The Power of α–Points in Preemptive Single Machine Scheduling

ANDREAS S. SCHULZ
Massachusetts Institute of Technology
Sloan School of Management
and Operations Research Center
E53-361, Cambridge, MA 02142, USA
E-mail: schulz@mit.edu

MARTIN SKUTELLA
Technische Universität Berlin
Fachbereich Mathematik, MA 6-1
Straße des 17. Juni 136
D-10623 Berlin, Germany
E-mail: skutella@math.tu-berlin.de

March 1999; revised September 2000

Abstract

We consider the NP-hard preemptive single machine scheduling problem to minimize the total weighted completion time subject to release dates. A natural extension of Smith's ratio rule is to preempt the currently active job whenever a new job arrives that has higher ratio of weight to processing time. We prove that the competitive ratio of this simple on-line algorithm is precisely 2. We also show that list scheduling in order of random α–points drawn from the same schedule results in an on-line algorithm with competitive ratio $4/3$. Since its analysis relies on a well-known integer programming relaxation of the scheduling problem, the relaxation has performance guarantee $4/3$ as well. On the other hand, we show that it is at best an $8/7$–relaxation.

KEY WORDS: scheduling theory; approximation algorithm; on-line algorithm; randomized algorithm; LP relaxation; combinatorial optimization

1 Introduction

We study the preemptive single-machine on-line scheduling problem with release dates so as to minimize the average weighted completion time. A set of independent jobs $J = \{1,\dots,n\}$ has to be scheduled on a single machine, where jobs arrive over time and the number of jobs is unknown in advance. Each job $j \in J$ becomes available at its integral release date r_j, which is not known in advance; at time r_j we learn both its positive integral processing time p_j and its nonnegative weight w_j. The machine cannot process more than one job at a time, and each job j has to be scheduled for p_j time units on the machine. The processing of a job may repeatedly be interrupted and continued at a later point in time, i. e., we sequence in a preemptive fashion. For each time t, we must construct the schedule until time t without any knowledge of the jobs that will arrive afterwards. The aim is to minimize the sum of weighted job completion times $\sum_j w_j C_j$ or, equivalently, the average weighted completion time $\frac{1}{n}\sum_j w_j C_j$; here, C_j denotes the completion time of job j in a schedule. The corresponding off-line optimization problem, where all job data is known in advance, is usually denoted by $1|\,r_j, pmtn\,|\sum w_j C_j$ [11]. It is strongly NP-hard [16].

The performance of an on-line algorithm is typically measured by its *competitive ratio*, which is the largest ratio of the objective function value achieved by the on-line algorithm to the value of the off-line optimum, taken over all instances. For randomized algorithms, we use expected objective function value in the definition of the competitive ratio. This corresponds to the so-called oblivious adversary model. We refer the reader to [3] for a general reference on on-line algorithms, and to [19] for a survey of on-line scheduling. Because we will discuss at times the off-line scenario as well, let us introduce the related concept of an approximation algorithm. A ρ–*approximation algorithm* is a polynomial-time algorithm that produces a solution of value not worse than ρ times the optimal value. In case the algorithm has access to randomness, we call it a randomized ρ–approximation algorithm, provided that it still runs in polynomial time; the performance guarantee ρ, however, only needs to hold in expectation. Obviously, any on-line algorithm that runs in polynomial time and has competitive ratio ρ is a ρ–approximation algorithm.

The main result of the paper is a randomized on-line algorithm for $1|\,r_j,\,pmtn\,|\sum w_jC_j$ with competitive ratio $4/3$ and running time $O(n\log n)$. For the off-line setting, it can be derandomized without loss of performance guarantee, but at the cost of an increased running time of $O(n^2)$. Our result also implies a bound of $4/3$ on the quality of a well-known linear programming relaxation (which happens to be integer) of the problem under consideration. Moreover, we present a class of instances showing that the ratio between the true optimum and the LP lower bound can be arbitrarily close to $8/7$.

The three key ingredients of the algorithm and its analysis are the conversion of a preemptive schedule to another preemptive schedule, the use of α–points in connection with randomness to sample more information from the given preemptive schedule, and the exploitation of a linear programming relaxation as a lower bound on the optimal objective function value. All three techniques have in recent years evolved as important tools to derive constant-factor approximation algorithms for a series of scheduling problems with min-sum objective.

The conversion of preemptive schedules to (nonpreemptive) schedules was introduced by Phillips, Stein and Wein [17] and was subsequently also used in [4, 5, 8], among others. Slightly varying notions of α–points were considered in [17, 12], but their full potential was revealed when Chekuri, Motwani, Natarajan, and Stein [5] as well as Goemans [8] chose the parameter α at random. For $0 < \alpha \leqslant 1$, the α–*point* $C_j^P(\alpha)$ of job j with respect to a given (preemptive) schedule P is the first point in time at which an α–fraction of job j has been completed, i. e., when j has been processed on the machine for $\alpha \cdot p_j$ time units. In particular, $C_j^P(1) = C_j$ and for $\alpha = 0$ we define $C_j^P(0)$ to be the starting time of job j. Later, α–points with individual values of α for different jobs have been used, see [9]. We refer to [20, Chapter 2] for a detailed account of approximation algorithms for min-sum criteria scheduling and α–point scheduling.

The actual history of constant-factor approximation algorithms for $1|\,r_j,\,pmtn\,|\sum w_jC_j$ is rather short. Phillips, Stein and Wein [17] designed an $(8+\varepsilon)$–approximation algorithm for the more general problem of minimizing the average weighted completion time on unrelated parallel machines subject to release dates. Hall, Schulz, Shmoys, and Wein [13] gave a 2–approximation algorithm for the single-machine problem, in which there may also be precedence constraints among the jobs. Their algorithm is based on a related LP relaxation, rather than on the preemptive schedule that is an optimal solution of the integer programming relaxation that we employ in our analysis. In fact, it was Goemans [8] who first showed that one can use this preemptive schedule to construct a nonpreemptive schedule whose value is at most twice the optimum value of the integer programming relaxation. In particular, Goemans' algorithm is a 2–approximation algorithm for $1|\,r_j,\,pmtn\,|\sum w_jC_j$ as well. Goemans, Wein and Williamson [10] then presented a randomized 1.466–approximation algorithm which also works in the on-line setting, as does the randomized variant of Goemans' algorithm; our work may be seen as a simpler analysis of their algorithm that at the same time yields a better performance guarantee. Subsequently, a polynomial-time approximation scheme has been obtained for the off-line problem $1|\,r_j,\,pmtn\,|\sum w_jC_j$, see [1].

The rest of the paper is organized as follows. In Section 2 we embed the algorithm under consideration in the general class of preemptive list scheduling algorithms. Its actual analysis is given in Section 3. We conclude with some remarks on its derandomization and a discussion of open problems in Section 4.

2 Preemptive List Scheduling

There is a straightforward way to construct a feasible preemptive schedule from a given list of jobs representing some order: schedule at any point in time the first available job in this list. Here, a job is available if its release time has elapsed. We refer to this routine as *preemptive list scheduling*; the resulting schedule is called *preemptive list schedule*. Preemptive list scheduling has been used in various settings before, e.g., for minimizing the maximum lateness on a single machine [15]. An application of this routine in the context of min-sum criteria approximation has been proposed by Hall, Schulz, Shmoys, and Wein [13] in order to turn an optimum solution to an LP relaxation in completion time variables into a feasible preemptive schedule. Goemans [8] showed that the routine can be used to construct an optimum solution to an LP relaxation in time-indexed variables; he also pointed out that preemptive list schedules can be constructed in $O(n \log n)$ time using a priority queue.

As a consequence of the following lemma, one can in fact restrict to schedules that are generated by preemptive list scheduling.

Lemma 2.1. *Given a feasible preemptive schedule P, preemptive list scheduling in order of nondecreasing completion times does not increase completion times of jobs.*

Proof. Although this lemma belongs to the folklore of the field, let us provide a proof for the sake of completeness.

We denote the completion time of a job j in the given schedule by C_j^P and in the preemptive list schedule by C_j. By construction, the new schedule is feasible since no job is processed before its release date. For a fixed job j, let $t \geqslant 0$ be the earliest point in time such that there is no idle time in the preemptive list schedule during $(t, C_j]$ and only jobs k with $C_k^P \leqslant C_j^P$ are processed. We denote the set of these jobs by K. By the definition of t, we know that $r_k \geqslant t$, for all $k \in K$. Hence, $C_j^P \geqslant t + \sum_{k \in K} p_k$. On the other hand, the definition of K implies $C_j = t + \sum_{k \in K} p_k$ and therefore $C_j \leqslant C_j^P$. □

An important property of preemptive list schedules is that whenever a job is preempted from the machine, it is only continued after all available jobs with higher priority are finished. Moreover, a job is only preempted if another job is released at that time. Therefore, since all the release dates are integral, preemptions only occur at integral points in time. Throughout the paper we restrict to schedules meeting this property. Notice also that there are at most $n-1$ preemptions.

In the absence of nontrivial release dates there is no need for preemption and an optimal schedule can be constructed in $O(n \log n)$ time using Smith's ratio rule [21]: schedule the jobs in order of nonincreasing ratios w_j/p_j. In the following, we will always assume that jobs are numbered such that $w_1/p_1 \geqslant \cdots \geqslant w_n/p_n$; moreover, whenever we talk about scheduling in order of nonincreasing ratios w_j/p_j, we refer to this order of jobs. A natural generalization of Smith's Ratio Rule to $1|\, r_j, pmtn \,|\sum w_j C_j$ is preemptive list scheduling in order of nonincreasing ratios w_j/p_j; notice that this algorithm also works on-line since at any point in time the ratios of all available jobs are known. Of course, the schedule constructed in this way is in general not optimal. The following lemma gives a lower bound on the performance of this simple heuristic.

Lemma 2.2. *The competitive ratio of preemptive list scheduling in order of nonincreasing ratios w_j/p_j is not better than 2, even if $w_j = 1$ for all $j \in J$.*

Proof. For an arbitrary $n \in \mathbb{N}$, consider the following instance with n jobs. Let $w_j = 1$, $p_j = n^2 - n + j$, and $r_j = -n + j + \sum_{k=j+1}^{n} p_k$, for $1 \leqslant j \leqslant n$. Preemptive list scheduling in order of nonincreasing ratios of w_j/p_j preempts job j at time r_{j-1} and finishes it only after all other jobs $j-1, \ldots, 1$ have been completed. The value of this schedule is therefore $n^4 - \frac{1}{2}n^3 + \frac{1}{2}n$. The shortest remaining processing time rule [2], which solves instances of $1|\, r_j, pmtn \,|\sum C_j$ optimally and does so on-line, sequences the jobs in order $n, \ldots, 1$. It has value $\frac{1}{2}n^4 + \frac{1}{3}n^3 + \frac{1}{6}n$. Consequently, the ratio of the objective function values of the "SPT-rule" and the "SRPT-rule" goes to 2 when n goes to infinity. □

Notice that the negative result in Lemma 2.2 does not result from the on-line nature of the problem; it follows from the proof that this is rather an inherent drawback of preemptive list scheduling in order of nonincreasing ratios w_j/p_j. On the other hand, one can give an upper bound of 2 on the performance of this algorithm. The following observation is due to Goemans, Wein and Williamson (personal communication, August 1997).

Lemma 2.3. *Preemptive list scheduling in order of nonincreasing ratios w_j/p_j has competitive ratio* 2.

Proof. We use two different lower bounds on the value Z^* of an optimal solution in order to prove the claim. Since the completion time of a job is always at least as large as its release date, we get $Z^* \geqslant \sum_j w_j r_j$. The second lower bound is the value of an optimal solution for the relaxed problem in which all the release dates are zero. This yields $Z^* \geqslant \sum_j \big(w_j \cdot \sum_{k \leqslant j} p_k\big)$ by Smith's ratio rule. Let C_j denote the completion time of job j in the preemptive list schedule. By construction one has $C_j \leqslant r_j + \sum_{k \leqslant j} p_k$ and thus

$$\sum_j w_j C_j \;\leqslant\; \sum_j w_j r_j + \sum_j \Big(w_j \cdot \sum_{k \leqslant j} p_k\Big) \;\leqslant\; 2 \cdot Z^* \;\;.$$

□

In spite of the negative result in Lemma 2.2, preemptive list scheduling in order of nonincreasing ratios w_j/p_j can help to construct a preemptive schedule whose value is at most a factor $4/3$ away from the optimum. The idea is to transform this schedule by preemptive list scheduling in order of nondecreasing α–points. The underlying intuition is that the given schedule gives rise to different job orders if different values of α are used. There is, however, no instance that is simultaneously bad for the preemptive list schedules obtained from all different values of α. Consider for instance the example constructed in the proof of Lemma 2.2. For most values of α, the corresponding preemptive list schedule is optimal. We will analyze the following simple algorithm:

Algorithm 1

1) Draw α randomly from $[0,1]$.
2) Construct the preemptive list schedule P in order of nonincreasing ratios w_j/p_j.
3) Apply preemptive list scheduling in order of nondecreasing $C_j^P(\alpha)$.

The on-line variant of Algorithm 1 constructs the two preemptive list schedules from Step 2 and Step 3 simultaneously. Notice that preemptive list scheduling can be implemented on-line if a job can be inserted at the correct position in the list with respect to the jobs that are already known, as soon as it becomes available. As already mentioned, preemptive list scheduling in order of nonincreasing ratios w_j/p_j works on-line since at any point in time the ratios of all available jobs are known. Unfortunately, this is not true for the α–points of jobs because the future development of the schedule in Step 2 is not known. However, at any point in time and for an arbitrary pair j, k of already available jobs we can predict whether $C_j^P(\alpha)$ will be smaller than $C_k^P(\alpha)$, or not. If one or even both values are already known, we are done. Otherwise the job with higher priority in the ratio list of Step 2, say j, will win since job k cannot be (re)started in Step 2 before j is finished. Thus, we have proved that Algorithm 1 can be implemented as a randomized on-line algorithm. Since preemptive list scheduling can be done in $O(n \log n)$ time, the running time of Algorithm 1 and its on-line variant is $O(n \log n)$, too. For fixed α, we call the schedule computed in Step 3 *preemptive α–schedule*.

Goemans analyzed in [8] a variant of Algorithm 1 in which the jobs are scheduled nonpreemptively in order of nondecreasing α–points, where α is chosen uniformly at random. Because the value of the resulting schedule is an upper bound on the value of the schedule computed in Step 3, it follows that Algorithm 1 has competitive ratio 2 in this case.

Goemans, Wein and Williamson [10] showed that Algorithm 1 achieves competitive ratio 1.466 if α is chosen from the interval $[0,\beta]$ according to a probability distribution with density function $f(\alpha) = \frac{1-\beta}{\beta}\cdot(1-\alpha)^{-2}$, where $\beta \approx 0.682$.

The following theorem contains the main result of this paper.

Theorem 2.4. *Let the random variable* α *be chosen from a probability distribution over* $[0,1]$ *with the density function*

$$f(\alpha) = \begin{cases} \frac{1}{3}\cdot(1-\alpha)^{-2} & \text{if } \alpha \in [0,\frac{1}{2}], \\ \frac{4}{3} & \text{otherwise.} \end{cases}$$

Then, Algorithm 1 *has competitive ratio* $4/3$.

The proof of Theorem 2.4 is presented in the next section. Besides the better performance ratio, its major advantage compared to the analysis in [10] is its simplicity. In particular, in contrast to [10], our analysis is job-by-job, i. e., we compare the expected completion time of each job with its completion time in the integer programming relaxation.

3 Analysis of the Algorithm

The analysis of Algorithm 1 is divided into three parts. First, we discuss an integer linear programming relaxation which gives a lower bound on the value of an optimal schedule. Then we derive a general upper bound on the completion time of an arbitrary job in the schedule computed by Algorithm 1, which depends on α. Finally, in the third part, we compare the expected value of this upper bound to the corresponding term in the integer linear programming relaxation derived in the first part.

3.1 An Integer Linear Programming Relaxation

To obtain a good lower bound on the value of an optimum solution, we use an integer linear programming relaxation in time-indexed variables that was originally introduced by Dyer and Wolsey [6]. Although each integral feasible solution to this program corresponds to a feasible preemptive schedule, the program is a true relaxation of $1|\,r_j, pmtn\,|\sum w_jC_j$ since the objective function underestimates the value of a preemptive schedule. On the other hand, this integer linear program (*ILP*) can be solved to optimality in polynomial time such that we need not consider its LP relaxation.

The idea of the formulation is to discretize time between 0 and a fixed horizon $T+1 := \max_j r_j + \sum_j p_j$ into intervals of length 1. One introduces binary variables y_{jt} for each job j and each time interval $(t,t+1]$, $t = 0,1,\ldots,T$, where $y_{jt} = 1$ if and only if job j is being processed during the time interval $(t,t+1]$. Note that T and thus the number of y_{jt}–variables may be exponential in the input size of the scheduling problem. We get the following ILP:

$$
(ILP) \qquad
\begin{aligned}
&\text{minimize} && \sum_{j\in J} w_j C_j^{ILP} && \\
&\text{subject to} && \sum_{t=r_j}^{T} y_{jt} = p_j && \text{for all } j\in J, && (1)\\
& && \sum_{j\in J} y_{jt} \leqslant 1 && \text{for } t=0,\dots,T, && (2)\\
& && C_j^{ILP} = \frac{p_j}{2} + \frac{1}{p_j}\sum_{t=0}^{T} y_{jt}\left(t+\tfrac{1}{2}\right) && \text{for all } j\in J, && (3)\\
& && y_{jt} = 0 && \text{for all } j\in J \text{ and } t=0,\dots,r_j-1, && (4)\\
& && y_{jt} \in \{0,1\} && \text{for all } j\in J \text{ and } t=r_j,\dots,T. && (5)
\end{aligned}
$$

Equations (1) ensure that the whole processing requirement of every job is satisfied. The machine capacity constraints (2) express that the machine can process at most one job at a time. Because of equations (4), no job can be processed before its release date. The following lemma, due to Goemans [8], offers one way to understand equations (3).

Lemma 3.1. *Consider an arbitrary preemptive schedule P that is finished before time $T+1$, and assign the values to the ILP variables y_{jt} as defined above, i. e., $y_{jt}=1$ if j is being processed in the interval $(t,t+1]$, and $y_{jt}=0$ otherwise. Then,*

$$\int_0^1 C_j^P(\alpha)\, d\alpha \;=\; \frac{1}{p_j}\sum_{t=0}^{T} y_{jt}\left(t+\tfrac{1}{2}\right) \;\leqslant\; C_j^P - \frac{p_j}{2} \tag{6}$$

for each job j, and equality holds if and only if job j is never preempted from the machine.

Proof. For a fixed job j, denote by β_t, $t=0,\dots,T+1$, the fraction of j that is finished in the preemptive schedule P by time t. Since $0=\beta_0\leqslant\beta_1\leqslant\cdots\leqslant\beta_{T+1}=1$, we can write

$$\int_0^1 C_j^P(\alpha)\, d\alpha \;=\; \sum_{t=0}^{T}\int_{\beta_t}^{\beta_{t+1}} C_j^P(\alpha)\, d\alpha \;=\; \sum_{t=0}^{T}(\beta_{t+1}-\beta_t)\cdot\left(t+\tfrac{1}{2}\right) \;=\; \sum_{t=0}^{T}\frac{y_{jt}}{p_j}\cdot\left(t+\tfrac{1}{2}\right) \;,$$

which proves the equation in (6). Since $C_j^P(\alpha)\leqslant C_j^P-(1-\alpha)\cdot p_j$ for $0\leqslant\alpha\leqslant 1$, we get

$$\int_0^1 C_j^P(\alpha)\, d\alpha \;\leqslant\; C_j^P - p_j\int_0^1 (1-\alpha)\, d\alpha \;=\; C_j^P - \frac{p_j}{2} \;.$$

Equality holds if and only if $C_j^P(\alpha)=C_j^P-(1-\alpha)\cdot p_j$ for all $0<\alpha\leqslant 1$, that is, iff job j is scheduled nonpreemptively. □

As a consequence of Lemma 3.1, the value of an optimum solution to (ILP) is a lower bound on the value of an optimum preemptive schedule.

As pointed out by Dyer and Wolsey [6], it follows from the work of Posner [18] that (ILP) can be solved in $O(n\log n)$ time. Goemans [8] showed that preemptive list scheduling in order of nonincreasing ratios w_j/p_j defines an optimum solution to (ILP) if the variables y_{jt} are set as described above. This basically follows from the observation that eliminating the variables C_j^{ILP} by plugging (3) into the objective function leads to a transportation problem, which can be solved in a greedy manner.

As a result, the encoding size of the optimum solution is polynomial in the input size and it can be constructed in time $O(n\log n)$ although (ILP) itself may be exponentially large. In addition, Algorithm 1 computes an optimum solution to (ILP) in Step 2.

3.2 Preemptive α–Conversion

This subsection presents the insights in the structure of the preemptive list schedules computed in Step 2 and Step 3 of Algorithm 1 that are needed to prove Theorem 2.4.

For a fixed job j, we define J' to be the subset of J consisting of all jobs that are started before j in the preemptive list schedule P. Notice that no job $k \in J'$ is being processed between the start and the completion time of j; if k is not yet completed when j is started, then $j < k$. We denote the fraction of job $k \in J'$ that is completed by time $C_j^P(0)$ by η_k. Since we can write the starting time $C_j^P(0)$ of job j as the amount of idle time plus the time during which the machine is busy processing jobs in J' before j is started, we get

$$C_j^P(0) \;\geqslant\; \sum_{k \in J'} \eta_k \cdot p_k \ . \tag{7}$$

To analyze the completion times of jobs in the schedule computed in Step 3 of Algorithm 1 we consider schedules that are constructed by a slightly different off-line conversion routine, which we call *preemptive α–conversion.*

Preemptive α–conversion:

Consider the jobs $j \in J$ in order of nonincreasing $C_j^P(\alpha)$ and iteratively change the current preemptive schedule by applying the following steps:

i) postpone the whole processing that is done later than $C_j^P(\alpha)$ by $(1-\alpha)\cdot p_j$;

ii) remove the $(1-\alpha)$–fraction of job j that is being processed later than $C_j^P(\alpha)$ from the machine and shrink the corresponding time intervals;

iii) process the removed fraction of job j in the released time interval $\big(C_j^P(\alpha), C_j^P(\alpha) + (1-\alpha)\cdot p_j\big]$.

Note that the completion time of any job in the schedule produced by preemptive α–conversion is not smaller than its completion time obtained from preemptive list scheduling according to nondecreasing $C_j^P(\alpha)$; this follows from Lemma 2.1 since the order of completion times in the preemptive α–schedule coincides with the order of α–points in P. Figure 1 depicts a small example with 4 jobs illustrating the action of preemptive α–conversion.

Lemma 3.2. *For fixed α, the completion time C_j of job j in the schedule computed by Algorithm* 1 *can be bounded by*

$$C_j \;\leqslant\; C_j^P(\alpha) \;+\; (1-\alpha)\cdot p_j \;+ \sum_{k \in J' : \eta_k \geqslant \alpha} (1-\eta_k)\cdot p_k \ . \tag{8}$$

Proof. We prove that the right-hand side of (8) is equal to the completion time of job j in the schedule constructed by preemptive α–conversion. Notice that preemptive α–conversion does not modify the schedule P within the time interval $[0, C_j^P(\alpha)]$ before the iteration in which j is being converted. Therefore, directly after the conversion of job j its completion time in the current preemptive schedule is equal to $C_j^P(\alpha) + (1-\alpha)\cdot p_j$.

Consider a subsequent iteration corresponding to a job k with $C_k^P(\alpha) < C_j^P(\alpha)$. We distinguish two cases: If $C_k^P(\alpha) > C_j^P(0)$ then $k < j$ since job j is interrupted by k in the schedule P. In particular, k is completed before the processing of j is resumed in P. Therefore, the completion time of j in the current schedule is not affected by the sum of the postponement in Step i and the shrinking in Step ii. If $C_k^P(\alpha) \leqslant C_j^P(0)$, then $k \in J'$ and $\eta_k \geqslant \alpha$. In this case Step i and Step ii cause a delay of the completion time of job j by $(1-\eta_k)\cdot p_k$.

job j		r_j	p_j	w_j
1		6	1	2
2		4	1	1
3		2	4	2
4		0	3	1

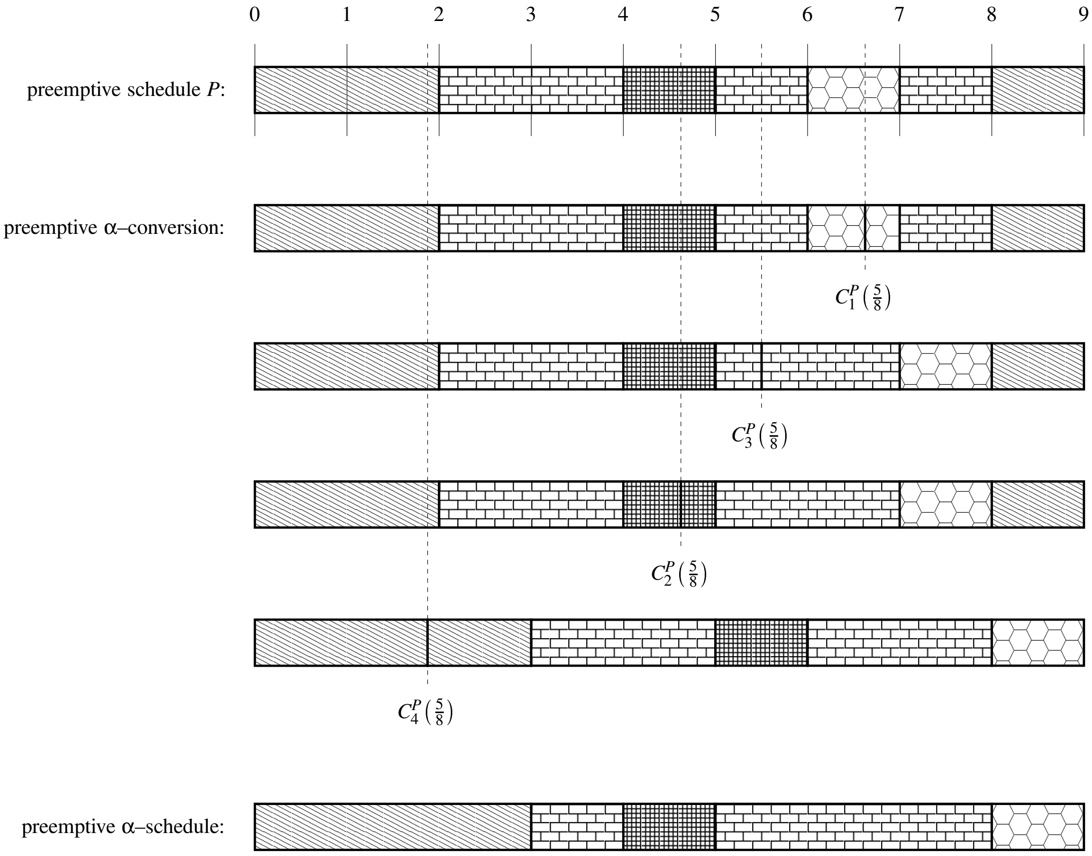

Figure 1: The conversion of the preemptive list schedule P by preemptive α–conversion and by preemptive list scheduling in order of nondecreasing α–points for $\alpha = \frac{5}{8}$.

As already mentioned before, it follows from Lemma 2.1 that the completion time of job j in the preemptive list schedule in order of nondecreasing α–points is smaller than the completion time of job j in the schedule constructed by preemptive α–conversion. $\square$

3.3 An Appropriate Probability Distribution

The key to the analysis of Algorithm 1 is to bound the expected completion time of job j in the resulting schedule by the expected value of the right-hand side of (8). We then compare the latter expected value to C_j^{ILP} in the optimum solution to (ILP), which is computed in Step 2 of Algorithm 1. The following lemma highlights the connection between the chosen probability distribution and the achieved performance guarantee.

Lemma 3.3. *Let f be a density function on $[0,1]$ and denote the expected value of a random variable that is distributed according to f by E_f, i. e., $E_f := \int_0^1 f(\alpha)\cdot\alpha\, d\alpha$. Assume that $\gamma > 0$ and*

(i) $\max_{\alpha\in[0,1]} f(\alpha) \leqslant 1+\gamma$,

(ii) $1-\mathrm{E}_f \leqslant \dfrac{1+\gamma}{2}$,

(iii) $(1-\eta)\displaystyle\int_0^{\eta} f(\alpha)\, d\alpha \leqslant \gamma\cdot\eta$ *for every* $\eta\in[0,1]$.

Moreover, let the random variable α be drawn from $[0,1]$ according to a probability distribution with density function f. Then, the expected completion time of every job $j\in J$ in the schedule constructed by Algorithm 1 is at most $(1+\gamma)\cdot C_j^{ILP}$.

The intuition underlying the three conditions (i), (ii), and (iii) on the density function f is to bound the expectations of the three corresponding terms on the right-hand side of (8) with respect to C_j^{ILP}.

Proof of Lemma 3.3. Lemma 3.2 yields

$$\mathrm{E}[C_j] \leqslant \mathrm{E}\big[C_j^P(\alpha)\big] + \mathrm{E}\big[(1-\alpha)\cdot p_j\big] + \mathrm{E}\Bigg[\sum_{k\in J':\eta_k\geqslant\alpha}(1-\eta_k)\cdot p_k\Bigg] .$$

The three terms on the right-hand side can be bounded as follows:

$$\begin{aligned}
\mathrm{E}\big[C_j^P(\alpha)\big] &= C_j^P(0) + \int_0^1 f(\alpha)\cdot\big(C_j^P(\alpha)-C_j^P(0)\big)\, d\alpha \\
&\leqslant C_j^P(0) + (1+\gamma)\int_0^1 \big(C_j^P(\alpha)-C_j^P(0)\big)\, d\alpha && \text{by (i),} \\
&= (1+\gamma)\cdot C_j^{ILP} - \gamma\cdot C_j^P(0) - (1+\gamma)\cdot\frac{p_j}{2} && \text{by Lemma 3.1;}
\end{aligned}$$

$$\mathrm{E}\big[(1-\alpha)\cdot p_j\big] = (1-\mathrm{E}_f)\cdot p_j \leqslant (1+\gamma)\cdot\frac{p_j}{2} \qquad \text{by (ii);}$$

$$\begin{aligned}
\mathrm{E}\Bigg[\sum_{k\in J':\eta_k\geqslant\alpha}(1-\eta_k)\cdot p_k\Bigg] &= \sum_{k\in J'}(1-\eta_k)\cdot p_k\int_0^{\eta_k} f(\alpha)\, d\alpha \\
&\leqslant \gamma\sum_{k\in J'}\eta_k\cdot p_k \leqslant \gamma\cdot C_j^P(0) && \text{by (iii) and (7),}
\end{aligned}$$

and the result follows. Notice that it is essential for the analysis to implicitly divide the interval $[0, C_j^P(\alpha)]$ into $[0, C_j^P(0)]$ and $(C_j^P(0), C_j^P(\alpha)]$ in order to bound $\mathrm{E}\big[C_j^P(\alpha)\big]$. This division reflects the structural insight that led to the introduction of the subset J'. □

Proof of Theorem 2.4. Observe that the density function f given in Theorem 2.4 meets the requirements (i), (ii), and (iii) of Lemma 3.3 for $\gamma = 1/3$. Thus, the competitive ratio $4/3$ follows from Lemma 3.3 by linearity of expectations. □

Since the inequalities (i) and (iii) are tight in the proof of Theorem 2.4, we can show that the density function f is optimal with respect to the analysis given in Lemma 3.3. Suppose that there exists a density function g that fulfills properties (i) and (iii) for $\gamma < 1/3$. Using (i), this leads to the following contradiction: Let $\eta = 1/2$, then

$$(1-\eta)\int_0^{\eta} g(\alpha)\,d\alpha \;=\; \tfrac{1}{2} - \tfrac{1}{2}\int_{\frac{1}{2}}^{1} g(\alpha)\,d\alpha \;>\; \tfrac{1}{2} - \tfrac{1}{2}\cdot\tfrac{1}{2}\cdot\tfrac{4}{3} \;=\; \tfrac{1}{3}\cdot\eta \;.$$

3.4 Further Results

One can also prove competitive ratio $4/3$ for the variant of Algorithm 1 where α is chosen from the interval $[0, 3/4]$ according to the probability distribution with density function $f(\alpha) = \frac{1}{3}\cdot(1-\alpha)^{-2}$. Notice that this distribution is of the form as the one used in [10] (with $\beta = 3/4$). However, the analysis is slightly more complicated in this case.

In our analysis of Algorithm 1 we have bounded the expected value of the computed schedule in terms of the lower bound given by an optimal solution to (ILP). Thus, we have derived the same bound on the quality of the integer linear programming relaxation (ILP).

Theorem 3.4. *The relaxation* (ILP) *is a* $4/3$*–relaxation, but it is not better than an* $8/7$*–relaxation for the problem* $1|\,r_j, pmtn\,|\sum w_jC_j$.

Proof. The upper bound on the quality of (ILP) follows from the analysis of Algorithm 1. To prove the negative result, consider the following instance with n jobs, where n is assumed to be even. The processing times of the first $n-1$ jobs $j = 1, \dots, n-1$ are 1, their common release date is $n/2$, and all weights are $1/n^2$. The last job has processing time $p_n = n$, weight $w_n = 1/(2n)$, and is released at time 0. This instance is constructed such that every reasonable preemptive schedule without idle time on the machine has value $2 - 3/(2n)$. However, an optimal solution to (ILP) has value $7/4 - 5/(4n)$ such that the ratio goes to $8/7$ when n increases. □

4 Concluding Remarks

It was observed by Goemans [8] that there are at most n combinatorially different values of α, i. e., over all possible choices of α one gets at most n different preemptive list schedules in order of α–points. Since each preemptive list schedule can be evaluated in $O(n)$ time, this results in an off-line deterministic $4/3$–approximation algorithm with running time $O(n^2)$ by choosing the best one from these schedules, i. e., the one with smallest weighted sum of completion times.

We close by discussing some open questions that we regard as interesting. We have shown that (ILP) is a $4/3$–relaxation of $1|\,r_j, pmtn\,|\sum w_jC_j$ and not better than an $8/7$–relaxation. What is its true quality? Interestingly, the very same integer program is known to be a 1.686–relaxation for the nonpreemptive problem $1|\,r_j\,|\sum w_jC_j$. In this context, it is not better than a 1.581–relaxation. See [9] for both results.

In the on-line setting, it seems that no good lower bound on the achievable competitive ratio of any on-line algorithm for the preemptive problem is known, neither for deterministic, nor for randomized algorithms. As for the upper bounds, we do not believe that the presented algorithms, which have competitive ratio 2 and $4/3$, respectively, will be the ultimate answer. In contrast, the corresponding nonpreemptive problem seems better understood. For the total completion time objective ($w_j = 1$ for all $j \in J$), Hoogeveen and Vestjens [14] as well as Phillips, Stein and Wein [17] gave deterministic 2–competitive algorithms; Hoogeveen and Vestjens also showed that competitive ratio 2 is best possible for deterministic on-line algorithms. In [5], Chekuri, Motwani, Natarajan, and Stein presented a randomized $e/(e-1)$–competitive algorithm; Vestjens [22] proved this is optimal against oblivious adversaries. The best known deterministic and randomized on-line algorithms for the more general total weighted completion time objective have competitive ratio 2.415 and 1.686, respectively, see [8] and [9]. They use (*ILP*) in the analysis and α–points drawn from the corresponding preemptive schedule in the algorithm, as we do. The current knowledge on the cost of the lack of complete information is the same as in the unit-weight case.

References

[1] F. Afrati, E. Bampis, C. Chekuri, D. Karger, C. Kenyon, S. Khanna, I. Milis, M. Queyranne, M. Skutella, C. Stein, and M. Sviridenko. Approximation schemes for minimizing average weighted completion time with release dates. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, 32–43, 1999.

[2] K. R. Baker. *Introduction to Sequencing and Scheduling*. John Wiley & Sons, New York, 1974.

[3] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[4] S. Chakrabarti, C. Phillips, A. S. Schulz, D. B. Shmoys, C. Stein, and J. Wein. Improved scheduling algorithms for minsum criteria. In F. Meyer auf der Heide and B. Monien, editors, *Automata, Languages and Programming*, volume 1099 of *Lecture Notes in Computer Science*, 646–657. Springer, Berlin, 1996.

[5] C. S. Chekuri, R. Motwani, B. Natarajan, and C. Stein. Approximation techniques for average completion time scheduling. *SIAM Journal on Computing*, to appear. An earlier version appeared in *Proceedings of the 8th Annual ACM–SIAM Symposium on Discrete Algorithms*, 609–618, 1997.

[6] M. E. Dyer and L. A. Wolsey. Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Applied Mathematics*, **26**, 255–270, 1990.

[7] M. X. Goemans. A supermodular relaxation for scheduling with release dates. In W. H. Cunningham, S. T. McCormick, and M. Queyranne, editors, *Integer Programming and Combinatorial Optimization*, volume 1084 of *Lecture Notes in Computer Science*, 288–300. Springer, Berlin, 1996.

[8] M. X. Goemans. Improved approximation algorithms for scheduling with release dates. In *Proceedings of the 8th Annual ACM–SIAM Symposium on Discrete Algorithms*, 591–598, 1997.

[9] M. X. Goemans, M. Queyranne, A. S. Schulz, M. Skutella, and Y. Wang. Single machine scheduling with release dates, 1999. Submitted.

[10] M. X. Goemans, J. M. Wein, and D. P. Williamson. A 1.47–approximation algorithm for a preemptive single-machine scheduling problem. *Operations Research Letters*, **26**, 149–154, 2000.

[11] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, **5**, 287–326, 1979.

[12] L. A. Hall, D. B. Shmoys, and J. Wein. Scheduling to minimize average completion time: Off–line and on–line algorithms. In *Proceedings of the 7th Annual ACM–SIAM Symposium on Discrete Algorithms*, 142–151, 1996.

[13] L. A. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein. Scheduling to minimize average completion time: Off–line and on–line approximation algorithms. *Mathematics of Operations Research*, **22**, 513–544, 1997.

[14] J. A. Hoogeveen and A. P. A. Vestjens. Optimal on–line algorithms for single-machine scheduling. In W. H. Cunningham, S. T. McCormick and M. Queyranne, editors, *Integer Programming and Combinatorial Optimization*, volume 1084 of *Lecture Notes in Computer Science*, 404–414. Springer, Berlin, 1996.

[15] W. A. Horn. Some simple scheduling algorithms. *Naval Research and Logistics Quarterly*, **21**, 177–185, 1974.

[16] J. Labetoulle, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Preemptive scheduling of uniform machines subject to release dates. In W. R. Pulleyblank, editor, *Progress in Combinatorial Optimization*, 245–261. Academic Press, New York, 1984.

[17] C. Phillips, C. Stein, and J. Wein. Minimizing average completion time in the presence of release dates. *Mathematical Programming* B, **82**, 199–223, 1998.

[18] M. E. Posner. A sequencing problem with release dates and clustered jobs. *Management Science*, **32**, 731–738, 1986.

[19] J. Sgall. On-line scheduling. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms: The State of the Art*, Chapter 9, 196–231. Springer, Berlin, 1998.

[20] M. Skutella. *Approximation and Randomization in Scheduling*. Ph.D. thesis, Technische Universität Berlin, Germany, 1998.

[21] W. E. Smith. Various optimizers for single–stage production. *Naval Research and Logistics Quarterly*, **3**, 59–66, 1956.

[22] A. P. A. Vestjens. *On-line Scheduling*. Ph.D. thesis, Eindhoven University of Technology, The Netherlands, 1997.