Universal Sequencing on a Single Machine

Leah Epstein[1], Asaf Levin[2], Alberto Marchetti-Spaccamela[3,⋆], Nicole Megow[4], Julián Mestre[4], Martin Skutella[5,⋆⋆], and Leen Stougie[6,⋆⋆⋆]

[1] Dept. of Mathematics, University of Haifa, Israel
lea@math.haifa.ac.il
[2] Chaya fellow. Faculty of Industrial Engineering and Management, The Technion, Haifa, Israel
levinas@ie.technion.ac.il
[3] Dept. of Computer and System Sciences, Sapienza University of Rome, Italy
alberto.marchetti@dis.uniroma1.it
[4] Max-Planck-Institut für Informatik, Saarbrücken, Germany
{nmegow,jmestre}@mpi-inf.mpg.de
[5] Inst. für Mathematik, Technische Universität Berlin, Germany
martin.skutella@tu-berlin.de.
[6] Dept. of Econometrics and Operations Research, Vrije Universiteit Amsterdam & CWI, Amsterdam, The Netherlands
stougie@cwi.nl

Abstract. We consider scheduling on an unreliable machine that may experience unexpected changes in processing speed or even full breakdowns. We aim for a universal solution that performs well without adaptation for any possible machine behavior. For the objective of minimizing the total weighted completion time, we design a polynomial time deterministic algorithm that finds a universal scheduling sequence with a solution value within 4 times the value of an optimal clairvoyant algorithm that knows the disruptions in advance. A randomized version of this algorithm attains in expectation a ratio of e. We also show that both results are best possible among all universal solutions. As a direct consequence of our results, we answer affirmatively the question of whether a constant approximation algorithm exists for the offline version of the problem when machine unavailability periods are known in advance.

When jobs have individual release dates, the situation changes drastically. Even if all weights are equal, there are instances for which any universal solution is a factor of $\Omega(\log n/\log\log n)$ worse than an optimal sequence. Motivated by this hardness, we study the special case when the processing time of each job is proportional to its weight. We present a non-trivial algorithm with a small constant performance guarantee.

1 Introduction

Traditional scheduling problems normally assume that jobs run on an ideal machine that provides a constant performance throughout time. While in some settings this is a good

⋆ Supported by EU project 215270 FRONTS.
⋆⋆ Supported by DFG research center MATHEON in Berlin.
⋆⋆⋆ Supported by the Dutch BSIK-BRIKS project.

F. Eisenbrand and B. Shepherd (Eds.): IPCO 2010, LNCS 6080, pp. 230–243, 2010.

enough approximation of real life machine behavior, in other situations this assumption is decidedly unreasonable. Our machine, for example, can be a server shared by multiple users; if other users suddenly increase their workload, this can cause a general slowdown; or even worse, the machine may become unavailable for a given user due to priority issues. In other cases, our machine may be a production unit that can break down altogether and remain offline for some time until it is repaired. In these cases, it is crucial to have schedules that take such unreliable machine behavior into account.

Different machine behaviors will typically lead to widely different optimal schedules. This creates a burden on the scheduler who would have to periodically recompute the schedule from scratch. In some situations, recomputing the schedule may not even be feasible: when submitting a set of jobs to a server, a user can choose the order in which it presents these jobs, but cannot alter this ordering later on. Therefore, it is desirable in general to have a fixed master schedule that will perform well regardless of the actual machine behavior. In other words, we want a *universal schedule* that, for any given machine behavior, has cost close to that of an optimal clairvoyant algorithm.

In this paper we initiate the study of universal scheduling by considering the problem of sequencing jobs on a single machine to minimize average completion times. Our main result is an algorithm for computing a universal schedule that is always a constant factor away from an optimal clairvoyant algorithm. We complement this by showing that our upper bound is best possible among universal schedules. We also consider the case when jobs have release dates. Here we provide an almost logarithmic lower bound on the performance of universal schedules, thus showing a drastic difference with respect to the setting without release dates. Finally, we design an algorithm with constant performance for the interesting case of scheduling jobs with release dates and proportional weights. Our hope is that these results stimulate the study of universal solutions for other scheduling problems, and, more broadly, the study of more realistic scheduling models. In the rest of this section we introduce our model formally, discuss related work, and explain our contributions in detail.

The model. We are given a job set J with processing times $p_j \in \mathbb{Q}^+$ and weights $w_j \in \mathbb{Q}^+$ for each job $j \in J$. Using a standard scaling argument, we can assume w.l.o.g. that $w_j \geq 1$ for $j \in J$. The problem is to find a sequence π of jobs to be scheduled on a single machine that minimizes the total sum of weighted completion times. The jobs are processed in the prefixed order π no matter how the machine may change its processing speed or whether it becomes unavailable. In case of a machine breakdown the currently running job is preempted and will be resumed processing at any later moment when the machine becomes available again. We analyze the worst case performance by comparing the solution value provided by an algorithm with that of an optimal clairvoyant algorithm that knows the machine behavior in advance, and that is even allowed to preempt jobs at any time.

We also consider the more general problem in which each job $j \in J$ has its individual release date $r_j \geq 0$, which is the earliest point in time when it can start processing. In this model, it is necessary to allow job preemption, otherwise no constant performance guarantee is possible as simple examples show. We allow preemption in the actual scheduling procedure, however, as in the case without release dates, we aim for non-adaptive universal solutions. That is, a schedule will be specified by a total ordering

of the jobs. At any point in time we work on the first job in this ordering that has not finished yet and that has already been released. This procedure is called *preemptive list scheduling* [9, 28]. Note that a newly released job will preempt the job that is currently running if it comes earlier than the current job in the ordering.

Related work. The concept of *universal* solutions, that perform well for every single input of a superset of possible inputs, has been used already decades ago in different contexts, as e.g. in hashing [4] and routing [31]. The latter is also known as *oblivious routing* and has been studied extensively; see [26] for a state-of-the-art overview. Jia et al. [12] considered universal approximations for TSP, Steiner Tree, and Set Cover Problems. All this research falls broadly into the field of robust optimization [3]. The term *robust* is not used consistently in the literature. In particular, the term *robust scheduling* refers mainly to robustness against uncertain processing times; see e.g. [17, chap. 7] and [23]. Here, quite strong restrictions on the input or weakened notions of robustness are necessary to guarantee meaningful worst case solutions. We emphasize, that our results in this paper are robust in the most conservative, classical notion of robustness originating by Soyster [30], also called *strict robustness* [22], and in this regard, we follow the terminology of universal solutions.

Scheduling with limited machine availability is a subfield of machine scheduling that has been studied for over twenty years; see, e.g., the surveys [27, 20, 7]. Different objective functions, stochastic breakdowns, as well as the offline problem with known availability periods have been investigated. Nevertheless, only few results are known on the problem of scheduling to minimize the total weighted completion time, and none of these deal with release dates. If all jobs have equal weights, a simple interchange argument shows that sequencing jobs in non-increasing order of processing times is optimal as it is in the setting with continuous machine availability [29]. Obviously, this result immediately transfers to the universal setting in which machine breakdowns or changes in processing speeds are not known beforehand. The special case of proportional jobs, in which the processing time of each job is proportional to its weight, has been studied in [32]. The authors showed that scheduling in non-increasing order of processing times (or weights) yields a 2-approximation for preemptive scheduling. However, for the general problem with arbitrary job weights, it remained an open question [32] if a polynomial time algorithm with constant approximation ratio exists, even without release dates. In this case, the problem is strongly NP-hard [32].

A major line of research within this area focused on the offline scheduling problem with a single unavailable period. This problem is weakly NP-hard in both, the preemptive [19] and the non-preemptive variant [1, 21]. Several approximation results have been derived, see [19, 21, 32, 13, 24]. Only very recently, and independently of us, Kellerer and Strusevich [16] derived FPTASes with running time $\mathcal{O}(n^4/\epsilon^2)$ for the non-preemptive problem and $\mathcal{O}(n^6/\epsilon^3)$ in the preemptive case. An even improved non-preemptive FPTAS with running time $\mathcal{O}(n^2/\epsilon^2)$ is claimed in [14]. However, the proof seems incomplete in bounding the deviation of an algorithm's solution from an optimal one; in particular, the claim after Ineq. (11) in the proof of Lem. 1 is not proved.

Our results. Our main results are algorithms that compute deterministic and randomized universal schedules for jobs without release dates. These algorithms run in polynomial time and output an ordering of the jobs such that scheduling the jobs in this order will

always yield a solution that remains within multiplicative factor 4 and within multiplicative factor e in expectation from any given schedule. Furthermore, we show that our algorithms can be adapted to solve more general problem instances with certain types of precedence constraints without loosing performance quality. We also show that our upper bounds are best possible for universal scheduling. This is done by establishing an interesting connection between our problem and a certain online bidding problem [5].

It may seem rather surprising that universal schedules with constant performance guarantee should always exist. In fact, our results immediately answer affirmatively an open question in the area of offline scheduling with limited machine availability: whether there exists a constant factor approximation algorithm for scheduling jobs in a machine having multiple unavailable periods that are known in advance.

To derive our results, we study the objective of minimizing the total weight of uncompleted jobs at any point in time. First, we show that the performance guarantee is given directly by a bound on the ratio between the remaining weight of our algorithm and that of an optimal clairvoyant algorithm at every point in time on an ideal. Then, we devise an algorithm that computes the job sequence iteratively backwards: in each iteration we find a subset of jobs with largest total processing time subject to a bound on their total weight. The bound is doubled in each iteration. Our approach is related to, but not equivalent to, an algorithm of Hall et al. [9] for online scheduling on ideal machines—the doubling there happens in the time horizon. Indeed, this type of *doubling* strategy has been applied successfully in the design of algorithms for various problems; the interested reader is referred to the excellent survey of Chrobak and Kenyon-Mathieu [6] for a collection of such examples.

The problem of minimizing the total weight of uncompleted jobs at any time was previously considered [2] in the context of on-line scheduling to minimize flow time on a single machine; there, a constant approximation algorithm is presented with a worst case bound of 24. Our results imply an improved 4-approximation for this problem. Furthermore, we show that the same guarantee holds for the setting with release dates; unfortunately, unlike in the case without release dates, this does not translate into the same performance guarantee for universal schedules. In fact, when jobs have individual release dates, the problem changes drastically.

In Section 4 we show that in the presence of release dates, even if all weights are equal, there are instances for which the ratio between the value of any universal solution and that of an optimal schedule is $\Omega(\log n / \log\log n)$. Our proof relies on the classical theorem of Erdős and Szekeres [8] on the existence of long increasing/decreasing subsequences of a given sequence of numbers. Motivated by this hardness, we study the class of instances with proportional jobs. We present a non-trivial algorithm and prove a performance guarantee of 5. Additionally, we give a lower bound of 3 for all universal solutions in this special case.

Our last result, Section 5, is a fully polynomial time approximation scheme (FPTAS) for offline scheduling on a machine with a single unavailable period. Compared to the FPTAS presented recently in [16], our scheme, which was discovered independently from the former, is faster and seems to be simpler, even though the basic ideas are similar. Our FPTAS for the non-preemptive variant has running time $\mathcal{O}(n^3/\epsilon^2)$ and for the preemptive variant $\mathcal{O}(n^4/\epsilon^3 \log p_{\max})$.

2 Preliminaries and Key Observations

Given a single machine that runs continuously at unit speed (ideal machine), the completion time C_j^π of job j when applying preemptive list scheduling to sequence π is uniquely defined. For some point in time $t \geq 0$ let $W^\pi(t)$ denote the total weight of jobs that are not yet completed by time t according to sequence π, i.e., $W^\pi(t) := \sum_{j:C_j^\pi > t} w_j$. Then,

$$\sum_{j \in J} w_j C_j^\pi = \int_0^\infty W^\pi(t)dt. \tag{1}$$

Clearly, breaks or fluctuations in the speed of the machine delay the completion times. To describe a particular machine behavior, let $f : \mathbb{R}_+ \to \mathbb{R}_+$ be a non-decreasing continuous function, with $f(t)$ being the aggregated amount of processing time available on the machine up to time t. We refer to f as the *machine capacity function*. If the derivative of f at time t exists, it can be interpreted as the speed of the machine at that point in time.

For a given capacity function f, let $S(\pi, f)$ denote the single machine schedule when applying preemptive list scheduling to permutation π, and let $C_j^{S(\pi,f)}$ denote the completion time of job j in this particular schedule. For some point in time $t \geq 0$, let $W^{S(\pi,f)}(t)$ denote the total weight of jobs that are not yet completed by time t in schedule $S(\pi, f)$. Then,

$$\sum_{j \in J} w_j C_j^{S(\pi,f)} = \int_0^\infty W^{S(\pi,f)}(t)dt\,.$$

For $t \geq 0$ let $W^{S^*(f)}(t) := \min_\pi W^{S(\pi,f)}(t)$.

Observation 1. *For a given machine capacity function f,*

$$\int_0^\infty W^{S^*(f)}(t)dt \tag{2}$$

is a lower bound on the objective function of any schedule.

We construct a universal sequence of jobs π such that, no matter how the single machine behaves, the objective value of the corresponding schedule $S(\pi, f)$ is within a constant factor of the optimum.

Lemma 1. *Let π be a sequence of jobs, and let $c > 0$. Then, the value $\sum_{j \in J} w_j C_j^{S(\pi,f)}$ is at most c times the optimum for all machine capacity functions f if and only if*

$$W^{S(\pi,f)}(t) \leq cW^{S^*(f)}(t) \qquad \textit{for all } t \geq 0\textit{, and for each } f.$$

Proof. The "if" part is clear, since by Observation 1

$$\sum_{j \in J} w_j C_j^{S(\pi,f)} = \int_0^\infty W^{S(\pi,f)}(t)dt \leq c \int_0^\infty W^{S^*(f)}(t)dt.$$

We prove the "only if" part by contradiction. Assume that $W^{S(\pi,f)}(t_0) > cW^{S^*(f)}(t_0)$ for some t_0 and f. For any $t_1 > t_0$ consider the following machine capacity function

$$f'(t) = \begin{cases} f(t) & \text{if } t \le t_0, \\ f(t_0) & \text{if } t_0 < t \le t_1, \\ f(t - t_1 + t_0) & \text{if } t > t_1 \end{cases}$$

which equals f up to time t_0 and then remains constant at value $f'(t) = f(t_0)$ for the time interval $[t_0, t_1]$. Hence,

$$\sum_{j \in J} w_j C_j^{S(\pi,f')} = \sum_{j \in J} w_j C_j^{S(\pi,f')} + (t_1 - t_0) W^{S(\pi,f')}(t_0). \tag{3}$$

On the other hand, let π^* be a sequence of jobs with $W^{S(\pi^*,f')}(t_0) = W^{S^*(f')}(t_0)$. Then,

$$\sum_{j \in J} w_j C_j^{S(\pi^*,f')} = \sum_{j \in J} w_j C_j^{S(\pi^*,f')} + (t_1 - t_0) W^{S^*(f')}(t_0). \tag{4}$$

As t_1 tends to infinity, the ratio of (3) and (4) tends to $W^{S(\pi,f')}(t_0)/W^{S^*(f')}(t_0) > c$, a contradiction. □

In case that all release dates are equal, approximating the sum of weighted completion times on a machine with unknown processing behavior is equivalent to approximating the total remaining weight at any point in time on an ideal machine: $f(t) = t,\ t \ge 0$. Scheduling according to sequence π on such a machine yields for each j, $C_j^\pi := \sum_{k:\pi(k) \le \pi(j)} p_k$. The completion time under machine capacity function f is

$$C_j^{S(\pi,f)} = \min\{t \mid f(t) \ge C_j^\pi\}.$$

Observation 2. *For any machine capacity function f and any sequence π of jobs without release dates,*

$$W^{S(\pi,f)}(t) = W^\pi(f(t)) \qquad \textit{for all } t \ge 0.$$

For $f(t) = t$ let $W^*(t) := W^{S^*(f)}(t)$. With Observation 2 we can significantly strengthen the statement of Lemma 1.

Lemma 2. *Let π be a sequence of jobs with equal release dates, and let $c > 0$. Then, the objective value $\sum_{j \in J} w_j C_j^{S(\pi,f)}$ is at most c times the optimum for all machine capacity functions f if and only if*

$$W^\pi(t) \le cW^*(t) \qquad \textit{for all } t \ge 0.$$

Simple counter examples show that this lemma is only true if all release dates are equal, otherwise, Observation 2 is simply not true.

3 Universal Scheduling without Release Dates

3.1 Upper Bounds

In the sequel we use for a subset of jobs $J' \subseteq J$ the notation $p(J') := \sum_{j \in J'} p_j$ and $w(J') := \sum_{j \in J'} w_j$. Based on key Lemma 2, we aim at approximating the minimum total weight of uncompleted jobs at any point in time on an ideal machine, i.e., we approximate the value of $W^*(t)$ for all values of $t \leq p(J)$ for a machine with capacity function $f(t) = t, t \geq 0$. In our algorithm we do so by solving the problem to find the set of jobs that has maximum total processing time and total weight within a given bound. By sequentially doubling the weight bound, a sequence of job sets is obtained. Jobs in job sets corresponding to smaller weight bounds are to come later in the schedule, breaking ties arbitrarily.

Algorithm DOUBLE:

1. For $i \in \{0, 1, \ldots, \lceil \log w(J) \rceil\}$, find a subset J_i^* of jobs of total weight $w(J_i^*) \leq 2^i$ and maximum total processing time $p(J_i^*)$. Notice that $J^*_{\lceil \log w(J) \rceil} = J$.
2. Construct a permutation π as follows. Start with an empty sequence of jobs. For $i = \lceil \log w(J) \rceil$ down to 0, append the jobs in $J_i^* \setminus \bigcup_{k=0}^{i-1} J_k^*$ in any order at the end of the sequence.

Theorem 1. *For every scheduling instance,* DOUBLE *produces a permutation π such that the objective value $\sum_{j \in J} w_j C_j^{S(\pi,f)}$ is less than 4 times the optimum for all machine capacity functions f.*

Proof. Using Lemma 2 it is sufficient to show that $W^\pi(t) < 4W^*(t)$ for all $t \geq 0$. Let $t \geq 0$ and let i be minimal such that $p(J_i^*) \geq p(J) - t$. By construction of π, only jobs j in $\bigcup_{k=0}^{i} J_k^*$ have a completion time $C_j^\pi > t$. Thus,

$$W^\pi(t) \leq \sum_{k=0}^{i} w(J_k^*) \leq \sum_{k=0}^{i} 2^k = 2^{i+1} - 1. \tag{5}$$

In case $i = 0$, the claim is trivially true since $w_j \geq 1$ for any $j \in J$, and thus, $W^*(t) = W^\pi(t)$. Suppose $i \geq 1$, then by our choice of i, it holds that $p(J_{i-1}^*) < p(J) - t$. Therefore, in any sequence π', the total weight of jobs completing after time t is larger than 2^{i-1}, because otherwise we get a contradiction to the maximality of $p(J_{i-1}^*)$. That is, $W^*(t) > 2^{i-1}$. Together with (5) this concludes the proof. □

Notice that the algorithm takes exponential time since finding the subsets of jobs J_i^* is a KNAPSACK problem and, thus, NP-hard [15]. However, we adapt the algorithm by, instead of J_i^*, computing a subset of jobs J_i of total weight $w(J_i) \leq (1 + \epsilon/4)2^i$ and processing time $p(J_i) \geq \max\{p(J') \mid J' \subseteq J \text{ and } w(J') \leq 2^i\}$. This can be done in time polynomial in the input size and $1/\epsilon$ adapting, e.g., the FPTAS in [11] for KNAPSACK. The subsets J_i obtained in this way are turned into a sequence π' as in DOUBLE.

Theorem 2. *Let $\epsilon > 0$. For every scheduling instance, we can construct a permutation π in time polynomial in the input size and $1/\epsilon$ such that the value $\sum_{j\in J} w_j C_j^{S(\pi,f)}$ is less than $4+\epsilon$ times the optimum for all machine capacity functions f.*

Proof. Again, by Lemma 2 it is sufficient to prove that $W^{\pi}(t) < 4W^*(t)$ for all $t \geq 0$. Instead of inequality (5) we get the slightly weaker bound

$$W^{\pi'}(t) \leq \sum_{k=0}^{i} w(J_k) \leq \sum_{k=0}^{i} (1+\epsilon/4)2^k = (1+\epsilon/4)(2^{i+1}-1) < (4+\epsilon)\,2^{i-1}.$$

Moreover, the lower bound $W^*(t) > 2^{i-1}$ still holds. □

We improve Theorem 1 by adding randomization to DOUBLE in a quite standard fashion. Instead of the fixed bound of 2^i on the total weight of job set J_i^* in iteration $i \in \{0, 1, \ldots, \lceil \log w(J) \rceil\}$ we use the randomly chosen bound Xe^i where $X = e^Y$ and Y is picked uniformly at random from $[0, 1]$ before the first iteration. We omit the proof.

Theorem 3. *Let $\epsilon > 0$. For every scheduling instance, randomized* DOUBLE *constructs a permutation π in time that is polynomial in the input size and $1/\epsilon$ such that the objective value $\sum_{j\in J} w_j C_j^{S(\pi,f)}$ is in expectation less than $e+\epsilon$ times the optimum value for all machine capacity functions f.*

A natural generalization of the universal sequencing problem requires that jobs are sequenced in compliance with given precedence constraints. We extend the results in Theorems 1 and 3 to this model for certain classes of precedence constraints such as directed out-trees, two dimensional orders, and the complement of chordal bipartite orders.

3.2 Lower Bounds

In this section we show a connection between the performance guarantee for sequencing jobs on a single machine without release dates and an online bidding problem investigated by Chrobak et al. [5]. This allows us to prove tight lower bounds for our problem.

In online bidding, we are given a universe $\mathcal{U} = \{1, \ldots, n\}$. A bid set is just a subset of $\mathcal{U}$. A given bid set $\mathcal{B}$ is said to be α-competitive if

$$\sum_{b\in\mathcal{B}\,:\,b<T} b \;+\; \min_{b\in\mathcal{B}\,:\,b\geq T} b \;\leq \alpha\, T \qquad \forall\, T \in \mathcal{U}. \tag{6}$$

Chrobak et al. [5] gave lower bounds of $4-\epsilon$ and $e-\epsilon$, for any $\epsilon > 0$, for deterministic and randomized algorithms, respectively.

Theorem 4. *For any $\epsilon > 0$, there exists an instance of the universal scheduling problem without release dates on which the performance ratio of any deterministic schedule is at least $4-\epsilon$ and the performance ratio of any randomized schedule is at least $e-\epsilon$.*

Proof. Take an instance of the online bidding problem and create the following instance of the scheduling problem: For each $j \in \mathcal{U}$ create job j with weight $w_j = j$ and processing time $p_j = j^j$. Consider any permutation π of the jobs $\mathcal{U}$. For any $j \in \mathcal{U}$, let $k(j)$ be the largest index such that $\pi_{k(j)} \geq j$. Since $p_j > \sum_{i=1}^{j-1} p_j$, at time $t = p(\mathcal{U}) - p_j$ we have $W^{\pi}(t) = \sum_{k=k(j)}^{n} w_{\pi_k}$, while $W^*(t) = w_j$. If sequence π yields a performance ratio of α then, Lemma 2 tell us that

$$\sum_{k=k(j)}^{n} \pi_k \leq \alpha\, j \qquad \forall\, j \in \mathcal{U}. \tag{7}$$

From sequence π we extract another sequence of jobs as follows: $\mathcal{W}_1 = \pi_n$, $\mathcal{W}_k = \operatorname{argmax}_{i \in \mathcal{U}} \left\{ \pi^{-1}(i) \mid i > \mathcal{W}_{k-1} \right\}$. Then $\mathcal{W}_{i+1} > \mathcal{W}_i$, and all j with $\pi^{-1}(\mathcal{W}_{i+1}) < \pi^{-1}(j) < \pi^{-1}(\mathcal{W}_i)$ have weight less than $\mathcal{W}_i$. Therefore, we have $\{i \in \mathcal{W} \mid i < j\} \cup \min\{i \in \mathcal{W} \mid i \geq j\} \subset \{\pi_{k(j)}, \ldots, \pi_n\}$, for all $j \in \mathcal{U}$. Hence, if π achieves a performance ratio of α then

$$\sum_{i \in \mathcal{W}\,:\, i<j} i \;+\; \min_{i \in \mathcal{W}\,:\, i \geq j} i \;\leq\; \sum_{k=k(j)}^{n} \pi_k \;\leq\; \alpha\, j \qquad \forall\, j \in \mathcal{U},$$

that is, the bid set $\mathcal{W}$ induced by the sequence π must be α-competitive. Since there is a lower bound of $4 - \epsilon$ for the competitiveness of deterministic strategies for online bidding, the same bound holds for the performance ratio of deterministic universal schedules.

The same approach yields the lower bound for randomized strategies. □

4 Universal Scheduling with Release Dates

In this section we study the problem of the previous section when jobs have release dates. Algorithm DOUBLE, which aims at minimizing the total remaining weight, can be adapted to the setting with release dates: Instead of a knapsack algorithm we use, within a binary search routine, Lawler's pseudo-polynomial time algorithm [18] or the FPTAS by Pruhs and Woeginger [25] for preemptively scheduling jobs with release dates and due dates on a single machine to minimize the total weight of late jobs.

However, this approach does not yield a bounded performance guarantee for the universal scheduling problem. In the presence of release dates approximation ratios on an ideal machine do not translate directly to a performance guarantee of the universal scheduling strategy, see Section 2. In fact, universal scheduling with release dates cannot be approximated within a constant ratio as we show below.

4.1 Lower Bound

Theorem 5. *There exists an instance with n jobs with equal weights and release dates, where any universal schedule has a performance guarantee of $\Omega(\log n / \log\log n)$.*

In our lower bound instance each job j has $w_j = 1, j = 0, 1, \ldots, n-1$. Their processing times form a geometric series $p_j = 2^j$, $j = 0, 1, \ldots, n-1$, and they are released in reversed order $r_j = \sum_{i>j}^{n} 2^i = \sum_{i>j} p_i$, $j = 0, 1, \ldots, n-1$.

To show the bound, we rely on a classic theorem of Erdős and Szekeres [8] or, more precisely, on Hammersley's proof [10] of this result.

Lemma 3 (Hammersley [10]). *Given a sequence of n distinct numbers $x_1, x_2, \ldots, x_n$, we can decompose this set into k increasing subsequences $\ell_1, \ell_2, \ldots, \ell_k$ such that:*

- *There is a decreasing subsequence of length k;*
- *If x_i belongs to ℓ_a then for all $j > i$ if $x_j < x_i$ then x_j belongs to ℓ_b and $b > a$.*

The idea is now to view a universal schedule as a permutation of $\{0, 1, \ldots, n-1\}$ and use Lemma 3 to decompose the sequence into k increasing subsequences. This decomposition is then used to design a breakdown pattern that will yield Theorem 5. The next two lemmas outline two kinds of breakdown patterns that apply to the two possibilities offered by Lemma 3.

Lemma 4. *The performance guarantee of a universal schedule that has ℓ as a decreasing subsequence is at least $|\ell|$.*

Proof. Let j be the first job in ℓ. The machine has breakdowns $[r_j, r_0]$ and $[r_0 + 2^j - 1, L]$ for large L. At time r_0 all jobs have been released. $2^j - 1$ time units later, at the start of the second breakdown, all jobs in ℓ belong to the set of jobs uncompleted by the universal schedule, whereas an optimal solution can complete all jobs except j. Choosing L large enough implies the lemma. □

Lemma 5. *Let $\ell_1, \ell_2, \ldots, \ell_k$ be the decomposition described in Lemma 3 when applied to a universal schedule. Then for all $i = 1, \ldots, k$ the performance guarantee is at least $\frac{|\ell_i|+|\ell_{i-1}|+\cdots+|\ell_1|}{1+|\ell_{i-1}|+\cdots+|\ell_1|}$*

Proof. For each job j in ℓ_i there is a breakdown $[r_j, r_j+\epsilon]$. For each job j in $\ell_{i+1}, \ldots, \ell_k$ there is a breakdown $[r_j, r_j + p_j] = [r_j, r_j + 2^j]$. As a consequence, at time $2^n - 1$ the universal schedule has all jobs in ℓ_i and all jobs in $\ell_{i+1}, \ldots, \ell_k$ uncompleted, whereas, a schedule exists that leaves the last job of ℓ_i and all jobs in $\ell_{j+1}, \ldots, \ell_k$ uncompleted. Therefore, a breakdown $[2^n - 1, L]$ for L large enough implies the lemma. □

Proof (Proof of Theorem 5). Consider an arbitrary universal scheduling solution and its decomposition into increasing subsequences $\ell_1, \ldots, \ell_k$ as in Lemma 3 and let α be its performance guarantee. Using Lemma 5, one can easily prove by induction that $|\ell_i| \leq \alpha^{k-i+1}$. Since $\ell_1, \ldots, \ell_k$ is a partition of the jobs, we have

$$n = \sum_{i=1}^{k} |\ell_i| \leq \sum_{i=1}^{k} \alpha^{k-i+1} \leq \alpha^{k+1}.$$

By Lemma 4, it follows that $k \leq \alpha$. Therefore $\log n = O(\alpha \log \alpha)$ and $\alpha = \Omega\left(\frac{\log n}{\log \log n}\right)$.

□

4.2 Jobs with Proportional Weights

Motivated by the negative result in the previous section, we turn our attention to the special case with proportional weights, that is, there exists a fixed $\gamma \in \mathbb{Q}$ such that $w_j = \gamma p_j$, for all $j \in J$. Using a standard scaling argument we can assume w.l.o.g. that $p_j = w_j$, for all j. We provide an algorithm with a performance guarantee 5, and prove a lower bound of 3 on the performance guarantee of any universal scheduling algorithm.

Algorithm SORTCLASS:

1. Partition the set of jobs into $z := \lceil \log \max_{j \in J} w_j \rceil$ classes, such that j belongs to class J_i, for $i \in 1, 2, \ldots, z$, if and only if $p_j \in (2^{i-1}, 2^i]$.
2. Construct a permutation π as follows. Start with an empty sequence of jobs. For $i = z$ down to 1, append the jobs of class J_i in non-decreasing order of release dates at the end of π.

Theorem 6. *The performance guarantee of* SORTCLASS *for universal scheduling of jobs with proportional weights and release dates is exactly* 5.

Proof. Let π be the job sequence computed by SORTCLASS. By Lemma 1, it is sufficient to prove

$$W^{S(\pi,f)}(t) \;\leq\; 5W^{S^*(f)}(t) \quad \forall t > 0. \tag{8}$$

Take any time t and any machine capacity function f. Let $j \in J_i$ be the job being processed at time t according to the schedule $S(\pi, f)$. We say that a job other than job j is *in the stack* at time t if it was processed for a positive amount of time before t. The algorithm needs to complete all jobs in the stack, job j, and jobs that did not start before t, which have a total weight of at most $p(J) - f(t)$, the amount of remaining processing time at time t to be done by the algorithm.

Since jobs within a class are ordered by release times, there is at most one job per class in the stack at any point in time. Since jobs in higher classes have higher priority and job $j \in J_i$ is processed at time t, there are no jobs in $J_{i+1}, \ldots, J_z$ in the stack at time t. Thus the weight of the jobs in the stack together with the weight of job j is at most $\sum_{k=1}^{i} 2^k = 2^{i+1} - 1$. Hence,

$$W^{S(\pi,f)}(t) \;< 2^{i+1} + p(J) - f(t)\,. \tag{9}$$

A first obvious lower bound on the remaining weight of any schedule at time t is

$$W^{S^*(f)}(t) \;\geq p(J) - f(t)\,. \tag{10}$$

For another lower bound, let t' be the last time before t in which the machine is available but it is either idle or a job of a class $J_{i'}$ with $i' < i$ is being processed. Note that t' is well-defined. By definition, all jobs processed during the time interval $[t', t]$ are in classes with index at least i, but also, they are released in the interval $[t', t]$ since at t' a job of a lower class was processed or the machine was idle. Since at time t at least one of these jobs is unfinished in $S(\pi, f)$, even though the machine continuously processed

only those jobs, no algorithm can complete all these jobs. Thus, at time t, an optimal schedule also still needs to complete at least one job with weight at least 2^{i-1}:

$$W^{S^*(f)}(t) \geq 2^{i-1}. \tag{11}$$

Combining (9), (10), and (11) yields (8) and thus the upper bound of the theorem.

We omit the example that shows that the analysis is tight. □

We complement this result by a lower bound of 3, but have to omit the proof.

Theorem 7. *There is no algorithm with performance guarantee strictly smaller than* 3 *for universal scheduling of jobs with release dates and* $w_j = p_j$*, for all* $j \in J$.

5 The Offline Problem

Clearly, the performance guarantees derived in Sections 3 and 4 also hold in the offline version of our problem in which machine breakdowns and changes in speed are known in advance. Additionally, we investigate in this section the special case in which the machine has a single, a priori-known non-availability interval $[s, t]$, for $1 \leq s < t$.

Theorem 8. *There exists an FPTAS with running time* $\mathcal{O}(n^3/\epsilon^2)$ *for non-preemptive scheduling to minimize* $\sum w_j C_j$ *on a single machine that is not available for processing during a given time interval* $[s, t]$*. The approximation scheme can be extended to the preemptive (resumable) setting with an increased running time of* $\mathcal{O}(n^4/\epsilon^2 \log p_{\max})$.

Due to space limitations we defer all details to the full version of the paper. The idea for our FPTAS is based on a natural non-preemptive dynamic programming algorithm, used also in [16]. Given a non-available time interval $[s, t]$, the job set must be partitioned into jobs that complete before s and jobs that complete after t. Clearly, the jobs in each individual set are scheduled in non-increasing order of ratios w_j/p_j. This order is known to be optimal on an ideal machine [29].

The main challenge in designing the FPTAS is to discretize the range of possible total processing times of jobs scheduled before s in an appropriate way. Notice that we cannot afford to round these values since they contain critical information on how much processing time remains before the break. Perturbing this information causes a considerable change in the set of feasible schedules that cannot be controlled easily. The intuition behind our algorithm is to reduce the number of states by removing those with the same (rounded) objective value and nearly the same total processing time before the break. Among them, we want to store those with smallest amount of processing before the break in order to make sure that enough space remains for further jobs that need to be scheduled there.

The algorithm can be extended easily to the preemptive (resumable) problem. We can assume, w.l.o.g., that in an optimal solution there is at most one job j interrupted by the break $[s, t]$ and it resumes processing as soon as the machine is available again. For a given job j and with start time S_j, we define a non-preemptive problem with non-available period $[S_j, S_j + p_j + t - s]$, which we solve by the FPTAS above. Thus, we can solve the preemptive problem by running the FPTAS above $\mathcal{O}(n \log p_{\max})$ times.

6 Further Remarks

In Section 4 we have shown that the performance of universal scheduling algorithms may deteriorate drastically when generalizing the universal scheduling problem slightly. Other generalizations do not admit any (exponential time) algorithm with bounded performance guarantee. If a non-adaptive algorithm cannot guarantee to finish within the minimum makespan, then an adversary creates an arbitrarily long breakdown at the moment that an optimal schedule has completed all jobs. Examples of such variations are the problem with two or more machines instead of a single machine, or the problem in which preempting or resuming a job requires (even the slightest amount of) extra work.

The offline version of our problem (without release dates) in which preemption is not allowed or causes extra work is not approximable in polynomial time; a reduction from 2-PARTITION shows that the problem with two or more non-available periods is not approximable, unless P=NP, even if all jobs have equal weight. A reduction in that spirit has been used in [33] for a scheduling problem with some jobs having a fixed position in the schedule. Similarly, we can rule out constant approximation factors for any preemptive problem variant in which the makespan cannot be computed exactly in polynomial time. This is shown by simple reductions from the corresponding decision version of the makespan minimization problem. Such variations of our problem are scheduling with release dates and scheduling with general precedence constraints.

References

1. Adiri, I., Bruno, J., Frostig, E., Rinnooy Kan, A.: Single machine flow-time scheduling with a single breakdown. Acta Informatica 26(7), 679–696 (1989)
2. Becchetti, L., Leonardi, S., Marchetti-Spaccamela, A., Pruhs, K.: Online weighted flow time and deadline scheduling. In: Goemans, M.X., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) RANDOM 2001 and APPROX 2001. LNCS, vol. 2129, pp. 36–47. Springer, Heidelberg (2001)
3. Ben-Tal, A., Nemirovski, A.: Robust solutions of linear programming problems contaminated with uncertain data. Mathematical Programming 88, 411–424 (2000)
4. Carter, J., Wegman, M.: Universal classes of hash functions. Journal of Computer and System Sciences 18, 143–154 (1979)
5. Chrobak, M., Kenyon, C., Noga, J., Young, N.E.: Incremental medians via online bidding. Algorithmica 50(4), 455–478 (2008)
6. Chrobak, M., Kenyon-Mathieu, C.: Sigact news online algorithms column 10: Competitiveness via doubling. SIGACT News 37(4), 115–126 (2006)
7. Diedrich, F., Jansen, K., Schwarz, U.M., Trystram, D.: A survey on approximation algorithms for scheduling with machine unavailability. In: Lerner, J., Wagner, D., Zweig, K.A. (eds.) Algorithmics of Large and Complex Networks. LNCS, vol. 5515, pp. 50–64. Springer, Heidelberg (2009)
8. Erdős, P., Szekeres, G.: A combinatorial problem in geometry. Compositio Mathematica 2, 463–470 (1935)
9. Hall, L., Schulz, A.S., Shmoys, D., Wein, J.: Scheduling to minimize average completion time: off-line and on-line approximation algorithms. Mathematics of Operations Research 22, 513–544 (1997)
10. Hammersley, J.: A few seedlings of research. In: Proceedings Sixth Berkeley Symp. Math. Statist. and Probability, vol. 1, pp. 345–394. University of California Press, Berkeley (1972)
11. Ibarra, O.H., Kim, C.E.: Fast approximation algorithms for the knapsack and sum of subset problems. Journal of the ACM 22(4), 463–468 (1975)

12. Jia, L., Lin, G., Noubir, G., Rajaraman, R., Sundaram, R.: Universal approximations for TSP, Steiner tree, and set cover. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC '05), pp. 386–395 (2005)
13. Kacem, I.: Approximation algorithm for the weighted flow-time minimization on a single machine with a fixed non-availability interval. Computers & Industrial Engineering 54(3), 401–410 (2008)
14. Kacem, I., Mahjoub, A.R.: Fully polynomial time approximation scheme for the weighted flow-time minimization on a single machine with a fixed non-availability interval. Computers & Industrial Engineering 56(4), 1708–1712 (2009)
15. Karp, R.M.: Reducibility among combinatorial problems. In: Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center), pp. 85–103. Plenum, New York (1972)
16. Kellerer, H., Strusevich, V.A.: Fully polynomial approximation schemes for a symmetric quadratic knapsack problem and its scheduling applications (2009) (accepted for publication in Algorithmica)
17. Kouvelis, P., Yu, G.: Robust Discrete Optimization and Its Applications. Springer, Heidelberg (1997)
18. Lawler, E.L.: A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. Annals of Operations Research 26, 125–133 (1990)
19. Lee, C.Y.: Machine scheduling with an availability constraint. Journal of Global Optimimization 9, 395–416 (1996)
20. Lee, C.Y.: Machine scheduling with availability constraints. In: Leung, J.Y.T. (ed.) Handbook of scheduling. CRC Press, Boca Raton (2004)
21. Lee, C.Y., Liman, S.D.: Single machine flow-time scheduling with scheduled maintenance. Acta Informatica 29(4), 375–382 (1992)
22. Liebchen, C., Lübbecke, M., Möhring, R.H., Stiller, S.: Recoverable robustness. Technical report ARRIVAL-TR-0066, ARRIVAL Project (2007)
23. Mastrolilli, M., Mutsanas, N., Svensson, O.: Approximating single machine scheduling with scenarios. In: Goel, A., Jansen, K., Rolim, J.D.P., Rubinfeld, R. (eds.) APPROX and RANDOM 2008. LNCS, vol. 5171, pp. 153–164. Springer, Heidelberg (2008)
24. Megow, N., Verschae, J.: Note on scheduling on a single machine with one non-availability period (2008) (unpublished)
25. Pruhs, K., Woeginger, G.J.: Approximation schemes for a class of subset selection problems. Theoretical Computer Science 382(2), 151–156 (2007)
26. Räcke, H.: Survey on oblivious routing strategies. In: Ambos-Spies, K., Löwe, B., Merkle, W. (eds.) Mathematical Theory and Computational Practice, Proceedings of 5th Conference on Computability in Europe (CiE). LNCS, vol. 5635, pp. 419–429. Springer, Heidelberg (2009)
27. Schmidt, G.: Scheduling with limited machine availability. European Journal of Operational Research 121(1), 1–15 (2000)
28. Schulz, A.S., Skutella, M.: The power of α-points in preemptive single machine scheduling. Journal of Scheduling 5, 121–133 (2002)
29. Smith, W.E.: Various optimizers for single-stage production. Naval Research Logistics Quarterly 3, 59–66 (1956)
30. Soyster, A.: Convex programming with set-inclusive constraints and applications to inexact linear programming. Operations Research 21(4), 1154–1157 (1973)
31. Valiant, L.G., Brebner, G.J.: Universal schemes for parallel communication. In: Proc. of STOC, pp. 263–277 (1981)
32. Wang, G., Sun, H., Chu, C.: Preemptive scheduling with availability constraints to minimize total weighted completion times. Annals of Operations Research 133, 183–192 (2005)
33. Yuan, J., Lin, Y., Ng, C., Cheng, T.: Approximability of single machine scheduling with fixed jobs to minimize total completion time. European Journal of Operational Research 178(1), 46–56 (2007)