4. Bansal, N., Dhamdhere, K.: Minimizing weighted flow time. ACM Trans. Algorithms **3**(4), 39 (2007)
5. Bansal, N., Pruhs, K.: The geometry of scheduling. In: Proc. FOCS, pp. 407–414 (2010)
6. Bansal, N., Pruhs, K.: Weighted geometric set multi-cover via quasi-uniform sampling. In: Algorithms-ESA, pp. 145–156. Springer, Berlin (2012)
7. Bessy, S., Giroudeau, R.: Some parametric complexity results on a coupled-task scheduling problem, personal communication (2013)
8. Bodlaender, H.L., Fellows, M.R.: $W[2]$-hardness of precedence constrained K-processor scheduling. Oper. Res. Lett. **18**(2), 93–97 (1995)
9. Brauner, N., Crama, Y., Grigoriev, A., van de Klundert, J.: A framework for the complexity of high-multiplicity scheduling problems. J. Comb. Optim. **9**(3), 313–323 (2005)
10. Chekuri, C., Khanna, S.: Approximation schemes for preemptive weighted flow time. In: Proc. STOC, pp. 297–305 (2002)
11. Chekuri, C., Khanna, S., Zhu, A.: Algorithms for minimizing weighted flow time. In: Proc. STOC, pp. 84–93 (2001)
12. Chu, G., Gaspers, S., Narodytska, N., Schutt, A., Walsh, T.: On the complexity of global scheduling constraints under structural restrictions. In: Proc. IJCAI (2013)
13. Chudak, F.A., Hochbaum, D.S.: A half-integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine. Oper. Res. Lett. **25**(5), 199–204 (1999)
14. Coffman E.G. Jr., Garey, M.R., Johnson, D.S.: An application of bin-packing to multiprocessor scheduling. SIAM J. Comput. **7**, 1–17 (1978)
15. Ebenlendr, T., Krčál, M., Sgall, J.: Graph balancing: a special case of scheduling unrelated parallel machines. In: Proc. SODA, pp. 483–490 (2008)
16. Engels, D.W., Karger, D.R., Kolliopoulos, S.G., Sengupta, S., Uma, R.N., Wein, J.: Techniques for scheduling with rejection. J. Algorithms **49**(1), 175–191 (2003)
17. Fellows, M.R., Gaspers, S., Rosamond, F.A.: Parameterizing by the number of numbers. Theory Comput. Syst. **50**(4), 675–693 (2012)
18. Fellows, M.R., Koblitz, N.: Fixed-parameter complexity and cryptography. In: San Juan, P.R. (ed.) Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, Lecture Notes Comput. Sci., vol. 673, pp. 121–131 (1993)
19. Fellows, M.R., McCartin, C.: On the parametric complexity of schedules to minimize tardy tasks. Theor. Comput. Sci. **298**(2), 317–324 (2003)
20. Friesen, D.K.: Tighter bounds for the multifit processor scheduling algorithm. SIAM J. Comput. **13**, 170–181 (1984)
21. Garey, M.R., Johnson, D.S.: Computers and Intractability. W. H. Freeman and Co., San Francisco (1979)
22. Goemans, M.X., Rothvoß, T.: Polynomiality for bin packing with a constant number of item types. In: Proc. SODA, pp. 830–839 (2014)
23. Graham, R.L.: Bounds on multiprocessing timing anomalies. SIAM J. Appl. Math. **17**, 263–269 (1969)
24. Heinz, S.: Complexity of integer quasiconvex polynomial optimization. J. Complex. **21**(4), 543–556 (2005)
25. Hochbaum, D.S., Shmoys, D.B.: Using dual approximation algorithms for scheduling problems: theoretical and practical results. J. ACM **34**, 144–162 (1987)

26. Hoogeveen, H., Skutella, M., Woeginger, G.J.: Preemptive scheduling with rejection. Math. Program. **94**(2–3, Ser. B), 361–374 (2003)
27. Jansen, K., Kratsch, S., Marx, D., Schlotter, I.: Bin packing with fixed number of bins revisited. J. Comput. Syst. Sci. **79**(1), 39–49 (2013)
28. Köppe, M.: On the complexity of nonlinear mixed-integer optimization. In: Lee, J., Leyffer, S. (eds.) Mixed Integer Nonlinear Programming, The IMA Volumes in Mathematics and its Applications, vol. 154, pp. 533–557 (2012)
29. Langston, M.A.: Processor scheduling with improved heuristic algorithms. Ph.D. thesis, Texas A&M University (1981)
30. Lenstra, J., Kan, A.R., Brucker, P.: Complexity of machine scheduling problems. In: Studies in Integer Programming, Ann. Discrete Math., vol. 1, pp. 343–362 (1977)
31. Lenstra, J.K., Shmoys, D.B., Tardos, É.: Approximation algorithms for scheduling unrelated parallel machines. Math. Program. **46**(1–3), 259–271 (1990)
32. Marx, D.: Fixed-parameter tractable scheduling problems. In: Packing and Scheduling Algorithms for Information and Communication Services (Dagstuhl Seminar 11091), vol. 1, p. 86 (2011)
33. Marx, D., Schlotter, I.: Stable assignment with couples: parameterized complexity and local search. Discret. Optim. **8**(1), 25–40 (2011)
34. Mnich, M., Wiese, A.: Scheduling and fixed-parameter tractability. In: Proc. IPCO 2014, Lecture Notes Comput. Sci., vol. 8494, pp. 381–392 (2014)
35. Sahni, S.: Algorithms for scheduling independent tasks. J. ACM **23**, 116–127 (1976)
36. Shmoys, D.B., Tardos, É.: An approximation algorithm for the generalized assignment problem. Math. Program. **62**(1–3), 461–474 (1993)
37. Smith, W.E.: Various optimizers for single-stage production. Nav. Res. Logist. Q. **3**, 59–66 (1956)
38. Svensson, O.: Santa claus schedules jobs on unrelated machines. SIAM J. Comput. **41**(5), 1318–1341 (2012)
39. Sviridenko, M., Wiese, A.: Approximating the configuration-LP for minimizing weighted sum of completion times on unrelated machines. In: Proc. IPCO 2013, Lecture Notes Comput. Sci., vol. 7801, pp. 387–398 (2013)