Scheduling and Fixed-Parameter Tractability

Matthias Mnich[1] and Andreas Wiese[2]

[1] Cluster of Excellence MMCI, Saarbrücken, Germany
m.mnich@mmci.uni-saarland.de
[2] Max Planck Institute for Computer Science, Saarbrücken, Germany
awiese@mpi-inf.mpg.de

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 FPT-algorithms. In particular, our positive and negative results for scheduling with rejection explore a research direction proposed by Dániel Marx.

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 [1,2].

However, the running time of these approximation schemes usually has a bad dependence on ϵ, and in practise 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).

1.1 Our Contributions

In this paper we present the first fixed-parameter algorithms for several fundamental scheduling problems. In Section 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 where our variables model how many jobs of each size are scheduled on each machine. To ensure that an assignment of all given jobs to these "slots" exists we argue via Hall's Theorem and ensure that for each subset of jobs there are enough usable slots. 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. [3]. Note that if the number of machines or the number of processing times are constant, the problem is still NP-hard [4], and thus no fixed-parameter algorithms can exist for those cases (if P $\neq$ NP).

Then, in Section 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. 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). Our results respond to a question by Dániel Marx [5] for investigating the fixed-parameter tractability of scheduling with rejection.

Finally, in Section 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)* [6]. 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 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, respectively, see Table 1. Our contributions are summarized in Table 1.

Due to space constraints, proofs are deferred to the full version of this paper.

1.2 Related Work

Scheduling. One 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 [7]. After a series of

Table 1. Summary of our results. For a job j we denote by p_j its processing time, by w_j its weight, by e_j its rejection cost, by f_j its cost function. and by C_j its completion time in a computed schedule.

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

improvements [8,9,10,11], Hochbaum and Shmoys present a polynomial time approximation scheme (PTAS), even if the number of machines is part of the input [2]. On unrelated machines, the problem is NP-hard to approximate with a better factor than 3/2 [12,4] and there is a 2-approximation algorithm [4] that extends to the generalized assignment problem [13]. 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 [14] 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 General Scheduling Problem (GSP). In its full generality, Bansal and Pruhs [6] give a $O(\log \log nP)$-approximation, where P is the maximum ratio of processing times. One particularly important special case is the weighted flow time objective where previously to Bansal and Pruhs [6] the best known approximation factors where $O(\log^2 P)$, $O(\log W)$, and $O(\log nP)$ [15,16]; where W is the maximum ratios of job weights. Also, a quasi-PTAS with running time $n^{O(\log P \log W)}$ is known [17].

A generalization of classical scheduling problems is *scheduling with rejection*. There, each job j is has additionally 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. [18] 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 [19] give a PTAS for arbitrary release dates. For objective the makespan minimization and given multiple machines, Hoogeveen et al. [20] 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 Brauner et al. [21].

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 [22] and scheduling unit-length tasks with deadlines and precedence constraints and k tardy tasks is W[1]-hard [23], for parameter k. Mere exemptions seem to be an algorithm by Marx and Schlotter [24] 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. [25] 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. [26] 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. [3].

2 Minimizing the Makespan

We first consider the problem $P||C_{\max}$, where a given a 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 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 $p_j = t$; we define $J_t := \{j \in J \mid p_j = t\}$. 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, which can be proven by starting with an arbitrary optimal schedule and exchanging jobs carefully between 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.*

Denote by $J' = J \setminus J_0$ be 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.)

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

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

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

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 deep results about integer programming in fixed dimension. As we will need it later, we cite here an algorithm due to Heinz [27,28] that can even minimize over convex spaces described by (quasi-)convex polynomials, rather than only over polytopes.

Theorem 1 ([27,28]). *Let $f, g_1, \ldots, g_m \in \mathbb{Z}[x_1, \ldots, x_t]$ be quasi-convex polynomials of degree at most $d \geq 2$, 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 $\mathbf{x}^\star \in \mathbb{Z}^t$ of the following problem (4) or reports that no minimizer exists. If the algorithm outputs a minimizer $\mathbf{x}^\star$, its binary encoding size is $\ell \cdot d^{O(t)}$.*

$$\min f(x_1, \ldots, x_t), \quad \text{subject to } g_i(x_1, \ldots, x_t) \leq 0, \quad i = 1, \ldots, m \quad \mathbf{x} \in \mathbb{Z}^t \ . \tag{4}$$

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.

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

Recall that without choosing a parameter, problem $P||C_{\max}$ is strongly NP-hard (as it contains 3-Partition). When parameterizing $P||C_{\max}$ by the number $\overline{p}$ of distinct processing times, Lemma 1 is no longer true (details deferred to full version of this paper). However, for constantly many processing times the problem was recently shown to be polynomial time solvable for any constant $\overline{p}$ [29].

2.1 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 of machines m of the instance.

We model this problem as an integer program in fixed dimension. Denote by $q_1, \dots, q_{\overline{p}}$ the distinct finite processing times in a given instance. For each combination of a machine i and a finite processing time q_ℓ we introduce a variable $y_{i,\ell} \in \{0, \dots, n\}$ that models how many jobs of processing time q_ℓ are assigned to i. Note that the number of these variables is bounded by $m \cdot \overline{p}$. An assignment to these variables can be understood as allocating $y_{i,\ell}$ slots for jobs with processing time q_ℓ to machine i, without specifying what actual jobs are assigned to these slots. Assigning the jobs to the slots can be understood as a bipartite matching: introduce one vertex v_j for each job j, one vertex $w_{s,i}$ for each slot s on each machine i, and an edge $\{v_j, w_{s,i}\}$ whenever job j has the same size on machine i as slot s. According to Hall's Theorem, there is a matching in which each job is matched if and only if for each set of jobs $J' \subseteq J$ there are at least $|J'|$ slots to which at least one job in J' can be assigned. For one single set J' the latter can be expressed by a linear constraint; however, the number of subsets J' is exponential. We overcome this as follows: 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. As we will show, it suffices to add a constraint for sets of jobs J' such that for each job type either all or none of the jobs of that type are contained in J' (those sets „dominate" all other sets). This gives rise to the following integer program. Denote by Z the set of all job types and for each $z \in Z$ denote by $J_z \subseteq J$ the set of jobs of type z. For each set $Z' \subseteq Z$ denote by $Q_{i,Z'}$ the set of distinct finite processing times of jobs of types in Z' on machine i. Using Theorem 1 we can solve the following IP:

$$\min\ T \quad \text{s.t.} \sum_{\ell \in \{1,\dots,\overline{p}\}} y_{i,\ell} \cdot q_\ell \le T, \qquad i = 1, \dots, m \tag{5}$$

$$\sum_{z \in Z'} |J_z| \le \sum_i \sum_{\ell : q_\ell \in Q_{i,Z'}} y_{i,\ell} \quad \forall\, Z' \subseteq Z \tag{6}$$

$$y_{i,\ell} \in \{0, \dots, n\} \qquad i = 1, \dots, m,\ \ell = 1, \dots, \overline{p} \tag{7}$$

$$T \ge 0 \tag{8}$$

Theorem 3. *Instances of $R||C_{\max}$ with m machines and n jobs with $\overline{p}$ distinct finite processing times $q_1, \dots, q_{\bar{p}}$ can be solved in time $f(\overline{p}, m) \cdot (n + \log \max_\ell q_\ell)^{O(1)}$ for a suitable function f.*

A natural question is the case when only the number of machines is a fixed parameter. Even if one requires the processing times to be polynomially bounded, then already $P||C_{\max}$ is still W[1]-hard [30]. On the other hand, $R||C_{\max}$ is NP-hard if only processing times $\{1, 2, \infty\}$ are allowed [12,4]. This justifies to take both m and $\bar{p}$ as a parameters in the unrelated machine case.

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 $\sum_{j \in J \setminus J'} w_j C_j + \sum_{j \in J'} e_j$ where C_j denotes the completion time in the computed schedule.

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 $n'' > n'$ (proof deferred). 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 [31, 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, problem becomes 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 [18].) 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 [32].

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

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. [3]. 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 $1, \ldots, \tau$ such $w^{(i)}/p^{(i)} \geq w^{(i+1)}/p^{(i+1)}$ for each $i \in \{1, \ldots, \tau - 1\}$. 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

$$\sum_{\ell=1}^{x_i} w^{(i)} \cdot (\ell \cdot p^{(i)} + \sum_{i'<i} x_{i'} \cdot p^{(i')}) = w^{(i)} \cdot x_i \cdot \sum_{i'<i} x_{i'} \cdot p^{(i')} + w^{(i)} \cdot p^{(i)} \sum_{\ell=1}^{x_i} \ell$$

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

Note that $s_i(x)$ is a convex polynomial of degree 2 (being the sum of quadratic polynomials with only positive coefficients). 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 $j_1^{(i)}, \dots, j_{n_i}^{(i)}$ 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 $f_i(s)$ 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 2 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 2. *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 $s \in \{0, \dots, n_i\}$.*

Lemma 2 allows modeling the entire problem with the following convex program, where for each type i, variable g_i models the rejection cost for jobs of type i.

$$\min \sum_{i=1}^{\tau} g_i + s_i(x) \text{ s.t. } \sum_{i=1}^{\tau} (n_i - x_i) \le k,$$
$$g_i \ge p_i^{(\ell)}(x_i) \;\forall\, i \in \{1, \dots, \tau\} \;\forall\, \ell \in \{1, \dots, n_i\}, \mathbf{g}, \mathbf{x} \in \mathbb{Z}^\tau \ge 0\;. \tag{9}$$

Observe that (9) admits an optimal solution with $g_i = \max_\ell p_i^{(\ell)}(x_i) = f_i(x_i)$ for each i. Thus, solving (9) yields an optimal solution to the overall instance.

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

4 Profit Maximization for General Scheduling Problems

The parameterized dual problem of scheduling jobs with rejection is 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. This is equivalent to the following problem where here we allow even non-trivial release dates and job dependent profit functions:

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 $f_j(t)$. 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 difficult scheduling objectives such as weighted flow time or weighted tardiness.

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 preemptive or non-preemptive schedule in time $2^{O(s)}s^{O(\overline{p})}n^4 \log n$.*

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. The next theorem assumes that numeric input values are allowed to be exponentially large.

Theorem 8. *(Non-)preemptive s-GPSP is* W[1]*-hard for parameter s.*

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 related General Scheduling Problem (GSP) [6]. While all processing times are either 1 or 3, in our reduction we use cost functions whose values can be exponentially large.

Theorem 9. *The General Profit Scheduling Problem (GPSP) and the General Scheduling Problem (GSP) are (weakly)* NP*-hard, even if $p_j \in \{1, 3\}$ for each job j. This holds in both the preemptive and non-preemptive setting.*

Acknowledgment. We thank the IPCO reviewers, as well as an anonymous reviewer of an earlier version for suggestions how to improve the algorithms in Sect. 4 and to prove Theorem 8.

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 1999, pp. 32–43 (1999)
2. Hochbaum, D.S., Shmoys, D.B.: Using dual approximation algorithms for scheduling problems: Theoretical and practical results. J. ACM 34, 144–162 (1987)
3. Fellows, M.R., Gaspers, S., Rosamond, F.A.: Parameterizing by the number of numbers. Theory Comput. Syst. 50(4), 675–693 (2012)
4. Lenstra, J.K., Shmoys, D.B., Tardos, É.: Approximation algorithms for scheduling unrelated parallel machines. Mathematical Programming 46(1-3), 259–271 (1990)
5. 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)
6. Bansal, N., Pruhs, K.: The geometry of scheduling. In: Proc. FOCS 2010, pp. 407–414 (2010)
7. Graham, R.L.: Bounds on multiprocessing timing anomalies. SIAM J. Appl. Math. 17, 263–269 (1969)
8. Coffman Jr., E.G., Garey, M.R., Johnson, D.S.: An application of bin-packing to multiprocessor scheduling. SIAM J. Comput. 7, 1–17 (1978)
9. Friesen, D.K.: Tighter bounds for the multifit processor scheduling algorithm. SIAM J. Comput. 13, 170–181 (1984)
10. Langston, M.A.: Processor scheduling with improved heuristic algorithms. PhD thesis, Texas A&M University (1981)

11. Sahni, S.: Algorithms for scheduling independent tasks. J. ACM 23, 116–127 (1976)
12. Ebenlendr, T., Krčál, M., Sgall, J.: Graph balancing: a special case of scheduling unrelated parallel machines. In: Proc. SODA 2008, pp. 483–490 (2008)
13. Shmoys, D.B., Tardos, É.: An approximation algorithm for the generalized assignment problem. Mathematical Programming 62(5), 461–474 (1993)
14. Svensson, O.: Santa claus schedules jobs on unrelated machines. SIAM J. Comput. 41(4), 1318–1341 (2012)
15. Bansal, N., Dhamdhere, K.: Minimizing weighted flow time. ACM Trans. Algorithms 3(4) (November 2007)
16. Chekuri, C., Khanna, S., Zhu, A.: Algorithms for minimizing weighted flow time. In: Proc. STOC 2001, pp. 84–93 (2001)
17. Chekuri, C., Khanna, S.: Approximation schemes for preemptive weighted flow time. In: Proc. STOC 2002, pp. 297–305 (2002)
18. 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)
19. Sviridenko, M., Wiese, A.: Approximating the configuration-LP for minimizing weighted sum of completion times on unrelated machines. In: Goemans, M., Correa, J. (eds.) IPCO 2013. LNCS, vol. 7801, pp. 387–398. Springer, Heidelberg (2013)
20. Hoogeveen, H., Skutella, M., Woeginger, G.J.: Preemptive scheduling with rejection. Math. Program. 94(2-3, Ser. B), 361–374 (2003)
21. 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)
22. Bodlaender, H.L., Fellows, M.R.: $W[2]$-hardness of precedence constrained K-processor scheduling. Oper. Res. Lett. 18(2), 93–97 (1995)
23. Fellows, M.R., McCartin, C.: On the parametric complexity of schedules to minimize tardy tasks. Theoret. Comput. Sci. 298(2), 317–324 (2003)
24. Marx, D., Schlotter, I.: Stable assignment with couples: Parameterized complexity and local search. Discr. Optimization 8(1), 25–40 (2011)
25. Alon, N., Azar, Y., Woeginger, G.J., Yadid, T.: Approximation schemes for scheduling on parallel machines. J. Sched. 1(1), 55–66 (1998)
26. Chu, G., Gaspers, S., Narodytska, N., Schutt, A., Walsh, T.: On the complexity of global scheduling constraints under structural restrictions. In: Proc. IJCAI 2013 (2013)
27. Heinz, S.: Complexity of integer quasiconvex polynomial optimization. J. Complexity 21(4), 543–556 (2005)
28. Köppe, M.: On the complexity of nonlinear mixed-integer optimization. In: Mixed Integer Nonlinear Programming. The IMA Volumes in Mathematics and its Applications, vol. 154, pp. 533–557 (2012)
29. Goemans, M.X., Rothvoß, T.: Polynomiality for bin packing with a constant number of item types. In: Proc. SODA 2014 (to appear, 2014)
30. Jansen, K., Kratsch, S., Marx, D., Schlotter, I.: Bin packing with fixed number of bins revisited. J. Comput. System Sci. 79(1), 39–49 (2013)
31. 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)
32. 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)